
Open source Software
오픈소스 소프트웨어
25. 브랜치 리베이스 rebase

학습 개요

1. base를 수정하는 rebase
2. 3-way merge와 rebase 비교

학습 목표

1. Base를 재배치하는 rebase 병합을 이해하고 수행할 수 있다.
2. 3-way 병합과 rebase의 차이를 이해할 수 있다.

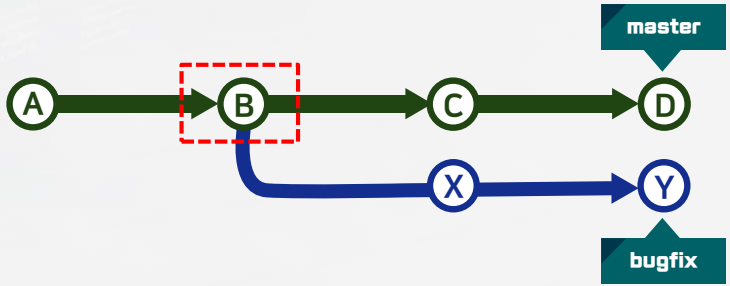
LESSON 01

병합 rebase

3-way 상태에서 base의 이해

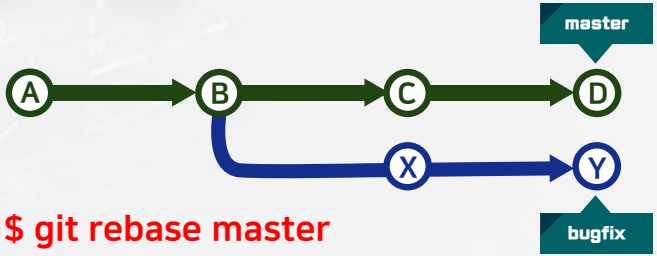
'master' 브랜치 커밋 B에서 분기되는 'bugfix' 브랜치

- 커밋 B
  - 현재 master와 bugfix의 공통 조상
    - ➡ 이를 base라 칭함



선형적 통합 rebase 이해

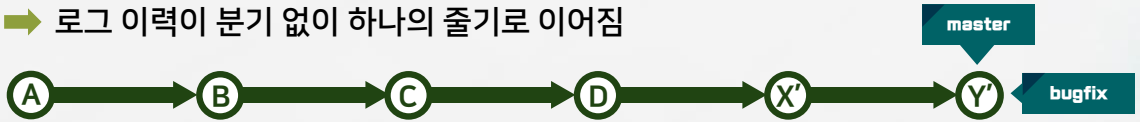
브랜치 bugfix에서 base를 바꾸는 재배치하기 이전



$ git rebase master

재배치 rebase 병합 수행

- base를 수정
  - B에서 마스터의 최신 커밋인 D로 수정
    - ➡ D 이후에 bugfix를 배치
- 이후 다시 'fast-forward 병합' 수행: 이 병합을 직접 다시 해야함
  - 'master' 브랜치의 HEAD가 'bugfix' 브랜치 마지막 커밋으로 이동
    - ➡ 로그 이력이 분기 없이 하나의 줄기로 이어짐



1 브랜치 병합 개요

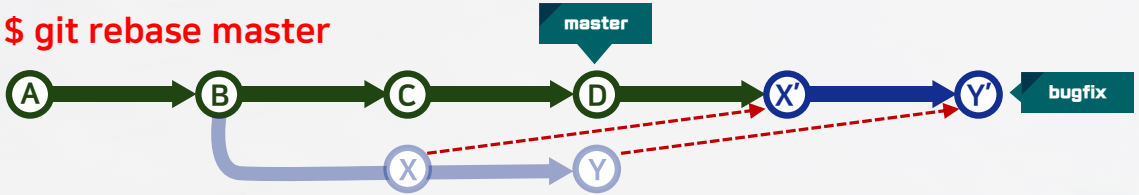
rebase를 이용한 브랜치 병합 과정

'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

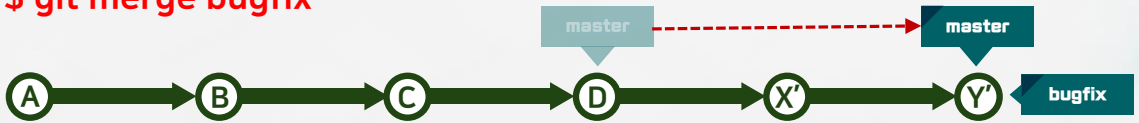
```
$ git rebase master
```



마스터 브랜치의 위치를 변경하기 위해서는

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

```
$ git merge bugfix
```



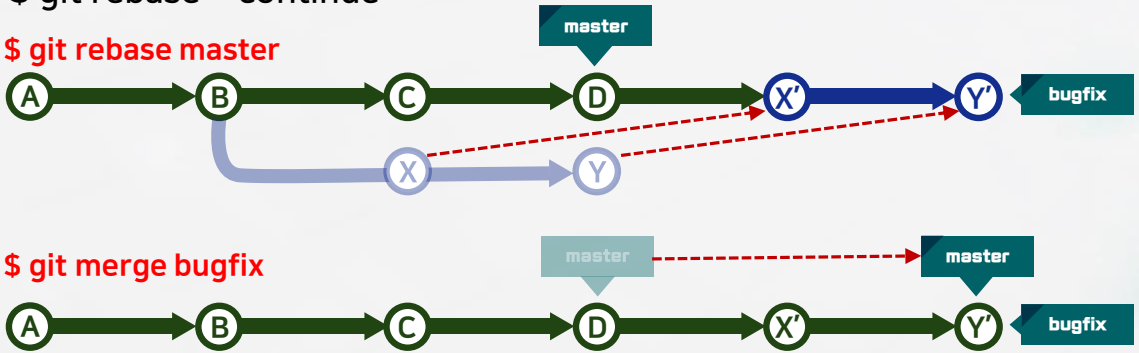
Rebase에서의 충돌

충돌 발생 가능

- 이동되는 X와 Y의 내용이 'master'의 C, D 커밋들과 충돌하는 부분이 생길 수 있음
  - 각각의 커밋에서 발생한 충돌 내용을 수정 후, 추가, 계속 수행 필요

충돌 발생 후 해결 절차

- 1. 파일 수정
- 2. 파일 추가
  - $ git add <수정파일>
- 3. rebase 계속 수행, 마지막 메시지 메시지 수정
  - $ git rebase --continue



LESSON 02

3-way, fast-forward와 rebase 비교

2 3-way, fast-forward와 rebase 비교

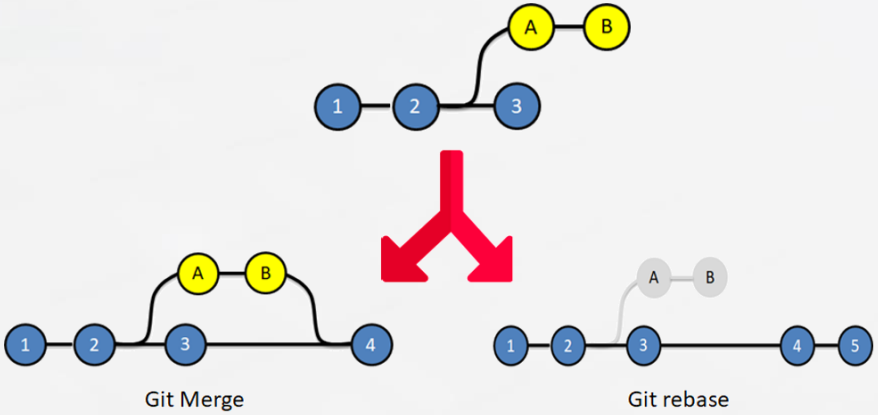
3-way merge와 rebase 비교

merge

- 여러 분기가 생긴 변경 내용의 이력이 모두 그대로 남아 있기 때문에 이력이 복잡해짐

rebase

- 히스토리가 선형으로 단순해지고 좀 더 깨끗한 이력을 남김
- 원래의 커밋 이력이 변경됨
  - 정확한 이력을 남겨야 할 필요가 있을 경우에는 사용하면 안됨



2 3-way, fast-forward와 rebase 비교

fast-forward merge와 rebase 비교

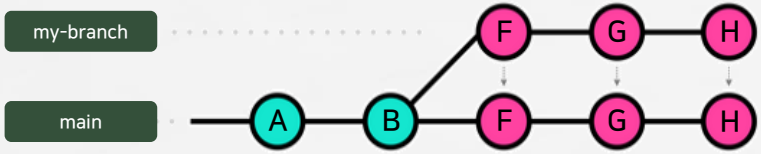
fast-forward merge

- 조상에 위치한 브랜치에서 선행 브랜치의 마지막으로 이동하는 병합

rebase

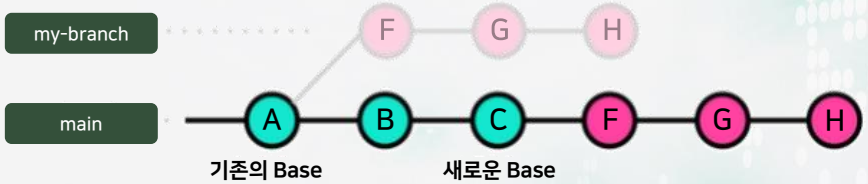
- 두 갈래의 브랜치에서
  - 기존의 베이스를 수정
    - ➡ 병합할 브랜치 마지막 커밋을 새로운 베이스로 수정하는 병합

Merge (Fast Forward)



Fast-Forward Merge

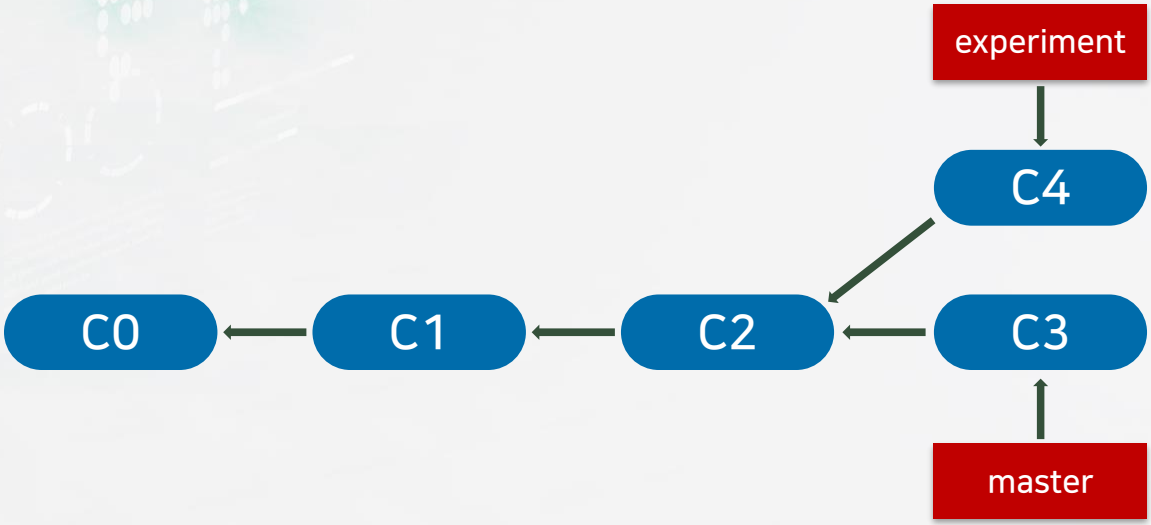
Rebase & Merge



Rebase&Merge

2 3-way, fast-forward와 rebase 비교

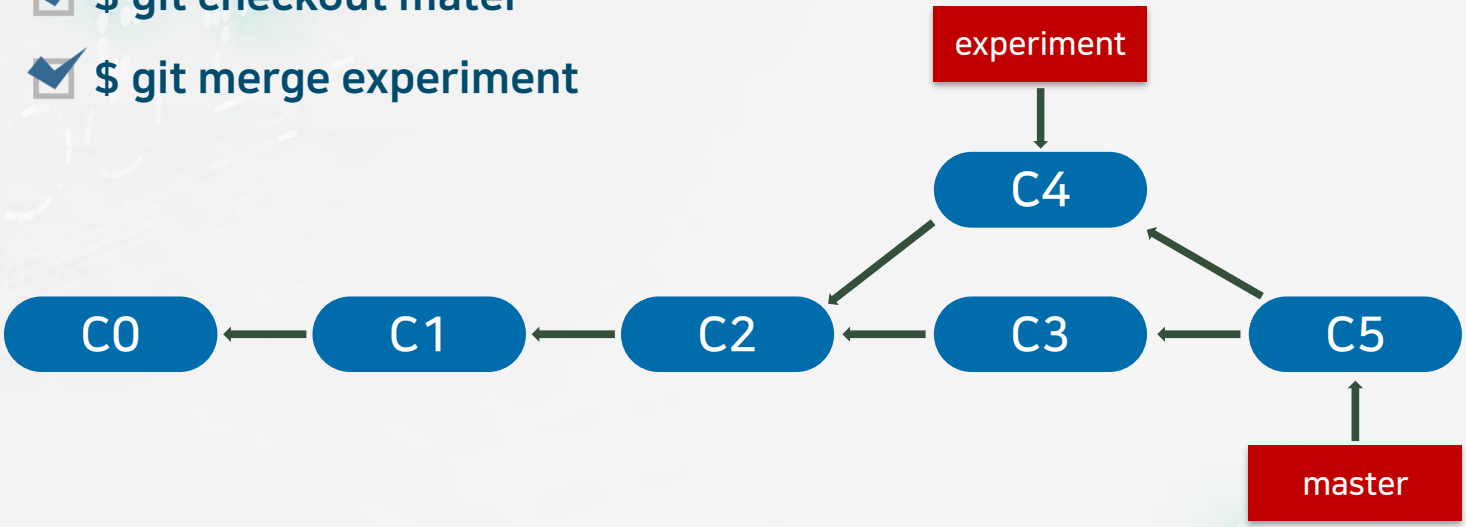
두 브랜치



2 3-way, fast-forward와 rebase 비교

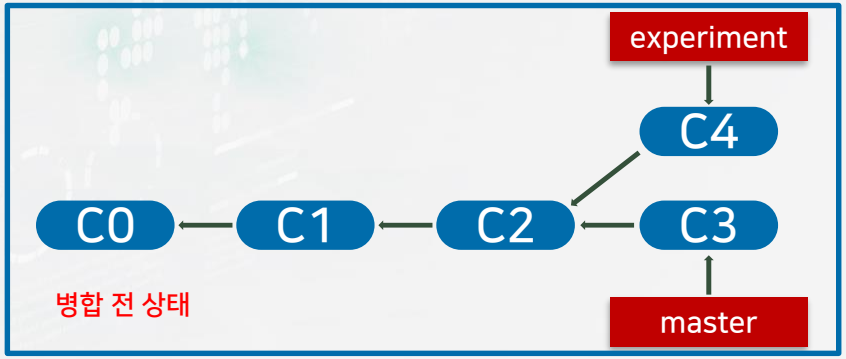
1. 3-way 병합

- $ git checkout mater
- $ git merge experiment

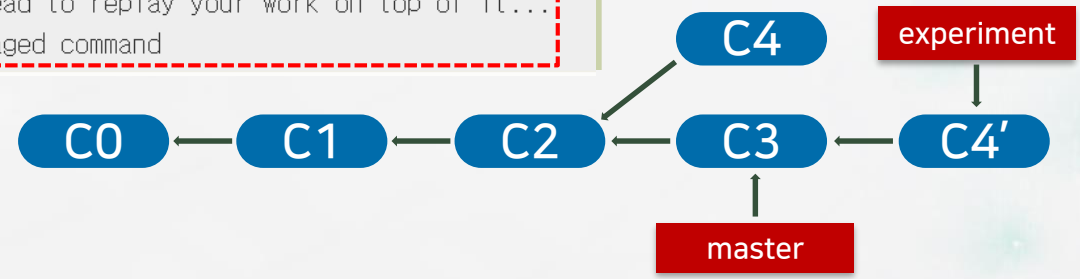


2 3-way, fast-forward와 rebase 비교

2. Rebase 병합



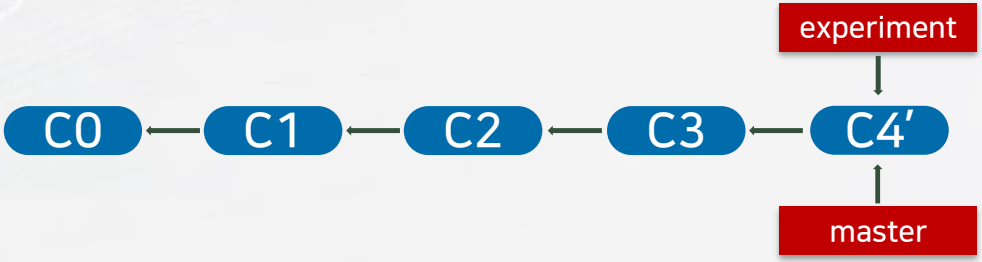

```
$ git checkout experiment
$ git rebase master
First, rewinding head to replay your work on top of it...
Applying: added staged command
```



2 3-way, fast-forward와 rebase 비교

2. Rebase 병합

```
$ git checkout master
$ git merge experiment
```



2 3-way, fast-forward와 rebase 비교

$ git rebase <newparent> <branch>

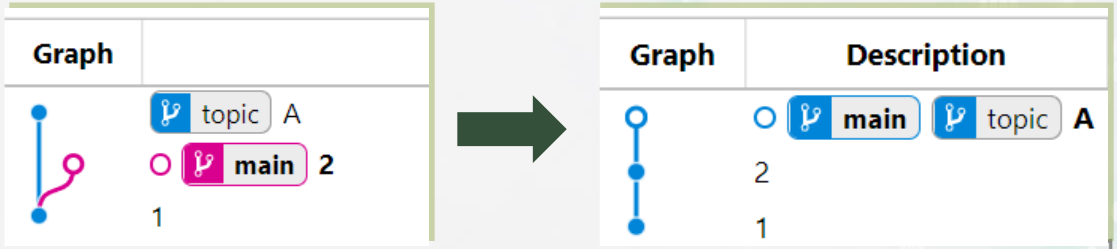
일반적 rebase 방법

- 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

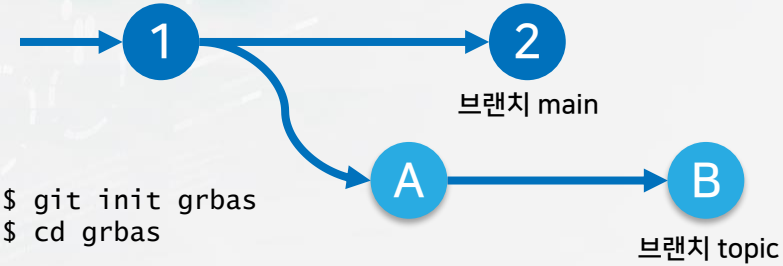
$ git rebase <newparent> <branch>
<branch>에서 선행으로 <newparent> 브랜치의
최근 커밋 이후에 <branch>를 배치해 선형 병합



2 3-way, fast-forward와 rebase 비교

[실습] Rebase 이전

다음 상태



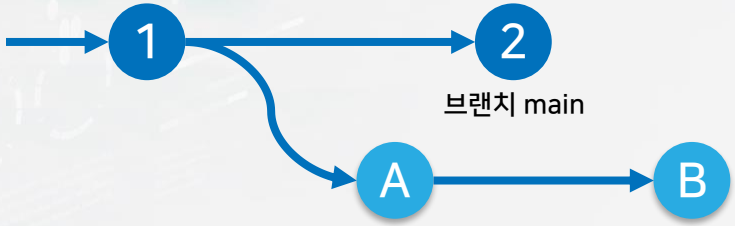

```
$ git init grbas
$ cd grbas

$ echo 111 > f
$ git add f
$ git commit -m 1

$ git checkout -b topic
$ echo aaa > g
$ git add g
$ git commit -m A
$ echo bbb >> g
$ git commit -am B

$ git checkout main
$ echo 222 >> f
$ git commit -am 2
```

```
$ git log --graph --oneline
--all
* f297212 (HEAD -> main) 2
| * 2147770 (topic) B
| * 66517c6 A
|/
* 0f1745a 1
```

2 3-way, fast-forward와 rebase 비교

Rebase 수행

브랜치 topic에서 수행

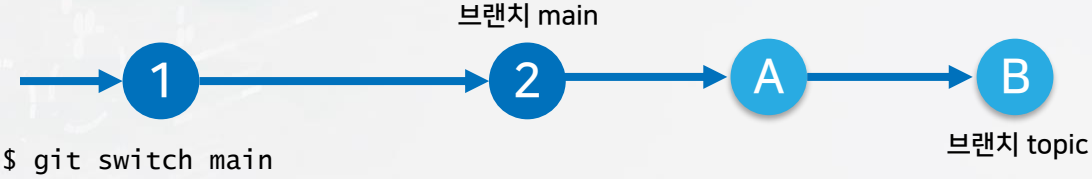



```
$ git switch topic
$ git rebase main
Successfully rebased and updated refs/heads/topic.

$ git log --graph --oneline --all
* 3ef33a8 (HEAD -> topic) B
* 20c99aa A
* f297212 (main) 2
* 0f1745a 1
```

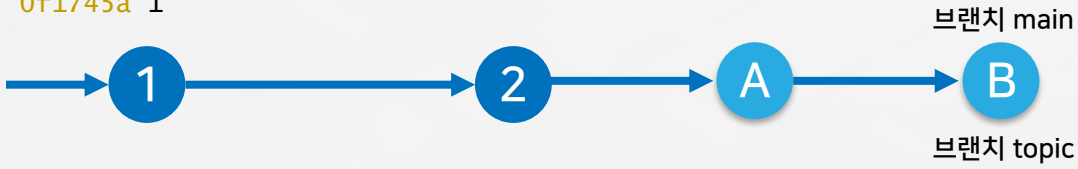
2 3-way, fast-forward와 rebase 비교

Rebase 수행

브랜치 main에서 브랜치 topic으로 빨리 감기




```
$ git switch main

$ git merge topic
Updating f297212..3ef33a8
Fast-forward
g | 2 ++
1 file changed, 2 insertions(+)
create mode 100644 g

$ git log --graph --oneline --all
* 3ef33a8 (HEAD -> main, topic) B
* 20c99aa A
* f297212 2
* 0f1745a 1
```



Summary

기준 브랜치에서 main 브랜치 rebase 병합

- $ git checkout topic
- $ git merge main

다시 main을 돌아와 fast-forward 병합 진행

- $ git checkout main
- $ git merge topic