Git Advanced

Git, github seminar

Microsoft Student Partner
김다영

목차

Commit 되돌리기 : git commit –amend

--amend

--amend 옵션을 지정하여 커밋을 수행하면 같은 브랜치 상에서 이전에 커밋했던 내용에 새로운 내용을 추가하거나 설명을 추가할 수 있음.

- 누락된 파일을 새로 추가하거나 기존의 파일을 업데이트 해야할 때 사용
- 이전의 커밋 내용을 변경하고 싶을 때 사용

```
✘ dayoung@gimdayeong-ui-MacBook-Pro ~/Documents/msp/2nd_git_seminar master ✚ git commit --amend -m 'edit sample.txt and add sample2.txt'
master b35a4fd] edit sample.txt and add sample2.txt
Date: Wed Aug 15 12:20:56 2018 +0900
2 files changed, 4 insertions(+), 1 deletion(-)
create mode 100644 sample2.txt
```

Commit 되돌리기 : git reset

$ git reset <옵션> <commit ID >

시계를 다시 돌리듯이 이력(커밋)을 그 당시로 돌려놓음.

<옵션>

- --hard
  - 돌아가려는 커밋 이후의 모든 작업디렉토리와 인덱스 모두 유지하지 않고 이전 커밋으로 HEAD를 되돌림
  - 작업하던 내용을 의도적으로 모두 삭제하고자 할 때만 사용

- --soft
  - 현재 인덱스 상태와 작업 디렉토리 내용을 그대로 보전한 채 커밋만 되돌릴 경우 사용.
  - add까지 진행된 상태, $git status로 확인하면 남아있다.
  - 바로 commit 될 수 있는 상태
- --mixed
  - default
  - 모든 작업디렉토리는 유지하면서 인덱스를 HEAD와 함께 돌아가고자하는 커밋으로 돌아감.

$ git reset HEAD~5
: 현재로부터 5개 이전 커밋으로 돌아감
$ git reset HEAD^
: 최신 커밋을 취소함
: 커밋은 했지만, push 는 하지 않은 경우에 유용

$ git reset --hard ORIG_HEAD
: reset 전의 커밋은 ORIC_HEAD 라는 이름으로 참조할 수 있음.
: 실수로 reset 을 한 경우에는 ORIC_HEAD 로 reset하여 reset 실행 전으로 돌아갈 수 있음

Commit 되돌리기 : git reset

$ git reset <옵션> <commit ID >

$ git reset HEAD~5
: 현재로부터 5개 이전 커밋으로 돌아감
$ git reset HEAD^
: 최신 커밋을 취소함
: 커밋은 했지만, push 는 하지 않은 경우에 유용

$ git reset --hard ORIG_HEAD
: reset 전의 커밋은 ORIC_HEAD 라는 이름으로 참조할 수 있음.
: 실수로 reset 을 한 경우에는 ORIC_HEAD 로 reset하여 reset 실행 전으로 돌아갈 수 있음

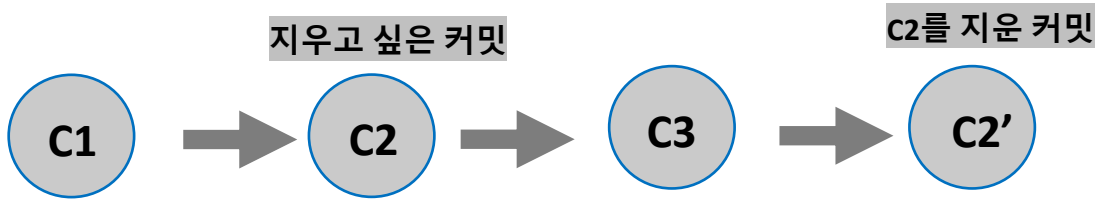
git add 한 것을 commit 전에 취소하기

$ git reset HEAD <file이름>
$ git status

Commit 되돌리기 : git revert

$ git revert <commit ID >

이전 이력은 그대로 두고, 그 되돌릴 커밋의 코드만 원복시킴.



revert VS reset

Reset : 시계를 다시 돌리듯이 이력(커밋)을 그 당시로 돌려놓음.
Revert : 이전 이력은 그대로 두고, 그 되돌릴 커밋의 코드만 복구 시키는 커밋을 하나 만듦

Reset이 이전 커밋으로 돌아갔다면, revert 는 돌이키고자 하는 커밋을 복구시키고 새로운 커밋으로 덮어씌움.

Revert는 commit 을 이미 push해서 원격 저장소에 저장해버린 경우 많이 사용

Commit 되돌리기 : git cherry-pick

$ git cherry-pick <옵션> <commit ID >

다른 브랜치에서 지정한 커밋을 복사하여 현재 브랜치로 가져올 수 있음

- 특정 브랜치에 잘못 추가한 커밋을 올바른 브랜치로 옮기고 싶을 때
- 다른 브랜치의 커밋을 현재 브랜치에도 추가하고 싶을 때

<옵션>
-n : add까지만 수행 commit은 수행하지 않음.

<3개의 커밋을 하나의 커밋으로 합쳐서 반영>
$ git cherry-pick -n asd123
$ git cherry-pick -n fghdf45
$ git cherry-pick -n tjy3545

$ git commit -m 'cherry commit'

Commit 되돌리기 : git rebase -i

$ git rebase -i <commit ID >

Tag

$ git tag <옵션> <tag name>

Commit을 참조하기 쉽도록 알기 쉬운 이름을 붙이는 것.

한 번 붙인 태그는 브랜치처럼 위치가 이동하지 않고 고정됨.

1. Lightweight tag (일반 태그)

: 이름만 붙일 수 있다.

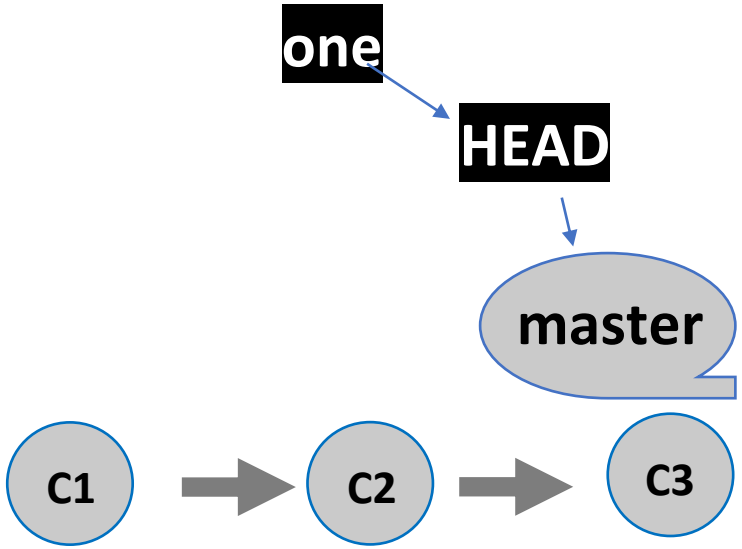
2. Annotated tag (주석 태그)

: 이름
: 태그에 대한 설명
: 서명
: 이 태그를 만든 사람의 이름, 이메일과 날짜 정보

릴리즈 브랜치에서는 보통 주석태그를 사용하여 설명이나 서명을 넣은 태그를 사용하고,

토픽 브랜치처럼 로컬에서 일시적으로 사용할 땐 이름만 붙이는 태그 사용

Tag



$ git tag <옵션> <tag name>

1. 일반 태그

$ git tag one
: 현재 HEAD 가 가리키고 있는 커밋에 one라는 태그를 붙임
$ git tag
: tag 목록 조회
$ git log --decorate
:태그 정보를 포함한 이력 조회

2. 주석태그

$ git tag -am ”git tag teat” two
: 현재 HEAD 가 가리키고 있는 커밋에 two라는 태그와 설명을 붙임

$git tag -n
: 태그 목록과 주석 내용 확인

3. 태그 삭제

$ git tag -d <tag name>

브랜치 합치기: merge

$ git merge

브랜치 합치기: rebase

분산환경에서의 Git

감사합니다.