Reset vs Revert

At first glance, *resetting* might seem coincidentally close to *reverting*, but they are actually quite different. Reverting creates a new commit that reverts or undos a previous commit. Resetting, on the other hand, *erases* commits!

> ### ⚠ *Resetting Is Dangerous* ⚠
>
> *You've got to be careful with Git's resetting capabilities. This is one of the few commands that lets you erase commits from the repository. If a commit is no longer in the repository, then its content is gone.*
>
> *To alleviate the stress a bit, Git does keep track of everything for about 30 days before it completely erases anything. To access this content, you'll need to use the `git reflog` command. Check out these links for more info:*
>
> - *git-reflog*
> - *Rewriting History*
> - *reflog, your safety net*

Relative Commit References

You already know that you can reference commits by their SHA, by tags, branches, and the special `HEAD` pointer. Sometimes that's not enough, though. There will be times when you'll want to reference a commit relative to another commit. For example, there will be times where you'll want to tell Git about the commit that's one before the current commit...or two before the current commit. There are special characters called "Ancestry References" that we can use to tell Git about these relative references. Those characters are:

- `^` – indicates the parent commit
- `~` – indicates the *first* parent commit

Here's how we can refer to previous commits:

- the parent commit – the following indicate the parent commit of the current commit
  - HEAD^
  - HEAD~
  - HEAD~1
- the grandparent commit – the following indicate the grandparent commit of the current commit
  - HEAD^^
  - HEAD~2
- the great-grandparent commit – the following indicate the great-grandparent commit of the current commit
  - HEAD^^^
  - HEAD~3

The main difference between the `^` and the `~` is when a commit is created *from a merge*. A merge commit has *two* parents. With a merge commit, the `^` reference is used to indicate the *first* parent of the commit while `^2` indicates the *second* parent. The first parent is the branch you were on when you ran `git merge` while the second parent is the branch that was merged in.

It's easier if we look at an example. This what my `git log` currently shows:

```
* 9ec05ca (HEAD -> master) Revert "Set page heading to "Quests & Crusades""
* db7e87a Set page heading to "Quests & Crusades"
*   796ddb0 Merge branch 'heading-update'
|\
| * 4c9749e (heading-update) Set page heading to "Crusade"
* | 0c5975a Set page heading to "Quest"
|/
*   1a56a81 Merge branch 'sidebar'
|\
| * f69811c (sidebar) Update sidebar with favorite movie
| * e6c65a6 Add new sidebar content
* | e014d91 (footer) Add links to social media
* | 209752a Improve site heading for SEO
* | 3772ab1 Set background color for page
|/
* 5bfe5e7 Add starting HTML structure
* 6fa5f34 Add .gitignore file
```

```
* a879849 Add header to blog
* 94de470 Initial commit
```

Let's look at how we'd refer to some of the previous commits. Since `HEAD` points to the `9ec05ca` commit:

- `HEAD^` is the `db7e87a` commit
- `HEAD~1` is also the `db7e87a` commit
- `HEAD^^` is the `796ddb0` commit
- `HEAD~2` is also the `796ddb0` commit
- `HEAD^^^` is the `0c5975a` commit
- `HEAD~3` is also the `0c5975a` commit
- `HEAD^^^2` is the `4c9749e` commit (this is the grandparent's (`HEAD^^`) *second parent* (`^2`))

Which Commit?

Use this repository to answer the following quiz questions:

```
* 9ec05ca (HEAD -> master) Revert "Set page heading to "Quests & Crusades""
* db7e87a Set page heading to "Quests & Crusades"
*   796ddb0 Merge branch 'heading-update'
|\
| * 4c9749e (heading-update) Set page heading to "Crusade"
* | 0c5975a Set page heading to "Quest"
|/
*   1a56a81 Merge branch 'sidebar'
|\
| * f69811c (sidebar) Update sidebar with favorite movie
| * e6c65a6 Add new sidebar content
* | e014d91 (footer) Add links to social media
* | 209752a Improve site heading for SEO
* | 3772ab1 Set background color for page
|/
* 5bfe5e7 Add starting HTML structure
* 6fa5f34 Add .gitignore file
* a879849 Add header to blog
* 94de470 Initial commit
```

QUESTION 1 OF 4

Which commit is referenced by `HEAD~6`?

- 4c9749e
- 0c5975a
- 1a56a81
- f69811c
- e014d91
- ✓ 209752a

SUBMIT

✓

You did so well on that last one, why not give this one a go! Using the same repository, which commit is referenced by `HEAD~4^2`?

f69811c

The `git reset` Command

The `git reset` command is used to reset (erase) commits:

```
$ git reset <reference to commit>
```

```
$ git reset <reference-to-commit>
```

It can be used to:

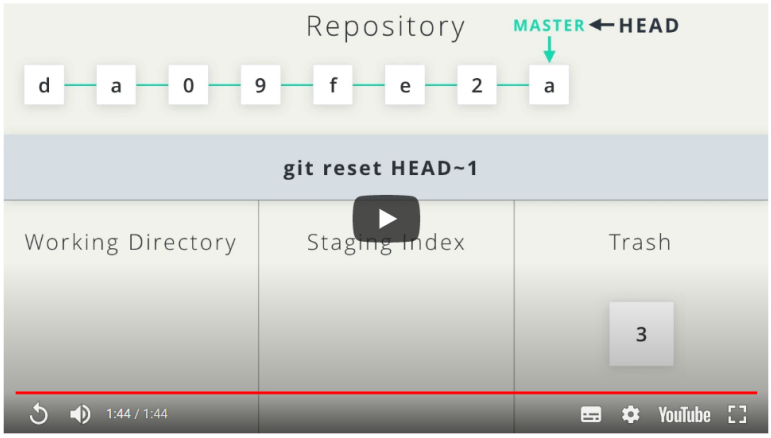
- move the HEAD and current branch pointer to the referenced commit
- erase commits
- move committed changes to the staging index
- unstage committed changes

Git Reset's Flags

The way that Git determines if it erases, stages previously committed changes, or unstages previously committed changes is by the flag that's used. The flags are:

- `--mixed`
- `--soft`
- `--hard`

It's easier to understand how they work with a little animation.



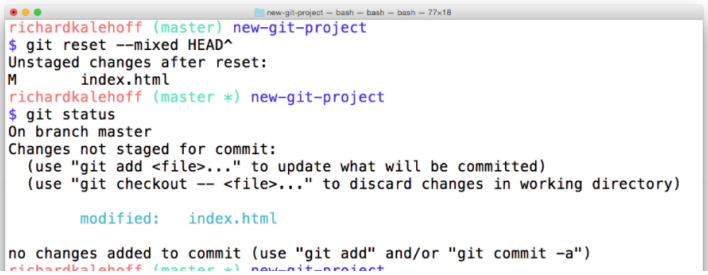
> *Backup Branch*
>
> *Remember that using the `git reset` command will erase commits from the current branch. So if you want to follow along with all the resetting stuff that's coming up, you'll need to create a branch on the current commit that you can use as a backup.*
>
> *Before I do any resetting, I usually create a `backup` branch on the most-recent commit so that I can get back to the commits if I make a mistake:*
>
> ```
> $ git branch backup
> ```

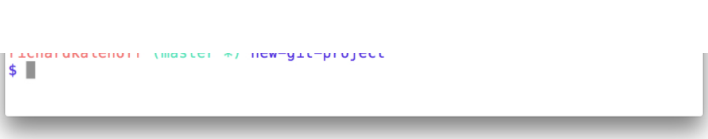
Reset's `--mixed` Flag

Let's look at each one of these flags.

```
* 9ec05ca (HEAD -> master) Revert "Set page heading to "Quests & Crusades""
* db7e87a Set page heading to "Quests & Crusades"
* 796ddb0 Merge branch 'heading-update'
```

Using the sample repo above with `HEAD` pointing to `master` on commit `9ec05ca`, running `git reset --mixed HEAD^` will take the changes made in commit `9ec05ca` and move them to the working directory.





The Terminal application showing the result of resetting with the `--mixed` flag. The changes are unstaged.

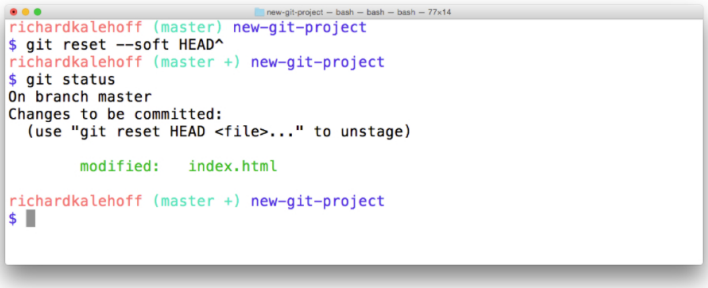
> ### 👂 Back To Normal 👂
>
> *If you created the `backup` branch prior to resetting anything, then you can easily get back to having the `master` branch point to the same commit as the `backup` branch. You'll just need to:*
>
> 1. *remove the uncommitted changes from the working directory*
> 2. *merge `backup` into `master` (which will cause a Fast-forward merge and move `master` up to the same point as `backup`)*
>
> ```
> $ git checkout -- index.html
> $ git merge backup
> ```

Reset's `--soft` Flag

Let's use the same few commits and look at how the `--soft` flag works:

```
* 9ec05ca (HEAD -> master) Revert "Set page heading to "Quests & Crusades""
* db7e87a Set page heading to "Quests & Crusades"
* 796ddb0 Merge branch 'heading-update'
```

Running `git reset --soft HEAD^` will take the changes made in commit `9ec05ca` and move them directly to the Staging Index.



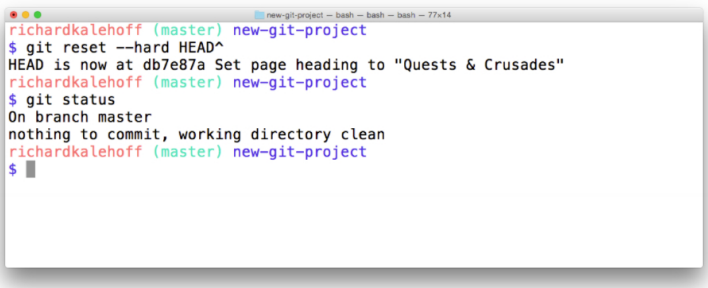
The Terminal application showing the result of resetting with the `--soft` flag. The changes are moved to the Staging Index.

Reset's `--hard` Flag

Last but not least, let's look at the `--hard` flag:

```
* 9ec05ca (HEAD -> master) Revert "Set page heading to "Quests & Crusades""
* db7e87a Set page heading to "Quests & Crusades"
* 796ddb0 Merge branch 'heading-update'
```

Running `git reset --hard HEAD^` will take the changes made in commit `9ec05ca` and erases them.



The Terminal application showing the result of resetting with the `--hard` flag. The changes are moved

erased.

Now it's your turn!

Refer to the following repository:

```
* e014d91 (HEAD -> master, footer) Add links to social media
* 209752a Improve site heading for SEO
* 3772ab1 Set background color for page
* 5bfe5e7 Add starting HTML structure
* 6fa5f34 Add .gitignore file
* a879849 Add header to blog
* 94de470 Initial commit
```

QUESTION 3 OF 4

What will happen to the changes from the `3772ab1` commit if `git reset --hard HEAD~3` is run? Will the changes be in the Staging Index, in the Working Directory, or complete erased?

- ○ Staging Index
- ○ Working Directory
- ✓ erased

SUBMIT

QUESTION 4 OF 4

What will happen to the changes from the `209752a` commit if `git reset --soft HEAD^^` is run? Will the changes be in the Staging Index, in the Working Directory, or complete erased?

- ✓ Staging Index
- ○ Working Directory
- ○ erased

SUBMIT

Reset Recap

To recap, the `git reset` command is used erase commits:

```
$ git reset <reference-to-commit>
```

It can be used to:

- move the HEAD and current branch pointer to the referenced commit
- erase commits with the `--hard` flag
- moves committed changes to the staging index with the `--soft` flag
- unstages committed changes `--mixed` flag

Typically, ancestry references are used to indicate previous commits. The ancestry references are:

- `^` – indicates the parent commit
- `~` – indicates the first parent commit

Further Research

- git-reset from Git docs
- Reset Demystified from Git Blog
- Ancestry References from Git Book

NEXT