Basic File Commands

coursera.org/learn/git-distributed-development/supplement/PBju3/basic-file-commands

git add

Adds one or more files as blobs to the object store and adds references to the new blobs in the index, by **sha1** hash. (Remember, the index contains a complete picture of the state of a project at a given time.) Thus, we are staging one or more files and directories. It can be used either to add new files or stage changed ones. The files will not be committed until there is a **git commit** done.

There are a number of options (see **git help add**). For instance, you can use **-i** for interactive choosing of files to stage, or **-u** to update only files already known to git. You can also specify wildcard patterns or directory names (in which the entire subdirectory tree is added).

It is worth repeating that until you do a **git commit** the changes are only staged; the repository is not updated.

git rm

Removes a file from the working tree and the index. This can be pretty dangerous if you do not realize what you are doing. If you only want to remove the file from the working directory, and not the repository index, just use the normal **rm** command.

While you remove files from the repository, you do not remove them from the history, as that would be dishonest. If you want to remove a file that has been staged but not committed, you have to add the **-cache** option, as in:

```
1

2

$ git add myfile

$ git rm myfile --cached
```

git mv

Renames a file and stages the new filename in the repository. It is equivalent to renaming the working file, and then doing a **git rm** on the old file name and a **git add** on the new one; i.e. the following operations are equivalent:

```
1

2

$ git mv oldfile newfile

$ mv oldfile newfile ; git rm oldfile ; git add newfile
```

This is an easier operation in git than it is in older revision control systems, where renaming a file means actually deleting the old one and adding a new one. In this case, the binary blobs associated with the file remain the same, only the index is updated.

Here is a table that shows how all three of these stages work (changes induced by basic **git** file commands):

Command	Source Files	Index	Commit Chain	References
git add	Unchanged	Updated with new file	Unchanged	Unchanged
git rm	File removed	File removed	Unchanged	Unchanged
git mv	File moved/renamed	Updates file name/location	Unchanged	Unchanged

git ls-files

Shows information about files in the index and working tree. By default, this command shows only files in the repository. If you want to show the untracked files, you can do:

```
1

$ git ls-files --others --exclude-standard
```

where the **--others** option shows the untracked files, and the **--exclude-standard** option says to ignore standard exclusions such as **.gitignore** files.