A good [Git cheatsheet](https://towardsdatascience.com/a-git-cheatsheet-that-all-coders-need-bf8ad4d91576) at <https://towardsdatascience.com/a-git-cheatsheet-that-all-coders-need-bf8ad4d91576> with more commands than we need now but a very good reference.  
<https://www.bitdegree.org/learn/git> -- good stuff but with lots of fluff.

# Install Git

<https://youtu.be/Rhc0KzfLaBk> This tutorial will explain how to install the git version control system on your computer for the Final Project.

Git is a free and open-source distributed version control system. Get Git from [git-scm.com/](http://git-scm.com/). The Download button for the latest source release automatically selects Windows or [macOS](http://git-scm.com/download/mac). Take the defaults during installation with the possible exception of the default editor.

|  |  |
| --- | --- |
| To launch git:  Windows-Key, “git”, then select Git Bash.  This is an emulated Linux / Unix (\*nix) environment.  Git GUI is available but making it work takes more effort than using the Bash shell. |  |
| Running Git Bash results in a \*nix emulated terminal window.  Enter git to verify the installation.  In your project folder containing source files, create a Shortcut to save time.  Target: "C:\Program Files\Git\git-bash.exe"  Start in *example*: "%USERPROFILE%\Documents\Seneca\CP4P Final Project" |  |

In the Bash shell, do not use the back slash \ as a folder separator in a path; use the forward slash / as if in \*nix. Bash interprets \ as the escape char for special characters, e.g. \$. Avoid special characters in folder and file names – it will make life at the command line easier.

At the $ prompt in the Bash git shell

* Git commands are preceded with "git "
* \*nix commands like cd or ls can be entered normally
* use the Insert key to paste from clipboard instead of Ctrl-V ([see this](https://danlimerick.wordpress.com/2011/07/23/git-for-windows-tip-how-to-copy-and-paste-into-bash/))
* select text by click + drag with mouse
* copy selected text with right click or Enter key.
* Up or down arrow keys will recall commands from the stack

## Bash shell examples

$ cd "Documents/Seneca/CPR101 Final Project"

$ ls # list all files

## Git setup

The global git username and password are associated with commits on all repositories on your system.

$ git **config** **--global** **user.name** "*Your Name*"

$ git **config --global user.email** "*UserID*@mySeneca.ca"

$ git **config --list #** Confirm the setup. Type **q** to quit the list, **h** for help.

$ cd "*path to dir/folder where repository will be*"

git **init** # Create an empty Git repository in the current folder/directory

If you see Documents/*path*/.**git:No such file or directory**  
allow git.exe to write to your drive in your anti-virus or malware protection software

The response should be   
 Initialized empty Git repository in Documents/*path*/.git/

* To reset git and start again, delete the *hidden* **.git** folder

## Essential Git Commands

$ git **add** *file\_name*[type first character(s) of filename and press TAB key for auto complete]

N.B. filenames should not include any version indication. Git merges and tracks the code differences within the *same* filename across committed versions. Different filenames are unrelated to each other. Version control happens only when the *same* filename is modified.

$ git **commit -m** *version\_name*# -m is message switch: use a unique description for each commit.

$ git **status -v** # files with changes, yet to be committed   
 -v switch also shows source files' content differences:   
 lines + added, - deleted, -/+ changed. e.g.

diff --git a/converting.c b/converting.c  
index b493251..c815139 100644  
--- a/converting.c  
+++ b/converting.c  
@@ -1,4 +1,5 @@  
-// CONVERTING V2  
+// CONVERTING V2, changed this line to test git  
+// added this line to test git  
 #include "converting.h" !! there is no -/+ flag, line listed for context and location of -/+ changes within the source file.

$ git **log** # displays summary of commits (versions)

$ git **log -p** # displays commit differences (versions). Page-Down, Page-Up, **q** to quit the screen-by-screen listing, **h** for help.

$ git **--no-pager log -p > "complete\_git\_log.txt"** # outputs commit differences to all files in the repo

$ git **--no-pager log -p *module\** > "*module*\_git\_log.txt"** # outputs commit differences to *module* files only

$ **exit** #To finish your git session

## To process the next version of source files after the previous commit

* Make the changes to the same source filename. Comment, compile, write test cases, record test results.
* $ git **add** *file\_name*# adds the latest changes made to a source code file into the git repo.
* $ git **commit -m** *next\_version\_name*# e.g. if previous version was "V1", this will be "V2"
* $ git **status -v** #

# Additional Git Commands

$ git **ls-tree -r master** # lists files tracked in current branch named 'master'

$ git rm --cached file\_name   
 # removes a file from git repo, but not from the filesystem (source file remains)

$ git commit -m "removingfile\_name from repo only"# to commit the removal

See <https://www.git-tower.com/learn/git/ebook/en/command-line/advanced-topics/diffs/>   
<https://intellipaat.com/community/12299/how-to-exit-git-log-or-git-diff>

# File States

Diagram, timeline

Description automatically generated