용어

- Commit
  - 커밋을 하는 행위가 HEAD에 반영한다는 뜻
  - 파일 or 폴더의 변경사항 저장.
  - 영문/숫자로 이루어진 40자리 고유이름을 가진다.
  - 커밋 메세지 (Git 권장)
    - line 1: 변경내역 요약
    - line 2: 공란
    - line 3: 변경이유
  - `git commit --ammend` 가장 최근 커밋내용에 내용 수정 (설명뿐만 아니라, **파일도 추가/수정** 가능하다.)
  - `git checkout -- <파일명>` 로컬의 변경내용을 변경 전 HEAD로 되돌림추가한 내용과 변경내용은 남아있음.(commit만 안한 상태)
- Stage (=Index)
  - commit한 이후 push는 안한 상태
  - 등록되지 않은 파일은 commit 되지 않는다.
- Checkout
  - 사용법
    - `git checkout <브랜치명>/<태크명>`
    - `git checokut -b <새브랜치> <기존브랜치>` 새로운 브랜치를 만들면서 입장합니다.
  - 해당 브랜치 입장
  - `git checkout --` 커밋전 파일의 변경내용을 취소하고 이전 커밋으로 되돌림
- HEAD
  - 현재 사용주인 Branch 의 선두부분
  - 이동
    - HEAD를 이동하면 Branch가 변경
    - ^ or ~1 or ~ : 바로 이전 상태로
    - ^^ or 2: 2단계 앞으로
  - `git reset HEAD --` 커밋된 파일들을 취소
- Stash
  - 파일의 변경 내용을 일시적으로 기록해두는 영역
  - Commit 보류
  - 나중에 다시 불러와서 branch에 commit 할 수 있음
- Branch
  - 사용법
    - `git branch bugFix` (만들고) / `git checkout bugFix` (bugFix로 갈아탐)
    - `git checkout -b bugFix` bugfix 브랜치를 만들고 갈아탐`
    - `git checkout -b release-0.90 development` evelopment 브랜치로부터 만듦`
    - `git branch -m <기존브랜치> <새로운브랜치>` 기존브랜치를 새로운 브랜치로 변경합니다. -M 옵션을 사용하면 이미 있는 브랜치라도 덮어씁니다.
  - 통합 브랜치: Master Branch
  - 토픽 브랜치: Feater Branch

    - 성공적인 Git 브랜칭 모델
        - Main Branch (master, develop)
        - Feature or Topic Branch (develop-featcher)
        - Release Branch (develop - release(bugfix) - mastert순)
        - Hotfix Branch (master-hotfix 긴급수정)
- 통합
    - Merge
        - 사용법
            - `git merge <브랜치>` <브랜치>를 현재 브랜치로 합침.
            - `git checkout master`
            - `git merge --no-ff release-0.9.0` release 브랜치를 merge. no-ff는 히스토리를 명시적으로 만듦
        - FF(Fast Forward) Merge: 분기된 branch만 변경사항이 있을때
            - ex. Master -> BugFix -> Master
            - non fast-forward 옵션 : bugFix branch가 그대로 남기 때문에 관리상 유용
        - Merge Commit : 양쪽의 변경을 가져와서 수정후 Commit
        - Merge 단점: 변경사항이 그대로 남아있기때문에 이력이 복잡해짐.
        - 토픽브랜치에 통합 브랜치의 최근 코드를 적용할 경우 무슨 코드들이 바뀌었는지 확인할 필요가 있기 때문에 rebase 사용.
        - ex) `git checkout master` `git merge hotfix` : Host가 통합 브랜치이다.
        - `git merge --squash`
            - 해당 브랜치의 커밋 전체를 통합한 커밋 추가
            - 토픽브랜치 안의 커밋을 한꺼번에 모아 통합 브랜치에 병합할때 사용
    - Rebase
        - 사용법
            - `git rebase <브랜치>` <브랜치>를 현재 브랜치에 적용합니다.
        - 이력은 단순해지지만, 토픽 브랜치의 commit 이력이 변경됨.
        - 토픽 브랜치가 그대로 통합 브랜치의 끝(HEAD)에 얹혀지는 스타일.
        - 각각의 commit에서 충돌내용을 수정할 필요가 있음.
        - 통합 브랜치의 HEAD(위치)는 이전 상태로 그대로 있으니 bugFix Branch와 FF Merge를 해서 옮겨주어야 한다.
        - ex) 통합 브랙치에 토픽 브랜치를 불러올 경우 rebase 한뒤 merge
        - ex) git checkout issue3 / git rebase master : merge와 달리 Host가 토픽브랜치이다.
        - 충돌시 취소하려면 git rebase --abort
        - git rebase -i
            - 특정 커밋을 다시 쓰거나 다른 커밋과 바꾸기. 특정 위치 커밋 삭제나 여러 커밋을 하나로 통합도 가능.
            - `git rebase -i HEAD~5`
                - s(squash)는 이전 커밋과 병합
            - 용도
                - push 하기 전에 커밋내용 정리
                - 이전 커밋에 누락된 파일 추가
                - 알기 쉽게 그룹으로 통합

        - 취소
            - `git reflog`
            - `git reset --hard HEAD@{N}`
- Pull
    - 가져오기 and 병합하기
    - 내부적으로 Fetch + Merge
- Fetch
    - 가져오기
    - 원격저장소(origin)의 내용 확인만 하기
    - 이름없는 브랜치로 가져옴. git checkout FETCH_HEAD 로 체크아웃 가능
- Push
    - 밀어넣기
- Tag
    - 일반 태그 (Lightweight Tag) - 이름정보만
        - `git tag <tag-name>`
    - 주석 태그 (Anootated Tag) - 상세정보 포함
        - `git tag -a <tag-name>`
        - `git tag -am <comment> <tag-name>`
    - `git tag` 태그목록
    - `git tag -n` 태그목록과 주석내용
    - `git tag -n <tag-name>` 태그삭제
    - `git checkout <tag-name>` 바로 특정상태로 되돌리기
- Remote
    - 사용법
        - `git remote add origin <원격저장소 주소>` 원격저장소 주소 등록
        - `git push orgin master` 원격저장소 브랜치로 밀어냄.
        - `git pull` 원격 저장소에 맞춰 갱신. 내부적으로 fetch후 merge
        - 초기화
            - `git fetch origin`
            - `git reset --hard origin/master`
- Diff
    - 병합 충돌(conflicts)시 직접 수정후
    - `git diff <원래 브랜치> <비교대상 브랜치>`
- Revert
    - 특정 커밋내용 삭제
    - 내부적으로 실제 삭제 하는게 아니라, 해당 커밋의 삭제 커밋을 새로 만들어 안전하게 처리
- Reset
    - 커밋만 되돌리고 싶을때 --soft
    - 변경한 인덱스의 상태를 원래대로 되돌리고 싶을때 --mixed
    - 최근의 커밋을 완전히 버리고 이전상태로 복구 --hard
    - 실수로 reset 했을경우
        - `git reset --hard ORIG_HEAD`
- Cherry-pick
    - 다른 브랜치의 특정 커밋을 복사하여 현재 브랜치로 가져옴

    - `git cheery-pick e4ec`
- Log
    - 기본적으로 log가 아니라 식별자 알아보기
    - `git log --decorate` 태그정보를 포함한 이력을 확인
    - -1이나 -2의 옵션을 출력 로그의 갯수를 지정할수 있음
- Reflog - HEAD의 이동 내역
- Clean - 관리대상이 아닌 파일 삭제
- Tip
    - git reset --hard HEAD~ : commit 취소