A Summary of Commonly Used Commands

There are hundreds of different Git commands, but to get started you only need to remember a handful of them. Here is a summary of the commands you'll use most often:

- `git init` initializes your local directory as a new git repository. You must run this before you can commit any of your work.
- `git status` shows the current status of your repo. It will show you if you have any work that is unstaged, what branch you are on, how many commits you are ahead of the master remote on github, and other useful things.
- `git diff` shows you the changes in your unstaged code.
- `git remote -v` shows you all the remotes for your repo. The `v` stands for verbose, which shows you the URL of the repository on github, if any, that your local repository is pointing to rather than just the name of the remote repo.
- `git add .` takes all unstaged work and stages it, making it ready to be committed. You can also specify a particular file to stage with `git add file-path/name-of-file`
- `git commit -m "write commit message here"` commits all staged work. It's important to write a brief, clear commit message so you know what each commit is for. "Final commit" is not the commit message you're looking for exactly 100% of the time.
- `git pull` once you've committed all your local work and running `git status` shows that you have nothing to commit, you pull down any changes from your remote. By default, this will pull from the `origin` remote's `master` branch. To be specific about which remote and branch to pull from, you can use: `git pull name-of-remote name-of-branch`
- `git push` pushes your local changes up to your remote. By default, this will push to the `origin` remote's `master` branch. Like pull, you can push to a specific remote and branch

with: `git push name-of-remote name-of-branch`. This is useful if you are using **branches** and **pull requests**. If you get an error message, it's probably because you haven't pushed your local branch up to github yet. Try `git push -u name-of-remote name-of-branch`.

- `git branch` shows you all your local branches and indicates which branch you are currently on.
- `git checkout -b name-of-new-branch` makes a new branch and switches to that branch.
- `git merge name-of-branch` will merge the specified branch into the branch you are currently on.
- `git branch -d name-of-branch-to-delete` deletes the specified branch
- `git log` will show you the full list of commits and authors for your repo
- `history` will show you your past git commands
- `git stash` stashes any unstaged changes in your repository. They will not be present in your codebase, but they are not deleted.
- `git stash pop` gives you back the last staged changes you stashed
- `git blame file-path/name-of-file` shows you line-by-line who wrote the code in the specified file. Useful when you have a question about how something works and want to figure out who to ask, and also great source of shame when you realize you wrote the chunk of code you've been swearing at for the last hour.