

# Git Reflog

In this section, we will take a look at `git reflog` command.

- If commit is not necessary at all and wants to remove then we can run the `git reset --hard` command: -

```
$ git reset --hard HEAD~1
```

- After running this command, that data will be gone forever. But we don't need to be panic at all, `git reflog` command shows the all actions that have been taken to the repository. This includes resets, reverts and merges.

```
$ git reflog
```

- You can easily undo the mistakes you made in the repository that information `reflog` command gives us.
- You can get the hash value from the `git reflog` command as we already see in the previous process.

```
$ git reset --hard 8ad5
```

- After this repository has been set into the previous state.
- You can see that `git reflog` status also changed.

```
$ git reflog
```

- `git log` and `git reflog` may look similar. `git log` will show you only information about the commits not the status about repository that `git reflog` does.

```
$ git log
```

```
$ git reflog
```



The image shows two terminal windows side-by-side. The left window, titled 'bash', shows the output of the `git reflog` command. It lists five entries with their commit hashes and descriptions: a reset to 8ad5d8c, a reset to HEAD~1 (a340aae), and three commits adding or changing stories. The right window, also titled 'bash', shows the output of the `git log` command, displaying the same three commits but omitting the reset operations.

```
$ git reflog
8ad5d8c HEAD@{0}: reset: moving to 8ad5d
a340aae HEAD@{1}: reset: moving to HEAD~1
8ad5d8c HEAD@{2}: commit: Added third story
aaba51e HEAD@{3}: commit: Changes to second story
fb9f13e HEAD@{4}: commit: Added second story

$ git reset --hard 8ad5
```

```
$ git log
8ad5d8c Added third story
aaba51e Changes to second story
fb9f13e Added second story
```