

## 🐼 [Git] 명령어(5) - reset, revert 🐼

2017. 11. 27. 23:56

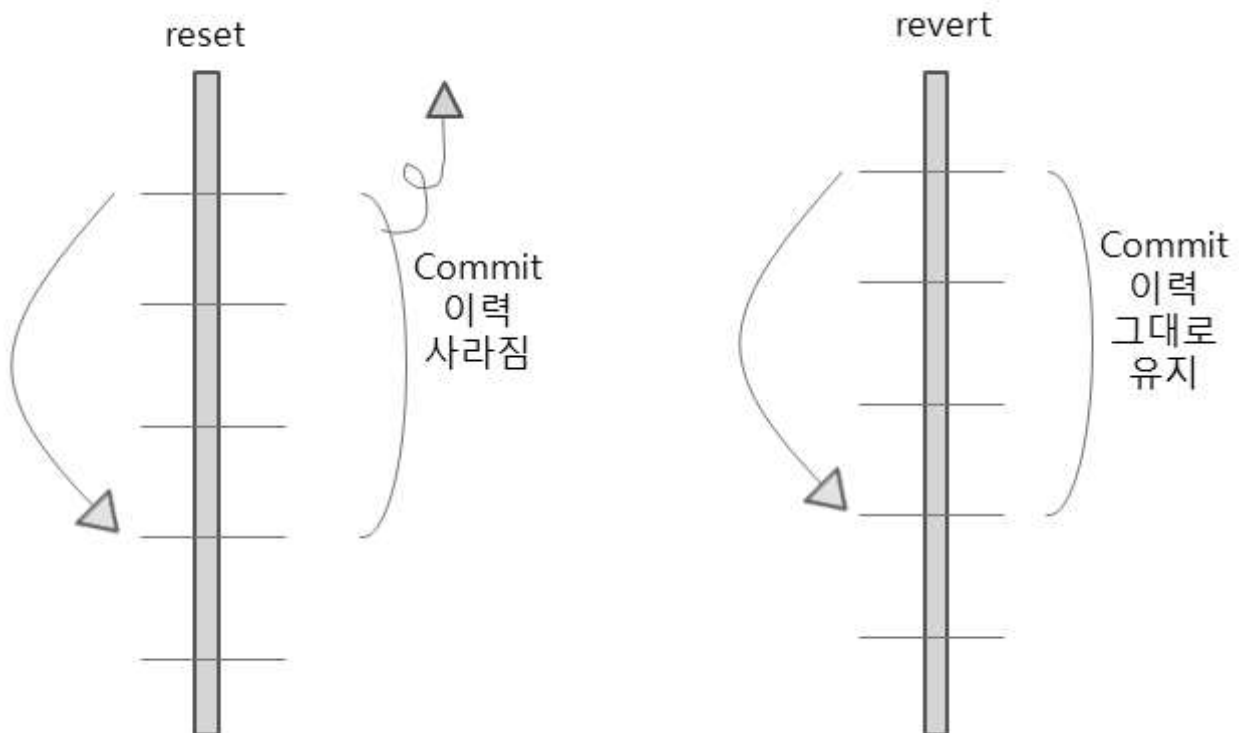
### 1. reset과 revert

작업을 진행하다가 실수로 중요한 파일을 삭제했거나 제대로 병합이 안됐을 경우, 잘 작동이 되던 이전 버전으로 돌아가야 합니다.

이것이 바로 버전 관리를 하는 이유이며, 이 때 사용하는 명령어가 git reset과 git revert라는 명령어입니다.

git reset을 명령어는 특정 커밋으로 되돌아갈 수 있는데, 되돌린 버전 이후의 버전들은 **히스토리에서 삭제**됩니다.

git revert는 reset처럼 특정 버전으로 되돌아갈 수 있지만, 되돌린 버전 이후의 버전들의 **이력은 남아있다**는 점에서 차이가 있습니다.



reset과 revert 명령어를 통해 과거 버전으로 돌아갈 수 있지만 그 밖에 잘못된 병합을 취소할수도 있는 기능이 있습니다.

victolee 구독하기

이 글에서 이전 버전으로 되돌리기, 병합 취소 등의 기능에 대해 알아보도록 하겠습니다.

## 2. 명령어

### ■ git checkout

- .
  - working directory에서 수정한 모든 파일(git add이전)을 현재 버전으로 되돌리기
- -- {file}
  - working directory에서 수정한 특정 파일(git add이전)을 현재 버전으로 되돌리기

### ■ git reset

- .
  - 현재 버전으로 되돌리기 ( add 무효화하기 )
- {commit번호}
  - 특정 버전으로 되돌리지만, 커밋 이력 삭제
  - 파일 내용은 유지
- --hard {commit번호}
  - 파일 내용까지 되돌림
- --soft {commit번호}
  - 파일 내용은 그대로 유지하면서 staging area에 올림 ( add 상태 )
- -merge ORIG\_HEAD
  - 바로 이전 병합 취소

### ■ git revert

- {commit번호}
  - 특정 버전으로 되돌리는데, 파일 내용은 그대로 유지하고 되돌린 버전 이후의 모든 commit 이력은 그대로 보존
- --mainline 숫자
  - 과거 병합을 취소
  - 숫자로 명시된 브랜치 라인을 기준으로 되돌아간다.

### 3. 되돌리기 - working directory 내용 비우기

아직 add 명령어를 하지 않은 상태에서 작업했던 내용을 모두 되돌리고 싶은 경우입니다.

즉, HEAD가 가장 최근에 커밋된 버전으로 이동하는 것이죠.

이 때 사용하는 명령어는 reset, revert도 아니지만 되돌리기와 관련된 기능이므로 언급하도록 하겠습니다.

실습 진행을 위해 텍스트 파일을 생성하여 내용을 작성하고, 커밋을 해줍니다.

```
# git add

# git commit -m "메시지"
```

다음으로 파일을 아무렇게나 수정합니다.

이 때 git add 명령어를 실행하면 안됩니다. (add를 안했을 때 되돌리는 명령어니까요.)

그리고 상태를 확인해보도록 할게요.

```
# git status
```

```
PS C:\Users\samsung\Desktop\gitTest> git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

no changes added to commit (use "git add" and/or "git commit -a")
```

당연히 commit을 한 후에, 파일을 수정 했으니 add할 것이 있다고 하겠죠?

이 상황에서 방금 수정한 파일을 수정한 적이 없던 것처럼 하려면 status에서 보는 것처럼 다음 명령을 실행하면 됩니다.

```
# git checkout -- { file }
```

모든 파일을 working directory에서 되돌리고 싶다면, git checkout . 명령어를 실행하면 되고, 폴더를 되돌리고 싶다면 git checkout {폴더명} 명령어를 실행하면 됩니다.

```
# git checkout .  
# git checkout { 폴더명 }
```

victolee 구독하기

정말로 되돌아 갔는지 확인을 위해 상태를 보겠습니다.

이번에는 수정할 것이 없다고 합니다.

```
PS C:\Users\samsung\Desktop\gitTest> git status  
On branch master  
nothing to commit, working directory clean
```

#### 4. reset으로 되돌리기 - add를 무효화

이번에는 git add를 한 상태에서, 변경된 staging area를 무효화해서 가장 최근 버전으로 되돌아가는 방법에 대해 알아보겠습니다.

이전 예제처럼 working directory에서 txt 파일을 아무렇게나 수정하고, 차례대로 아래의 명령어를 실행해보겠습니다.

```
# git status  
# git add .  
# git status  
# git reset {파일명}  
# git status
```

```

PS C:\Users\samsung\Desktop\gitTest> git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\samsung\Desktop\gitTest> git add .
PS C:\Users\samsung\Desktop\gitTest> git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   newFile.txt

PS C:\Users\samsung\Desktop\gitTest> git reset .
Unstaged changes after reset:
M       newFile.txt
PS C:\Users\samsung\Desktop\gitTest> git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

no changes added to commit (use "git add" and/or "git commit -a")

```

파일을 수정한 후, 상태를 보면 변경 내역이 있다고 합니다.

그리고 git add 명령어를 실행하면 newFile.txt 파일은 staging area에 올라간 상태가 됩니다.

이 때, add된 것을 무효화 하려면 reset 명령어를 실행합니다.

git reset { 파일명 } 명령어를 실행하면 working directory 상태로 돌아가게 됩니다.

즉, 수정된 내용은 유지한 채로 add 이전 상태로 되돌린 것입니다.

## 5. reset으로 되돌리기 - 다양한 옵션으로 버전 되돌리기

이번에는 이미 커밋된 과거의 커밋으로 버전을 되돌리는 reset 명령어의 옵션에 대해 알아보겠습니다.

실습을 위해 파일을 적절하게 수정하고 커밋을 해줍니다.

```

# git add .
# git commit -m "메시지 1"
# git add .
# git commit -m "메시지 2"
# git add .
# git commit -m "메시지 3"

```

```
# git add .
# git commit -m "메시지 4"
```

victolee 구독하기

```
victolee@DESKTOP-QD9V37L MINGW64 ~/Desktop/gitTest (master)
$ git log --oneline -4
52f0f10 (HEAD -> master) 메 시 지 4
a97945e 메 시 지 3
7e094f8 메 시 지 2
2735753 메 시 지 1
```

## 1) --hard 옵션

git reset 명령어로 메시지3 버전( a979452 )으로 되돌아 가고자 합니다.

이 때, --hard 옵션을 주면 working directory의 내용까지 모두 바꿉니다.

```
# git reset --hard {버전명}
```

```
victolee@DESKTOP-QD9V37L MINGW64 ~/Desktop/gitTest (master)
$ git reset --hard a97945e
HEAD is now at a97945e 메 시 지 3

victolee@DESKTOP-QD9V37L MINGW64 ~/Desktop/gitTest (master)
$ git log --oneline -4
a97945e (HEAD -> master) 메 시 지 3
7e094f8 메 시 지 2
2735753 메 시 지 1

victolee@DESKTOP-QD9V37L MINGW64 ~/Desktop/gitTest (master)
$ git status
On branch master
nothing to commit, working tree clean
```

reset --hard 명령을 실행한 후, log를 확인해보니 "메시지 4"가 사라졌습니다.


그리고 상태를 확인해보니 working directory도 깔끔합니다.

## 2) 옵션 없이

이번에는 옵션 없이 reset해서 working directory 내용은 유지하고, 커밋 이력만 삭제하도록 하겠습니다.

마찬가지로 다음과 같이 히스토리 있을 때, 메시지3 버전( a97945e )으로 되돌아가겠습니다.

```
# git reset {버전명}
```



```
victolee@DESKTOP-QD9V37L MINGW64 ~/Desktop/gitTest (master)
$ git log --oneline -5
7a3c9f8 (HEAD -> master) 메 시 지 4
a97945e 메 시 지 3
7e094f8 메 시 지 2
2735753 메 시 지 1

victolee@DESKTOP-QD9V37L MINGW64 ~/Desktop/gitTest (master)
$ git reset a97945e
Unstaged changes after reset:
M      test.txt.txt

victolee@DESKTOP-QD9V37L MINGW64 ~/Desktop/gitTest (master)
$ git log --oneline -4
a97945e (HEAD -> master) 메 시 지 3
7e094f8 메 시 지 2
2735753 메 시 지 1

victolee@DESKTOP-QD9V37L MINGW64 ~/Desktop/gitTest (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   test.txt.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

victolee 구독하기

--hard 옵션과 마찬가지로 커밋 이력은 삭제됐습니다.

그런데 상태를 보면, working directory 내역이 남아있는 것을 확인할 수 있습니다.

### 3) --soft 옵션

이번에는 옵션없이 실행해서 add까지 된, 즉 staging area까지 적용되는 상태를 --soft 옵션을 추가해서 reset 해보겠습니다.

```
# git reset --soft {버전명}
```

```
victolee@DESKTOP-QD9V37L MINGW64 ~/Desktop/gitTest (master)
$ git log --oneline -5
4df6af3 (HEAD -> master) 메 시 지 4
a97945e 메 시 지 3
7e094f8 메 시 지 2
2735753 메 시 지 1

victolee@DESKTOP-QD9V37L MINGW64 ~/Desktop/gitTest (master)
$ git reset --soft a97945e

victolee@DESKTOP-QD9V37L MINGW64 ~/Desktop/gitTest (master)
$ git log --oneline -4
a97945e (HEAD -> master) 메 시 지 3
7e094f8 메 시 지 2
2735753 메 시 지 1

victolee@DESKTOP-QD9V37L MINGW64 ~/Desktop/gitTest (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   test.txt.txt
```

victolee 구독하기

마찬가지로 커밋 이력은 삭제 됐습니다.

그런데 상태를 보니 add까지 적용되어 있는 것을 확인할 수 있습니다.

## 6. revert로 되돌리기

reset은 버전을 되돌리면서 되돌린 버전 이후의 히스토리를 모두 삭제하는 반면,

revert는 버전을 되돌리지만 모든 히스토리를 유지합니다.

reset 예제와 마찬가지로 히스토리와 다음과 같을 때, revert 명령으로 메시지3 버전( a97945e )으로 되돌아가보겠습니다.

```
# git revert {버전명}
```



```
victolee@DESKTOP-QD9V37L MINGW64 ~/Desktop/gitTest (master)
$ git log --oneline -5
ad613ef (HEAD -> master) 메 시 지 4
a97945e 메 시 지 3
7e094f8 메 시 지 2
2735753 메 시 지 1

victolee@DESKTOP-QD9V37L MINGW64 ~/Desktop/gitTest (master)
$ git revert a97945e
error: could not revert a97945e... 메 시 지 3
hint: after resolving the conflicts, mark the corrected paths
hint: with 'git add <paths>' or 'git rm <paths>'
hint: and commit the result with 'git commit'

victolee@DESKTOP-QD9V37L MINGW64 ~/Desktop/gitTest (master|REVERTING)
$ git log --oneline -5
ad613ef (HEAD -> master) 메 시 지 4
a97945e 메 시 지 3
7e094f8 메 시 지 2
2735753 메 시 지 1

victolee@DESKTOP-QD9V37L MINGW64 ~/Desktop/gitTest (master|REVERTING)
$ git status
On branch master
You are currently reverting commit a97945e.
(fix conflicts and run "git revert --continue")
(use "git revert --abort" to cancel the revert operation)

Unmerged paths:
  (use "git reset HEAD <file>..." to unstage)
  (use "git add <file>..." to mark resolution)

        both modified:   test.txt.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

reset과 비교해보겠습니다.

먼저 revert 명령을 실행할 때, 충돌이 발생한 것을 확인할 수 있습니다.

reset의 경우는 커밋이력이 사라지기 때문에 충돌이 발생하지 않지만, revert의 경우 커밋이력이 남기 때문에 같은 곳을 수정했다면 충돌이 발생할 것입니다.

그리고 메시지4 커밋 이력은 그대로 남아 있네요.

상태를 확인해보니 working directory도 그대로 유지하고 있습니다.

즉, 그대로 유지하려다 보니 충돌이 발생한 것이겠죠.

## 병합 취소

### 7. reset으로 병합 취소하기

reset 명령어로 브랜치 간의 병합을 취소할 수도 있습니다.

예를 들어, master와 child 두 브랜치가 있을 때, master 브랜치에서 child 브랜치를 병합하려고 했는데 병합을 하면 안되는 작업이었던 것이었습니다.

이 경우에 병합을 한 후, 바로 병합을 취소해야 합니다.

병합은 위험한 작업이기 때문에 Git은 병합을 하기 전에, 최신 커밋에 포인터를 지정(**ORIG HEAD**)합니다.

따라서 **방금 병합한 것을 되돌리려면** -merge 옵션과 ORIG\_HEAD 포인터를 지정하여 reset 명령어를 실행하면 됩니다.

```
# git reset --merge ORIG_HEAD
```

## 8. revert로 병합 취소하기

reset과 마찬가지로 revert 명령어를 통해 병합을 취소할 수도 있습니다.

reset은 방금 병합한 것을 취소할 때 사용하지만, 조금 시간이 지난 병합을 취소하고 싶다면 revert 명령어로 병합을 취소하면 됩니다.

이 때 rebase로 병합을 했다면 merge에 대한 커밋 이력이 남지 않기 때문에 불가능합니다.

우선 병합 상태를 --graph 옵션을 통해 확인해보겠습니다.

```
PS C:\Users\Administrator\Desktop\gittest\empty_repo> git log --oneline --graph
* 1a83efc <HEAD -> master> Merge branch 'child'
|
| * 22d2f35 <child> child
* | beb1e77 master
|/
* 71d8bcc start
```

master와 child 브랜치가 있고, master 브랜치에서 child 브랜치를 병합한 상황입니다.

그런데 이 병합이 잘못된 것을 알게 되어 병합을 하기 전인, beb1e77으로 되돌아가고자 합니다.

revert의 --mainline 옵션을 사용해서 병합을 취소할 수 있습니다.

**mainline**이란 병합을 취소한 후에, 어느 라인을 기준으로 되돌아 갈 것인지 기준을 정하는 것입니다.

mainline은 병합 그래프를 기준으로 왼쪽에서부터 1로 계산을 합니다.

즉, beb1e77가 1이 되고, 22d2f35는 2가 됩니다.

따라서 아래와 같이 --mainline 숫자 옵션으로 명령어를 실행하면 병합을 취소하고 beb1e77 버전으로 되돌아갈 수 있게 됩니다.

```
# git revert --mainline 1 {취소할 병합 커밋ID}
```

{취소할 병합 커밋ID}는 병합이 완료된 커밋ID를 말하며, 여기서는 1a83efc가 되겠네요.

victolee 구독하기

명령어를 실행하면 편집기 창이 나오면서 커밋 메시지를 작성하라고 합니다.  
메시지를 작성한 후, 로그를 살펴보면 다음과 같습니다.

```
PS C:\Users\Administrator\Desktop\gittest\empty_repo> git log --oneline
1d78b79 <HEAD -> master> Revert "Merge branch 'child'"
1a83efc Merge branch 'child'
22d2f35 <child> child
beb1e77 master
71d8bcc start
```

또한 되돌린 파일 내역을 보시면 beb1e77 버전을 기준으로 맞춰져 있을 것입니다.

만약 병합이 merge로 이루어진 것이 아니라 rebase를 통해 이뤄졌다면, 병합된 커밋을 찾기가 힘들어집니다.

팀원과 잘 상의를 한 후에..... 커밋을 찾았다면, 아래의 명령어를 실행해서 과거의 버전으로 되돌아 가면 됩니다.

```
# git checkout { 커밋ID }
```

## 9. reset과 revert의 차이점

그렇다면 언제 git reset을 쓰고 언제 git revert를 사용할까요?

git reset은 remote repository까지 컨트롤할 수 없습니다.

커밋 이력이 삭제되므로, remote와 싱크가 안맞아서 결국 push 할 수 없습니다.

따라서 이미 원격 저장소에 push를 한 상태에서 되돌리고 싶다면 git revert를 사용해야 합니다.

병합 과정에서는 revert 명령어가 더 먼 과거의 버전으로 돌아갈 수 있다는 점이 다르네요.

이상으로 reset과 revert 명령어로 버전을 되돌리는 방법에 대해 알아보았습니다.  
버전을 되돌리는 명령어이므로 꼭 알아둬야 할 명령어입니다!

victolee 구독하기

4

구독하기

**'Git > 개념과 활용' 카테고리의 다른 글**

[Git] 명령어(6) - rebase ( merge와 비교, --interactive 옵션 ) (0)	2018.08.31
[Git] Github으로 협업하기 ( 토이 팀프로젝트 시나리오, 브랜치 전략 ) (2)	2017.12.02
[Git] .gitignore로 일부 파일 제외하여 push하기 (0)	2017.11.28
<b>[Git] 명령어(5) - reset, revert</b> (0)	2017.11.27
[Git] 명령어(4) - merge, conflicts (0)	2017.11.27
[Git] 명령어(3) - branch (0)	2017.11.27
[Git] 명령어(2) - remote, push, clone, pull (0)	2017.11.27
[Git] 명령어(1) - init, add, commit, status (0)	2017.11.27

NAME

PASSWORD

댓글은 큰 힘이 됩니다😊

잘못된 부분, 해결 과정 등은 댓글로 공유해주시면 다른 분이 읽을 때 많은 도움이 될것 같습니다🙏



댓글 등록 

victolee 구독하기

PREV 1 ... 224 225 226 227 228 229 230 231 232 ... 278 NEXT

Powered by Tistory, Designed by wallel

Rss Feed and Twitter, Facebook, Youtube, Google+