Git Console Commands

Updated: November 19, 2019

Legend:

- '$' indicates a command in git bash. Do not include them when writing the command.
- Blue writing indicates a command in PowerShell.
- Words surrounded by <> indicate that something needs to be inserted into that spot. Do not include <> when writing the command.

Resources:

Git Command Line Tutorial: https://www.youtube.com/watch?v=HVsySz-h9r4

Background:

Your local repository consists of 3 "trees" maintained by git:

1. Working Directory (holds the working files)
2. Index (staging area for files before a commit is published)
3. HEAD (most recent published commit)

These trees are only local and committed files must also be pushed if using remote repositories. It's generally not good practice to commit many files at the same time with a generic message for all files. Instead, staging specific files for commit with their own individual messages is recommended.

Cloning Remote Repositories:

$ git clone <url> <where to clone>	// Clones an existing remote repository to local machine
$ git clone <url> .	// Clones an existing remote repository to current path

Setup:

$ git init	// Initializes local git repository
$ git status	// Check status of working tree

Staging:

$ git add <fileName>	// Adds file(s) to index/staging area
$ git add .	// Adds all files to index/staging area
$ git add *.html	// Adds all .html files to index/staging area
$ git reset <filename>	// Removes file from index/staging area
$ git reset	// Removes all files from index/staging area

Commits:

$ git commit -m '<comment>'	// Commits changes to index ('m' is for message)
$ git log	// Shows a list of commits w/ unique hash numbers

Updating Remote Repositories:

When working with others, **always pull** first before pushing your latest commit. When using branches, the $ git push -u command associates the local with the remote branch, so that in the future we can just use the $ git push and $ git pull commands without indicating the branch name.

$ git diff	// Shows changes from the last commit
$ git pull origin <branchName >	// Pulls latest from specific branch of remote repository
$ git pull	// Pulls from the current branch of remote repository
$ git push origin <branchName>	// Push to specific branch of remote repository
$ git push -u origin <branchName>	// The -u allows us to associate local w/ remote branches
$ git push	// Push to the current branch of remote repository

Branches:

$ git branch	// Lists all the local branches, including current
$ git branch -a	// Lists all the local and remote branches
$ git branch <branchName>	// Creates new branch
$ git checkout <branchName>	// Changes working branch
$ git branch -d <branchName>	// Deletes branch locally
$ git push origin --delete <branchName>	// Deletes branch on remote repository

Merges:

Before merging back to master, always remember to $ git checkout master, then $ git pull origin master to sync any changes that may have been made while we were working on the branch.

$ git branch --merged	// Shows the branches that have been merged
$ git merge <branchName>	// Merges another branch into active branch
$ git add <fileName>	// After editing conflicted files, mark files as merged
$ git diff <sourceBranch> <targetBranch>	// Preview changes before merging

Cloning Remote Repositories:

$ git clone <url> <where to clone>	// Clones an existing remote repository to local machine
$ git clone <url> .	// Clones an existing remote repository to current path

Configuring Remote Repositories:

$ git remote -v	// Lists remote repositories
$ git remote add origin <URL>	// Links remote repository with local repository
$ git remote remove origin	// Removes the remote repository

Console Navigation:

$ ls	// Lists items in current directory
ls -Force	// PowerShell: Lists all items (including hidden)
$ cd	// Changes directory

Git Ignore:

$ touch .gitignore	// Creates a blank .gitignore file
new-item .gitignore	// PowerShell: Creates a blank .gitignore file

Now, open file and place names of items to be ignored:

- filename.txt
- /foldername or foldername/
- *.txt (all text files)

Other:

$ git --version	// Indicates current version of git
$ touch <filename.filetype>	// Creates blank file of specified type
new-item <filename.filetype>	// PowerShell: Creates a blank file of specified type
$ git config --global user.name "<Name>"	// Stores username that will show in central repository
$ git config --global user.email "<Email>"	// Stores email that will show in central repository
$ git config --list	// Shows the git config details
$ git help <verb>	// Shows help for the specified git action

Remove Git Tracking:

$ rm -rf git	// Removes the .git folder (ending version control tracking)

Storing Credentials:

git config --global credential.helper store
git pull (use after storing)

Git Console Examples

Connect Project to Github Repository Example:

1. Have pre-existing files to be backed on github
2. Navigate to project folder containing all relevant files
3. Right click on/in folder and select "Git Bash Here"
    - touch .gitignore
    - paste relevant gitignore from internet
4. $ git init
5. $ git remote add origin <git-link>
6. $ git add . // Note the period after "add" to indicate to add <u>all</u> files
7. $ git commit -m "Initial commit"
8. $ git push origin master

Clone Git Project From Github Repository Example:

1. Have clone link of github repository
2. Navigate to local folder where repository is to be cloned
3. Right click on/in folder and select "Git Bash Here"
4. $ git clone <git-link>

Clone Git Project From Github Repository Alternative Example:

1. Have clone link of github repository
2. Navigate to local project folder where files are to be cloned
3. Right click on/in folder and select "Git Bash Here"
4. $ git init
5. $ git remote add origin <git-link>
6. $ git pull origin master

Create/Merge Branch Example:

Note:

- (b-name → branch name)
- Useful command: "$ git branch -a" // Lists all branches (* indicates the current working branch)
  // Can be used to double check working branch or if branch was
  // created/deleted successfully

1. Navigate to master branch.
2. $ git branch <b-name> // Creates new branch
3. $ git checkout <b-name> // Changes local working directory to named branch
4. Make changes to files as usual
5. Commit changes locally as usual
6. $ git push -u origin <b-name> // Pushes and associates local branch with remote branch
   // After this command is used for the first time, can use
   // "$ git push" and "$ git pull" for future changes to branch
7. $ git checkout master // Changes local working directory back to master branch
8. $ git pull origin master // Pulls changes from remote master branch (if any)
9. $ git merge <b-name> // Merges branch into local master
10. $ git push origin master // Pushes changes to master remote repository
11. $ git branch -d <b-name> // Deletes local branch
12. $ git push origin –delete <b-name> // Deletes remote branch