GIT COMMANDS:

//Command to initialize and know the status of git:

*git init*

*git status*

//use this command to add file to git folder.

*git add <file>*

/\*

To store our staged changes, we run the commit command with a message describing what we've changed. Let's do that now by typing

\*/

*git commit -m "Add i have just added a file jkm"*

//add all the new files using a wildcard with git add. Don't forget the quotes!

*git add '\*.txt'*

/\*

There's git log. Think of Git's log as a journal that remembers all the changes we've committed so far, in the order we committed them. Try running it now:

\*/

*git log*

//Go ahead and run git remote add with the options below:

*git remote add origin https://github.com/try-git/try\_git.git*

/\* The name of our remote is origin and the default local branch name is master. The -u tells Git to remember the parameters, so that next time we can simply run git push and Git will know what to do. Go ahead and push it!

\*/

*git push -u origin master*

/\* We can check for changes on our GitHub repository and pull down any new changes by running:

\*/

*git pull origin master*

/\* In this case we want the diff of our most recent commit, which we can refer to using the HEAD pointer.

\*/

*git diff HEAD*

/\*Let's use git add to stage octofamily/octodog.txt, which I just added to the family for you.

\*/

*git add octofamily/octodog.txt*

/\*Good, now go ahead and run git diff with the --staged option to see the changes you just staged. You should see that octodog.txt was created.

\*/

*git diff --stage*

/\*So now that octodog is part of the family, octocat is all depressed. Since we love octocat more than octodog, we'll turn his frown around by removing octodog.txt.

You can unstage files by using the git reset command. Go ahead and remove octofamily/octodog.txt.

\*/

*git reset octofamily/octodog.txt*

/\*Files can be changed back to how they were at the last commit by using the command: git checkout -- <target>. Go ahead and get rid of all the changes since the last commit for octocat.txt

\*/

*git checkout -- octocat.txt*

/\*When developers are working on a feature or bug they'll often create a copy (aka. branch) of their code they can make separate commits to. Then when they're done they can merge this branch back into their main master branch.

\*/

/\*We want to remove all these pesky octocats, so let's create a branch called clean\_up, where we'll do all the work:

\*/

*git branch clean\_up*

/\*Great! Now if you type git branch you'll see two local branches: a main branch named master and your new branch named clean\_up.

\*/

/\*You can switch branches using the git checkout <branch> command. Try it now to switch to the clean\_up branch:

\*/

*git checkout clean\_up*

\*/

Removing All The Things

Ok, so you're in the clean\_up branch. You can finally remove all those pesky octocats by using the git rm command which will not only remove the actual files from disk, but will also stage the removal of the files for us

\*/

/\*You're going to want to use a wildcard again to get all the octocats in one sweep, go ahead and run:

\*/

*git rm '\*.txt'*

/\*

Commiting Branch Changes

Now that you've removed all the cats you'll need to commit your changes.

Feel free to run git status to check the changes you're about to commit.

\*/

*git commit -m "Remove all the cats"*

/\*

Switching Back to master

Great, you're almost finished with the cat... er the bug fix, you just need to switch back to the master branch so you can copy (or merge) your changes from the clean\_up branch back into the master branch.

\*/

//Go ahead and checkout the master branch:

*git checkout master*

/\* Preparing to Merge

Alrighty, the moment has come when you have to merge your changes from the clean\_up branch into the master branch. Take a deep breath, it's not that scary.

We're already on the master branch, so we just need to tell Git to merge the clean\_up branch into it:

\*/

*git merge clean\_up*

/\*

Keeping Things Clean

Congratulations! You just accomplished your first successful bugfix and merge. All that's left to do is clean up after yourself. Since you're done with the clean\_up branch you don't need it anymore.

\*/

/\*You can use git branch -d <branch name> to delete a branch. Go ahead and delete the clean\_up branch now:

\*/

*git branch -d clean\_up*

/\*

The Final Push

Here we are, at the last step. I'm proud that you've made it this far, and it's been great learning Git with you. All that's left for you to do now is to push everything you've been working on to your remote repository, and you're done!

\*/

*git push*