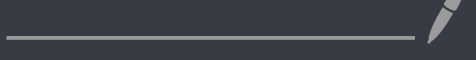


Git이란?

→ 버전관리 시스템.

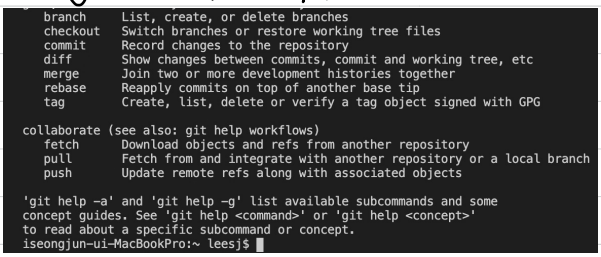
→ 파일 한개에서 히스토리를 남겨두어 이전에 작성/저장 했던 버전으로 돌아갈 수 있고 분기점을 만들어서 여러가지 방법으로 문서를 편집, 비교 + 최종버전만 두고 삭제까지도 가능.

a MAC에 git 설치

① VSCode 설치

② terminal → new terminal

③ git 입력 후 설치여부 확인.



↳ 설치 된 상태.

④ 설치가 안되어 있다면.

```
### 명령어를 복사할때 가장 왼쪽의 $를 빼고 복사해서 입력해주세요.
## mac용 패키지 관리자인 homebrew 를 설치합니다. (잠시동안 멈추는것은 다운로드되고있는것입니다.)
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/
install/master/install.sh)"

## brew update 명령어로 brew 를 업데이트 합니다. (잠시동안 멈추는것은 업데이트진행중입니다.)
$ brew update

## brew 를 이용해서 git을 설치합니다.
$ brew install git

## install 이 끝난후 해당명령어를 입력후 바로 아래줄과 같이 버전정보가 나온다면 설치성공
$ git --version
git version 2.21.0 (Apple Git-122.2)
```

a Windows에 깃 설치

①Git-Scm 사이트 접속 & Git 설치

설치할 구성요소를 선택 아래를 참고해 주세요.

1. Additional icons - On the Desktop
   - 윈도우 바탕화면 Git 아이콘 추가

default

2. Windows Exporer integration - Git Bash Here 과 Git GUI Here
   - 마우스 오른쪽 메뉴에 *Git Bash Here 과 Git GUI Here가 추가

3. Git LFS (Large File Support)
   - 큰파일 지원

4. Associate .git configuration files with the default text editor
   - git 구성파일을 기본 텍스트 편집기와 연결

5. Associate .sh files to be run with Bash
   - 확장자.sh 파일을 Bash와 연결

6. Use a TrueType font in all console windows
   - 윈도우 콘솔창에서 올바른 글꼴 사용 (체크시 한글이 깨진다고 함)

7. Check daily for Git for Windows updates
   - 윈도우즈용 Git 업데이트 매일 확인

② VSCode 설치

◦ git 기초설정

```
## 해당명령어의 NAME 에 깃헙 닉네임 설정
$ git config --global user.name "NAME"

## 해당 명령어의 EMAIL 에 깃헙 계정 메일 설정
$ git config --global user.email EMAIL
```

◦ Git 사용.

① vscode 실행

② File → open → 원하는 위치에 폴더생성 및 선택
terminal 실행, pwd 입력 → 현재 파일 위치 출력.

```
$ pwd
/Users/leesj/Desktop/for8th

## init을 통해서 git을 초기화(initialize) 합니다.
$ git init
Initialized empty Git repository in /Users/leesj/Desktop/for8th/.git/

## .md 파일(markdown 확장자)을 한개 만듭니다.
### 파일이 만들어지면 왼쪽에 파일이 만들어지고 이름에 초록색이 보이며 U 표시가 나타납니다.
$ touch README.md

## git 의 로그를 확인해보겠습니다.
$ git log
fatal: your current branch 'master' does not have any commits yet ## 간단히 해석하자면 아무것도 없다고 하네요.

## 현재 git의 상태를 확인합니다.
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        README.md

nothing added to commit but untracked files present (use "git add" to track)
## 위와 같은 상태를 간단히 해석한다면 'README.md 파일이 생성되었고 git을 통해 해당 파일을 관리하고자 한다면 git add 명령어를 사용해주세요.' 입니다.

## 설명을 그대로 따라보도록 하겠습니다. add를 실행하고 바로 상태를 확인해보겠습니다.
$ git add README.md
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   README.md
## git에 의해서 new file이 추적되었다는 뜻이지만 이것이 commit 되지 않았다고 합니다.

## 이를 쉽게 설명하자면 변경사항이 확인되었으나 현재 상태가 저장되지 않았다는 뜻입니다. 바로 다음 과정까지 진행하고 연달아서 상태확인!
$ git commit -m "REAMDE.md 파일생성"
[master (root-commit) e14e935] REAMDE.md 파일생성
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.md

$ git status
On branch master
nothing to commit, working tree clean
## commit 명령어로 파일 변경을 저장함과 이후 추가로 저장할 변경사항이 없는것을 확인할 수 있습니다.

## 그렇다면 정상적으로 저장되었는지 log를 확인해보죠
$ git log
commit '숫자와 영어가 포함된 난수' (HEAD -> master)
Author: '계정명과 이메일'
Date:   '날짜'

    REAMDE.md 파일생성
## 정상적으로 저장된것을 확인할 수 있습니다.
```

github 사용.

→ ① repository 생성

② 코드 업로드 → vscode에서 git remote add origin github repo 주소

git push -u origin master

처음 입력 이후로는 git push로 단축. (같은 repo)

git add 파일

git commit -m "메세지"

git reset HEAD^ (커밋 한개 삭제)

git reset HEAD^2 (커밋 두개 삭제)

git remote rm origin → remote origin 삭제.

git push -f origin "브랜치 명"

→ 원격 저장소 커밋 내역 갱신.

깃허브 흐름.

웹 1주차 pdf 정리.

- 웹서비스 → 웹을 통해 유용한 (서비스)제공.
- HTML & CSS != 프로그래밍 언어 ⇒ 마크업언어
- Uniform Resource Location, URL → 자원의 위치를 전달하기 위한 기본방식.

Client :// Server address / Page Address

http :// www.facebook.com / likelion.net.

- http
  - 프로토콜
  - 데이터 교환 및 전송 위한 방법의 세트.
  - (사용자가 요청하고 서버가 응답할 때 사용하는 언어)
  - http:// → 보안X
  - https:// → 보안 O (암호화하여 전송)
- www.facebook.com
  - 도메인 이름.
  - (페이지 혹은 이미지를 브라우저나 봇이 위한 서버의 주소)
  - 표준 HTTP가 아니면, 포트번호 표기.
  - .com :80
    - URL접근위한 관문 (gate)
- likelion.net.
  - 페이지 혹은 이미지의 주소

Git → 분산버전 관리 시스템

특징 ① Backup (소스코드 백업)

② Recovery (옛 버전으로 복귀)

③ Collaboration (협업)

④ Snapshot (파일의 상태 저장, 버전의 저장 단위)

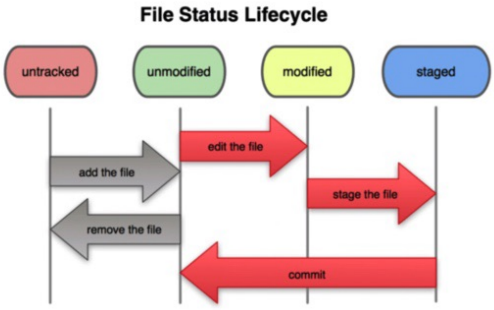
작업공간 → 스테이지 영역 (index) → 로컬저장소 → 원격 저장소.

git 실행흐름 ① git init → git 저장소 초기화.

- 워킹 디렉터리 : 소스코드 작업하는 공간
- 스테이징 영역 : 워킹 디렉터리에 git add실행하면 파일들은 git의 스테이징 영역으로 이동. → 소스코드의 상태정보 관리
- 저장소 영역 : git commit 실행시 최종적으로 git 저장소에 반영

* 파일관점

파일 관점에서 Git은 다시 4가지 단계로 나눌 수 있다.



① untracked : 워킹 디렉터리에 추가되었지만 git에서 관리X
② Unmodified : add 된 상태지만 수정X
③ modified : 수정됨
④ staged : 스테이징 영역에 반영

② git add . → 스테이지 영역으로 파일 추가 (add . → 파일전체 추가)

- git rm --cached staged.txt → 스테이징 파일 삭제. (untracked 영역으로감)
-

③ git commit -m "message" → 스냅샷 찍기.

* 스냅샷이란? 특정 시점에서 파일, 폴더 또는 워크스페이스의 상태를 의미한다. 스냅샷을 통해 특정 시점에 어떤 파일이 어떤 내용이 기록되어 있었는지, 폴더 구조는 어떠 했는지, 어떤 파일이 존재했는지 등 저장소의 모든 정보를 확인할 수 있다. → commit을 통해 메세지와 함께 스냅샷 찍음.

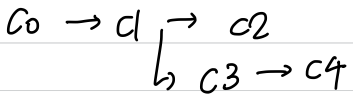
④ git remote add origin "github 레포 URL" → 로컬 저장소를 원격저장소로 연결.

⑤ git push -u origin master
- 원격 저장소 (github)에 변경사항 업로드

---

a Branch ?
- 독립적으로 어떤 작업을 진행할 수 있음.
- 즉 구성원들이 자유롭게 수정가능

→ git branch 브랜치 이름. - 새 브랜치 만들기



Branch 작업.
- 원격 저장소에 반영 git push github 브랜치명
- 브랜치 전환 git checkout 브랜치명 or master // note!! git checkout -b 브랜치 → 생성및 전환 동시에 실행.
- 브랜치 합치기 git merge 브랜치명
- 다른 브랜치를 원격브랜치로 Push git push <remote_name> <branch_name> → git push origin master.

브랜치 삭제 : git branch -d <branch name>

git 명령어 추가 정리

① cat .git/config → 원격저장소 연결여부 확인

② git remote add 약속어 깃헙주소

origin은 관례적 사용

③ git push origin master → origin으로 push

주의 ④ rm -rf .git (.git폴더 삭제를 통해 commit이력 및 전체삭제)

⑤ remote으로 설정된 저장소의 주소를 바꾸고 싶을 때. (link 변경)

① Vi .git/config

→② i 입력후 입력모드 진입 ③ 편집을 원하는 장소로 커서이동

④ 편집후 esc ⑤ 입력모드탈출 후 :wq 입력 (enter)

⑥ git rebase → 커밋 수정 삭제. > 나중에 확인.

⑦ git cherry pick. → 커밋 브랜치 이동 / 마스터 삭제

대신 / 마스터로 checkout 후

⇒ 선택커밋들만 마스터에.

↓

나중에 확인