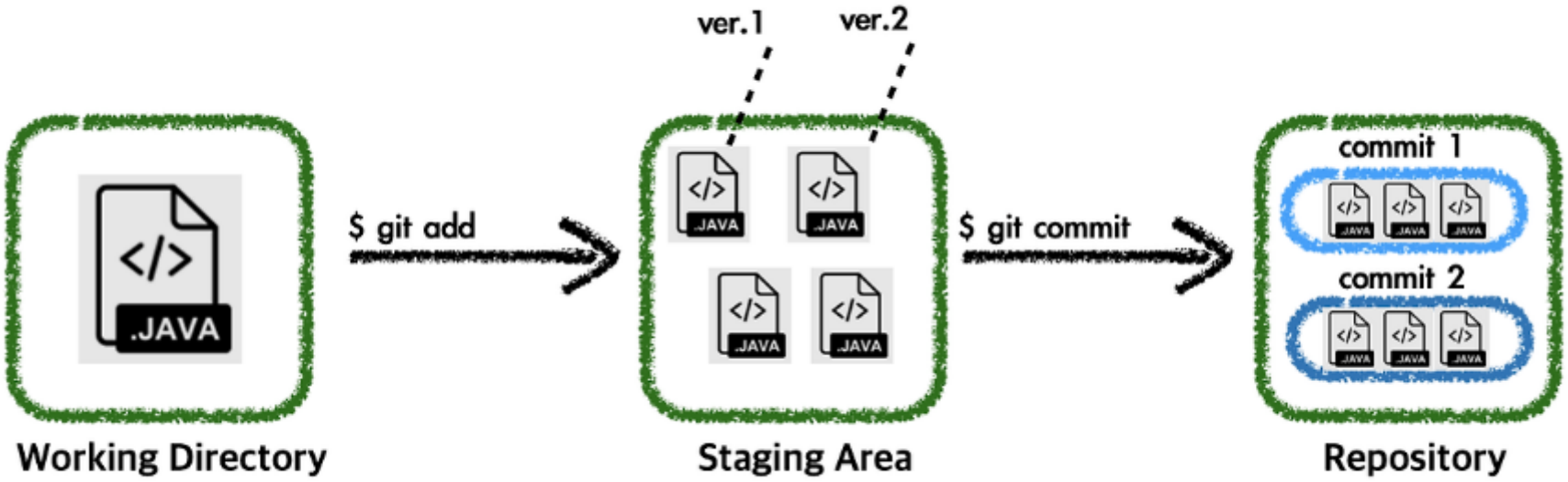
Git 초급

Git의 단계 (commit 의 이해)

- Commit : 변경 사항을 Git 저장소에 영구적으로 기록하는 작업

- Working Directory : 소스코드 작업하는 영역, 내가 작업하고 있는 프로젝트의 디렉토리
- Staging Area : 워킹 디렉토리에서 git add 명령어를 통해 추가한 파일들이 모여 있는 공간
- Repository : Staging Area에 있는 소스코드들이 commit 명령을 실행하면 최종적으로 Git 저장소에 반영 됨

Git 단계

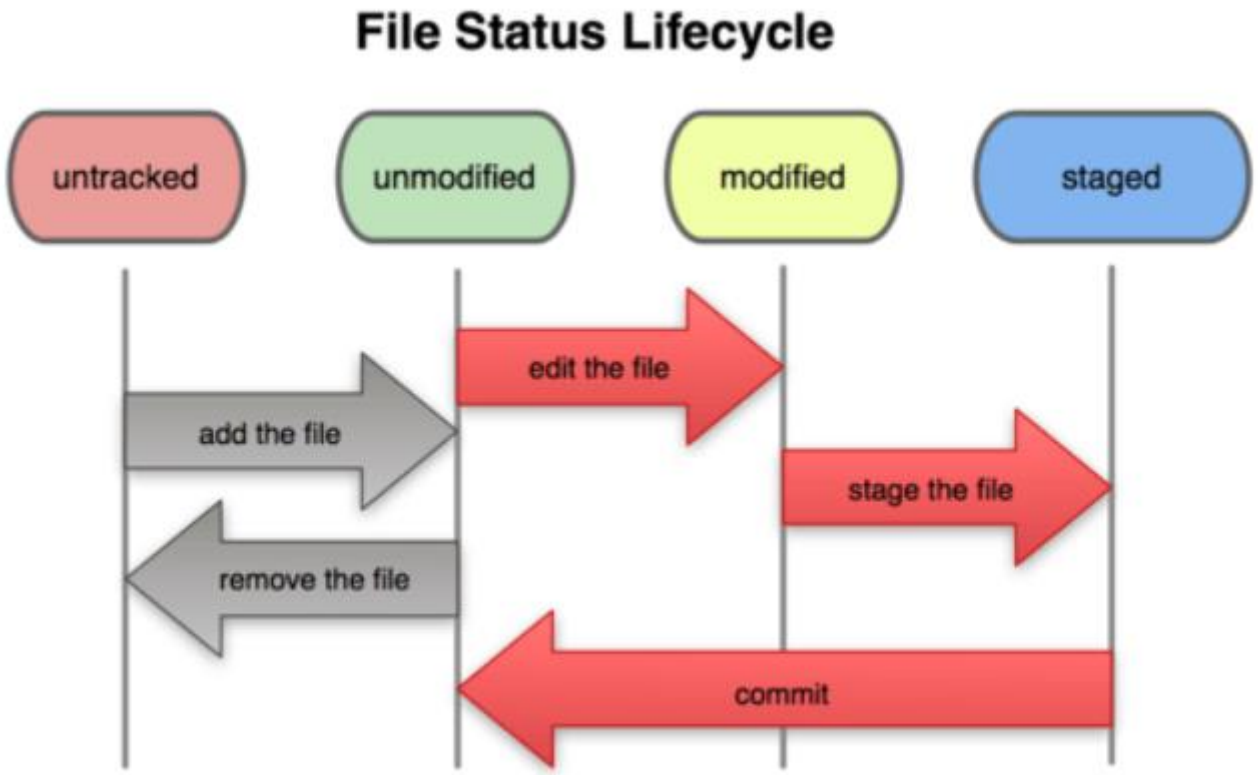


출처 : https://iseunghan.tistory.com/322

파일의 생명주기

- Untracked : Working Directory에 있는 파일이지만 Git으로 버전관리를 하지 않는 상태, 파일이 있으나 한번도 add 되지 않은 상태
- Unmodified : 신규로 파일이 추가되었을 때, new file 상태와 같음, 파일을 add하여 commit을 1회 이상 한 뒤, 수정사항 없음
- Modified : 파일이 추가된 이후 해당 파일이 수정되었을 때의 상태
- Staged : Staging Area에 반영된 상태, git add 한 상태

파일의 생명주기



출처 : https://wit.nts-corp.com/2015/03/26/3412

실습) 파일의 생명주기를 확인해 보자!

git status로 현 상태 확인

```
admin@DESKTOP-15RVNPD MINGW64 /d/git_it_b (main)
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        a.txt
        b.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Untracked 상태

실습) 파일의 생명주기를 확인해 보자!

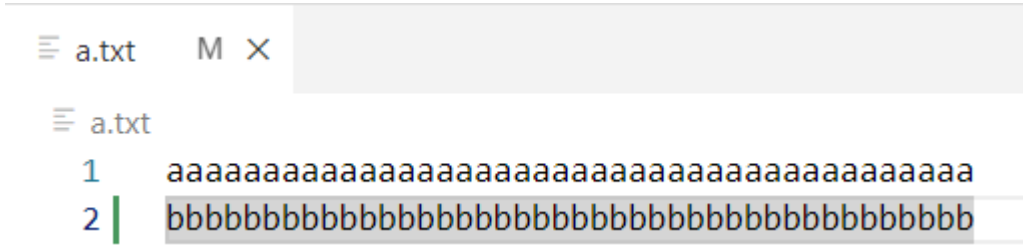
git add로 신규 파일 추가

```
admin@DESKTOP-15RVNPD MINGW64 /d/git_it_b (main)
$ git add a.txt

admin@DESKTOP-15RVNPD MINGW64 /d/git_it_b (main)
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   a.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        b.txt
```

Unmodified 상태
= new file

실습) 파일의 생명주기를 확인해 보자!

a.txt 파일을 수정하여 보자!



modified 상태

```
admin@DESKTOP-15RVNPD MINGW64 /d/git_it_b (main)
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   a.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   a.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        b.txt
```

git status로 상태 확인!

실습) 파일의 생명주기를 확인해 보자! commit!

commit : 변경 사항을 Git 저장소에 영구적으로 기록
Staging Area 에서 Repository로 이동하는 것
예) git commit 파일 -m “로그메세지 "

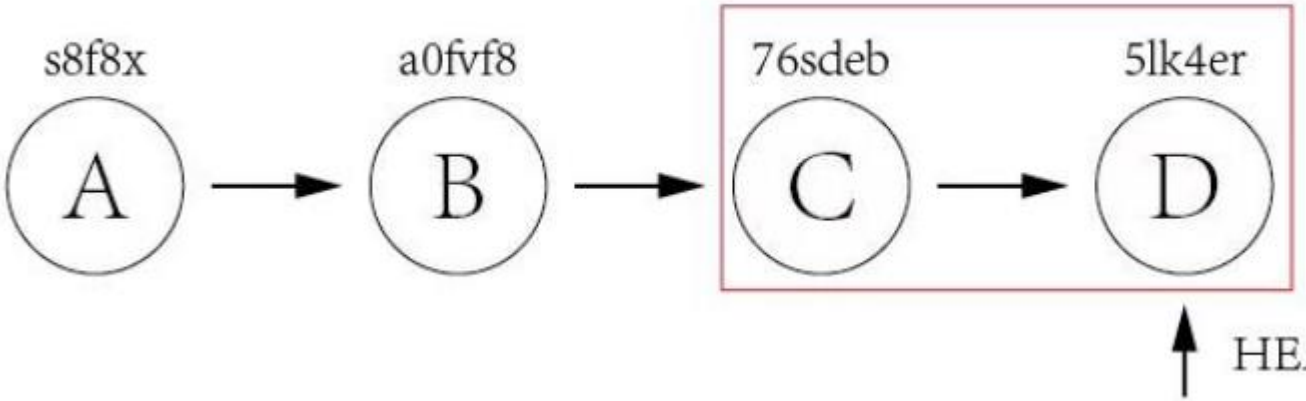
```
admin@DESKTOP-15RVNPD MINGW64 /d/git_it_b (main)
$ git commit a.txt -m "파일생명주기를 위해 a.txt추가"
[main (root-commit) 7abd456] 파일생명주기를 위해 a.txt추가
 1 file changed, 2 insertions(+)
 create mode 100644 a.txt
```

```
admin@DESKTOP-15RVNPD MINGW64 /d/git_it_b (main)
$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        b.txt

nothing added to commit but untracked files present (use "git add" to track)
```

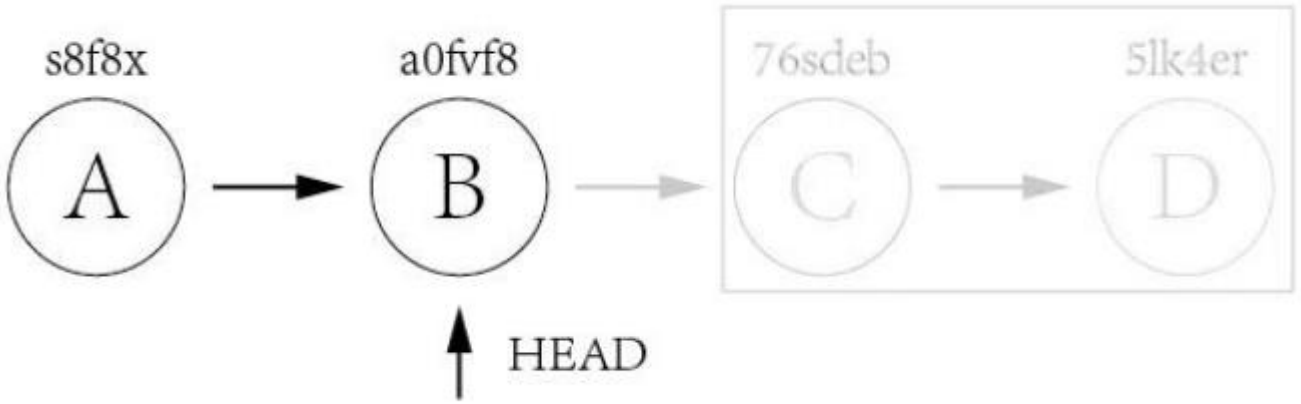
이전 버전으로 원복해야 된다면?!

reset : 원하는 시점으로 돌아간 뒤 이후 내역들 삭제.
revert : 되돌리기 원하는 시점의 커밋을 거꾸로 실행.

reset

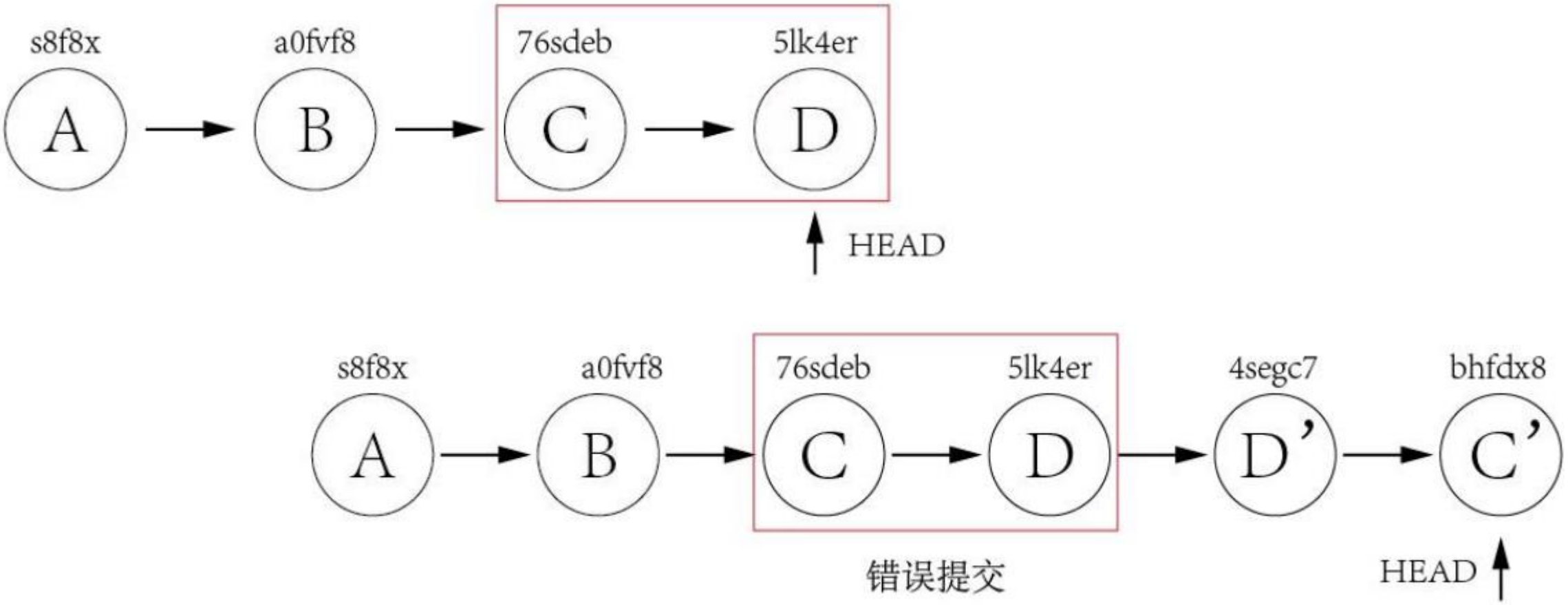


*HEAD : 현재 작업 중인 브랜치나 커밋을 가리키는 포인터



출처 : https://velog.io/@njs04210/Git-reset%EA%B3%BC-revert-%EC%95%8C%EA%B3%A0-%EC%82%AC%EC%9A%A9%ED%95%98%EA%B8%B0

revert



출처 : https://velog.io/@njs04210/Git-reset%EA%B3%BC-revert-%EC%95%8C%EA%B3%A0-%EC%82%AC%EC%9A%A9%ED%95%98%EA%B8%B0

reset 사용법

git reset --soft [commit ID]
git reset --mixed [commit ID]
git reset --hard [commit ID]

soft : commit된 파일들을 staging area로 돌려놓음. (commit 하기 전 상태로)
mixed(default) : commit된 파일들을 working directory로 돌려놓음. (add 하기 전 상태로)
hard : commit된 파일들 중 tracked 파일들을 working directory에서 삭제.

revert 사용법

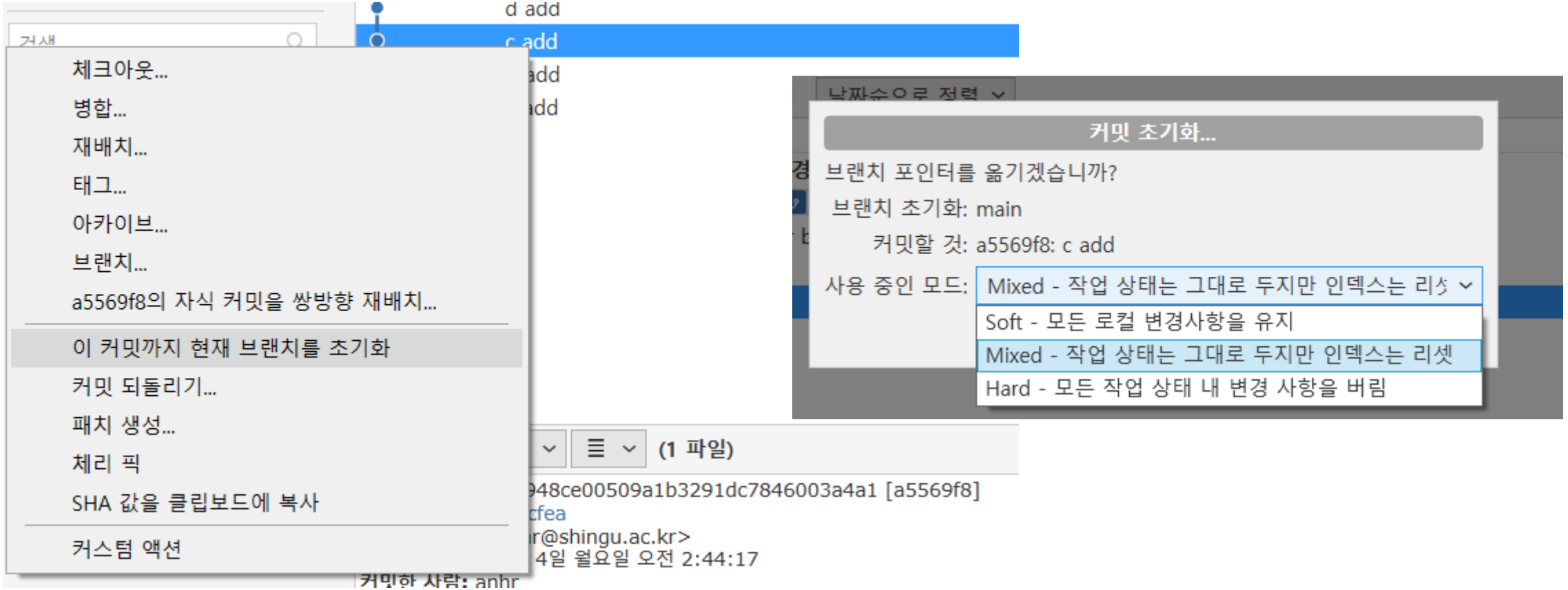
git revert [commit id]

revert를 사용하면 중간에 무슨 문제가 있었는지, 왜 돌아갔는지 등의 기록이 가능하다는 장점.
또한 다른 사람과 같은 브랜치에서 함께 작업할 때 코드 충돌을 최소화 가능

```
admin@DESKTOP-15RVNPD MINGW64 /d/git_it_b (main)
$ git revert 9b849eb5911078c2d28257d3dd6e07f2951e2531
[main a1725a2] Revert "d add"
 1 file changed, 1 insertion(+), 2 deletions(-)
```

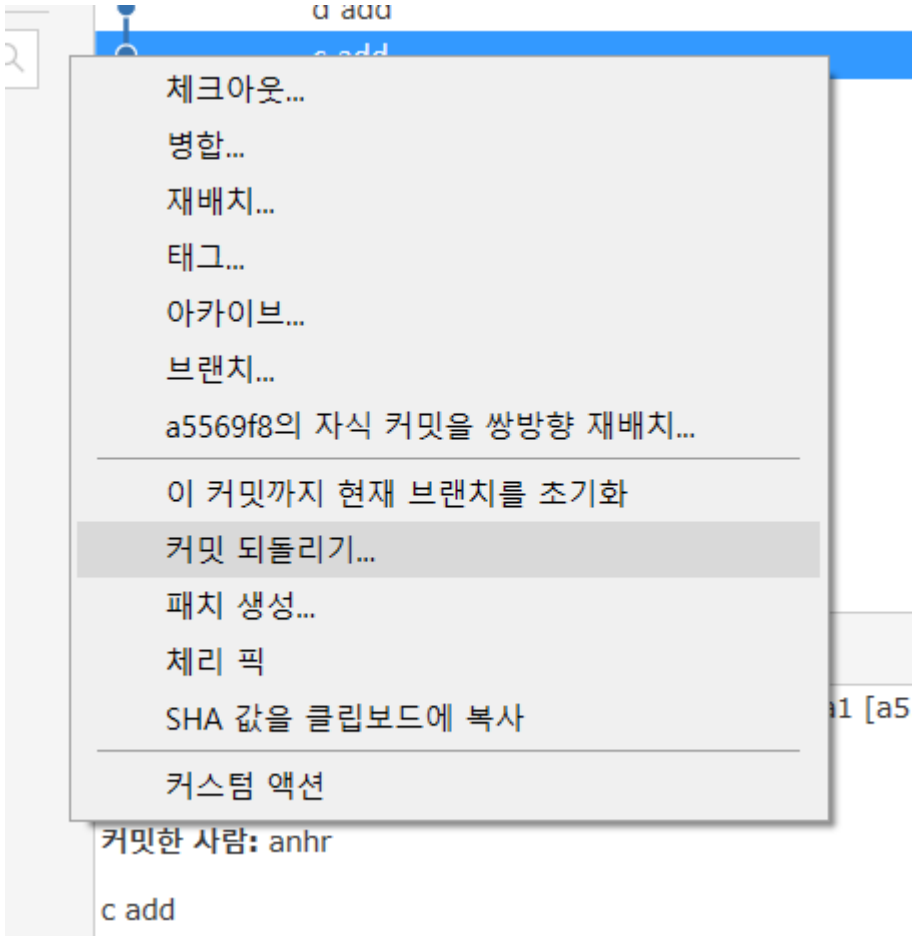
source tree (reset, revert)

reset



source tree (reset, revert)

revert



실습) reset, revert

1. a.txt파일에 aaaaaaa 쓰고 add 후 commit
2. a.txt파일에 bbbbbb 쓰고 add 후 commit
3. a.txt파일에 cccccccc 쓰고 add 후 commit
4. a.txt파일에 dddddd 쓰고 add 후 commit

5. git log로 commit 확인

```
admin@DESKTOP-15RVNPD MINGW64 /d/git_it_b (main)
$ git log
commit 9b849eb5911078c2d28257d3dd6e07f2951e2531 (HEAD -> main)
Author: anhr <anhr@shingu.ac.kr>
Date:   Mon Nov 4 02:44:31 2024 +0900

    d add

commit a5569f84c5948ce00509a1b3291dc7846003a4a1
Author: anhr <anhr@shingu.ac.kr>
Date:   Mon Nov 4 02:44:17 2024 +0900

    c add

commit ba737acfea39aa3568d92666d24b0ebeb7a95c0d
Author: anhr <anhr@shingu.ac.kr>
Date:   Mon Nov 4 02:44:04 2024 +0900

    b add

commit 66b064f29ec130494cb379bfac9ea7e7265867c2
Author: anhr <anhr@shingu.ac.kr>
Date:   Mon Nov 4 02:43:50 2024 +0900

    a add
```

실습) reset, revert

1. a.txt파일에 aaaaaaa 쓰고 add 후 commit
2. a.txt파일에 bbbbbb 쓰고 add 후 commit
3. a.txt파일에 cccccccc 쓰고 add 후 commit
4. a.txt파일에 dddddd 쓰고 add 후 commit

5. git log로 commit 확인
6. .git 폴더를 다른 경로에 백업

```
admin@DESKTOP-15RVNPD MINGW64 /d/git_it_b (main)
$ git log
commit 9b849eb5911078c2d28257d3dd6e07f2951e2531 (HEAD -> main)
Author: anhr <anhr@shingu.ac.kr>
Date:   Mon Nov 4 02:44:31 2024 +0900

    d add

commit a5569f84c5948ce00509a1b3291dc7846003a4a1
Author: anhr <anhr@shingu.ac.kr>
Date:   Mon Nov 4 02:44:17 2024 +0900

    c add

commit ba737acfea39aa3568d92666d24b0ebeb7a95c0d
Author: anhr <anhr@shingu.ac.kr>
Date:   Mon Nov 4 02:44:04 2024 +0900

    b add

commit 66b064f29ec130494cb379bfac9ea7e7265867c2
Author: anhr <anhr@shingu.ac.kr>
Date:   Mon Nov 4 02:43:50 2024 +0900

    a add
```

실습) reset, revert

git reset --soft [commit ID]
git reset --mixed [commit ID]
git reset --hard [commit ID] 실습진행

a.txt파일에 b만 있게 하라!
reset soft, mixed, hard를 진행해 보기!
revert로 진행해 보기!

실습) reset, revert

a.txt가 d까지 있게 원복

1. b.txt add 후 커밋
2. b.txt 수정 후 커밋
3. a.txt가 c까지 있게 reset, revert 실습!

실습 종료 후, 계정 초기화

Git bash에서 아래 git명령어 입력: 설정정보 초기화

git config --global --unset user.name

git config --global --unset user.email

git config --global --list 로 확인!