**Important Git Operations**

**head commit**

In Git, the *commit* you are currently on is known as the ***HEAD*** commit. In many cases, the most recently made commit is the HEAD commit.

To see the HEAD commit, enter:



The output of this command will display everything the *git log command* displays for the HEAD commit, plus all the file changes that were commited

**git checkout**

What if you decide to change the ghost’s line in the working directory, but then you decide you wanted to discard that change?

You could rewrite the line how it was originally, but what if you forgot the exact working? The command…



…will restore the file in your working directory to look exactly as it did when you last made a commit.

Here, *filename* again is the actual name of the file. If the file is named *changes.txt*, the command would be:



**more git add**

In Git, it’s common to change many files, add those files onto the staging area, and commit them to a repository in a single commit.

For example, say you want to change the character ‘larry’ to ‘laertes’ in the script. The name currently appears in two files. After you change the name in both files, you could add the changed files to the staging area with:



**git reset I**

What if, before you commit, you accidentally delete an important line from *scene-2.txt*? Unthinkingly, you add *scene-2.txt* to the staging area. The file change is unrelated to the Larry/LAERTES swap and you don’t want to include it in the commit.

We can *unstage* that file from the staging area using:



This command ***resets*** the file in the staging area to be the same as the HEAD commit. It does not discard the file changes from the working directory, it just removes them from the staging area

**git reset II**

Just like retracing you steps on a hike, Git enables you to rewind to the part before you made the wrong turn. You can do this with:



This command works by using the first 7 characters of the SHA of a previous commit. For example, if the SHA of the previous commit is ‘5d692065cf51a2f50ea8e7b19b5a7ae512f633ba’, use:

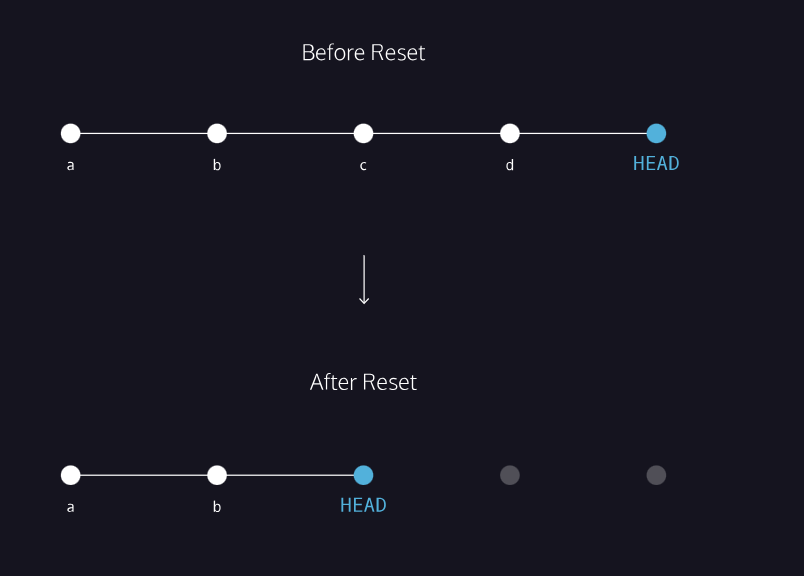


HEAD is now set to that previous commit

Note: SHA is used to uniquely identify commits. Each commit is assigned a SHA-1 hash, which is a 40-character long string. It stands for Secure Hash Algorithm

**git reset review**

To better understand the examples above, look at the diagram below. Each circle represents a *commit*.



Before reset:

* HEAD is the most recent commit

After resetting:

* HEAD goes to a previously made commit of your choice
* The grey commits are no longer part of your project
* You have in essence rewound the project’s history



This allows you to remove this file from the staging area