Using git

why git?

- 분산버전관리 (Distributed Version Control System) → 이것이 주는 장점? → 기대효과
- 사용자의 요구사항
  - 추가할 기능 많음
  - 항상 모든 이슈사항은 응급!
  - 개발과 실제 운용 소스가 다름
  - 개발 사항에 대한 이력이 필요
  - 새 기능을 추가 중에 응급 사항의 수정 요청이 생기면 실제 운용 소스를 고치고 개발 소스에도 적용해야 함,

setting up a git

- 설치 osx, windows, linux
- global settings

git 시작하기

- 생성 init, clone
- git이 파일(디렉토리)를 관리하는 방법 working directory → stage area → git repository
- git repository의 파일 관리 add, remove, commit
- 상태 보기 status
- 뒤로 되돌아 가기 reset
- 버전 되돌아보기 log
- 태그 붙이기 tag
- 이건 저장하지 않을 거야 .gitignore
  - repository에 넣으면 좋은 것들; 스펙문서, 설계서, 매뉴얼, 도움말, 소스코드, 이미지, 외부라이브러리, 빌드스크립트
  - 넣지 말아야 할 것들; 바이너리, 설치본, 영업자료, 마케팅자료, 판매자료 등.
- stash

git branch

- 기준이 되는 메인 프로젝트와 개발 프로젝트 혹은 디버그 프로젝트, 핫픽스 프로젝트 master, branch
  - 가지 치기, 가지 잘라내기
- 지금 작업하고 있는 위치 head
  - 개발 브랜치에서 핫픽스 브랜치로 전환 checkout
- 메인+브랜치⇒새로운 메인, 브랜치1+브랜치2⇒새 버전의 브랜치1 merge
  - 병합(merge)하기 전에 diff

    - rebase

원격 repository

- 서버 repository에서 혹은 서버 repository으로 pull, push

git flow stories

- 도구의 사용법 보다 사용 방법이 더 중요하다.
- 메인 프로젝트는 놔두고 개발 프로젝트로 작업하기
    - master에서 develop 브랜치 만들어 작업하고 develop에서 release 브랜치 생성, release
- 한참 개발 중인데 급한 버그 수정 요청이 들어왔다
    - master에서 hotfix 브랜치를 만들어 작업하고 master와 현재 개발중인 브랜치로 merge
- 하나의 프로젝트 여러 명의 개발자
    - develop 브랜치에서 각 개발자별로 브랜치 생성후 작업 (보통 기능별로 개발자에게 분배) 혹은 develop에서 기능별 브랜치 생성

submodule

fork, pull request

- fork → clone forked repository → make branch and move to branch on forked repository → coding jobs on branch → push → pull request

references of git

- git 간편 안내서
- Pro git 온라인 북 - 한글판
- git-flow
- git 사용자 설명서