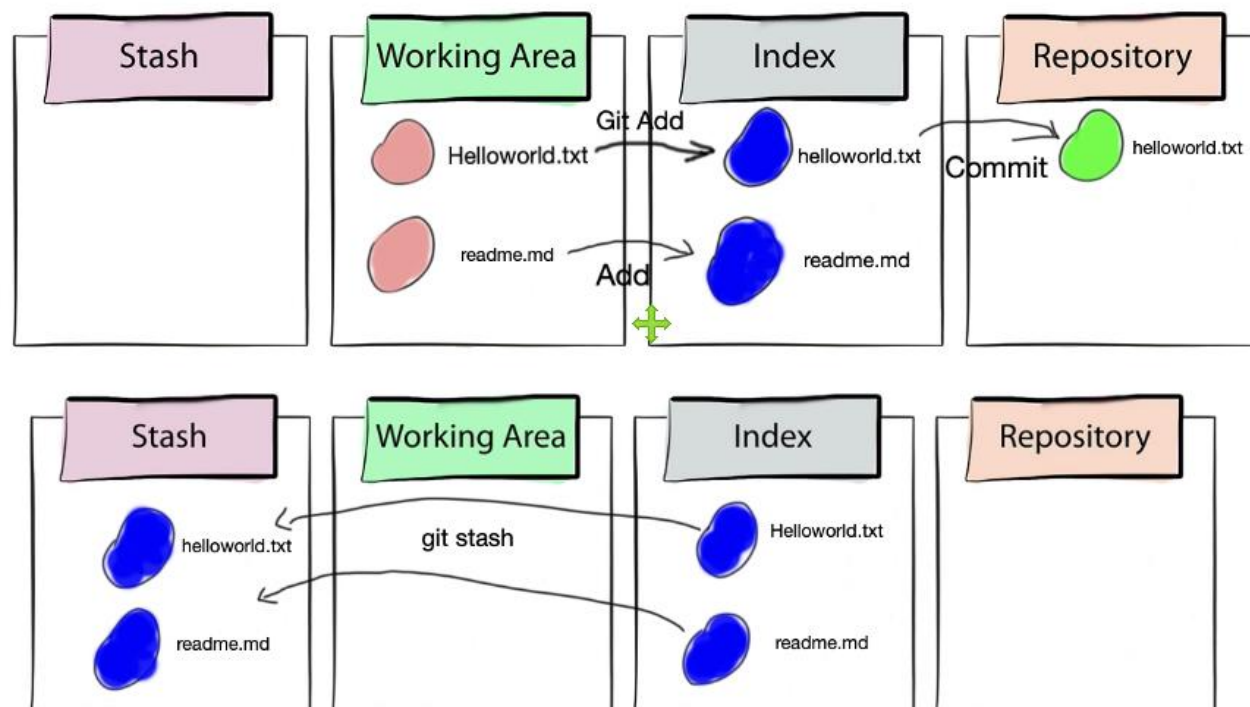


# 저장소와 작업디렉토리 확인

## 8-2

git revert: 커밋 내용 되돌리기  
git reset [option] id: 커밋id로 이동

# Git의 4가지 영역



영역	역할	위치
Working Directory	프로젝트 디렉토리이며, 개발자가 직접 코드를 수정하는 공간을 의미합니다.	.
Index (Staging Area)	Working Directory 에서 Repository로 정보가 저장되기 전 준비 영역입니다.	.git/index
Repository	파일이나 폴더를 변경 이력별로 저장해두는 곳입니다.	.git
stash	임시적으로 작업사항을 저장해두고, 나중에 꺼내올 수 있는 영역입니다.	.git/refs/stash

# Git의 4가지 영역

- **1. Working Directory(작업영역)**
  - 프로젝트 디렉토리이며, 개발자가 직접 코드를 수정하는 공간을 의미
  - .git을 제외한 모든 영역에 해당
- **2. Index (Staging Area)**
  - Working Directory 에서 Repository로 정보가 저장되기 전 준비 영역
  - .git/index 파일로 관리
- **3. Repository(저장소)**
  - 파일이나 폴더를 변경 이력 별로 저장해두는 영역
  - .git 디렉토리 내에 존재
  - Local, Remote Repository로 구분
- **4. Stash(임시영역)**
  - 임시적으로 작업 사항을 저장해두고, 나중에 꺼내올 수 있는 영역

# \$ git revert <commitid>

- 현재 HEAD를 특정 시점 commit 이전 상태로 변경
  - <commitid> 바로 이전으로 이동
    - commit을 추가로 수행
    - 과거의 모든 커밋은 그대로 유지 관리되며
    - 새로운 커밋을 추가해 특정 시점으로 이동

# \$ git revert <commit> 실습

- 백로그닷컴

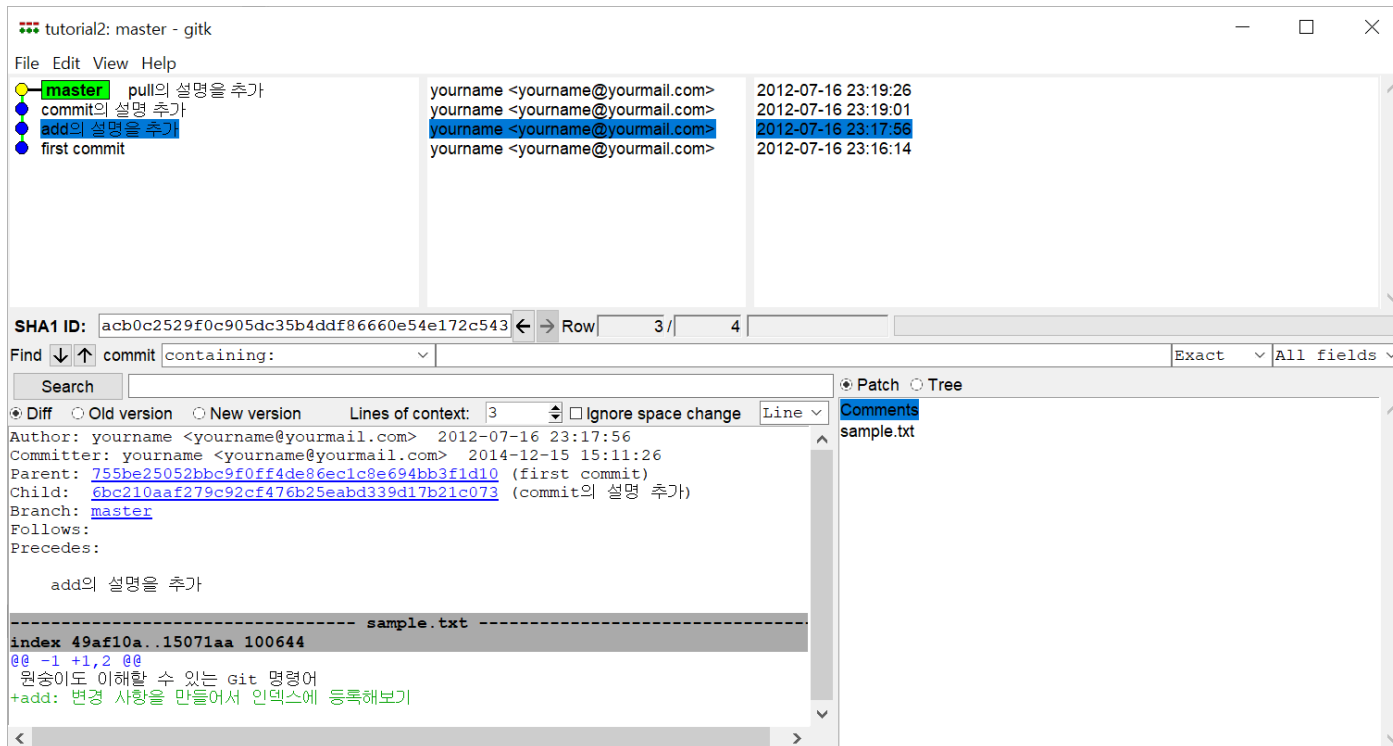
- [https://backlog.com/git-tutorial/kr/stepup/stepup7\\_2.html](https://backlog.com/git-tutorial/kr/stepup/stepup7_2.html)

- revert 를 사용

- 마지막 커밋한 「pull의 설명을 추가」를 지워보도록 하겠습니다.
- 다운로드 한 'stepup-tutorial/tutorial2' 폴더로 이동
- 폴더에서 git gui, git bash 모두 실행

# Git gui로 현황 파악

- 압축을 푼 폴더에서 git bash 실행
  - 'stepup-tutorial/tutorial2'
- Git GUI 한글 처리
  - \$ git config --global gui.encoding utf-8
- GUI에서 4개의 커밋 확인



# Bash에서도 확인

- \$git log --oneline

```
$ git log --oneline
09d4473 (HEAD -> master) pull의 설명을 추가
6bc210a commit의 설명 추가
acb0c25 add의 설명을 추가
755be25 first commit
```



- \$ git status

# Revert 실행과 확인

- **revert 를 사용**

- 마지막 커밋한 「pull의 설명을 추가」를 지워보자
- \$ git revert HEAD
- \$ git log
- \$ git status
  - 초기화된 상태

- **파일을 열어 보면**

- 파일에서 「pull의 설명을 추가」를 지워짐
  - 지정한 커밋인 head를 취소하는 개념이라 그 이전 단계로 이동됨



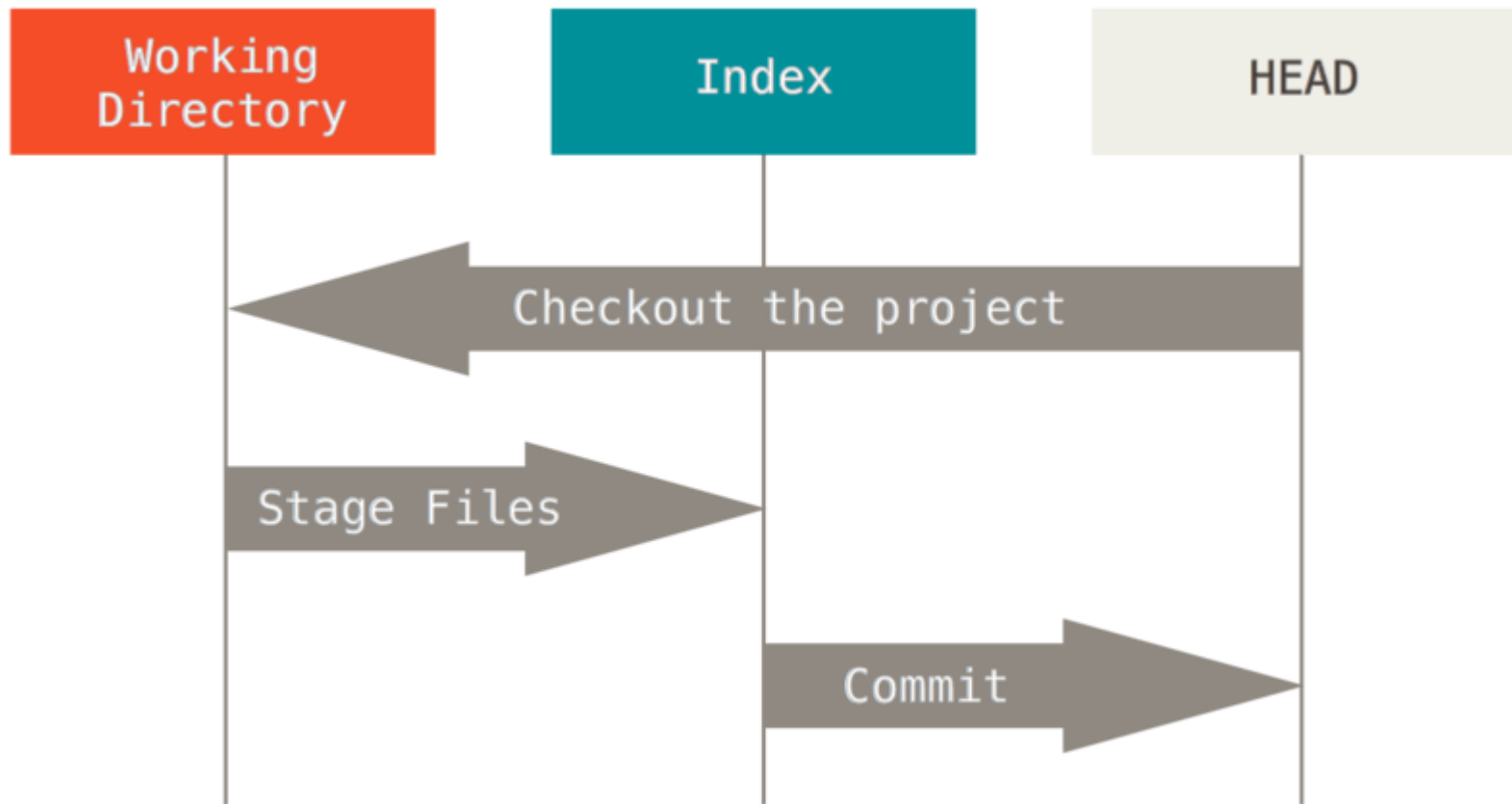


# 저장소와 작업디렉토리 확인

## 8-3

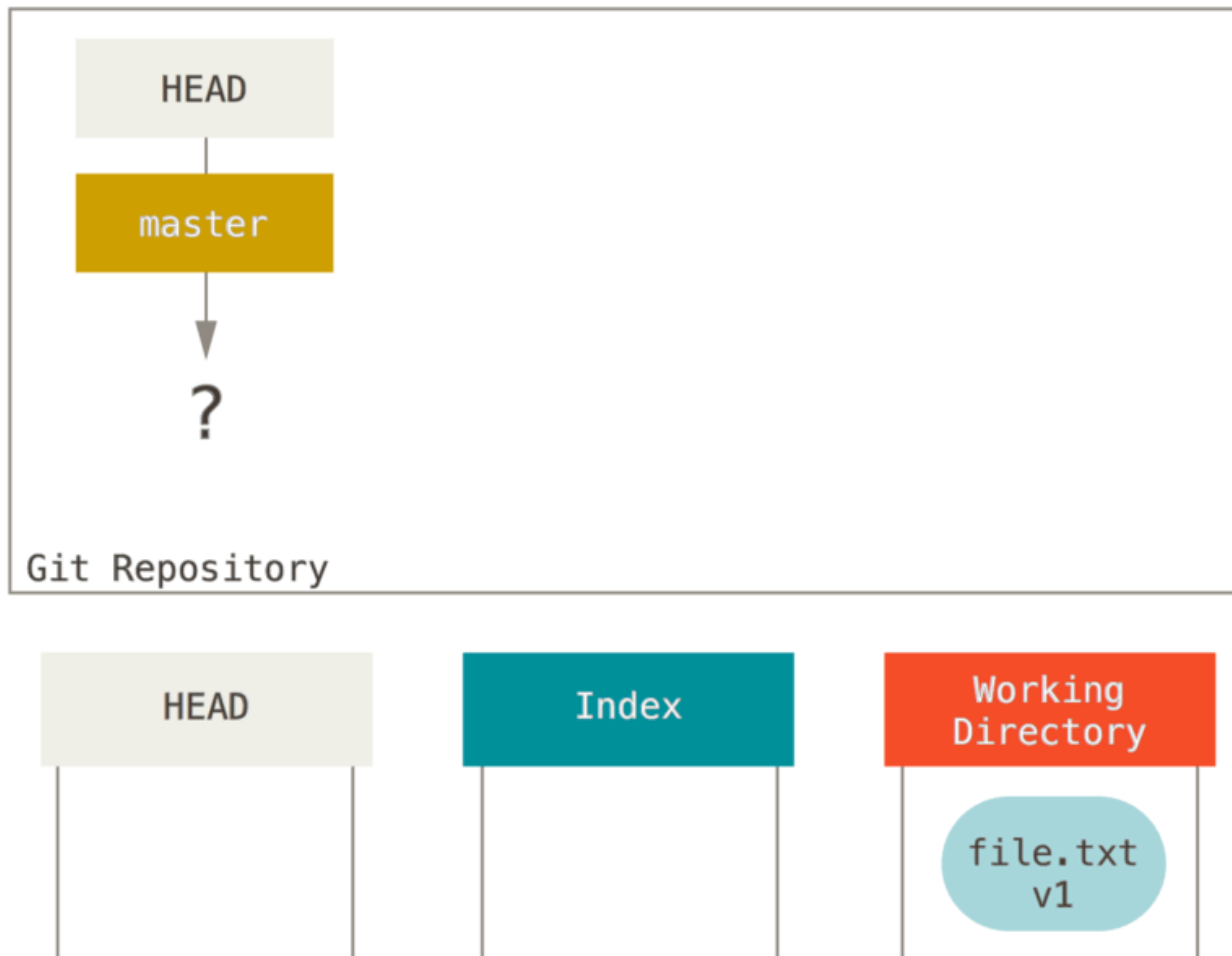
git revert: 커밋 내용 되돌리기  
git reset [option] id: 커밋id로 이동

### 3 상태



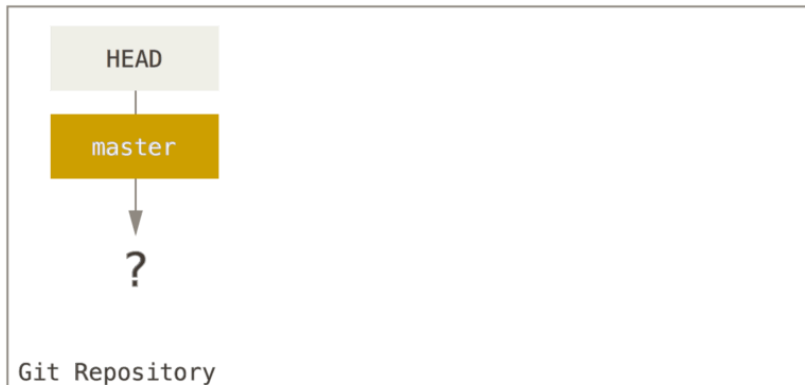
# 상태 현황 1

- 작업 디렉토리에 파일 저장

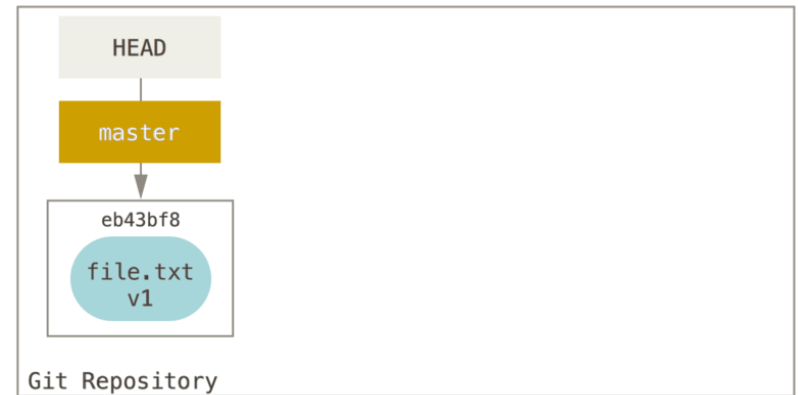


## 상태 현황 2, 3

- \$ git add
- \$ git commit



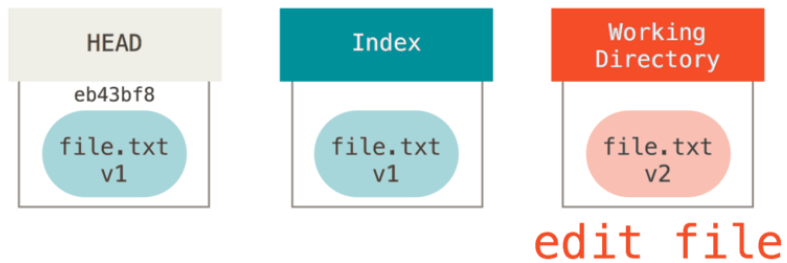
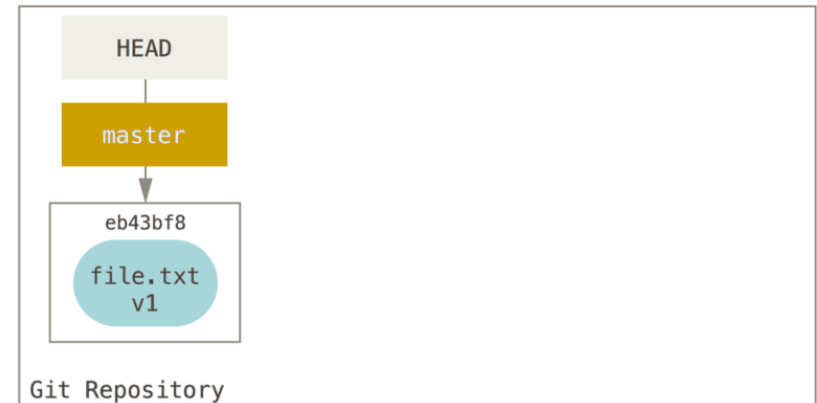
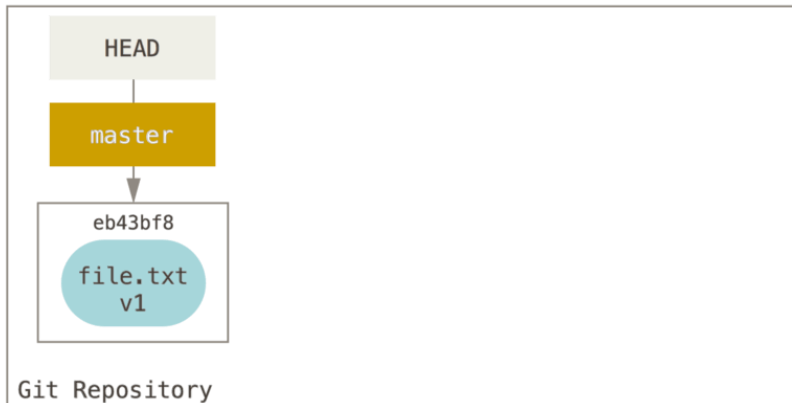
git add



git commit

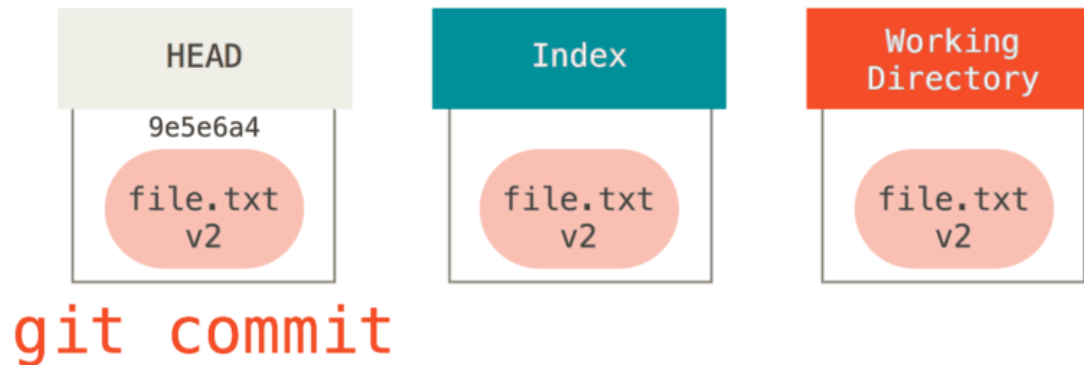
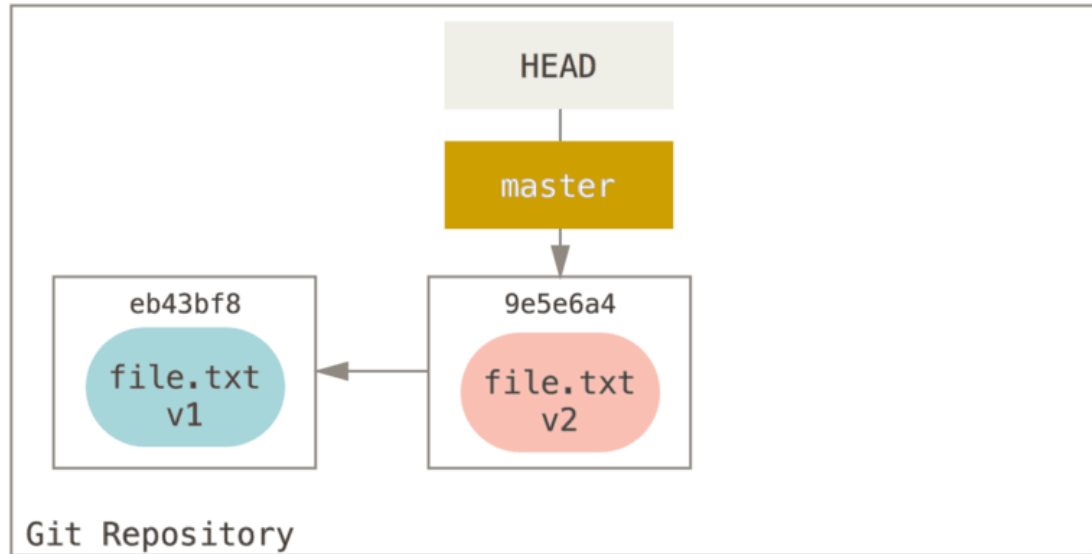
# 상태 현황 4, 5

- 파일 수정
- \$ git add

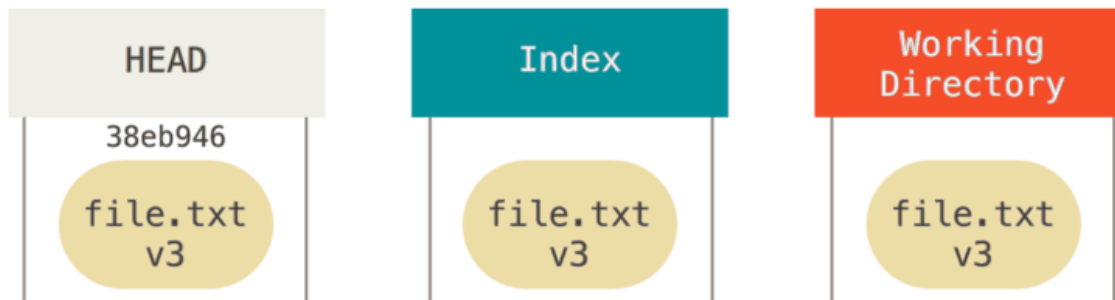
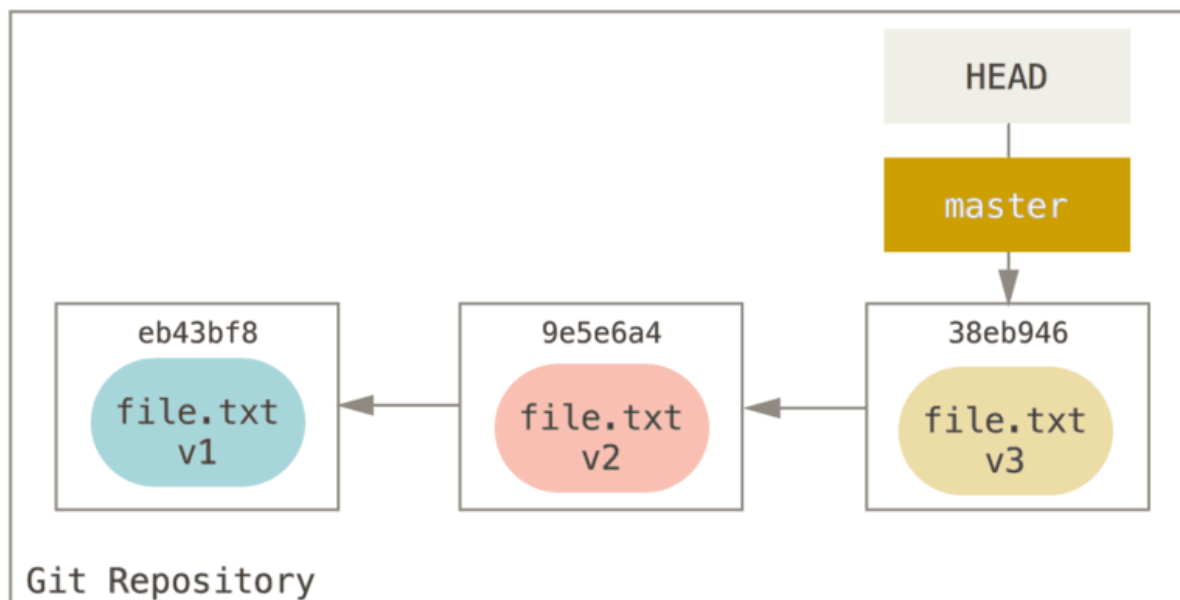


## 상태 6

- \$ git commit



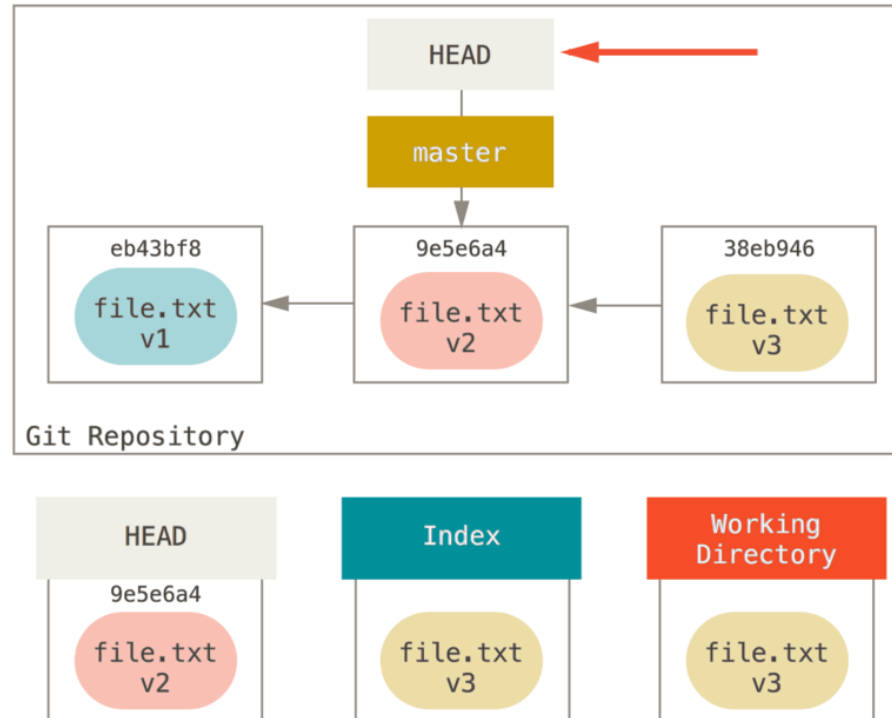
# 커밋 3회 상태



# 헤드만 이동시키는 연산 `reset --soft`

- 옵션 `--soft`

- `$ git reset --soft head~`
- 브랜치가 가리키는 커밋만 이전으로 되돌림
  - 지정한 커밋(HEAD의 이전 커밋)을 가리키도록 업데이트
  - `reset` 명령 뒤에 `HEAD~` (HEAD의 부모 커밋)
- Index나 워킹 디렉토리는 그대로 놔두고(수정하지 않음)



`git reset --soft HEAD~`



- 백로그 튜토리얼 실습
  - stepup-tutorial/tutorial3

```
MINGW64/c/2021 backlog git tutorial/stepup-tutorial/tutorial3
$ cat sample.txt
원숭이도 이해할 수 있는 Git 명령어
add: 변경 사항을 만들어서 인덱스에 등록해보기
commit: 인덱스의 상태를 기록하기
pull: 원격 저장소의 내용을 가져오기

PC@LAPTOP-QK7LS4IS MINGW64 /c/2021 backlog git tutorial/stepup-tutorial/tutorial3 (master)
$ git reset --soft orig_head

PC@LAPTOP-QK7LS4IS MINGW64 /c/2021 backlog git tutorial/stepup-tutorial/tutorial3 (master)
$ git log --oneline
09d4473 (HEAD -> master) pull의 설명을 추가
6bc210a commit의 설명 추가
acb0c25 add의 설명을 추가
755be25 first commit

PC@LAPTOP-QK7LS4IS MINGW64 /c/2021 backlog git tutorial/stepup-tutorial/tutorial3 (master)
$ git st
On branch master
nothing to commit, working tree clean

PC@LAPTOP-QK7LS4IS MINGW64 /c/2021 backlog git tutorial/stepup-tutorial/tutorial3 (master)
$ git reset --soft 6bc210a

PC@LAPTOP-QK7LS4IS MINGW64 /c/2021 backlog git tutorial/stepup-tutorial/tutorial3 (master)
$ git log --oneline
6bc210a (HEAD -> master) commit의 설명 추가
acb0c25 add의 설명을 추가
755be25 first commit

PC@LAPTOP-QK7LS4IS MINGW64 /c/2021 backlog git tutorial/stepup-tutorial/tutorial3 (master)
$ cat sample.txt
원숭이도 이해할 수 있는 Git 명령어
add: 변경 사항을 만들어서 인덱스에 등록해보기
commit: 인덱스의 상태를 기록하기
pull: 원격 저장소의 내용을 가져오기

PC@LAPTOP-QK7LS4IS MINGW64 /c/2021 backlog git tutorial/stepup-tutorial/tutorial3 (master)
```

# 저장소와 작업디렉토리 확인

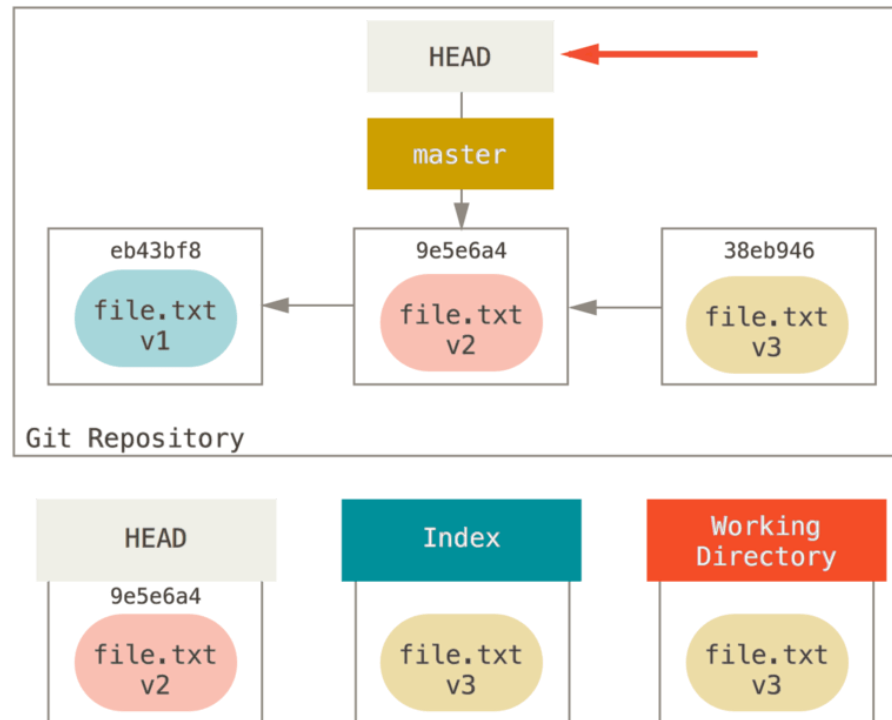
## 8-4

git revert: 커밋 내용 되돌리기  
git reset [option] id: 커밋id로 이동

# 헤드만 이동시키는 연산 reset --soft

- 옵션 --soft

- \$ git reset --soft head~
- 브랜치가 가리키는 커밋만 이전으로 되돌림
  - 지정한 커밋(HEAD의 이전 커밋)을 가리키도록 업데이트
  - reset 명령 뒤에 HEAD~ (HEAD의 부모 커밋)
- Index나 워킹 디렉토리는 그대로 놔두고(수정하지 않음)

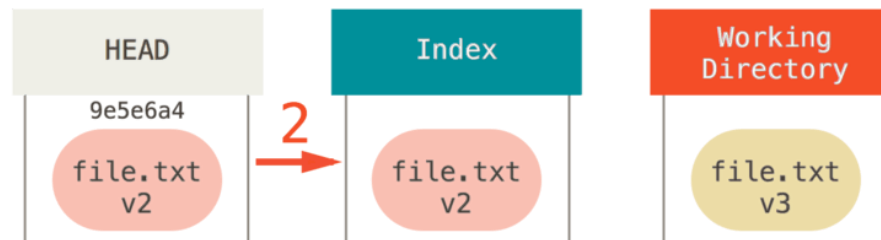
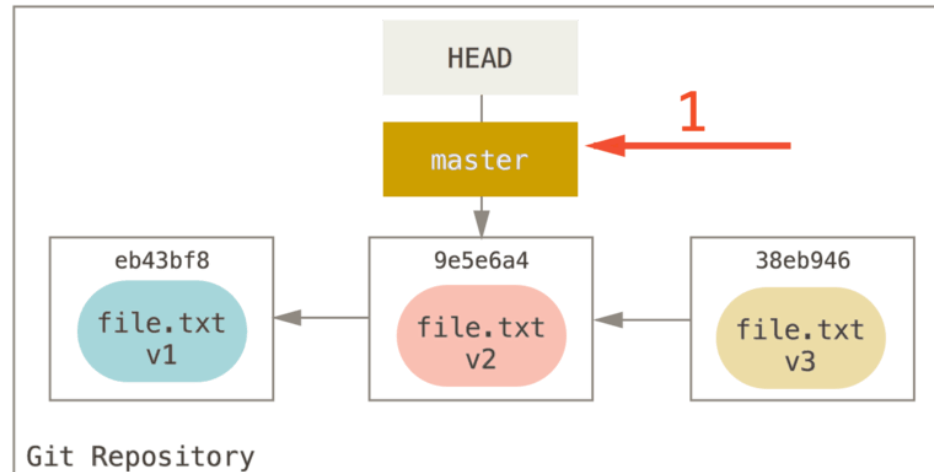


`git reset --soft HEAD~`

# 헤드 이동시키고 index도 수정하는 연산 reset --mixed

- 옵션 --mixed, 옵션이 없는 것과 같음

- \$ git reset --mixed head~
- 브랜치가 가리키는 커밋은 이전으로 되돌림
- Index를 수정된 HEAD가 가리키는 스냅샷으로 수정, 워킹 디렉토리는 그대로 놔두고
  - Index를 수정된 커밋 상태로 초기화

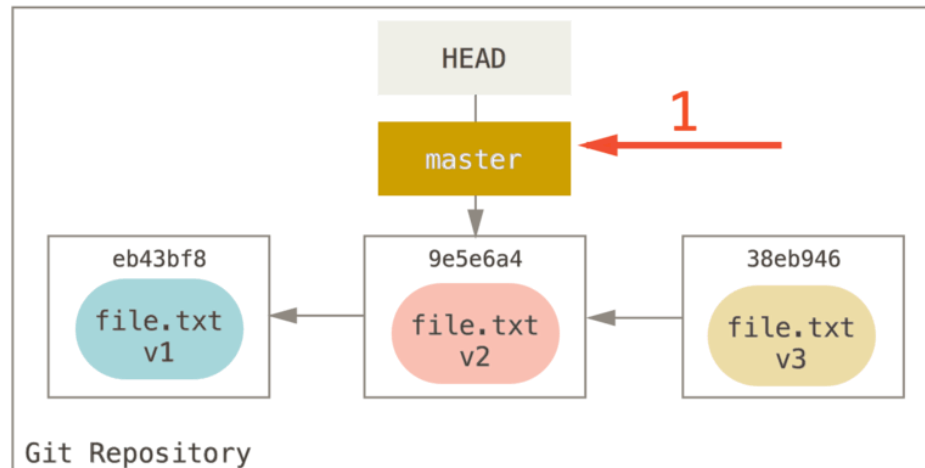


git reset [--mixed] HEAD~

# 헤드 이동, index와 작업 폴더도 수정시키는 연산 `reset --hard`

- 옵션 `--hard`

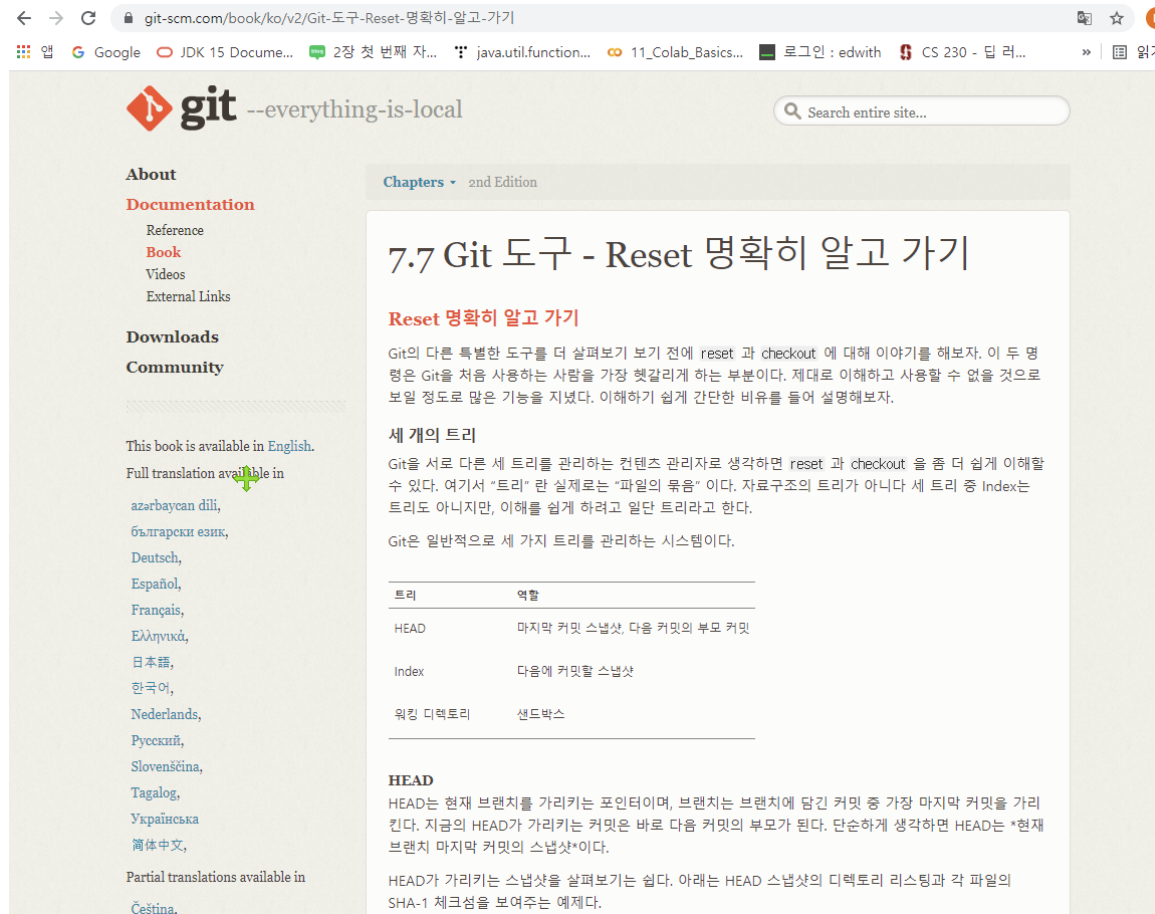
- `$ git reset --hard head~`
- 브랜치가 가리키는 커밋은 이전으로 되돌림
- Index를 수정된 HEAD가 가리키는 스냅샷으로 수정, 워킹 디렉토리도 마찬가지로 수정
  - Index와 워킹 디렉토리를 수정된 커밋 상태로 초기화



`git reset --hard HEAD~`

# 깃 홈 페이지 문서

- <https://git-scm.com/book/ko/v2/Git-%EB%8F%84%EA%B5%AC-Reset-%EB%AA%85%ED%99%95%ED%9E%88-%EC%95%8C%EA%B3%A0-%EA%B0%80%EA%B8%B0>




The screenshot shows the Git website's Korean version. The main heading is '7.7 Git 도구 - Reset 명확히 알고 가기'. Below it, there's a section titled 'Reset 명확히 알고 가기' which explains the purpose of the reset command. A table lists three types of branches and their roles:

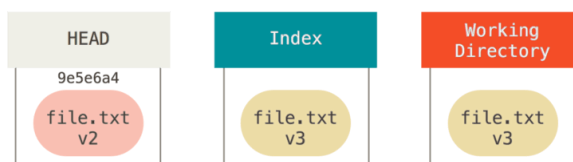
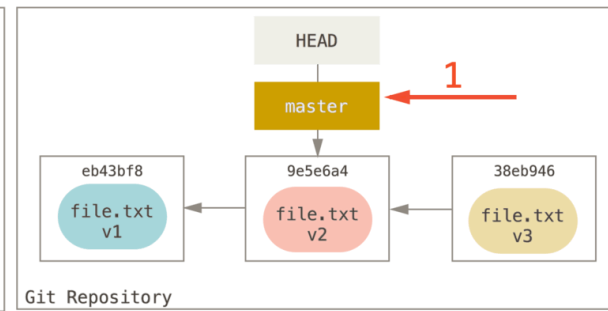
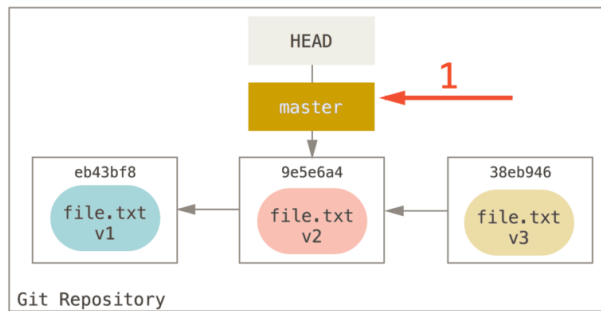
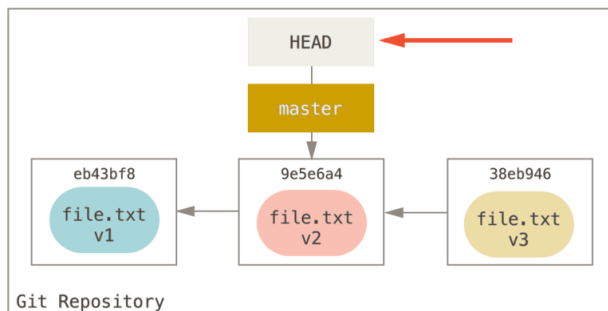
트리	역할
HEAD	마지막 커밋 스냅샷, 다음 커밋의 부모 커밋
Index	다음에 커밋할 스냅샷
워킹 디렉토리	샌드박스

Below the table, there's a section titled 'HEAD' which explains that HEAD is a pointer to the current branch, and it's updated to point to the next commit's parent. It also mentions that HEAD is a snapshot of the current branch's state.

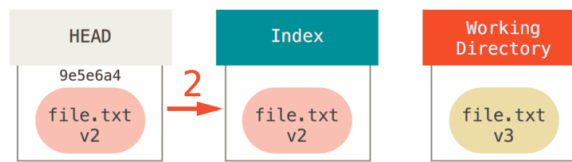
# \$ git reset 복습

- <https://juyoungit.tistory.com/391>

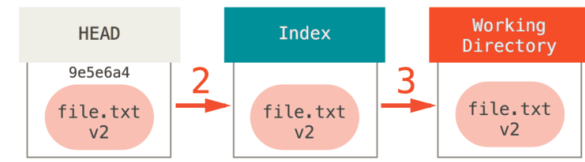
-- hard	해당 commit ID의 상태로 Working directory와 staging area 모두 초기화 된다 (Working directory의 내용까지 모두 변경되므로 이에 대한 주의가 필요하다.)
-- mixed	해당 commit ID의 상태로 staging area는 초기화 되고 Working directory는 변경되지 않는다. 옵션을 별도로 입력하지 않는 경우 이 --mixed를 기본값으로 사용한다. 
-- soft	해당 commit ID의 상태로 Working directory와 staging area 모두 변경되지 않는다.



git reset --soft HEAD~



git reset [--mixed] HEAD~



git reset --hard HEAD~

# \$ git reset

- 과거 커밋 지점으로 이동하고, 이동된 이후의 커밋은 삭제하는 명령어
- git reset에는 3가지 종류
- **2.1 git reset –hard**
  - 해당 커밋ID의 상태로 이동하고
  - Working Directory와 Index영역 모두 초기화합니다.
- **2.2 git reset –mixed**
  - 해당 커밋ID의 상태로 이동하고,
  - Index영역은 초기화되고
  - Working Directory는 변경되지 않습니다.
- **2.3 git reset –soft**
  - 해당 커밋ID의 상태로 이동하고,
  - Index영역과 Working Directory 모두 변경되지 않고