git init

버전관리 할 수 있도록 초기화

git config --global user.email "내 이메일"

git config --global user.name "닉네임"

git status

git이 관리하는 파일 목록 출력

git add

git이 관리하는 파일 추가 (뒤에 *이나 .붙이면 모두 추가)

git commit

지금 상태를 기억하라는 의미 (vim에디터가 나오는데 이때 어떤 commit인지 메모, 처음은 보통 Initial commit로 정함)

git commit -m "Initial commit"

vim 없이 commit 적용

git commit -am "commit 이름"

전부 add하면서 commit를 적용

git log

commit 목록

git shortlog

commit 목록을 더 간단하게

git checkout -- 파일이름

수정사항 복구

git diff

어떤 글자가 바뀌었는지 알 수 있다

git remote add origin "github 주소"

원격서버 주소를 origin이라는 이름으로 추가

git remote -v

생성된 remote 목록

git remote remove origin

origin이라는 remote를 제거

git push origin master

로컬 컴퓨터에 있는 변경사항을 원격 서버 주소로 업로드 (master는 로컬 브랜치의 이름)

git pull origin master

서버(github)에서 수정했던 commit을 가져오고 병합

set LC_ALL=ko_KR.UTF-8

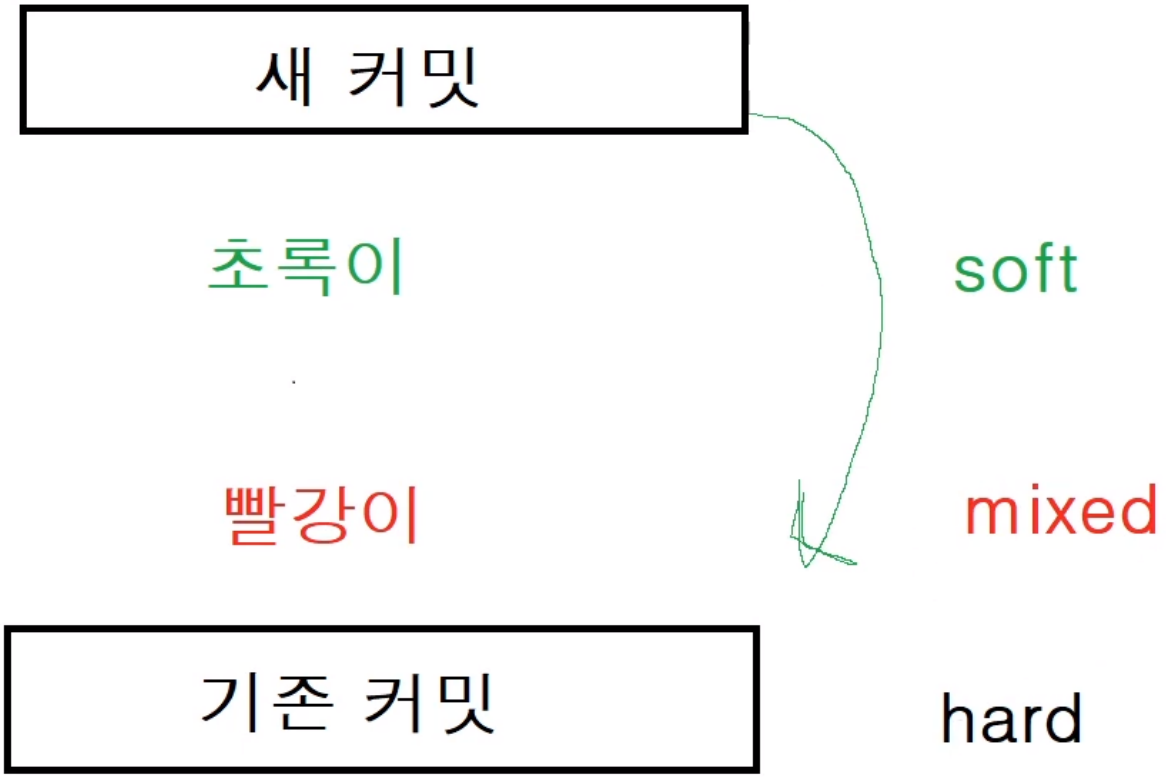
터미널에서 한글이 깨졌을 때 해결방법 (터미널에 따라 다름)

git config --global core.quotepath false

터미널에서 한글파일 이름이 깨졌을 때 해결 방법

git reset HEAD~1

바로 1단계 전의 commit으로 되돌리기 (HEAD는 현재 커밋, 기본 속성 mixed, 단계식 말고 commit ID로도 가능)



git reset HEAD~1 --soft

git reset HEAD~1 --hard

git reflog

HEAD와 브랜치의 이동 기록을 보여주는 명령

git revert HEAD

커밋에서 발생한 변경을 취소하고 새로운 커밋으로 생성 (vim에디터가 나오며 commit 이름 정할 수 있음, 마찬가지로 HEAD 말고 commit ID 써도 됨)

git branch

branch 목록 출력

git branch development

development라는 이름의 branch 추가

git checkout development

현재 작업 중인 브랜치를 development 브랜치로 전환

git branch -m <이전브랜치이름> <새로운브랜치이름>

브랜치 이름을 변경

git merge development

다른 브랜치에 있는 commit을 현재 commit과 병합 (충돌 가능)

git merge –-abort

merge한 것을 취소

git rebase –-continue

merge와 동일하지만 커밋 히스토리에 병합 기록이 안남음

git cherry-pick commitID

다른 branch의 특정 commit를 불러와서 병합

git tag 태그이름

commit 아이디를 외우기 힘들 때 이런식으로 태그 추가 가능

git tag

태그를 붙여 놓은 목록 보기

git stash

지금까지 해놓은 것들 임시저장

git stash list

임시저장 목록

git stash apply

임시저장 적용

git fetch

pull이랑 비슷하나 pull은 병합까지하고, fetch는 로컬영역을 바꾸지 않음 (즉, 비교만 한다)

git merge origin/master

origin 원격 주소로부터 fetch한 것을 master 브랜치와 병합