2/6- 2/7

Init : git이 관리하고 있는 로컬저장소 (git init 로컬저장소 생성) .git 파일

**Git init 저장소에 필요한 뼈대 파일 🡪 git add 명령을 통해 파일추가🡪 git commit으로 명령저장**

Ctrl +u 명령어 취소

Hello.txt 파일 생성

🡪 Git add hello.txt (W.D에서 STAGE에 올리기)

Staging area에 있는 파일들은 git diff명령을 해도 아무것도 출력x

Git diff –cached/ staged의 명령을 통해, unstaged 상태인 변경 부분을 확인가능

🡪 Git rm hello.txt (unstage 과정)

Commit 명령어 입력직전 git add . 필수로 입력을 해야만 stage에 올릴 수 있음

🡪 Git commit –m(message) ‘[수정파일의 수정(merge/ delete/ edit)] 정파일’

git commit -m '[hello2.txt] hello.txt hello.txt: 내용'

git commit -a -m [merge] master & br1-1: hello3.txt"

GIT LOG 이력확인 (q 입력시 명령어 입력가능)

!! : 방금전 명령어 입력

Txt 파일에서 내용을 추가하여 저장시 modified: hello.txt 로 나옴

Git restore hello.txt 입력시 저장된 내용 삭제 (git status 상태확인 가능)

ls: local 파일 불러오기 / ls –l 파일 개수 밑 생성시간

git log –oneline 해시키별로 기록확인 가능

Ctrl + c 전에 쳤던 명령어 취소

Git add . (모든 파일 추가)

* Git 은 staging area에 속한 스냅샷을 커밋하기 때문에 –a옵션 추가 시 tracked 상태의 파일을 자동으로 staging area에 넣는다

git commit -a -m '[수정] hello3.txt: 'the end'" add와 동시에 commit하기

-a 한번 관리가 되었던 파일에 한해서 -a 사용가능 (이전에 한번이라도 git add . 입력했으면 –a –m 한번에 입력가능

Q 뒤로가기

Git reflog 모든기록확인 git show hashcode입력 시 해당작업 내용 확인가능

Git tag 작업지점표시 git tag 해쉬키 (해당작업지점표시) / git tag 브랜치 해쉬키

Git branch branch추가 git branch –-all(or –a) 모든 branch보기

git branch –ㅣ local branch만 확인가능

Git switch(=checkout) : branch 의 이동

생성과 동시에 이동하려면 git switch –b 새로branch [브랜치를 만들면서 체크아웃 b=branch]

Git branch –D 해당브랜치 브렌치 강제삭제 / git branch –d 해당브랜치 원하는 브렌치 삭제

W 명령어 저장 / q명령어 종료 / ctrl + c 뒤로가기

Branch의 분기점 설정 git branch hotfix-1

Branch 생성시 해쉬키/ tag명 으로도 생성가능 git branch br- 해쉬키 / 태그명

* 전환점이 되는 경우 중간에 branch 생성을 통해 표시가능

Git merge 해당branch : 병합가능 cat(alogue) 해당파일 : 내용확인 가능

Rm(remove) –r(ecursive) 하위 폴더 지우기

임시 저장

Git stash save “사용자 옵션 메모작성 가능”

Stash 된 변경 사항은 해당 저장소의 모든 브랜치에 접근가능

새 파일을 만든 후 변경사항을 stash하려고 하면 지정할 로컬 변경사항이 없을 수 도 있으므로 애러 발생가능 ( no local change to save) 🡪 따라서 git stash –include-untracked 실행하면 untracked file도 stash가 가능

Stash한 변경사항 확인하기 git stash list 임시 저장 내용 확인 시 git stash show <stash의 이름>

(라인별 변경 사항을 +와 -기호로 표시된)으로 변경 사항을 확인하고 싶다면, 명령어에 패치라는 의미의 -p 플래그를 포함하면 됩니다. 예시

git stash show -p stash@{0}

# 예시 결과물:

diff --git a/PathToFile/fileA b/PathToFile/fileA

index 2417dd9..b2c9092 100644

--- a/PathToFile/fileA

+++ b/PathToFile/fileA

@@ -1,4 +1,4 @@

-현재 브랜치에 보이는 내용

+stash된 변경 사항이 반영된 내용

Git stash drop <stash-이름> 브랜치에 적용하지 않은 상태에서 삭제

Git stash clear stash영역에 있는 모든 내용을 삭제