OSS 1-3 ~ 4-3

Git 정의 : 컴퓨터 파일의 변경을 추적하는데 사용되는 버전 관리 시스템

기능
- 여러 개발자가 함께 작업
- 소스 코드의 변경 사항을 추적하는데 사용
- 소스 코드 관리에 분산 버전 제어 도구가 사용
- 여러 개의 평행 분기를 통해 비선형 개발을 지원

Git Bash : CLI (Command Line Interface) 명령행 인터페이스
Git GUI : GUI (Graphical user interface)

Github : 버전 관리를 위한 서버 저장소 및 프로젝트 개발을 위한 협업 관리 서비스
시스템 개발자와 운영을 담당하는 정보기술 전문가 사이의 소통, 협업, 통합 및 자동화 지원
프로젝트 소스 공유와 협업 소프트웨어 빌드 플랫폼

Git 설정
- 설정 명령 구조 : $ git config –설정범위 설정변수 설정값
- [--system | --global | --local] → 모든 사용자 | 모든 사용자의 모든 저장소 | 현재 사용자의 현재 저장소
→ $ git config ––global user.name ai7dnn
- $ git config –global core.editor 'code –wait' : 기본 편집기 설정
- $ C:\User\[사용자계정]\ .gitconfig : global인 경우 설정 파일

저장소 생성
- $ git init
- $ git init. : 현재 디렉토리를 git repository로 만들기 위해 사용
- $ git init basic : 현재 폴더 하부에 폴더 basic을 생성하고 git repository로 만들기 위해서 사용

리눅스 명령
- $ pwd : 현재 폴더 표시
- $ cd : Change Directory
- $ mkdir dname : Make Directory
- $ Is : File or folder list (-l 파일의 상세정보, -a 숨김 파일 표시, -al 위 두 개를 조합)
- $ touch fname : 빈 파일 fname 생성
- $ echo git bash, $ echo 'print( )' : 문자열을 컴퓨터에 출력, 따옴표 사용
-echo aaa > a.txt (파일 a.txt에 문자열 aaa 복사 / echo bbb >> a.txt (문자열 bbb 추가))
- $ cat fname : 파일 내용 보이기
- $ cp a b : 파일 a를 b로 복사
- $ mv f1 f2로 이름 수정
- $ rm fname : 파일 fname 삭제
- $ rm -rf dname : 하부에 서브폴더와 파일이 있어도 폴더 삭제, 옵션 사용
- f : 강제로 파일, 디렉토리 삭제 / - r : 디렉토리 내부의 모든 내용을 삭제
- '>' 기호 : 기존에 있는 파일 내용을 지우고 저장
- '>>' 기호 : 기존 파일 내용 뒤에 덧붙여서 저장
- '<' 기호 : 파일의 데이터를 명령에 입력 (cat<file1 : file1의 결과 출력)

버전관리를 위한 add, commit 명령
- $ git status [--long] : 현재의 상태 표시, 기본값
- $ git status [--short | -s] : 현재의 상태를 간단히 표시
- $ git config --global --edit : 도움말 보기

Commit : 버전 관리를 위해 현재 스테이지 영역의내용에 대해 스냅샷을 찍는 명령
- (-m --message , -a --all)

status
- Untracked file : 붉은색 표시는 작업 디렉토리를 의미 / 파일 처음 생성
- A new file : 녹색 표시는 스테이징 영역을 의미 / 처음 add한 파일
- Modified file : 스테이징 영역과 비교해서 작업 디렉토리의 파일이 수정된 것을 의미
- Modified file : 최근 커밋과 비교해서 스테이징 영역의 파일이 수정된 것을 의미

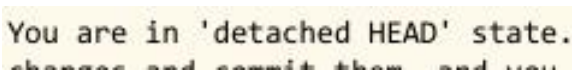
버전 로그 이력 확인
- $ git log : 로그 이력 정보를 표시
- $ git log --oneline : 로그 이력을 한 줄로 표시
- $ git log [--patch -p] : 로그 이력과 함께 파일의 변화 (이전 커밋과의 차이)를 표시

▶ 옵션 --oneline : 커밋 ID 일곱자리, (마지막 커밋 HEAD → 브랜치이름), 커밋 메시지 제목
▶ 옵션 --patch or -p : 커밋 정보 뿐 아니라 이전 커밋과 현재 커밋 파일의 차이 표시

- $ git show : 마지막 커밋(HEAD)의 커밋 정보 표시
- $ git show -- oneline : 커밋 로그 한 줄과 파일 차이 표시 [-s : 파일 차이는 표시되지 않음]
- $ git show[HEAD] : 지정한 HEAD 커밋 정보 표시
- $ git show [commitID] : 지정한 commitID의 커밋 정보 표시
  ▶ 인자가 없으면 HEAD 생략

여러 커밋과 로그 이력
- $ git log -- graph : 문자 그림으로 로그 이력 그리기
- $ git log -- reverse : 오래된 커밋부터 표시 / -- graph 함께 사용 불가능
- $ git log -- all : 모든 브랜치의 로그 이력 표시
- $ git log - n :최근 n개의 로그 이력 표시

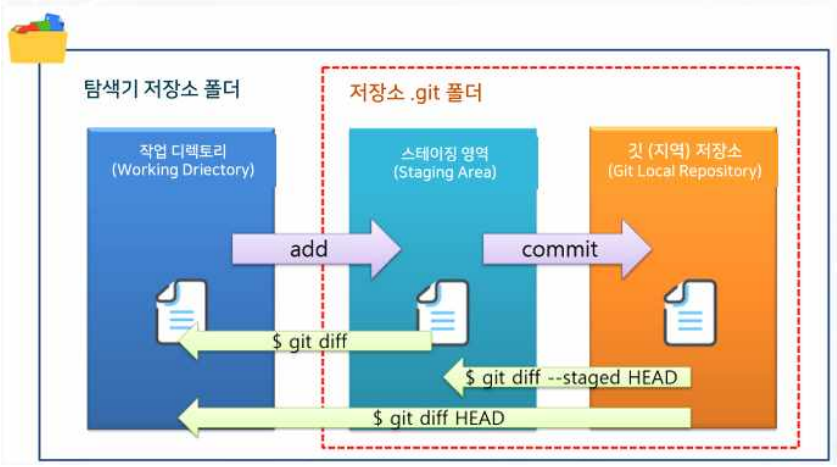
과거로의 시간 여행
- $ git checkout HEAD~ : HEAD 이전 커밋으로 이동
- $ git checkout - : 이전 checkout으로 이동
- $ git checkout main : 브랜치의 마지막 커밋으로 이동



▶ 현재 상태가 깨끗해야 checkout이 가능
▶ 떨어진 HEAD (detached HEAD) 상태 : HEAD가 마지막 커밋이 아닌 그 이전을 가리킨다는 의미

설명	git checkout	git switch
이전 커밋으로 이동	$ git checkout [이전커밋]	$ git switch -d [이전커밋]
다른 브랜치로 이동	$ git checkout [branch]	$ git switch [branch]
새로운 브랜치를 생성 후 이동	$ git checkout -b [newBranch]	$ git switch -c [newBranch]

3영역 파일 비교 diff
- HEAD~ HEAD^ : 하나 전 커밋
- HEAD~~ HEAD~2 HEAD^^ HEAD^~ : 두 개 이전 커밋



파일 삭제

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 file 삭제하겠다는 의미
    - Tracked 상태의 파일을 제거하여 Untracked 상태
- $ git rm -- cached [file]
    스테이징 영역에서 file 삭제 → 작업 디렉토리에서는 삭제되지 않음
    $ git ls-files 결과에서 보이지 않음 → 기본적으로 스테이징 영역의 파일 목록을 표시

▶ ??은 Untrached를 의미, 스테이징 영역에서 파일이 삭제되어 추적되지 않는 파일이 된 상태

파일 복원
- $ git restore [file] : 작업 디렉토리의 파일 f를 스테이징 영역의 파일 상태로 복구
- $ git restore -- staged f : 깃 저장소의 커밋 상태의 파일 f를 스테이징 영역에 복구
- $ git restore -- source=HEAD -- staged -- worktree f
    : 깃 저장소의 최신 커밋 상태의 파일 f를 작업 디렉토리와 스테이징 영역에 한 번에 복구
- $ git restore -- source=HEAD : 깃 저장소의 최신 커밋 상태 파일 f를 작업 디렉토리에 복구
- $ git restore -- source=HEAD^ -- staged f : 깃 저장소의 최신 커밋 파일 f를 스테이징 영역에 복구