Help Topic: The Git Add and Remove Commands

Spring 2022

Oscar Viquez, oviquezr@mit.edu
Michael Benjamin, mikerb@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139

The Git `add` and `remove` Commands

The `git add` command allows you add files or a directories to your locally checked out tree in preparation for committing them to the Git server. And as you may guess, the `git rm` command tags a file in your local checkout for removal on the server upon your next commit. More info on `git add` and `git rm` can always be found by Googling "Git book" and reading the full PDF online free, or just typing `git help add` anytime on the command line.

Basic usage of the git `add` command

The basic syntax for the `git add` command is:

```
$ git add my_file.cpp my_file.h        #(one or more files)
$ git add my_project/                  #(a directory)
```

The `git add` command does *not* require a network connection to the server and only declares an intention to add file(s) to the master copy on the Git server upon the next commit and push. Since `git status` also doesn't require a network connection to the server, you can confirm the results of your `add` immediately:

```
$ git status
Your branch is up to date with 'origin/main'.

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   my_file.cpp
    new file:   my_file.h
    new file:   my_project/my_project.cpp
    new file:   my_project/my_project.h
```

Note: When adding a directory, the operation is applied recursively to all files and subdirectories.

Note (1): When adding a file, don't be afraid to use the command line conveniences such as `git add *.pdf`. All the previously unversioned PDF files will now be scheduled for addition.

Note (2): Git does not track empty directories, only files. If you need to ensure the availability of an empty directory in your repository, you can create a `.gitignore` file within that directory, and

add that file to version control.

Basic usage of the Git `remove` command

The basic syntax for the `git rm` command (`rm` for "remove") is:

```
$ git rm my_file.cpp my_file.h        #(one or more files)
$ git rm -r my_project/               #(a directory; use -r for recursive)
```

The `git rm` command does *not* require a network connection to the server and only declares an intention to remove the file(s) from the master copy on the Git server upon the next commit. It will also delete the local copy.

Note (1): If removing files with staged changes (to be committed later), the `git rm` command will produce an error. The message will point out the files that have been staged and inform you that you can either:

- Unstage the file with the `git rm --cached <file>` command, but keep the local copy.
- Force removal of the file with the `git rm -f <file>` command, which will unstage the file and remove the local copy.

Note (2): If removing files that are unversioned (not under version control), the `git rm` command will produce an error. The message will point out that Git does not currently track any information about that file and thus cannot remove it from the repository. The local file can be removed using the (non-git) `rm` command.

```
$ touch test.txt
$ git rm test.txt
fatal: pathspec 'test.txt' did not match any files
$ rm test.txt
```