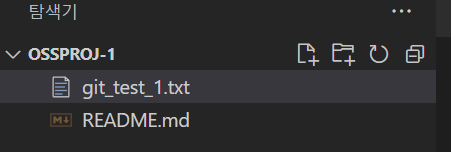
**OSSProj 과제 – Git & Github**

**2018111750 이은학**

repo : <https://github.com/Ag-crane/OSSProj-1.git>



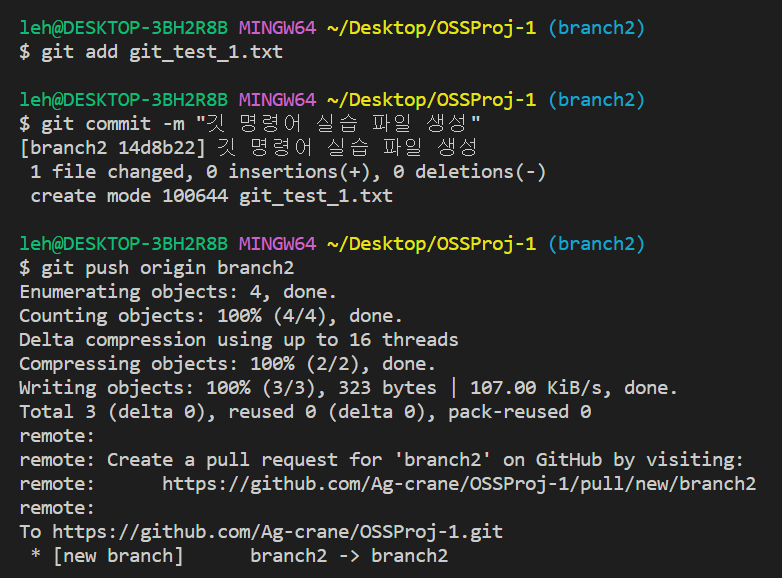
수업시간에 만들었던 repo를 Working directory로 선택, git\_test\_1.txt 라는 빈 텍스트파일을 생성하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

git checkout branch2 : 작업 branch를 branch2로 변경

( 미리 생성된 branch를 사용했습니다. 생성할 시에는 git branch branch2 명령어 사용 )



git add git\_test\_1.txt : 생성한 git\_test\_1.txt 파일을 staging area에 저장한다

git commit -m “깃 명령어 실습 파일 생성” : staging area에 저장된 변경사항을 “” 안의 커밋메시지와 함께 local repository로 옮겨 저장한다.

git push origin branch2 : local repository에 있는 변경사항들을 origin ( 리모트 저장소. 여기서는 <https://github.com/Ag-crane/OSSProj-1.git> )의 branch2 에 업로드한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

깃허브 GUI로 Pull Request 수행 : branch2의 변경사항(커밋된 내용)을 main branch에 merge 한다.

텍스트이(가) 표시된 사진

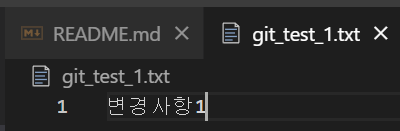
자동 생성된 설명

git checkout main : 다시 작업 branch를 main으로 바꾸자, 2개의 commit 이 있다는 문구가 뜨며, can be fast-forwarded : 충돌 없이 pull 이 가능하므로 git pull을 사용하여 로컬 브랜치를 업데이트하라는 뜻이다.

git pull : fast-forward가 가능하므로 한번에 pull. 깃허브에서 merge하여 바뀐 main 브랜치의 변경사항이 local 에도 적용된다 => local의 main 브랜치에도 git\_test\_1.txt 파일이 생성되었다.

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

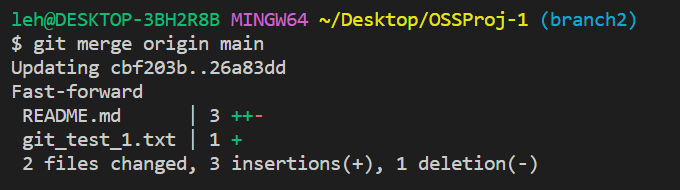
로컬의 main 브랜치에서 텍스트파일을 수정하고 (변경사항1), add, commit, push를 수행하였다.

마찬가지로 변경사항2까지 추가한 후에, 로컬의 branch2 브랜치로 pull을 받아보는데, 이번에는 pull 명령어가 아니라 fetch & merge 명령어를 사용해보았다.

텍스트이(가) 표시된 사진

자동 생성된 설명

git fetch origin main : 리모트 저장소의 main 브랜치의 내용을 로컬 저장소로 가져온다. 이때 충돌이 있으면 해결하도록 문구가 뜨는데, 이번에는 충돌이 일어나지 않은 모습이다.



git merge origin main : 무사히 fetch가 완료된 후 충돌이 없으므로 로컬로 가져온 내용을 working directory로 가져와 병합한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

로컬에서의 branch2 브랜치에도 변경된 내용이 모두 적용된 모습이다.