Title: git usage on Windows (command line)
Project Descriptor: LANDISII Upgrades Corrections Project
Project ID: 2016SoE021_LANDIS_Upgrades
Author: bmarron
Origin Date: 07 Nov 2016
Revision Date: 09 Nov 2016

######################
Setup steps and checks
######################

======= STEP 1: get the ProGit manual ==================

        a. download .pdf of book
https://git-scm.com/book/

======= STEP 2: install Windows git with "git BASH" ===============
https://git-for-windows.github.io/

        a. install defaults
        a1. after installation there will be a git BASH terminal icon on your Desktop
        a2. all subsequent directions in this document assume the installation of
            "git BASH"
        a3. all command-line commands below are typed into the BASH terminal
        a4. all command-line commands below are prefixed with a "$"; expected output
            from command-line commands is given in many cases

==== (optional) STEP 2: install "GitHub for Windows" (in addition to "git BASH") ====
https://desktop.github.com/
https://help.github.com/desktop/guides/getting-started/installing-github-desktop/
#platform-windows

        a. install defaults

#######################################
know the git workflow for file changes
#######################################

        a. the git file processing sequence:

tracked ==> modified   ==> staged ==> commited ==> pushed
                new   ==> tracked and staged ==> commited ==> pushed

###
clone a (global) offsite repository to a (local) onsite repository
###

===== STEP 1: create a directory (folder) for a (local) onsite repo ==========

        a. make a new folder to house the to-be cloned repo

====== STEP 2: clone a (global) offsite repo from GitHub ======================

```
        a. Example: clone a repo from the LANDIS-II Foundation GitHub site
        a1. go to the LANDIS-II Foundation GitHub site
        a2. Get address of the repo of interest (eg LANDVIZ Development)
<<https://github.com/LANDIS-II-Foundation>>
<<https://github.com/LANDIS-II-Foundation/LANDVIZ-Development>>

        b. in Windows FileExplorer, go to the newly-created directory (folder)
           for the (local) onsite repo
        b1. Right-Click on the folder; select 'Git Bash here'; open git BASH terminal
        b2. clone the (global) repo
$ git clone https://github.com/LANDIS-II-Foundation/LANDVIZ-Development

Cloning into 'LANDVIZ-Development'...
remote: Counting objects: 868, done.
remote: Compressing objects: 100% (481/481), done.
remote: Total 868 (delta 341), reused 868 (delta 341), pack-reused 0
Receiving objects: 100% (868/868), 84.04 MiB | 2.07 MiB/s, done.
Resolving deltas: 100% (341/341), done.
Checking connectivity... done.

        c. check contents of the newly-created (local) onsite repo
        c1. NB. a git-tracked directory will have a .git folder and
            git BASH will have the path will end in "(master)" (in green)
$ ls

LANDVIZ-Development/

$ cd LANDVIZ-Development
$ ls -A

.git/ .gitignore  Documentation/  PreProcTool/  README.md  WebVisTool/




======= STEP 3: do checks on general git (local) repo ===========

        a. check current git settings
$ git config --list

        b. add git command to see the last commit
        b1. check last commit
$ git config --global alias.last 'log -1 HEAD'
$ git last

commit f9911efc7156a5f40cedf87923fca0426f4ac67d
Author: johannesliem <johannes.liem@digitalcartography.org>
Date:   Thu Jul 23 11:10:44 2015 +0200

    Update README.md


        c. check the onsite/offsite repository connections
        c1. a 'repository' may be a (local) onsite repository OR
            its (global) offsite image; the two are connected through git
        c2. NB. The default name of the cloned offsite repo is "origin"
        c3. NB. The default name of the main branch in git is 'master'
        c4. NB. "HEAD" is a pointer to the current (local) repo branch you're on.
$ git remote -v

origin  https://github.com/LANDIS-II-Foundation/LANDVIZ-Development (fetch)
origin  https://github.com/LANDIS-II-Foundation/LANDVIZ-Development (push)
```

```
$ git remote show origin

* remote origin
  Fetch URL: https://github.com/LANDIS-II-Foundation/LANDVIZ-Development
  Push  URL: https://github.com/LANDIS-II-Foundation/LANDVIZ-Development
  HEAD branch: master
  Remote branch:
    master tracked
  Local branch configured for 'git pull':
    master merges with remote master
  Local ref configured for 'git push':
    master pushes to master (up to date)

        d. check branches
        d1. NB. --all shows both local and remote branches
$ git branch --all

* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/master


$ git checkout master

Already on 'master'
Your branch is up-to-date with 'origin/master'.




#########################################################################
the basic workflow:
1) make changes to a tracked file (or create a new file)
2) then stage it, commit it, and push it
#########################################################################

=== STEP 1. check status of git-tracked files and list currently tracked files =

        a. check status of git-tracked files
$ git status

On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean

        b. list the currently tracked files
$ git ls-files



==== STEP 2. make changes and check status ===================================

        a. Example: paste a new file (lyrics.txt) into the (local) onsite repo
        a1. copy and paste a new file into the (.git tracked)
            LANDVIZ-Development directory

lyrics.txt ==> pasted into local repo, LANDVIZ-Development


        b1. check new status of git-tracked files
        b2. NB new/modified/untracked/unstaged files are in red;
            tracked/staged files are in green
$ git status
```

```
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        lyrics.txt              <== in red

nothing added to commit but untracked files present (use "git add" to track)


==== STEP 3. stage new/modified files and check  ================================

        a. stage newly added files
        a1. "git add <file>" OR "git add --all"
$ git add --all
$ git status

On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   lyrics.txt          <== in green

        b. see (tracked) changes made (could use your favorite GUI here)
        b1. NB. this produces a long output; use "q" to quit
$ git diff --staged


==== STEP 4. commit the staged file ============================================

        a. set core editor for commits messages
$ git config --global core.editor "notepad"

        b. commit changes
        b1. NB. a text editor opens to add message; no message, no commit
        b2. check status post-commit
$ git commit

[master fc934c0] This a test of an https protocol, command line "push" to a
LANDISII repository (Marron)
 1 file changed, 29 insertions(+)
 create mode 100755 lyrics.txt

$ git status

On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
nothing to commit, working directory clean


==== STEP 5. push changes back up to the (global) offsite GitHub repo  =====

        a. push the changes made in the (local) onsite "LANDVIZ-Development" repo
           to the (global) offsite "LANDVIZ-Development" repo
        a1. will automatically ask for your GitHub user name and GitHub password
        a2. can only use the following command IF you have been given access to the
            global repo; o/w you will need to use a Pull Request to upload changes
        a3. on Windows, a GitHub box will open requesting your GitHub account
            info (name and password)
        a4. on Linux, command-line requests for GitHub info will open
$ git push origin master
```

```
Username for 'https://github.com': bmarron18
Password for 'https://bmarron18@github.com':
Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 720 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local objects.
To https://github.com/LANDIS-II-Foundation/LANDVIZ-Development
   f9911ef..fc934c0  master -> master



######################
when things go wrong
######################


        a. untrack all newly staged files
$ git reset

        b. uncommit a bad commit
$ git commit -m "Something terribly misguided"              (1)
$ git reset HEAD~                                           (2)
<< edit files as necessary >>                               (3)
$ git add << edited files >>                                (4)
$ git commit -c ORIG_HEAD                                   (5)

        c. untracking folders/files deleted since last update
$ git rm <folder> -r


        d. git help
$ git help <verb>
$ git <verb> --help
$ man git-<verb>







########################################
Another example of a git work cycle
########################################

==== STEP 1. make changes and check status  ====================================

        a. make changes to 'LANDIS_on_Linux1.txt' file
        a1. check status

$ git status
                                                    #unstaged: in red
                                                    #staged: in green
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   LANDIS_1/LANDIS_on_Linux1.txt              #unstaged: in red
```

```
==== STEP 2. stage and check  ========================================

        a. use 'git add <filename>'
        a1. stage the modified file

$ git add LANDIS_1/LANDIS_on_Linux1.txt
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   LANDIS_1/LANDIS_on_Linux1.txt              #staged: in green


==== STEP 3. commit the staged file ==================================

        a. use 'git commit -a -m "<message>"'
        a1.  -a flag ==> all staged files committed
        a2.  -m flag ==> adds the (required) message to the commit; o/w sends
            you to a text editor

$ git commit -a -m "12Aug2015"
[master e8e02ce] 12Aug2015
 1 file changed, 1 insertion(+), 1 deletion(-)


==== STEP 4. push the committed file ================================

        a. use 'git push <offsite name> HEAD'
        a1. (HEAD ==> A handy way to push the current working branch to the
             same name in the offsite)
        a2. use 'git push <offsite name> <onsite branch>'

$ git push Repo_1 master
Username for 'https://github.com': bmarron18
Password for 'https://bmarron18@github.com':
Counting objects: 7, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 457 bytes | 0 bytes/s, done.
Total 4 (delta 1), reused 0 (delta 0)
To https://github.com/bmarron18/Repo_1.git
   7831ea4..e8e02ce  master -> master
```