Github tutorial:

Other links for learning more:

<https://git-scm.com/downloads>

^ Go to this link, and download the Git Version control program

<http://rogerdudler.github.io/git-guide/>

is a pretty solid simplistic tutorial-guide for understanding the basics of git

(I’ve highlighted all the commands you need to enter like this so that you don’t accidentally miss a step. There’s a full command list at the bottom)

If the command line doesn’t make sense to you, you can try

<https://www.gitkraken.com/> or

<https://desktop.github.com/>, which are git GUIs so that you don’t have to use the command line at all. I’ve never used them, so I have no idea how they work.

Setup:

Download git from the link above and install it.

MAC: Open terminal (If you don’t know where it is, hit command-enter, type in ‘terminal’ (without quotes), and hit enter.

WINDOWS: Open command line. (if you don’t know where it is, hit Windows+R, type in ‘cmd’ and hit enter)

MAC: Enter ‘ls’. This will show you the files in your current folder.

WINDOWS: Enter ‘dir’. This will show you the files in your current folder.

Both:

You can switch folders using ‘cd’. So if your documents folder is in your current folder, enter ‘cd Documents’ and you will move to that folder. In your Documents folder, you should make a new folder called ‘SRC’. Once you are in your documents folder, enter ‘mkdir SRC’ and the new folder will be created. ‘cd SRC’ to enter the folder.

When you are in the correct place where you want to keep your new code folder, enter ‘mkdir SeniorDesignGithub’ and then enter ‘cd SeniorDesignGithub’. You can change ‘SeniorDesignGithub’ to anything you want to name the folder.

Enter ‘git init’, and git will set-up your folder for use.

Enter ‘git clone <https://github.com/AlasdairMcLean/MATLABprototyping.git>’.

Git will ask you for your username and password.

When you enter your password, the characters won’t appear- this is normal. It’s a security thing so that nobody sees your password, and your password isn’t stored in a log where somebody can find it.

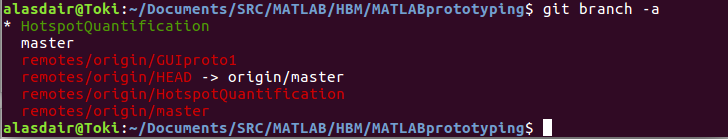
When done, you will have all the files that are currently uploaded on github, and you can see all the code.

Before you start coding, you should make a new ‘branch’ which is an offshoot of the main ‘master’ branch, allowing you to make changes and merge back into the main branch later. This way, if multiple people are working on the code at the same time, you don’t have to worry about accidentally changing something that breaks somebody else’s code. In your new branch you can alter the files, change them, and experiment without altering the main branch that everyone else is working out of. To make a new branch:

Enter ‘git checkout –b mynewbranchname’, or whatever you want to call your branch. If you want to publish your branch so that other members of the group can access it, then enter

‘git push origin mynewbranchname’.

To see ALL branches, type in ‘git branch -a’, which will show you all the current branches in the repository.

Here’s how the branches work:

the ‘master’ branch is the top-level branch where all of the features will eventually be added. The idea is that master should always work no matter what, and it should be tested to make sure there aren’t any conflicts. Master is basically the most stable version of the entire repository that we have.

The branch marked with a \* (also green in my terminal) is the branch that you are currently working in. Any changes you make will go to that branch, and no others.

The red branches that start with remotes/origin are the branches that are online on github.com. There’s a remote/online version of all branches that have been updated. You can see in the picture that even though I am in the HotspotQuantification branch, there’s also a remotes/origin/HotspotQuantification branch.

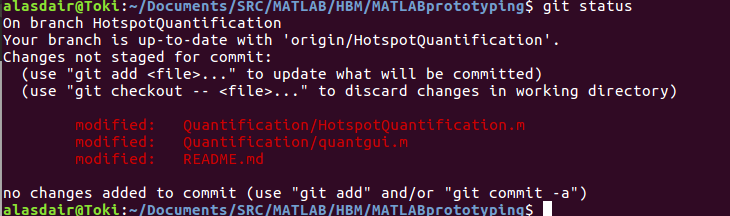
If you want to checkout somebody else’s branch (for example the ‘GUIproto1’ branch) then the command is ‘git checkout -b GUIproto1 origin/GUIproto1’. Make sure that you don’t forget the last ‘origin/GUIproto1’ or else github won’t actually pull the changes from the online branch.

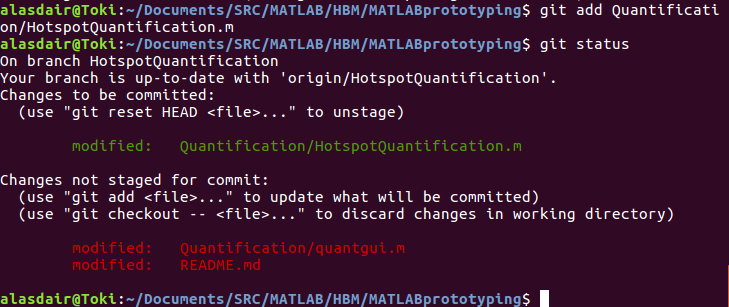
Once you’ve checked out branches for the first time, they are stored locally on your computer. If you type in ‘git branch’ without the ‘-a’, then you can see all the branches that are stored locally on your computer. You can switch between any of those branches freely by typing in ‘git checkout master’ or any other branch name.

Making changes:

If you make changes or a new file:

First, enter ‘git status’

Once you make changes, you have to manually tell github which changes you want to upload. You can see above I have three files that are modified and are not ‘staged for commit’. Everytime you upload your code and changes to the github repository, you upload using a ‘commit’. You add your changes to the commit, and then finally push them to the internet. Before you actually commit, you ‘stage’ your changes by entering ‘git add filename’.



You can see above that once I added the file and rechecked the status, git updates that file to green, indicating that when you commit your changes, Quantification/HotspotQuantification.m will be updated too. If you add something that you don’t actually want to update, enter ‘git reset HEAD Quantification/HotspotQuantification.m’ to ‘unstage’ that change.

When you are finally ready to commit, the command is ‘git commit -m “Commit message here”’ (make sure you put the commit message in double quotes). The commit message that you write will be seen by everyone- it’s a description of the changes you made. So if you fixed a bug or added a new file, then you should write that so that everyone knows what you did. Once your commit is done, finally enter ‘git push origin HotspotQuantification’ or whatever branch you are on and want to update. Everyone will be able to see your code now.

Updating with changes others made

If somebody else writes some changes that you want to add, use ‘git pull’ to pull those changes from github.com. Make sure that you ONLY do this when you don’t have local changes, or else git won’t let you pull the changes (you could accidentally overwrite your work this way). Alternatively, look into ‘git stash’ online for an alternative.

**Cheat sheet:**

|  |  |
| --- | --- |
| Command | Action |
| git init | Sets up a folder on your computer to support git commands |
| git clone [url] | Clones a repository from github.com so that you have access on your computer |
| git branch | Lists all of the local branches. Does not show online branches. |
| Git branch -a | Lists all local and online branches |
| Git checkout [branchname] | Switches your current working branch to branch ‘branchname’. |
| Git branch -b [branchname] origin/[branchname] | Creates a new branch that is identical to the branch online called ‘branchname’ |
| Git branch -b [branchname] | Creates a new local branch but does not pull anything from online. Only use this command if you are making a completely new branch- one that is not found online. |
| Git pull | Checks github for updates to the repository and downloads them to your system. |
| Git add [filename] | Stages the file ‘filename’ to your next commit, so when you commit that file is uploaded to the github.com repository. |
| Git reset HEAD [filename] | Un-stages the file ‘filename’ from your next commit so that it won’t be uploaded when you commit |
| Git commit -m “commit message here” | Commits your changes with the message ‘commit message here’, which everyone will see online |
| Git push | Pushes your commits to the internet so everyone has access to your code and changes |
| Git status | Checks the status of your next commit and shows you all tracked and untracked files |