**깃허브의 브랜치 병합**

201610161 경영회계학부 회계학 전공 박재형

Git에서 브랜치를 합치는 방법으로는 일반적으로 merge를 사용하는 방법과 rebase를 사용하는 방법이 있으며, 어느 명령어를 사용하느냐에 따라 병합 후 브랜치의 이력이 크게 달라진다.

Merge를 사용하면, 여러 개의 브랜치를 하나로 모을 수 있다. 사용법은 ‘git merge [브랜치 명]’으로, 실행하면 지정한 커밋의 내용이 헤드가 가리키고 있는 브랜치에 넣어지며, 헤드는 현재 사용 중인 브랜치에 위치하게 된다. Merge는 현재의 브랜치에서 [브랜치 명]의 변경사항을 병합한다. 따라서 A브랜치에서 B브랜치를 Merge 하는 것과 B브랜치에서 A브랜치를 Merge 하는 것은 차이가 있다.

A브랜치에서 분기하는 B브랜치가 있고 이 B브랜치를 A브랜치로 병합할 때, A브랜치의 상태가 이전부터 변경되어 있지만 않으면 매우 쉽게 병합할 수 있는데, B브랜치의 이력은 A브랜치의 이력을 모두 포함하고 있기 때문에 A브랜치는 단순히 이동하기만 해도 B브랜치의 내용을 적용할 수 있다. 이처럼 A브랜치에서 B브랜치를 분기해준 것 외에 별도의 커밋 로그 없이 B브랜치가 만든 최신의 커밋 로그를 가리키게 만드는 것을 Fast-Foward(빨리 감기) 병합이라고 한다.

빨리 감기 병합은 수행 속도가 빠르고 불필요한 브랜치를 제거해 주는 역할을 하지만, 두 개의 브랜치의 구분이 사라지기 때문에 프로젝트를 관리하는 면에서 모호함이 발생할 가능성이 있다.

이와 다르게 A브랜치에서 B브랜치를 생성한 후, A브랜치와 B브랜치 모두 여러 번의 커밋이 있었을 때 A브랜치에서 B브랜치를 Merge할 경우 빨리 감기 병합이 불가능한데, 이때 git은 A브랜치와 B브랜치의 조상 커밋을 찾아 내부적으로 3Way Merge라는 방법을 이용하여 Merge를 수행하며, 성공적으로 병합할 경우 ‘Merge made by recursive strategy’라는 메시지가 표시된다. 3Way Merge 작업을 수행하고 나면 합칠 브랜치에서 가져온 커밋이 몇 개 였든지 간에, 현재 브랜치에는 딱 하나의 커밋만 새로 생성되므로 Merge를 취소하고 싶다면 새로 생성된 커밋만 삭제해 주면 된다.

Merge 명령에 별도의 옵션을 주지 않는다면 기본적으로 빨리 감기 병합을 수행하며, 빨리 감기 병합이 불가능할 경우 3way merge를 수행하지만, git merge —no-ff [브랜치 명]처럼 별도의 옵션을 줄 경우 빨리 감기 병합이 수행 가능하더라도 3way merge를 수행한다. 경우에 따라 의도적으로 작업 기록을 남겨두기 위해 빨리 감기 병합을 수행하지 말아야 할 경우 이러한 옵션을 활용할 수 있다.

Merge가 실패하여 충돌이 발생할 경우, 개발자가 직접 해당 부분의 소스코드를 수정하여 관리를 해줄 필요가 있다. 여러 명이 같은 파일에서 작업을 할 경우 기존의 내용이 서로 다를 수 있으며, 이 경우 충돌이 발생하게 된다. 충돌이 발생할 경우 충돌을 해결하기 위한 임의의 브랜치로 변경되며, 사용자는 충돌이 발생한 코드를 확인, 변경하여 커밋 하면 충돌을 해결할 수 있다. 이때, git diff, git status 명령어 들을 이용하면 Git이 어떤 파일을 Merge 할 수 없었는지, 어디서 충돌이 발생하였으며, 어떤 변경이 있었는지를 추적할 수 있다.

혼자서 개발하는 것이 아닌 협력 환경에서 병합 시 충돌을 줄이기 위해서는 master 브랜치의 변화를 지속적으로 브랜치로 가져와서 충돌이 발생하는 부분을 매번 제거하면서 자신의 브랜치를 개발해야 이후 master 브랜치로 병합할 때 충돌의 발생 수를 줄일 수 있다.

Rebase는 Merge와는 다르게 브랜치의 공통 조상이 되는 base를 다른 브랜치의 커밋 지점으로 바꾸는 것으로, Rebase 하려는 브랜치 커밋들의 변경사항을 Patch라는 것으로 만든 다음에 어딘가에 저장해 둔 후. 이를 master 브랜치에 하나씩 적용하여 새로운 커밋을 만드는 것이다. ‘git rebase [브랜치 명]’으로 사용할 수 있으며, --interactive 옵션을 사용하여 커밋에서 작업했던 내용들은 그대로 유지하면서 불필요한 커밋 이력을 제거해서 필요한 커밋들만 남기는 등 커밋 로그를 조작할 수 있다.

Merge와 Rebase를 비교해보면, Merge로 브랜치를 병합하게 되면 커밋 내역에 Merge 커밋이 추가로 남게 된다. 따라서 Merge를 사용하면 브랜치가 생기고 병합되는 모든 작업 내용을 그대로 기록하게 되므로 이력이 복잡해질 가능성이 있으며, 반대로 Rebase의 경우는 브랜치를 병합할 때 이런 Merge 커밋을 남기지 않으므로, 마치 다른 브랜치는 없었던 것처럼 프로젝트의 작업 내용이 하나의 흐름으로 유지되지만 정확한 이력을 남겨야 할 필요가 있을 경우 사용하면 안 된다.

브랜치를 병합할 때 Merge를 써야 하는지 Rebase를 써야 하는지에 대해서는 정답이 없으며, 프로젝트나 팀의 상황에 따라 다른 전략을 사용할 수 있다. 일반적으로 토픽 브랜치에 통합 브랜치의 최신 코드를 적용할 경우에는 Rebase를 사용하며, 통합 브랜치에 토픽 브랜치를 불러올 경우에는 우선 Rebase를 한 후 Merge를 사용한다. 로컬에서 히스토리 정리를 위해 Rebase를 사용할 수도 있으나, 이미 원격 저장소에 Push된 커밋은 Rebase를 하지 않는 것이 일반적이다.

앞의 두 가지 방법 외에 cherry-pick을 사용하는 방법도 있는데, 이는 다른 브랜치에 있는 커밋을 선별적으로 현재의 브랜치에 반영하기 위한 명령어이다. ‘git cherry-pick {커밋 ID}’를 통해 사용할 수 있으며, 사용 시 현재 브랜치는 작업 내역을 반영할 브랜치여야 하고, 가져오려는 커밋이 여러 개인 경우 먼저 작성한 커밋부터 순서대로 가져와야 한다. 또, git cherry-pick –n처럼 -n 매개변수를 지정하여 Git을 합치기는 하지만 커밋 하지 않게 할 수도 있다. cherry-pick은 엄밀히 말해 커밋을 가져오는 것이 아닌, 가져올 커밋과 동일한 커밋을 새로 만들어서 현재의 브랜치에 덧붙이는 작업이다. 그러므로 현재 브랜치에 추가되는 커밋은 가져온 커밋과 커밋 ID가 달라진다. 따라서 cherry-pick은 커밋을 새로 하는 것과 같다고 볼 수 있다.

참고자료

uang, 「Github 기초 사용법1」, uang's blog, <https://uang.tistory.com/9>

backlog, 「누구나 쉽게 이해할 수 있는 Git 입문」, backlog.com, <https://backlog.com/git-tutorial/kr/stepup/stepup1_1.html>

git-scm, 「Git 브랜치」, gut-scm.com, <https://git-scm.com/book/ko/v2>

이승준,「병합하고 Commit 재정렬하기: cherry-pick, rebase, merge」, tuwlab, <https://www.tuwlab.com/ece/22218>

godori, 「Git Rebase 활용하기」, godori.log, <https://velog.io/@godori/Git-Rebase>