codecademy

How to Backtrack in Git

Showing Latest Commit Log

In Git, the commit you are currently on is known as the `HEAD` commit.

The output of the `git show HEAD` command will display everything the `git log` command displays for the `HEAD` commit, plus all the file changes that were committed.

```
$ git show HEAD
commit
735359632f3ca3fe572484a4ec3e0d7b0d9c8f2
d
Author: codecademy
<exampleuser@codecademy.com>
Date:   Wed Jul 6 10:20:58 2016 -0400

    scene-5.txt

diff --git a/scene-5.txt b/scene-5.txt
index b12dd97..5dd5d4e 100644
--- a/scene-5.txt
+++ b/scene-5.txt
@@ -12,3 +12,7 @@ Hamlet:
I will.


+Ghost:
+My hour is almost come,
+When I to sulphurous and tormenting
flames
+Must render up myself.
\ No newline at end of file
```

Git Reset Using SHA

In Git, the `git reset commit_SHA` command can be used to set `HEAD` to the `commit_SHA` commit. The `commit_SHA` argument is the first seven digits of a previous commit's SHA. In this example, the `HEAD` was reset to the commit made on `Wed Jan 6`.

You can use `git log` to see a record of previous commits and their SHA values.

```
$ git log
commit
9d63f80111447544c303e9f1776fa08593a8731
0
Author: codecademy
<exampleuser@codecademy.com>
Date:   Wed Jan 13 18:55:53 2021 +0000

    Added updates to the file

commit
3ba6efbeece6ed530d85de5e313e52123fdf8cb
4
Author: codecademy
<exampleuser@codecademy.com>
Date:   Wed Jan 6 10:11:13 2021 -0400

    Completed first line of dialogue

$ git reset 3ba6efb
```

Staging Multiple Files

In Git, the `git add filename_1 filename_2` command is used to add multiple files to the staging area at once.
You can use `git status` to check if you properly added your files to the staging area.

```
$ git add scene-5.txt scene-7.txt
$ git status
On branch master
Changes to be committed:
  (use ""git reset HEAD <file>..."" to
unstage)

        modified:   scene-5.txt
        modified:   scene-7.txt
```

Remove File from Staging

In Git, the `git reset HEAD filename` command will remove `filename` from the staging area. Note that this command does *not* discard file changes from the working directory. You might use this command if you've added a file to the staging area, but the file includes incorrect edits.
You can use the `git status` command to make sure your file was properly removed from the staging area.

```
$ git reset HEAD scene-3.txt
Unstaged changes after reset:
M       scene-3.txt
```

Rolling Back to Last Commit

In Git, the `git checkout HEAD filename` command rolls back all changes that have been made to `filename` since the last commit. In other words, this command will change your working directory to look exactly as it did when you last made a commit.
You can use the `git diff` command to see if the rollback was successful. If `git diff` doesn't output anything, this means your working directory matches your last commit.

```
$ git checkout HEAD scene-5.txt
$ git diff
$
```