10주차

오픈소스 소프트웨어

- 누구나 특별한 제한 없이 그 코드를 보고 사용할 수 있는 오픈소스 라이선스를 만족하는 SW

- open Source Initiative (OSI)
  - 공개 소스 정의의 관리 및 촉진을 담당하는 비영리 조합

오픈소스 방식의 의미

- 소스 코드를 통해 여러 개발자가 협업하고 공유하며 이를 지원하는 방안을 마련
  - github
  - gitlab
  - bitbucket

- 오픈소스 커뮤니티 내의 사고 및 협업 양식
  - 커뮤니티에서 개발된 아이디어와 소프트웨어를 교환
  - 창의적이고 과학적이며 기술적인 발전을 이끌어낼 수 있다

오픈소스 소프트웨어 장단점

- 소스 코드를 공개
  - 누구든 보다 쉽게 연구하여 새로운 프로그래밍 기술을 개발

- 커스터마이징과 혁신 지원
  - 기업이 구체적인 요구 사랑에 맞게 커스터마이징 가능
  - 원래 소스코드에는 포함되지 않았던 새로운 사용자 경험으로 혁신할 수 있도록 지원

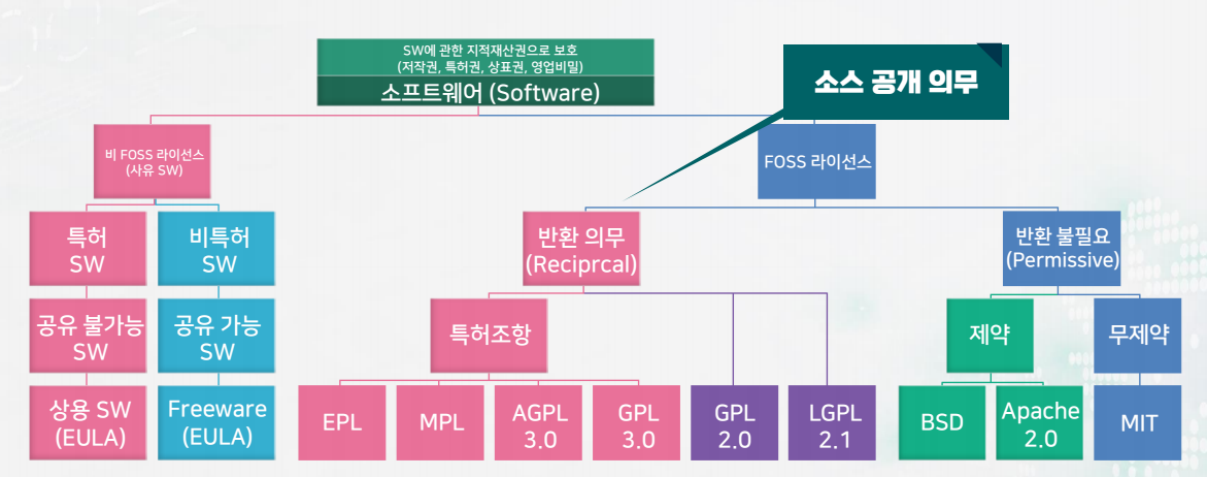
- 단점
  - 공개의 의무
  - 품질보증 및 유지보수, 보안 등의 어려움

오픈소스 개발 모델

- 대부분의 웹을 지원하는 서비스 스택 모델 : LAMP
  - Linux : 오픈소스 운영 체제이자 세계 최대 규모의 오픈소스 프로젝트
  - Apache : 초기 웹에서 핵심 역활을 한 오픈소스 크로스 플랫폼 웹 서버
  - MySQL : 대부분의 데이터베이스 기반 웹 애플리케이션에서 사용하는 오픈소스 관계형 데이터베이스 관리 시스템
  - PHP : 소프트웨어 개발에 사용되는 범용 스크립팅 언어

오픈소스 라이선스 저작권

오픈소스 SW 개발자가 만들어 놓은 저작권의 범위에 따라 해당 소프트웨어를 사용



소스코드 반환 의무

- GPL, AGPL , LGPL , MPL , EPL 등
  - 링크되거나 코드가 포함된 SW의 소스 코드를 공개
  - 저작권 고지 의무
    - 저작권법에 따른 법적 권리를 보장하는 것으로 공개 SW 하이선스에서 준수하지 않을 시 법정 분쟁의 가능성이 생긴다.

GPL (General public License)

- 자유소프트웨어재단 에서 만든 라이선스
  - 자유소프트웨어재단 설립자인 리처드 스톨만에 의해서 만들어졌다
  - 가장많이 알려진 카피레프트에 속한 라이선스
- 라이센스 적용 예
  - 리눅스 커널 ,Git , 마리아 DB , 워드프레스 등

AGPL (Affero GPL)

- GPL을 기반으로 만든 라이선스
  - 네트워크로 상호 작용하는 소프트웨어 소스 코드도 공개해야 한다는 조항을 추가한 라이선스
  - 서버에서 프로그램 실행해서 다른 사용자들과 통신 중
- 라이센스 적용 예
  - 몽고 DB등

Apache License

- 소스 코드 공개에 대한 의무 사항은 라이센스에 포함되어 있지 않다
- 라이센스 적용 예
  - 안드로이드 , 하둡 등

MIT License

- MIT에서 학생들을 지원하기 위해서 만든 라이선스
  - 라이센스와 저작권 관련 명시 의무
  - 가장 느슨한 조건을 가지고 있어서 많은 사람들이 사용하기 용이
- 라이센스 적용 예
  - 부트스트랩, Angular.js 등

대표 라이선스 정리

- 주요 의무 사항

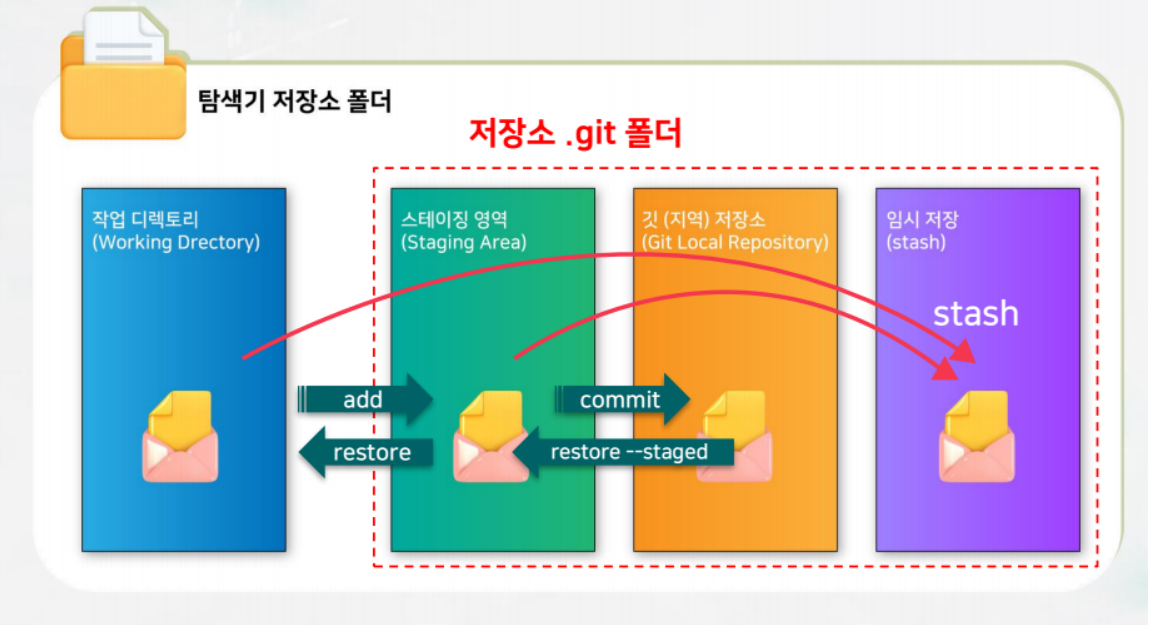
라이선스	배포 시 라이선스 사본 첨부	저작권 고지 사항 유지	소스코드 제공 의무와 범위	타 라이선스와 통합 및 배포 허용	수정 시 수정 내용 고지
GPL 2.0	O	O	전체 코드	조건부	X
GPL 3.0	O	O	전체 코드	X	O
LGPL 3.0	O	O	전체 코드	O	O
BSD	X	O	X	조건부	X
Apache2.0	O	O	X	O	X

- 의무 강도에 따른 분류

카테고리	소스코드 공개 의무	사례
Strong copy	2차 저작물 소스코드 공개 의무 있음	GPL
Weak copyleft	특정 조건에서 2차 저작물 소스코드 공개 의무 없음	LGPL
Non copyleft	2차 저작물 소스코드 공개 의무 없음	Apache, BSD, MIT

임시저장 Stash

- 깃 3영역 + stash 영역



- git Stash
  - 커밋할 필요 없이 파일의 변경 사항을 숨기거나 비밀리에 저장할 수 있는 강력한 도구
  - 따로 안전한 곳에 저장했다가 다시 가져올 수 있는 기능
  - 커밋할 게 없고 작업 트리가 깨끗해야 함

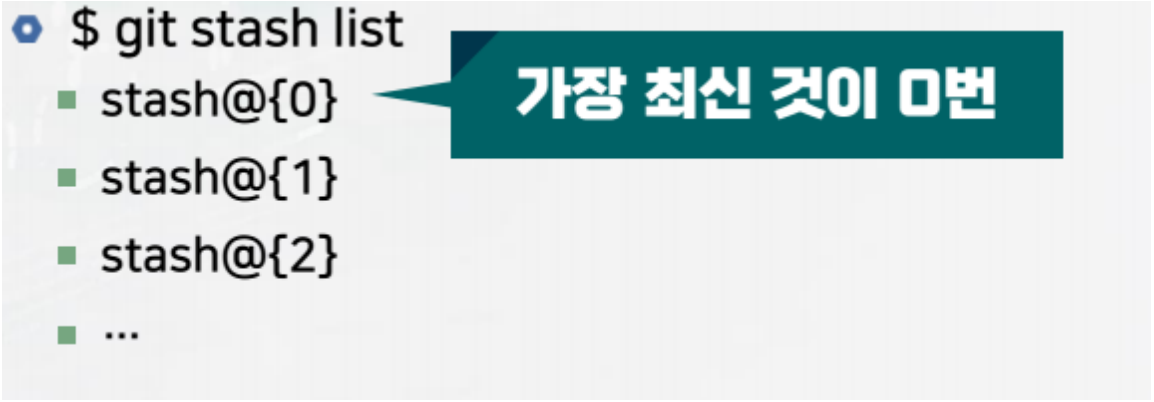
- stash 로 저장되는 내용
  - 작업 디렉토리 내용과 스테이징 영역 내용이 stash에 저장되고 작업 디렉토리 내용과 스테이징 영역 내용이 최신 커밋 자료로 남는다

- 임시저장 명령 stash
  - 작업 폴더와 스테이징 영역을 stash에 저장하고 작업 폴더를 정리
  - $ git stash
  - $ git stash -m “ 메시지 “
  - $ git stash save
  - $ git stash save “ 메시지 “

- 옵션
  - - -keep-index, -k
    - 스테이징 영역은 제외하고 작업 폴더만 저장
  - - -include-untracked, -u
    - Untracked 파일도 포함해 저장

- 임시저장의 최신 저장 내용으로 다시 복원
  - 기본으로 작업 디렉토리 내용만 다시 복사해 활용
    - $ git stash apply
  - 스테이지 여역도 함께 복사하기 위해서는 옵션 사용
    - $ git stash apply - -index

- 목록 보기
  - $ git stash list



- 최근 또는 지정된 임시 저장소 내용을 가져와 반영하고 삭제
  - $ git stash pop
  - $ git stash pop stash@{ n }

- 최근 또는 지정된 임시저장소 내용을 가져와 반영 , 작업디렉토리만 반영, stash 목록은 그대로
  - $ git stash apply
  - $ git stash apply stash@{ n }

- 최근 또는 지정된 임시저장소 내용을 가져와 반영, 작업 디렉토리와 스테이징 영역도 반영 , stash 목록은 그대로
  - $ git stash apply - -index
  - $ git stash apply - -index stash@{n}

- 임시 저장 목록 보기
  - $ git stash list

- 커밋 자료와 최신 stash 항목 간의 차이로 표시
  - $ git stash show
  - $ git stash show -p

- 커밋 자료와 해당 stash 항목 간의 차이로 표시
  - $ git stash show stash@{n}
  - $ git stash show stash@{n} -p

- 최근 임시 저장 내용을 삭제
  - $ git stash drop

- 지정된 임시 저장 내용을 삭제
  - $ git stash drop stash@{n}

- 모든 stash 목록을 모두 제거
  - git stash clear