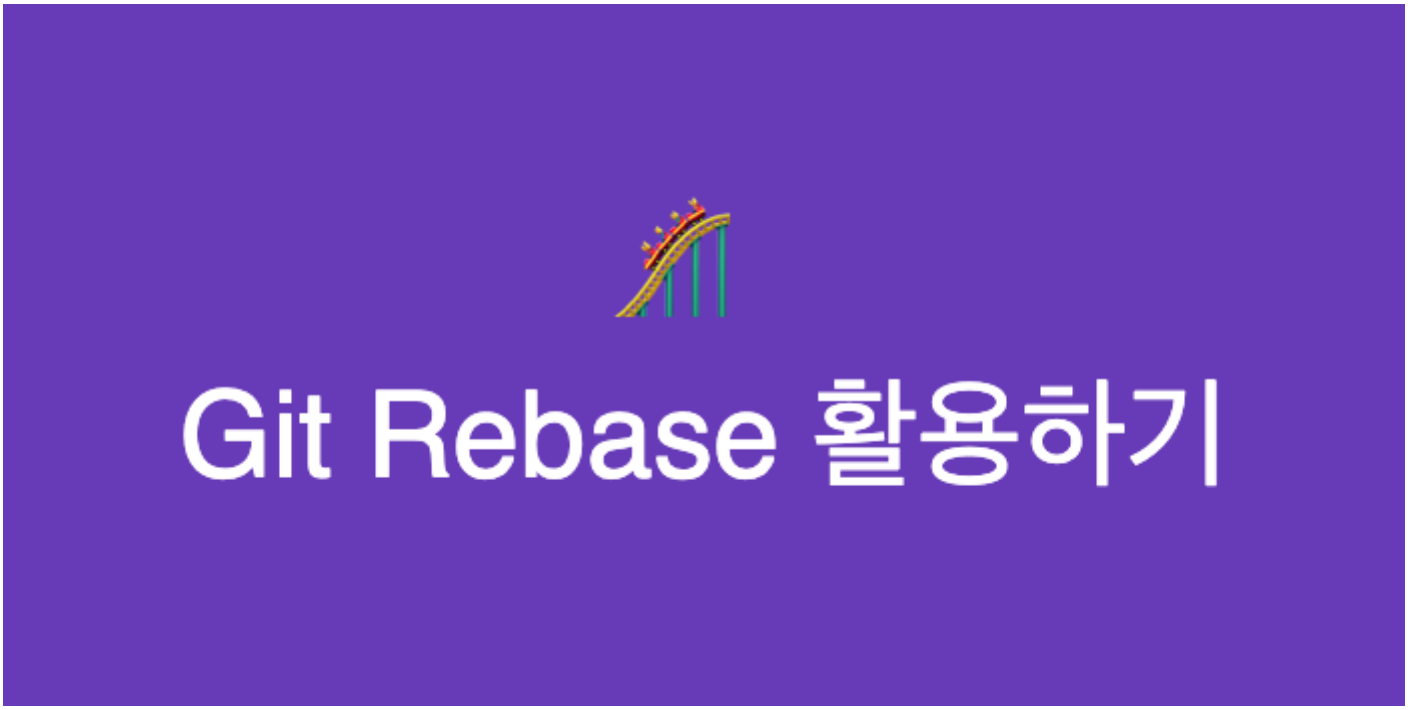
Git Rebase 활용하기

godori · 2019년 7월 25일

3 way merge branch git merge rebase



Read as Dev Post [Eng]

> Git의 Rebase는 다양한 쓰임이 있습니다만, 이번 포스트에서는 브랜치 병합시 Rebase를 활용하는 방법과 그 과정에 대해 자세히 알아보겠습니다.

브랜치 병합 전략

두개의 브랜치가 존재하는 상황일 때, 하나의 브랜치에서 다른 브랜치로 합치게 되는 경우 Git에서는 일반적으로 다음 두 가지 방법을 사용할 수 있습니다.

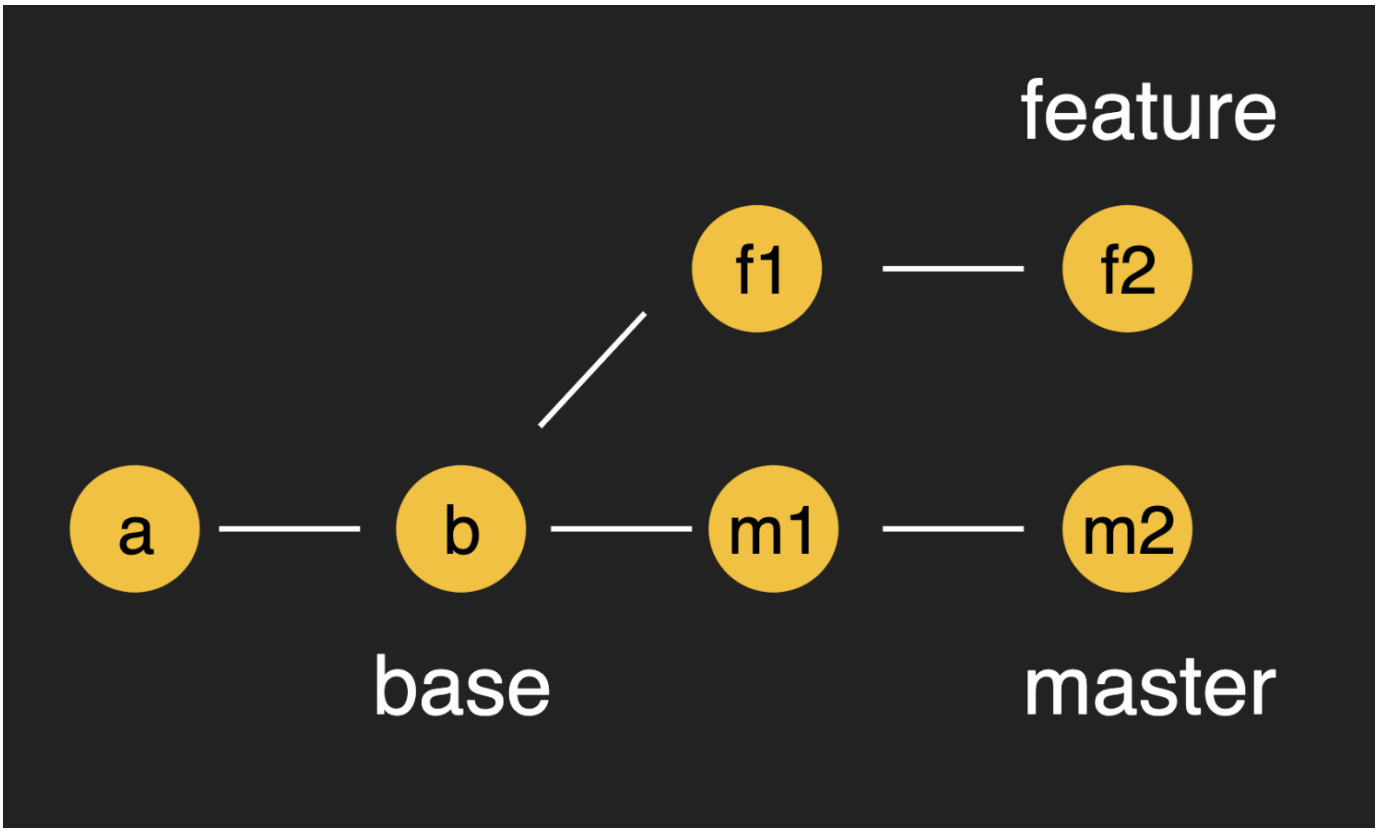
1. Merge
2. Rebase

자주 사용하는 Merge에 비해 Rebase는 조금 생소하신 분들이 많을지도 모르겠습니다. Rebase를 알아보기에 앞서 먼저 우리에게 익숙한 Merge에 대해 간단히 살펴보겠습니다.

Merge

Merge 브랜치에서 사용하는 전략은 각 브랜치의 **마지막 커밋 두 개**와 **공통 조상**의 총 **3개의 커밋**을 이용하는 `3-way merge` 를 수행하여 새로운 커밋을 만들어내는 것입니다.

가령, 다음 그림에서 보이는 feature와 master의 마지막 커밋은 각각 `f2` 와 `m2` , 그리고 공통 조상(base)은 커밋 `b` 입니다. 따라서, 이 세 커밋으로 새 커밋을 만들게 됩니다. (라임이 기가막히죠)



그렇다면 여기서 `3-way merge` 는 정확히 어떤 과정일까요? 마지막 커밋 두개만 비교해서 2-way merge를 하면 안 되는 걸까요?

3-way-merge

비교를 위해 필요한 3개의 커밋을 다시 정리하면 필요한 것은 다음 세 가지 커밋입니다.

1. 내 브랜치 커밋
2. 남의 브랜치 커밋
3. 두 브랜치의 공통 조상이 되는 커밋

이제 3-way merge가 효율적인 이유를 알아보겠습니다.

우선, 공통 조상이 되는 Base에 커밋되어 변경된 부분이 a, b, c, d라고 가정하고 다음 표에 기록해 두겠습니다.

My	Base	Other
	a	
	b	
	c	
	d	

내 브랜치(My)와 남의 브랜치(Other)에서 변경된 내역은 각각 아래의 표에 적힌 것과 같습니다.

My	Base	Other
a	a	a'
b	b	b
c'	c	c''
d'	d	d

첫 번째 a 를 살펴보면 내 브랜치에서는 a부분에 대해 변경 사항이 없이 그대로 a이며, 다른 사람은 a'로 변경했습니다. 두 번째 b에서는 양쪽 다 변경되지 않았으며, c는 둘 다 서로 다른 내용으로 변경했습니다. 마지막 d 부분은 내 쪽에서만 변경했습니다.

이때, Base가 되는 공통 조상 커밋이 없고 나와 다른 사람의 브랜치만 비교하여 Merge를 한다고 해봅시다.

My	Other	Merge
a	a'	?
b	b	b
c'	c''	?
d	d'	?

양쪽에서 동일하게 관찰되는 b 부분을 제외하고는 a가 원래는 a였는지 a'였는지 확실하게 정하기가 어렵습니다. 따라서 충돌이 난 것인지의 여부도 알 수가 없는 것이죠.

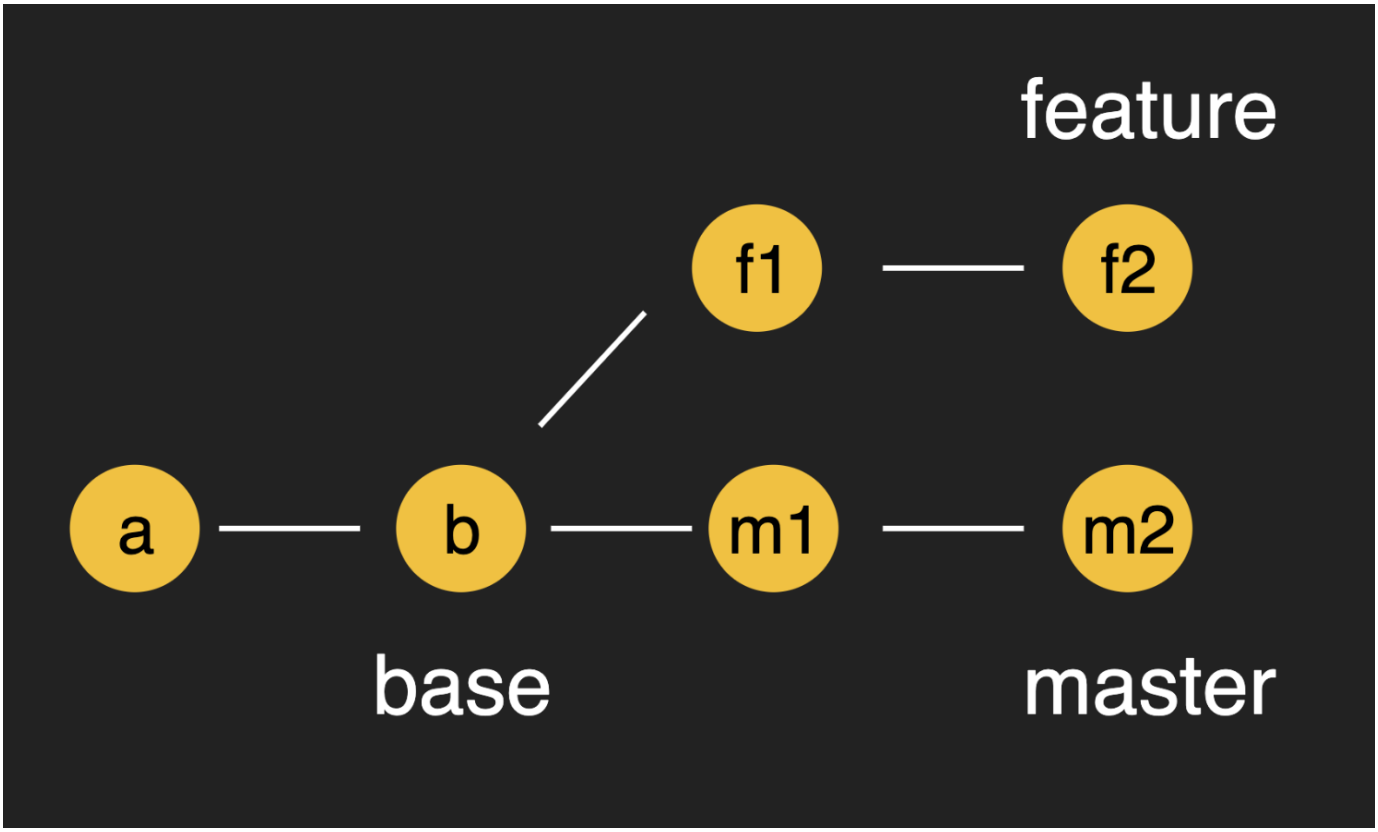
하지만 Base 커밋을 함께 비교하여 3-way merge를 수행하면 다음 표와 같이 Merge 커밋의 상태를 보다 **명확하게 결정할 수 있게 됩니다.**

My	Base	Other	Merge
a	a	a'	a'
b	b	b	b
c'	c	c''	*conflict*
d'	d	d	d'

다시 정리하면, git은 Merge를 할 때 **각 브랜치의 마지막 커밋 두 개, 브랜치의 공통 조상 커밋** 총 3개의 커밋을 비교하여 새로운 커밋을 만들어 병합을 수행합니다.

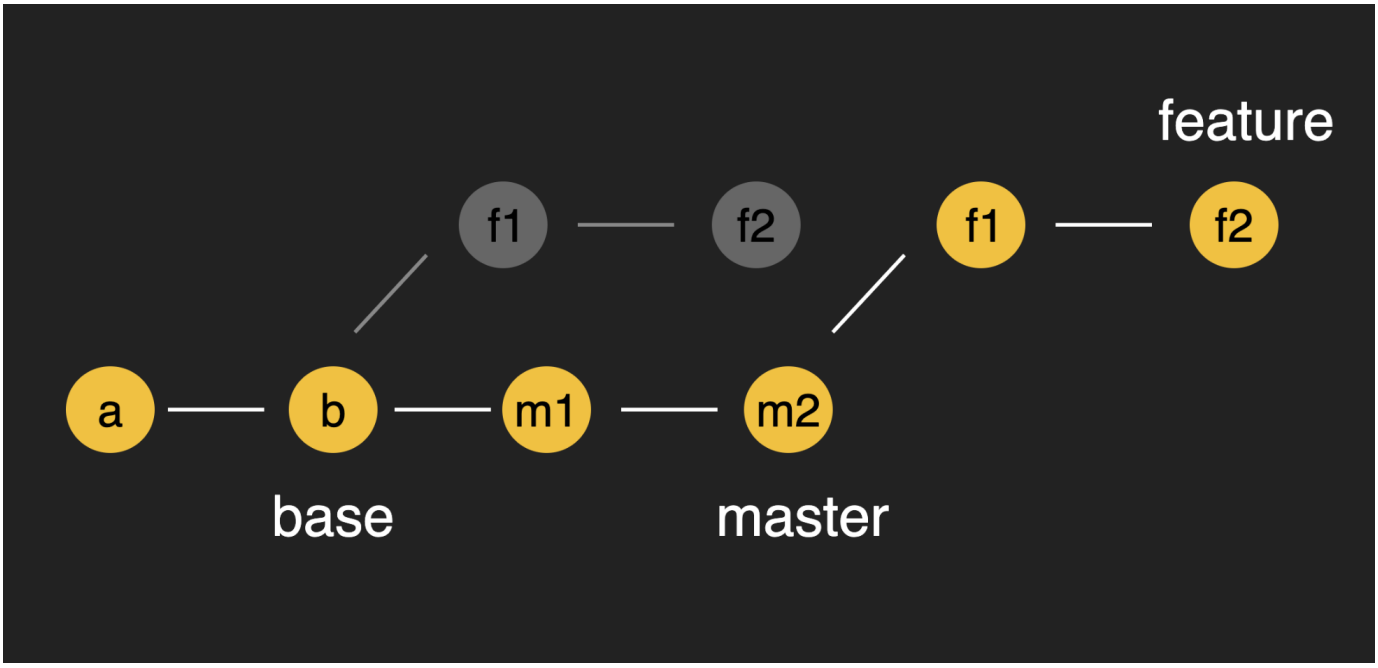
Rebase

두 브랜치를 합치는 두 번째 방법인 Rebase는 Merge와는 다르게 이름 그대로 브랜치의 공통 조상이 되는 base를 다른 브랜치의 커밋 지점으로 바꾸는 것입니다. 두 브랜치의 그림을 다시 보겠습니다.



우리가 rebase를 수행하여 얻고자 하는 목적은 master 브랜치의 마지막 커밋인 `m2` 이후에 feature 의 변경 사항인 `f1` 과 `f2` 가 일어난 것 처럼 보이게 하고 싶은 것입니다.

이런 모양으로요!



즉, feature의 base를 b가 아니라 m2로 **재설정(Rebase)**하는 것입니다.

feature를 master에 rebase 한다

=

feature의 master에 대한 공통 조상인 base를 master로 변경한다

Rebase의 기본 전략은 다음과 같습니다.

먼저 Rebase 하려는 브랜치 커밋들의 변경사항을 Patch라는 것으로 만든 다음에 어딘가에 저장해 둡니다. 그리고 이를 master 브랜치에 하나씩 적용하여 새로운 커밋을 만드는 것입니다.

feature를 master 브랜치로 Rebase하는 명령어를 살펴보면 일련의 단계를 거쳐 두 브랜치의 병합이 완료됩니다.

```
1. feature 브랜치로 checkout
2. master 브랜치로 rebase
3. feature 브랜치를 master로 fast-forward merge
```

```
$ git checkout feature
$ git rebase master
First, rewinding head to replay your work on top of it...
Applying: f1
Applying: f2
$ git merge feature
```

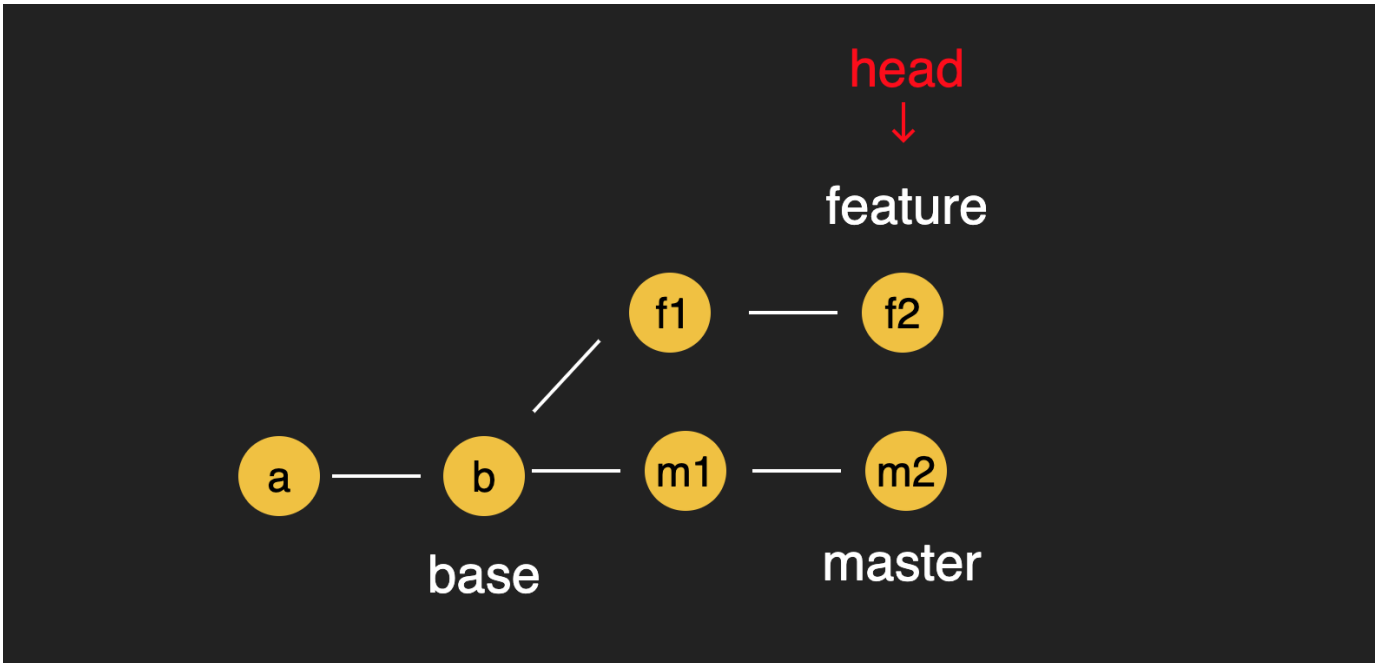
> Rebase를 하는데 왜 Merge가 나올까요? fast-forward merge에 대해서는 이 영상을 참고하세요.

이제 rebase 과정을 하나씩 자세히 살펴보겠습니다.

Step 1

```
git checkout feature
```

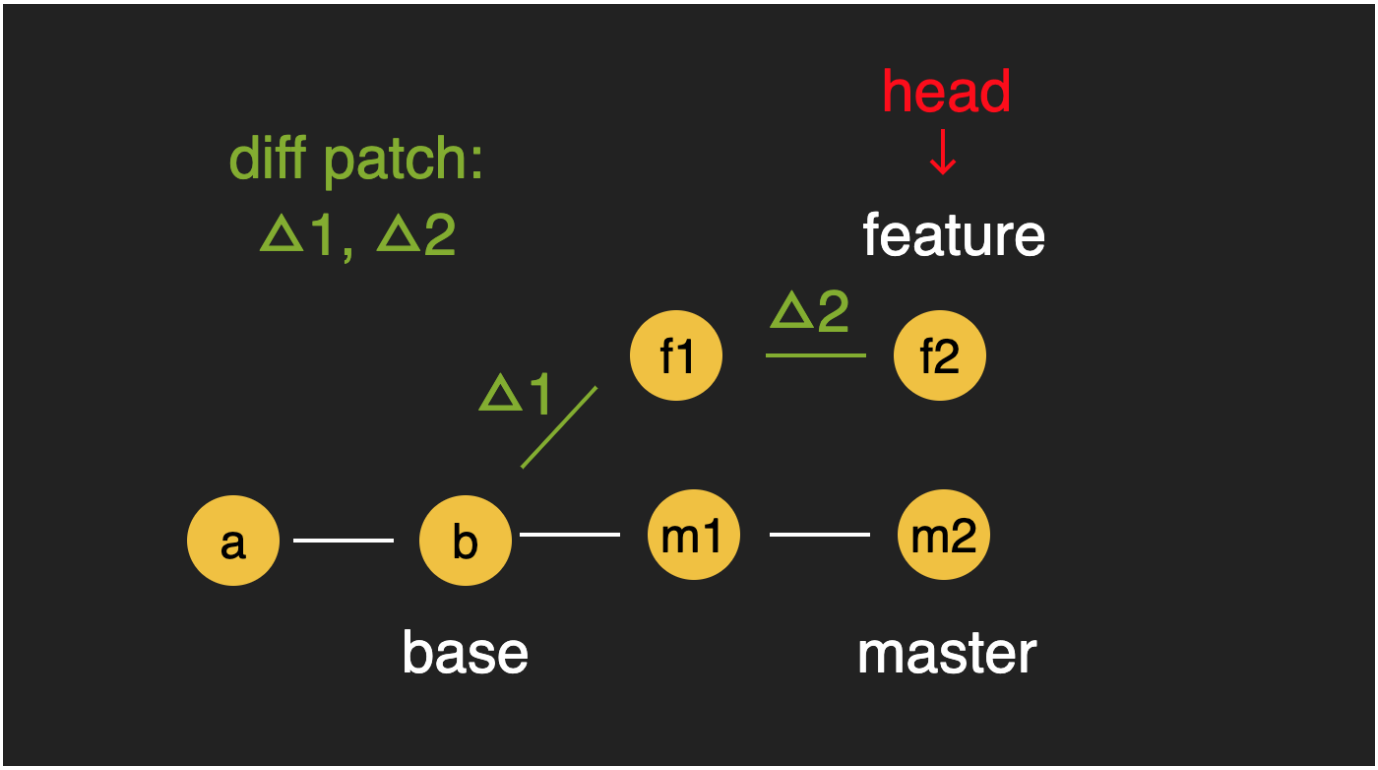
feature 브랜치로 체크아웃한 상태입니다. head는 feature를 가리키고 있습니다.



Step 2

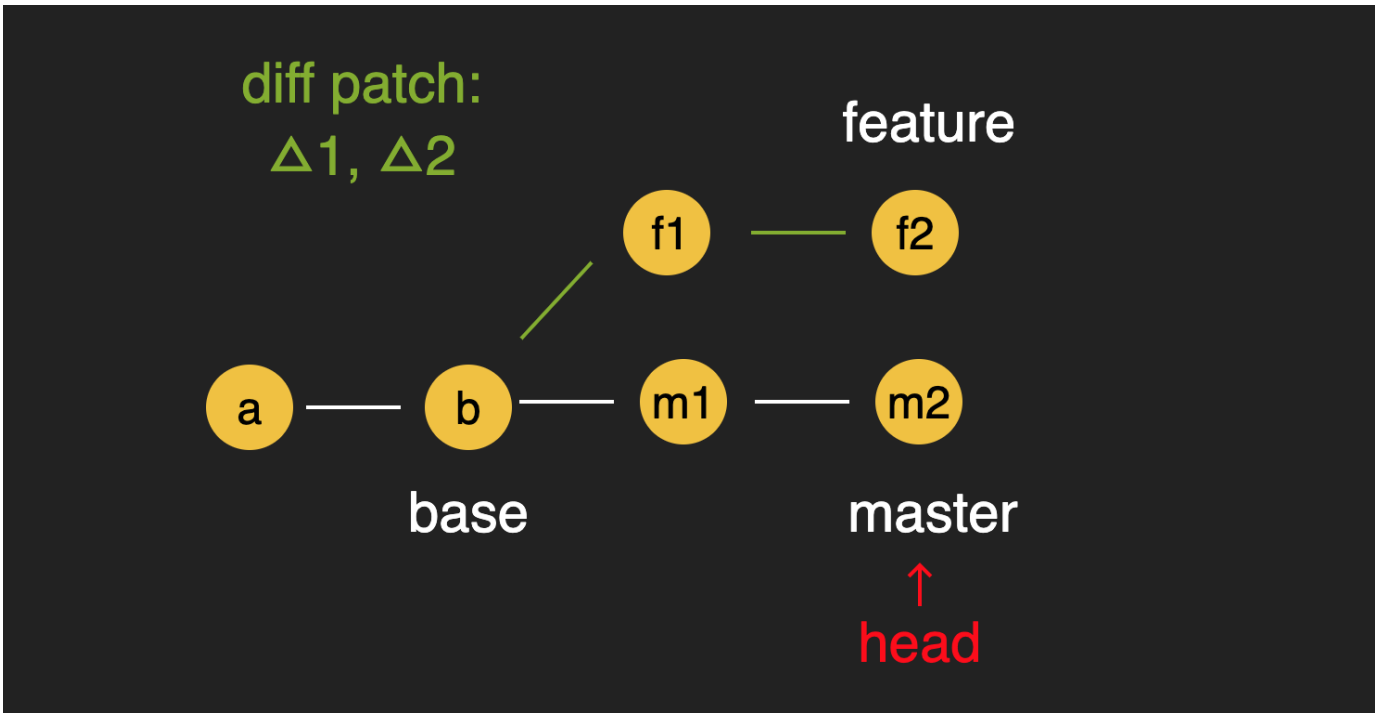
```
git rebase master
```

master와 feature의 공통 조상이 되는 base 커밋부터 현재 브랜치까지의 변경 사항(`△1` , `△2`)을 구해서 patch로 저장해 둡니다.



Step 3

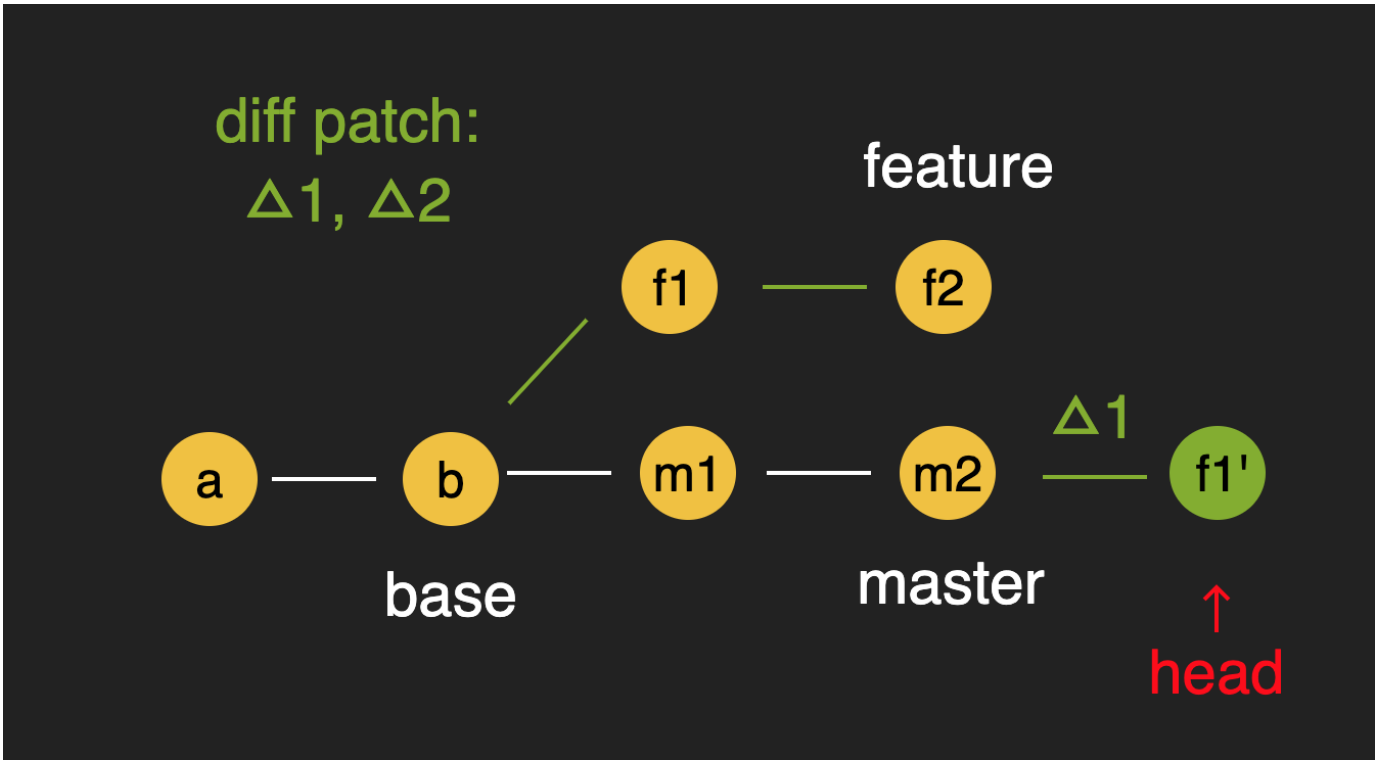
head를 master로 변경합니다.



Step 4

```
Applying f1
```

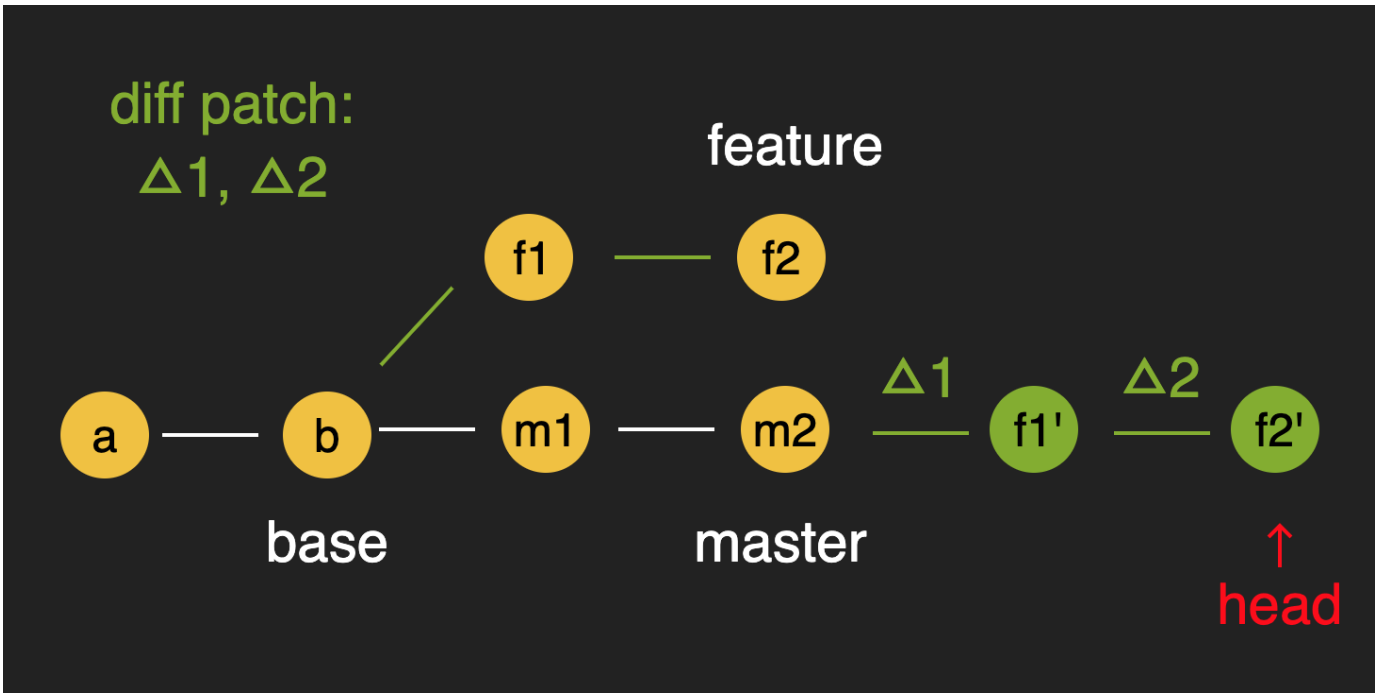
head가 현재 가리키고 있는 `m2` 에 변경사항 `△1` 을 적용하여 새로운 커밋 `f1'` 을 생성합니다.



Step 5

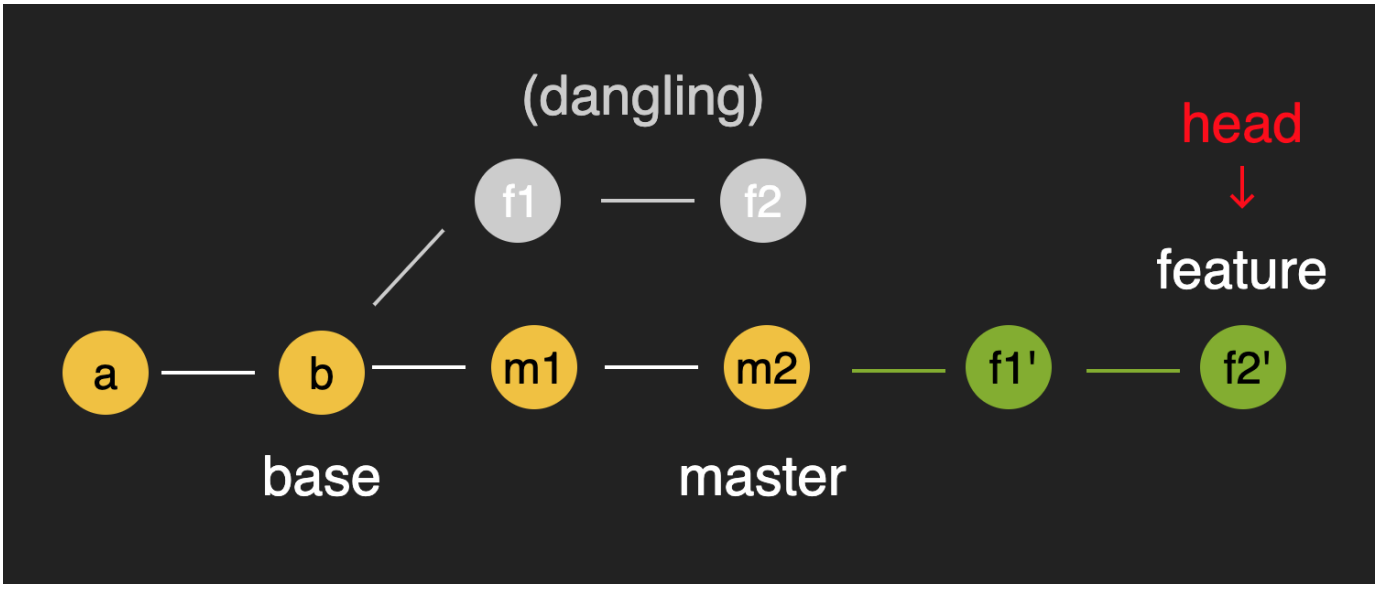
```
Applying f2
```

`f1'` 에 변경사항 `△2` 을 적용하여 새로운 커밋 `f2'` 을 생성합니다.



Step 6

이제 feature가 f2'를 가리키도록 합니다.

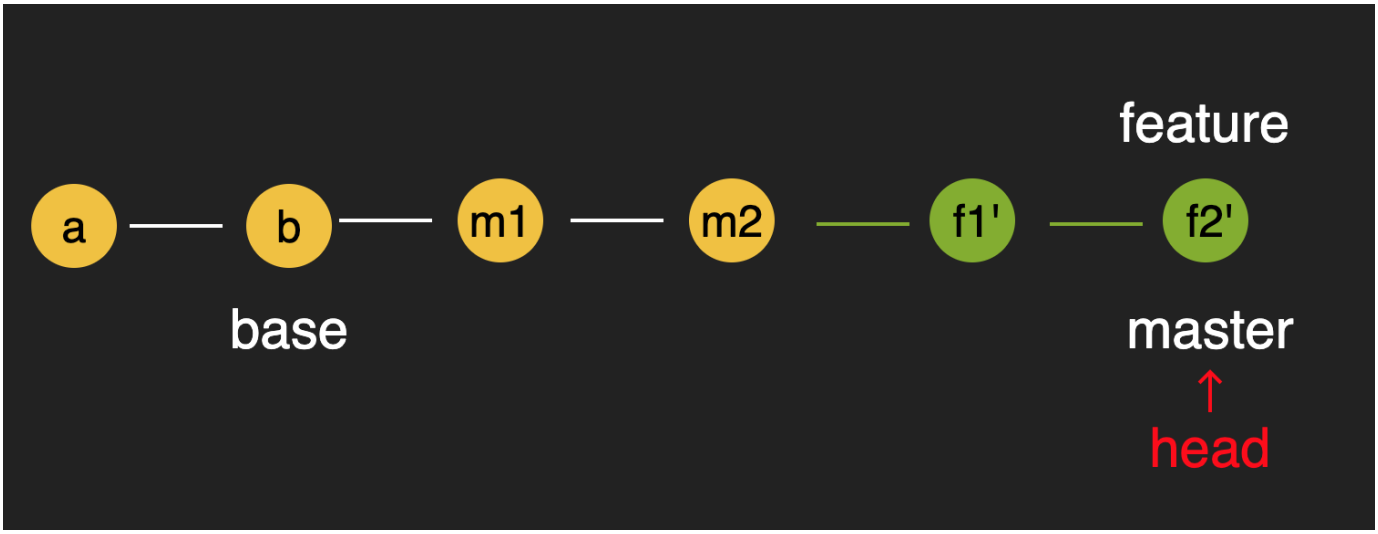


> f1과 f2는 저장소 내에는 존재하지만, tag나 branch에서 가리킬 수 없는 dangling 상태가 되며, dangling 된 커밋은 가비지 콜렉션의 대상이 됩니다.

Step 7

```
git merge feature
```

feature를 master로 fast-forward merge하여 완료합니다.



브랜치 병합: Merge vs Rebase

이제까지 두 가지 방법의 브랜치 병합 전략을 살펴보았습니다. Merge를 사용할 지, Rebase를 사용할 지는 프로젝트의 히스토리를 어떤 것으로 생각하느냐에 따라 달라집니다.

먼저 Merge의 경우 히스토리란 작업한 내용의 사실을 기록한 것입니다. Merge로 브랜치를 병합하게 되면 커밋 내역에 Merge commit이 추가로 남게 됩니다. 따라서 Merge를 사용하면 브랜치가 생기고 병합되는 모든 작업 내용을 그대로 기록하게 됩니다.

다음으로 Rebase의 경우는 브랜치를 병합할 때 이런 Merge commit을 남기지 않으므로, 마치 다른 브랜치는 없었던 것처럼 프로젝트의 작업 내용이 하나의 흐름으로 유지됩니다.

브랜치를 합칠 때 Merge를 써야 하는지 Rebase를 써야 하는지에 대해서는 정답이 없습니다.
프로젝트나 팀의 상황에 따라 다른 전략을 사용할 수 있습니다. 다만, 로컬에서는 히스토리 정리를 위해 Rebase를 할 수도 있지만 이미 원격 저장소에 Push된 커밋은 Rebase를 하지 않는 것이 일반적입니다.

참고 자료 및 추천 링크

- Pro Git - Rebasing
- GIT4 Open Tutorials
- 3-way merge Open Tutorials
- 오픈소스를 쓰려는 자, 리베이스의 무게를 견뎌라 / [영상]
  2017 Deview 컨퍼런스 세션 내용으로, 크로미움 엔진 오픈소스를 사용하는 웨일 브라우저를 개발하면서 두 차례의 리베이스를 진행하는 과정에서의 고군분투가 담긴 발표 자료입니다.

GODORI

Love 🎧
Play 🎮
Drink