# GIT Revisiostn Control

The following notes pertain to GIT and GIThub when doing revision control.

When one or more files has changed

**Using Git Bash**

* To display which files have changed

$ git status provides detail

$ git status –s provides an overview of status

* To examine how a file has changed

$ git diff *path\_of\_file*

* To stage a changed file (or files)

$ git add *path\_of\_file* (‘\*’ denoting wild-card is permitted)

* To stage all of the changed files

$ git add –u

Git doesn’t provide any indication that the files have been staged, but you can use “$ git status” or “$ git status –s” to see the change of status.

* To examine changes for all of the staged files

$ git diff - -staged

$ git commit [*options and parameters*] prompts you for a description of the changes that are being committed. It does this by popping up a window in the editor program that you specified during the personalization of the git install process.

* To commit all staged files

$ git commit

* To **add** and **commit** all changed files (but ignore new files)

$ git commit -a

* To display history use a variation of $ git log . . .

$ git log provides a low-detail overview

$ git log –p provides details

$ git log –p -2 provides details for the most recent 2 **commit**s

$ git log –p –since=2.days provides details for **commit**s that occurred in last 2 days

(you could also say “3.days” or “2.weeks”)

$ git log –p –since=2020-08-05 provides details for **commit**s that occurred after 8/2/2020

* To revise the description that was just supplied for a **Commit**

$ git commit –amend

Git pops up the editor program which contains your descriptive text; modify it, save it, and close the editor.

* To revise a **commit**, because you forgot to modify or stage another file (or maybe more than one file), make the file modifications (if not already made), stage the files, and then

$ git commit -amend

Git pops up the editor program which contains your descriptive text; modify it, save it, and close the editor.

* To undo file editing, and to restore the file to the contents that it had as of the last **commit**

$ git restore *path\_of\_file*

*path\_of\_file* may contain the wildcard designator (\*) to indicate more than one file

**Using Git GUI**

* Start GitGui; git responds with a dialog titled Git GUI.
* You have 2 options to select the repository: Click **Open Existing Repository** to use a folder browser, or click the name of the repository if it is listed under **Open Recent Repository:** (The pull-down menu **Repository** – on the menu bar – provides the same 2 options in a different format.)
* The upper-left corner of the dialog - under **Unstaged Changes** - shows the names of the files that have been changed. It also shows files that have not yet been added (unless these files are listed in .gitignore). (I have noticed that Git GUI is sometimes not aware of not-yet-added files in a subfolder; this misbehavior stops when the subfolder becomes known to git, e.g. as a consequence of adding at least one file in that subfolder via git Bash.)
* Click the “name” part of one of the files under **Unstaged Changes**; Git GUI responds by displaying a brief indication of the changes in the upper-right panel and it displays **Untracked, not staged** in the title of this panel.
* To stage a changed file, click the “file icon” part of the file; Git GUI responds by moving the file from the list under **Unstaged Changes** to the list under **Staged Changes (Will Commit)**.
* Click the “name” part of one of the files under **Staged Changes (Will Commit)**; Git GUI responds by displaying a brief indication of the changes in the upper-right panel and it displays **Staged for Commit** in the title of this panel.
* You can change the status of a file from “staged” to “unstaged” by clicking the “file icon” part of the file; Git GUI responds by moving the file from the list under **Staged Changes (Will Commit)** back to the list under **Unstaged Changes**.
* To commit the staged files, (1) type a description of the changes into the text box in the lower-left corner – under **Commit Message**, and (2) click the **Commit** command button.
* To revise the description that was just supplied for a **Commit**, click the **Amend Last Command** check box at the right edge of the dialog, revise the descriptive text, and click the **Commit** command button again.