**Please Notice, Git and GitHub are file data/text change version control, i.e., monitors any change in data or text within a text file and addition or deletion of files to ensure data integrity.**

**SVN is a Version control, NOT A TEXT CONTROL…it tracks the version of the software or document rather than what has been changed in it.**

**You can press the tab key for auto complete.**

**SSL certificate problem: Unable to get local issuer certificate**

Problem

The following is seen on the command line when pushing or pulling:

SSL Certificate problem: unable to get local issuer

Cause

This error occurs when a self-signed certificate cannot be verified.

Workaround

Tell git to not perform the validation of the certificate using the global option:

**git config --global http.sslVerify false** // use this in Bash to install and then it is okay

Resolution

There are several ways this issue has been resolved previously:

A. Ensure the root cert is added to git.exe's certificate store as discussed [here](http://blogs.msdn.com/b/phkelley/archive/2014/01/20/adding-a-corporate-or-self-signed-certificate-authority-to-git-exe-s-store.aspx).

B. Tell Git where to find the CA bundle by running:

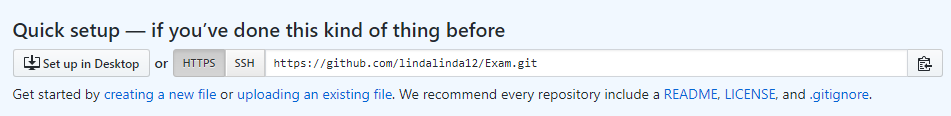
git config --system http.sslCAPath /absolute/path/to/git/certificates

or copying the CA bundle to the /bin directory and adding the following to the gitconfig file:

sslCAinfo = /bin/curl-ca-bundle.crt

C. Reinstalling Git.

D. Ensuring that the complete CA is present, including the root cert.



**…or create a new repository on the command line**

echo "# Exam" >> README.md

git init

git add README.md

git commit -m "first commit"

git remote add origin https://github.com/lindalinda12/Exam.git

git push -u origin master

**…or push an existing repository from the command line**

git remote add origin https://github.com/lindalinda12/Exam.git

git push -u origin master

**…or import code from another repository**

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

**Setting Up global credentials** **to set your account's default identity.:-**

git config --global user.email "lindalinda12@outlook.ie"

git config --global user.passward "Freed@m1234"

git config --global user.id "lindalinda12"

git config --global user.name "Lynda Starr"

then type the following so you can use a credential helper to tell Git to remember your GitHub username and password every time it talks to GitHub.

**git config --global credential.helper wincred**

Omit --global to set the identity only in this repository.

**Using PowerShell & Bash:-**

**To stop infinite loop in bash or cmd shells, then click Ctrl + C**

**If you make changes to the file on the Github server only, you can’t use git push, before integrating your files in the GitHub server and your files locally. So, first type the following command to do so;**

**<git pull>**

**And then you can use <git push>**

**If you're cloning GitHub repositories using HTTPS, you can use a credential helper to tell Git to remember your GitHub username and password every time it talks to GitHub.**

git config --global credential.helper wincred

**How to migrate a local git repository created on desktop to GitHub;**

1. Create a repository on GitHub website with the same name as the local desktop repository, e.g., **Java2**. Do not select Create ReadMe file and do Not select License.
2. Start Git Bash and change the directory to be to the local git repository directory, e.g.,

**cd “D:\Java2”**

1. In the Git Bash type the following equivalent;

**git push --set-upstream https://github.com/lindalinda12/Java2 master**

<git clone “GitHub Repository web address”> to clone a repository from GitHub website to your GitHub / Git desktop application.

Or, we use;

<git init “new Repository Directory Name”> to create a repository on local machine.

Then, add files to the repository in desktop, either manually or with a command line.

<git add file filename> to add a file or, <git add –A> for several files, or <git add folder/> to add a folder **(make sure the folder or file name has no space!)**

<git add **.** > to add all files, i.e., stage all files

use <git add filename> to update, i.e., stage, what will be committed.

Use <git add foldername/\*> to add all files in this folder, even without the star.

use <git checkout -- file> to discard changes in working directory.

use <git reset HEAD file> to unstage

<git commit –m “comment statements”> commits what’s in the staging index area.

<git commit -m “comment statement” filename> to commit a single file only

<git push> to merge your file/branch with Master branch on remote server.

git remote add origin <https://github.com/lindalinda12/Java2.git>

<git status> to get the status of files in the repository (i.e., gets the difference between the Repository, staging index and work

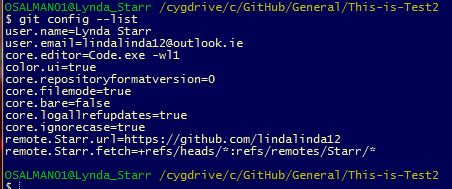
<git pull> to see all changes done recently.

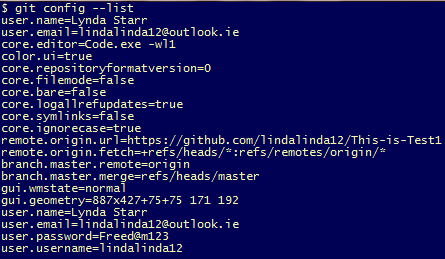
<git> shows you all different git commands.

< (-) + shift + s + return > will cause fold formatting text lines on bash shell.

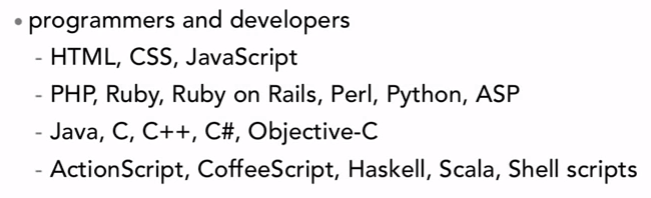
<git diff –color-words filename> to show information in different formatting.

<git commit -m “comment”> will add and commit all files in one go, but it does not work for deleted files and not tracked files, so works well for modifications, but deleted files and new files, it does not work well.

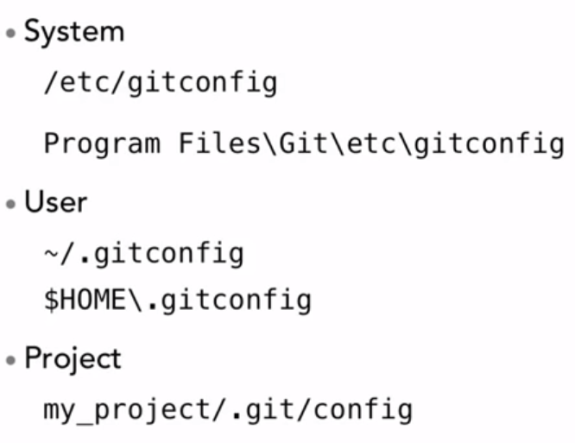


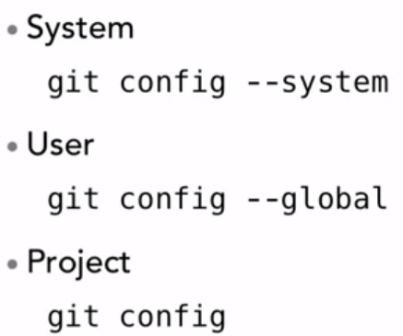






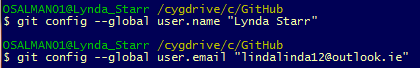
Git shell and bash configuration;





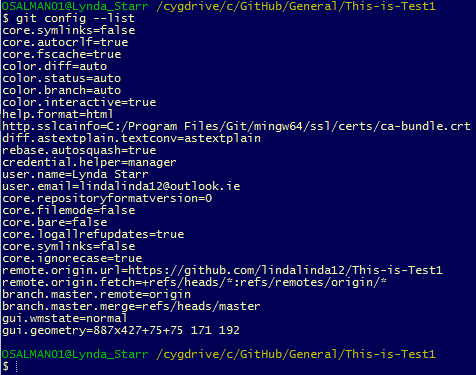
Notice you decide the level by current working directory, say inside a project or in higher directory.

**Notice, Cygwin bash always start in the current user directory, e.g., Lynda\_Starr**



To find the configuration info, type;

<git config --list>



or to find specific configuration;

<git config –global user.name “user name”> to find the name configured

<git config -- global user.email “email”> to find the email configured

<git config --global user.url “url of GitHub account”>

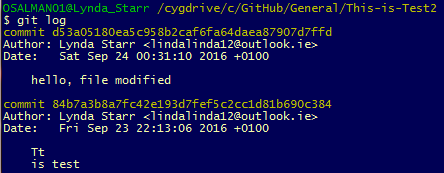
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

<cd ~> to go to your user directory

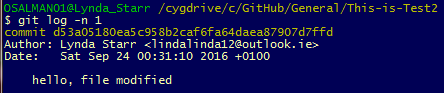
<ls –la> to see directory files and folders.

<ls -la .git> to see files in the git directory.

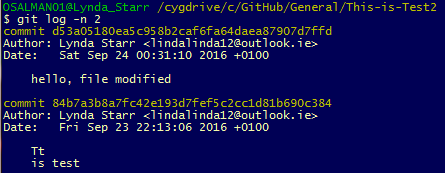
<git log> shows the changes commit messages.



<git log -n 1> Will limit the number of returned log comments to 1



<git log –n 2> will return 2 commit comments.



<git log --since=1012-06-14> returns the commit comments since this date

<git diff> tells us the difference between the old version in the Repository and what’s in the working directory.

<git diff filename> for a certain file only.

<git diff –-staged> will look in the staged files, i.e., added to index, and compares it to what’s in the Repository.

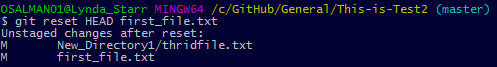
<git diff --cached> will look in the staged files and compares it to what’s in the Repository.

**To Undo changes to a un-committed file and Unstaged files, i.e., not added files by using git add, that exists in the working directory, before recommitting the changed version, you do:-**

<git checkout -- filename> undo changes done in the working directory for the same file name, copying the unmodified version into the working directory instead.

**To Undo changes to the staging index, i.e., to unstage files that were staged (using git add), we do the following:-**

<git reset HEAD filename> will put the head pointing to filename in the working directory.



**To Undo changes done and amend to the Repository, i.e., after committing files to the Repository, we do:-**

<git commit --amend -m “comment” filename>

**To revert a committed file to a previous committed version, using the commit code, we do the following:-**

<git checkout “SHA-1 code” -- fileName>



Notice, file will be moved to the index as staged that needs to be committed.

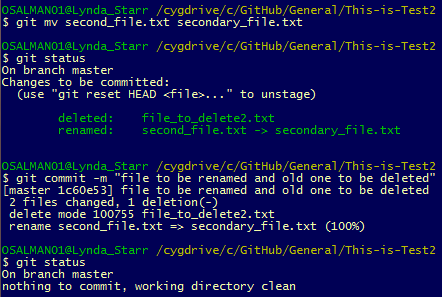
SHA-1 (Secure Hash Algorithm 1) is a cryptographic hash function.

**To delete a file:-**

1. <git rm filename> will add a file in the staged area to be deleted (it completely remove file from working area, without moving to recycle basket.)
2. <git commit –m “comment”>

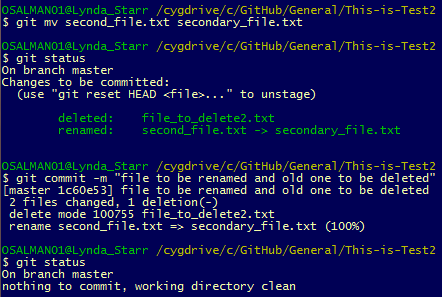
**To rename a file while deleting the old one, we do the following:-**

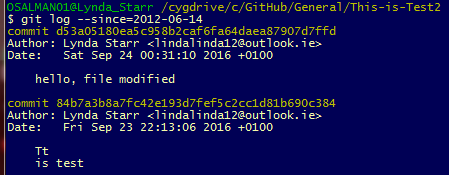
1. <git mv currentFileNmae newFileName> This rename file and move it to the staging area, while the old file will staged to be deleted.
2. <git commit -m “comment”>



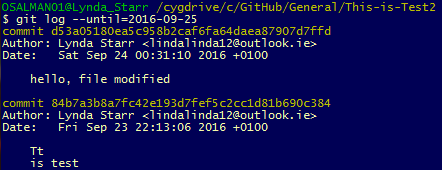
**To move a file:-**

1. < git mv filename newdirectory/filename>
2. <git commit -m “comment”>



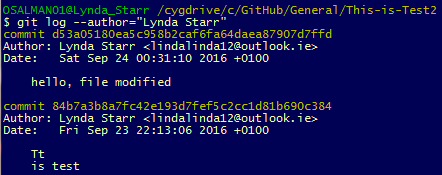


<git log --until=2016-09-25> returns the commit comments until this date

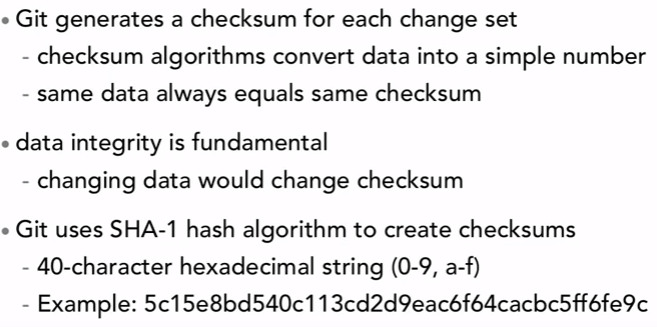


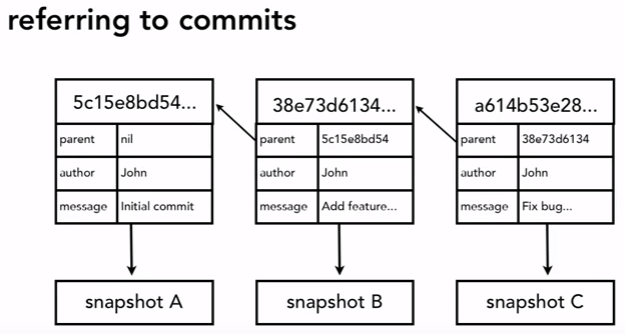
We can use since and until together also.

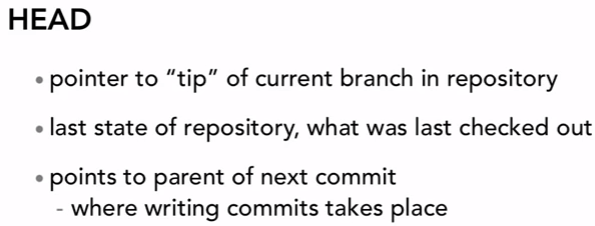
<git log --athor=”Lynda Starr”> will return the comment comments for this author.

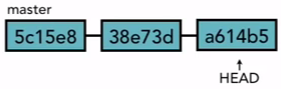


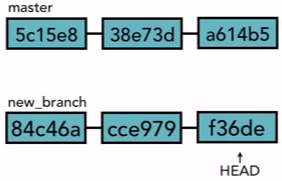
<git log --grep=”hello”> will return all commit comments that contain the word hello.



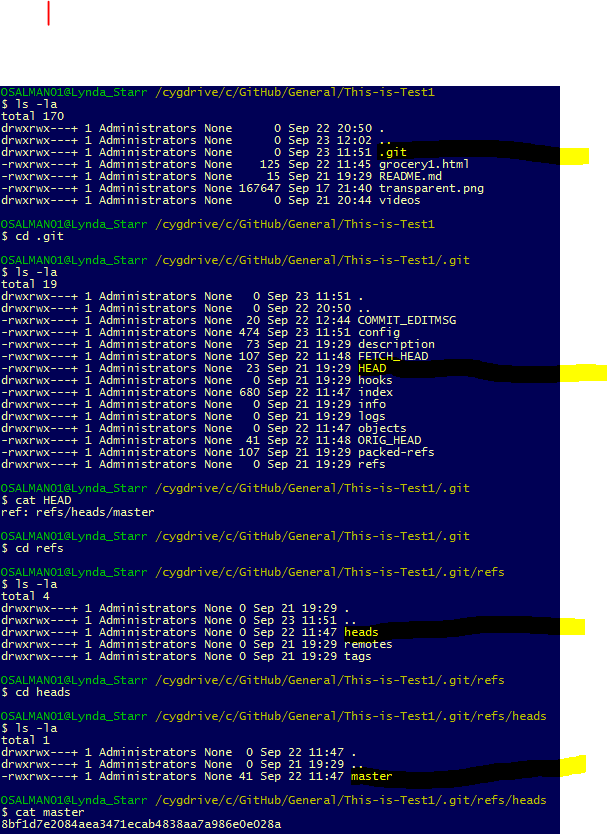








The following shows how find where the head points to location/tip of currently checked out branch;



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

To look at the global configuration file, .gitconfig, we do;

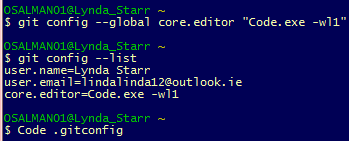
First go back to user home directory, by typing <cd ~>

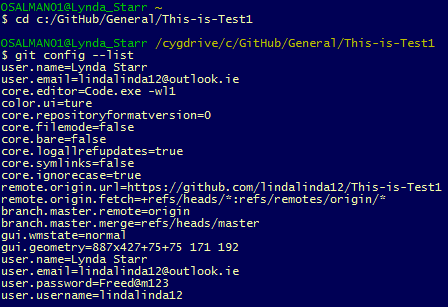
Then, we type <cat .gitconfig> to allow git to use different colours.

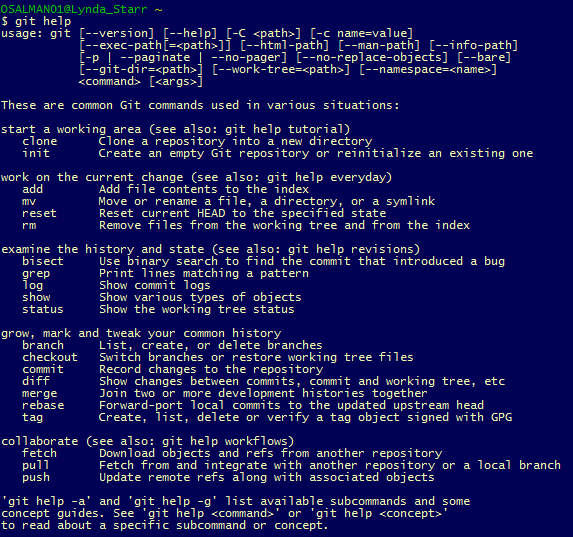
<git config --global core.editor “your favourite editor –wl1”> to decide the default editor to open text files.

-wl1 is used to tell bash to wait for the text editor to finish, and to start cursor on line 1.

<git config--global color.ui true> is used to







You can also use <man git log>

<git help “command”> or <man git -“command”> for example, <man git-log> to get info about a git command.

**For comprehensive explanation of help, we use <man git>**

When in help, you can go forward by pressing f and backward by pressing b. You quit by pressing q.

**<git init>** This command creates an empty Git repository - basically a .git directory with subdirectories for objects, refs/heads, refs/tags, and template files. An initial HEAD file that references the HEAD of the master branch is also created. Running <git init> in an existing repository is safe. It will not overwrite things that are already there. The primary reason for rerunning <git init> is to pick up newly added templates (or to move the repository to another place if --separate-git-dir is given).

System level, global to the user level and project level configuration setting.

Add the following to target of the Cygwind64 Terminal to have the directory inside the correct folder;

C:\cygwin64\bin\mintty.exe -i /Cygwin-Terminal.ico

To add clear screen, cls to Cygwin terminal, do the following;

Open Windows Explorer then navigate to:

[Cygwin Installation Directory]\home\[User]

e.g.: C:\cygwin\home\OMSALMAN

or to, C:\Program Files\Git\etc

then open the file .bashrc with Notepad++.

Move your cursor at the end of the last line and hit Enter key, then add the text below.

alias cls='echo -e "\033c"' // creates extra space at start

or the text below:

**alias cls='printf "\033c"' // this is better**

Restart cygwin, type cls. Done!

In Git on local machine only and not online, we only need to do the following;

