**source tree** : 본래 git은 명령어를 사용하여 다루지만

편리하게 사용하기 위해서 사용

((->clone을 만들어 내 컴퓨터내에 해당 파일을 받고,

add – commit – push // add – commit – push .. 를 반복))

**[Commit]**

게임의 세이브와 같이 commit시점으로 돌아갈 수 있다.

원하는 파일을 묶어 commit명령을 수행함으로 시점을 저장할 수 있음!

**[add]** – 스테이지에 올린다

commit하기 전, 저장을 원하는 파일들을 묶는 작업

(특정 부분의 코드만 업데이트했다면 그 부분만 스테이지에 올린다)

**[push]**

commit내용은 자신의 컴퓨터에 저장되므로

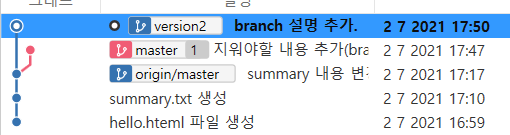
github에 push하여 업로드한다

[checkout(소스트리의 코드뭉치 버리기)]

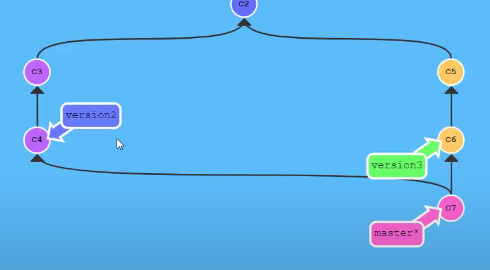
저장된 파일의 내용을 업로드 하기 전 코드뭉치 버리기를 통해 수정 전 내용으로 돌아간다.

[branch] – 특정 시점에서 수정내용을 변경하고 싶을 때

(소스트리에서는 더블클릭으로 사용)

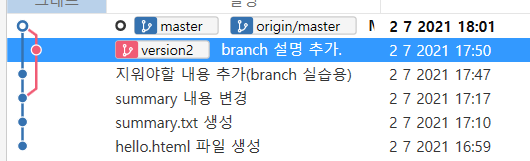


**[merge]** – 두 brach를 합치는 것(현재 작업중인 branch – head branch에 얹어서 병합)



때에 따라 충돌이 발생할 수도 있다.

합칠 때는 대게 master를 더블클릭(checkout)해서 merge



-병합 후 충돌된 파일에 들어가면 두 파일의 내용이 있는데 수동으로 관리하여 commit

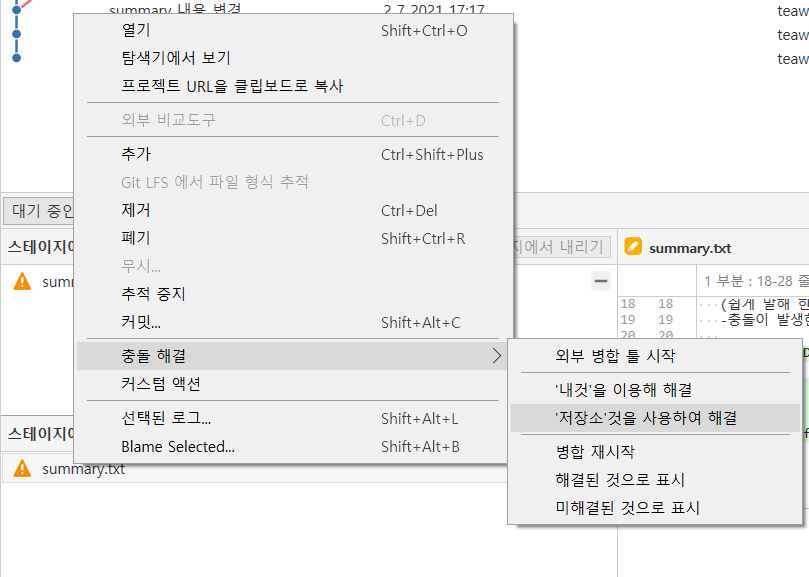
-위의 그림과 같이 branch내용이 필요가 없어졌다면 지워도 된다.

**[pull]**

지금 내 컴퓨터의 상태보다 앞선 상태가 서버에 저장되어있을 때

서버(깃허브)로부터 최신상태를 받아온다.

(대채로 여러 사람들과 함깨하면 발생)



1. 내것 – HEAD(현재 선택된 것)
2. 저장소 – 병합 할 대상

**[reset]**

3가지 모드가 있는데 hard가 완전히 되돌리는 것

특정 분기점에서 우클릭을 통해 내용을 되돌린다.

**(**특정 시점에서 우클릭 – 이 커밋까지 브랜치 초기화 – HARD 선택으로 이전상태로 돌아가기)

만약 원격저장소에 없는 내용을 reset했다면 다 날라간다..

reset의 문제점

최신 버전을 원격저장소에 push 후 , 과거 시점으로 돌아간 후 ,

새로운 내용을 commit하고 push를 하려고 하면 push와 pull모두 불빛이 들어온다. (오류발생)

* 강제 push필요
* 혹은 병합 후 (충돌문제 해결)

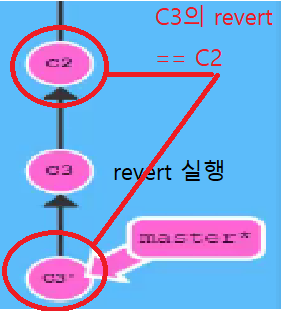
[**branch**를 이용한 되돌아가기]

-특정 시점에서 branch를 만들고 작업을 이어서 하는 것

-트리가 더러워져서 혼자 할 때 좋다.

[revert를 이용한 되돌아가기]

-소스코드) 우클릭 + 커밋되돌리기 = 변경사항 이전의 결과가 commit결과로 생긴다.



Commit 1,2,3 순서대로 되어있는 경우에서

revert로 직전이 아닌, 전전상태로 이동하려면??

-명령어 사용 : git revert HEAD HEAD~l

(HAED 가장 최근의 것을 되돌려라 // HEAD와 헤드아빠?를 순서대로 되돌려라)

-소스트리) 최신부터 순서대로 revert를 반복 적용

-commit하지 않은 변경사항이 있을 때 다른 브랜치로 checkout하려면 오류 발생

1. 임시로 commit후 다른 branch로 checkout한다. / 작업을 마치고 본래 branch로외서 이어작성

(단 commit시 마지막 commit정정을 통해 덮어쓰기 가능)

// 그냥 해도 되는데 의미없는 분기점이 생긴다. 어처피 중간에 쓰다만내용이라 다시 로드해서 쓸 일도 없고 아에 덮어쓰기해서 불필요한 분기점을 줄이는 것

//임시 commit을 서버에 push했다면 강제push를 해야 할 경우가 발생하기도 한다.

**[stash]**

-임시 저장공간

-파일의 변경사항(commit해야 할 내용)을 임시저장 후 나중에 불러오기 가능

-단 처음 작성된 파일의 경우 stach가 불가 -> 스테이지에 add하면 됨

**[rebash]**

-소스트리) 재배치

-commit 기록이 깔끔하다

-단 위험할 수 있다(이미 원격저장소에 올라갔으며 협업중이라면..)

master branch는 항상 최종결과물이 들어가 있어야 한다.

master에서 작업하는 것은 비추