멋쟁이사자처럼 
X
Hongik Univ 11th.

LIKELION UNIV.

Git & Github

Git & Github 란?

Git : 사용자가 파일을 로컬, 원격 저장소에서 관리할 수 있도록 도와주는 버전관리 시스템

Github : Git 시스템을 사용하여 파일들을 관리해주는 원격저장소

즉, Git으로 본인의 프로그램 버전관리를 하고, 개개인의 버전을 타인과 공유하여 관리할 수 있게 해주는 원격 클라우드 시스템이 Github!

📌 Git & Github를 왜 사용할까?

→ 버전관리와 협업을 위해 !

→ 개발자들의 필수 TOOL로 자리매김하고 있음

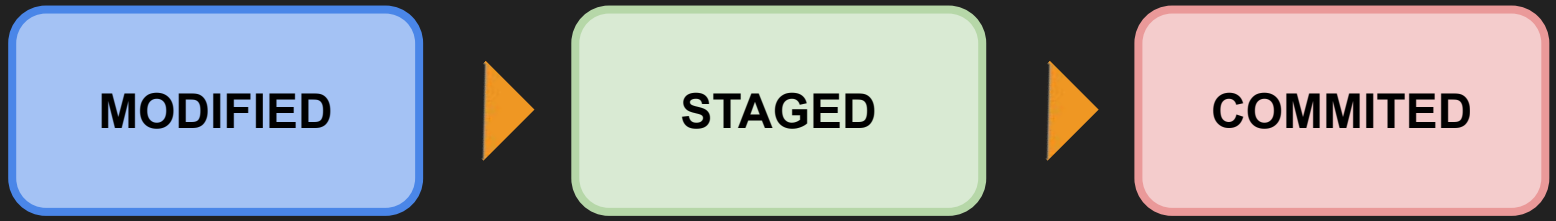
Git의 사용성

(1) 내가 작성한 파일 및 디렉토리를 저장해 놓을 수 있음

(2) 원하지않는 실수가 생겼을 때, 저장해놓은 버전 중 하나로 롤백이 가능함

(3) 여러명과 협업 시 깃을 사용하여 프로젝트 개발을 동시에 진행할 수 있음

GIT 상태



→ 파일이 수정중인 상태
→ 자신의 로컬 환경에서 파일을 저장하는 행위
→ 소스코드의 수정, 작성 등의 모든 작업은 로컬에서 이루어짐

* 로컬과 원격의 개념 숙지할것!!

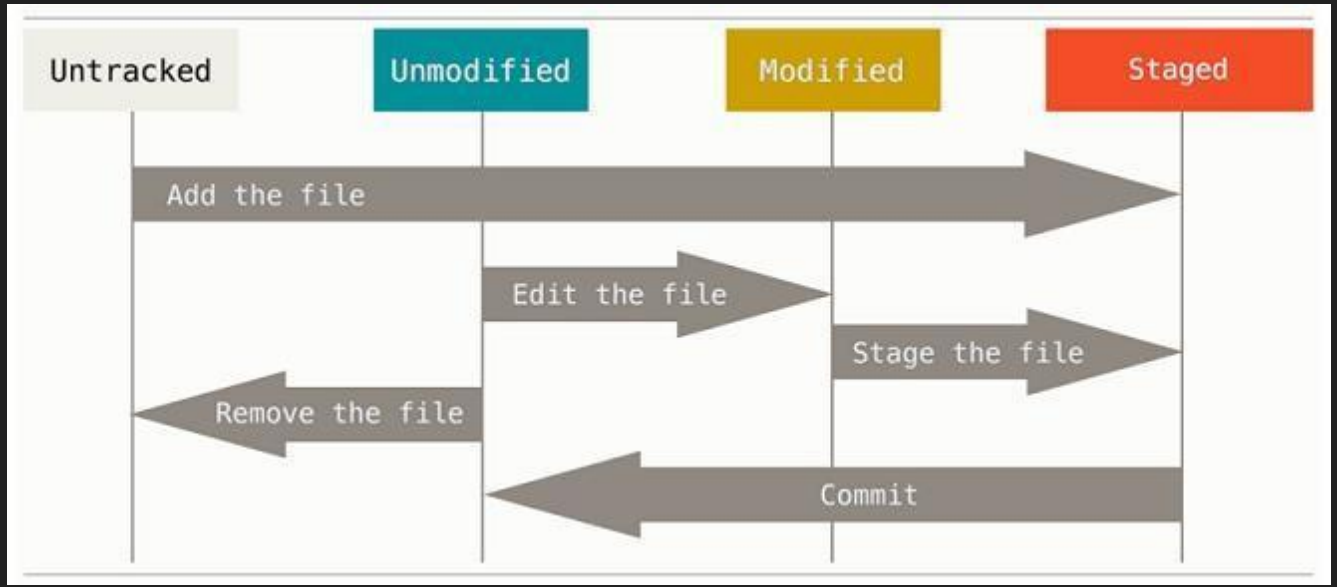
→ commit 되기 이전의 상태
→ 파일 최종 저장 전에 중간 저장지점이라고 생각하면 편함!

명령어
→ git add .
(현재까지의 모든 작업물 중간저장하기)

→ 자신의 수정사항을 하나의 버전으로 만드는 역할을 함

명령어
→ git commit -m “”
(“”안에 커밋메시지, 즉 부가설명 추가

GIT의 상태 변화


Untracked
Unmodified
Modified
Staged
Add the file
Edit the file
Stage the file
Remove the file
Commit

주요 깃 명령어

- git pull origin [원격브랜치명] : 원격 특정 브랜치에서 로컬로 가져오기
- git push origin [원격브랜치명] : 현재 내 로컬 브랜치에 있는 코드를 원격 특정 브랜치에 옮기기
- git log : 지금까지 현재 브랜치에서 일어난 일, 즉 로그 보기
- git branch : 현재 브랜치 및 브랜치의 목록 확인
- git checkout [브랜치명] : 현재 브랜치 옮기기
- git status : 현재 내 깃의 상태를 확인하는 것

  → 수시로 체크하기.

GITHUB 사용법

깃허브는 여러명이 공유하는 원격저장소이며, 다음과 같은 절차로 프로젝트 진행을 함

1. Repository를 생성하여 팀원들과 협업할 온라인 깃허브 조직 생성
2. 팀원이 작성한 원격코드를 내 로컬로 가져옴
3. 내 로컬에서 작성한 코드를 원격 깃허브로 전송

Github Repository

: Git으로 관리하는 프로젝트 저장소로, 다음과 같이 두종류로 나뉨

- Local repository : 로컬이란 단어 말 그대로 개발자 컴퓨터에 저장된 로컬 버전의 프로젝트 저장소

  → git init 이란 명령어를 통해 프로젝트 관리 시작 가능

- Remote repository : 원격 프로젝트 저장소이며 보통 github 사용

중요

→ 원격과 로컬 레포지토리의 브랜치명을 헷갈리지 않도록 주의!

→ 원격의 main 과 로컬의 main은 다른 브랜치임을 유의하자.

→ 하지만 같은 이름의 브랜치가 보통 원격,로컬 모두 연동시켜 레포지토리 내 동일 역할 수행하도록 함.

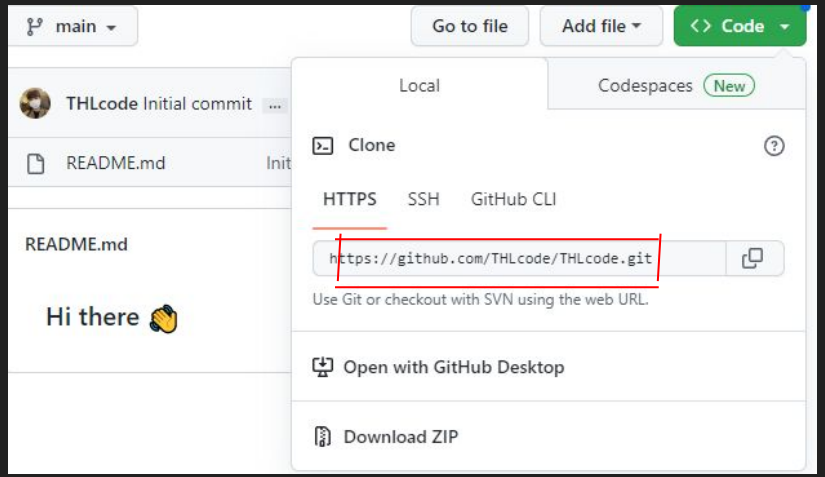
로컬 레포지토리 생성순서

1) git init

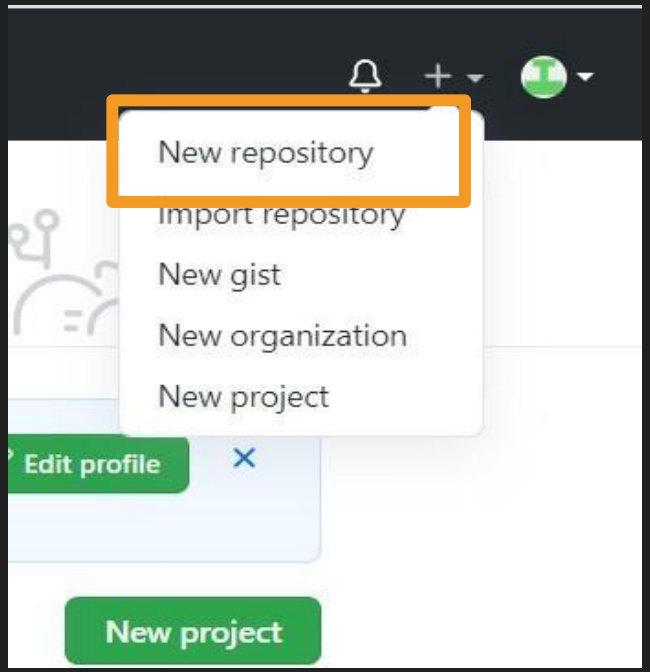
: 로컬 터미널에서 작성하며, 로컬 레포지토리 생성함

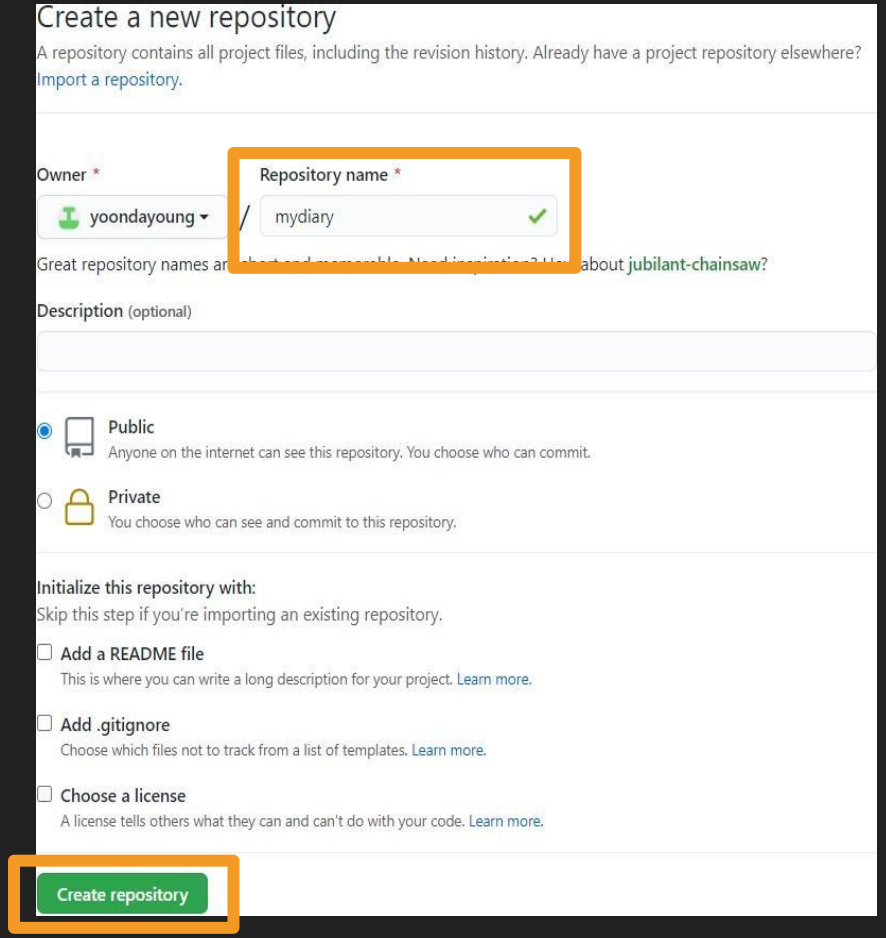
2) git remote add origin “”

: 원격 깃허브 주소를 복사해와서 “” 안에 넣으면 되며, 원격 Repository와 내 로컬 repository를 연결해줌



Github repository 생성하기





원격 -> 로컬

- git pull origin master (전 슬라이드에서 origin에 등록해둔 repository의 master 브랜치에서 현재 내 브랜치로 pull 받아오기)
- 처음 git init을 하고나선 git clone “” 명령어를 통해 레포지토리의 master 브랜치를 통채로 복사해서 가져올 수 있음

로컬 -> 원격

- 로컬에서 작업이 끝난 후, 원격으로 코드를 옮기는 작업을 수행해야함

```
git add .
git commit -m "원하는 커밋 메시지"
git push origin [원격 브랜치]
```

→ 위 사진의 명령어 순서대로 작성하여 내 코드를 옮길 수 있음

Convention

→ 프로젝트, 혹은 어떠한 서비스 마다 “convention”을 세워서 작업을 함

- Convention : 어떠한 조직내의 약속

→ 보통 변수명, 혹은 commit message등 자주 쓰이는 곳에서 정함

ex) 변수명은 대문자로 시작하고, 합성어는 _ 로 구분하자

ex) commit message는 커밋형태에따라 Fix(오류해결) dev(개발사항) 을 붙여서 메시지로 커밋의 의미가 유추가능하게하자

→ 이러한 컨벤션을 정해놓는 것이 프로젝트를 원활히 돌아가게 도와줌

Branch convention

1) Main : 서비스 배포된 버전이 담겨있는 브랜치

   → main (or master) 브랜치는 안건드는게 상책이다

2) Dev : 개발이 이루어지는 브랜치

   → 모든 push & pull은 이곳에서 이뤄지는것이 바람직함

3) Release : 사내 or 프로젝트 내에서 테스트서버 배포 브랜치

   → dev 개발 완료 후 테스팅을 릴리즈 브랜치에서 배포한 서버로 진행

   → 메인 서버에 문제 없이 테스트할 수 있음

4) Hotfix

   → 버전 상관없이 급하게 고쳐야할 버그를 고치는 브랜치

GITHUB로 협업하기

- 동시에 여러명의 인원이 프로젝트 내 여러 다른 코드를 작성한다면?

→ 문제가 없음 ! 서로 충돌날 일이 없기 때문!

Q) 같은 코드를 여러명이 작성해서 푸시한다면?

→ 여러명이 쓴 코드가 충돌되어 오류가 발생함!

→ 이러한 상황이 프로젝트 진행시 무조건 생김.

따라서, merge(여러명의 코드를 합치는 과정)이 매우 중요해짐!

GITHUB로 협업하기

Q) 충돌을 방지하기 위해선?

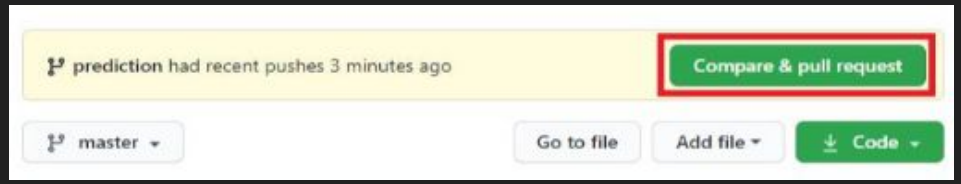
1. 조직화 및 추상화가 우선적으로 실행되어야함.

→ 서버마다, 혹은 프론트 & 백엔드를 같은 repository를 사용하는 것이 아닌, 다른 repository를 사용하여 프로젝트의 분리 및 세분화를 강조할것

2. 현재 내 브랜치를 수시로 확인할것
3. 프로젝트 내 머지하는 방식을 통일시킬것
    a. 원격에서 merge 수행
    b. 로컬에서 merge 수행

GITHUB로 협업하기 - 원격 merge

- PR (Pull Request)를 날리는 방법
  - 로컬 브랜치에서 작성한 코드를 원격으로 push를 함
  - 이때, 원격의 브랜치에 자동으로 PR이 생성되며, push를 한 당사자를 제외한 다른 인원이 merge를 해주어야함



→ 이 버튼을 누르고 merge 가능한지 체크를 하고, merge conflict (충돌 발생)이 있다면 코드를 확인하여 merge를 해준다음 “create pull request” 버튼을 누르고 다른 팀원이 merge pull request해주길 기다린다

- 장점 : 다른 인원의 코드 체킹을 통해 오류 방지
- 단점 : 자체적으로 머지하여 프로젝트 진행하기 어려움

GITHUB로 협업하기 - 로컬 merge

```
//현재 내 브랜치 : dev
//현재 원격 브랜치 : dev
//git status : 모든 작업 커밋된 상태
git pull origin dev
//merge conflict 발생했다면 로컬 vscode에서 보고 판단하여 merge
git add .    //merge하여 파일이 변경됐으므로 다시 수행
git commit -m ""
git push origin dev
```

→ 로컬에서 merge 하였기 때문에 pr과 관계없이 원격에 푸시 후 자동 머지되며 덮어씌워짐

→ 이 방법 택한다면 푸시 전 원격에서 풀 받고 머지하고 푸시하는 것이 필수!

실습

프론트 & 백엔드

- 자신의 깃허브 Repository를 생성하여 “Hello, LikeLion!”을 출력하는 코드를 깃허브에 올리기
- 그 과정에서 중요하다 생각하는 부분 사진 첨부 & 부연설명 멋사 홍익 홈페이지 각 트랙 게시판에 게시할것!

기획/디자인

- 깃허브아이디를 만들고 Repository를 생성하여 기획/디자인 트랙 아기사자들 서로 레포지토리에 추가해주기

홍대 멋사 홈페이지

과제

- 공통
- 각 트랙 게시판에 자기소개 올리기(이름, 학번, 나이, 학과, 사는곳, 취미, tmi 등)
- 팀별로 댓글 달아주기 (댓글 파워 기대해볼게요~~!!)

프론트 & 백엔드

- 실습 때 만든 Repository에 로컬 원격 모두 브랜치 1개 더 생성 후 main에 merge
- Push, merge하는 작업 정리해서 각 트랙 게시판에 정리한거 올리기

기획/디자인

- 자신이 생각하는 기획자의 정의 및 기획자가 요구되는 역량 공부 및 정리해서 기디 트랙 게시판에 작성하기