
GIT REAL
2

REFLOG

CHAPTER 6

LOST DATA

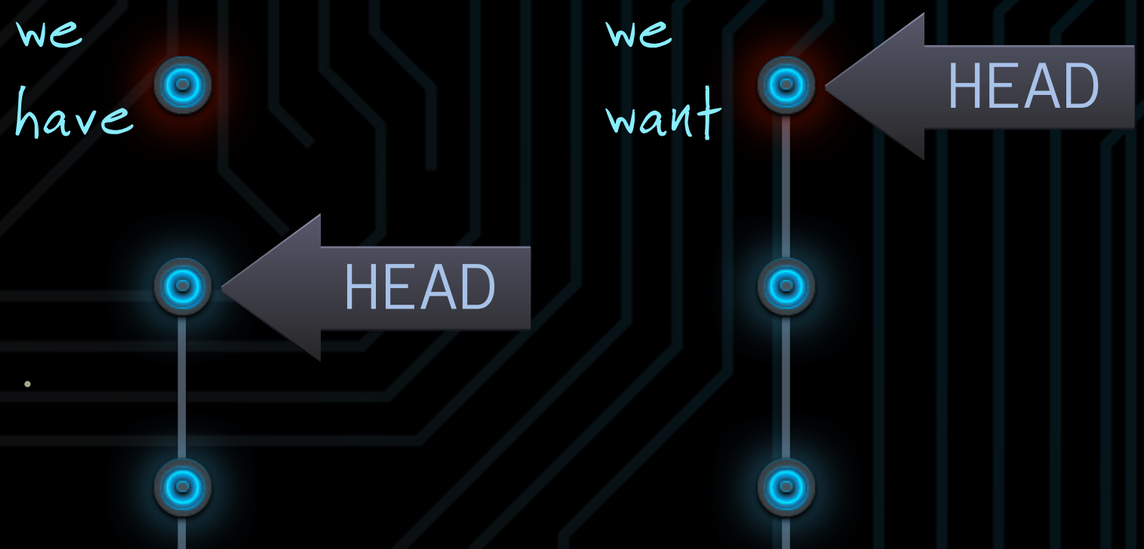
Gregg decided he didn't need that latest commit, and did a hard reset

```
$ git log --pretty=oneline
1e62107... Add third section.
43c13e7... Add second section.
2bd404a... Add first section.

$ git reset --hard 43c1
HEAD is now at 43c13e7 Add second section.
```

And now he's realized
that was a mistake

Git *never* deletes a commit
(partly because of situations like this)



It's just that no
branch points to
it right now

REFLOG

How can you get those commits back? They're not in the log...

```
$ git log --pretty=oneline
43c13e7... Add second section.
2bd404a... Add first section.
```

But Git keeps a *second* log, only in your local repo, called the *reflog*

```
$ git reflog
43c13e7 HEAD@{0}: reset: moving to 43c1
1e62107 HEAD@{1}: commit: Add third section.
43c13e7 HEAD@{2}: commit: Add second section.
2bd404a HEAD@{3}: commit (initial): Add first section.
```

here's after the reset

here's the commit you want back

REFLOG

Git updates the reflog anytime HEAD moves (due to new commits, checking out branches, or resetting)

```
$ git reflog
43c13e7 HEAD@{0}: reset: moving to 43c1
1e62107 HEAD@{1}: commit: Add third section.
43c13e7 HEAD@{2}: commit: Add second section.
2bd404a HEAD@{3}: commit (initial): Add first section.
```

commit SHA

reflog shortname

operation that caused HEAD to move

HEAD@{0} is always your current commit

RESTORING COMMITS

```
$ git reflog
43c13e7 HEAD@{0}: reset: moving to 43c1
1e62107 HEAD@{1}: commit: Add third section.
43c13e7 HEAD@{2}: commit: Add second section.
2bd404a HEAD@{3}: commit (initial): Add first section.
```

we need this commit back

```
$ git reset --hard 1e62
```

OR

```
$ git reset --hard HEAD@{1}
```

Now that we know where the commit is, we can point the branch back to it

you can use reflog shortnames in place of commit SHAs

RESTORING COMMITS

If we run a "git log", we'll see the commit is back

```
$ git log --oneline
1e62107 Add third section.
43c13e7 Add second section.
2bd404a Add first section.
```

The files will be waiting in the directory

LOCAL-ONLY

Note that your reflog exists only in your local repository

```
$ git log --oneline
1e62107 Add third section.
43c13e7 Add second section.
2bd404a Add first section.
```

If Jane clones Gregg's repo, the log will still be there

```
$ git reflog
1e62107 HEAD@{0}: clone: from git@example.com:aquarium.git
```

But the reflog starts over from scratch

DELETED BRANCHES

Jane decided the aquarium project doesn't need its "aviary" branch

```
$ git branch -d aviary
error: The branch 'aviary' is not fully merged.
If you are sure you want to delete it, run 'git branch -D aviary'.

$ git branch -D aviary
Deleted branch aviary (was 280ee63).
```

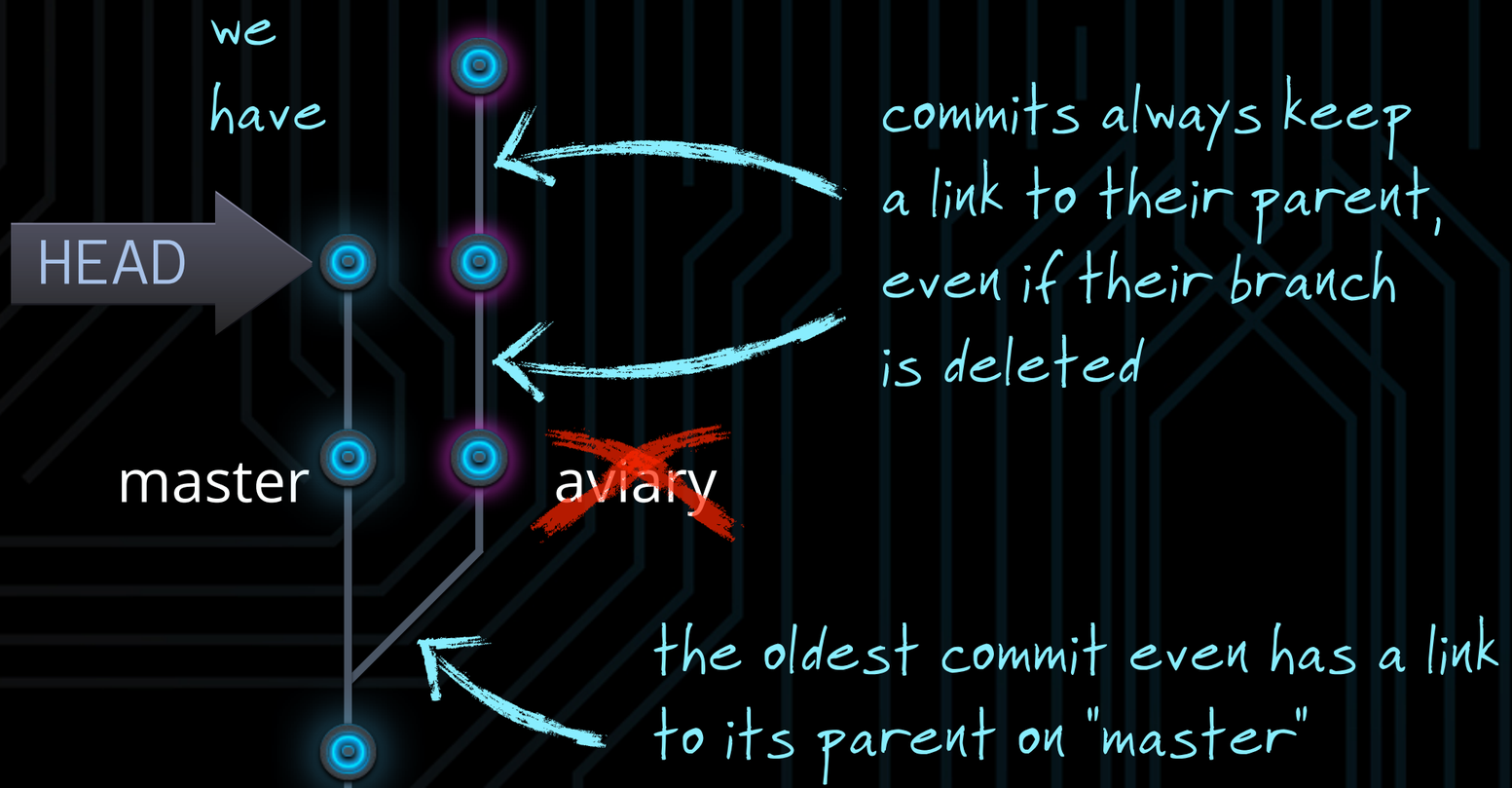
It wasn't merged

So she force-deleted it

She forgot that it had a new menu system she wants for the next feature

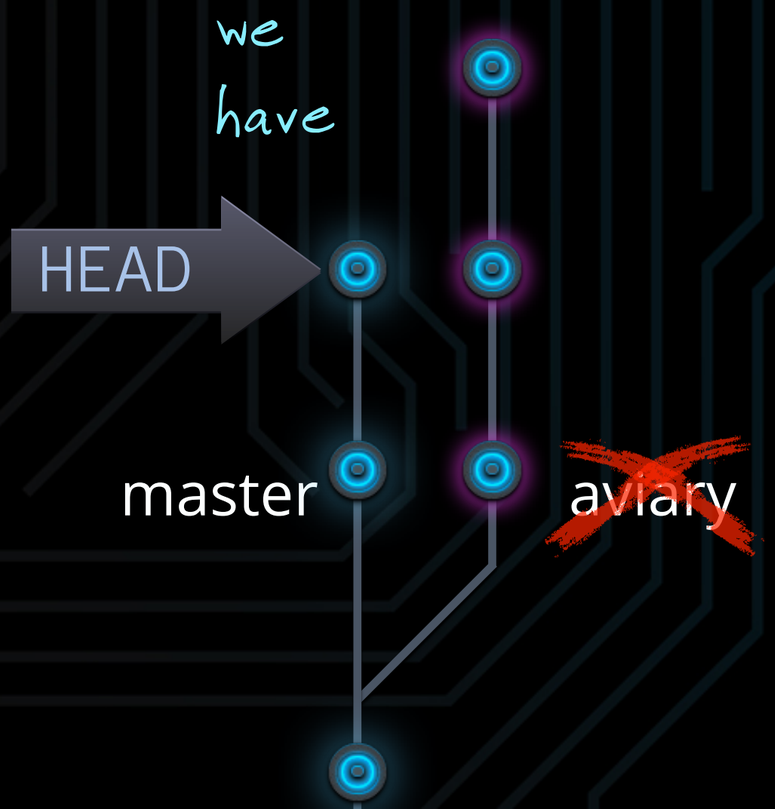
DELETED BRANCHES

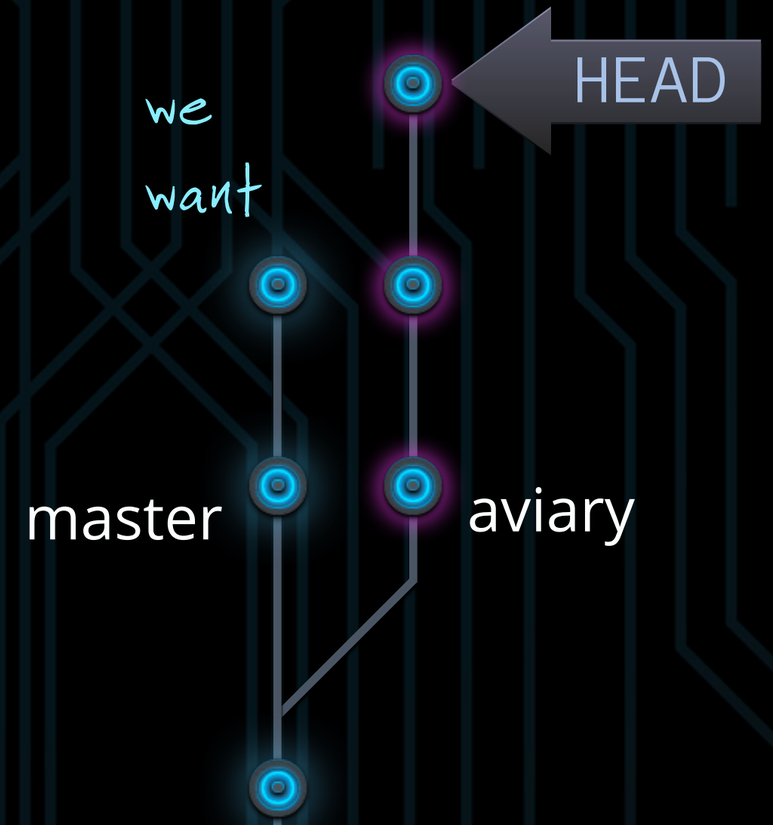
As before, the commits aren't really gone, just the branch



DELETED BRANCHES

If we can find the latest commit and re-create the branch that points to it, it'll be like the branch was never deleted





WALK-REFLOGS

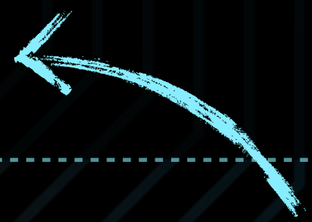
The output of "git reflog" isn't as detailed as the full log

```
$ git log --walk-reflogs
commit b7a5df73f03de5bdf534f889d8d2c0a9bf89b1e0
Reflog: HEAD@{0} (Jane <jane@example.com>)
Reflog message: checkout: moving from aviary to master
Author: Jane <jane@example.com>

    Add clownfish.

commit 280ee635634f8c97b6e7d47beef1a110a1679811
Reflog: HEAD@{1} (Jane <jane@example.com>)
Reflog message: commit: Add birds.
Author: Jane <jane@example.com>

    Add birds.
...
```



There's the most recent commit from our deleted branch!

Pass the --walk-reflogs option to "git log" to see the reflog info in full log format

Includes reflog shortnames and messages

RESTORING DELETED BRANCHES

Instead of a reset, Jane creates another branch, and points it at that commit

```
$ git branch aviary 280e
```

OR

```
$ git branch aviary HEAD@{1}
```

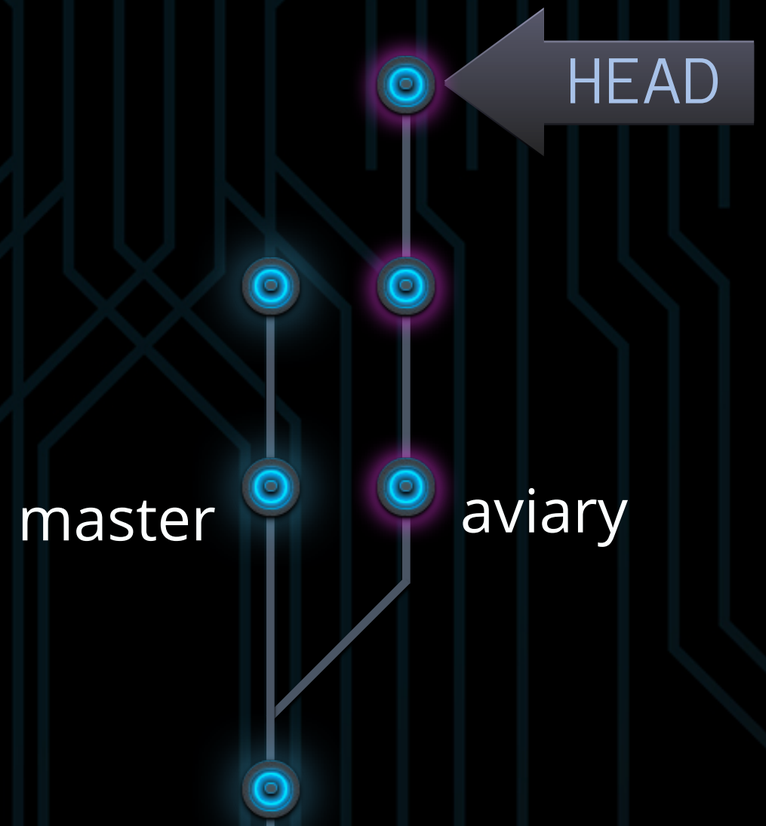
SHA or reflog shortname

RESTORING DELETED BRANCHES

When she checks out the new branch, the aviary work is back

```
$ git checkout aviary

$ git log --oneline
280ee63 Add birds.
b7a5df7 Add license.
```



The deleted branch is resurrected!