**GIT 명령어**

▣ 프로젝트 버전 관리

$ git init // .git 하위 저장소를 생성

$ git remote // 원격 저장소 목록 확인

$ git remote -v // 원격 저장소 URL 확인

$ git remote add <원격별칭> <URL> // 원격 저장소 추가

$ rmdir /s .git // 버전 관리 초기화 (Windows용)

▣ 저장소 복제 및 다운로드

$ git clone <https:...> // 기존 소스 코드 다운로드

$ git clone /로컬/저장소/경로 // 로컬 저장소 복제

$ git clone 사용자명@호스트:/원격/저장소/경로 // 원격 서버 저장소 복제

▣ 스테이징 (변경사항을 임시로 보관)

$ git add <파일명> // 특정 파일 하나를 스테이징

$ git add \* // 숨김 파일을 제외한 일반 파일 스테이징

$ git add . // 모든 파일 스테이징

$ git mv <파일A> <파일B> // 스테이징된 파일 이름 변경

$ git rm -r -cached <파일> // 추적 목록에서 제거

$ git clean -fdn // 삭제 가능한 추적되지 않은 파일 목록 확인

$ git clean -fd // 추적되지 않은 파일 삭제

$ git restore --staged <파일> // 특정 파일 언스테이징

$ git restore --staged . // 모든 파일 언스테이징

▣ 커밋 (변경사항 적용하여 버전 생성)

$ git commit -m ‘<메시지>’ // 버전 생성

$ git commit -am ‘<메시지>’ // 추적 파일 스테이징 및 버전 생성

▣ 브랜치 (나뭇가지처럼 분기점을 나눠서 병령로 진행)

$ git branch // 로컬 브랜치 목록 확인

$ git branch -r // 원격 브랜치 목록 확인

$ git branch -a // 로컬 및 원격 브랜치 목록 확인

$ git branch <브랜치> // 브랜치 생성

$ git branch -D <브랜치> // 브랜치 삭제

$ git branch -m <브랜치> // 현재 브랜치 이름 변경

$ git branch -m <이전 브랜치> <새 브랜치> // 특정 브랜치 이름 변경

$ git branch --show-current // 현재 브랜치 이름 확인

$ git checkout <브랜치> || git switch <브랜치> // 브랜치 전환

$ git checkout -b <브랜치> || git switch -c <브랜치> // 브랜치 생성 및 전환

$ git checkout <해시> // 특정 버전 체크아웃

▣ 로그확인

$ git log // 현재 브랜치의 버전 내역을 확인

$ git log --all // 모든 브랜치 내역 확인

$ git log --oneline // 간략한 버전 내역 확인

$ git log --graph // 그래프 형태로 버전 내역 확인

$ git show // 현재 브랜치의 최신 버전 확인

$ git show <브랜치> || <해시> || <파일> // 특정 브랜치의 최신 버전 확인

$ git blame <파일> // 특정 파일의 작업자 확인

$ git blame -L <시작>,<종료> <파일> // 시작부터 종료 줄까지 작업자 확인

▣ 밀어내기 (로컬 저장소 -> 원격 저장소)

$ git push <원격별칭> <브랜치> // 원격 저장소로 밀어내기

$ git push <원격별칭> <브랜치> -f // 원격 저장소로 강제로 밀어내기

$ git push <원격별칭> -all // 원격 저장소로 모든 브랜치 밀어내기

▣ 당겨오기 (원격 저장소 -> 로컬 저장소)

$ git pull <원격별칭> <브랜치> // 원격 저장소에서 브랜치 당겨오기

$ git pull --rebase <원격별칭> <브랜치> // 로컬 브랜치에 덮어쓰기

▣ 가져오기 (원격 저장소의 내역을 로컬의 원격 내역과 동기화, 로컬 브랜치에 영향을 주지 않는다.)

$ git fetch // 현재 원격 저장소의 브랜치 목록 가져오기

$ git fetch <원격별칭> // 특정 원격 저장소의 브랜치 목록 가져오기

$ git fetch --all // 모든 원격 저장소의 브랜치 목록 가져오기

$ git fetch --prune // 브랜치 목록 가져와서 로컬의 원격 브랜치 목록 덮어쓰기

▣ 비교

$ git diff <파일> // 파일의 수정 내용 확인

$ git diff <파일A> <파일B> // A와 B 파일 비교

$ git diff <브랜치> // 특정 브랜치와 현재 브랜치 비교

$ git diff <브랜치A> <브랜치B> // A와 B 브랜치 비교

$ git diff <브랜치A>:<파일> <브랜치B>:<파일> // A와 B 브랜치의 파일 비교

$ git diff <해시> // 현재 버전과 특정 버전 비교

$ git diff <해시A> <해시B> // A와 B 버전 비교

▣ 작업취소 (Rollback)

$ git checkout HEAD -- <파일> // 특정 파일 롤백

$ git restore <파일> // 특정 파일 롤백

$ git restore . // 모든 파일 롤백

$ git reset --hard HEAD // 모든 파일 롤백

▣ 초기화

$ git reset --hard HEAD~<번호> // 번호만큼 이전 버전으로 리셋

$ git reset --hard HEAD~1 // 직전 버전으로 리셋 (1버전 전으로)

$ git reset --hard <해시> // 특정 버전으로 리셋

$ git reset --hard HEAD^ // 마지막 버전을 삭제

$ git reset --hard // 수정 내용을 버리기

$ git reset --soft // 수정 내용을 스테이징

$ git reset --mixed // 수정 내용을 스테이징하지 않음

▣ 되돌리기

$ git revert <해시> // 특정 버전을 취소하고 새로운 버전 생성

▣ 임시저장

$ git stash list // 임시 저장된 작업 목록 확인

$ git stash -m ‘<메시지>’ // 메시지와 함께 현재 작업을 임시저장

$ git stash -am ‘<메시지>’ // 위와 같으나 미추적 파일 포함 임시저장

$ git stash show <번호> // 특정 번호의 임시저장 내용 보기

$ git stash apply <번호> // 특정 번호의 임시저장 내용을 현재 브랜치에 적용

$ git drop 번호 // 특정 번호의 임시저장 내용 삭제

$ git stash pop // 가장 최신의 임시저장을 적용하고 목록에서 삭제

git stash clean // 모든 임시저장 목록 삭제

▣ 병합

$ git merge <브랜치> // 현재 브랜치에 특정 브랜치 병합

$ git merge --abort // 충돌 시, 병합 과정 중단

▣ 재배치

- 재배치는 현재 브랜치의 내역을 대상 브래친의 최신 버전 다음으로 이동하는 기능이다.

$ git rebase <브랜치> // 현재 브랜치를 대상 브랜치로 재배치

$ git rebase --continue // 재배치 계속 진행

$ git rebase --abort // 재배치 과정 중단

ex) (main) -- A - B - C (main) -- A - B - C - D - E

| ->

(dev) ---- D - E

1. git checkout dev : 재배치할 브랜치로 전환

2. git rebase main : 현재 브랜치(dev)를 대상 브랜치(main)로 재배치 시작

3. 충돌(Conflict) 발생 시 해결

4. git add . : 충돌 해결 후 스테이징

5. git rebase --continue : 재배치 계속 진행

6. 버전 메시지 수정 및 저장(:wq)

7. 3~6번 과정 반복 및 재배치 완료!

**정리 (https://github.com/KennethanCeyer/tutorial-git)**

새로운 버전 생성하기 스테이징(git add) -> 커밋(git commit)

상대방 커밋 파일 업데이트 당겨오기(git pull) or 가져오기(git fetch)

복원하기 초기화(git reset)

브랜치 (가지치기 및 관리) 생성(git branch new) -> 접속 (git checkout new)

or 생성 및 접속 (git checkout -b new)

브랜치를 원격 저장소에 등록 밀어내기(git push origin new)

브랜치의 삭제 (git branch -d new)

브랜치의 삭제를 원격 저장소에 반영 (git branch push origin :new)