

A Quick Git How-To

I keep forgetting how to set up and use Git. If I write everything down in one place then I won't forget anymore. You might like it too. This is just the basics.

1 How to set up for the first time

I trust you can install Git and make a Github account.

Open up the command prompt and navigate to your folder where you want to start using Git. Use `cd` and then your folder path downwards. If you need to go up, use `cd ..` to go up one folder. Or you can just paste the full folder path after `cd`.

In your folder, start by typing `git init`. Git has now been initialized.

To tell Git which files to ignore, create a new text document in your folder and rename it to `.gitignore`. For every file you'd like Git not to track, add its name to a new line in `.gitignore`. You can also ignore entire file types, like for example `.pdfs`, with an asterisk: `*.pdf`. (You generally should only use Git to track source code.) And you can ignore entire folders and their contents by adding `folder_name/`.

Next, go to Github and create a new repository. After you've created it, it will give you a URL. Go to the command prompt and type `git remote add origin that_URL`. I think you could call it something other than `origin` but I don't and I'm pretty sure most people don't either.

2 Tracking changes, committing, and uploading

The command `git status` tells you what files have changed since your last commit, as well as comparing your local branch to the remote repository `origin`. Red entries are untracked, while green are ready to commit.

To tell Git what to track for the next commit (and turn the red entries green), use `git add *` to add everything (that has not been explicitly ignored), or use `git add file_name` to track a specific file. Unless you have a good reason, configuring the files you want to ignore and using `git add *` is best.

The command `git commit -m "message"` saves a snapshot of your changes and the "message" parameter lets you add an explanation as to what you changed and why. Many people neglect to put helpful "message"s but it's really valuable so don't neglect it, especially the why. Commit often.

To send your commits to Github, use `git push -u origin trunk`. (I use `trunk` instead of any alternatives, but this is determined by what's set up on your Github repository.)

3 Branches

Use branches when you want to work on a new feature or topic and want to wait to include it until it's finished.

To create a new branch locally, type `git checkout -b new_branch_name`. Changes you commit will now commit to the branch `new_branch_name`. To push them, use `git push -u origin new_branch_name`.

To switch back and forth between existing branches, use `git checkout name_of_branch`, where `name_of_branch` is the branch you'd like to switch to.

To merge two branches, first switch to the branch that you want to remain. Typically that's `trunk`, so we'll just assume it is from here on. That means first you type `git checkout trunk`. Next use the command `git merge name_of_branch`. If the old branch is no longer needed (if you are merging then this is often the case – merge to `trunk` once you're overall happy with what you've completed), then you can delete it with `git branch -d name_of_branch`.

If you are working on your Git project solo and only editing one branch at a time, then you should not experience any merge conflicts. If you do, it's probably best to resolve in Github, not the command prompt.

4 Miscellaneous

The command `git log` lets you see the complete changelog, but this can also be done on Github directly (preferably).

Using `git diff` shows you the changes you've made since your last commit. If you are committing often like you should, this will be manageable. Press `q` to quit if you get stuck on this screen. Github also shows you the `diffs` for each commit.

If the project already exists remotely on Github but you'd like to set it up locally on a new machine, instead of starting with `git init`, you should use `git clone that_URL`. You should be good to go after that; nothing else necessary from Section 1.

If you're working with someone else and they've pushed updates to Github that you'd like to have locally, use the command `git pull origin trunk`. This is a combination of two commands: `git fetch origin trunk` which grabs the changes from Github, and `git merge origin/trunk`, which attempts to merge to your current branch (see Section 3). Also you should probably be using pull requests if there's a possibility you'll step on each others' toes.