

week12

Rebase

base란?



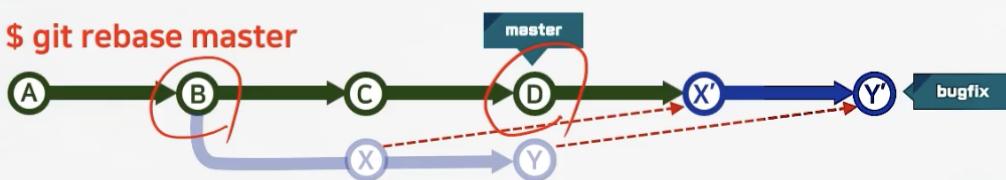
rebase를 이용한 브랜치 병합 과정

'fast-forward 병합' 방식

- master 브랜치 뒤로 bugfix 브랜치의 이력이 이동
 - 이력이 하나의 줄기로 이어짐
 - 충돌 발생이 가능

rebase만 하면 다음 그림처럼 'master'의 위치는 그대로 유지

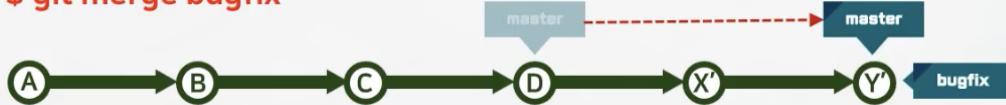
```
$ git rebase master
```



마스터 브랜치의 위치를 변경하기 위해서는

- master 브랜치에서 bugfix 브랜치를 fast-forward(빨리 감기) 병합 필요

\$ git merge bugfix



rebase 후 충돌 발생 시 해결 절차

1. 파일 수정
 2. 파일 추가 : git add 수정파일
 3. rebase 계속 수행, 마지막 메시지 설정 : git rebase —continue

rebase 의 효과

- 히스토리가 선형으로 단순해지고 더 깨끗한 이력을 남김(+ merge만 하는거에 비해 깔끔함)
 - 원래의 커밋 이력이 변경됨. but 이력이 남겨야 할 경우 사용하면 안됨

fast-forward merge와 rebase 비교

fast-forward merge

- 조상에 위치한 브랜치에서 선행 브랜치의 마지막으로 이동하는 병합

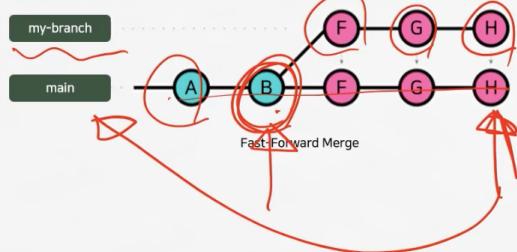
rebase

- 두 갈래의 브랜치에서

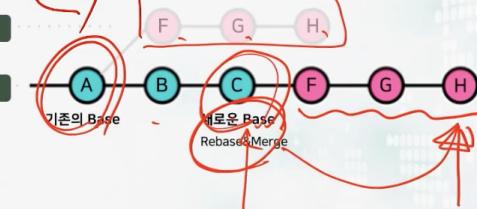
- 기존의 베이스를 수정

▶ 병합할 브랜치 마지막 커밋을 새로운 베이스로 수정하는 병합

Merge (Fast Forward)



Rebase & Merge



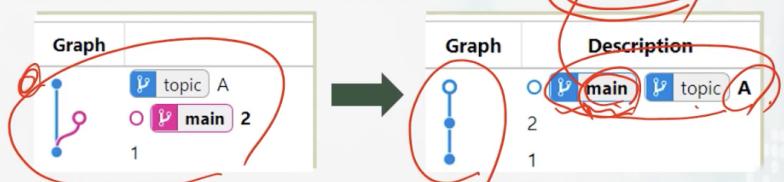
\$ git rebase <newparent> <branch>

일반적 rebase 방법

- topic에서 main을 rebase 한 이후, 다시 main으로 이동 fast-forward 병합 수행
 - \$ git checkout topic
 - \$ git rebase main
 - \$ git checkout main
 - \$ git merge topic

다른 rebase 방법: 어느 브랜치든 main topic 순서로 재배치 방법

- \$ git rebase main topic
- \$ git checkout main
- \$ git merge topic

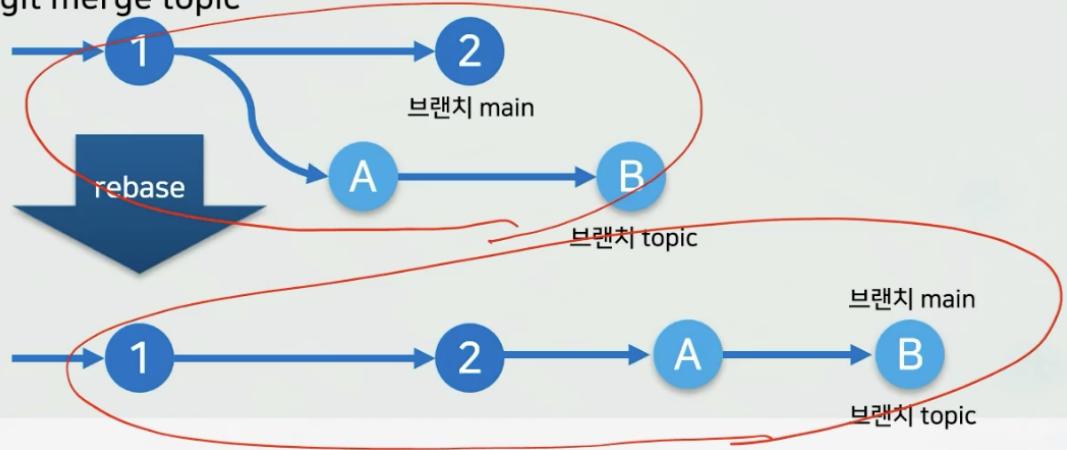


» 기준 브랜치에서 main 브랜치 rebase 병합

- ◆ \$ git checkout topic
- ◆ \$ git rebase main

» 다시 main을 돌아와 fast-forward 병합 진행

- ◆ \$ git checkout main
- ◆ \$ git merge topic



커밋 이력 수정

최근 커밋 내용 수정

HEAD의 내용 수정

- 새로운 파일을 추가하거나 파일을 수정한 후
- 새로운 커밋으로 생성하지 않고
- 최신 커밋에 반영
 - 새로운 커밋 ID로 수정됨

커밋 옵션 --amend 사용

- 기본 편집기로 메시지 편집
 - \$ git commit --amend
- 명령어 상에 직접 메시지 입력
 - \$ git commit --amend -m 'new message'
- 메시지 수정 없이 다시 커밋 수정
 - \$ git commit --amend --no-edit

2 rebase-i로 여러 커밋수정

rebase의 --interactive를 사용

- 작업공간이 깨끗한 이후, 이전 여러 개의 커밋을 수정
 - 이전 커밋을 다시 작성한 경우 새 ID가 부여
- rebase의 --interactive를 사용하여 커밋 시퀀스를 새로운 기본 커밋에 결합
 - \$ git rebase -i HEAD~3
 - \$ git rebase --interactive HEAD~3
 - HEAD~3: 수정할 커밋의 직전 커밋
 - 실제 HEAD~2부터 수정 가능

주요 rebase -i 대화형 명령어

- p(ick) : 해당 커밋을 수정하지 않고 그대로 사용
- r(eword) : 개별 커밋 메시지를 다시 작성
- s(quash) : 계속된 이후 커밋을 이전 커밋에 결합
- d(rop) : 커밋 자체를 삭제

ctrl+save, ctrl+f4

Summary

» 최신 커밋 메시지 수정

- \$ git commit --amend -m 'new message'

» 편집기로 최신 커밋 메시지 수정

- \$ git commit --amend

» 파일 수정 후 추가, 메시지 수정 없이 최신 커밋으로 수정

- \$ git commit --amend --no-edit

» 이전 커밋 HEAD~2..HEAD까지 각각의 커밋을 수정

- \$ git rebase --interactive HEAD~3

◆ 주요 rebase -i 대화형 명령어

- p(ick): 해당 커밋을 수정하지 않고 그대로 사용
- r(eword): 개별 커밋 메시지를 다시 작성
- s(quash): 계속된 이후 커밋을 이전 커밋에 결합
- d(rop): 커밋 자체를 삭제