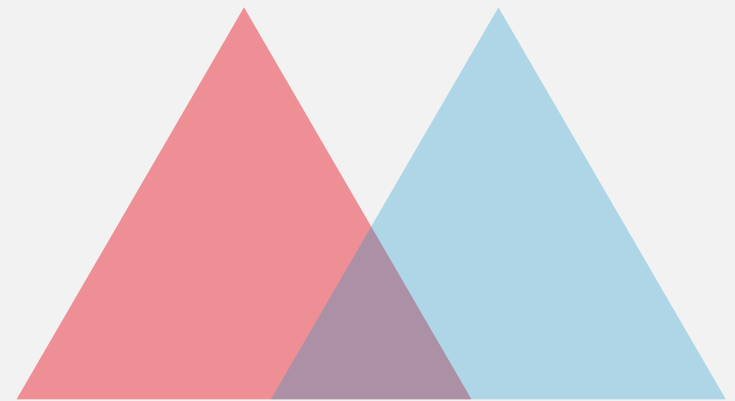


Git 사용법

2탄

멋쟁이 사자처럼 9기 운영진
김경은



Content

1 Clone

- Clone이란
- Clone 사용법

2 Branch

- Branch란
- Branch 사용법

3 Fetch/Merge

- 협업의 순서
- Fetch /merge 사용법
- Merge conflict

4 Pull

- Pull 사용법

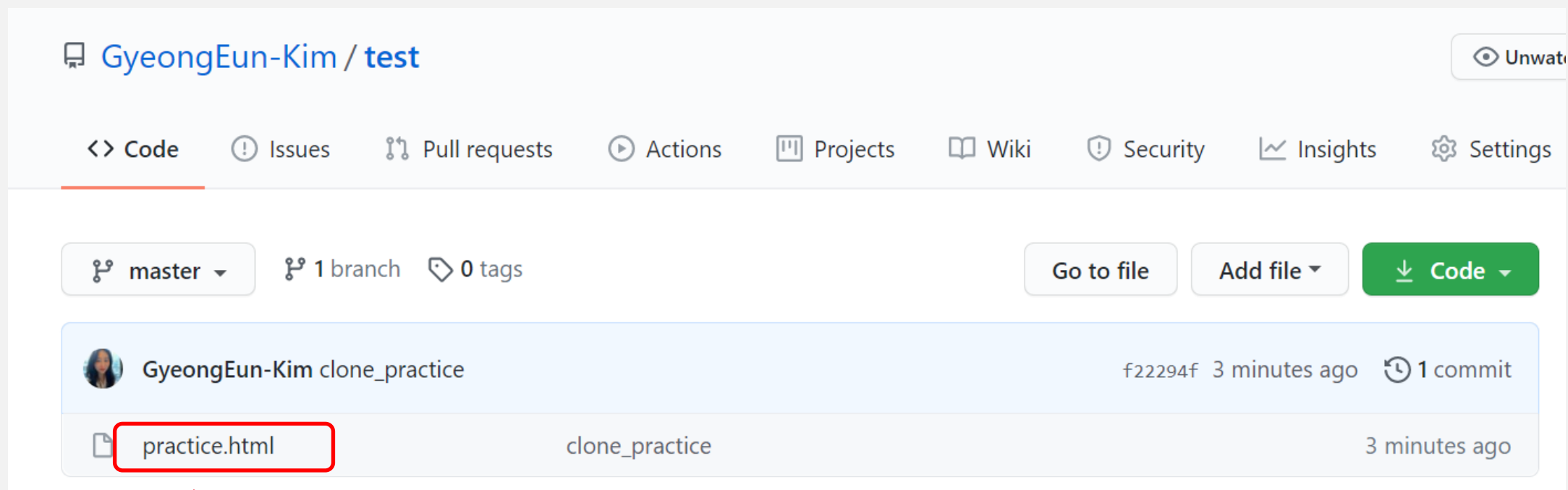
01

Clone



Clone이란?


Clone이란 복제한다는 뜻으로, 원격 저장소의 코드를 자신의 로컬 저장소로 복제해올 때 사용하는 명령어이다.



이 파일을 로컬 저장소로 가져가 보겠습니다

Clone 사용법

원하는 위치에서 마우스 우클릭을 이용해 git bash을 열고,
(혹은 cmd창에서 원하는 위치로 이동한 후, 혹은 Vscode에서 원하는 곳에서 터미널 창을 열고)
git clone [원격 레포지토리 주소.git]

 MINGW64:/c/Users/USER/Documents/멋쟁이사자처럼

```
USER@DESKTOP-OLUGOQQ MINGW64 ~/Documents/멋쟁이사자처럼
$ git clone https://github.com/GyeongEun-Kim/test.git
Cloning into 'test'...
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.

USER@DESKTOP-OLUGOQQ MINGW64 ~/Documents/멋쟁이사자처럼
$ |
```

Clone 완료

내 PC > 문서 > 멧쟁이사자처럼



test

내 PC > 문서 > 멧쟁이사자처럼 > test

이름

수정한 날짜

유형

크기

.git

2021-03-29 오후 6:38

파일 폴더

practice

2021-03-29 오후 6:38

Chrome HTML Docu...

1KB



파일 | C:/Users/USER/Documents/멧쟁이사자처럼/test/practice.html

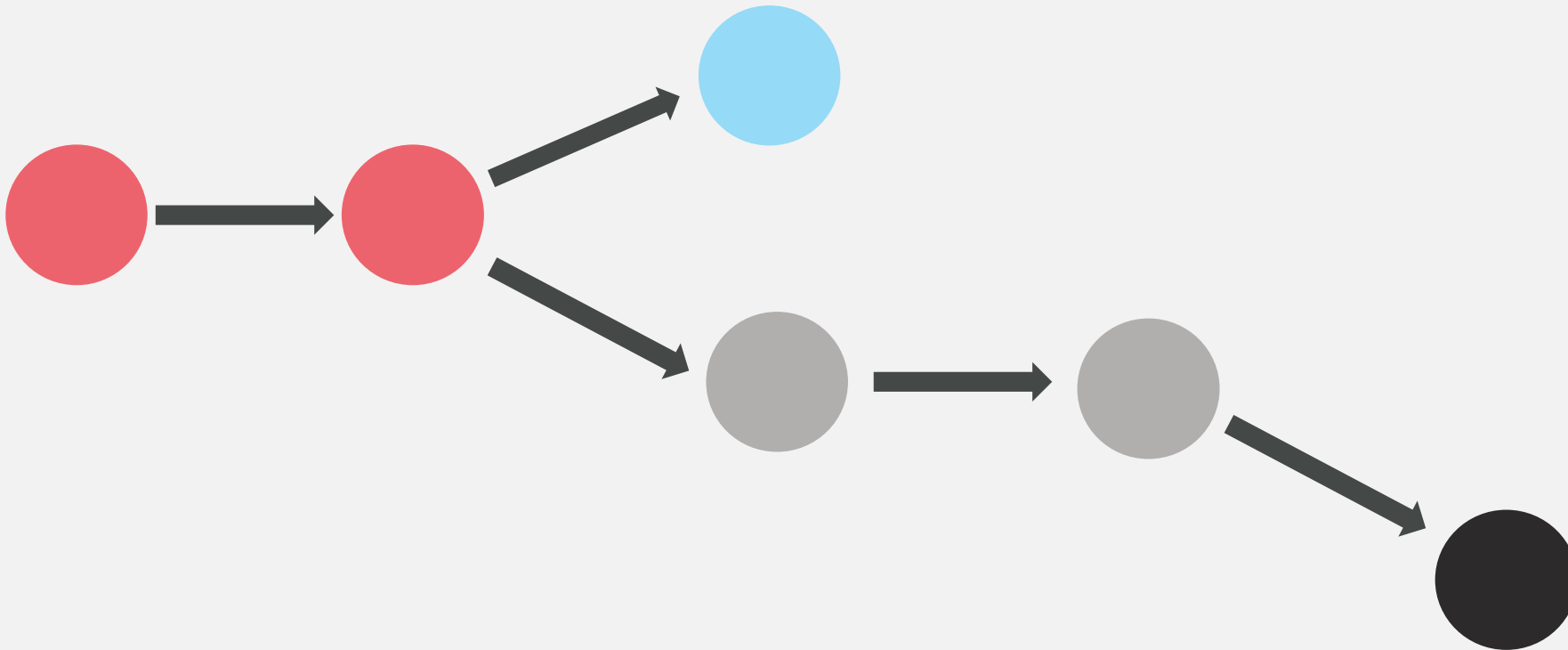
이것을 복사해오겠습니다

02

Branch



Branch란, 나무의 가지를 의미한다. 나뭇가지가 여러 갈래로 뻗어나가는 것처럼 Git에서도 마찬가지로 branch를 통하여 소스코드를 여러 버전으로 발전시키고 관리할 수 있다.



Branch 생성/조회

새로운 브랜치 만들기 : `git branch [새로운 브랜치 이름]`

로컬 브랜치 목록 조회 : `git branch`

원격 브랜치 목록 조회 : `git branch -r`

로컬 + 원격 브랜치 목록 조회 : `git branch -a`

```
USER@DESKTOP-OLUGOQQ MINGW64 ~/Documents/멋쟁이사자처럼/test (master)
$ git branch
* master

USER@DESKTOP-OLUGOQQ MINGW64 ~/Documents/멋쟁이사자처럼/test (master)
$ git branch -r
origin/HEAD -> origin/master
origin/master

USER@DESKTOP-OLUGOQQ MINGW64 ~/Documents/멋쟁이사자처럼/test (master)
$ git branch -a
* master
remotes/origin/HEAD -> origin/master
remotes/origin/master
```

현재 내가 위치한 브랜치

브랜치 삭제 : **git branch -d [삭제할 브랜치 이름]**

현재 브랜치에서 다른 브랜치로 이동 : **git checkout [이동할 브랜치 이름]**

```
USER@DESKTOP-0LUGOQQ MINGW64 ~/Documents/몇쟁이사자처럼/test (master)
$ git branch my_branch

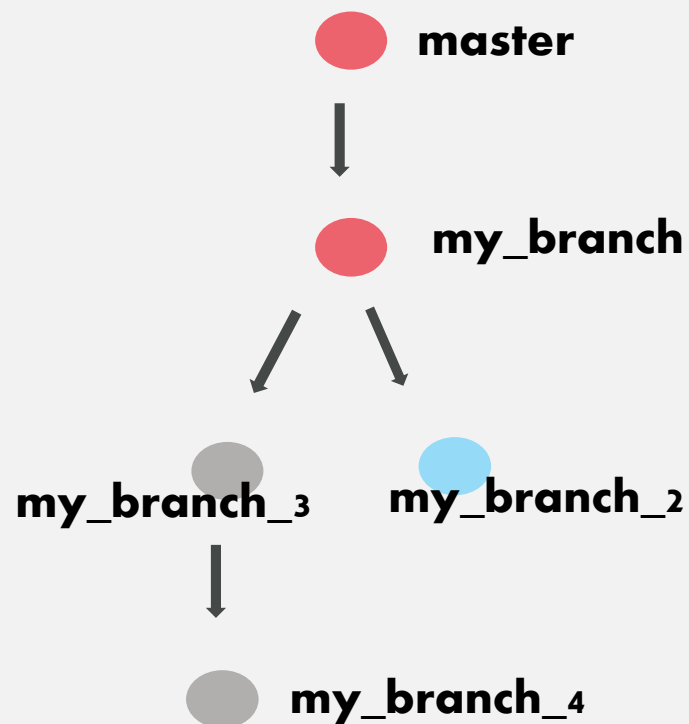
USER@DESKTOP-0LUGOQQ MINGW64 ~/Documents/몇쟁이사자처럼/test (master)
$ git checkout my_branch
Switched to branch 'my_branch'

USER@DESKTOP-0LUGOQQ MINGW64 ~/Documents/몇쟁이사자처럼/test (my_branch)
$ git branch my_branch_2

USER@DESKTOP-0LUGOQQ MINGW64 ~/Documents/몇쟁이사자처럼/test (my_branch)
$ git branch my_branch_3

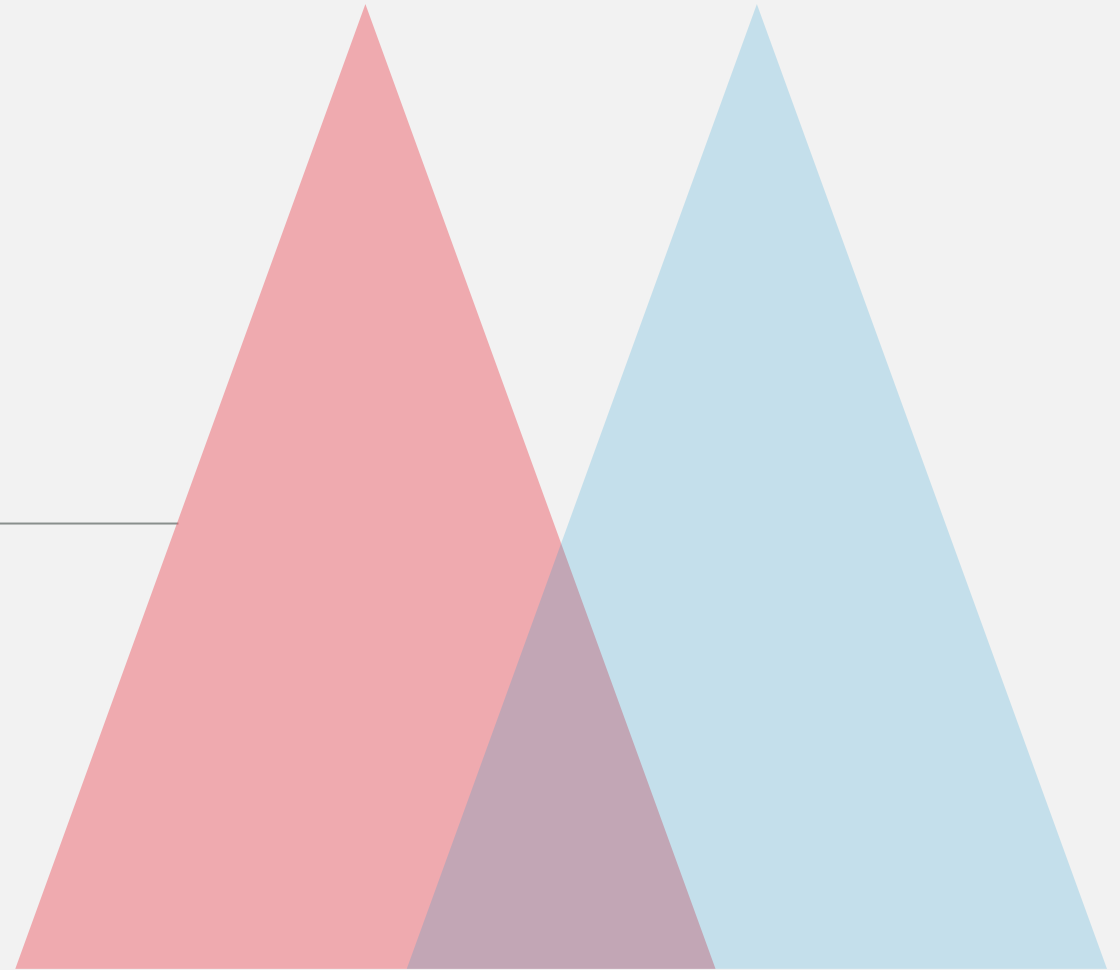
USER@DESKTOP-0LUGOQQ MINGW64 ~/Documents/몇쟁이사자처럼/test (my_branch)
$ git checkout my_branch_3
Switched to branch 'my_branch_3'

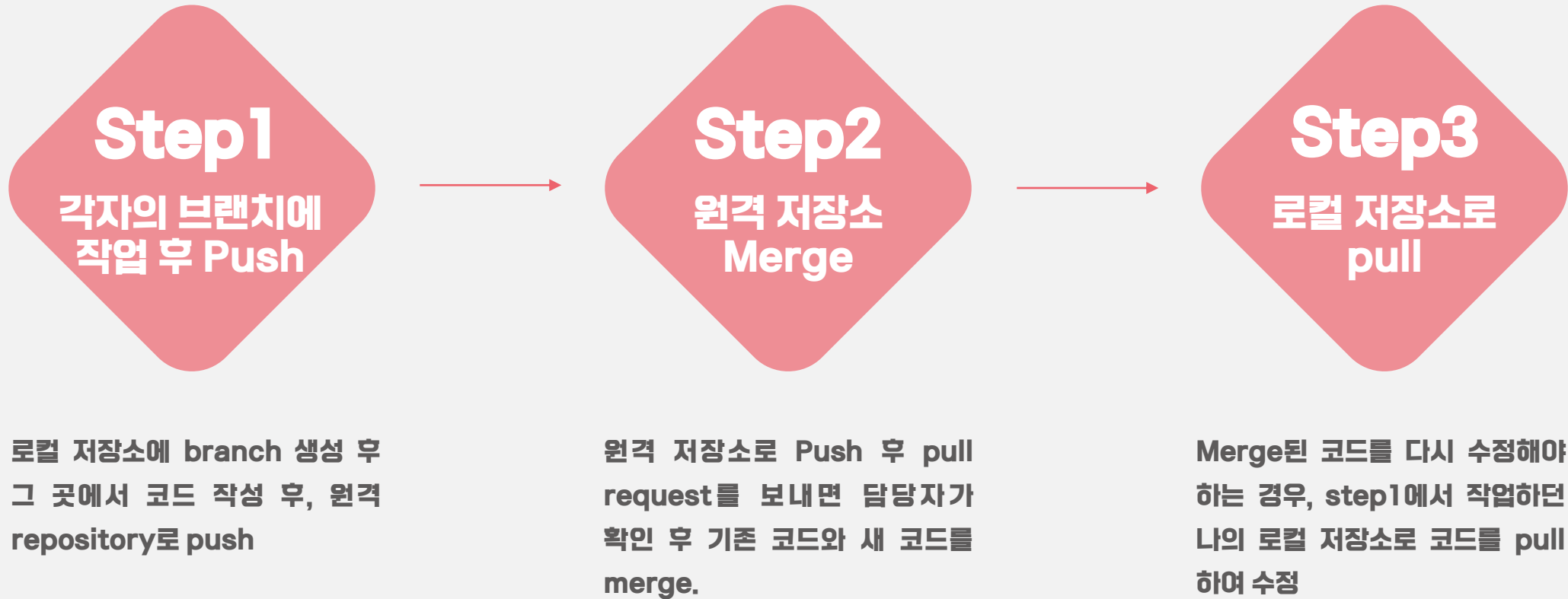
USER@DESKTOP-0LUGOQQ MINGW64 ~/Documents/몇쟁이사자처럼/test (my_branch_3)
$ git branch my_branch_4
```



03

Fetch & Merge





fetch + merge = pull

- Fetch : 가져오다 라는 뜻. 원격저장소의 코드를 로컬 저장소로 가져온다
- Merge : 새로 가져온 코드를 기존의 코드와 비교하여 합침
- fetch 와 clone의 차이점?
 - clone = pull + git remote add [레포주소.git]
- Merge의 대상 : 서로 다른 브랜치

Fetch & Merge

- 원격저장소(origin)의 소스코드를 fetch : **Git fetch origin**
- branch 끼리 merge :

Ex) exp의 내용을 master로 merge하고 싶으면

master로 체크아웃 후 merge

(이때 문서들은 commit된 이후여야 함!)

```
USER@DESKTOP-OLUGOQQ MINGW64 ~/Documents/멋쟁이사자처럼/test (exp)
$ git checkout master
Switched to branch 'master'

USER@DESKTOP-OLUGOQQ MINGW64 ~/Documents/멋쟁이사자처럼/test (master)
$ git merge exp
Merge made by the 'recursive' strategy.
 exp.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 exp.txt

USER@DESKTOP-OLUGOQQ MINGW64 ~/Documents/멋쟁이사자처럼/test (master)
$ |
```

[illegible]

- Commit 메시지를 입력하라는 창
당황하지 말고 'i'를 누르고 # 아래에
메시지를 입력해 준다
그리고 Esc를 누르고 :wq 입력, 엔터를
치고 나가면 된다

Merge Conflict

Merge 진행 시 A라는 사람과 B라는 사람이 **수정한 부분이 겹칠 때** 충돌이 일어난다.

어느 것으로 merge를 할지 git은 알 수 없다.

이러한 상황을 **충돌(conflict)**이라고 한다.

→ 개발자가 직접 수정해서 충돌을 없애야 한다

3

Merge Conflict

exp branch에는

a
bread
c

master branch에는

a
bee
c

라고 작성했다고 하고, 둘을 merge시켜 보자

```
USER@DESKTOP-0LUGOQQ MINGW64 ~/Documents/멋쟁이사자처럼/test (exp)
$ git merge master
CONFLICT (add/add): Merge conflict in test_a.txt 충돌 발생!
Auto-merging test_a.txt
Automatic merge failed; fix conflicts and then commit the result.
```

```
USER@DESKTOP-0LUGOQQ MINGW64 ~/Documents/멋쟁이사자처럼/test (exp|MERGING)
$ vi test_a.txt <- test_a.txt 파일을 연다는 의미의 명령어
```

3

Merge Conflict

```
a  
<<<<<< HEAD  
bread  
=====  
bee  
>>>>>> master  
c
```

《《《《《 HEAD 부터 》》》》》까지가 충돌이 일어난 부분이고

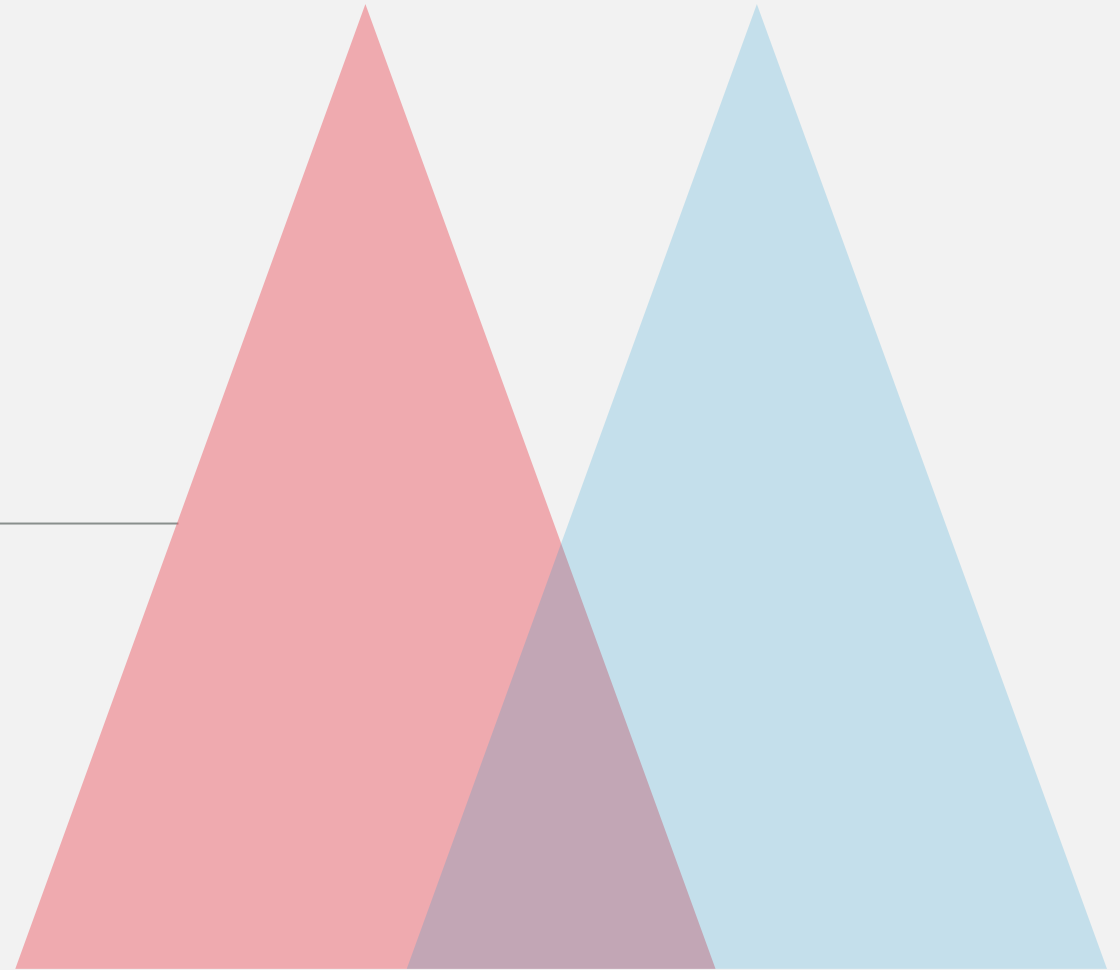
《《《《《 HEAD 부터 =====까지는 exp브랜치의 소스코드고
=====부터 》》》》》 master까지는 master브랜치의 소스코드다.

```
a  
boo|  
c
```

충돌이 일어나지 않게 고쳐준 후 다시 commit 하면 올바르게 merge가 된다

04

Pull



다른 사람들과 프로젝트 작업 시 or 한 사람이 두 대의 컴퓨터를 이용하여 작업하는 경우

동기화할 repository를 upstream이라고 지정 : `git remote add upstream [repository address].git`

연결 되어 있는 원격 저장소 목록 확인 : `git remote -v`

현재 로컬 브랜치로 내용 가져오기 : `git pull upstream main(=가져올 브랜치 이름)`

감사합니다

Thank you