Git course

This is reading notes from an interactive introduction course from <https://try.github.io/>

A more complete book is at <https://git-scm.com/book/en/v2/.>

$git init create the local repository of project

$git status show state of the project



$git add octocat.txt stage a file in master repository

$git status to be called regularly before committing to show state of the project

$git commit –m "Add cute octocat story" commit staged file with descriptor

$git add '\*.txt' stage multiple files with wildcard, recursively in subfolders

* We can use get add –A . to add everything in the current directory and beneath. The flag –A ensures that even file deletions are staged
* We can use git reset <filename> to remove a file, or several files from the staging area.

$git status

$git commit –m "Add all the octocats stories" commit staged files with descriptor

* A commit is a snapshot of our repository. This way if we ever need to look back at the changes that we made (or someone else did) we will see a nice timeline of all changes.
* We need quotes with wildcards so that Git will receive the wildcard without our shell can interfere with it. Without quotes our shell will only search within the current folder, and Git will receive a list of files the shell found, instead of the wildcard, and it will will not be able to add the files inside the subfolders.
* When using wildcards, you want to be extra careful when doing commits. Make sure to check what files and folders are staged by using egt status before you do the actual commit. This way you can be sure you're committing only the things you want.
* Use git log --summary to seemore infromation on each commit. You can see where the new files were added for the first time or where the files were deleted. It's a good overview of what's going on in the project.



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

* Remote repositories are versions of your project that are hosted on the Internet somewhere. Collaborating with others involves managing these remote repositories, pusing and pulling data when you need to share work.
* With this command we make git associate the current local branch with the remote branch named origin and located at the URL
* The default remote name is origin. The default local branch name is master.
* git remote add origin <URL> is identical to git clone <URL>
* git clone implicitly creates the remote branch origin
* git remote add <name> adds the remote named <name>.



$git push –u origin master

* With this command we push the local branch master to the remote origin.
* -u makes git to remember the parameters so that next time we simply run git push.
* We push only the files that are committed, not the one that are staged.
* If the remote is the URL of a web server, there is no need to use ftp to sync the server with the local HTML. Just use git push.
* If someone else has pushed to origin since the last time you pulled from origin, you have to pull and merge the changes with your master (pull is identical to fetch + merge) before doing push.

$git pull origin master

* Fetches and merges the changes in the remote origin with the local branch master.
* Sometimes you may have local changes that you want to commit only after the pull. Use git stash to stash your changes and git stash apply to re-apply your changes after the pull.
* In Github, as soon as you have committed changes in a branch off of master, you can open a pull request. When you open a pull request, you’re proposing your changes and requesting that someone review and pull in your contribution and merge them into their branch. Pull requests show diffs, or differences, of the content from both branches. The changes, additions, and subtractions are shown in green and red.

$git diff HEAD

* Looks for the differences between the current master (after a pull for example) and the most recent commit, pointed to by HEAD.

$git add octofamily/octodog.txt

$git diff --staged show changes in files that were just staged.

$git reset octofamily/octodog.txt unstage the file octodog.txt.

$git checkout -- octocat.txt get rid of all the changes since the last commit of octocat.txt.

$git branch clean\_up clean\_up is the name of the branch.

* Creates a branch on which changes will be committed and eventually be merged with master when ready, and then pushed to the remote.

$git branch show the 2 branches.

$git checkout clean\_up switch the working branch to clean\_up.

$git 'rm \*.txt' delete all txt files from the disk and stage the deletions.

$git commit –m "Remove all the cats" commit the deletions.

* Use git commit –am "…" if you have manually deleted files without using git rm. The flag –a will apply the deletion of files to the git tree.

$git checkout master switch the working branch back to master.

$git merge clean\_up merge the clean\_up branch with the current working branch master.

$git branch -d clean\_up delete the branch clean\_up.

* $git branch -df clean\_up and $git branch -D clean\_up : force deletion of a branch that has not been yet merged

$git push push the current local branch to the remote.