Bash and Git summary

Diego Ramírez Milano

May 28, 2015

1 Introduction

This document will contain a beginner's (or rather start-up) guide for using bash code and git commands. All the code and commands here stated are executed in Ubuntu 14.04 LTS version.

2 Bash

Bash (Bourne again Shell) is interpreter program for unix or shell commands. Its most common use and access is from unix-based OS terminals or consoles. To open and start using the terminal just hit `ctrl + alt + T` on your keyboard. As the use that can be given to the terminal heavily depends on your knowledge of unix commands this section will summarize some basic commands.

- `whoami` : Prints the name of the current user.
- `pwd` : Prints the current working directory
- `cd` : Changes the current directory. Just typing `cd` will return you to the `home` directory. Typing `cd ..` will go up one directory. Typing `cd directoryName` will go further down to that directory.
- `ls` : Lists all the files and directories inside the current directory.
- `echo` : Prints arguments separated by a space and terminated by a new line.
- `type` : Displays information about a command.
- `read` : Allows to input the value of a variable.
- `tee` : Copies the standard input to the standard output and one or more files.
- `date` : Prints the current date and time.
- `test` : Compares between two inputs.
- `if` : Evaluates a list of one or more commands and executes a list if the execution of the <condition list>is successful.
- `then` : Executes when the `if` statement is *true*.
- `else` : Executes when the `if` statement is *false*.
- `fi` : Finish the `if` statement execution block.
- `case` : Compares a word (usually a variable) against one or more patterns and executes the commands associated with that pattern.
- `&&` : And operator.
- `||` : Or operator.

- `help` : Displays a help page of the basics of using a command.
- `while` : Repeatedly executes a set of commands while a list of commands executes successfully.
- `until` : Opposite of while. Repeatedly executes a set of commands until a list of commands executes successfully.
- `for` : Repeatedly executes a set of commands for each word in a list.
- `break` : Exits form a loop.
- `continue` : Starts the next iteration of a loop immediately.
- `do` : Execute command block inside a loop.
- `done` : Finishes up the loop.
- `man` : Followed by another command displays the manual of the latter command.
- `mkdir` : Creates a directory with the name that follows the command.
- `rmdir` : Deletes the directory with the name that follows the command.
- `touch` : Useful for creating empty files, also used to change the *timestamps* of existing files and directories.
- `cp` : Copies the file with the name that follows the command to the desired location.
- `mv` : Moves the file with the name that follows the command to the desired location.
- `source` : Executes a script in the current shell environment.
- `.` : Executes a script in the current shell environment.
- `sh` : Executes a script in the current shell environment.
- `chmod` : Modifies the permission of a file. `chmod +x` is used to make files executable.
- `printf` : Prints the argument as specified by a format string.
- `wget` : Downloads a file from the URL that follows the command.
- `tar -cvf` : Used to compress files into `compressed.tar`.
- `tar -xvf` : Used to decompress `.tar` files.
- `gzip` : Used to compress files into `compressed.gz`.
- `gunzip` : Used to decompress `.gz` files.
- `less` : Displays in the terminal a `.txt` file.
- `head -n` : Displays the first n lines of a `.txt` file in the terminal.
- `tail -n` : Displays the last n lines of a `.txt` file in the terminal.
- `cat file1 file2` : Concatenates and prints the files.
- `tac file1 fil2` : Concatenates and prints the files in reverse.
- `grep` : Looks for the input pattern in a file and prints the lines that contains it.
- `wc` : Prints the lines, words and bytes count in a file.
- `ssh` : Connects to another computer.

- `top` : Opens the resource and running programs in the computer. Linux equivalent to Windows task manager.
- `kill -9 PID` : Kills (terminates) the process with the entered PID.
- `sed 's\a\b\g'` : Replaces all the "a" occurrences for "b" in the desired file.

3 Git

To start off you must create a GitHub user, you may do so here. Afterwards download Git to be able to work on it. Go to the terminal and type the following lines of code:

`$ sudo apt-get install git` / installs Git
`$ sudo apt-get update` / gets the newest Git version

After the download is done it is time to configure your username, email (the ones you used to sign up in GitHub) and preferred text editor that will be used when Git needs you to type in a message. I'll use my information as an example.

```
$ git config --global user.name "Diego Ramírez Milano"
$ git config --global user.email "dl.ramirez1796@uniandes.edu.co"
$ git config --global core.editor gedit / you can use any other text editor such as emacs or vi[1]
```

You can check your whole configuration settings by typing `git config -l` and you will get something like this:

```
user.email=dl.ramirez1796@uniandes.edu.co
core.editor=gedit
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
remote.origin.url=https://github.com/diegolramirez/Notas.git
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
branch.master.remote=origin
branch.master.merge=refs/heads/master
```

To create a repository head to GitHub and click the plus sign on the top right corner and select the option "New repository". Enter the desired name and optionally add a description and a readme file. Once this is done head towards the directory you wish to store the repository locally in your computer. Now you have to clone the repository from GitHub to your own computer, this will enable you to further modify it and gain an https key that secures your connection and allows you to upload code to GitHub.

```
$ git clone https://github.com/diegolramirez/MC.git
```

Once you have cloned the repository a new directory with the repository's name, which you can access, will be created. Here you can manage all the files that the repository contains. To do so you have to follow three steps:

1. Add the files manually to the directory and then from the terminal add them to the repository using the command `git add FileName` or `git add .` to add all the changes that have been made.

[1] Depending on the chosen text editor you may want change the backup files settings in order to avoid the creation of unwanted hidden files.

2. Add a commit that describes the changes you made using `git commit -m "messageGoesHere"`.

3. All the above changes where made locally, to upload them to GitHub you need to *push* them up. Use the command `git push origin master` to do so. After you hit enter a message will be prompted asking for your username and password, correctly type them in and your changes will happen in your GitHub repository.

At any time you may check your current data modification status by typing the command `git status`. This will state any changes made, including added, deleted or modified files. Here you can see a scheme of how uploading things to GitHub and using Git works:

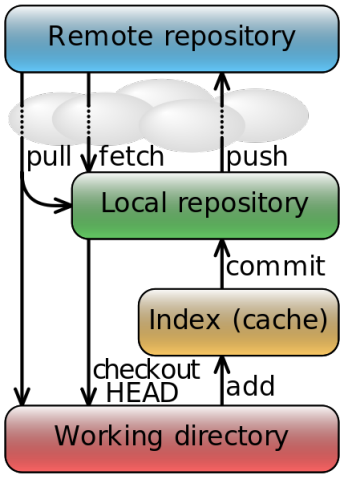

Figure 1: Git data flow scheme

You can check a log of the changes made to the repository, this is why you make a commit every time something is uploaded. To do so just type `$ git log`. This command has a variety of options to visualize the different commits ever made to the repository.

Given the case you made a mistake on your last commit and have to modify it type the command

```
$ git commit --amend
```