

COMP1531



Coding Together - Git - Undo

Lecture 9.3

Author(s): Hayden Smith



[\(Download as PDF\)](#)

In This Lecture

- **Why?** 🤔
 - Sometimes we make mistakes using git, and we need methods to be able to resolve them (carefully)
- **What?** 📄
 - Doing a hard reset to a point in time
 - Altering the git history
 - Amending a commit

Mistakes

- Everything we've done until this point continues to build on the git history. But we've largely considered the git history immutable.
- With git, sometimes we make mistakes. Sometimes we want to undo things, or change history.
- Two ways we're going to discuss this are:
 - git resets
 - git commit amend

Git Reset

Atlassian has a very [clear article](#) about git reset. We will use this as guidance. We will mainly discuss hard and soft resets through a demonstration.

Git Reset

```
git reset --hard [hash]
```

Sets all of your code to a specific commit **hash**. This is used for saying "I want to go back in time, and I don't care about anything that's happened since that point I'm going back to."

Git Reset

```
git reset --soft [hash]
```

Keeps all of your current code the same, but just changes what commit you're pointing to (i.e. point to a new **hash**). This is used for saying "I like the code I have, so let's not change anything, but I want to alter the history of commits that got me here."

Git Commit --Amend

```
git commit --amend -m "Commit"
```

- Sometimes we need to update our previous commit name. We can do that easily by making another commit that over-rides it.
- The --amend flag will make the commit, but it will replace the most recent commit with the new commit instead of adding another commit to the history
- Let's do a demonstration.

Feedback



Or go to the [form here](#).

