**MEEM 4990/5990: Getting Data in the Lab**

**Git Instructions**

Instructor: Jason D. Sommerville, Ph.D.

jdsommer@mtu.edu

Jan. 14, 2016

# Overview

Git is a distributed version control system. Version control means that it allows multiple versions of source files (or other files) to be maintained in a repository, allowing for one to revert to prior version of the code or do more advanced tasks like branching and merging.

# Get git

Students will need to install the git revision control software on their machines. This is a free, open source RCS which can be downloaded from

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

Download the version appropriate to your machine. When you install git you will have a few options. I recommend the following:

* **Checkout as-is, commit as-is**: don’t modify line endings. This controls how line endings, which are not standardized between operating systems are handled. This will not affect us significantly as LabVIEW Vis are binary files.
* **Run git from the Windows command line**: Allows you to run git from both the git bash shell and the windows command line, but doesn’t do anything to your system you may find unhelpful.

The basic git install comes with the git bash shell and a basic git GUI. I will be using the git bash shell exclusively for teaching purposes. However, you may find that you like a GUI better for your use. As long as you understand what you’re doing with the various commands, I don’t care how you invoke them. You may wish to check out other git GUIs besides the default here

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

Personally, I like the TortoiseGit GUI.

**Github**

Additionally, you will need to create an account at www.github.com. Preferably, use the same username as your MTU username, but this is not required. Once your account is created, please send me an e-mail with your full name and your github account username.

Once you have sent me your github username, I will forward it on to the maintainer of the MichiganTech organization on github. He will extend an invitation to you, which you receive via the e-mail account you registered with github. After accepting the invitation, I will be able to add you to the me5990 team within the MichiganTech organization. If you haven’t been added to the team in a timely fashion, you may request to join it by going to

<https://github.com/orgs/MichiganTech/teams>

and navigating to the me5990 team. There you will find a “Request to Join” button which will notify my that you are ready to join the team. Once you are added, you will have access to the class repository, which is located at

<https://github.com/MichiganTech/me5990_spring2016>

(Prior to being added to the team, you will find that the link appears to be a bad URL.)

Our MichiganTech organization administrator has asked that you fill out your profile on github, including appropriate, professional-looking photo. Do that via the “Your profile” link as shown below.

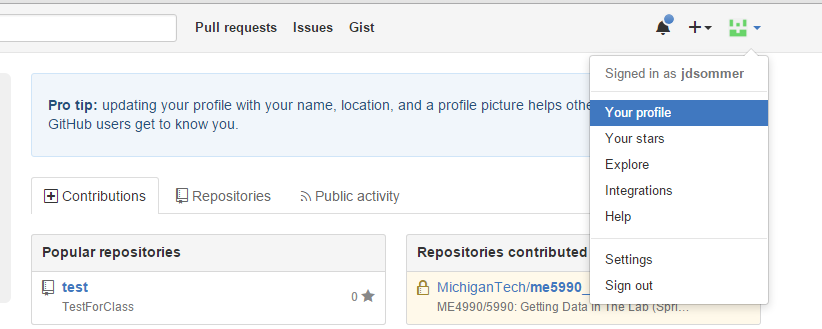


Figure 1: How to update your github profile

# Cloning and pulling the MichiganTech/me5990\_spring2016 repository (optional)

Once you’ve been granted access to the me5990 repo, you may directly clone it to your hard drive. This would allow you to view files in the repo locally on your machine, but, due to the permissions on the me5990 repo, you would not be able to push any changes you make back into the repo. None-the-less, if, for instance, you wanted to maintain a pristine copy of the me5990 repo on your hard drive, bring up the git bash shell, navigate to the directory in which you wish to create the copy of the repo, and issue the command

git clone https://github.com/MichiganTech/me5990\_spring2016 me5990\_pristine

Issue this command only once. This will create a copy of the repo in the “me5990\_pristine” subfolder of your current working directory. To pull future commands down, navigate to me5990\_pristine and issue the command:

git pull –u origin

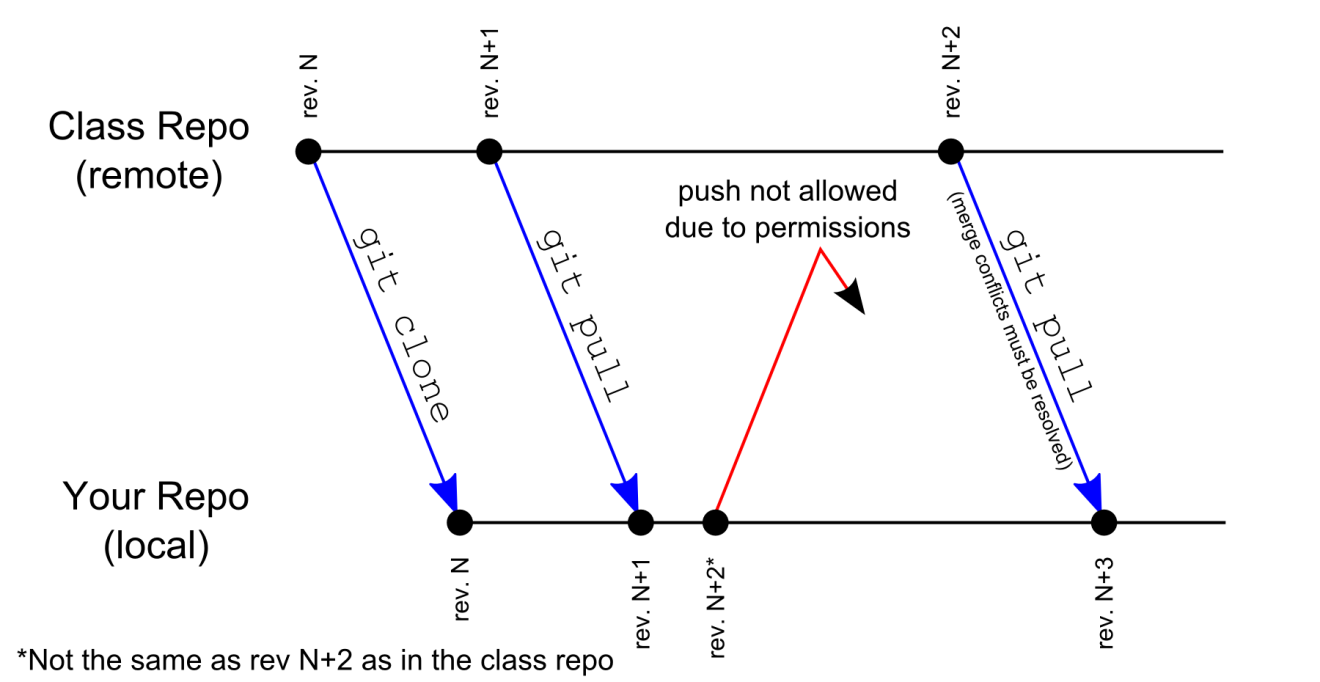


Figure 2: Cloning and pulling the class repository. Each dot represents a commit to the repository.

# Forking the MichiganTech/me5990\_spring2016 repo

In order to preserve appropriate read-only permissions on the class repo, yet allow you to share your changes with me for the purposes of grading, we are going to set up a more complicated system, involving a second github repository that you create and maintain. Your personal github repo should be forked from the class repo. Forking is conceptually the same thing as cloning a repo, but must be done through the github website.

After logging into your account, navigate to the class github repo

<https://github.com/MichiganTech/me5990_spring2016>

You will see a “Fork” button in the upper right. Press it.

If you have any options after pressing this (and you may not) choose to create the fork under your own user. This will allow you to create an identical copy of the repo, but in your own github account, rather than in the MichiganTech organization. After creating the fork, the screen will look nearly identical, except instead of saying MichiganTech/me5990\_spring2016 it will show <username>/me5990\_spring2016. This is your own repo which is linked back to the class repo, but you have complete control over. Most notably, you control who can access it. You must give me at least read access so that I can see into your github repo in order to grade your assignments. To do this, click the Settings tab, then Collaborators and Teams, and then add me (jdsommer) to your repo.



Figure 3: Giving your instructor access to your repo



Figure 4: Forking the class repo

# Working with your github repo (your remote)

Now that you’ve created the repo, you need to clone it down to your local machine so that you can actually use it. This will following the same commands used in the previous (optional) section, except you’ll point the clone command to your copy of the repo. Using the git bash shell, navigate to the directory in which you wish to create the copy of the repo, and issue the command

git clone https://github.com/<username>/me5990\_spring2016

Issue this command only once. This will create a copy of the repo in the “me5990\_pristine” subfolder of your current working directory. To pull future commands down, navigate to me5990\_pristine and issue the command:

git pull –u origin

Now, you go about your business viewing and editing the files. You may commit to your local repository (the one on your hard drive) as often as you like. (We will get to checkout and commits momentarily.) However, I will not be able to see anything you’ve committed until you push the changes up to your remote (github) repository. To do that, make sure that you’ve committed everything you want to submit locally, and then use the command:

git push –u origin

Provided that no one else is editing your remote repository (did you give anyone else permission to do so)? This push should have no conflicts, as the only changes that have been made have been made on your local repository. You can execute this command as often as you like to keep your remote in sync with your local. ***Be sure to do it as the last step of any homework submission.***

# Pulling updates from the class repo into your remote repo

Now that you have your own fork of the class repo, you need to keep it in sync. If the class repo is updated, you should see a notification when you inspect your remote repo on github (Figure 5). If you click the “Pull Request” link, you’ll have the opportunity to correct this. Unfortunately, the default pull request that comes up is backwards, and may tell you that there are no changes in your repo to push into the class repo (which you are not allowed to do anyway). You will need to change the direction by swapping the head and base repos by clicking the “switching the base” link (Figure 6). Once you’ve done that, you will be presented with a summary of the changes, and can click on the “Create Pull Request” button (Figure 6). This creates a request that changes from the class repo be pulled into your repo. In the general case, this is to give all of the developers on your repo an opportunity to discuss the proposed pull. However, since you are the only developer on your repo, the discussions should be pretty short. A few more clicks will get you to a screen labeled with a button labelled “Merge Pull Request,” and “Confirm Merge,” which will complete the pull and merge in the changes from the class repo to your repo.

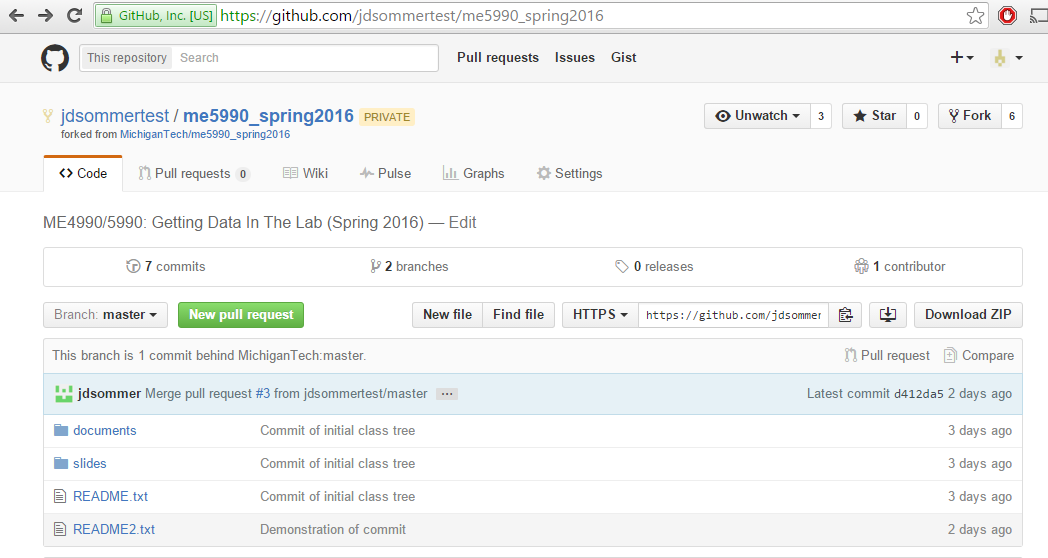
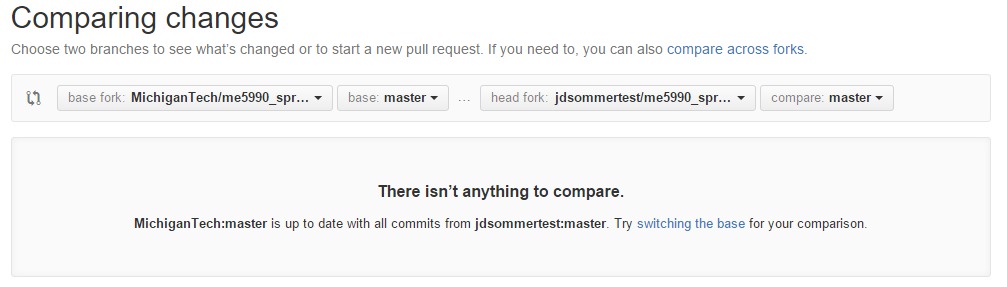


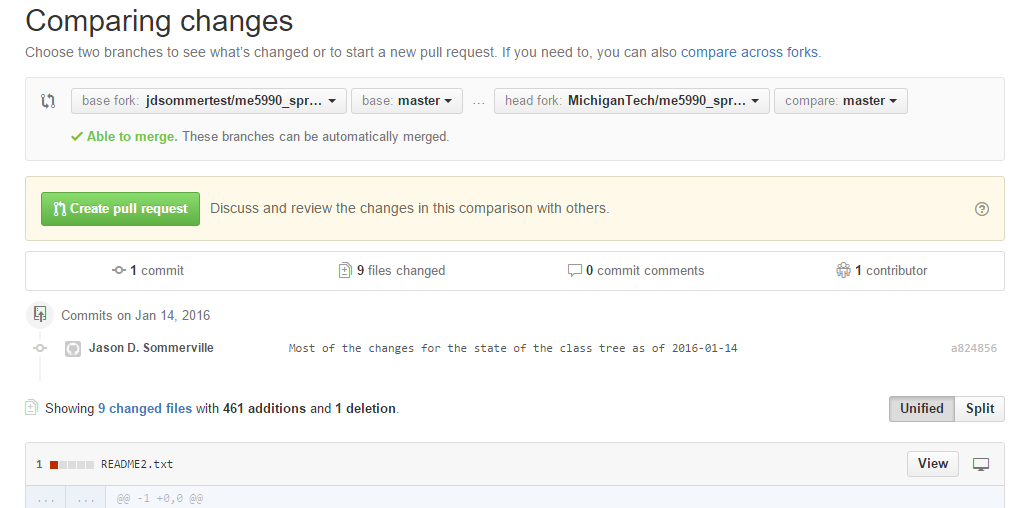
Figure 5: Notification that your repo is behind the MichiganTech master

If you commit changes to your remote repo without first pulling the changes from the class repo, you may find that your remote repo is both ahead and behind the class repo (see Figure 7). This is not, in and of itself, a problem. Create a pull request just as before. If the “switch the base” link isn’t showing, and you need to switch the head and the base branches, don’t panic. Switch one of the two, and then click “Compare across forks” (to get the repo drop-downs back) and then switch the second (see Figure 8). Then complete the creation and acceptance of the pull request as before.

If when you try to do this, you have a merge conflict, you will not be able to complete the merge without some more hassle. A merge conflict means that the same file was edited both in your repository as well as in

Figure 6: Switching the base and creating the pull request.





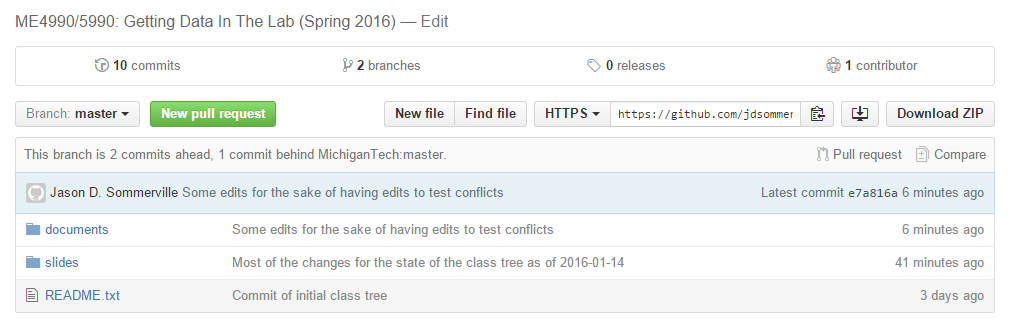


Figure 7: My github repo is both ahead and behind the class repo. I made a commit to the repo without pulling changes down from the class repo prior to editing my files.

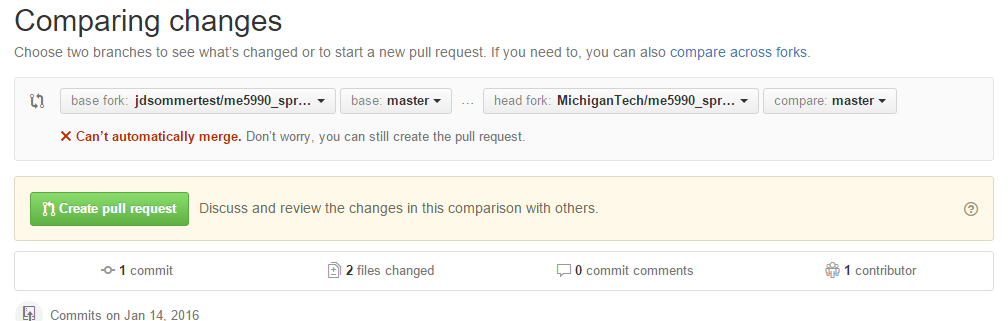


Figure 8: Create a pull request when there’s a conflict. Make sure that the base fork is your remote repository.