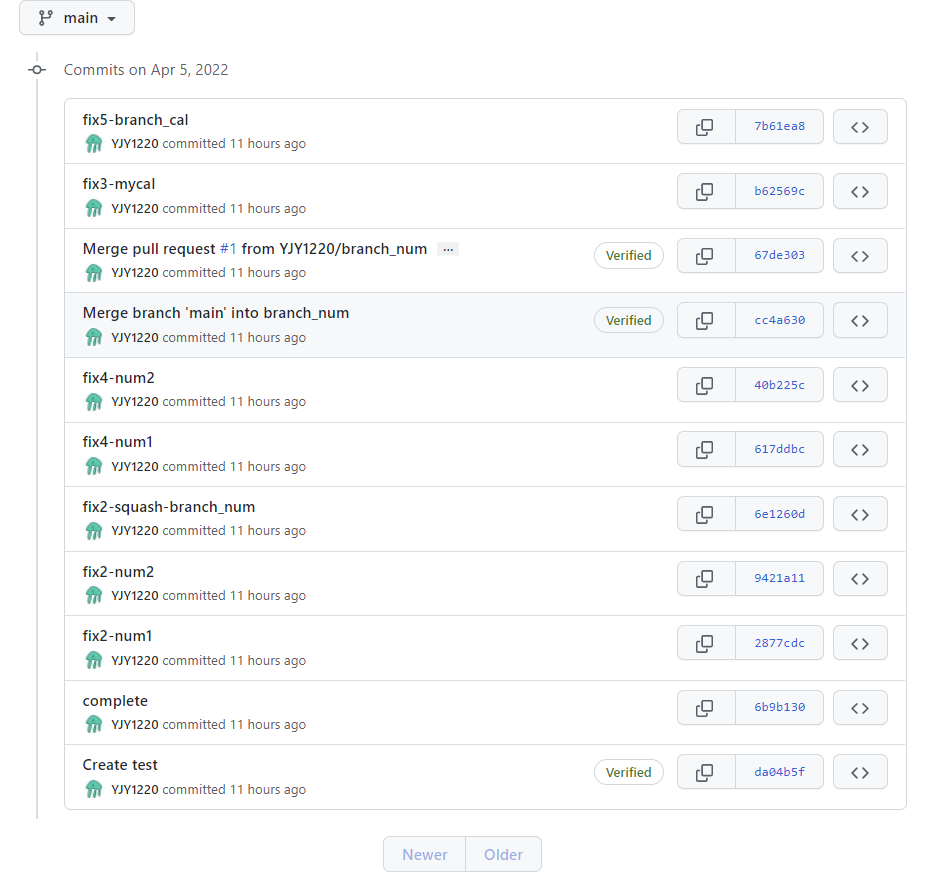
[오픈소스 프로그래밍 – HW04 <project management in github>]

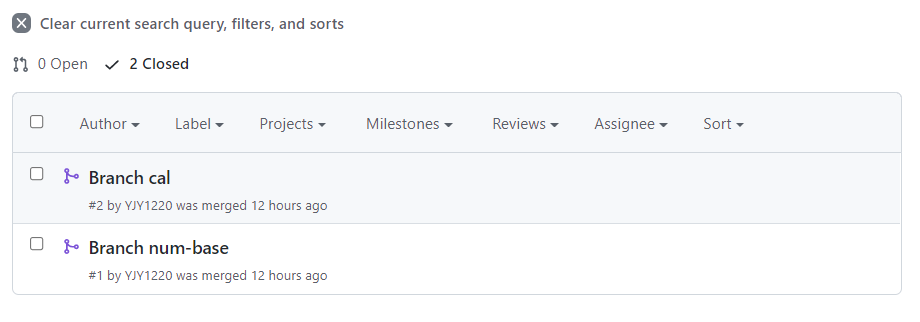
컴퓨터학부 글로벌sw융합전공

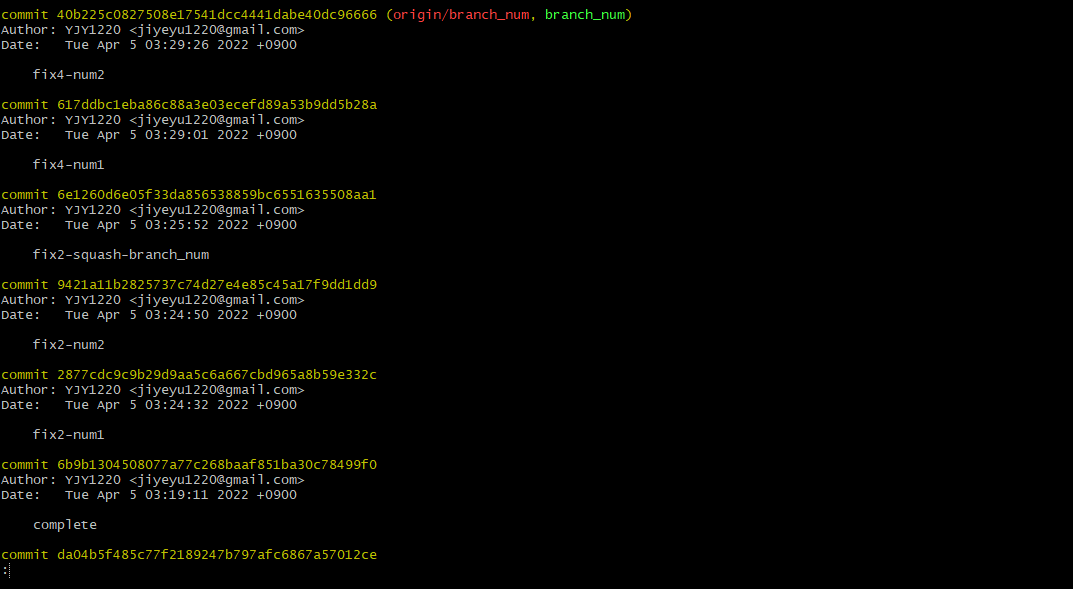
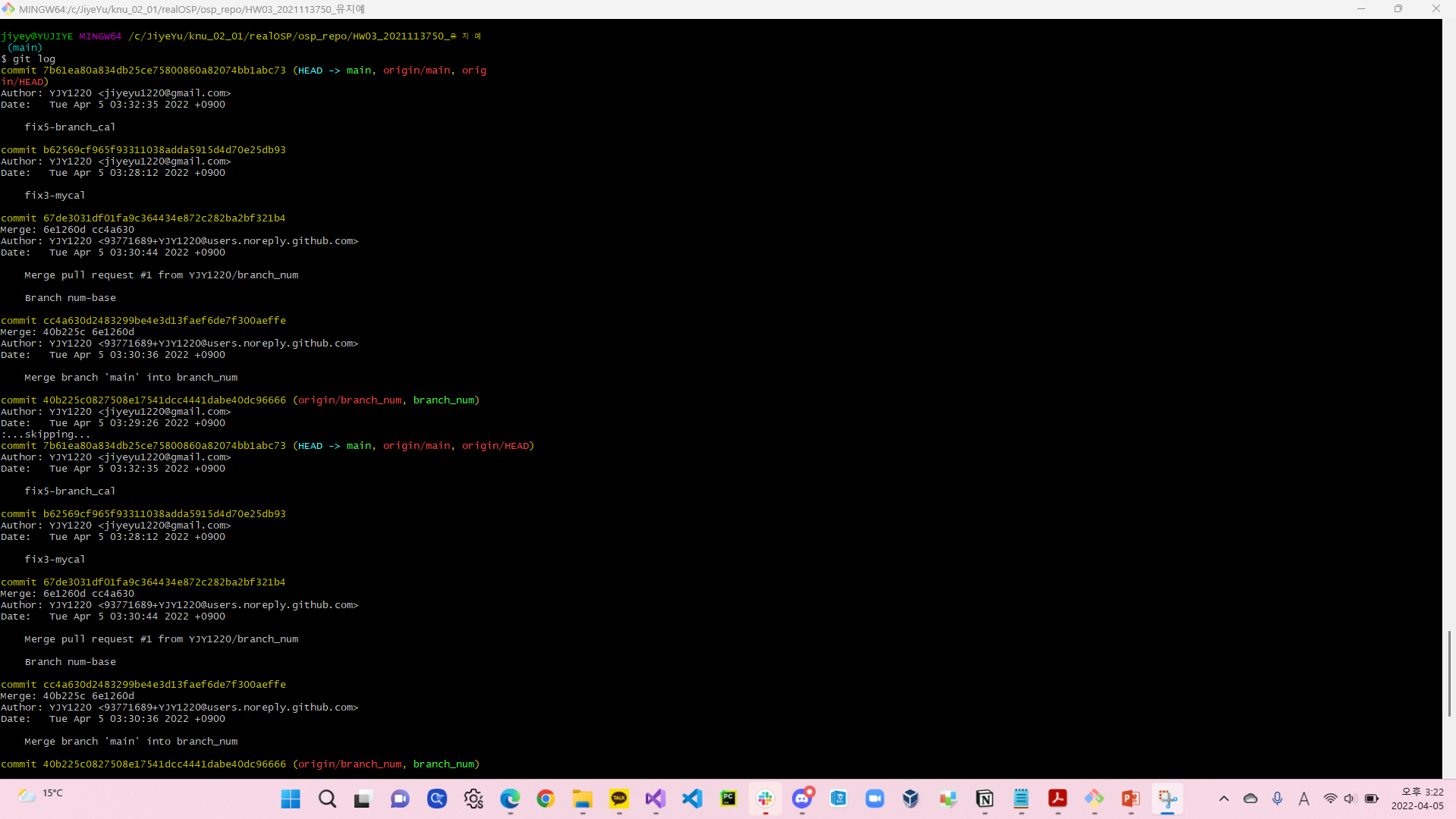
2021113750 유지예

<YJY1220 / osp\_repo / commit history>

* 4번을 하는 도중 local에서 ‘$git merge branch\_num’ 을 하다가 conflict가 났습니다. 그래서 튜터님께 여쭈어보고 기존 osp\_repo를 삭제하고 새로운 osp\_repo를 팠습니다!





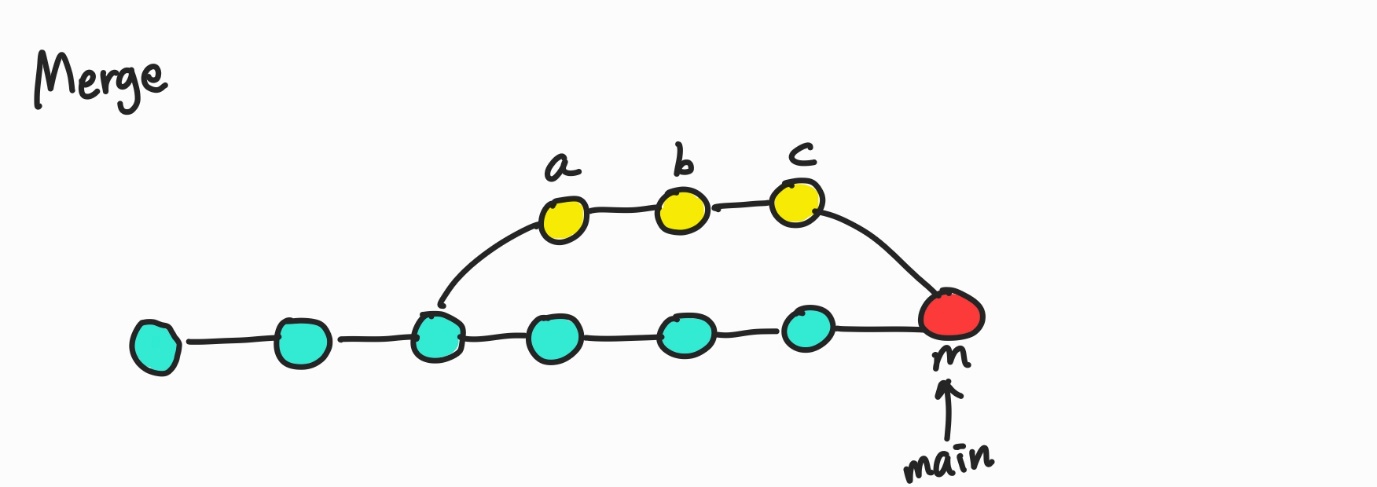


== 각 merge하는 방법 차이점 설명 ==

<Merge>

-정의-

하나의 브랜치와 다른 브랜치의 변경 이력 전체를 합치는 방법



-방법-

commit a, b, c를 refer하는 m이 생성되고 m을 통해 a + b + c가 main에 추가됨.

m 은 2개의 parent를 가짐

$ git checkout main

$ git merge branch\_num

$ git push origin main

-장점-

Git history에서 해당 pull request에 대해 상세하게 확인할 수 있음

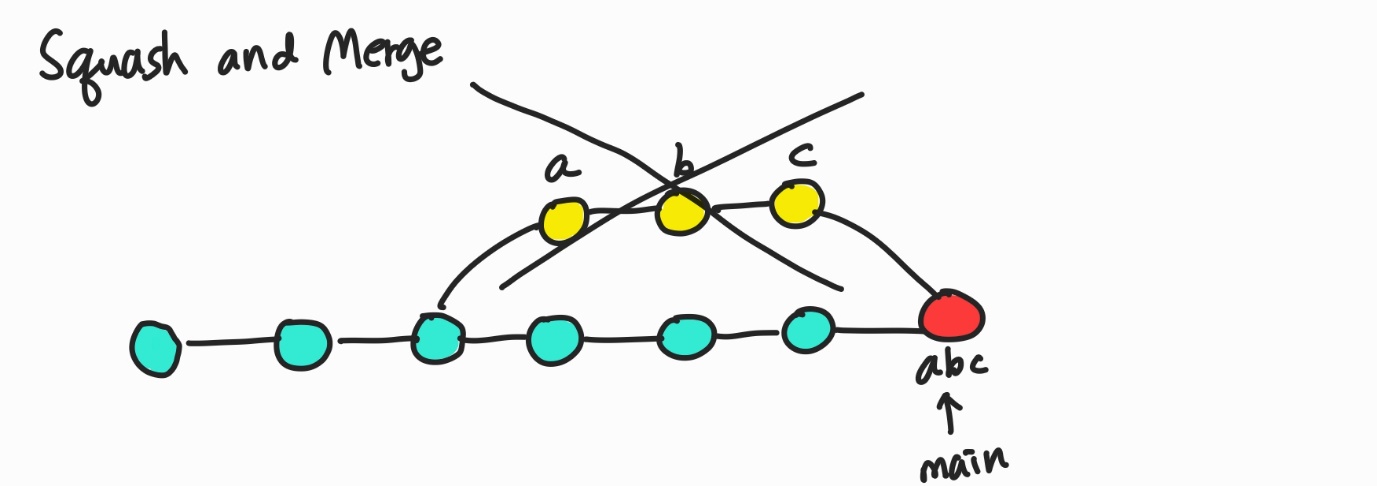
-단점-

Branch가 많아지고 commit 도 많아지고 하면 git history는 알아보기가 힘들어져서 프로젝트 규모가 크면 클수록 이 방식은 선호되지 않음.

<Squash and merge>

-정의-

각 브랜치들의 commit 변경 내역을 합쳐 새로운 commit을 만든 후 main에 추가함



-방법-

Commit a + b + c를 합쳐서 새로운 commit, abc를 만들어지고 main에 추가됨.

abc 는 1개의 parent를 가짐

feature 브랜치의 commit history를 합쳐서 깔끔하게 만들기 위해 사용함.

해당 브랜치의 여러 개의 history가 하나로 합쳐져서 새로운 commit을 생성함.

$ git checkout main

$ git merge –squash branch\_cal

$ git commit –m “fix4 – branch\_cal”

$ git push origin main

저는 conflict를 해결하기 위해서 원격에서 pull request 후 squash merge를 진행 한 후 원격에서 conflict를 해결했습니다. 이후 local에 원격의 최신 상태를 반영하기 위해 아래와 같은 작업을 진행했습니다.

$ git remote update

$ git pull origin main

-장점-

Git history의 branch도 깔끔하게 유지하되 자잘한 commit은 없애고 의미있는 commit으로만 history를 이룰 수 있음

-단점-

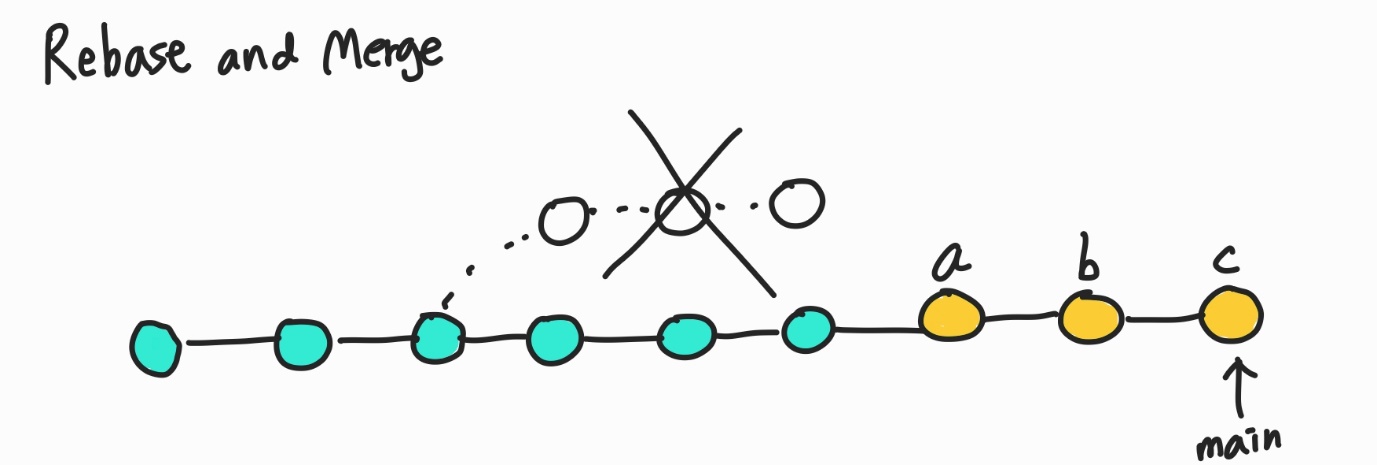
해당 merge에 대한 정보가 부족함 -> merge commit에 정보를 자세히 명시해놓을 필요가 있음

<Rebase and merge>

-정의-

Merge할 branch의 commit 내역들을 그대로 옮겨서 branch는 이어지지 않음.

모든 commit들이 합쳐지지 않고 각각 main 브랜치에 추가됨.



-방법-

각 commit은 모두 하나의 parent를 가짐.

Merge는 merge commit 기록이 추가로 남게 되지만 rebase의 경우에는 branch 병합 시 merge commit 기록이 남지 않아서 마치 하나의 브랜치에서 작업한 것처럼 보여짐.

$ git checkout branch\_num

$ git rebase main

$ git checkout main

$ git merge branch\_num

저는 local에서의 conflict를 해결하기 위해 원격에서 new pull request 후 rebase and merge했습니다. 이 후 local에 원격의 최신 상태를 반영하기 위해 아래와 같이 진행했습니다.

$ git remote update

$ git pull origin main

Q. 5번에서 rebase가 아닌 merge만 했다면?

만약 rebase가 아니라 merge만 했다면, 각 branch의 commit내역들을 그대로 옮겨서 branch가 이어지지 않는 것이 아니라 branch의 commit 내역들인 a + b + c를 main에 merge할 것입니다. Rebase가 아닌 merge만 한다면 commit들이 합쳐져서 main 브랜치에 추가될 것입니다.

-장점-

Git history를 볼 때 깔끔하게 볼 수 있음. 여러 개의 branch가 복잡하게 얽혀있지 않고 그냥 default branch 하나의 history만 쭉 읽으면 해당 프로젝트의 이력을 이해할 수 있음.

-단점-

여러 개의 commit을 rebase merge 했는데 commit conflict가 일어나면 merge하려는 모든 commit에서 conflict가 일어남.

또한, 거미줄처럼 이어지는 branch는 사라지지만, 여전히 의미가 없다시피 하는 commit도 history에 남음.