GIT BASIC COMMANDS

**git init** //terminal prompt is currently in directory we decided to initialize a Git repository here.

**git status** //to see what the current state of our project is.

**git add octocat.txt** //tell Git to start tracking changes.

**git add -A .** //where the dot stands for the current directory, so everything in and beneath it is added.

**git add '\*.txt'** //use wildcards if you want to add many files of the same type, also in subdirectory.

/\*

staged:

Files are ready to be committed.

unstaged:

Files with changes that have not been prepared to be commited.

untracked:

Files aren't tracked by Git yet. This usually indicates a newly created file.

deleted:

File has been deleted and is waiting to be removed from Git.

\*/

**git reset octofamily/octodog.txt** //unstage files.

**git checkout -- octocat.txt** //files can be changed back to how they were at the last commit.

**git commit -m "Add cute octocat story"** //To store our staged changes to repository

// we run the command with a message describing what we've changed.

/\*

Staging Area:

A place where we can group files together before we "commit" them to Git.

Commit:

A "commit" is a snapshot of our repository. This way if we ever need to look back at the changes we've made

(or if someone else does), we will see a nice timeline of all changes.

\*/

**git log** //a journal that remembers all the changes we've committed so far.

**git log --summary** //to see more information for each commit.

**git remote add origin https://github.com/try-git/try\_git.git** //to push our local repo to the GitHub server.

//.This command takes a remote name and a repository URL.

**git push -u origin master** //push our local changes to our origin repo (on GitHub).

//The name of our remote is origin and the default local branch name is master.

//The -u tells Git to remember the parameters.

**git push**

**git pull origin master** //check for changes on our GitHub repository and pull down any new changes.

**git stash** //to stash your changes.

**git stash apply** //to re-apply your changes after your pull.

/\*

sometimes when you go to pull you may have changes you don't want to commit just yet.

One option you have, other than commiting, is to stash the changes.

\*/

**git rm '\*.txt'** //not only remove the actual files from disk, but will also stage the removal of the files for us.

**git rm -r folder\_of\_cats** //recursively remove all folders and files from the given directory.

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

**git diff HEAD** //look at what is different from our last commit.

**git diff --staged** //to see the changes you just staged

**git branch new\_branch** //create a copy (aka. branch) of their code they can make separate commits to.

/\*

separate the code base into two "snapshots" (branches) and work on and commit to them separately.

As soon as one was ready, you might merge this branch (new\_branch) back into the master branch and push it to the remote server.

\*/

**git branch** //to see local branches.

**git checkout new\_branch** //switch to a new branch.

//do something.

//back to master branch.

**git merge new\_branch** //merge your changes from the new\_branch branch into the master branch.

**git branch -d new\_branch** //delete a branch.