git Description

Instructions: assume gitbash or equivalent is installed locally (<https://git-scm.com/>) (<https://desktop.github.com/>). You do not need a github account to work with git, but if you want to share files with other you will need to have server capability somewhere. To handle this we will assume you have a github account: If not, create one. Local commands are all describe using the git command line interface which is very easy to use.

**Basic git:** Git creates snapshots of files in a folder when commanded and then saves that snapshot with information about why that snapshot was taken. Snapshot means that the exact state of the files at an instant in time made, not just a list of changes from the previous version. git never overwrites a saved snapshot. You do not have to take a snapshot of everything at once, but can selectively snapshot specific items. In general, Git retains earlier snapshots when new snapshots are made. It does not delete or overwrite the old snapshot, and does not record a series of edits that could be undone. Rather it records the actual current state of the file at the time a snapshot is made. Even if you delete a file in the future, it likely still exists in an earlier git shapshot, because git does not delete anything in the archive. It just adds to it. You can synchronize your snapshots with a repository, like github, or just retain them locally for your own use. Note however, that unless repositories exist in more than one location, you can lose everything if the computer crashes or you accidentally delete the repository locally. But, if you use a repository, either the repository or your local version of the repository has a full set of information whenever you synchronize the two. Finally, git works with any type of file not just programs. So, you could have a git repository for your data files or paper drafts, or anything else.

## Step 1: To begin using git locally: (>cmd, >>response, <optional info>

A) configure git for use on local machine using git cmd line

>git config --global user.name “my name”

>git config --global user.email “my email account”

>git config --global color.ui auto

To check info out about your current status:

>git config --list # print out of the current set of git configuration settings

>git config --get <user.name> # <x> is the name for the thing you want a report on

***Example:***

>git config --get user.name

>>Paul Pfaffinger

## Step 2: Decide how you want to use git. Either as a local version control system or interacting with a central repository that you can share with other.

A) To create a new local GIT repository where there is no active repository on github (or another server):

>git init [project-name] ***# use this if there is no repository for this project already existing on github or if you just want to use git to manage the project locally.*** Most useful for converting an existing folder to using git version control.

B) To Create a new local GIT folder **for** a repository that is existing on github:

Go to a folder where you want to collect git folders on your local machine. Then:

>git clone <url\_to\_repository> # <x> is the https for the repository copied from github

This creates the local repository folder. Move to this folder to begin working. It should now be populated with the github contents for the repository you wanted based on the url. The url does not have to point to github, but could point to any git server. You can add files to this folder or begin to work on the existing files.

## Step 3: To save your work as a snapshot with version information:

>git status # checks to see if something needs saving

***To register with git that you have made important changes to the repository that you will want to record at some point. This process is called staging. Files that have been staged are the ones that will be added to the git repository when you commit.***

>git add <filename> # -A means all otherwise provide an exact file name

>git add -A

Then, you provide an explanation of what those changes are and put them in the permanent local repository. At this point they are registered as a new version of the file and the string says what is different about this version:

>git commit -m "String saying what is new in these files"

If you are working with a git server, then you need to submit the changes back to the server repository to make them permanent on the server. Typically this will be back to the github repository from whence they came, known as their origin. If the local repository is a replica of the main level (master) of the repository, then include that info also:

>git push <origin master> # origin master incorporates the changes into the top level master of the repository

## Step 4: To work on the repository from another computer,

Safest approach is to destroy any current folder and create a fresh version of the repository using the git clone <repository url> approach. Otherwise you will likely have to merge the exisiting folder contents with the repository contents.

To refresh your current local repository to be up to date with the github repository, by using the git pull command from the current repository rather than git clone.

>git pull # produces an up to date file set to work wiht.

***And work on it just as before…***

>git status

>git add -A

>git commit -m "My commitment message"

>git push

## Step 5: To go back and work on the repository replacing what is present use pull:

>git pull #brings down the latest version of the repository. It does not remove what was present in the current folder but doesn’t merge things together either. Again, easiest to do this to a clean folder or to use pull to identify things that are different from the repository, in which case you can work on merging the repository with what you have in your folder.

## Step 6: To create a link to a repository from an existing folder:

git remote -v # gives the existing links

git remote add origin <URL to Repository> # sets the origin of this folder to match the repository url. Note this approach can also be used to link to branches and forks by naming the brain or fork name rather than origin.

## Step 7: To work on a branch called upstream:

get remote add upstream <URL to upstream of Repository> # links folder to the branch rather than main

git fetch upstream # pulls in contents of the github branch into the local folder

git merge upstream /master # merges your changes with the master on your local folder

git push origin master # pushes these changes from the branch onto the master

## Step 8: To look at the history of changes in a repository use log:

>git log <-n> # -n flag is a number that says only gives the last “n” changes

## Step 9: To tag specific repository saves as having special importance (Like a major change in a program):

>git tag –a <tagname> -m “message about the tag” # creates a reference to a specific commit.

## Step 10: Work on a specific tagged verison: Checkout the tag:

>git checkout –b <branchname> <tagname> # creates a working relationship with a specific tag