**GitHub**

Create Repositories

$ git init [project-name]

$git clone [url]

Makes Changes

$ git status (lists all new or modified files to be omitted)

$ git diff (shows file differences not yet staged)

$ git add [file] (snapshots the file in preparation for versioning)

$ git commit –m “[descriptive message]” (records file snapshots permanently in version history)

Group Changes

$ git branch (lists all local branches in the current repository)

$ git branch [branch-name] (creates new branch)

$ git checkout [branch-name] (switches to the specified branch and updates the working directory)

$ git merge [branch] (combines the specified branch’s history into the current branch)

$ git branch –d [branch-name] deletes the specified branch)

Review History

$ git log (lists version history for the current branch)

Synchronize Changes

$ git push [alias] [branch] (uploads all local branch commits to Git Hub

$ git pull (downloads bookmark history and incorporates changes)

GitHub Tutorial

$ git init – initializes a Git repository

$ git status – see what the current state of your project. Also to see how the repository status has changed. It’s healthy to run this often.

$ git add filename.txt – add to staging area

$ git commit –m “Add description of change” – store changes from stage and adds a message describing what was changed

$ git add ‘\*.txt’ – add many files of the same type using a wildcard

$ git log – history/journal that remembers all the changes that have been committed

$ git remote add origin <http://github.com/try-git/try_git.git> - uses a remote name and repository URL to add files to

$ git add –A . – adds all. The dot stands for the current directory, so everything in and beneath it is added. The –A ensures even file deletions are included.

* When using wildcards, make sure to check the status to see what files and folders are staged before you actually commit.

$ git log --summary – see more information for each commit. You can see where new files were added for the first time or where files were deleted. It’s a good overview of what’s going on in a project.

$ git push –u origin master – push tells Git where to put our commits. The name of remote is “origin” and the default local branch is “master”. The –u tells Git to remember the parameters the next time we run $ git push.

$ git pull origin master – if other people have changed files and made their own commits, we can pull the new changes

$ git stash – when you got to pull, you may have changes you don’t want to commit yet; you can stash the changes. For WIPs, when you don’t want to commit and maybe switch to a different branch.

$ git stash apply – re-apply your changes after your pull

$ git stash list – see what stashes you’ve stored

$ git stash apply stash@ {2} – specify a specific stash. If not, Git assumes the most recent stash.

$ git diff HEAD – find what is different from our last commit

* HEAD – a pointer that holds your position within all our different commits. By default, it points to your most recent commit.
* Commit etiquette – try to keep related changes together in separate commits

$ git add folder\_anem/filename.ext

$ git diff --staged – see changes you just staged

$ git reset filename – removes a file or files from the staging area

$ git checkout – filename.txt – files can be changed back to how they were at the last commit

$ git branch\_name – make a branch where you’ll be doing all the work

$ git branch – see your branches

$ git checkout\_name – switch branches

$ git checkout –b new\_branch – checkout and create a branch at the same time

$ git rm ‘\*.txt’ – removes files from disk and stages the removal of the files

$ git rm –r folder\_name

$ git commit –am “Deletes files” – if you delete a file (from your computer) without using ‘git rm’, you’ll still have to ‘git rm’ the deleted files from the working ree. You can save a step by using the ‘-a’ option which auto removes deleted files with the commit.

$ git checkout master – switch back to master branch

$ git merge nonmaster\_name – merge branch w/ master

$ git branch –d branch\_name – deletes a branch. For when you merge it with your master and don’t need it anymore.

$ git branch -- force (-f) branch\_name – or $ git branch –D branch\_name – force deletes when you want to scrap an idea and delete a branch and all of its contents

$ git push – push everything you’ve been working on to Git