

Open source Software

오픈소스 소프트웨어

26. 커밋 이력 수정



학습 개요

1. `$ git commit --amend`
2. `$ git commit --amend -m 'new message'`
3. `$ git rebase --interactive HEAD~3`



학습 목표

1. 최신 HEAD의 커밋 메시지를 수정할 수 있다.
2. 최신 HEAD의 커밋 내용을 수정할 수 있다.
3. 명령 `rebase --interactive` 대화형으로 이전 여러 커밋을 수정할 수 있다.



LESSON 01

최신 커밋 수정



1 최신 커밋 수정

최근 커밋 메시지 수정

기본 편집기 설정 방법

- `$ git config --global core.editor 'code --wait'`

설정된 편집기로 최근 커밋 메시지 수정

- `$ git commit --amend`
 - 새로운 커밋 ID로 수정됨

최근 커밋 메시지를 직접 입력해 수정

- `$ git commit --amend -m "an updated commit message"`
 - 새로운 커밋 ID로 수정됨

1 최신 커밋 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`

1 최신 커밋 수정



[실습] 최신 커밋 메시지 수정



A를 AA로 수정

```
파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]
```

```
$ git init cupd
```

```
Initialized empty Git repository in C:/[smart git]/cupd/.git/
```

```
$ cd cupd
```

```
$ touch f
```

```
$ git add f
```

```
$ git commit -m A
```

```
[main (root-commit) 5346b5c] A
```

```
1 file changed, 0 insertions(+), 0 deletions(-)
```

```
create mode 100644 f
```

```
$ git log --oneline
```

```
5346b5c (HEAD -> main) A
```

```
$ git commit --amend
```

```
hint: Waiting for your editor to close the file...
```

```
vscode가 실행됨
```

```
[main e4736bd] AA
```

```
Date: Mon Jan 23 16:53:42 2023 +0900
```

```
1 file changed, 0 insertions(+), 0 deletions(-)
```

```
create mode 100644 f
```

```
$ git log --oneline
```

```
e4736bd (HEAD -> main) AA
```

```
≡ f
```

```
COMMIT_EDITMSG ●
```

```
C: > [smart git] > cupd > .git > COMMIT_EDITMSG
```

```
1 AA
2
3 # Please enter the commit message for your changes. Lines starting
4 # with '#' will be ignored, and an empty message aborts the commit.
5 #
6 # Date:      Mon Jan 23 16:53:42 2023 +0900
7 #
8 # On branch main
9 #
10 # Initial commit
11 #
12 # Changes to be committed:
13 #   new file:   f
14 #
15
```

**수정 후 저장(ctrl + s),
닫기(ctrl + F4)하면 표시**

1 최신 커밋 수정

[실습] 파일 추가 후 최신 커밋에 반영

파일 g h 추가한 후 반영

옵션 --amend

-m 'new message'

```
파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]/cupd (main)
```

```
$ touch g h
```

```
파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]/cupd (main)
```

```
$ git add g h
```

```
파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]/cupd (main)
```

```
$ git ls-files
```

```
f
```

```
g
```

```
h
```

```
파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]/cupd (main)
```

```
$ git commit --amend -m 'AA add'
```

```
[main 40cf763] AA add
```

```
Date: Mon Jan 23 16:53:42 2023 +0900
```

```
3 files changed, 0 insertions(+), 0 deletions(-)
```

```
create mode 100644 f
```

```
create mode 100644 g
```

```
create mode 100644 h
```

```
파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]/cupd (main)
```

```
$ git log --oneline
```

```
40cf763 (HEAD -> main) AA add
```

```
파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]/cupd (main)
```

```
$ git log -p
```

```
commit 40cf763b227fd96515b4b2e9fb359dd0763aa7d6 (HEAD -> main)
```

```
Author: ai7dnn <ai7dnn@gmail.com>
```

```
Date: Mon Jan 23 16:53:42 2023 +0900
```

```
AA add
```

```
diff --git a/f b/f
```

```
new file mode 100644
```

```
index 0000000..e69de29
```

```
diff --git a/g b/g
```

```
new file mode 100644
```

```
index 0000000..e69de29
```

```
diff --git a/h b/h
```

```
new file mode 100644
```

```
index 0000000..e69de29
```



LESSON 02

rebase -i로 여러 커밋 수정



2 rebase-i로 여러 커밋 수정

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

2 rebase-i로 여러 커밋 수정

rebase의 --interactive 주요 명령어

\$ git rebase --interactive HEAD~3

- 기본 편집기로 열리며, HEAD~3이면 HEAD~2, HEAD~1, HEAD 3개의 행을 편집 가능
- 로그 결과와 다르게 오래된 커밋부터 표시
 - pick 486bbc1 B
 - pick a471bb3 C
 - pick bb0a874 D

```
$ git log --oneline
```

```
bb0a874 (HEAD -> main) D
```

```
A471bb3 C
```

```
486bbc1 B
```

```
40cf763 AA add
```

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

**인자는 HEAD~3,
이후부터 수정가능**

2 rebase-i로 여러 커밋 수정

명령 squash 방법

✓ 이후 것(더 최신의 커밋)을 이전 커밋(더 오래된 커밋)에 뭉치는 방법

- 상위에 위치한 커밋에 pick이 있으면서
 - 아래로 연속적으로 squash 명령
 - 다음으로 커밋 E가 커밋 BB에 뭉쳐져 커밋 E가 제거됨
 - ➡ pick 0e88bee BB
 - ➡ squash d88525b E

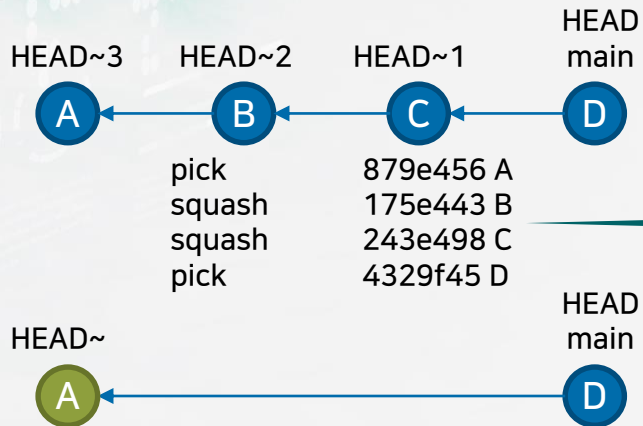
✓ 새로운 커밋 메시지를 입력하는 단계 있음

- 기본 편집기로 편집
 - BB와 E를 합친 커밋 메시지를 지정

2 rebase-i로 여러 커밋 수정

명령 squash 방법

☑ 사례



2 rebase-i로 여러 커밋 수정



[실습] 5개의 커밋 준비



파일 f를 5번 수정하면서 커밋

```
$ git init ireb
Initialized empty Git repository in C:/[smart git]/ireb/.git/

$ cd ireb

$ echo 111 > f

$ git add f

$ git commit -m A
[main (root-commit) 414f9f5] A
1 file changed, 1 insertion(+)
create mode 100644 f

$ echo 222 >> f

$ git commit -am B
[main b896380] B
1 file changed, 1 insertion(+)

$ echo 333 >> f
```

```
$ git commit -am C
[main 977e7f2] C
1 file changed, 1 insertion(+)
```

```
$ echo 444 >> f
```

```
$ git commit -am D
[main a204b25] D
1 file changed, 1 insertion(+)
```

```
$ echo 555 >> f
```

```
$ git commit -am e
[main 9b9d783] e
1 file changed, 1 insertion(+)
```

```
$ git log --oneline
9b9d783 (HEAD -> main) e
a204b25 D
977e7f2 C
b896380 B
414f9f5 A
```

2 rebase-i로 여러 커밋 수정

최신 4개를 부분 메시지 수정

✓ \$ git rebase --interactive HEAD~4

✓ hint: Waiting for your editor to close the file...

✓ 편집기에 다음이 표시

pick b896380 B

pick 977e7f2 C

pick a204b25 D

pick 9b9d783 e

✓ 다음으로 수정 후 저장, 닫기

pick b896380 B

reword 977e7f2 C

reword a204b25 D

reword 9b9d783 e

**해당 커밋 메시지를
수정한다는 명령**

2 rebase-i로 여러 커밋 수정

명령 reword로 계속 편집기 실행

3개의 커밋을 수정

계속 표시되는 편집기에서 수정 후 저장, 닫기 계속

- C => CC
- D => DD
- e => EE

```
파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]/ireb (main)
$ git rebase --interactive HEAD~4
hint: waiting for your editor to close the file...
```

```
[detached HEAD 2310abe] CC
Date: Mon Jan 23 18:03:47 2023 +0900
1 file changed, 1 insertion(+)
[detached HEAD f6549db] DD
Date: Mon Jan 23 18:04:03 2023 +0900
1 file changed, 1 insertion(+)
[detached HEAD 58754e2] EE
Date: Mon Jan 23 18:04:17 2023 +0900
1 file changed, 1 insertion(+)
```

```
Successfully rebased and updated refs/heads/main.
```

```
파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]/ireb (main)
$ git log --oneline
58754e2 (HEAD -> main) EE
f6549db DD
2310abe CC
b896380 B
414f9f5 A
```

2 rebase-i로 여러 커밋 수정

[실습] 명령 squash

✓ 커밋 EE, DD를 모두 커밋 CC에 합하기

• \$ git rebase --interactive HEAD~4

```
pick b896380 B
pick 2310abe CC
pick f6549db DD
pick 58754e2 EE
```

편집

```
pick b896380 B
pick 2310abe CC
Squash a204b25 DD
s 58754e2 EE
```

pick, squash
순으로 입력

```
$ git log --oneline
58754e2 (HEAD -> main) EE
f6549db DD
2310abe CC
b896380 B
414f9f5 A
```

로그
변화

```
$ git log --oneline
4fc7c87 (HEAD -> main) CC-DD-EE
b896380 B
414f9f5 A
```




2 rebase-i로 여러 커밋 수정

커밋 메시지 수정

뭉쳐지는 커밋 CC의 메시지 수정

저장 후 닫기

```
# This is a combination of 3 commits.
```

```
# This is the 1st commit message:
```

```
CC
```

```
# This is the commit message #2:
```

```
DD
```

```
# This is the commit message #3:
```

```
EE
```

편집

```
# This is a combination of 3 commits.
```

```
# This is the 1st commit message:
```

```
CC - DD - EE
```

```
# This is the commit message #2:
```

```
# This is the commit message #3:
```



2 rebase-*i*로 여러 커밋 수정

명령 squash 성공

로그 이력 수가 줄어듦

```
파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]/ireb (main)
```

```
$ git rebase --interactive HEAD~4
```

```
[detached HEAD 4fc7c87] CC-DD-EE
```

```
Date: Mon Jan 23 18:03:47 2023 +0900
```

```
1 file changed, 3 insertions(+)
```

```
Successfully rebased and updated refs/heads/main.
```

```
파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]/ireb (main)
```

```
$ git log --oneline
```

```
4fc7c87 (HEAD -> main) CC-DD-EE
```

```
f6549db DD
```

```
b896380 B
```

```
414f9f5 A
```

2 rebase-i로 여러 커밋 수정

명령 drop

마지막 커밋 제거

- \$ git rebase --interactive HEAD~2

```
pick b896380 B
pick 4fc7c87 CC-DD-EE
```

편집

```
pick b896380 B
d 4fc7c87 CC-DD-EE
```

```
파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]/ireb (main)
```

```
$ git log --oneline
```

```
4fc7c87 (HEAD -> main) CC-DD-EE
```

```
b896380 B
```

```
414f9f5 A
```

```
파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]/ireb (main)
```

```
$ git rebase --interactive HEAD~2
```

```
Successfully rebased and updated refs/heads/main.
```

```
파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]/ireb (main)
```

```
$ git log --oneline
```

```
b896380 (HEAD -> main) CC-DD-EE
```

```
414f9f5 A
```

```
파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]/ireb (main)
```

```
$ cat f
```

```
111
```

```
222
```

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