**Lab 3**

***Inspecting & Rolling Back Commits***

In this lab you will learn how to inspect previous commits and roll back changes. You’ll also use *gitk,* a GUI based program that allows you to view information about commits. This tutorial continues from Tutorial 2.

**Steps to Complete**

1. Open *foo.txt* and in the second line of text change “us” to “them”, then save.

**λ**  notepad foo.txt

1. Stage and commit this change. Note that we can stage and commit with a single command by using the “-a” flag with the commit command.

**λ**  git commit –a –m "Changed a word" –m "This should make it easier to follow"

1. Display all commits, most recent first by issuing this command:

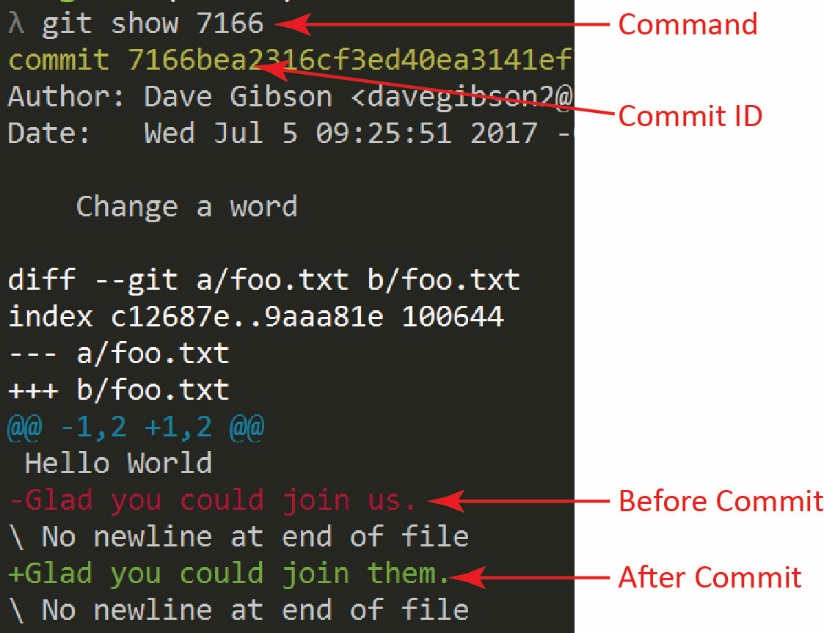
**λ**  git log

Remember: Press Spacebar to continue to next page, “q” to quit, when pagination results from a command.

1. (Read, no action required) To display exactly what was changed in a particular commit, use this syntax:

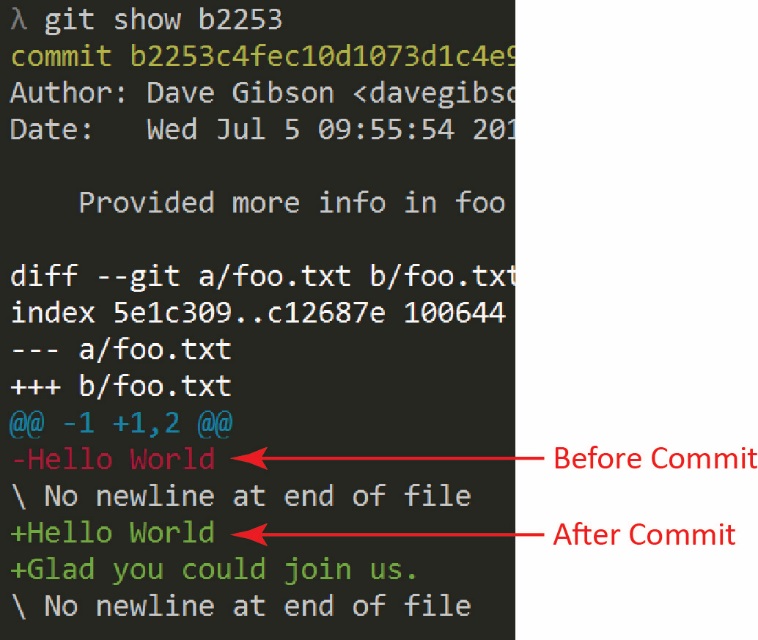
**λ**  git show *commitID*

For the *commitID* (see figure below on right) you can copy the entire string from *git log* or you can usually just type the first four characters assuming they are unique among all the commit id’s. For more information, see [documentation](https://git-scm.com/docs/git-show) for git-show.

1. Show the changes associated with the most recent commit (titled “Changed a word”). Be sure and use your commit ID:

**λ**  git show *commitID*

The result is shown on the right:

1. Next, show the changes associated with the second commit (title is “Provided more info in foo”). Be sure to use your particular commit ID:

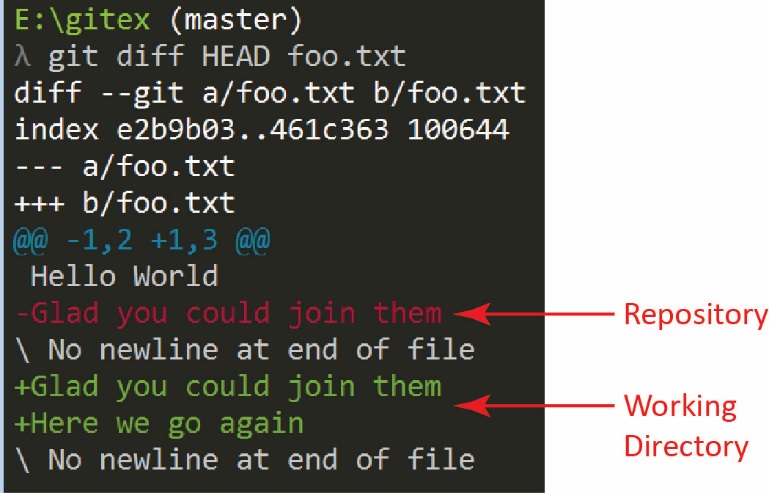
**λ**  git show *commitID*

The result is shown on the right. Depending on various things, including you experimenting with the commands, your next most previous commit might be different.

1. (Read, no action required) Git can be used to show the differences in different versions of files

|  |  |
| --- | --- |
| Example | Description |
| git diff HEAD [filename] | Compares the working directory with the local repository. In other words, show the changes after your last commit. Leaving the filename off shows differences among all files. |
| git diff [filename] | Compares the working directory with *index*. The *index* is the staged changes. In other words, you make changes on a file, stage them, then make more changes to the file – this command will show the differences. |
| git diff --cached [filename] | Compares the *index* with the local repository. In other words, show the difference between your last commit and changes to be committed next. |
| git diff commitID1 commitID2 | Show the differences between two commits. |

1. Do the following:
2. Add a line to the bottom of foo.txt: “Here we go again”



**λ**  notepad foo.txt

1. See the difference between foo.txt in the working directory and in the repository:

**λ**  git diff HEAD foo.txt

1. Stage the file

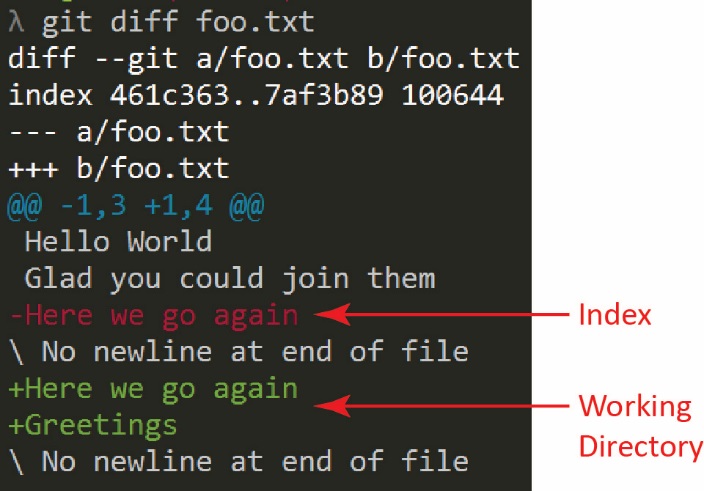
**λ**  git add .

or: **λ**  git add foo.txt

1. See the difference between the index (staged files) and the repository. Result is the same as above.

**λ**  git diff --cached foo.txt

1. Add another line to the bottom of foo.txt: “Greetings”



**λ**  notepad foo.txt

1. See the difference between the working directory and the index

**λ**  git diff foo.txt

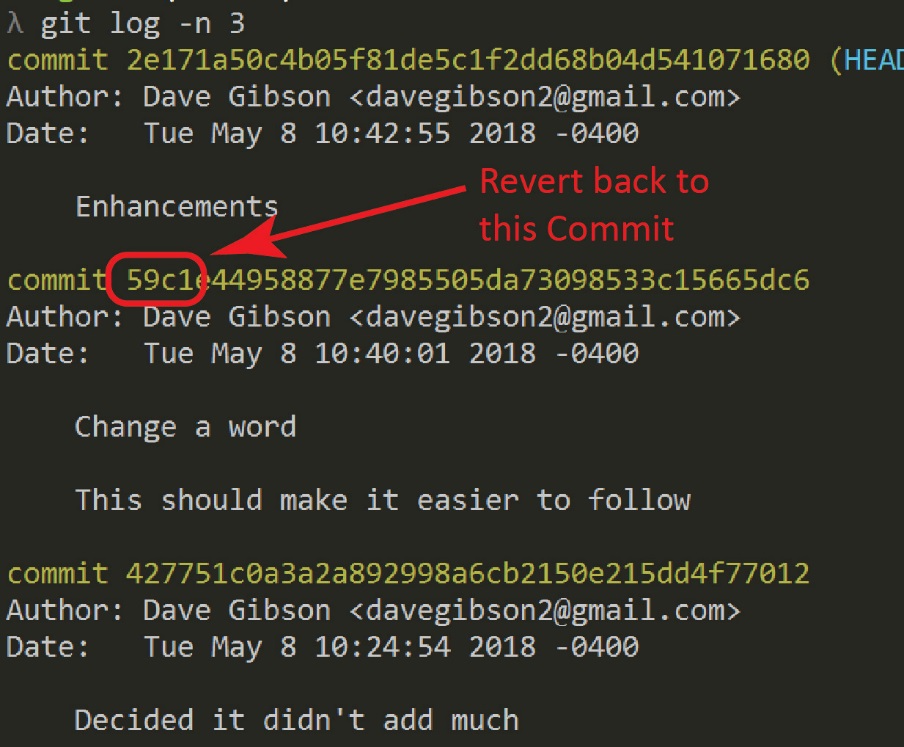
1. Stage and commit these changes:

**λ**  git commit -a -m "Enhancements"

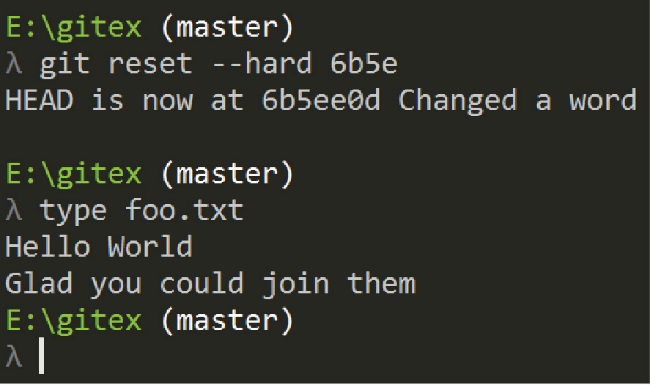
1. (Read, no action required) To roll-back all changes to the state immediately after a particular commit, use this syntax:

**λ**  git reset --hard *commitID*

In other words, the state will be exactly as if you had just made the commit with *commitID*. There are many other ways to use *reset.* For more information, see the [documentation](https://git-scm.com/docs/git-reset) for git-reset.

1. (Read, no action required) As you continue the tutorials, the log might get long, especially if you experiment beyond the steps of the tutorial. The command above: git log, will eventually paginate. You can tell this when you see a “:” in the lower portion of the window. Press <space> to see the next page, or “q” to quit and return to the prompt. One way to limit the number of commits displayed is to specify the number of most recent commits you want to display. For example, to display the last 5: git log –n 5
2. Roll-back the changes we just made (the last commit).
3. Get the commit ID for the second commit listed. Write down the first 4 characters or copy it.

**λ**  git log –n 3

1. Revert the changes. The result will be similar to that shown on the right.

**λ**  git reset --hard *commitID*

1. Verify the reset

**λ**  type foo.txt

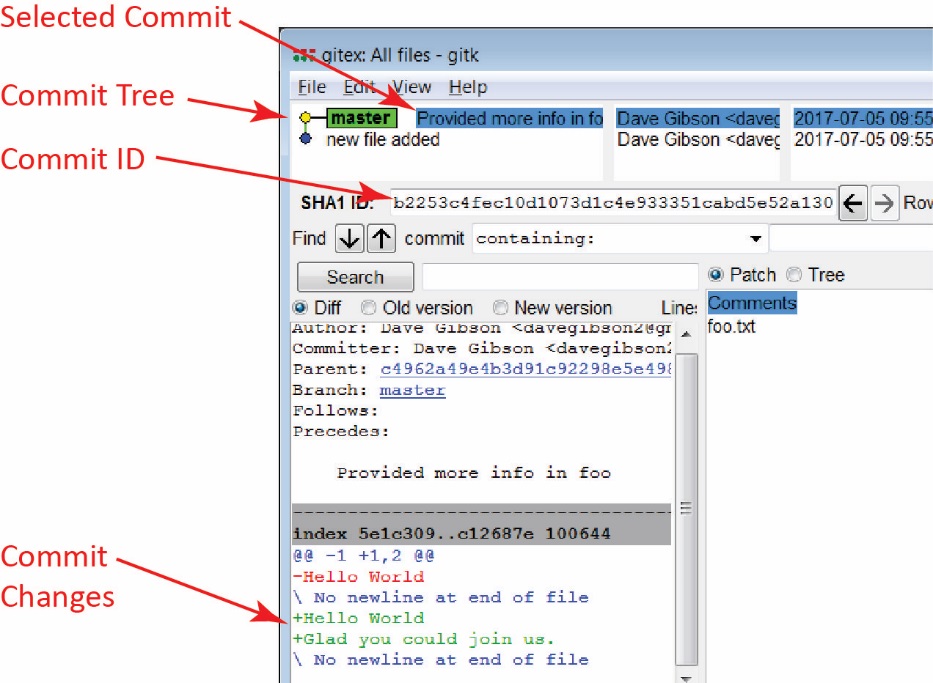
1. **Do the following:**
2. Make a screen shot similar to the one above on the right (showing the reset).
3. The image should be saved as a *jpg* or *png*.
4. Place the image in the *HW VCS* document in the appropriate place.
5. The image should easily readable without zooming in or out.
6. Continue with the tutorial.
7. Verify that the commit is removed from the log:

**λ**  git log –n 5

1. *gitk[[1]](#footnote-1)* is a windows-based Git repository browser that allows you to see changes. Display it now by typing:

**λ**  gitk

The result will look as shown below.



For more information, see the [documentation](https://git-scm.com/docs/gitk) for gitk.

1. Play with some of the options in *gitk*. For example:
2. Select the “Old version” and “New version” radio buttons.
3. Select the “new file added” node in the Commit Tree to examine that commit.
4. Close *gitk*.

Good idea: make a backup copy of your *gitex* folder with the name: *gitex\_3*. Then, you’ll start Lab 4 using the *gitex* repo.

1. <https://git-scm.com/docs/gitk> [↑](#footnote-ref-1)