10~12

커밋 간의 파일 비교

최신 커밋 : HEAD

하나 전 커밋 : HEAD~ 또는 HEAD^

- ' ~ ' → tilde,틸드,물결
- ' ^ ' → Caret,커렛,삿갓,모자

두개 이전 커밋 : HEAD~~ 또는 HEAD~2 또는 HEAD^^ 또는 HEAD^~

깃 3 영역

탐색기 저장소 폴더

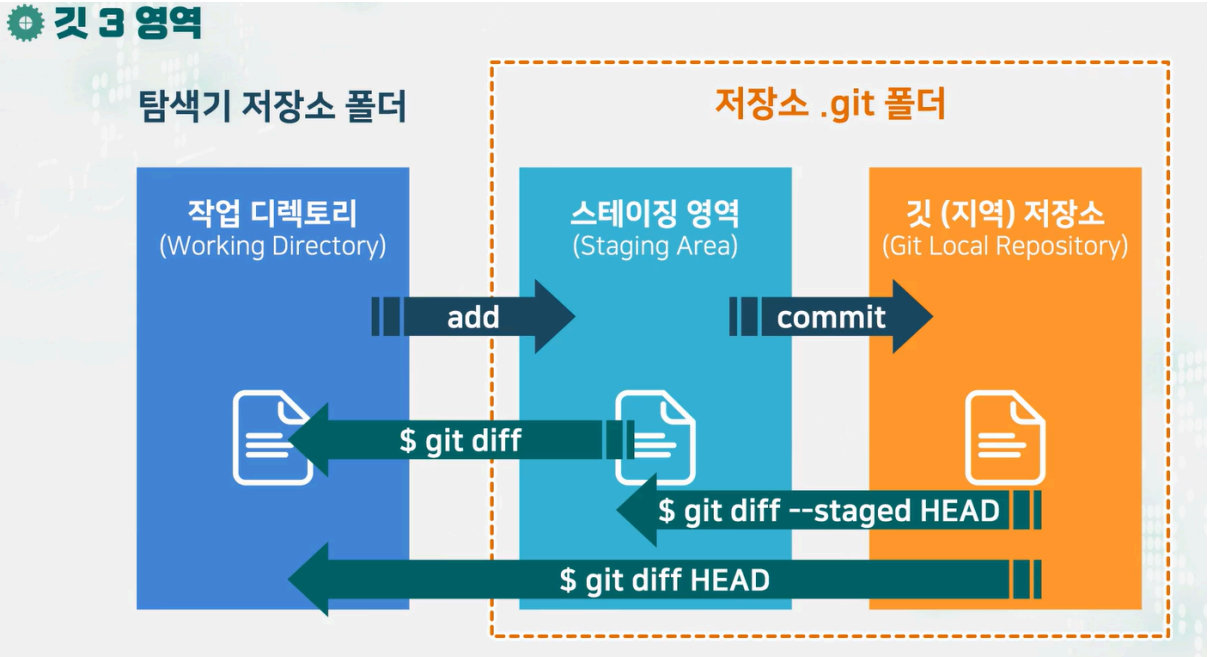
- 작업 디렉토리 : 리눅스 명령어 ls로 보이는 파일

저장소 .git 폴더

- 스테이징 영역 : 깃 명령 git ls-files로 보이는 파일
- 깃(지역) 저장소

[영상보면서 이해 필요]

(사진을 기억하기)



스테이징 영역 기준으로 작업 디렉토리 파일 비교

$ git diff : 작업 디렉토리와 스테이징 영역 비교(add 전 상태)

깃 저장소 기준으로 스테이징 영역 파일 비교

$ git diff - -staged 또는 $ git diff - -cached : 스테이징 영역과 마지막 커밋 비교 (양쪽이 같으면 결과 표시되지 않음)

깃 저장소 기준으로 작업 디렉토리 파일 비교

$ git diff HEAD : 작업 디렉토리 전체와 마지막 커밋 비교

$ git diff HEAD~

커밋 간의 파일 비교

$ git diff [비교할commit해쉬1][비교할commit해쉬2]

리눅스 명령 파일 삭제

$ rm [file] : 작업 디렉토리에서 file 삭제

깃 명령 파일 삭제

$ git rm [file] : 작업 디렉토리와 스테이징 영역에서 모두 file 삭제 —> 다음 커밋에서 지정한 file을 삭제하겠다는 의미

$ git rm - -cached [file] : 스테이징 영역에서 file 삭제 —> 작업 디렉토리에서는 삭제되지 않음 / $ git ls -files 결과에서 보이지 않음. 기본적으로 스테이징 영역의 파일 목록을 표시 —> Tracked 상태의 파일을 제거하여 Untracked 상태로 만듦

$ git commit m 'Delete g' : 파일 g가 삭제된 상태에서 커밋

파일 복원 restore

$ git restore [file] : 작업 디렉토리의 파일을 스테이징 영역의 파일 상태로 복구 / 작업 디렉토리에 있던 내용이 사라지고, 스테이징 영역과 같은 상태가 됨(작업 디렉토리 == *스테이징 영역*)

$ git restore - -staged [file] : 깃 저장소의 최신 커밋 상태의 파일을 스테이징 영역에 복구 / 스테이징 영역에 있던 내용이 사라지고, 깃 저장소 영역과 같은 상태가 됨(스테이징 영역 == *깃 저장소 영역*)

$ git restore - -source=HEAD - -staged [file]

$ git restore - -source=HEAD - -staged - -worktree [file] : 깃 저장소의 최신 커밋 상태의 파일을 작업 디렉토리와 스테이징 영역에 한번에 복구 / 셋 다 같은 상태가 됨(작업 디렉토리 == 스테이징 영역 == *깃 저장소*)

$ git restore - -source=HEAD [file] : 깃 저장소의 최신 커밋 상태의 파일을 작업 디렉토리에 복원 / 작업 디렉토리 == *깃 저장소*)

{12} - 내용정리 및 실습. 이해를 위해서 시청

계속 사용하는 깃 명령을 짧게 다른 이름으로 만드는 방법

$ git config - -global alias.별칭이름 '원래 명령어 - -긴옵션 -짧은옵션' : 설정 이후에는 다음 명령으로 사능 → $ git 별칭이름

'- -global'은 모든 프로젝트에서 공통적으로 사용하고자 하는 설정, 사용자의 홈디렉토리의 .gitconfig파일에 추가

[alials]

ss = status -s

s = status

co = checkout

br = branch

c = commit