Git commands have their own man pages. You can access them with git help [name of command]. For example here's the start of the help page for git status:

```
git help status

GIT-STATUS(1)                                   Git Manual
        GIT-STATUS(1)

NAME
       git-status - Show the working tree status

SYNOPSIS
       git status [<options>...] [--] [<pathspec>...]

DESCRIPTION
       Displays paths that have differences between the index file and the
         current HEAD commit,
       paths that have differences between the working tree and the index file,
         and paths in the
       working tree that are not tracked by Git (and are not ignored by
         gitignore(5)). The first
       are what you would commit by running git commit; the second and third are
         what you could
       commit by running git add before running git commit.
```

Just like any other help page that uses less, you can return to the prompt with the Q key.

If you want to see a list of your Git commits, enter git log into the console:

```
git log

## commit 12bb9f53b10c9b720dac8441e8624370e4e071b6
## Author: seankross <sean@seankross.com>
## Date:   Fri Apr 21 15:23:59 2017 -0400
##
##     added two files
##
## commit 73e53cae75301ce9b2802107b1956447241bb17a
## Author: seankross <sean@seankross.com>
## Date:   Thu Apr 20 14:15:26 2017 -0400
##
##     added readme.txt
```

If you've made many commits to a repository you might need to press the Q key in order to get back to the prompt. Each commit has its time, date, and commit message recorded, along with a SHA-1 hash that uniquely identifies the commit.

Git can also help show the differences between unstaged changes to your files compared to the last commit. Let's add a new line of text to readme.txt:

```
echo "The third line." >> readme.txt
git diff readme.txt

## diff --git a/readme.txt b/readme.txt
## index b965f6a..a3db358 100644
## --- a/readme.txt
## +++ b/readme.txt
## @@ -1,2 +1,3 @@
##  Welcome to My First Repo
##  Learning Git is going well so far.
## +I added a line.
```

As you can see a plus sign shows up next to the added line. Now let's open up this file in a text editor so we can delete the second line.

```
nano readme.txt
# Delete the second line
git diff readme.txt

## diff --git a/readme.txt b/readme.txt
## index b965f6a..e173fdf 100644
## --- a/readme.txt
## +++ b/readme.txt
## @@ -1,2 +1,2 @@
##  Welcome to My First Repo
## -Learning Git is going well so far.
## +I added a line.
```

A minus sign appears next to the line we deleted. Let's take a look at the status of our directory at this point.

```
git status

## On branch master
## Changes not staged for commit:
##   (use "git add <file>..." to update what will be committed)
##   (use "git checkout -- <file>..." to discard changes in working directory)
##
##  modified:   readme.txt
##
## no changes added to commit (use "git add" and/or "git commit -a")
```

If you read the results from git status carefully you can see that we can take this repository in one of two directions at this point. We can either git add the files we've made changes to in order to track those changes, or we can use git checkout in order to remove all of the changes we've made to a file to restore its content to what was present in the last commit. Let's remove our changes to see how this works.

```
cat readme.txt

## Welcome to My First Repo
## I added a line.

git checkout readme.txt
cat readme.txt

## Welcome to My First Repo
## Learning Git is going well so far.
```

And as you can see the changes we made to readme.txt have been undone.

✓ Complete