**How to Set Up Git for the Greger Lab: A Visual Guide**

**What is Git?**

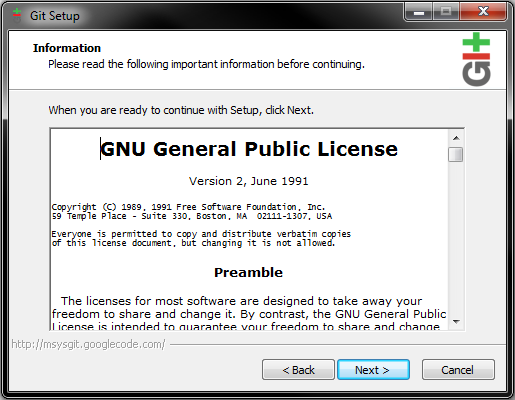
* Git is both a Source Code Management (SCM) and a Distributed Revision Control (DRC or DVC) system and was developed by Linus Torvalds, the same man who initially developed Linux. Git allows multiple users to share, alter, and backup code in tandem.

**Installation of Git**

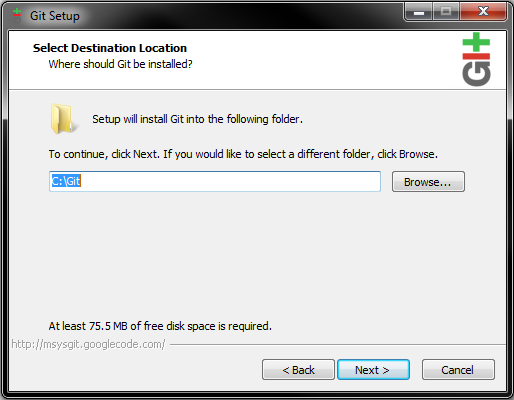
* Download the Windows version of Git from this address: <http://git-scm.com/downloads>
* Run the installer and follow along with the images below.
* ‘Next’



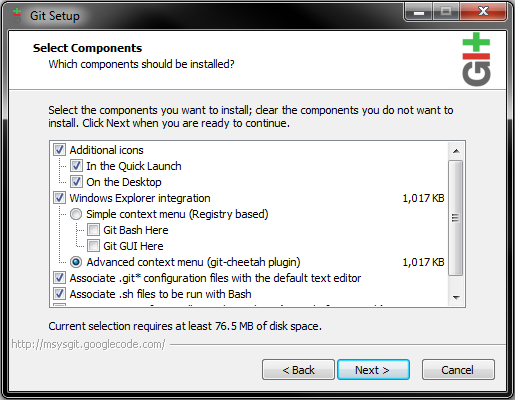
* ‘Next’



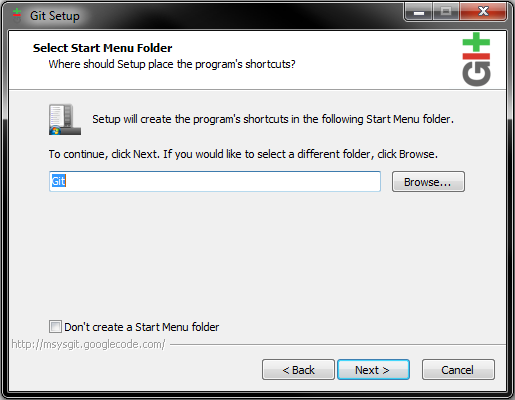
* Install the program to C:\Git



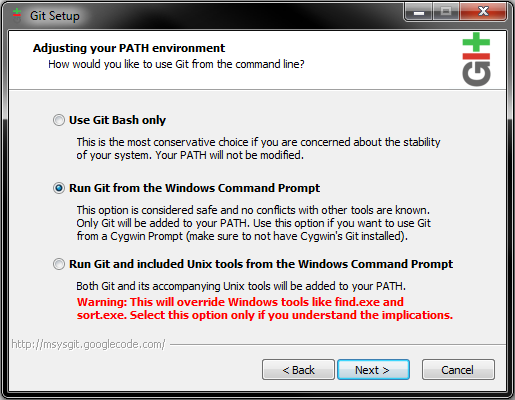
* Leave these options as the default options.



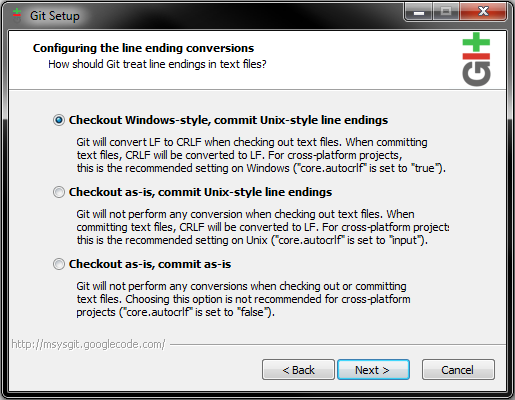
* ‘Next’



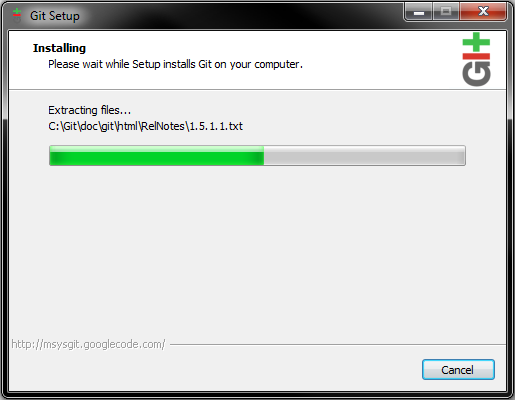
* **IMPORTANT:** Be sure to select the second choice: “Run Git from the Windows Command Prompt. It will make your life easier.



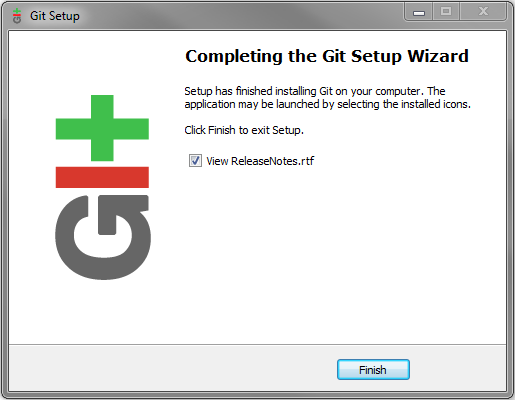
* Make sure the first radio button is selected.



- Begin the installation.

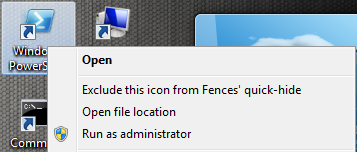


* Git is now installed on your system! It is now time to begin setting up the work environment.



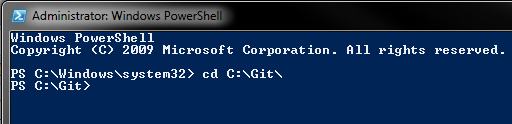
**Setting Up the Work Environment**

* Now that Git is installed we need to tell it what user is using it as well as change the Execution Policy for PowerShell
* Right click on the PowerShell icon and select “Run as administrator’



* Change directory to C:\Git\ with the command:

cd C:\Git\



* Type in the following command to see the Execution Policy for PowerShell scripts. The result returned should be ‘Restricted’. This means that no scripts will be allowed to run. However, we do want scripts to run.

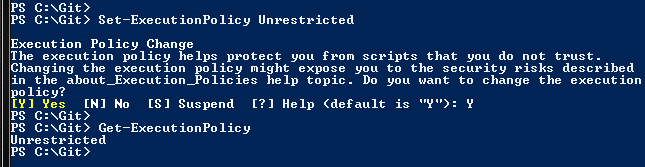
Get-ExecutionPolicy



* Type in the following command to allow PowerShell to run all scripts. Please not that this is dangerous as PowerShell is incredibly powerful, and there are those who would misuse this awesome power. Don’t run scripts from people you do not trust. (PowerShell scripts have the .ps1 extension)

Set-ExecutionPolicy Unrestricted

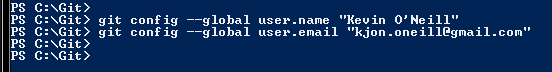
Then type in ‘Y’ to confirm the change.



* Enter in the following two commands so that Git knows who you are. This information is not sent anywhere; it is merely a way to identify your contributions to the lab code on the server. When the code breaks we can use the history feature to see who was the last committer; then subsequently, and politely, asking them to fix it.

git config --global user.name “Firstname Lastname”

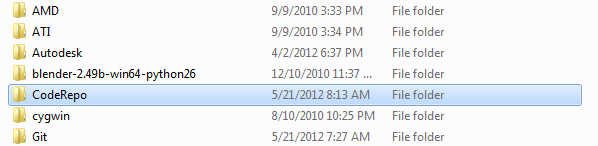
git config --global user.email “YourEmail@Location.Ext”



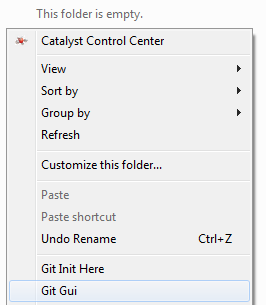
* The work environment is now set up and we can begin downloading code.

**The Initial Pull**

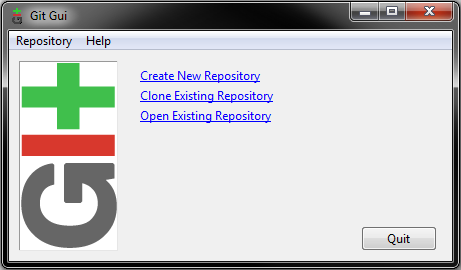
* What we now need to do is create a local repo directory and clone one of the already existing repos.
* With Windows Explorer change directory to a location where you want to store code and create a folder named ‘CodeRepo’. Depending on the computer station a good result would be D:\CodeRepo\



* Go inside CodeRepo and right click. A context menu should appear with some options for Git. Select Git Gui.



* While the Git Gui is active, select the ‘Clone Existing Repository’ option.

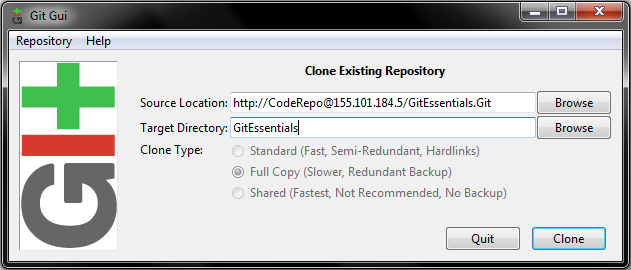


* Enter in the Location of the remote GitEssentials repo and the destination:

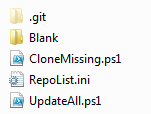
Remote: http://CodeRepo@155.101.184.5/GitEssentials.Git

Location: GitEssentials

Password: 1234



* This will clone (copy) the GitEssentials repo to the CodeRepo folder. Once the cloning process is complete open the GitEssentials folder. There will be a number of files. Run the CloneMissing.ps1 script. This will clone all missing repos to the CodeRepo Folder. Please input passwords as needed.



* With this all of the repos should be available to work with. Happy coding!