TUTORIAL GIT AND GITHUB EXPRESS - 2018

Note on this one page express tutorial

It may take from days to years to learn git. According to an expert on git, he is still learning git even after more than 5 years of working on git. However, in this page, I'll try to make the process of learning as simple as possible: one chapter tutorial!

Though we can use GUI version, git is absolutely a command line utility. So, we'll be working mostly on command line mode. As we learn git, we'll realize the command line is really a first class citizen, and it's the core of a git.

After finishing this one page tutorial, we'll learn:

1. Git install
2. Creating a git repository
3. Adding a file to a git
4. Github - repository
5. Collaborative working via forking central repo
6. Github - clone
7. Branching
8. Pull request
9. Pulling from a central repo & merging

In this express tutorial, we'll work on Ubuntu 14.04 as a local repository and we'll use two separate Github accounts: one for a primary repo and the other one for contributor to the primary repo.

Git Install

We can install the Git command line tool using the command below:

$ sudo apt-get install git

$ git --version

git version 1.9.1

For more details on installation:

1. [GIT and GitHub - 1. Installation](http://www.bogotobogo.com/cplusplus/Git/Git_GitHub_Installation.php)
2. [GIT on Ubuntu and OS X - Focused on Branching](http://www.bogotobogo.com/cplusplus/Git/Git_Ubuntu.php)

Creating a git repository

$ git init project1

Initialized empty Git repository in /home/k/GitTraining/project1/.git/

Note that we do not have any server, and there is no background daemon. We just used local file system to create the project1 directory and the nested .git directory.

$ cd project1

$ ls

$ ls -al

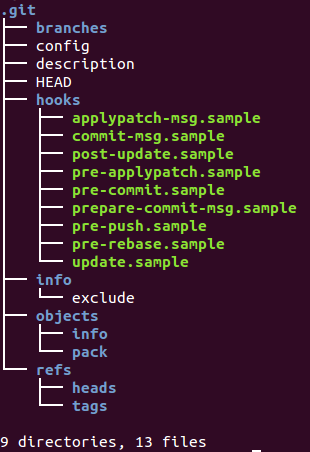
total 12

drwxrwxr-x 3 k k 4096 Jun 3 09:52 .

drwxrwxr-x 3 k k 4096 Jun 3 09:52 ..

drwxrwxr-x 7 k k 4096 Jun 3 09:52 .git

$ tree .git



Unlike other source control system such as CVS, there is only one **.git** folder at the top level. Only one **.git** per repository!

Also, note that we do not have any file in the repository yet:

$ git status

On branch master

Initial commit

nothing to commit (create/copy files and use "git add" to track)

Now we make our first ifle: **first.txt**.

Let's see how the git think of the file:

$ git status

On branch master

Initial commit

Untracked files:

(use "git add ..." to include in what will be committed)

first.txt

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

Adding a file to a git

$ git add first.txt

The **git add** is merely telling the git our intention of adding for the next transaction. It's not adding the file to a repo yet. It just signals our participation. We do not have a permanent recode of the file yet.

We can see the changes to be committed using **git status**:

$ git status

On branch master

Initial commit

Changes to be committed:

(use "git rm --cached ..." to unstage)

new file: first.txt

Now we can commit to the **master**:

$ git commit -m "My first commit"

[master (root-commit) b025f57] My first commit

1 file changed, 50 insertions(+)

create mode 100644 first.txt

Here, the **b025f57** is a global unique identifier. The **644** indicates the user can read and write and others and group just can read the file.

Now we have permanent record of the file and we can see our current directory is clean:

$ git status

On branch master

nothing to commit, working directory clean

So far, we've been working on **local**. Now, we may want to use network (remote): **github**.