Git and Github

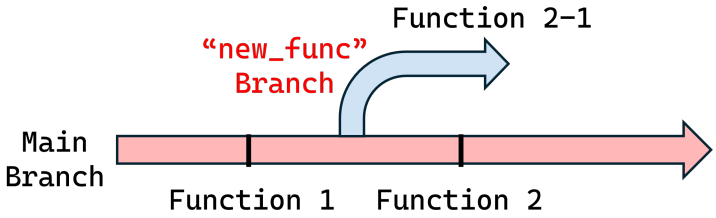
Author: Yongjae Lee

예전에 코딩애플 깃과 깃허브 강의보고 처음
배웠었는데 가끔 기억안날 때 보려고 만든 정리노트.

1 branch

커밋 사이를 가볍게 이동할 수 있는 어떤 포인터 같은 것을 Branch라고 함.

Figure 1.1. function2를 혹시 몰라서 두가지 버전으로 만드는 상황



새로운 브랜치를 만들고 싶으면 git branch "브랜치명"를 사용. 이후에 새로만든 브랜치에서 작업을 하고 싶으면 git switch "브랜치명". 이후, git status를 통해 현재 어떤 브랜치에서 작업하고 있는지 확인할것.

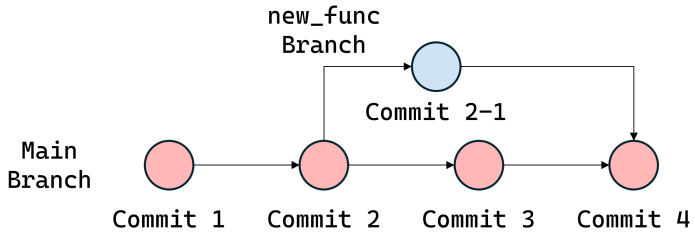
```
1  git branch new_func
2  git switch new_func
3  git status
```

이렇게 입력하면, new func이라는 브랜치를 만든후, 내 작업 위치를 new func으로 설정한거. git status를 입력하면, 내가 현재 어떤 브랜치에서 작업하고 있는지를 알려줌.

2 Merge

new func 브랜치에서 작업한 function 2-1이 잘 작동해서 Main 브랜치로 가져오고 싶은 상황에선 Merge를 사용해야함.

Figure 2.1. Commit 2-1을 통해 만든 function 2-1을 Main 브랜치로 가져오는 상황



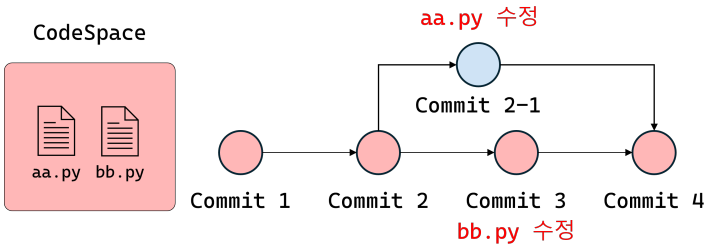
이런땐, 먼저 Main 브랜치로 이동해서, new func 브랜치를 merge하겠다고 입력하면됨. 현재 new func 브랜치에 있다고 가정하면,

```
1  git switch main
2  git merge new_func
```

2.1 3-Way Merge

Merge를 하다보면 여러가지 상황이 발생할 수 있는데 먼저, 가장 이상적인 상황은 브랜치에선 aa.py를 수정했고, Main브랜치에선 bb.py를 수정해서 둘이 합칠때 아무 일도 발생하지 않는 경우임.

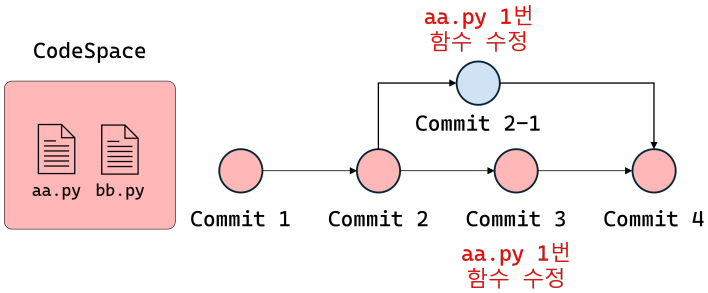
Figure 2.2. new func 브랜치에선 aa.py를 수정, main 브랜치에선 bb.py를 수정한 상황




```
git switch main
git merge new_func
```

이런 경우엔 그냥 merge하면 아무 문제도 발생하지 않고, aa.py와 bb.py 둘다 잘 업데이트됨.

Figure 2.3. 하지만, Main브랜치에서도 aa.py의 1번 함수를 수정하고, new func브랜치에서도 aa.py의 1번 함수를 수정하면, 충돌이 발생됨. 이런 경우엔 merge를 하게되면 충돌(Conflict)가 발생되었다는 화면이 나옴. 거기서 원하는 코드만 남기고 저장한 다음에 add와 commit을 해주어야함.

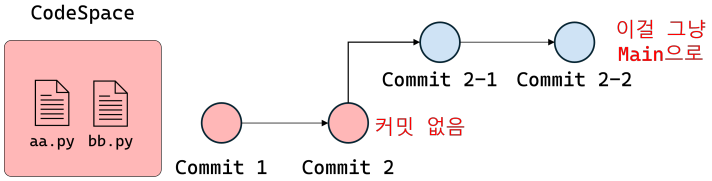



```
git switch main
git merge new_func
# after code fixing
git add .
git commit -m "Conflict clear"
```

2.2 Fast Forward Merge

만약 new func브랜치를 만들고 커밋을 여러번 하는동안, Main 브랜치에선 아무 변화도 없는 상황엔 Fast Forward Merge를 해야함. 말이 좋아 Fast Forward Merge인거지, 사실 그냥 new func 브랜치를 Main에 그대로 이어붙이는 걸 말하는것.

Figure 2.4. Main브랜치엔 변화가 없으니, 그냥 new func 브랜치를 Main으로 바꿈.



2.3 Delete Branch

new func 브랜치를 Main에 합쳤다고해서 new func 브랜치가 삭제되는게 아님.
Merge된 브랜치는 git branch -d 브랜치명
Merge안된 브랜치는 git branch - D 브랜치명

```
git branch -d new_func
```

Merge된 브랜치를 삭제함.

```
git branch -D new_func
```

Merge가 안된 브랜치를 삭제하는거.

2.4 Rebase

Rebase는 브랜치가 시작된 지점을 바꿀 수 있게해줌.

Figure 2.5. 일반적인 상황에선 Main 브랜치와 new func브랜치를 합치기위해 3-Way Merge될것.

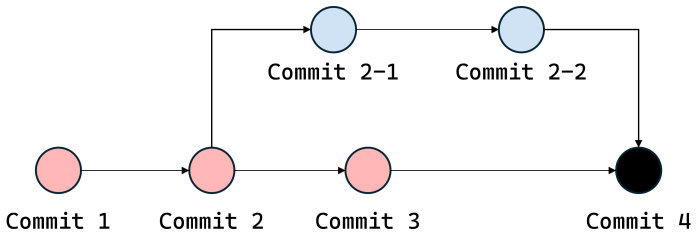
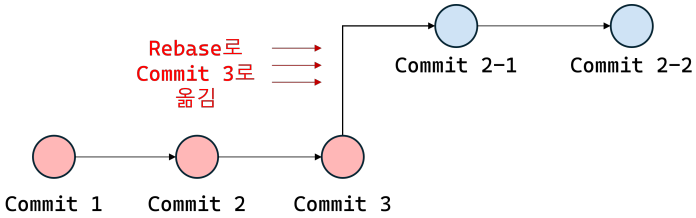


Figure 2.6. Rebase를 활용하면, new func브랜치가 Commit3에서 시작되었다고 변경해서 Fast Foward Merge할 수 있게됨.



Merge에선 Main으로 이동해서 git merge 브랜치명 을 통해 merge를 했지만, Rebase는 반대임. Rebase할 브랜치로 이동해서 어디로 Rebase 시킬지를 입력해야함.

2.5 Squash and Merge

깃허브에 로그를 깔끔하게 남기고 싶으면 찾아서 쓰면됨. 난 잘 안씀. 깃 로그는 자고로 많으면 많을수록 남자다움을 상징함.