중간고사 공부

1. 버전 관리 개요와 이해

▼ 버전 관리 시스템 종류

CVS, SVN, Git, Mercurial, Bazarr

▼ 버전 관리에서 파일이나 폴더, 버전 관리를 위한 관련 파일을 저장해 두는 곳

저장소(repository)

▼ 깃의 기능

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

▼ A website that hosts Git repositories oneline, making it easier for developers to share code

해석 : 온라인에서 Git 저장소를 호스팅하여 개발자들이 코드를 더 쉽게 공유할 수 있도록 하는 웹사이트

== github, gitlab, bitbucket

▼ **원격 저장소와 지역 저장소**

- **원격 저장소 → 지역 저장소 : pull (내리기)**
- **원격 저장소 ⇒ 지역 저장소 : clone (복사) / 원격 저장소의 별칭 이름 origin이 됨**
- **지역 저장소 → 원격 저장소 : push (올리기)**

▼ 시간 흐름에 따라 파일 집합에 대한 변경 사항을 추적, 관리하는 소프트웨어의 이름?

버전 관리 시스템 : version control system

2. 깃 설치와 리눅스 명령어

▼ 깃 설정의 범위

- '- - system' : 시스템 전체
- '- - global' : 전역. 현재 사용자에게만

- '- - local' : 기본값. 현재 깃 저장소에만

▼ **현재 폴더 하부에 저장소를 생성하는 명령어**

$ git init

▼ 마이크로소프트 사가 개발하는 오픈 소스 소프트웨어 편집기

vs code

▼ **리눅스 명령어 1**

- pwd : 현재 작업 중인 디렉토리(폴더)의 경로를 출력
- ls : 현재 디렉터리의 파일과 폴더 목록을 표시
- rm : 파일이나 디렉터리를 삭제
- mkdir : 새로운 디렉터리를 생성

▼ **리눅스 명령어 2**

- cat : 파일의 내용을 화면에 출력
- echo : 지정한 문자열이나 변수를 화면에 출력 / >,>>과 함께 사용하면 내용 추가 및 새로 저장
- cp : 파일이나 디렉터리를 다른 위치로 복사

3. 커밋과 과거로의 여행

▼ 깃의 세 가지 영역(상태)

- 작업 영역(작업 디렉토리)
- 스테이징 영역
- 깃 저장소 영역

▼ **깃 명령어 1**

- status : 현재 저장소의 상태를 확인
- config : 깃의 사용자 정보나 설정을 지정하거나 확인
- add : 변경된 파일을 커밋할 스테이징 영역에 추가헤
- commit : 내용을 스냅샷으로 저장

▼ **깃 명령어 2**

- branch : 저장소 내 브랜치 목록 보여줌

- checkout : 다른 브랜치나 특정 커밋으로 작업 디렉터리를 전환(브랜치가 아니라 커밋이라면 detached HEAD 가 됨) 또는 파일 복원
- switch : 브랜치 전환 전용
- log : 커밋 이력을 리스트 형태로 보여줌(커밋 내용 요약)
- show : 지정한 것의 상세 내용을 출력함(자세한 코드 변경까지)
- pull : 원격 저장소 가져와 현재 로컬 브랜치에 병합

▼ **깃 명령어 log의 옵션**

- - - oneline : 커밋 로그를 한 줄로 요약해서 보여줌
- - - reverse : 오래된 순서(첫 커밋) 부터 보여줌
- - - graph : 브랜치 병합 상태를 그래프로 시각화하여 보여줌

▼ **깃 명령어 commit의 옵션**

- - m : 커밋
- - a : 스테이징
- - am : 스테이징 커밋 한번에

4. 파일 비교와 삭제, 복원

▼ **diff : 파일 간 비교**

- $ git diff : 작업 디렉토리와 스테이징 영역 비교(add 전 상태)
- $ git diff - -staged 또는 $ git diff - -cached : 스테이징 영역과 마지막 커밋 비교(양쪽이 같으면 결과 표시되지 않음)
- $ git diff HEAD : 작업 디렉토리 전체와 마지막 커밋 비교

▼ **rm : 파일 삭제**

$ git rm [file] : 작업 디렉토리와 스테이징 영역에서 모두 file 삭제

$ 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

▼ **깃 파일 상태 표시**

- **A : 새로 스테이징 된 파일(커밋에 추가 될 예정)**

- **M : 기존 파일이 수정됨**
- **D : 파일이 삭제됨**
- **?? : Untracked. 깃이 아직 추적하지 않는 새 파일 add 전 상태**

$ git status -s 했을 때 해석하는 법 : XY 파일이름

X : 스테이징 영역 기준의 변화

Y : 작업 디렉토리 기준의 변화

▼ **restore : 파일 복구**

$ 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 --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***)

5. 브랜치 생성과 이동

▼ **버전(version) 이름 명명**

$ git tag 사용

▼ 브랜치란?

또 다른 작업의 흐름

▼ '- - patch'

변경 내용을 직접 선책하거나 수정할 수 있도롣 단계별로 패치를 적용하는 옵션

6. 저장소 생성과 지역과 원격저장소 연동

▼ 도구들

- Git Bash : 버전관리를 위해 사용하는 CLI 방식의 깃 도구
- vs code : 코드를 작성하고 git 버전 관리를 함께 할 수 있는 코드 편집기 겸 개발 환경
- SourceTree : git을 시각적으로 다룰 수 있게 해주는 GUI 방식의 버전 관리 도구
- git GUI : 기본 git에 포함된 간단한 그래픽 인터페이스로, 커밋과 스테이징 같은 작업을 마우스로 수행할 수 있는 도구

▼ **깃허브에서 다른 사용자 계정의 저장소를 자신의 깃허브 계정의 저장소로 복사하는 것**

fork

7. 기타

▼ **alias : 명령어 단축 별칭을 설정**

$ git config alias.<별칭><원래 명령어>