This document describes some generic code for Git repository.

First, go to a folder which you want to select and create a repository. **Right click** on the folder, select “**Git Bash Here**”,

* there will be a “.git” folder created in this folder which means this folder will be transformed as a repository. The “.git” folder also contains a lot of files which are necessary for making this folder as a repository.
* Along with the “Git Bash Here” action, it will also fire up the MINGW64 command window for you to operate the files within Git.

Below are some useful commands to operate within Git:

* **$ touch index.html**

Create a html file named index in the repository folder.

* **$ git init**

Initialize empty Git repository

* **$ git config –global user.name ‘My Name’**

Add name to the Git repository

* **$ git config –global user.email ‘myname@email.com’**

Add email to the Git repository

* **$ git add index.html**

Add the ‘index.html’ file into the created repository

* **$ git status**

Check the stage status of the files in the current repository. This will tell us what files has been added into the repository or in the staging area.

* **$ git rm –cached index.html**

Remove the ‘index.html’ file from the stating area of the created repository.

* **$ git add \*.html**

Add all the ‘.html’ files into the current repository

* **$ git add .**

Add all the files in this folder into the current repository

* **$ git commit**

Commit the change to the repository. This will allow the user enter the note editor, but user has to type ‘i’, stands for insert, in order to change any notes in this editor.

Then hit ‘Esc’, let the user out of the insert mode.

Then type ‘:wq’, allow the user to return to the MINGW64 Git command window.

However, there is a better to commit the change. See the code shown below.

* **$ Git commit –m ‘Changed index.html’**

This is a faster way to commit the change of the files in the repository, with note says ‘Changed index.html’. This faster way will skip the editing processing mentioned in the previous example.

* **$ Clear**

Clear the contents in the MINGW64 command window.

* **$ touch .gitignore**

This will create a file named ‘.gitignore’

Using text editor to open this ‘.gitignore’ file, and add a file you don’t want to commit. For example, add ‘myPython.py’.

**$ git add .**

* This will add all the files in this folder, but not ‘myPython.py’ file.
* You can also add a folder name into this ‘gitignore’ file, i.e. ‘/myfolder’. Then the entire folder will not be added.
* Or add ‘\*.txt’ into ‘.gitignore’ file, then all the .txt files will not be added into the staging.
* **$ git branch mybrach**

Create a branch of the master repository, the branch is named ‘mybrach’. Then you need to commit the change again.

* **$ git checkout mybranch**

Switch from the master files into the ‘mybrach’ files in the repository.

If you make any change in the ‘mybrach’, you need to switch back to the master files, and merge the changes from the branch to the master files, following the command below.

* **$ git merge mybranch**

It will open the editor file just like the ‘git commit’ command does. User can follow instruction there to add notes.

* **$ git remote add origin** [**https://github.com/zmmcase/MyModules.git**](https://github.com/zmmcase/MyModules.git)

**$ git push -u origin master**

The first line will add our remote repository. And the second line will push the local master repository to the online GitHub.

For later use, you only need type ‘git push’, since it already knows your remote repository

How to use git lfs to upload large data files, so that large data files is saved in the same repository, but using a pointer, not actual space in this repository.

* $ git lfs track “\*.zip”
* $ git add .gitattributes
* First type “git lfs track “\*.zip”.
* Track all the ‘.zip’ files, so all the zip files are not saved in the repository, but saved somewhere else, but can access from the repository using pointer.
* Then, need to type “$ git add .gitattributes”.
* After this, use git add, and git commit normally.

For large file clone, we might want to use

$ git lfs clone