Revert

The `git revert` command undoes a commit, but unlike `git reset`, which removes the commit from the commit history, it appends a new commit with the resulting content. This prevents Git from losing history, which is important for the integrity of your revision history and for reliable collaboration. When you are working on a repository with other developers, using `git reset` is **highly dangerous** because you alter the history of commits which makes it very difficult to maintain a consistent history of commits with other developers.

Let's imagine the following situation.

1. You are working on a file and you add and commit your changes
2. You then work on a few other things, and make some more commits
3. Now you realize, three or four commits ago, you did something that you would like to undo - how can you do this?

You might be thinking, just use `git reset`, but this will remove **all** of the commits after the one you would like to change - `git revert` to the rescue! Let's walk through an example:

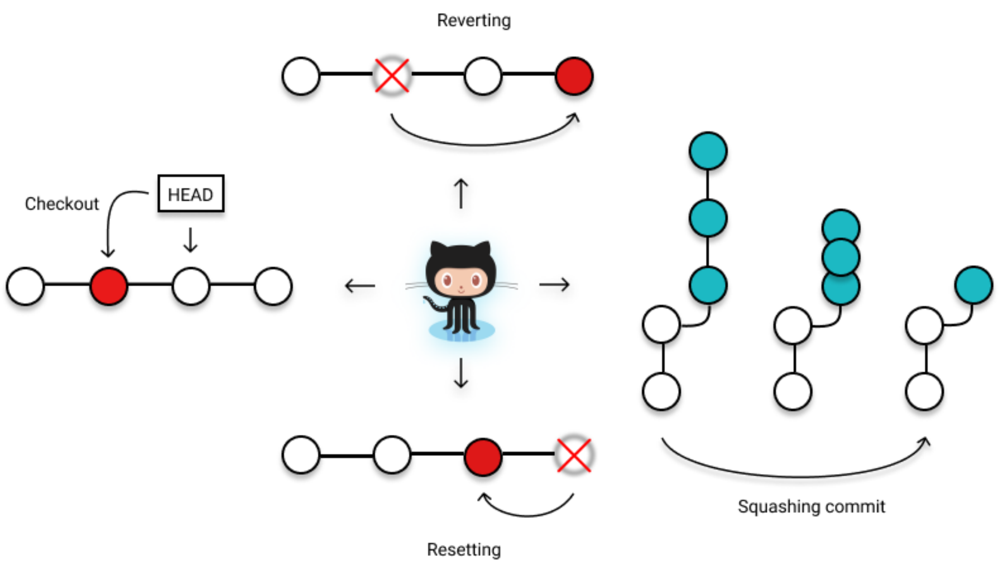
```
mkdir learn_revert # Create a folder called `learn_revert`
cd learn_revert # `cd` into the folder `learn_revert`
git init # Initialize a git repository

touch first.txt # Create a file called `first.txt`
echo Start >> first.txt # Add the text "Start" to `first.txt`

git add . # Add the `first.txt` file
git commit -m "adding first" # Commit with the message "Adding first.txt"

echo WRONG > wrong.txt # Add the text "WRONG" to `wrong.txt`
git add . # Add the `wrong.txt` file
git commit -m "adding WRONG to wrong.txt" # Commit with the message "Adding WRONG to
```

```
echo More >> first.txt # Add the text "More" to `first.txt`
git add . # Add the `first.txt` file
git commit -m "adding More to first.txt" # Commit with the message "Adding More to fi

echo Even More >> first.txt # Add the text "Even More" to `first.txt`
git add . # Add the `first.txt` file
git commit -m "adding Even More to First.txt" # Commit with the message "Adding More

# OH NO! We want to undo the commit with the text "WRONG" - let's revert! Since this

git revert HEAD~2 # this will put us in a text editor where we can modify the commit

ls # wrong.txt is not there any more!
git log --oneline # note that the commit history hasn't been altered, we've just adde
```

If you revert to a file that has been edited since the commit you want to revert to, you will have a conflict that you will need to fix. You can read more about `git revert` here.

Git reflog

Finally, if you make a change like undoing a commit using `git reset`, or have reverted, or squashed and you want to undo that change, You can type `git reflog` and you will see previous changes you have made with unique SHAs. You can `git reset --hard SHA` to go back to a previous state. To see this in action, try rebasing the example above to squash the last two commits into one. Then take a look at the history you get back when you type `git reflog`.



‹ Previous next ›