1.

git diff : working directory와 staging area를 비교

git diff –cached : 커밋내용과 staging area를 비교

git diff commit1 commit2 : 커밋1과 커밋2를 비교

git commit –amend : 최근 commit 후 추가 파일을 추가해 커밋내용을 수정한다

git reset --soft HEAD~ : head위치에 있는 커밋된 버전을 ~의 수만큼 전으로 옮기고 원래 head 버전의 파일을 WD와 Staging Area에 저장해 둔다. 커밋만 하면 원래상태로 돌아감

git reset HEAD~2 : 기본옵션으로 –soft와 같으나 WD에만 원래 head버전을 저장하기에 add와 commit해야 원래 상태로 돌아감. ~2는 전전버전을 의미

git reset --mixed HEAD~ : 기본옵션으로 위와 같은 역할을 하나 ~가 하나이기에 전버전을 의미함

git reset --hard HEAD~ : head버전의 파일 커밋내용을 아예삭제하고 WD에도 전 버전의 파일로 바꾼다.

git rm sample.txt : Tracked file인 sample.txt를 삭제한다

git rm --cached sample.txt : 로컬이 아닌 git 저장소에서만 sample.txt를 삭제한다

git clone “Remote repo URL” : 원격저장소를 현재 로컬저장소로 복제한다

git clone “Remote repo URL” abc : 원격저장소를 abc라는 장소로 복제한다

git clone -b Lab1 “Remote repo URL” abc : Lab1의 브랜치 이름으로 원격저장소를 abc라는 장소로 복제한다

git remote : 로컬저장소와 원격저장소의 연결 정보를 불러옴

git remote -v : 로컬저장소와 원격저장소의 push, pull, fetch등의 연결 이름을 알려줌

git remote add origin “remote repo URL” : origin의 별칭으로 원격저장소에 로컬저장소를 연결

git remote remove origin : origin의 연결을 삭제함

git pull 사용법 : git pull <원격 저장소 별칭> <pull을 하는 브랜치 이름>

git push 사용법 : git push <원격 저장소 별칭> <push 하는 브랜치 이름>

git fetch 와 git pull의 차이 : pull = fetch + merge로 pull은 연결되어 있는 원격저장소의 내용을 가져와 자동으로 병합 작업을 하지만 fetch는 원격 저장소의 최신 이력을 확인할 수 있음.

git branch –h : git branch 명령어의 추가 설정 명령어들에 대한 설명을 알려줌

git branch -a : 현재 생성 되어있는 브랜치들을 모두 알려줌

git branch : 생성되어 있는 브랜치 list를 출력하고 현재 브랜치를 표시함

git branch issue1 : issue1의 이름의 브랜치를 추가함

git branch –d issue1 : issue1의 이름의 브랜치를 삭제함

git branch issue2 : issue2의 이름의 브랜치를 추가함

git checkout issue2 : issue2 브랜치로 이동

git checkout –b issue1 : issue1 브랜치를 만들고 이동

git branch –D issue1 : 이유막론하고 issue1 브랜치를 삭제함

git checkout master : master 브랜치로 이동함

git stash : 작업 변경 내용을 stash로 임시 저장함

git stash save : 작업 변경 내용을 stash로 임시 저장함

git stash list : stash 목록을 확인한다

git stash pop : stash apply와 stash drop작업을 한번에 한다

git stash apply : 가장 최근의 stash의 내용을 가져와 적용함

git stash drop : 만든 stash를 스택에서 삭제한다

git stash clear : 모든 stash를 삭제함

git merge TopicA : Fast Forward 방식으로 TopicA 브랜치로 커밋한 내용을 main 브랜치에 병합 (이때 merge commit 내역 없이 main 브랜치에서 작업한걸로 표기)

git merge --no-ff TopicA : No Fast Forward 방식으로 TopicA 브랜치로 커밋한 내용을 main 브랜치에 병합 (main이 아닌 TopicA 브랜치이름으로 merge commit 내역을 생성)

merge conflict 상황설명 : 여러 브랜치에서 main 브랜치로 merge할 경우 파일의 내용등이 달라 일어나는 충돌

merge conflict 해결방법 : 사용자가 직접 해당 파일 내용을 add commit해서 수정, cherry-pick을 사용해 방지

revert 와 reset의 차이 : revert는 원하는 커밋 버전을 최신 head에 새로 가져 오는것이고 reset은 원하는 커밋 버전으로 head를 옮기고 그 이 후의 버전들은 모두 삭제

2. 실습

git push

Github login

Create new repository ( URL ) : <https://github.com/Youngkwan-Cho/Lab7.git>

Remote repo URL 복사

Local Repo에서 원격 repo 에 연결( 화면캡쳐 )

git push origin master( 화면캡쳐 )텍스트이(가) 표시된 사진

자동 생성된 설명

원격 repo 확인( 화면캡쳐 )

텍스트, 모니터, 스크린샷, 검은색이(가) 표시된 사진

자동 생성된 설명

git pull

원격 Repo 변경( 웹 commit 리스트 화면캡쳐 )텍스트, 모니터, 검은색, 스크린샷이(가) 표시된 사진

자동 생성된 설명

Local Repo 이력 변경(git log 화면캡쳐 )

텍스트이(가) 표시된 사진

자동 생성된 설명

git push origin master( 화면캡쳐 )텍스트이(가) 표시된 사진

자동 생성된 설명

오류 확인 ( 화면캡쳐 )

텍스트이(가) 표시된 사진

자동 생성된 설명

git pull origin master ( 화면캡쳐 )

텍스트이(가) 표시된 사진

자동 생성된 설명

git push origin master ( 화면캡쳐 )

텍스트이(가) 표시된 사진

자동 생성된 설명

3.

**s21900706@walab-HGU**:**~/2022OSS/lab7**$ git init

**s21900706@walab-HGU**:**~/2022OSS/lab7**$ vim text.txt

**s21900706@walab-HGU**:**~/2022OSS/lab7**$ git add text.txt

**s21900706@walab-HGU**:**~/2022OSS/lab7**$ git commit -m "C1”

**s21900706@walab-HGU**:**~/2022OSS/lab7**$ vim text.txt

**s21900706@walab-HGU**:**~/2022OSS/lab7**$ git commit -am "C2"

**s21900706@walab-HGU**:**~/2022OSS/lab7**$ vim text.txt

**s21900706@walab-HGU**:**~/2022OSS/lab7**$ git commit -am "C3"

**s21900706@walab-HGU**:**~/2022OSS/lab7**$ git checkout -b topicA

**s21900706@walab-HGU**:**~/2022OSS/lab7**$ vim text.txt

**s21900706@walab-HGU**:**~/2022OSS/lab7**$ git commit -am "C4"

**s21900706@walab-HGU**:**~/2022OSS/lab7**$ git checkout -

**s21900706@walab-HGU**:**~/2022OSS/lab7**$ vim text.txt

**s21900706@walab-HGU**:**~/2022OSS/lab7**$ git commit -am "C5"

**s21900706@walab-HGU**:**~/2022OSS/lab7**$ git checkout topicA

**s21900706@walab-HGU**:**~/2022OSS/lab7**$ vim text.txt

**s21900706@walab-HGU**:**~/2022OSS/lab7**$ git commit -am "C6"

**s21900706@walab-HGU**:**~/2022OSS/lab7**$ git checkout -

**s21900706@walab-HGU**:**~/2022OSS/lab7**$ git merge --no-ff topicA

**s21900706@walab-HGU**:**~/2022OSS/lab7**$ vim text.txt

**s21900706@walab-HGU**:**~/2022OSS/lab7**$ git add text.txt

**s21900706@walab-HGU**:**~/2022OSS/lab7**$ git commit -am "C7"

**s21900706@walab-HGU**:**~/2022OSS/lab7**$ git log --oneline --graph

Text

Description automatically generated with medium confidence

충돌이 예상되는 commit 번호는 C7에서 merge할 때이다. 충돌이 일어날 경우, 충돌이 일어난 파일을 다시 편집하여 다시 add commit을 하여 merge를 진행하였다.

vim a.txt후 파일 내용을 보면 충돌이 일어난 부분을 보여주는데 이 부분을 내가 다시 편집하여 저장한후 git add a.txt, git commit을 하면 merge 내용 이름을 정하는 편집기로 들어가게 된다. 이 편집기에서 commit 이름을 C7으로 하여 merge를 끝낼 수 있다.