1. Git에 대해 정리하자면

* 분산 버전(이력)관리 시스템
* 여러명이서 협업하는 경우에, 서로 하기로한 분량을 하는데 속도차이가 날 수 있는데, 작업의 분배와 병합, 협업을 용이하게 만들어주는 것
* 하나의 문서를 동시에 편집할 수도 있고, 각자의 내용을 기록으로 남길 수 있음
* 특히, 동일한 내용을 서로 다르게 수정했다면, 이를 자동으로 감지🡺어떤 내용을 최종으로 병합할 것인가? 와 같은 기능을 제공

1. Git 원격 저장소에 올리기
2. 깃은 클라우드 같은 원격 저장장소이다.
3. 내 로컬 저장소(내 노트북 등)에서 터미널을 이용해 바로 git 저장소에 업로드할 수 있다.
4. 몇가지 명령어만 입력하면 되는데
5. cd [디렉토리 위치] - 내가 올리고 싶은 소스가 있는 파일로 이동
6. git init - 깃 오픈, 지금 이 디렉토리를 내 로컬 Git 저장소라고 알려주는 것 🡺 현재 디렉토리에 .git 이라는 파일이 생성됨
7. git add . / git add [소스파일이름.확장자] – 올릴 파일 add ( “.” 은 모든 것을 의미)
8. git commit -m “[커밋하면서 메모할 내용]”

예를 들어, git commit “first commit” 🡪 first commit이라는 이름의 메모와 함께 로드됨

1. git remote add origin <https://github.com/lgaeun/> [업로드할 repository 이름].git

🡪 한마디로, git repository와 내 로컬저장소를 연결해주는 길을 내는 것

1. git push -u origin master: 내 로컬 저장소(master)에 commit된 내용을 git remote repository중 origin으로 옮겨줘!!

Repository마다 이름 정할 수 있음 (즉 a.git 에도 origin, b.git에도 origin 가능)

+ 주의: cd 폴더 이름에 한글ok, but 띄어쓰기 no!!

+ 커밋 = 이력 저장

1. Git에서 내 컴터로 가져오기

* git clone (저장소 가져오기)

# clone을 하게 되면 git init까지 자동으로 됨

<GitHub의 특징>

* 저장소에 팀원초대는 무제한
* 개인 저장소 private repository는 유료임
* pull -> pull request 형식임
* git clone (저장소 가져오기)

# clone을 하게 되면 git init까지 자동으로 됨

* git remote (저장소 url 설정하기) = git 원격저장소에 터미널로 폴더 추가하기!!

# .git으로 끝나는 URL>>이 저장소 이름과 함께 추가, 삭제, 수정 할 수 있는 기능임

git remote -v # 현재 연결되어있는 저장소 확인

git remote add <저장소 이름> <URL>.git # 원격 저장소 추가

git remote delete <저장소 이름> # 원격 저장소 삭제

git remote set -url 저장소 이름. <URL>.git # 원격 저장소 수정

rm -r .git # 실수로 git init 잘못했을 때,

만들어진 .git 폴더 삭제

git reset # commit 취소

- github remote 저장소에 잘못올라간 파일 삭제하기

// 원격 저장소와 로컬 저장소에 있는 폴더를 삭제한다.

$ git rm [Folder Name] e.g. git rm .VisualMangerfile

// 원격 저장소와 로컬 저장소에 있는 폴더를 삭제한다.

$ git rm [File Name] e.g. git rm VisualManger.swift

// 원격 저장소에 있는 파일을 삭제한다. 로컬 저장소에 있는 파일은 삭제하지 않는다.

$ git rm --cached [File Name]

// .idea/modules.xml 파일 삭제

$ git rm --cached .idea/modules.xml

// .idea 폴더 하위의 모든 파일 삭제

$ git rm --cached -r .idea/

* git status (현재 저장소에 있는 내용과, 로컬 컴퓨터에 있는 내용이 얼마나 다른지 알려줌)

\*\*오류뜨거나, 이 버전의 상태를 확인할 때 매우 유용한 깃 명령어임!!”

git status

* git add+commit (수정 사항 확정하기)

git. add <파일 이름.확장자> # 수정된 파일을 저장소에 올릴 준비(수정사항을 add하겠다)

git add . # 수정된 전체 파일을 저장소에 올릴 준비

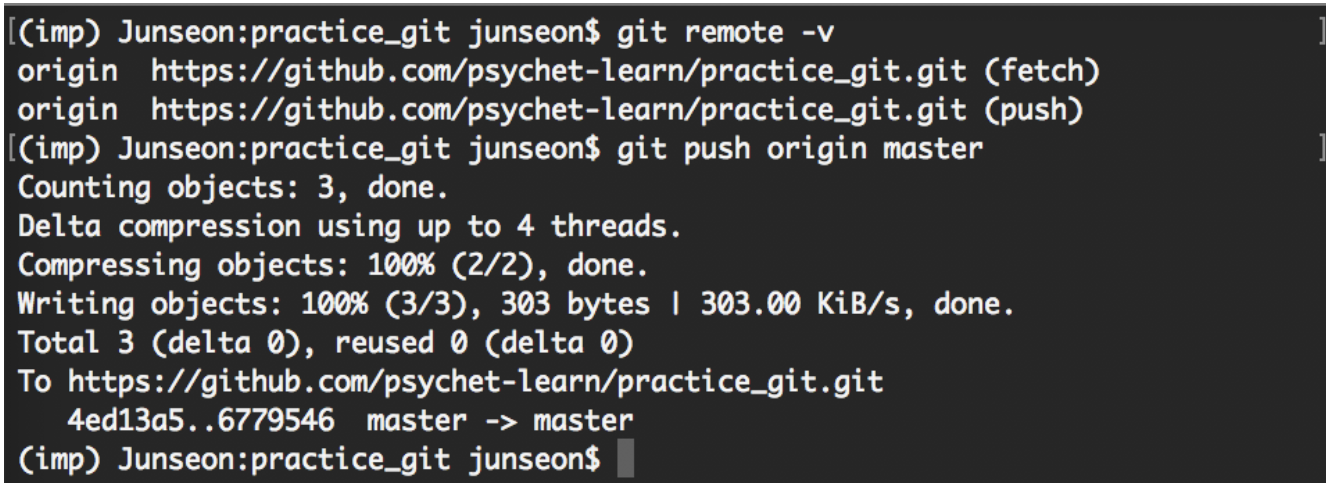
git commit -m “커밋 내용” # add한 수정된 파일을 저장소에 올릴 것을 확정

git commit -a # 편집기로 커밋할 메세지 적기 가능

* 즉, commit하면 수정사항을 뒤에 덧붙이기 까지만. 저장소엔 반영X, 로컬 컴터에만 확정
* 실제 저장소에 반영하려면 push를 해야함
* git push

git remote -v # 저장소의 url 확인 -> 그 이름을 통해서

git push <저장소 이름> master # 변경사항을 깃 저장소에 반영

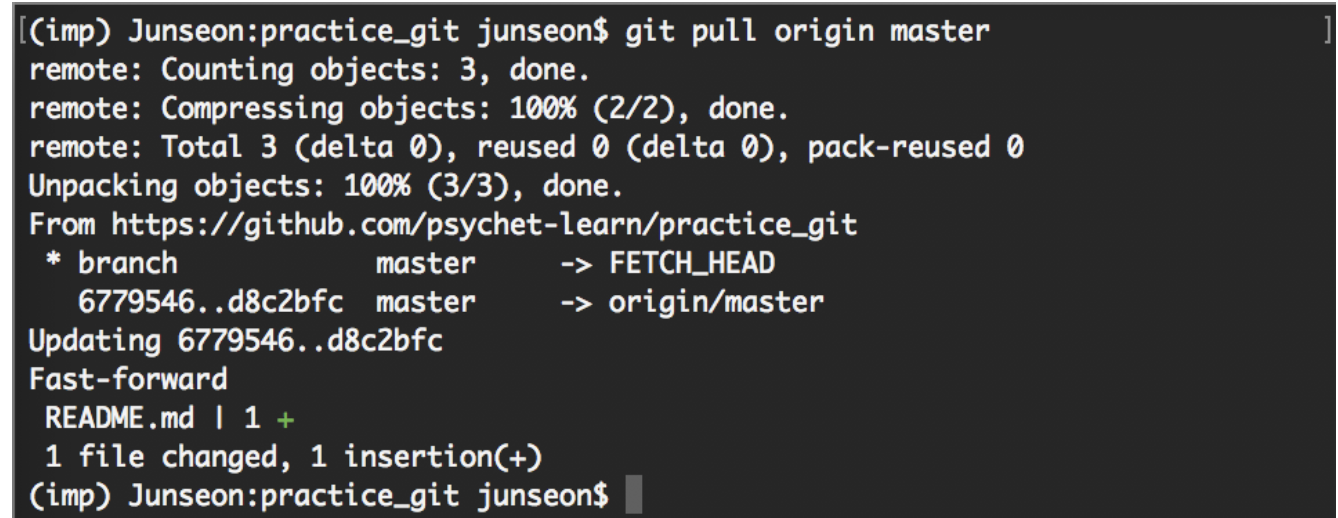


* git pull (수정된 저장소 내용 불러오기 – 수정된 내용만 받는 것)

한줄이나 조금 수정했을 경우, 다시 clone해서 전체 파일 다 받는 건 오래걸림

수정된 부분만 받는 명령어 사용 더 효율적!

git pull <저장소 이름> master



초록색 + 🡪 어떤 파일이 얼마나 최신화되었는디 알려주는 것

1. CLI 기본개념 및 git에서의 CLI

* Command line interface 또는 명령어 인터페이스, 즉 사용자가 텍스트 터미널을 통해 컴퓨터와 상호작용하는 방식
* 거의 모든 운영체제에서 GUI를 지원하지만, 여전히 많이 커맨드 라인을 사용함
* 윈도우에서는 cmd 창 (cmd.exe 실행), Mac OS에서는 Terminal창 (terminal.app 실행)을 사용하게 됨

<CLI 작동방식>

* ‘입력’과 ‘출력’ 크게 2개 단위로 구성

[입력] – 명령어를 기다리는 부분은?

Window에서는 명령 프롬프트 “>”

Mac os, linux에서는 “$

+ 보통 엔터키를 입력하여 명령 종료

* 터미널(terminal): 입출력이 가능하게 해주는 sw 또는 hw
* 셸(shell): 사용자가 입력한 명령어를 해석해주는 소프트웨어
* 자주 사용되는 명령어

1. ls : 현재 directory 폴더 및 파일 확인 / 윈도우에서는 dir

ls -a #숨김파일까지 확인 가능

1. cd: directory이동

$ cd <현재 디렉토리의 폴더 이름> #그 폴더로 이동

$ cd <디렉토리> # 특정 디렉토리로 이동

$ cd ../ # 상위 디렉토리로 이동

$ cd ~ # 루트 디렉토리로 이동

1. 폴더 관리
2. 파일 관리

<Git에서 CLI로 명령내리기>

* 폴더(디렉토리) 생성하기

Mkdir <폴더 이름> # ex. mkdir Kathy

mkdir <디렉토리> # ex. mkdir ~/Documents/gaeun

* Git 시작하기(초기화하ㅣ기)  
  git init # “Initialized empty Git repository in <디렉토리 패스>” 라는 출력이 나옴

= git을 사용할 수 있는 디렉토리가 되었다는 의미

* Git 프로젝트 가져오기

+ ‘git’ 명령어에 추가로 ‘clone’을 입력하고 그 저장소의 URL정보를 통해 그 저장소에 있는 프로젝트롤 가져올 수 있음

git clone <URL.git> # URL 뒤에 반드시 .git이 있어야 된다!!!

git clone https://github.com/어쩌구저쩌구~~

+ “cloning into <저장소 이름>”이 출력되는데, 좀 이따 디렉토리를 확인하면, 디렉토리 하나더 생긴걸 볼 수 잇음. 그게 git 저장소의 내용을 가져온 디렉토리임