Bash command line and Git

Sofiullah Iqbal Kiron

31 March, 2021

If git is already installed in your system, then bash command line interface are installed also. Type and run command from it, the command might be Linux, UNIX or standard git command. All of the git command starts with keyword `git`.
Simple bash commands:

Bash Commands

1. To get help or print manual list of any command just type `command --help`
2. Sometimes we want to create a new file or there may be times when the requirement is to change the timestamps of a file. The `touch [option]...FILE...` command is primarily used to change file timestamps, but if the file(whose name is passed as an argument with extension) doesn't exist, the the command will create it. Ex. `touch newText.txt` will create a new text file at working directory by that specified name if not already exist.
3. To remove a directory and all its contents, including sub-directories and files (cmd command): `rm -r "directory name"` r stands for recursive. That means recursively remove all.
4. Powershell version: Run as administrator, type `$PSVersionTable`, hit enter.
5. Linux/UNIX command to write something on a file: `echo message > fileNameWithExtension` Ex: `echo Hello > file.txt`
6. Directory:
   (a) Make: `mkdir dirName`
   (b) Change working directory: `cd dirNameWithFullPath`
   (c) Path of working directory: `pwd`
7. `Start`:
   (a) `explorer`
   (b) `texmaker`
   (c) `msedge` or `msedge <link>`
   (d) `chrome` or `chrome <link>`
   (e) `brackets`
   (f) `opera` or `opera <link>`
   (g) `cmd`

   (h) `powershell`

8. List:
   (a) Simple: `ls / dir`
   (b) Hidden included: `ls/dir -a`
   (c) Hidden included with details: `ls/dir -all`
9. Let's some fun:
   (a) `factor n` will print prime factors of `n`. It is a build-in program like we say.
10. If we wanna see all the commands(that we typed and hit enter, not matter right or wrong the command was) as a list: `history`
11. Clear the history: `history -c`
12. Clear console: `clear`
13. Exit from bash terminal: `logout`

Git commands

1. Check git version: `git --version`
2. Initialize a new empty local repository: `git init` :- a hidden folder will be created with name ".git"
3. Adding/Staging(give access to git tracker) files:
   (a) Specified: `git add fileName`
   (b) Multiple: `git add file1 file2 file3`, space separated.
   (c) Specified by extension: `git add *.ext`
   (d) All: `git add .`
4. Staging logs:
   (a) Full log: `git log`, press 'q' to quit from long log list. Each log come up with a unique hexadecimal ID(String with 40 characters maybe). Each commit contains a complete snapshoot of working directory. log is the history the snapshoots of a repository.
   (b) Press space for move to the next page of log.
   (c) One lined: `git log --oneline`
   (d) Last N commits by a non negative integer: `git log -N`, N means non-negative integer as, 2, 6. eg. `git commit -3` will show last 3 log.
   (e) In reverse order, full: `git log --reverse`.
   (f) In reverse order, one lined: `git log --oneline --reverse`.
   (g) Show changed files in each commit: `git log --stat` or `git log --oneline --stat`
   (h) See actual changes in each commit: `git log --patch` or `git log --oneline --patch`
   (i) Show commit changes: `git show commitID` or, `git show HEAD n`, here HEAD is the last commit we did and n is an integer indicating that go n steps back from HEAD and then show the commit changes.
   (j) Filter logs to be shown by
      i. Author: `git log --author="author_name"`
      ii. Date
      iii. Before with this date: `git log --before="yyyy-mm-dd"`
      iv. After with this date: `git log --after="yyyy-mm-dd"`

v. Indication: `git log --before/after="yesterday/one week ago/two week ago/one month ago/two month ago"`

vi. the commit messages those has the string/substring "WORD". It is case sensitive. : `git log --grep="WORD"`

5. File list in staging area: `git ls-files`

6. File list with tree: `git ls-tree commitID` or, `git ls-tree HEAD n`

7. Status:

   (a) Full status: `git status`

   (b) Short status: `git status -s`, I think, `-s` stands for "short".

8. Configuration:

   (a) Full: `git config --list`

   (b) Specified by key: `git config <key>` e.g. To get user name that already been configured, git config user.name

   (c) `git config --global user.name "My Name"`

   (d) `git config --email myEmail`

9. Get a full copy of an existing repository: `git clone <url>`

10. Set default editor for git

11. Open the dot git folder: `start .git`

12. Remove tracking access from git: delete the `.git` hidden directory.

13. Update git: `git update-git-for-windows`

14. Every git commit contains a distinct ID, Message, Date/Time, Author, Complete snapshot of project.

15. Remote:

   (a) Remote repository is a repository exits on server/online.

   (b) Add: `git remote add <shortname> <url>`

   (c) Lists all the remotes with server link details: `git remote -v`

   (d) Inspect: If you want to see more information about a particular remote: `git remote show <remoteShortName>`

   (e) Rename: `git remote rename <oldShortName> <newShortName>`

   (f) Remove: `git remote rm <shortname>` or `git remote remove <shortname>`

   (g) To see remote servers you have configured, you can run the `git remote` command. It lists the shortnames of each remote handle you've specified. If you've cloned your repository, you should at least see "origin" that is the default name Git gives to the server you cloned from. Can use `-v` option to shows Related URLs that git has stored for the shortname.

16. **Have Bug**, After adding a remote to a repository, we can fetch by the shortname: `git fetch <shortname>`