05_clone & pull

비고	
↗ 실습	끝말잇기

---

[1] 원격 저장소 가져오기

> 지금까지는 로컬 저장소의 내용을 원격 저장소에 업로드하는 것을 학습했습니다.
> 이번에는 반대로, 원격 저장소의 내용을 로컬 저장소로 가져오는 것을 학습합니다.

(1) git clone

- 원격 저장소의 커밋 내역을 모두 가져와서, 로컬 저장소를 생성하는 명령어
- clone은 `"복제"`라는 뜻으로, `git clone` 명령어를 사용하면 원격 저장소를 통째로 복제해서 내 컴퓨터에 옮길 수 있습니다.
- `git clone <원격 저장소 주소>`의 형태로 작성합니다.

```
$ git clone https://github.com/edukyle/TIL.git
Cloning into 'TIL'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

위에 작성한 대로 실행하면, `Github의 edukyle이라는 계정의 TIL 원격 저장소를 복제`하여 내 컴퓨터에 TIL이라는 이름의 로컬 저장소를 생성하게 됩니다.

- git clone을 통해 생성된 로컬 저장소는 `git init`과 `git remote add`가 이미 수행되어 있습니다.

(2) git pull

- 원격 저장소의 변경 사항을 가져와서, 로컬 저장소를 업데이트하는 명령어
- 로컬 저장소와 원격 저장소의 내용이 완전 일치하면 git pull을 해도 변화가 일어나지 않습니다.
- `git pull <저장소 이름> <브랜치 이름>`의 형태로 작성합니다.

```
$ git pull origin master
From https://github.com/edukyle/git-practice
 * branch            master     -> FETCH_HEAD
Updating 6570ecb..56809a9
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)


[풀이]
git 명령어를 사용할건데, origin이라는 원격 저장소의 master 브랜치의 내용을 가져온다(pull).
```

💡 **git clone vs git pull**

clone과 pull이 모두 원격 저장소로부터 가져오는 명령어라서 조금 혼동될 수 있습니다.

`git clone`은 git init처럼 처음에 한 번만 실행합니다. 즉 로컬 저장소를 만드는 역할이죠.
단, git init처럼 직접 로컬 저장소를 만드는 게 아니라, Github에서 저장소를 복제해서 내 컴퓨터에 똑같은 복제본을 만든다는 차이가 있습니다.

`git pull`은 git push처럼 로컬 저장소와 원격 저장소의 내용을 동기화하고 싶다면 언제든 사용합니다. 단, push는 로컬 저장소의 변경 내용을 원격 저장소에 반영하는 것이고, pull은 원격 저장소의 변경 내용을 로컬 저장소에 반영하는 것입니다. **즉 방향이 다릅니다!**

[2] 내 컴퓨터 ↔ Github(원격 저장소) ↔ 강의장 컴퓨터

> 두 개 이상의 로컬 저장소에서 하나의 원격 저장소에 접근하면 어떻게 될까요?
> 집과 강의장을 오가면서 `clone, push, pull` 하는 과정을 살펴보겠습니다.

(1) 규칙

- 수업 때는 두 개의 폴더를 `"내 컴퓨터"`와 `"강의장 컴퓨터"` 라고 가정합니다.
- 내 컴퓨터에 있는 로컬 저장소의 이름은 `TIL-home` 입니다.
- 강의장 컴퓨터에 있는 로컬 저장소의 이름은 `TIL-class` 입니다.
- Github에 있는 원격 저장소의 이름은 `TIL-remote` 입니다.

(2) 사전 세팅

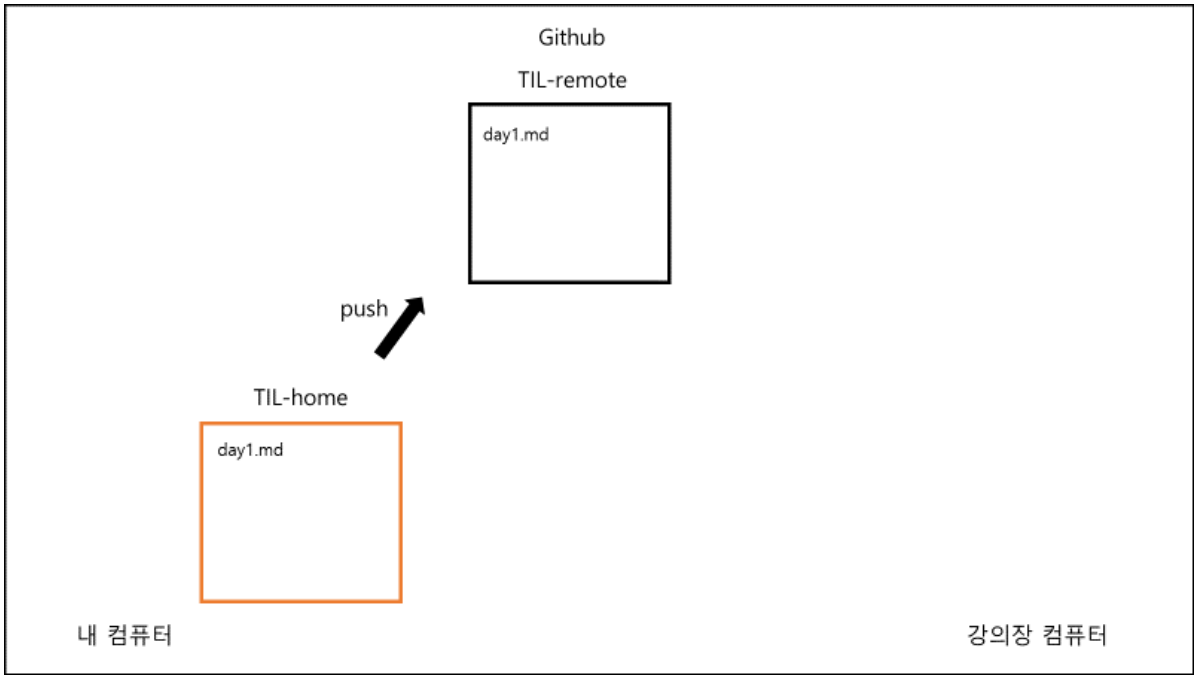
- 홈 디렉토리 안에 `TIL-home` 폴더를 생성합니다.
- Github에서 `TIL-remote` 라는 이름의 원격 저장소를 생성합니다.
- `TIL-home` 폴더에서 vscode를 엽니다.
- 아래와 같은 절차를 진행합니다.

```
# TIL-home

$ git init
$ touch day1.md
$ git add .
$ git commit -m "집에서 Day1 작성"
$ git remote add origin https://github.com/edukyle/TIL-remote.git
$ git push origin master
```

`TIL-home` 로컬 저장소의 내용이 `TIL-remote` 원격 저장소에 그대로 반영되었습니다.

- 결과



(3) git clone

> 여러분은 이제 강의장에 왔습니다. 강의장 컴퓨터에는 여러분의 TIL 폴더가 없습니다.

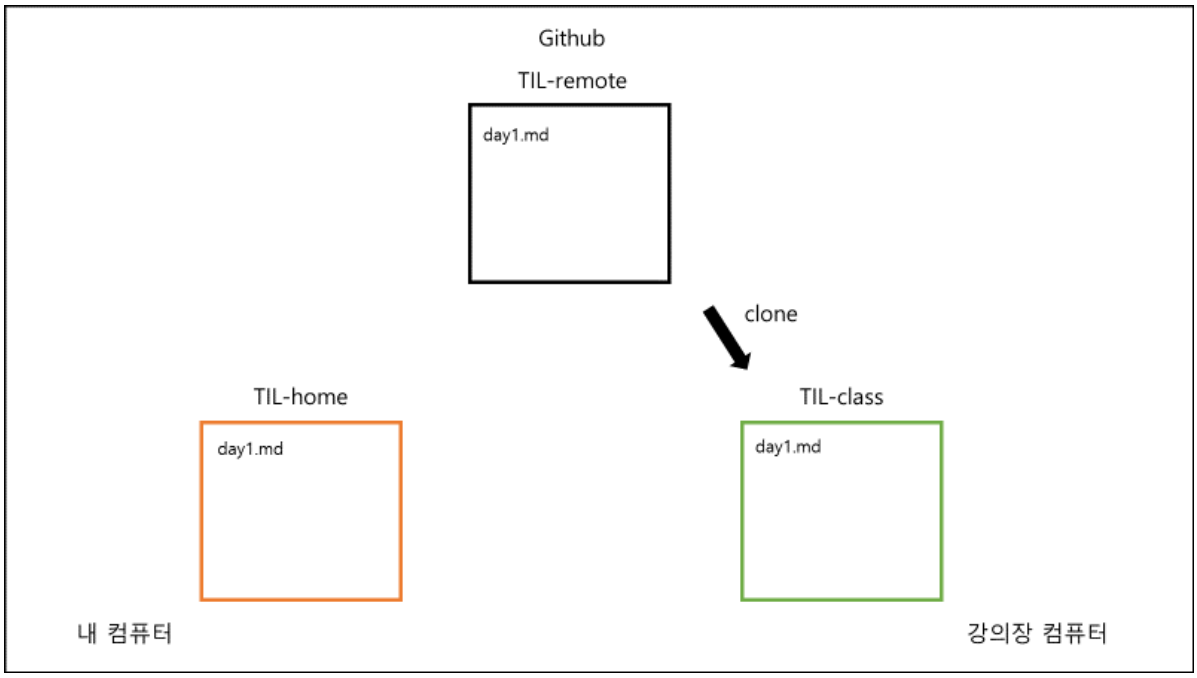
- Github에 있는 `TIL-remote`에서 `git clone`을 통해 내려 받습니다.

```
# TIL-class

$ git clone https://github.com/edukyle/TIL-remote.git TIL-class
```

원격 저장소는 `TIL-remote` 이지만, 위와 같이 작성하면 강의장 컴퓨터에는 `TIL-class`라는 이름으로 로컬 저장소가 생성됩니다. (내부 파일 내용은 똑같습니다. 단지 폴더의 이름만 바뀝니다.)

- 결과



(4) git push

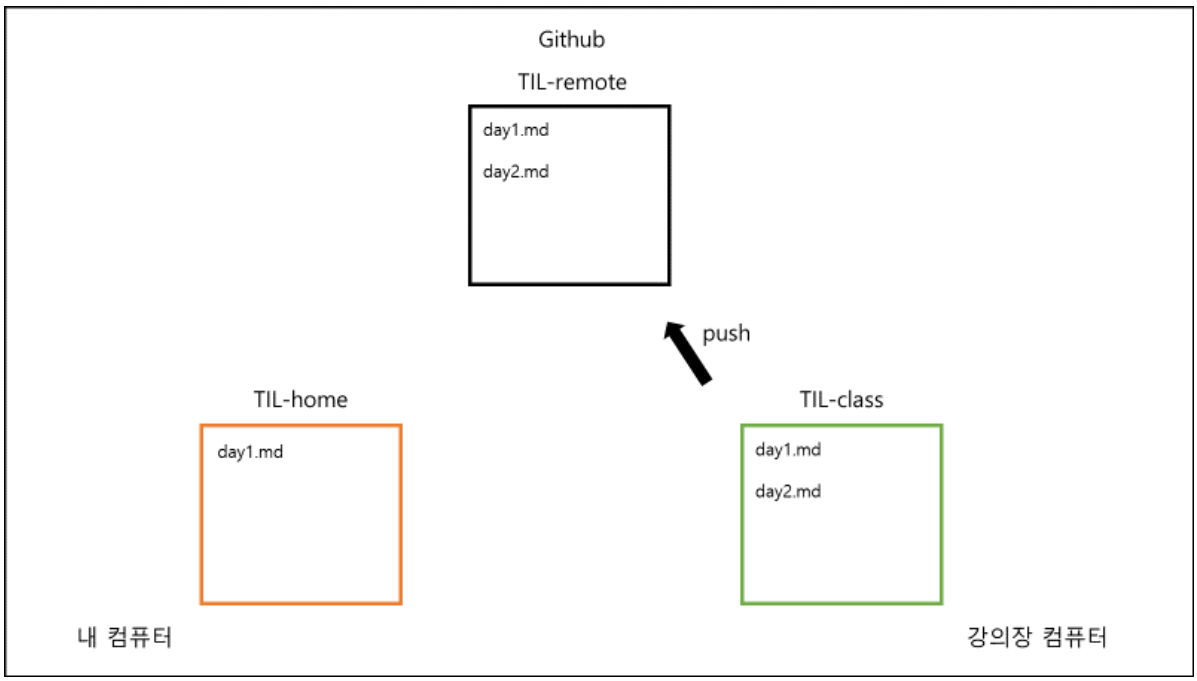
> 강의장 컴퓨터 → 원격 저장소

- 강의장에서 새로운 파일을 만들고 원격 저장소에 업로드 합니다.

```
# TIL-class

$ touch day2.md
$ git add .
$ git commit -m "강의장에서 Day2 작성"
$ git push origin master
```

- 결과



(5) git pull

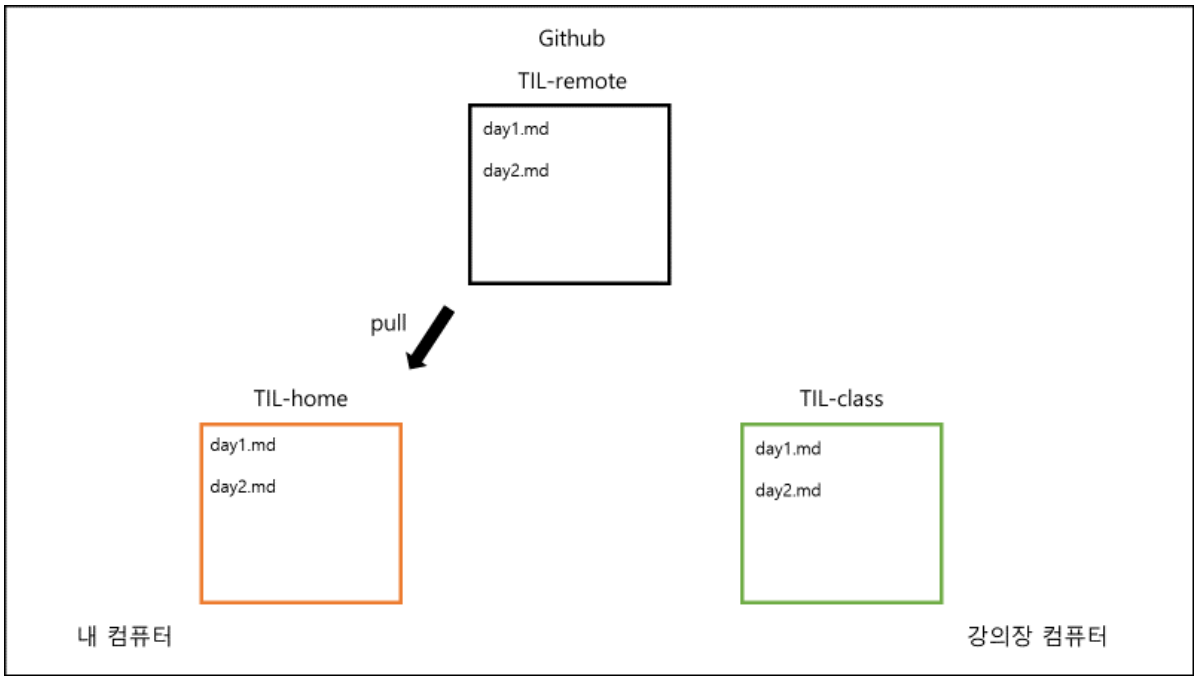
> 원격 저장소 → 내 컴퓨터

- 내 컴퓨터에는 day2.md가 없습니다. 왜냐하면 강의장 컴퓨터에서 day2.md를 만들어서 원격 저장소에 push 했기 때문입니다. 따라서 원격 저장소에서 day2.md에 대한 내역을 가져와야 합니다.

```
# TIL-home

$ git pull origin master
```

- 결과



이제 `내 컴퓨터, Github, 강의장 컴퓨터`의 내용은 동일합니다.

- **주의 사항 (글 만으로는 이해하기 어려우니, 직접 보여주면서 수업 합니다.)**

❗ **만약 TIL-home에서 pull이 아니라 commit을 먼저한 후 pull을 하면 어떻게 될까요?**
다음 세 가지의 경우가 있을 수 있습니다.

1. 내 컴퓨터와 강의장 컴퓨터에서 **서로 다른 파일을 수정**한 경우
→ 정상적으로 git pull이 됩니다.

2. 내 컴퓨터와 강의장 컴퓨터에서 **같은 파일을 수정했지만, 수정한 라인이 다른** 경우
→ 정상적으로 git pull이 됩니다.

3. 내 컴퓨터와 강의장 컴퓨터에서 **같은 파일의 같은 라인**을 수정한 경우
→ **충돌(conflict)**이 발생합니다. 어느 내용을 반영할지 직접 선택해야 합니다.

❗ **만약 TIL-home에서 pull이 아니라 commit을 먼저한 후 바로 push 하면 어떻게 될까요?**
아래와 같은 에러 메시지가 나타나면서 push가 실패합니다.

To https://github.com/edukyle/TIL-remote.git

! [rejected] master -> master (non-fast-forward)

error: failed to push some refs to 'https://github.com/edukyle/TIL-remote.git'

원격 저장소의 내용을 먼저 받아오지 않고, 로컬 저장소에서 새로운 커밋을 생성했기 때문에 서로의 커밋 내역이 달라져서 그렇습니다.

만약 로컬 저장소와 원격 저장소의 내용이 다르다면 일단 git pull을 통해 동기화를 시키고 새로운 커밋을 쌓아 나가야 합니다.