오픈소스 소프트웨어 개발 – 18

(Online) Git & Github - advanced

광운대학교 이윤상
2017년 2학기

이번 시간에 할 것

- Rebase & merge
- History 정리하기 (Squashing)
- Github "Merge pull request" Options
- 과거의 버전으로 돌아가기

Rebase & merge

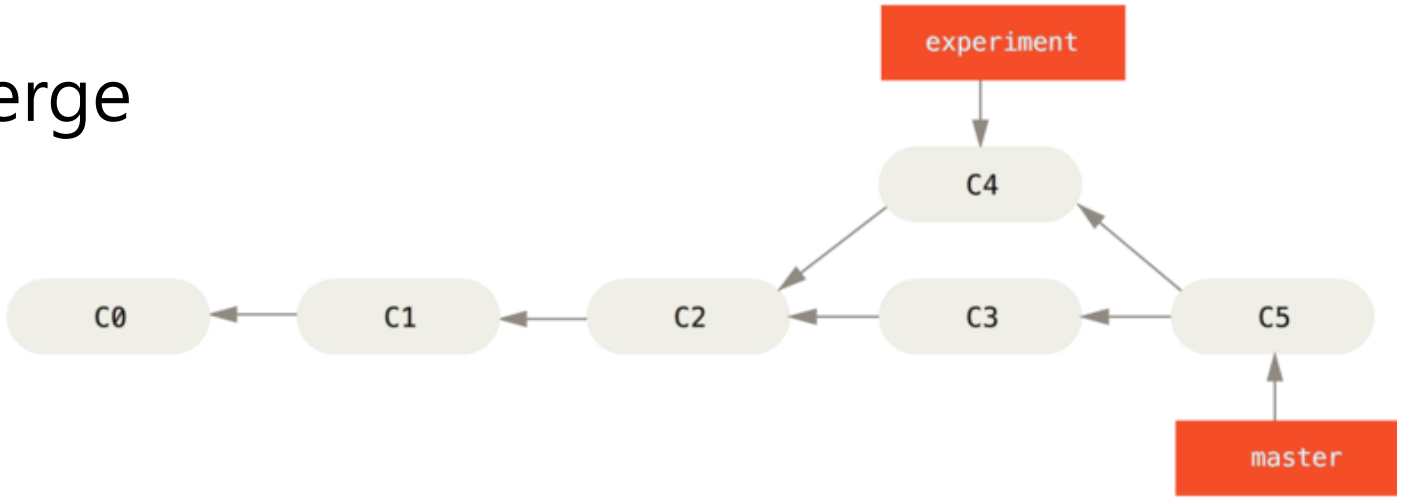
- Rebase와 merge 모두 두 개의 branch를 합칠 때 쓸 수 있다.

- 동작하는 방식은 완전히 다르다 (다음 슬라이드).

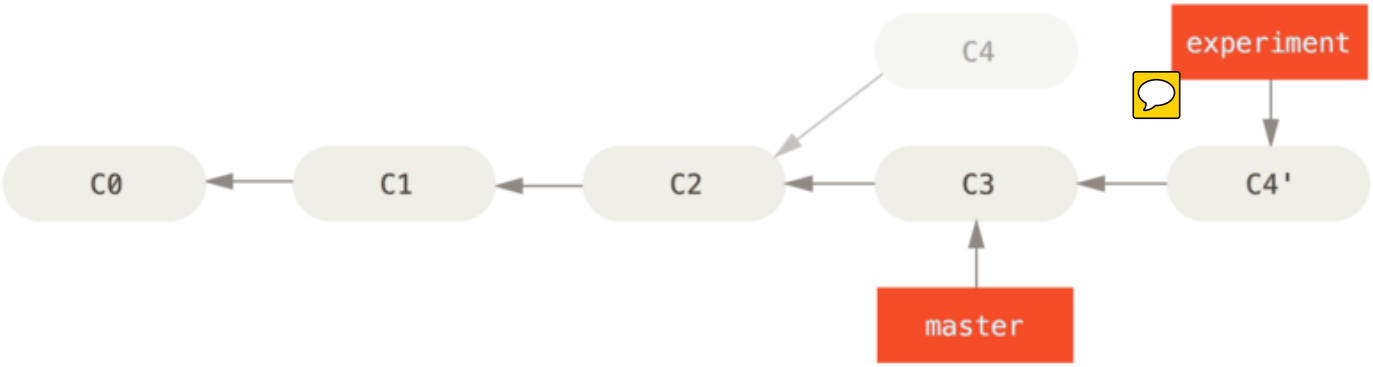
- Rebase가 좀 더 깨끗한 history를 만든다.

Rebase & Merge

- Merge



- Rebase



Rebase & Merge

```
          A---B---C topic
         /
    D---E---F---G master
```

- Merge

```
          A---B---C topic
         /         \
    D---E---F---G---E master
```

- Rebase

```
                    A'--B'--C' topic
                   /
    D---E---F---G master
```

Rebase & Merge

- Rebase와 merge 중 어떤 것을 사용하는 것이 나은지는 상황에 따라 다르다.
- Rebase 역시 merge와 마찬가지로 충돌이 발생할 수 있다.
- 이 경우, 각 commit의 rebase 진행 중 멈출 때마다 해당 commit에 발생한 충돌을 해결한 후 git rebase --continue로 계속 진행해야 한다.

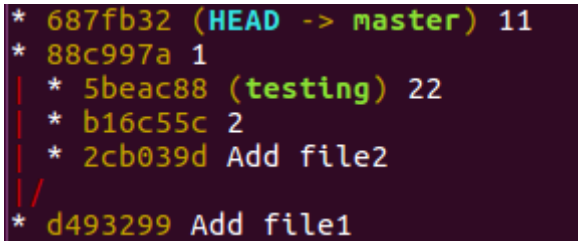
[실습] Rebase

(Shell)

```
# 적당한 디렉터리 만든 후,
git init
vi file1.txt
git add .
git commit  # Add file1

git checkout -b testing
vi file2.txt
git add .
git commit  # Add file2
vi file2.txt
git commit -a -m '2'
vi file2.txt
git commit -a -m '22'

git checkout master
vi file1.txt
git commit -a -m '1'
vi file1.txt
git commit -a -m '11'
```

[실습] Rebase

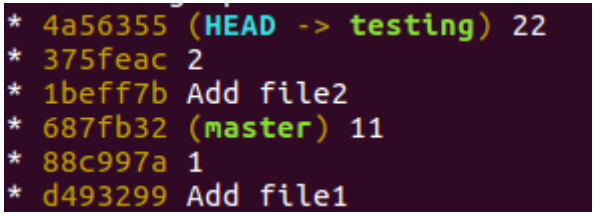
rebase 전



(Shell)

```
git log --branches --decorate --graph --oneline
git checkout testing
git rebase master
git log --branches --decorate --graph --oneline
```

rebase 후



Rebase 주의사항

- **원칙: 이미 push한 commit은 수정 / 삭제하지 말 것**

- Rebase 역시 이미 push한 commit에 대해서는 하면 안 된다 (pull request용 branch는 예외).

History 정리하기 (Squashing)

- 어느 쪽이 더 보기 좋은가?

```
7hgf8978g9... Added new slideshow feature, JIRA # 848394839
85493g2458... Fixed slideshow display issue in ie
gh354354gh... wip, done for the week
789fdfffdf... minor alignment issue
787g8fgf78... hotfix for #5849564648
9080gf6567... implemented feature # 65896859
gh34839843... minor fix (typo) for 3rd test
```

```
7hgf8978g9... 8483948393 Added new slideshow feature
787g8fgf78... 5849564648 Hotfix for android display issue
9080gf6567... 6589685988 Implemented pop-up to select language
```

- **git rebase -i** <commit범위>: 위와 같이 정리되지 않은 커밋들을 합치고 수정해서 아래처럼 정리할 수 있는 명령

[실습] History 정리하기 (Squashing)

(Shell)

```
git checkout master
git merge testing  # 우선 fast-forward merge를 하고,

git rebase -i HEAD~5  # 가장 최근 commit포함 5개의 commit을 정리
```

- Vim이 실행되어 해당 범위 중 어떤 commit을 합치고, 어떤 commit을 놔둘 것인지 입력하게 함.
- 'squash'를 입력하면 해당 commit과 바로 이전 commit을 합친다.
- 2번째, 5번째 commit을 squash 혹은 s로 변경 후 저장 & 종료하면
- pick으로 놔둔 commit들의 commit message를 수정하는 vim이 실행

[실습] History 정리하기 (Squashing)

정리 전

```
(HEAD -> master) 22
2
Add file2
11
1
Add file1
```

정리 후

```
(HEAD -> master) Edit file2
Add file2
Edit file1
Add file1
```

History 정리하기 (Squashing) 주의사항

- **원칙: 이미 push한 commit은 수정 / 삭제하지 말 것**

- Squashing 역시 이미 push한 commit에 대해서 하면 안 된다 (pull request용 branch는 예외).

Rebase, Squashing과 Pull Request

- 예전에는 Github에서 pull request할 때 최종 버전을 아래와 같이 정리해서 보내는 것이 일반적이었음.
  - official repository의 최신 commit 이후로 rebase
  - 한 개의 commit으로 squash

  - 이 때 pull request용 branch는 각 개발자의 forked repository에 이미 push했어도 rebase 혹은 squash 한 후 git push --force로 덮어쓰기 해도 무방하다고 보는 시각이 많음.

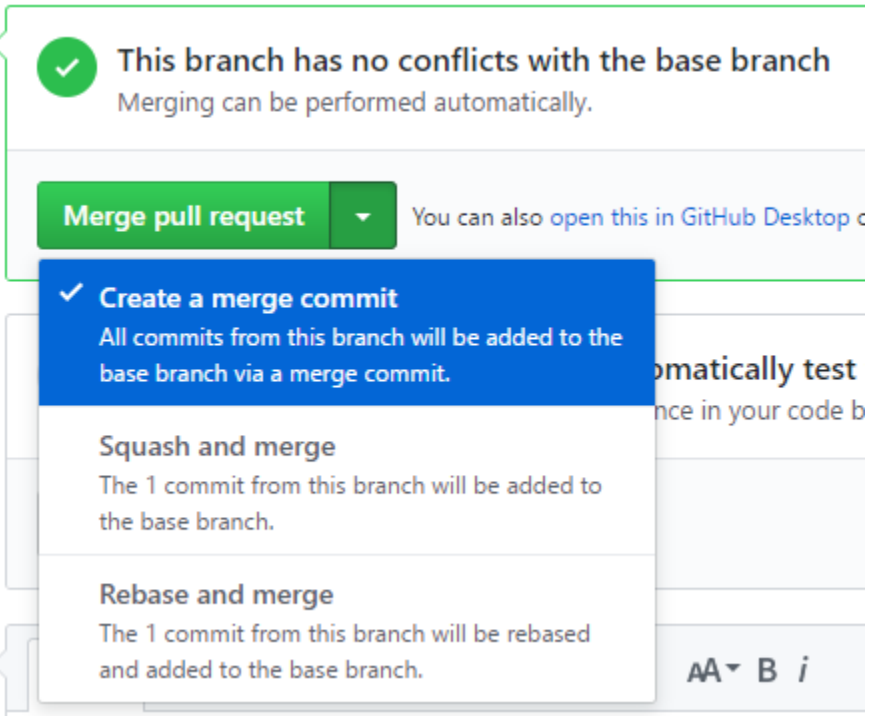
Rebase, Squashing과 Pull Request

- 2016년 Github에 "Squash and merge"와 "Rebase and merge" 버튼이 추가됨.
  - Contributor가 직접 rebase와 squash로 PR을 정리하지 않아도, official repository의 maintainer가 버튼을 한 번 클릭하는 것만으로 위와 같은 정리가 된 후 merge 할 수 있게 되었음.

- 하지만 지금도 rebase 혹은 squash를 요구하는 프로젝트도 있을 수 있으니, 프로젝트의 contribution guideline을 잘 살펴봐야 함.

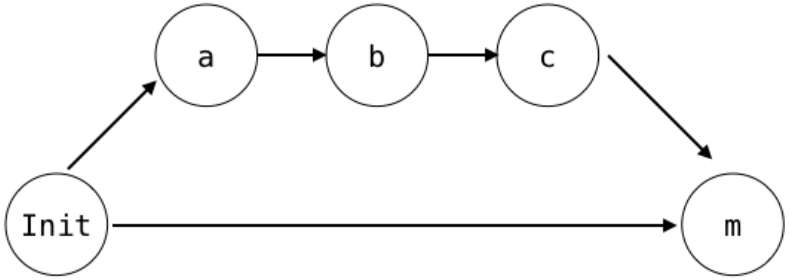
Github "Merge pull request" Options

- Github에서 프로젝트 운영자가 PR을 Merge하는 방법은 3가지가 있다.

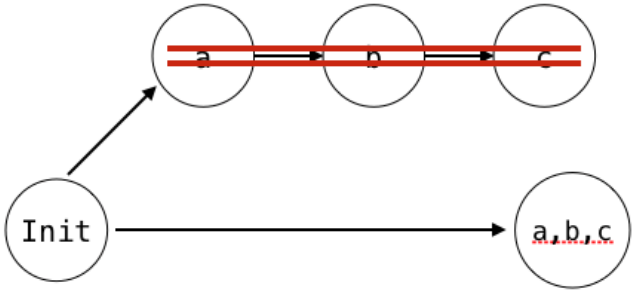


Github "Merge pull request" Options

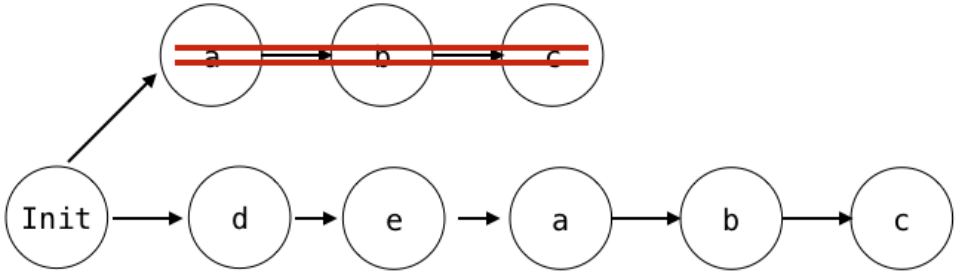
- Create a merge commit



- **Squash and merge**



- Rebase and merge



Github "Merge pull request" Options

- "Squash and merge"은 squash와 rebase를 함께 하는 개념.

- History를 가장 깔끔하게 유지할 수 있는 방법
  - Pull request당 commit 1개로 반영됨
  - History에 별도 branch graph가 그려지지 않음

- 그럼에도 프로젝트 별 contribution guide를 잘 따르도록 pull request를 작성해야 함.

과거의 버전으로 돌아가기

- Revert
  - 특정 commit을 취소하는 새로운 commit을 만들어 적용함.
  - 이미 push를 한 commit을 취소할 때
  - 혹은 history 중간의 특정 commit만 취소할 때

- Reset
  - 실제로 과거의 특정 commit이 저장소의 최신 commit이 되도록 HEAD를 변경
  - 아직 push 하지 않은 commit을 취소할 때

[실습] git revert

- **git revert** <commit id>
- **git revert** <commit id1>..<commit id2>

(Shell)

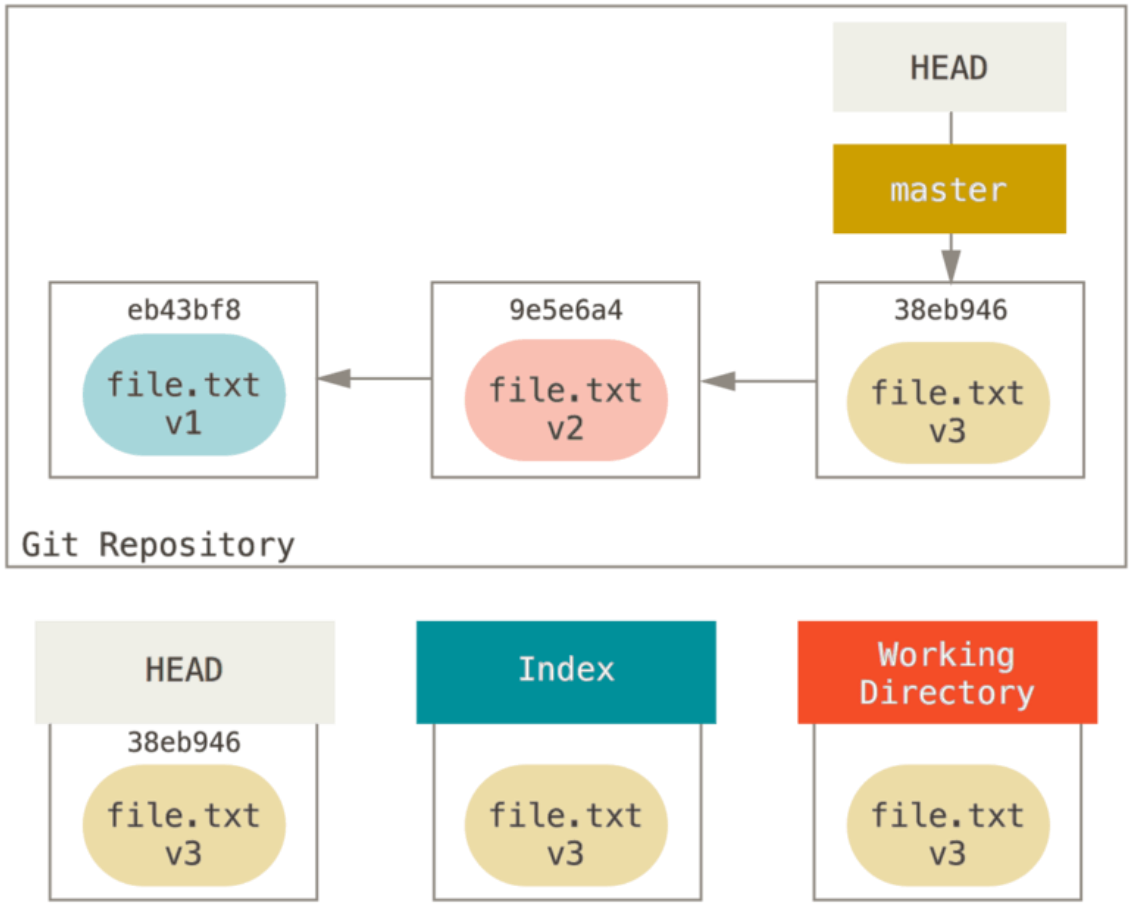
```
git log --branches --decorate --graph --oneline
git log -p  # 가장 마지막 commit의 내용 확인

git revert HEAD

git log --branches --decorate --graph --oneline
git log -p  # 추가된 revert commit의 내용 확인
```

git reset --soft, --mixed, --hard

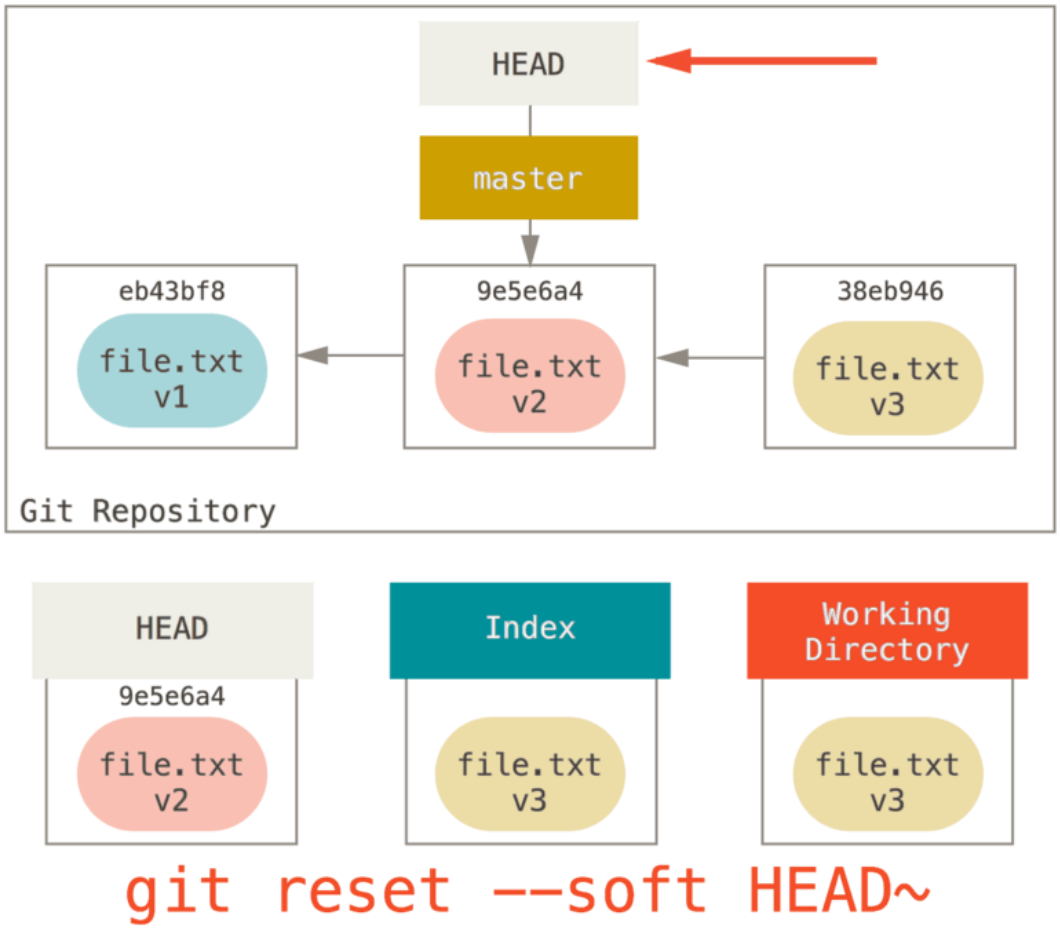
- **git reset** <commit id> : 해당 commit으로 상태를 되돌린다.
- 옵션에 따라 적용 범위가 다르다.

Working Directory Working Copy	Staging Area Index	Repository History
		git reset --soft
	git reset --mixed	
git reset --hard		

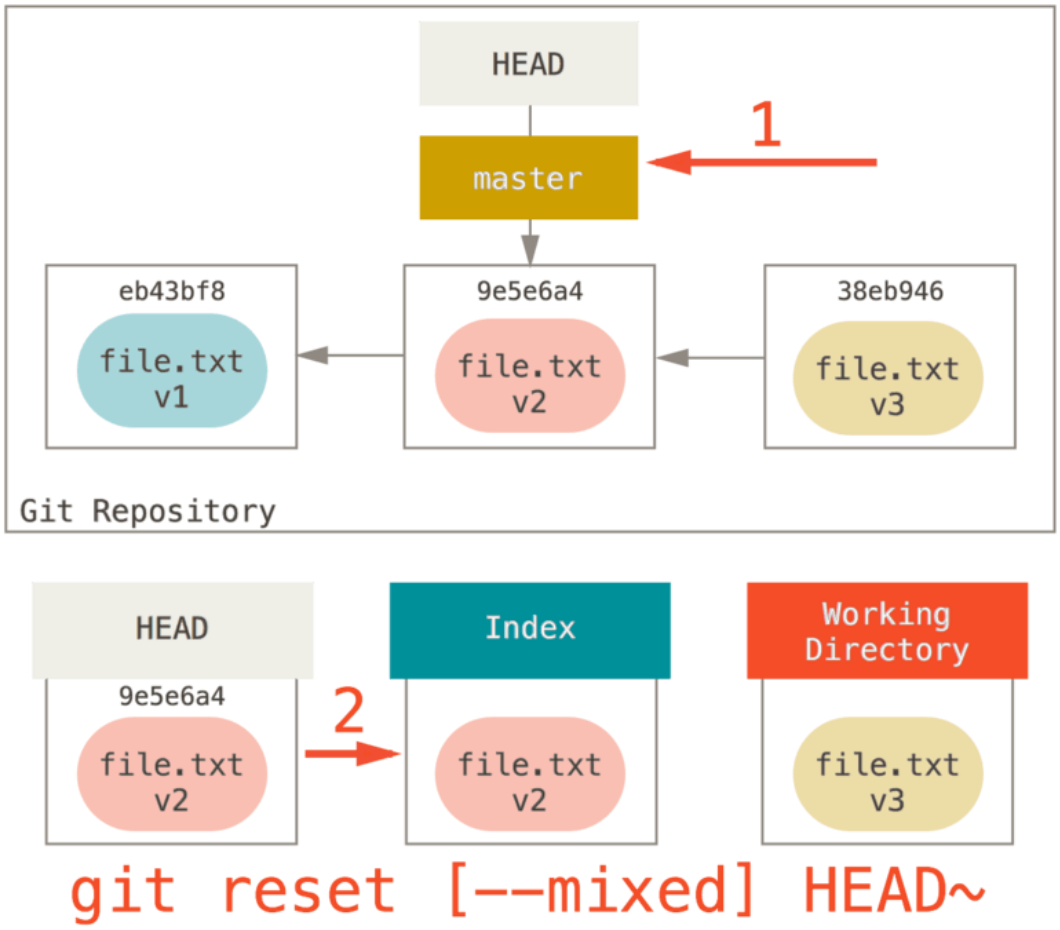
git reset의 예


HEAD
master
eb43bf8
file.txt
v1
9e5e6a4
file.txt
v2
38eb946
file.txt
v3
Git Repository
HEAD
38eb946
file.txt
v3
Index
file.txt
v3
Working
Directory
file.txt
v3

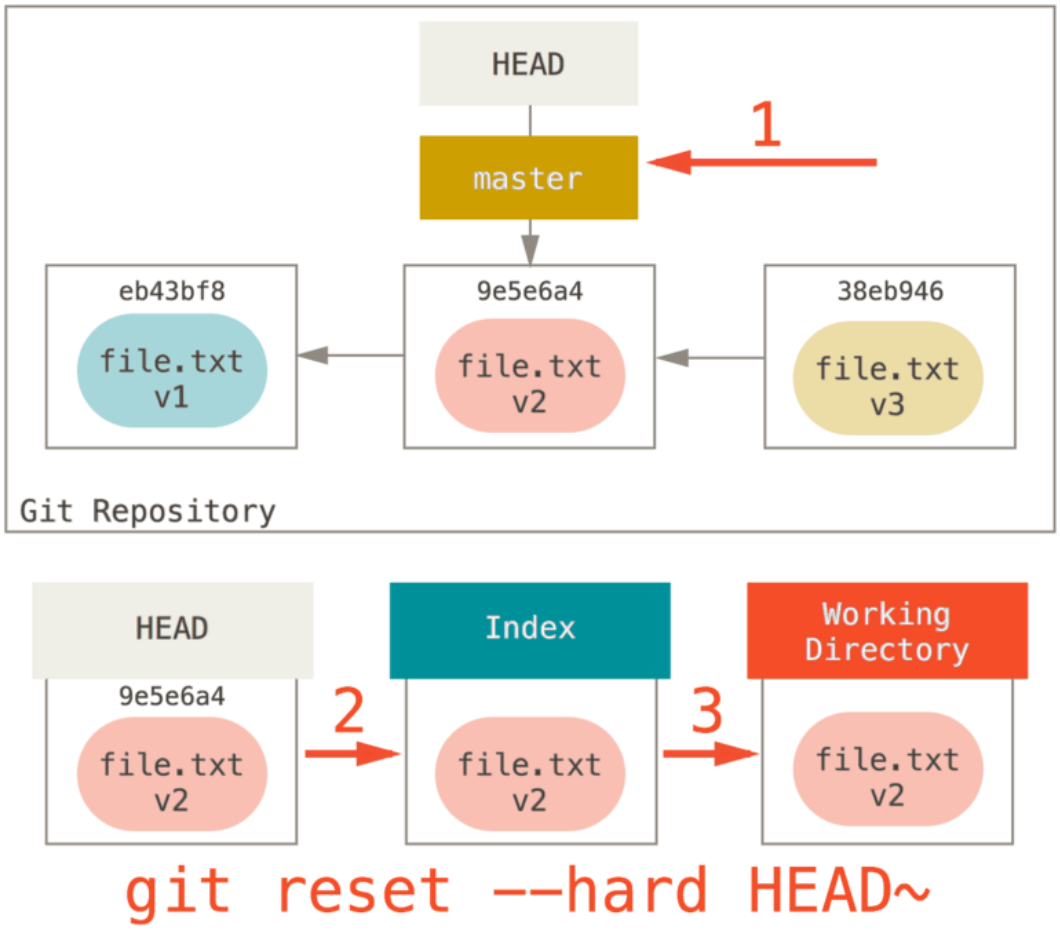
git reset의 예


HEAD
master
eb43bf8
file.txt
v1
9e5e6a4
file.txt
v2
38eb946
file.txt
v3
Git Repository
HEAD
9e5e6a4
file.txt
v2
Index
file.txt
v3
Working
Directory
file.txt
v3
git reset --soft HEAD~

git reset의 예


HEAD
master
1
eb43bf8
file.txt
v1
9e5e6a4
file.txt
v2
38eb946
file.txt
v3
Git Repository
HEAD
9e5e6a4
file.txt
v2
2
Index
file.txt
v2
Working
Directory
file.txt
v3
git reset [--mixed] HEAD~

git reset의 예


HEAD
master
1
eb43bf8
file.txt
v1
9e5e6a4
file.txt
v2
38eb946
file.txt
v3
Git Repository
HEAD
9e5e6a4
file.txt
v2
2
Index
file.txt
v2
3
Working
Directory
file.txt
v2
git reset --hard HEAD~

git reset 주의사항

- git reset --soft, --mixed는 현재 작업 중인 코드가 working directory 혹은 staging area에 남아 있는 반면,

- git reset --hard는 현재 작업 중인 코드를 전혀 남기지 않고 reset하므로, 주의해서 사용해야 함.

- 각자 git reset --soft, --mixed, --hard의 실습을 진행해보기 바람.