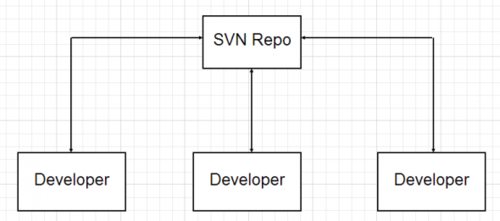
**A Guide to using GitHub**

Overview:

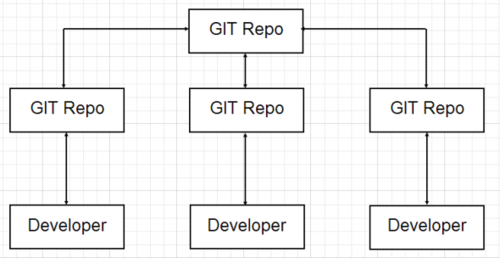
Having the right source code management system (CMS) is essential to allow developers to access and change different parts of code without interfering with each other. In addition, a good CMS offers version control, release notes, and a backup of the code, if needed.

Subversion (SVN) and GitHub are two well-known centralized control systems. The workflow and version control for each of these systems is different. SVN is a centralized version control system whereas GitHub is a distributed version control system. SVN does not have a central repository or server while GitHub uses multiple repositories that include a centralized repository. GitHub has a staging area where it’s possible for 100 changes in code to be broken into 10 or 20 commits.

Below is a breakdown of the SVN workflow:



Below is a breakdown of the GitHub workflow:



Purpose:

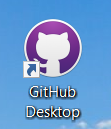
The IT environment here at GLFHC has endorsed the GitHub model. Therefore, this document outlines the ways that application can be used at GLFHC to augment the existing source code management processes and procedures. It describes how to install GitHub Desktop, create local or clone existing GitHub repositories, and upload new or modified code. Hopefully, this information will make it easier for others to use this system.

Step 1 – Install GitHub Desktop

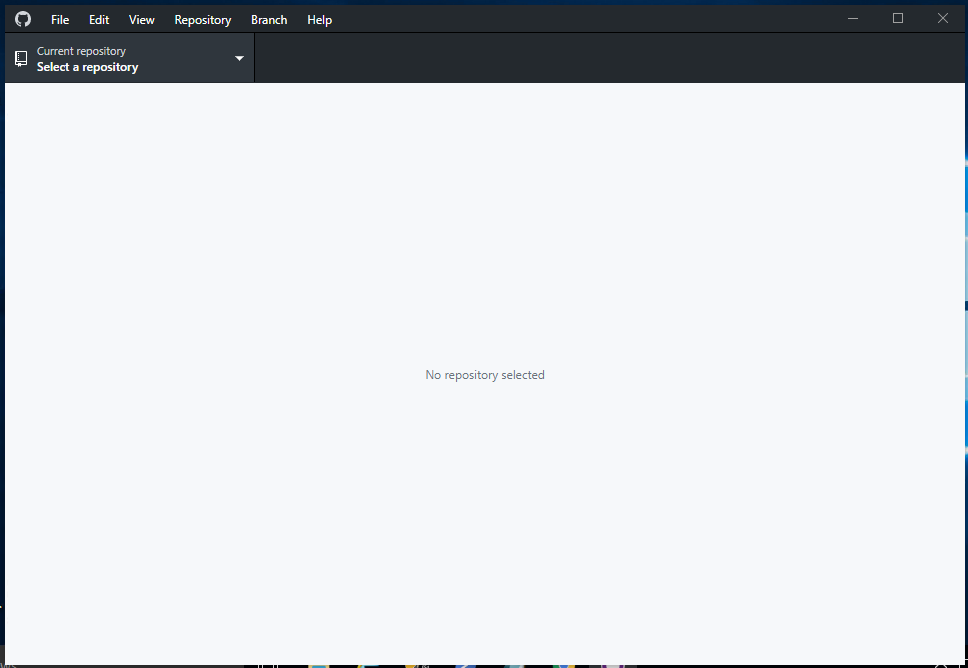
Follow the directions to download and install the GitHub Desktop application on your working environment. That could be a workstation or laptop. A complete set of instruction can be found here:

<https://docs.github.com/en/desktop/getting-started-with-github-desktop/installing-github-desktop>

Step 2 – Once this application is installed, double-click on this icon:



The first time that you perform this operation, you will have to log into your GitHub account. Thereafter, it will log you in directly and you will see a browser screen like this one:

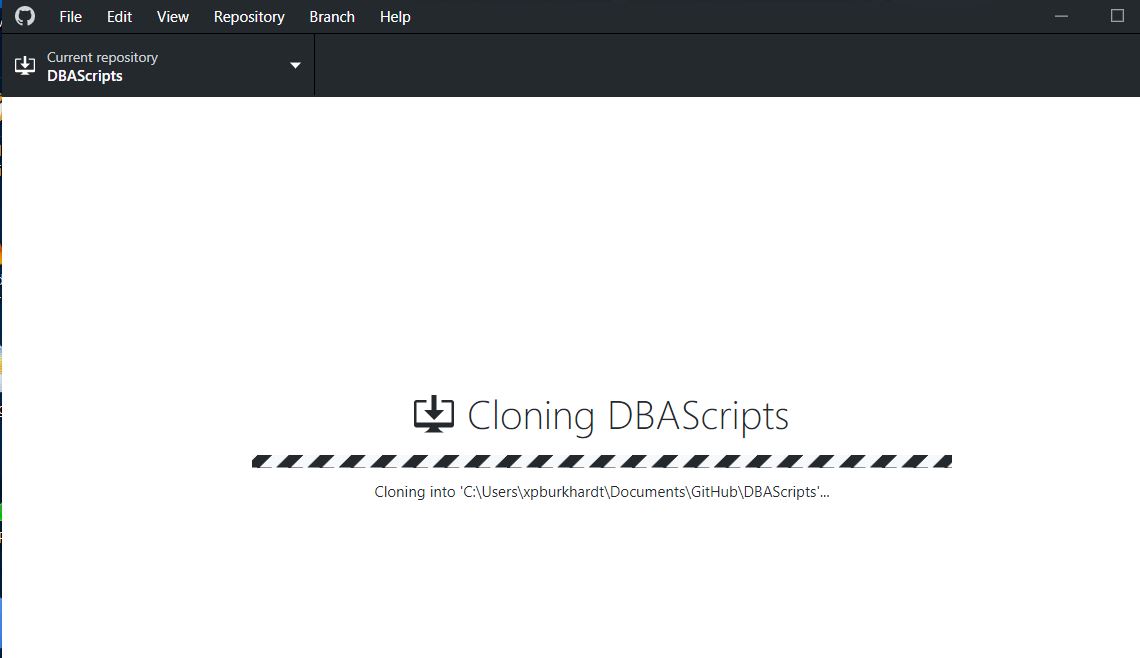


Step 3 – Clone a Repository:

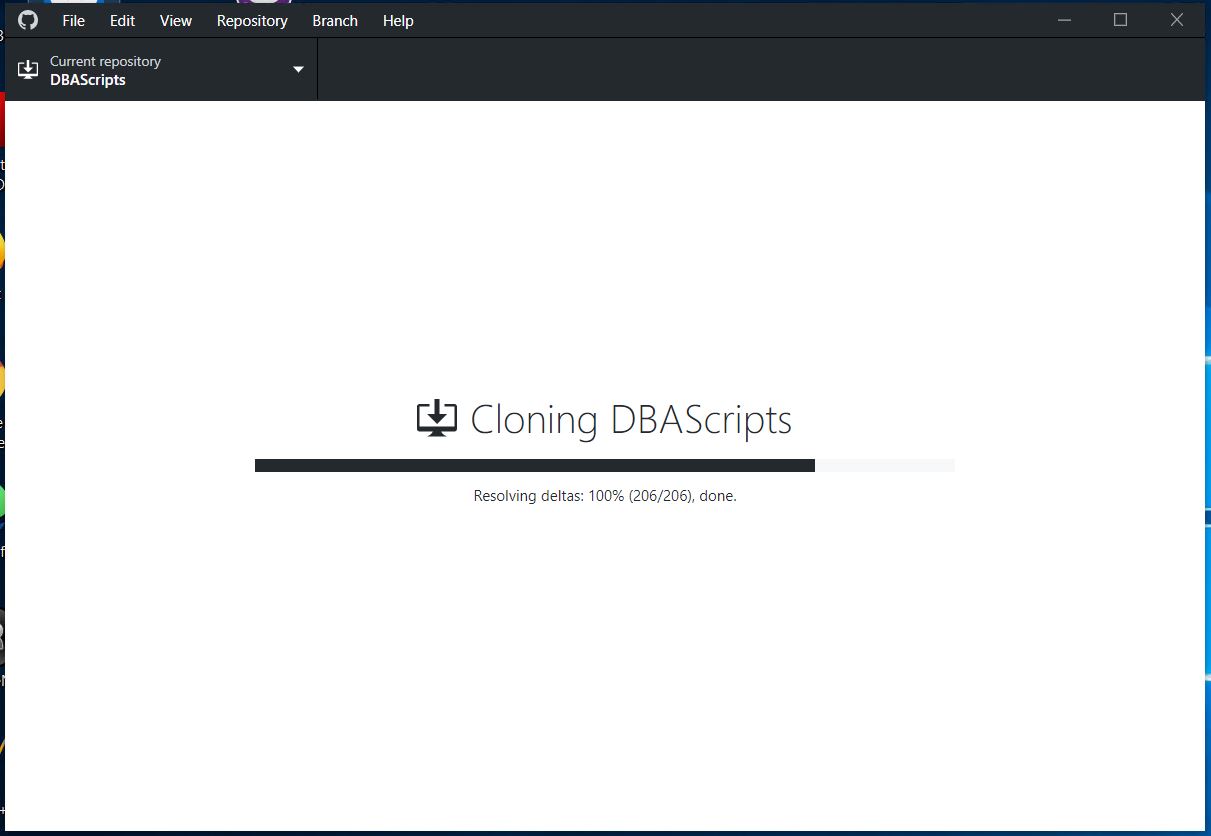
Part of the setup is to clone a repository that currently exists at GLFHC. There are several ways to accomplish this task. One method to achieve that goal is shown here.

Go to the “**File”** option on the top taskbar and select “**Clone Repository**”. Note that a quick shortcut is to press the CTRL + Shift + O buttons simultaneously.

Once you start the closing process, a screen like the following one will appear:

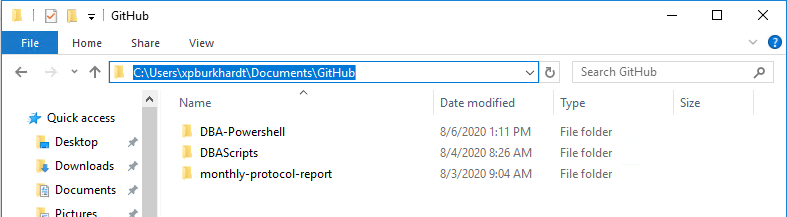


When this operation finishes successfully, the following screen will appear:



Step 4 – Verify Local Cloning:

When the cloning process has completed, open up Windows Explorer and navigate to the C:\Users\<Username>\Documents\GitHub directory and look for the Repository name. In this example, look for the DBAScripts folder.



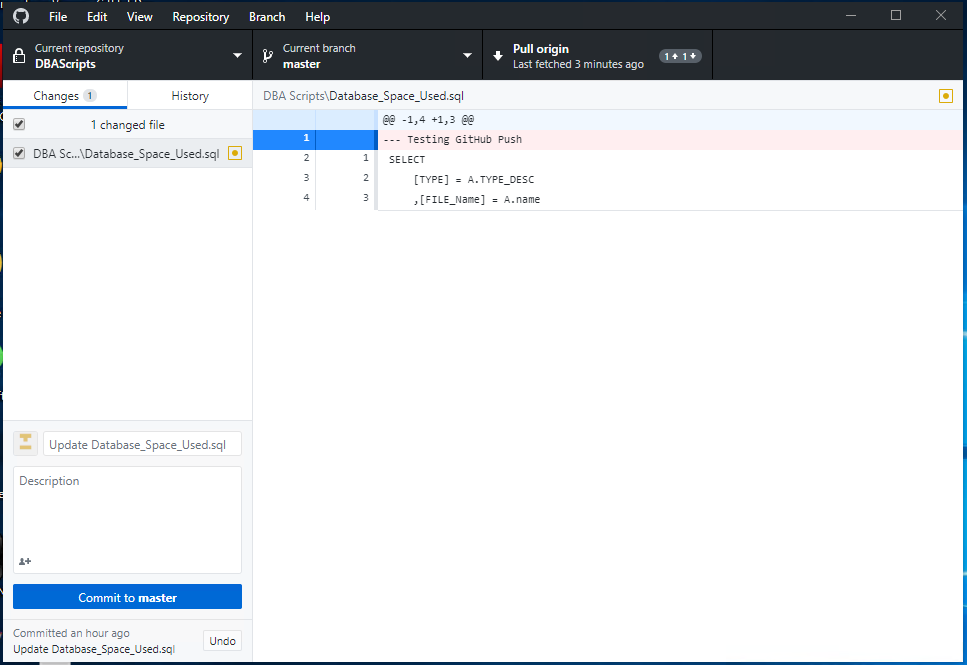
Step 5 – Upload (Push) Local Changes:

For any changes to be uploaded to the GitHub Repository, the modified code needs to be placed in the local repository. This code can either be changed:

* Directly in that folder or edited in another folder.
* Edited outside this folder and then placed back in to the folder when the changes have been tested and approved.

As an example, let make a change to the Database\_Space\_Used.sql file and then save them in that folder.

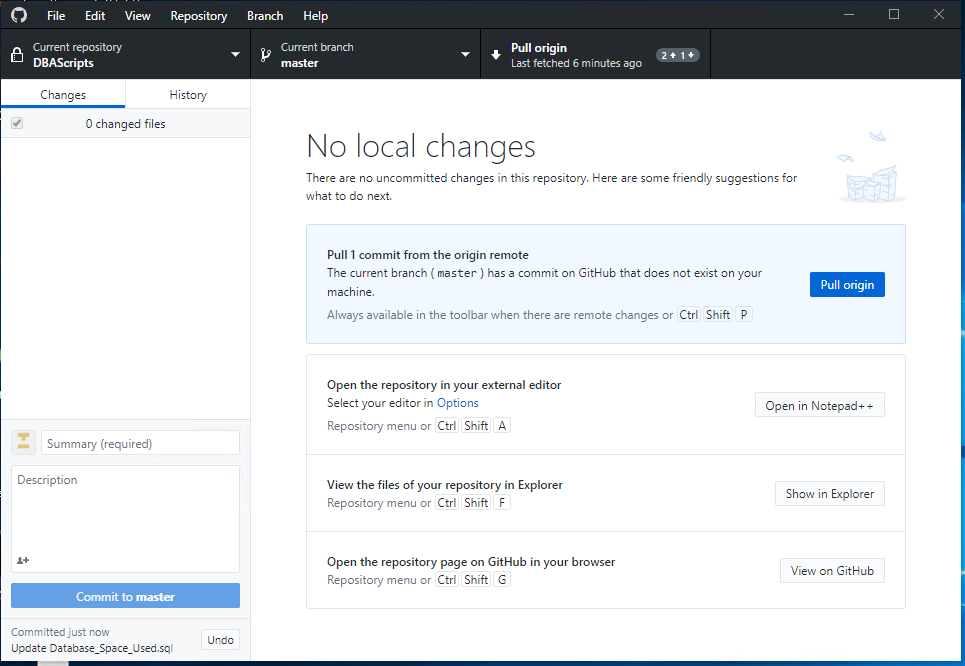
Switch over to the GitHub Desktop and you will see that changes are pending:



Not only do you see that a change exists, you can also see what’s changed in the right-hand window. Review these changes and make sure that they are correct.

The next step is to push these changes up to the GitHub Repository by clicking on the “Commit to master” bar located in the lower left corner of the above screen.

Once the commit process has finished, this screen will appear:



Since software projects may include multiple scripts that are created by different developers, it’s possible that changes are committed to the master GitHub repository that a developer is unaware of. However, if this occurs, the differences between the local and master repository will be displayed as shown in Step 5. The developer can then discuss their changes with the one that created the most recent version of that code and decide whether to commit or rollback their changes. Thus, as safeguard exists for this situation.