|  |  |
| --- | --- |
| GitHub Command Data List RCL-I-CMQA1Team: Configuration Management and Quality Assurance 8/28/2013 -- Revision: - |  |

Revision History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rev | Date | Description | Author | Approved | Pages |
| - | 8/28/2013 | Initial Release | Tom Moline | Tyler Olson | All |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# Primary Contacts:

Tom Moline tmoline@slu.edu 630-401-0791

Tyler Olson tolson6@slu.edu 812-204-1098

# Background

This document contains information on interfacing with GitHub version control software, as well as the most common commands utilized in the making, saving, editing, and sharing of information over said version control software system.

# Getting Started with GitHub

## What is GitHub?

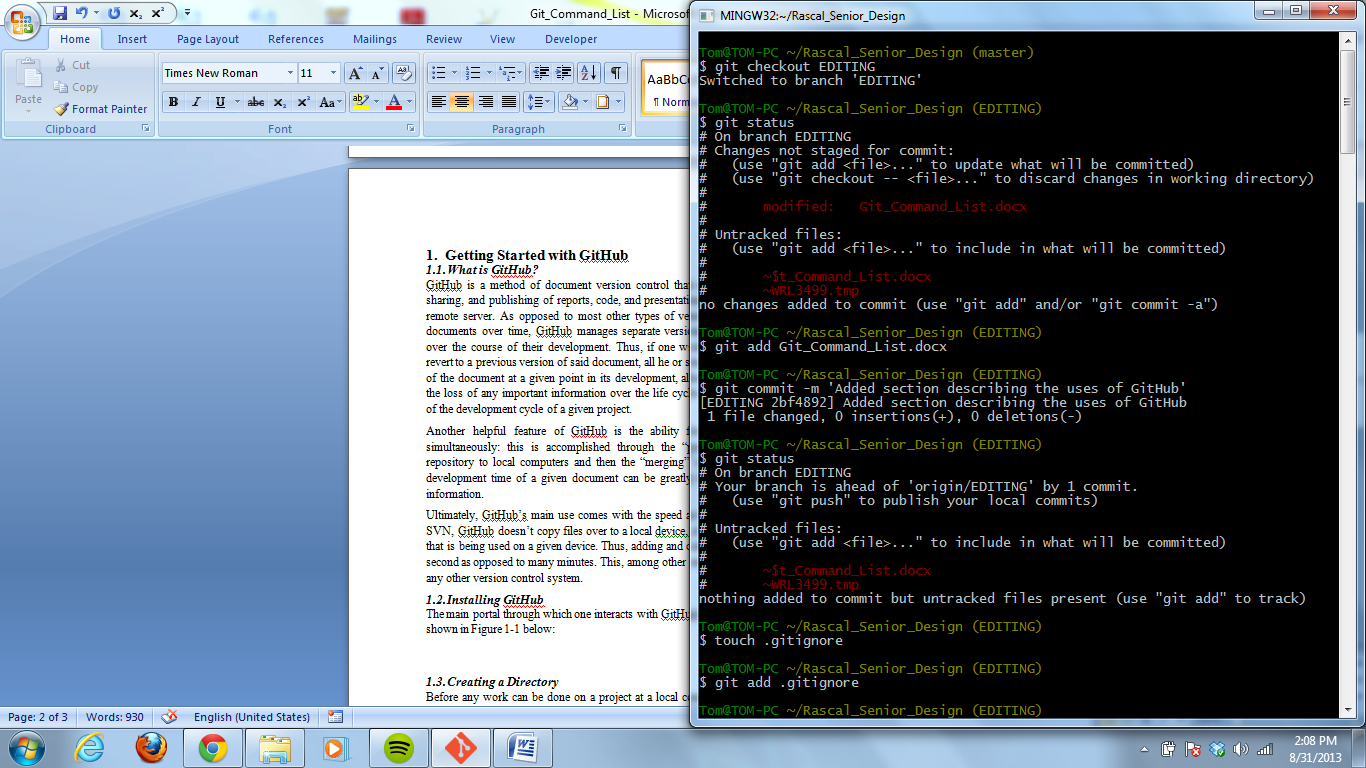
GitHub is a method of document version control that allows for the quick and easy creation, editing, sharing, and publishing of reports, code, and presentations through the use of a common hub located on a remote server. As opposed to most other types of version control software that track changes made to documents over time, GitHub manages separate versions of documents through the use of “snapshots” over the course of their development. Thus, if one wishes to discard changes made to a document and revert to a previous version of said document, all he or she has to do is refer back to a previous “snapshot” of the document at a given point in its development, allowing for new changes to be made. This prevents the loss of any important information over the life cycle of a given document, saving time and frustration of the development cycle of a given project.

Another helpful feature of GitHub is the ability for multiple people to work on one document simultaneously: this is accomplished through the “pulling” of common information from a remote repository to local computers and then the “merging” of changes to the common repository. Thus, the development time of a given document can be greatly reduced while eliminating the risk of losing any information.

Ultimately, GitHub’s main use comes with the speed at which it accomplishes all of these tasks. Unlike SVN, GitHub doesn’t copy files over to a local device, it simply creates different pointers for each branch that is being used on a given device. Thus, adding and committing a large number of files takes less than a second as opposed to many minutes. This, among other reasons, is the main purpose of using GitHub over any other version control system.

## Installing GitHub

The main portal through which one interacts with GitHub is through the GitBash command prompt, as shown in Figure 1-1 below:



***Figure 1-1. GitBash Command Prompt***

This command prompt can be downloaded at <http://git-scm.com/downloads>. Once at the site, simply click on the **Download for Windows** tab and follow the instructions of the Git prompt to download the command prompt.

Once GitBash has been downloaded, it can be accessed through the Windows button under **All Programs** and then in the **Git** folder. Once GitBash is open, it is necessary to initialize a repository through which work can be conducted, as discussed in the following sections.

## Creating a Directory

Before any work can be done on a project at a local computer, a directory for the project must be make. The command structure for accomplishing this is as follows:

Mkdir ~/Rascal\_Senior\_Design

This creates a directory in the user folder of your C:/ drive on your local computer. Before any files can be created or pushed to this directory, it is necessary to set up Git so that it properly refers to it. This should only be done the first time that one is installing GitHub on his or her local device.

## Setting a Directory

If one wishes to work within a particular directory on his or her computer, he or she must input the following command:

cd ~/Rascal\_Senior\_Design

If this command has been successfully executed, the command prompt should read:

~/Rascal\_Senior\_Design (master)

## Initializing Git Files

Now it is necessary to initialize the files git requires in order to be properly run within the established directory. This is accomplished by executing the following command:

git init

If this command has been successfully executed, the following prompt should be displayed:

Initialized empty Git repository in c:/Users/tmoline/Rascal\_Senior\_deign/.git/

To ensure that the files were initialized in the correct directory, enter the Rascal\_Senior\_Design directory under users *your name* and look for a folder called .git. If no such folder is visible, be sure to make hidden files visible within your directory. This is accomplished by clicking **View** on the windows explorer tool bar, then selecting the **Custimize this folder…** tab, clicking the **General** tab, and then uncheck the **Hidden** box. The hidden .git files should now be visible.

## Creating a Reference to a Remote Repository

A remote repository is a directory on a remote server that contains all files (“commits”) associated with a particular project. If any member of the project development team wishes to edit any such files, it is necessary to “push” them to his or her local computer. Before this can be accomplished, one must make sure that he or she is referring to the correct remote repository. This is accomplished thourh the following command:

Git remote add origin https://github.com/tomamoline/Rascal\_Senior\_Design

The “origin” designation inserted before the repository URL refers to a name that one wants to associate with said URL. In this case, in lieu of typing in the URL for every command that requires it, one only has to type in “origin” or any other name associated with a specific URL repository.

## Pushing Commits from a Remote Repository to a Local Directory

To gain access to files located on a remote repository, it is necessary to transfer them to a local directory. This is accomplished with the following command:

git pull origin master

Master refers to the name of a particular branch in the origin repository. A branch is a part of the repository that has a pointer associated with a specific set of commits. Multiple branches can exist in one repository, referring to different commits of the same file. This allows for multiple people to work on one file simultaneously without interfering with each other or risking the loss of any information that has been created or documented.