Version Control

**Version control** is a system that records changes to a file or set of files over time so that you can recall specific **versions** later.

GIT

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git course location

https://www.udacity.com//course/viewer#!/c-ud775/l-2980038599/e-2960778952/m-2960778953

GIT commands

To use these commands first navigate to the git repository

git log - lists out every commit that has ever been made starting with the most recent. Each commit will have an ID, an author, and a message associated with it. The message explains what changes have been added since the last commit. ID is sort of like a serial number that uniquely identifies each commit and helps you refer to it.



**Cloning and Exploring The Repo**

**Cloning an existing Repository**

**(command for cloning or downloading an entire repository.)**

To clone a repository, run git clone followed by a space and the repository URL.

**Asteroids URL**

Use the following url to clone the Asteroids repository: https://github.com/udacity/asteroids.git

**Exiting git log**

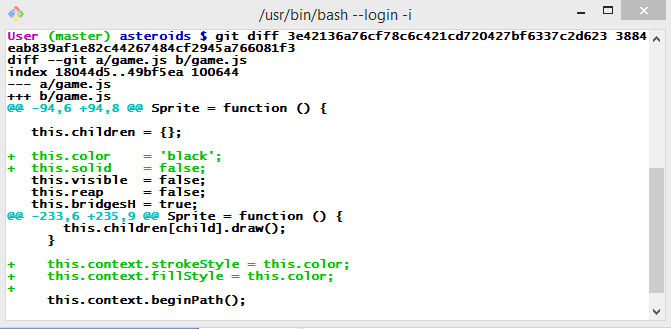
To stop viewing git log output, press q (which stands for quit).

**Getting Colored Output**

To get colored diff output, run git config --global color.ui auto

git diff - Used to check changes that a commit introduced. We type in the two commit ids to be compared after the git diff command.

git diff 3e42136a76cf78c6c421cd720427bf6337c2d623 3884eab839af1e82c44267484cf2945a766081f3



shows differences in game.js file in the two versions additions are denoted in green.

### Using git log and git diff

As a reminder, running git log will show a list of the recent commits with information about them, including commit IDs.  Running git diff followed by two commit IDs will compare the two versions of the code in those commits.

It is advisable not to commit changes in different files in one commit.

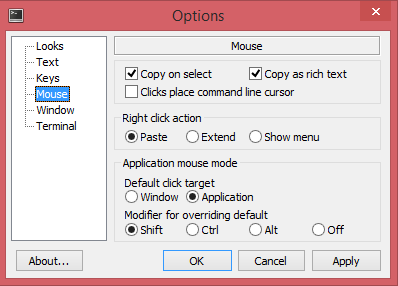
git checkout

Used to checkout older commit (in case current code has some unfindable error and one of the previous commits work correctly)

now if u use git log, the checked out commit will be shown as the most recent commit and the commits after that wont be shown at all.

If you know the most recent commit id before checkout of an older version then u can get back to it using git checkout that command id which will display the actual git log.

# Copying and Pasting in Git Bash



options -> mouse -> Right click action ----> paste

## Setting Up Your Workspace on Windows

### Changing background color

If you prefer the background color of Git Bash to be something other than black, you can change it in the "Defaults" menu under the "Colors" tab.  If you like the background color as-is, you don't need to make any changes.

### Downloading necessary files

* Save [this file](https://raw.githubusercontent.com/git/git/master/contrib/completion/git-completion.bash) in your home directory with the name git-completion.bash.
* Save [this file](https://raw.githubusercontent.com/git/git/master/contrib/completion/git-prompt.sh) in your home directory with the name git-prompt.sh.
* Save bash\_profile\_course from the Downloadables section in your home directory with the name .bash\_profile.  (If you're curious to learn more about how bash prompts work, see [this page](http://www.cyberciti.biz/tips/howto-linux-unix-bash-shell-setup-prompt.html).)

### Making Git configurations

Run the following Git configuration commands.  The first one will need to be modified if you are using a text editor other than Sublime, or if Sublime is installed in another location for you.  See [this page](https://help.github.com/articles/associating-text-editors-with-git/) for the correct command for a couple of other popular text editors.  For any other editor, you'll need to enter the command you use to launch that editor from Git Bash.

git config --global core.editor "'C:/Program Files/Sublime Text 2/sublime\_text.exe' -n -w"

git config --global push.default upstream

git config --global merge.conflictstyle diff3

### Make sure you can start your editor from Git Bash

If you use Sublime, you can do this by adding the following line to your .bash\_profile:

alias subl="C:/Program\ Files/Sublime\ Text\ 2/sublime\_text.exe"

### Restart Git Bash

You'll need to close and re-open Git Bash before all your changes take effect.

## Morsel 30: Setting Up Your Workspace on Mac

### Downloading necessary files

* Save [this file](https://raw.githubusercontent.com/git/git/master/contrib/completion/git-completion.bash) in your home directory with the name git-completion.bash.
* Save [this file](https://raw.githubusercontent.com/git/git/master/contrib/completion/git-prompt.sh) in your home directory with the name git-prompt.sh.
* Save bash\_profile\_course from the Downloadables section in your home directory with the name .bash\_profile.  If you use Linux, you may need to name this file .bashrc instead of .bash\_profile. (If you're curious to learn more about how bash prompts work, see [this page](http://www.cyberciti.biz/tips/howto-linux-unix-bash-shell-setup-prompt.html).)

### Make sure you can start your editor from the terminal

If you use Sublime, you can do this by add the following line to your .bash\_profile:

alias subl="/Applications/Sublime\ Text\ 2.app/Contents/SharedSupport/bin/subl"

### Making Git configurations

Run the following Git configuration commands.  The first one will need to be modified if you are using a text editor other than Sublime, or if Sublime is installed in another location for you.  See [this page](https://help.github.com/articles/associating-text-editors-with-git/) for the correct command for a couple of other popular text editors.  For any other editor, you'll need to enter the command you use to launch that editor from the terminal.

git config --global core.editor "subl -n -w"

git config --global push.default upstream

git config --global merge.conflictstyle diff3

### Restart the terminal

You'll need to close and re-open the terminal before all your changes take effect.

## Getting help on Git commands

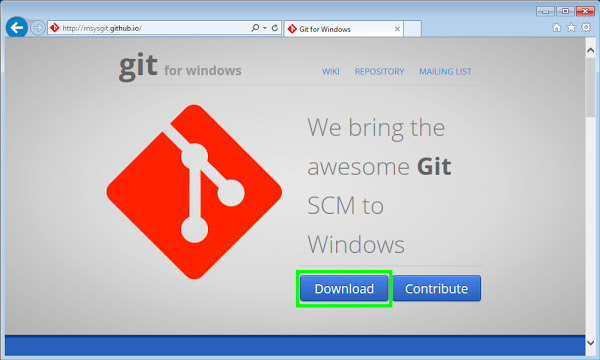
To get information about how to use a Git command, run git help <command>.  For example, to get help on the command git diff, the first Git command you'll learn in the course, run git help diff.  The help pages will be in a similar format to [Unix man pages](https://www.udacity.com/wiki/ud775/command-line-instructions#learn-about-man-pages).

# Git Installation on Windows

## Download Git

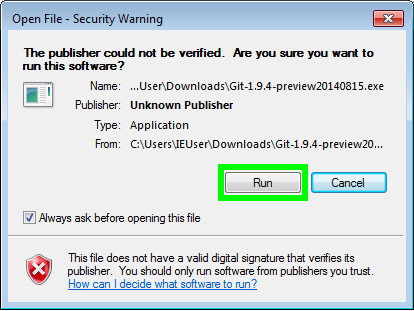
Visit [this page](http://msysgit.github.io/) and click Download.

URL for notes: https://www.udacity.com/wiki/ud775/install-git/install-git-windows

[](https://www.udacity.com/wiki/w/image/view?key=agpzfnVkYWNpdHl1chYLEglXaWtpSW1hZ2UYgIDgmN64-AsM)

## Run the Downloaded File

If you are prompted to either save the file or run it, select "Run".  Otherwise, after downloading the file, double click on it to open it, and select "Run".

[](https://www.udacity.com/wiki/w/image/view?key=agpzfnVkYWNpdHl1chYLEglXaWtpSW1hZ2UYgIDg-NWh8QgM)

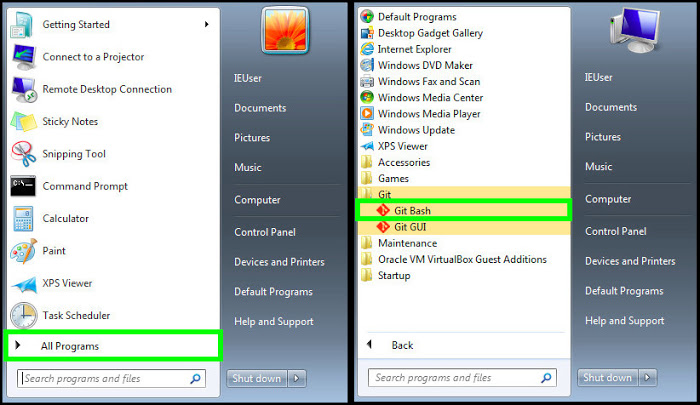
## Complete the Install Wizard

Continue through the install wizard, selecting the default options.

[](https://www.udacity.com/wiki/w/image/view?key=agpzfnVkYWNpdHl1chYLEglXaWtpSW1hZ2UYgIDgmObMtwgM)

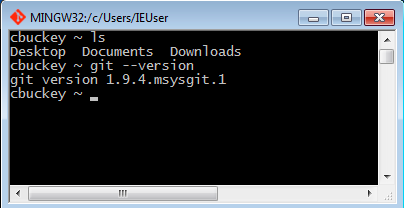
## Open Git Bash

Open the Git Bash program, which can be accessed from the Start Menu via All Programs > Git > Git Bash.  If you are using Windows 8, search for Git Bash in the app list.

[](https://www.udacity.com/wiki/w/image/view?key=agpzfnVkYWNpdHl1chYLEglXaWtpSW1hZ2UYgIDgxIvezggM)

## Run Git Commands

You can now use Git commands, as well as standard Unix commands, within Git Bash.  For example, try running git --version or ls.  To copy and paste in Git Bash, follow the instructions [here](https://www.udacity.com/wiki/ud775/git-bash-copy-paste).

[](https://www.udacity.com/wiki/w/image/view?key=agpzfnVkYWNpdHl1chYLEglXaWtpSW1hZ2UYgIDgxL3CiQkM)

# Git Installation on Linux

## Install Git

On Linux, open a terminal and install Git via your package manager.

For example, if you are using Ubuntu or another Debian-based distribution, run sudo apt-get install git.  If you're using Fedora, run sudo yum install git.

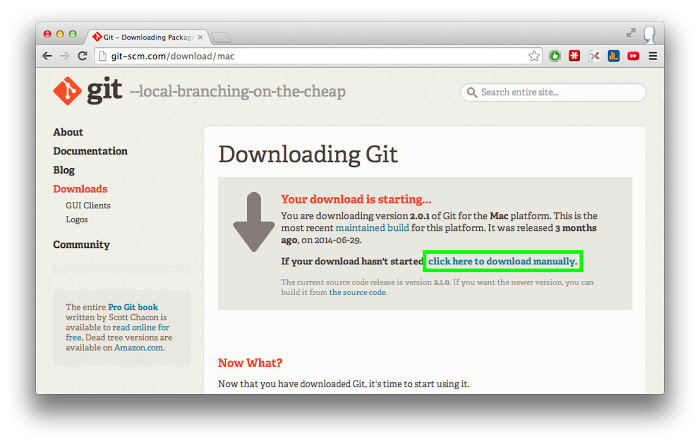
## Run Git Commands

Now open a new terminal, and you will be able to run Git commands.  For example, try running git --version.

# Git Installation on Mac

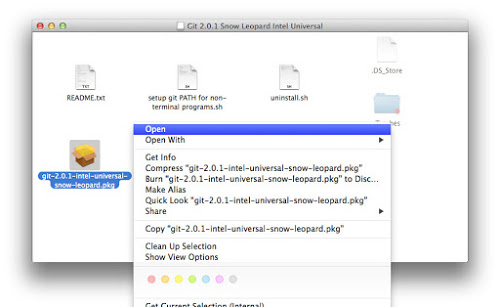
## Download Git

Visit [this page](http://git-scm.com/download/mac) and your download should start automatically.  If it doesn't, click the indicated link to download manually.

[](https://www.udacity.com/wiki/w/image/view?key=agpzfnVkYWNpdHl1chYLEglXaWtpSW1hZ2UYgIDg6MObowsM)

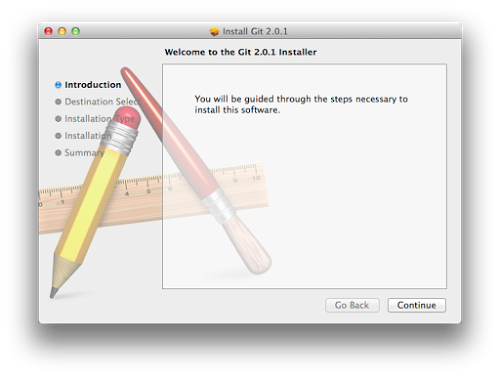
## Run the Downloaded File

Click on the downloaded file to open it in the finder.  Then press Control while clicking on the .pkg file and select open.

[](https://www.udacity.com/wiki/w/image/view?key=agpzfnVkYWNpdHl1chYLEglXaWtpSW1hZ2UYgIDguJyA8gsM)

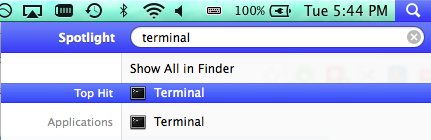
## Complete the Installer

Continue through the installer, selecting the default options.

[](https://www.udacity.com/wiki/w/image/view?key=agpzfnVkYWNpdHl1chYLEglXaWtpSW1hZ2UYgIDgmO6suQoM)

## Open the Terminal

Open the Terminal by searching for it using Spotlight in the top-right corner of the screen.  If you already have a Teminal open, you will need to open a new one to use Git.

[](https://www.udacity.com/wiki/w/image/view?key=agpzfnVkYWNpdHl1chYLEglXaWtpSW1hZ2UYgIDgpJ3S9AsM)

Note that if you are using a Mac OSX version prior to El Capitan there may be additional steps needed to install Git successfully. You can find these steps in the readme file included in the Git download. (Double-click on the README.txt file to open it.)

## Use Git commands

You can now use Git commands within the Terminal.  For example, try running git --version.

[](https://www.udacity.com/wiki/w/image/view?key=agpzfnVkYWNpdHl1chYLEglXaWtpSW1hZ2UYgIDguPaqtQoM)

### Admin privileges errors

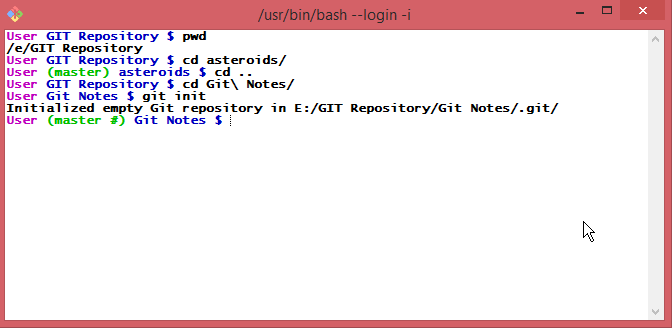
If running the above command gives you the error Agreeing to the Xcode/iOS license requires admin privileges, please re-run as root via sudo, then run sudo git --version, type the password you use to log in to your computer, and agree to the license by following the instructions on screen.  Pro tip: you can skip straight to the bottom of the license by pressing shift+G.  After you've agreed once, you should be able to run git --version without errors in the future.

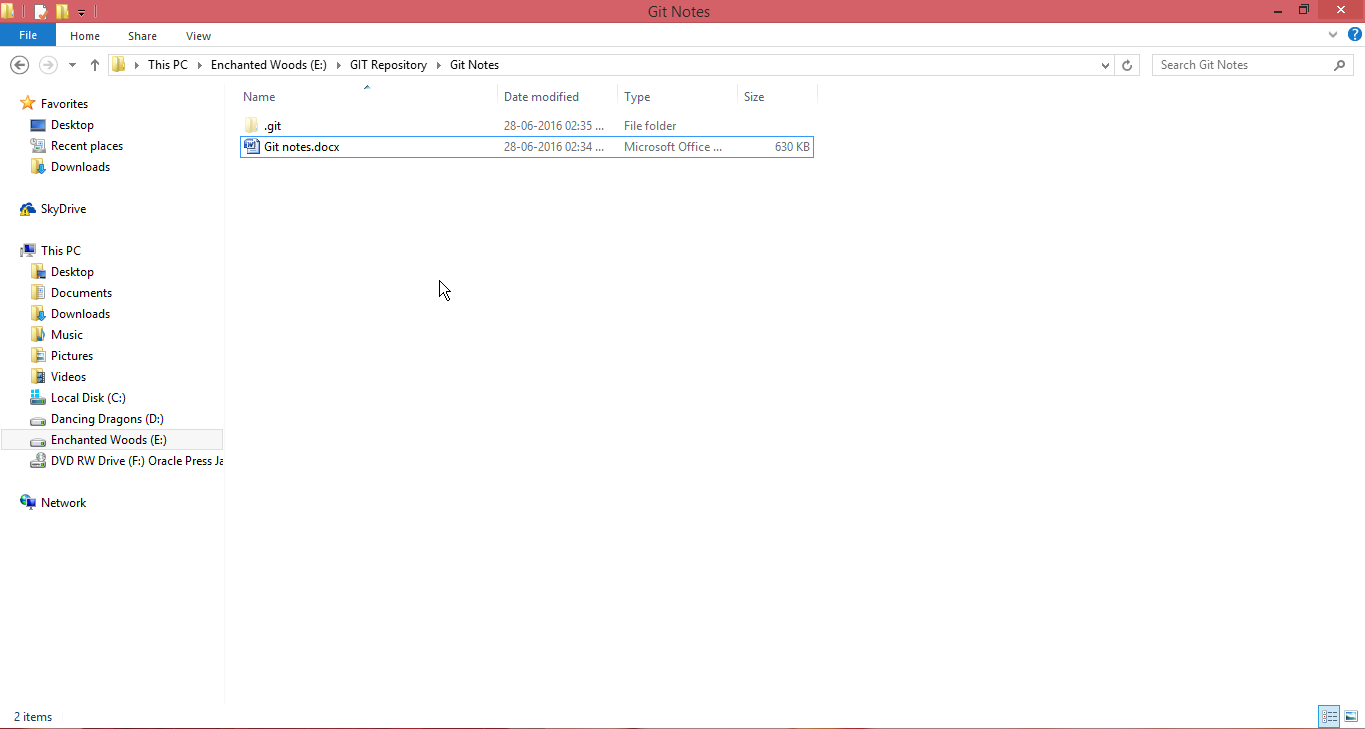
**Initializing a Repository (Creating a GIT repository)**

**Git repositories and directories**

Each Git repository is tied to a specific directory - the directory where you ran git init.  Only files from that directory (and subdirectories inside that directory) will be contained in that repository, and you can have different repositories in different directories.

Note: it's often the case that a Git repository in some directory will only contain, or track, **some** of the files in that directory, rather than **all** of them.  You'll see how this works later this lesson.





**Staging Area**

If you accidentally add a file to the staging area, you can remove it using git reset.  For example, if you accidentally add lesson\_2\_reflections.txt, but don’t want it to be committed yet, run git reset lesson\_2\_reflections.txt and the file will be removed from the staging area, but it will still be in your working directory.

Commiting changes to github

 [Create a new repository](https://help.github.com/articles/creating-a-new-repository) on GitHub. To avoid errors, do not initialize the new repository with README, license, or gitignore files. You can add these files after your project has been pushed to GitHub.

 Open Git Bash.

 Change the current working directory to your local project.

 Initialize the local directory as a Git repository.

git init

 Add the files in your new local repository. This stages them for the first commit.

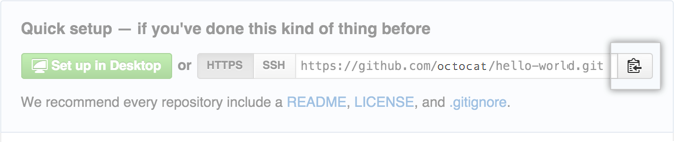
git add .

# Adds the files in the local repository and stages them for commit. To unstage a file, use 'git reset HEAD YOUR-FILE'.

 Commit the files that you've staged in your local repository.

git commit -m "First commit"

# Commits the tracked changes and prepares them to be pushed to a remote repository. To remove this commit and modify the file, use 'git reset --soft HEAD~1' and commit and add the file again.

 At the top of your GitHub repository's Quick Setup page, click

 to copy the remote repository URL.

 In the Command prompt, [add the URL for the remote repository](https://help.github.com/articles/adding-a-remote) where your local repository will be pushed.

git remote add origin remote repository URL

# Sets the new remote

git remote -v

# Verifies the new remote URL

 [Push the changes](https://help.github.com/articles/pushing-to-a-remote) in your local repository to GitHub.

git push origin master

# Pushes the changes in your local repository up to the remote repository you specified as the origin

<https://help.github.com/articles/adding-an-existing-project-to-github-using-the-command-line/>

github url : <https://github.com/HermiGinger/TestWorks.git>

https://github.com/HermiGinger/Practice-Notes.git