/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*A git/github tutorial\*

\*\*\*\*@author:michael\*\*\*\*

\*\*\*\*@date:Jun-10-15\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Hi, and welcome to a brief introduction to git and GitHub. We, in Dr. Colton's lab, switched from using SVN to git to track our changes on lab files because Google Code, where our code was stored, is shutting down. We switched to GitHub and also switched to git.

git is revision control software created by Linus Torvald. Although it is typically run through the command line, you do not need to be a 'l33t hacker' to be able to use it thanks to available GUIs. This tutorial will teach you the skills you need to be able to use git in our lab.

This is intended to be a living document so future users can add, edit, or remove information as necessary. If you need to contact me, the original author, see my contact information in Appendix C.

Chapter 0: terms to know

Some of the words I use might sound confusing so here’s a reference if you’re confused (or, you know, you could just Google it).

‘repo’ or ‘repository’ – where the code is stored. Our various repos can be found at

<https://github.com/coltonlab>

commit – a snapshot of a project at a certain time. This stores all the information about what a project looks like at the time the commit was created

branch – a moveable pointer to a commit. Most commits are sequential, but occasionally you may want to “branch” off from a certain commit. This usually happens when multiple users are working on different parts of a project and will both be working on something for a while. We don’t really use branches in our lab but if the need arises that can be added to this tutorial.

Chapter 1: 'git'ting started

The first thing you'll need to do is get git. If you're using OS X, it should already be installed. If you're on a Linux distro, you need to run

sudo apt-get install git

Odds are, though, that you're using Windows. That's OK. I forgive you. If you're on one of the lab computers git should already be installed. I know. I did it myself. If you're not on a lab computer or you're not sure right-click on a file and see if you see any options that start with 'Git'. If you don't, go to the following URL:

<http://git-scm.com/download/win>

and your download should begin immediately. git will install like a normal .exe

Once you have git installed, You're going to want to get a GUI. Unfortunately on Windows the GUIs like to install themselves inside C:\Users\your-username\appdata so you'll have to do this even if someone else has previously installed a GUI on the same computer. I recommend GitHub (which is related to but not the same as github.com). On Windows you can download it from the following URL:

<https://windows.github.com/>

If you don't already have one, you need a GitHub.com account. You guessed it, here's the URL:

<https://github.com/join>

Once you have a GitHub.com account get yourself added to the coltonlab organization. Send your GitHub.com username to John or me (if I’m still around) or someone else in the research group and they should be able to get you set up as far as that’s concerned. Once you’ve done that, open the GitHub GUI and log in with your GitHub.com credentials. It should already pull in all the configuration you’ll need automagically so you should be able to skip the configure step.

Once you’ve done that you’ll need to add our repositories to the GUI. Click the “+” in the top left corner and select “Add.” If you are on the lab computer, navigate to one of our repositories, select it, then click the “Add Repository” check mark. Our repositories on the lab computer are located at:

C:\LabVIEW-programs  
C:\LabVIEW-tutorial

Once you’ve done that, you’re all ready to start using git with LabVIEW.

If you are not on one of the LabVIEW computers, to add the repository you will click the “+” in the top left corner of the GUI and select “Clone.” It should pop-up with a window that shows your personal GitHub.com account as well as the coltonlab organization. Select the organization and then pick the repo you want to clone. Click the check mark and then browse to the folder in which you want to clone the repo.

Chapter 2: using git

The GitHub GUI is extremely simple. You should have an icon on your desktop now that links right to it. Open it up and select the repo in which you are planning on making changes. In the top right corner there is a magic “Sync” button. If it has a number with a down arrow, there are commits in the repo that are ahead of your local copy. Click “Sync” to pull the commits. You’ll want to “Sync” BEFORE you make changes, as well as BEFORE you try to commit changes.

If you have made changes, you will see “Uncommitted changes” above the “History” pane. Click “Show” and it will show the files that have been changed. Unfortunately most of the files in our repository are binaries (like, for example, this .doc file – I’m sorry; I wanted it to be a text file but this seemed like a better solution due to the use of hyperlinks) so you won’t be able to see the differences between your version and the previous version.

When you’re ready to commit your changes, add a commit message to the “Summary” box. (“Description” is optional.) Now, click “Commit to master.” You’re created a commit, meaning you’ve made a “snapshot” of the project at this moment. Notice the “Sync” button? It now has a number with an up arrow. Click it to push your changes to the repo.

Chapter 3: merge conflicts

The biggest pain in any revision control software is merge conflicts. These can be prevented if you pull commits before working on new changes and push your changes when they are finished. Hopefully merge conflicts will not happen in this lab since it is unlikely that two users will be working on different copies of the same file at any given time.

@TODO: I don’t know how the GitHub GUI handles merging/conflicts or really how to deal with merge conflicts on binary files, so I’ll figure this out and add this section later.

Chapter 4: branching

@TODO: If it becomes necessary, add this section.

Appendix A: the git GUI and Windows Explorer

git for Windows includes some tools for using git in the file explorer as well as a presumably more lightweight GUI for committing, etc.

@TODO: Add documentation for using this method

Appendix B: git bash/git shell

git for Windows and the GitHub GUI come with terminal-like software called git bash and git shell, respectively. For users familiar with using git from a terminal, this may be a preferred method.

@TODO: Documentation?

Appendix C: contact me

If you have any questions for me, your best bet at getting in touch is to speak with John.