

Practical 5 : Working with Git

Continuing from the previous practical, this practical is exploring on Git on local machine.

Getting Started.

First off, download Git from git-scm.com as shown in Figure 1.

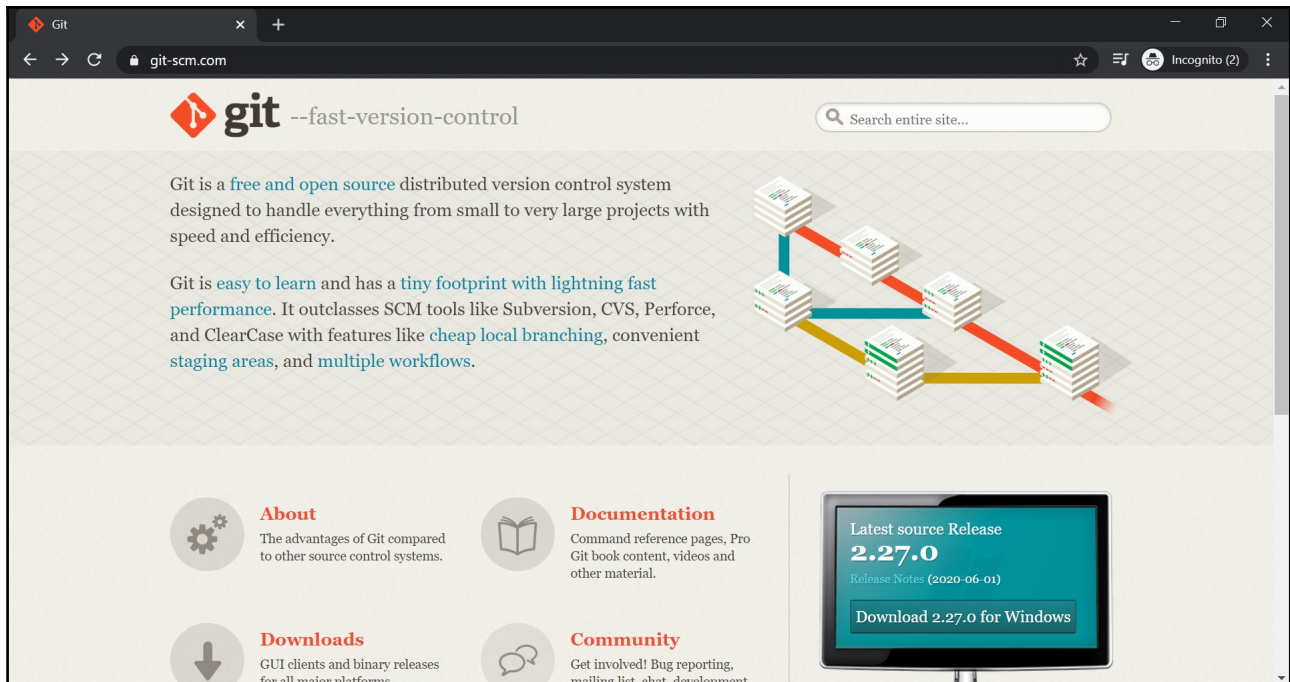


Figure 1: Git download.

After installing Git with all default settings, launch command line interface (CLI) / command prompt / shell, to start working with Git.

In order to make sure that Git is successfully installed in your workstation, type “git” and execute the command in CLI.

```

Microsoft Windows [Version 10.0.18363.900]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\looyi>git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
        [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
        [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
        [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
        <command> [<args>]

These are common Git commands used in various situations:

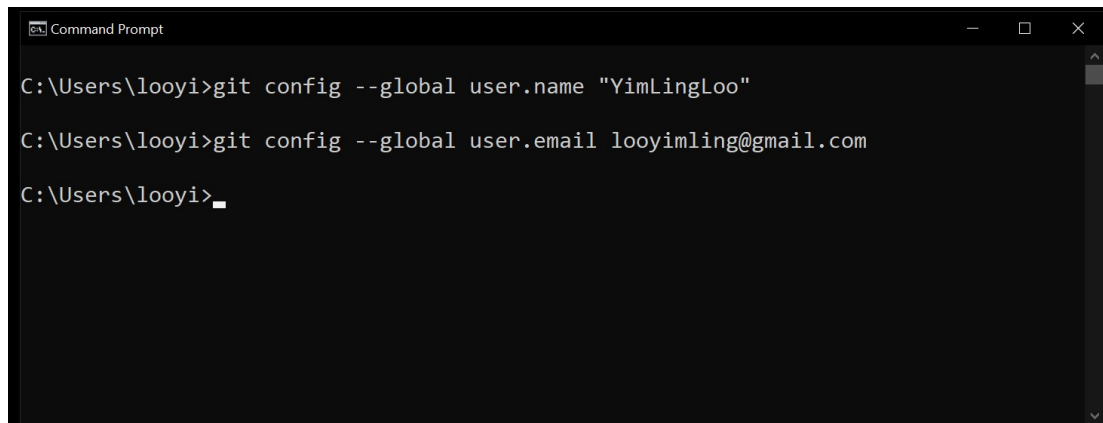
start a working area (see also: git help tutorial)
clone                  Clone a repository into a new directory
    
```

Figure 2: Git in CLI.

The fact that there are a list of usage of “git” as a command in CLI, confirmed that Git is successfully installed in the workstation. Look through the list of common Git commands. Looks familiar? Before trying any of the git command, let’s do exactly what was done in Github; login. It is the first and most crucial step to let Git know about the identity of the author as Git traces the authorization and keeps track of individuals who performs all the commits that are being done. In

UECS2363 SOFTWARE CONSTRUCTION AND CONFIGURATION

Github, we “signup” as a user through the GUI provided. In git, we register our identity through “git config” commands as shown in Figure 3 (*Take note: the email need to be the same email used to register Github account*).

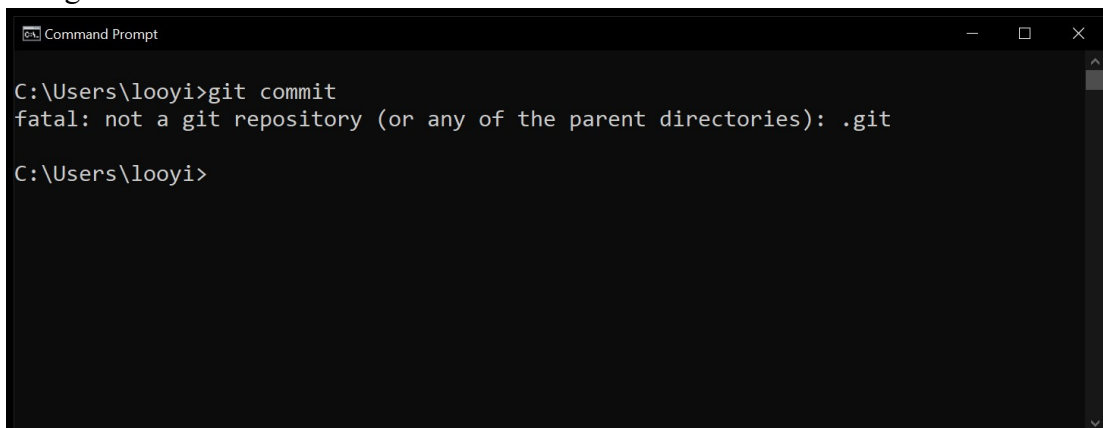


```
C:\Users\looyi>git config --global user.name "YimLingLoo"
C:\Users\looyi>git config --global user.email looyimling@gmail.com
C:\Users\looyi>
```

Figure 3: Configure identity of author to Git.

Execute “**git config -list**” command to double check whether the user.name and user.email are configured correctly.

After successful setup of identity, let’s try one of the most familiar ommand; “git commit” as shown in Figure 4.



```
C:\Users\looyi>git commit
fatal: not a git repository (or any of the parent directories): .git
C:\Users\looyi>
```

Figure 4: Execute a git commit on non-repository working directory.

Git command need to be executed on a repository. In other words, the workflow is the same as whatever was done in Github. Just that, the whole workflow is now being brought from a web server, into a local workstation and a GUI, into CLI.

In order to work on a repository, a repository need to be in the workstation. In order to have a repository in the workstation, one may choose to:

1. Clone a repo from Github by executing “git clone” command.
2. Initialize an empty repo by executing “git init” command.

As a starter, let’s have a copy of own repo that was initialized in Github in the previous week, to be in the workstation.

UECS2363 SOFTWARE CONSTRUCTION AND CONFIGURATION

Create a Clone Repo.

Login to Github account and go to the repository that was created before this. Click on “Code” drop-down button which was located on the right of “Add file” drop-down button and left of “About” section as shown in Figure 5.

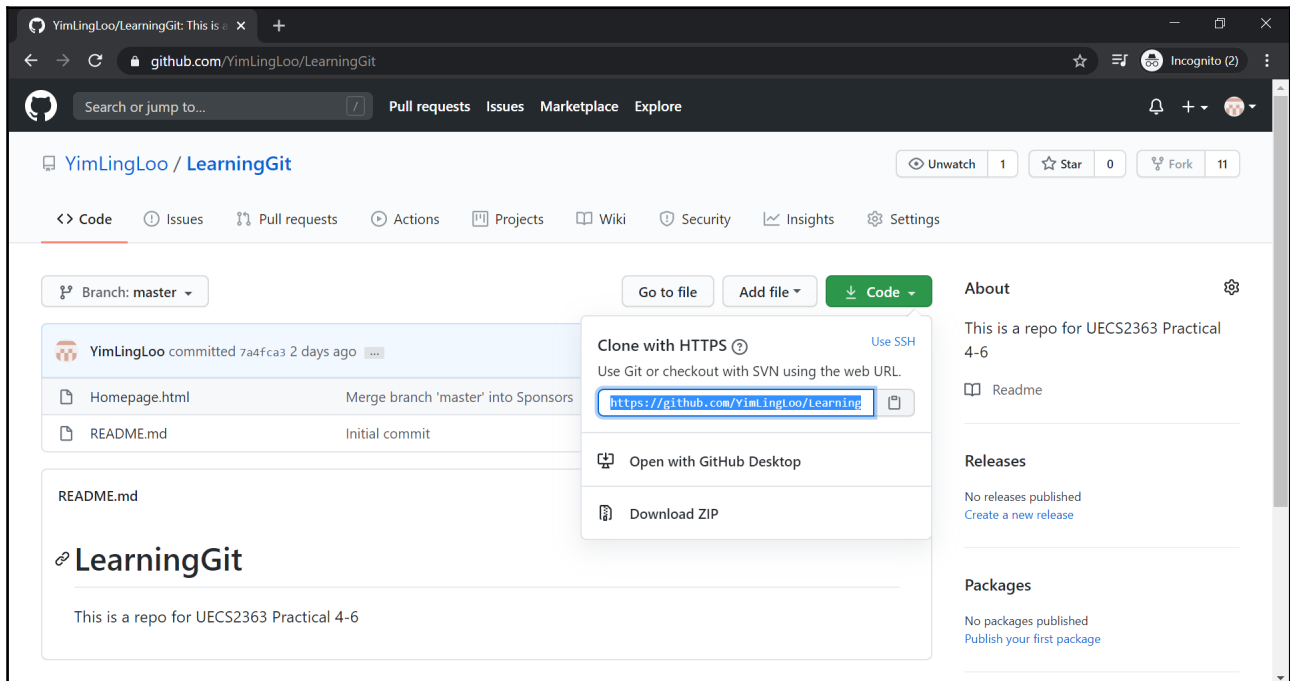


Figure 5: Clone a repo with url option in Github.

Take note to copy the special url provided in the textbox highlighted; to be used along “git clone” command in CLI as shown in Figure 6.

```
Select Command Prompt

C:\Users\looyi>git clone https://github.com/YimLingLoo/LearningGit.git
Cloning into 'LearningGit'...
remote: Enumerating objects: 19, done.
remote: Counting objects: 100% (19/19), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 19 (delta 4), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (19/19), 4.95 KiB | 32.00 KiB/s, done.

C:\Users\looyi>
```

Figure 6: Clone repo using “git clone” command and url.

By executing `git clone [repo url]`, Git list out the details of the clone and exit once cloning of repo is done. Confirm that the repository is now in the working directory by executing “dir” command.

If you are curious, open up the folder in your GUI to take a look into the working directory and the repository that had just been cloned as shown in Figure 7.

UECS2363 SOFTWARE CONSTRUCTION AND CONFIGURATION

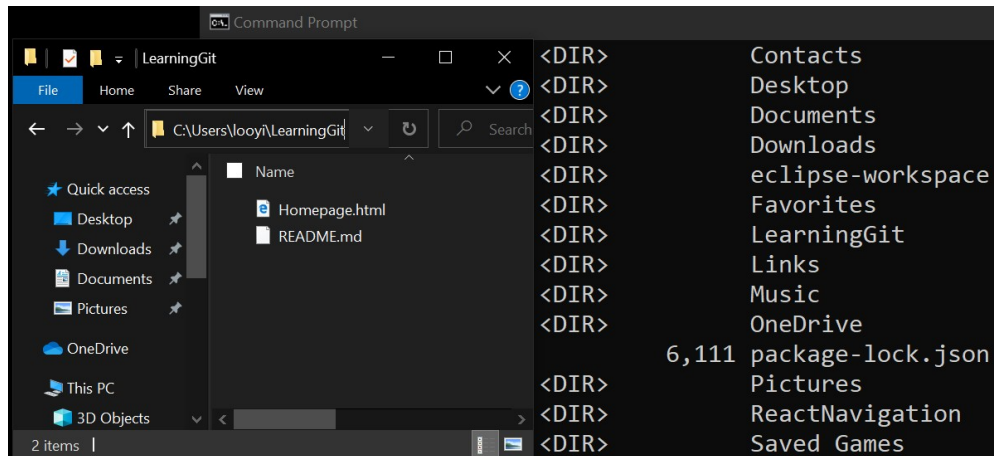


Figure 7: Cloned repo in working directory.

Monitor Repo Status.

Create some modifications to the repository. Let's open up the README of markdown file type in any text editors, modify the texts inside and save the changes.

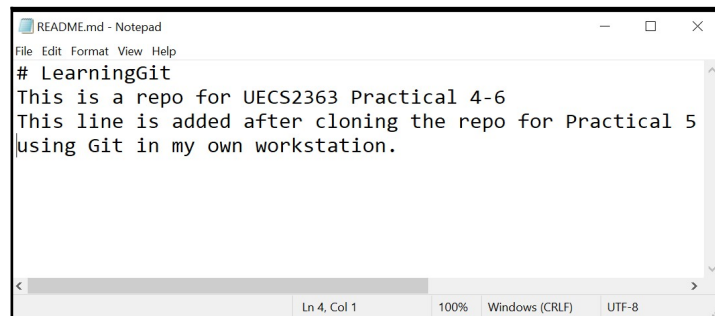


Figure 8: Modify the README.md

See to it that Git traced changes/modification made by executing “git status” in the repository. Remember that git commands need to be executed within a repository, just like how it was in Github? Thus, “git status” need to be executed inside the repo as well. How? The big command “cd” will do.

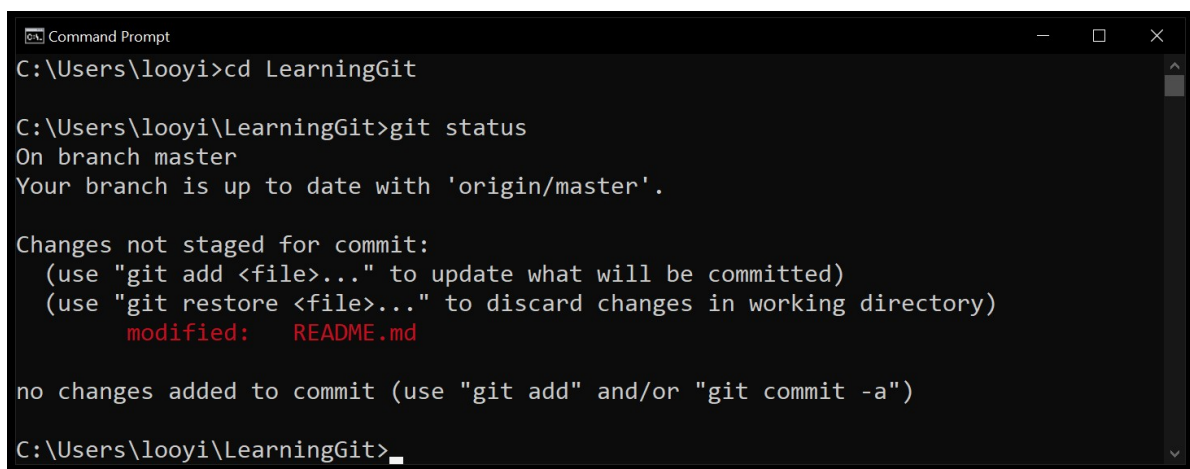


Figure 9: Monitor a repo using “git status” command.

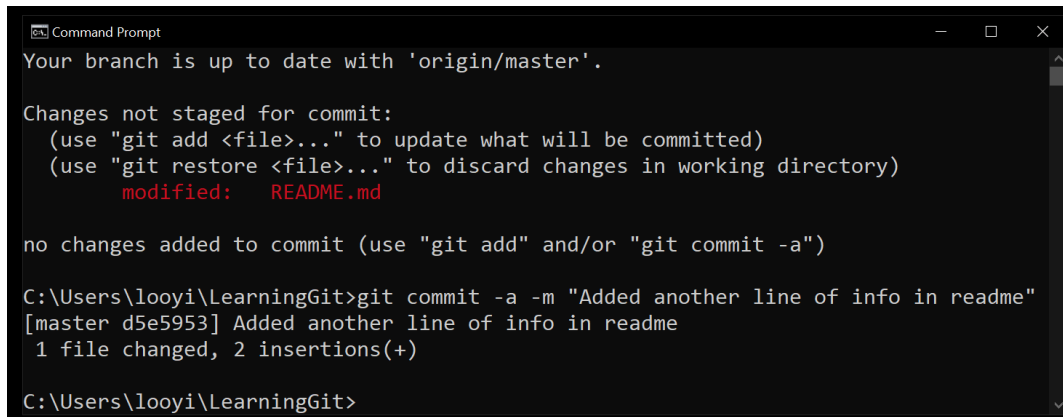
Take note that Git informs about details of the repo, inclusive of all the modifications made but **yet to commit into the repo**.

UECS2363 SOFTWARE CONSTRUCTION AND CONFIGURATION

Take note that one has saved the README.md, BUT, the change has not been committed into the repo. This implies, that for every change that is made into the repository files, once saved, the changes are in “float” status. One can choose to get rid of the change by not committing and keep the change by committing the changes.

Commit a Change to Repo.

If one decides to keep the change, “git commit” need to be executed in the CLI. “commit” in Git is similar as Github; committing a change need to have an argument; commit change **for** this file or that file and a descriptive **commit message**. Now, let’s just commit the changes for all file with argument “-a” and type in the commit message with argument “-m” as shown in Figure 10.



```
Command Prompt
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

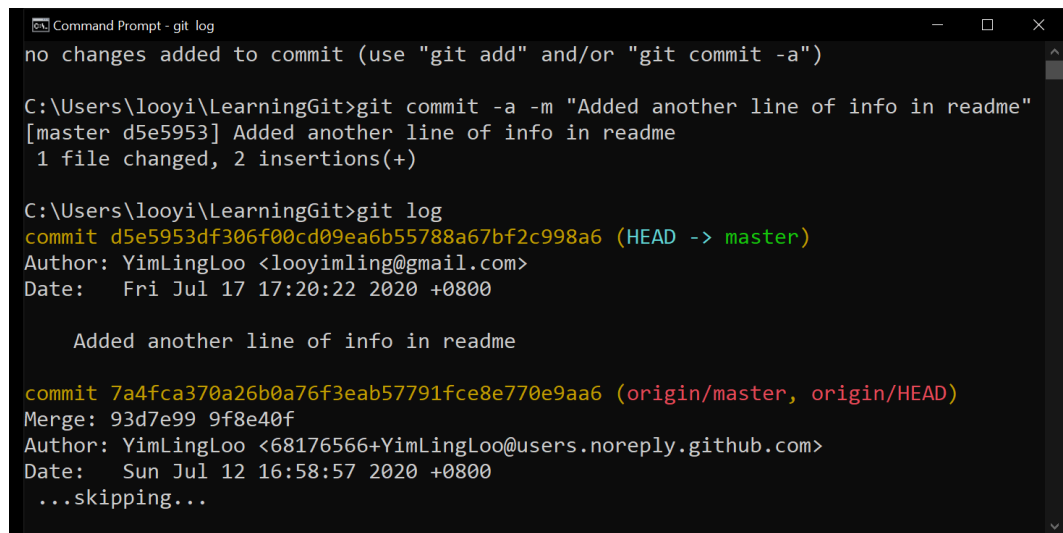
no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\looyi\LearningGit>git commit -a -m "Added another line of info in readme"
[master d5e5953] Added another line of info in readme
1 file changed, 2 insertions(+)

C:\Users\looyi\LearningGit>
```

Figure 10: Make commit object using git commit command.

In order to look at the commit objects made, “git log” is to be executed as shown in Figure 11.



```
Command Prompt - git log
no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\looyi\LearningGit>git commit -a -m "Added another line of info in readme"
[master d5e5953] Added another line of info in readme
1 file changed, 2 insertions(+)

C:\Users\looyi\LearningGit>git log
commit d5e5953df306f00cd09ea6b55788a67bf2c998a6 (HEAD -> master)
Author: YimLingLoo <looyimling@gmail.com>
Date:   Fri Jul 17 17:20:22 2020 +0800

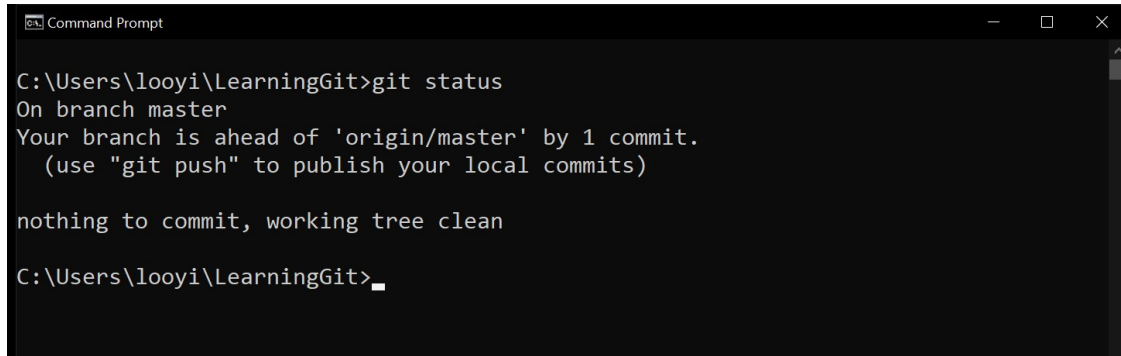
    Added another line of info in readme

commit 7a4fca370a26b0a76f3eab57791fce8e770e9aa6 (origin/master, origin/HEAD)
Merge: 93d7e99 9f8e40f
Author: YimLingLoo <68176566+YimLingLoo@users.noreply.github.com>
Date:   Sun Jul 12 16:58:57 2020 +0800
...skipping...
```

Figure 11: Listing out a log of commit objects made.

Take note: if Git brings you into the default Vim editor which displays the list with “END” as it’s ending point and did not quit the Git command execution, just type “q” or “shift” + “q” or “:” + “q” to end the command execution.

In order to confirm that commits are completed, execute the “git status” command again to see that the working tree is clean as shown in Figure 12.



```
Command Prompt
C:\Users\looyi\LearningGit>git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
C:\Users\looyi\LearningGit>
```

Figure 12: Always monitor repo with “git status” command.

After a successful commit of changes to the local repo, the original repo in Github did not reflect the change. Thus, a push request is to be executed to have the changes reflected in the original repository.

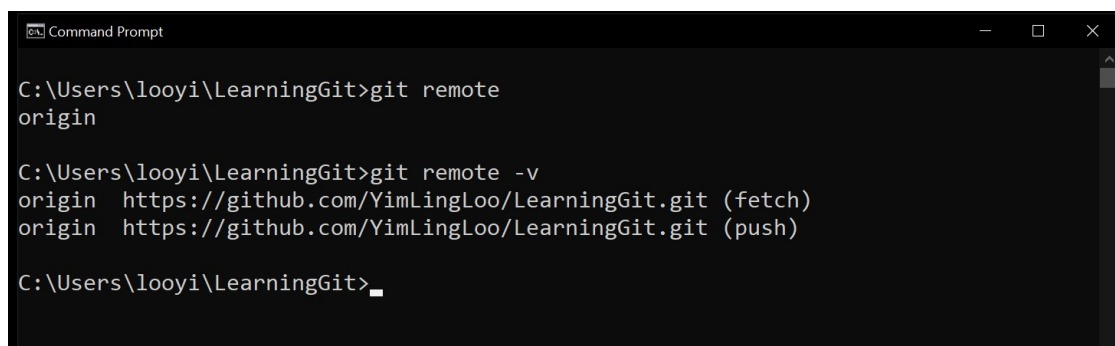
Create a Push Request to Update original Repo.

After a series of work locally on the workstation, one may update the original repo in Github using a push mechanism. When one clone a repo from Github to work in a local workstation, merging the changes done locally to Github is a push request. The merging of changes are made from the local server (developer’s workstation) to the remote server (Github).

In Git, one may use the “git push” command to push the changes to the original repo in Github.

However, there are certain things that developer needs to take note of, while working with Git. When one wants to create commit objects, Git needs to know the identity of the author who would like to create the commits. When one wants to push the changes to be merged with the original repo, one need to inform the location of this push to Git.

In order to know the location known by Git about the original repo, “git remote” or “git remote verbose” need to be executed as shown in Figure 13.



```
Command Prompt
C:\Users\looyi\LearningGit>git remote
origin

C:\Users\looyi\LearningGit>git remote -v
origin  https://github.com/YimLingLoo/LearningGit.git (fetch)
origin  https://github.com/YimLingLoo/LearningGit.git (push)
C:\Users\looyi\LearningGit>
```

Figure 13: Displaying the location of original repo.

The “git remote” command brings the name of the location for original repo as the output and if developer wants to know the detail of the name of location displayed verbose or “-v” argument should be used with the command.

With the information now, developer is able to inform Git that the “push request” is directing to the remote server known as “origin”. However, remember that this is a CLI. A merge of changes need to have information of not only the original repo, but the original branch. Thus, the “push request” need to be accompanied with the intended branch for the merging of changes as shown in Figure 14.

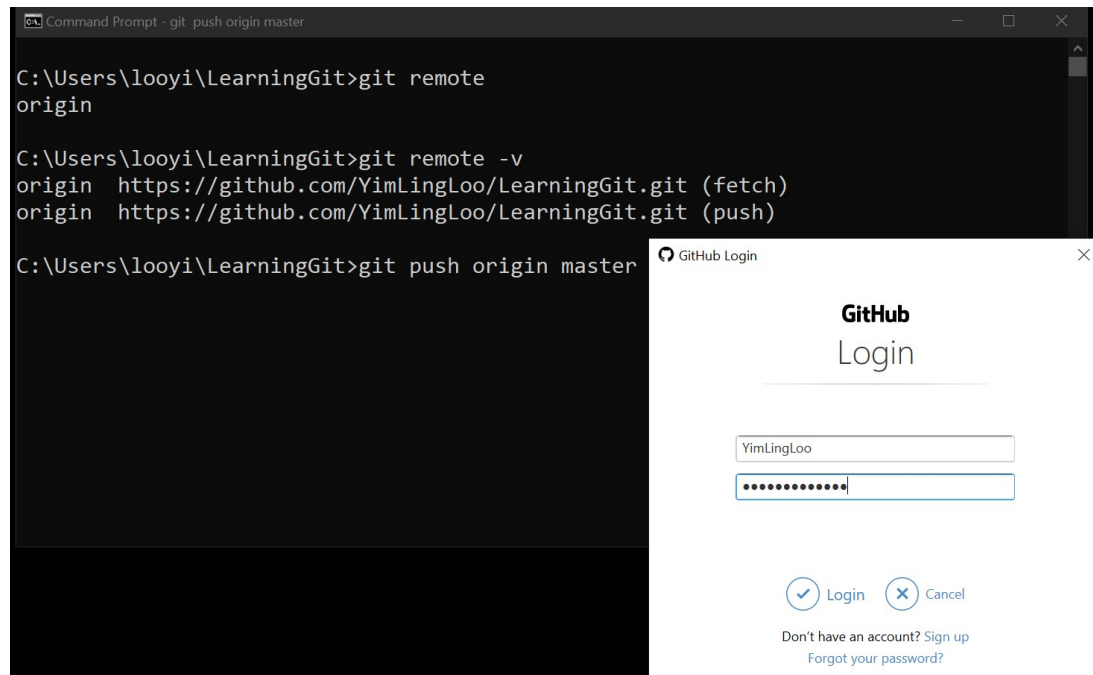


Figure 14: Git requests login to identify the user for correct authorization.

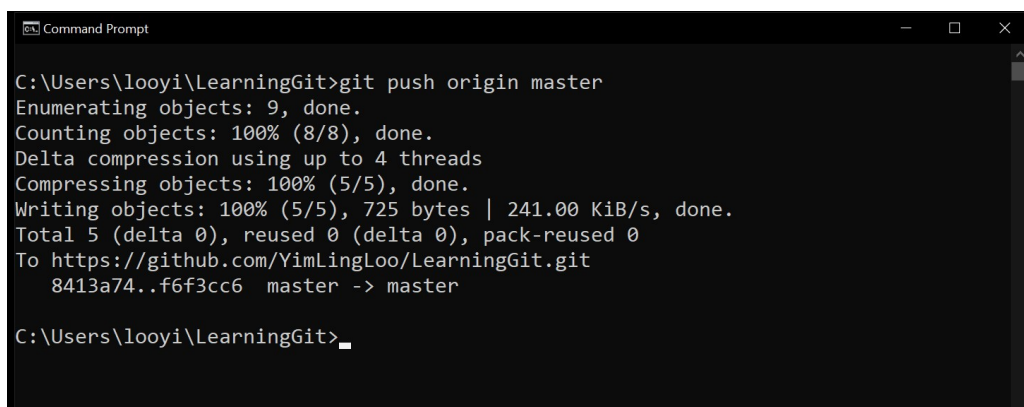


Figure 15: Successful push request displays log of changes made.

Double check the changes made with the original repo in Github and see that the GUI commands executed in CLI really did the operations that are supposed to be as illustrated in Figure 16.

UECS2363 SOFTWARE CONSTRUCTION AND CONFIGURATION

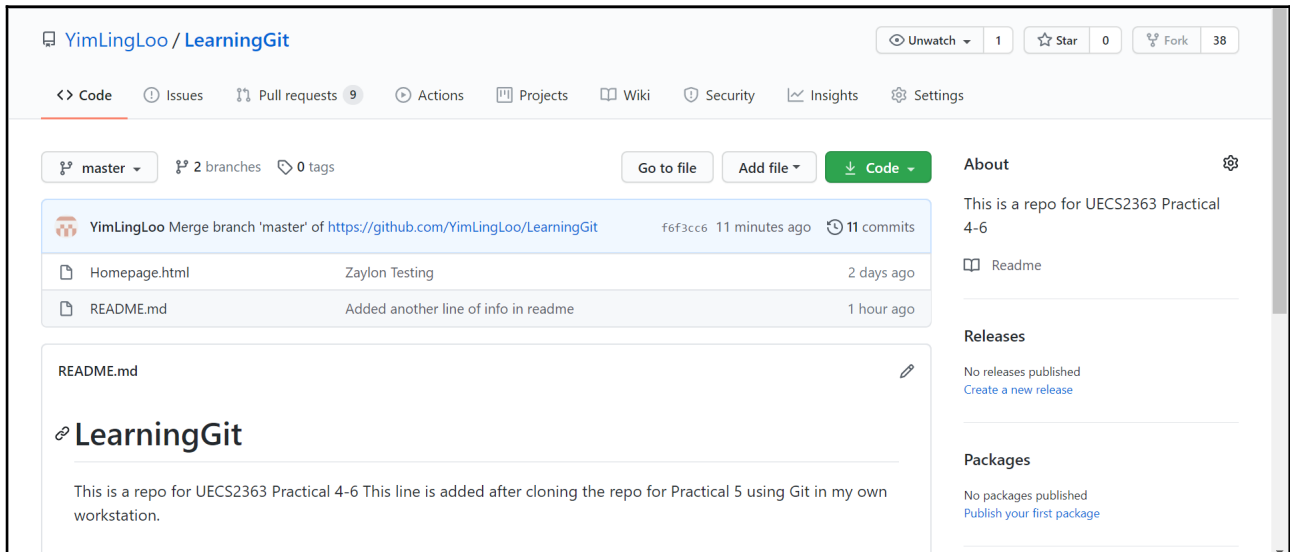


Figure 16: Changes made in the original repo.

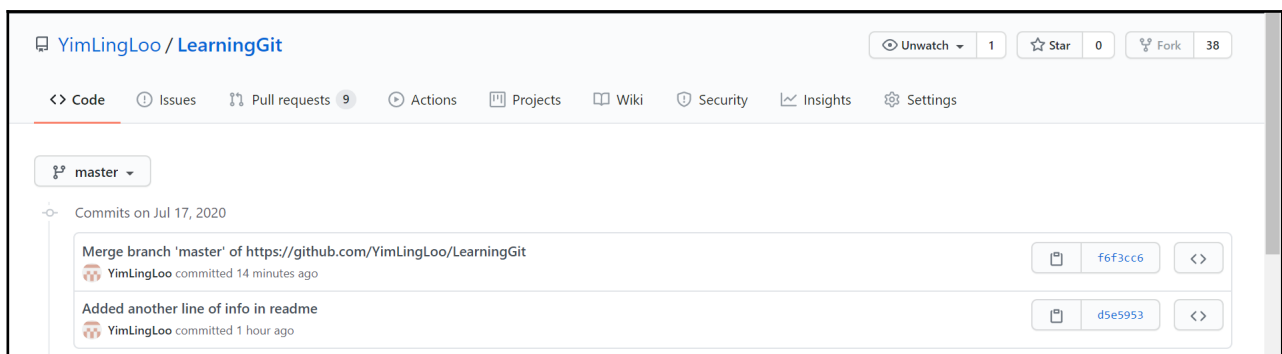


Figure 17: Commit log in Github.

Create a New Repo.

Refresh: There are two ways to have a repo locally on own workstation; one is by cloning a repo from Github and another is by initializing an empty repo by executing “git init” command. In this section, we explore on initializing a fresh repo in local workstation using Git.

Within the workstation, choose a place or folder that contains software project files which is to be made a repo. Change to the intended directory and execute “git init” within that folder as shown in Figure 18.

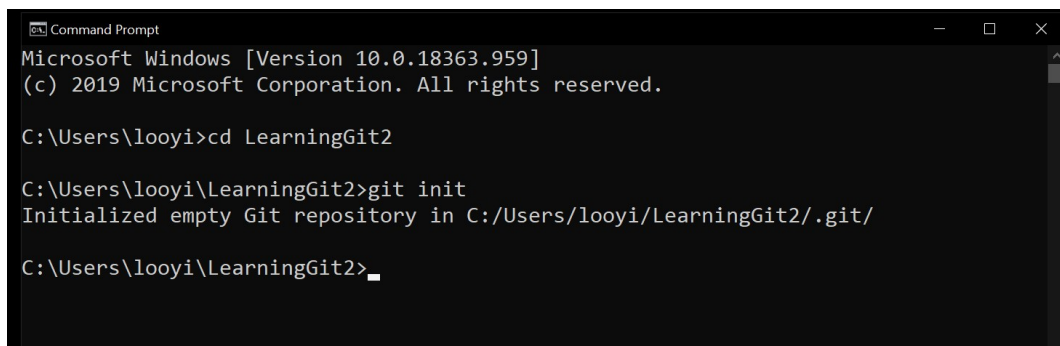
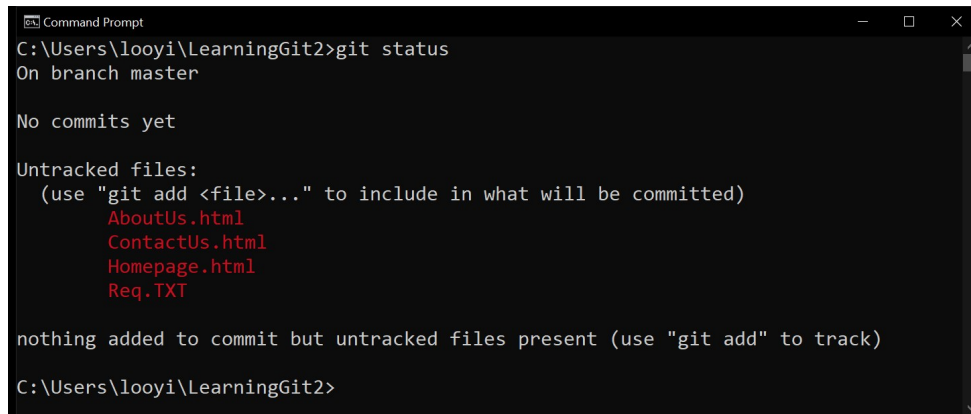


Figure 18: Initialize software project folder to be Git repo.

After the initialization, remember the workflow of Git: always execute “git status” to monitor the repo.



```
Command Prompt
C:\Users\looyi\LearningGit2>git status
On branch master

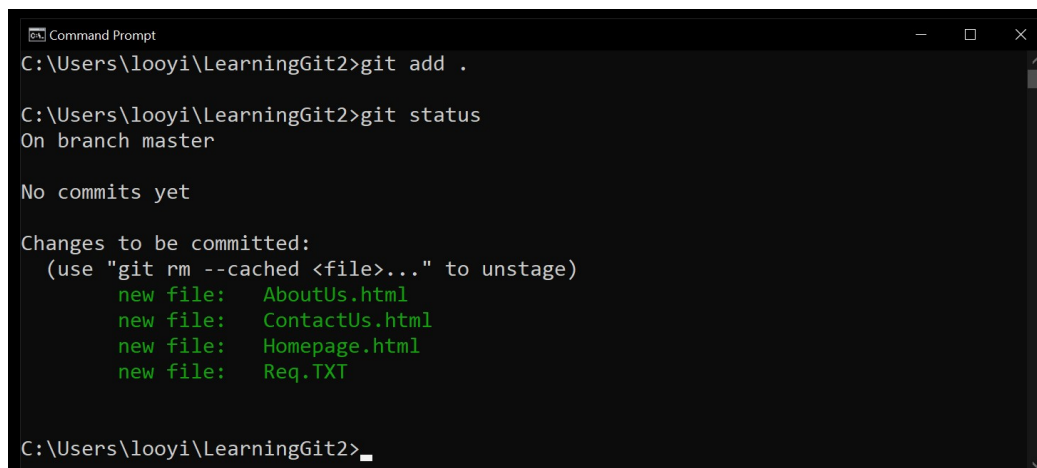
No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        AboutUs.html
        ContactUs.html
        Homepage.html
        Req.TXT

nothing added to commit but untracked files present (use "git add" to track)
C:\Users\looyi\LearningGit2>
```

Figure 19: Status of initialized repo.

Take note that creating a repo out of a software project folder that contains software project files does not imply that the existing files will be automatically committed as files that should exist in the repo. Remember one need to “commit new file” while creating a file in Github and “commit changes” while making changes to an existing file. Since the files listed in red are not actually “existing” files in the repo, “git add” need to be executed in order to add the files into the repository as shown in Figure 20.



```
Command Prompt
C:\Users\looyi\LearningGit2>git add .
C:\Users\looyi\LearningGit2>git status
On branch master

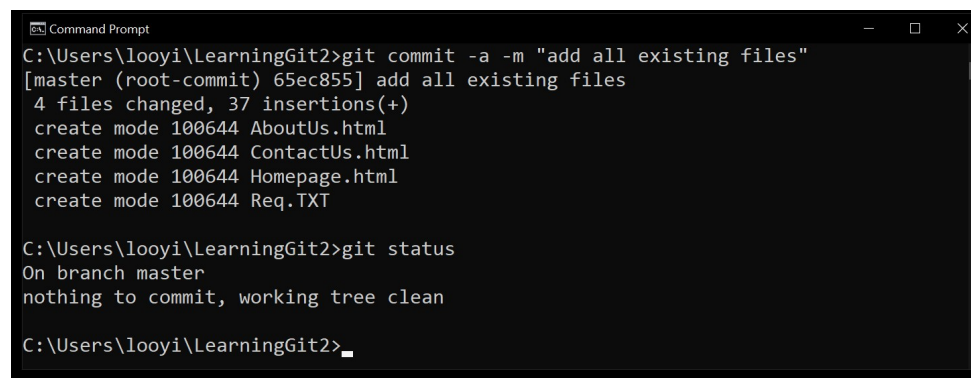
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   AboutUs.html
        new file:   ContactUs.html
        new file:   Homepage.html
        new file:   Req.TXT

C:\Users\looyi\LearningGit2>
```

Figure 20: Added files are listed/allowed to be committed.

Remember that one has to “commit” the add of new file in order to save or register the files into the repository. Thus, “git commit -a -m [“message”]” need to be executed to commit the new files as illustrated in Figure 21.



```
Command Prompt
C:\Users\looyi\LearningGit2>git commit -a -m "add all existing files"
[master (root-commit) 65ec855] add all existing files
4 files changed, 37 insertions(+)
create mode 100644 AboutUs.html
create mode 100644 ContactUs.html
create mode 100644 Homepage.html
create mode 100644 Req.TXT

C:\Users\looyi\LearningGit2>git status
On branch master

nothing to commit, working tree clean

C:\Users\looyi\LearningGit2>
```

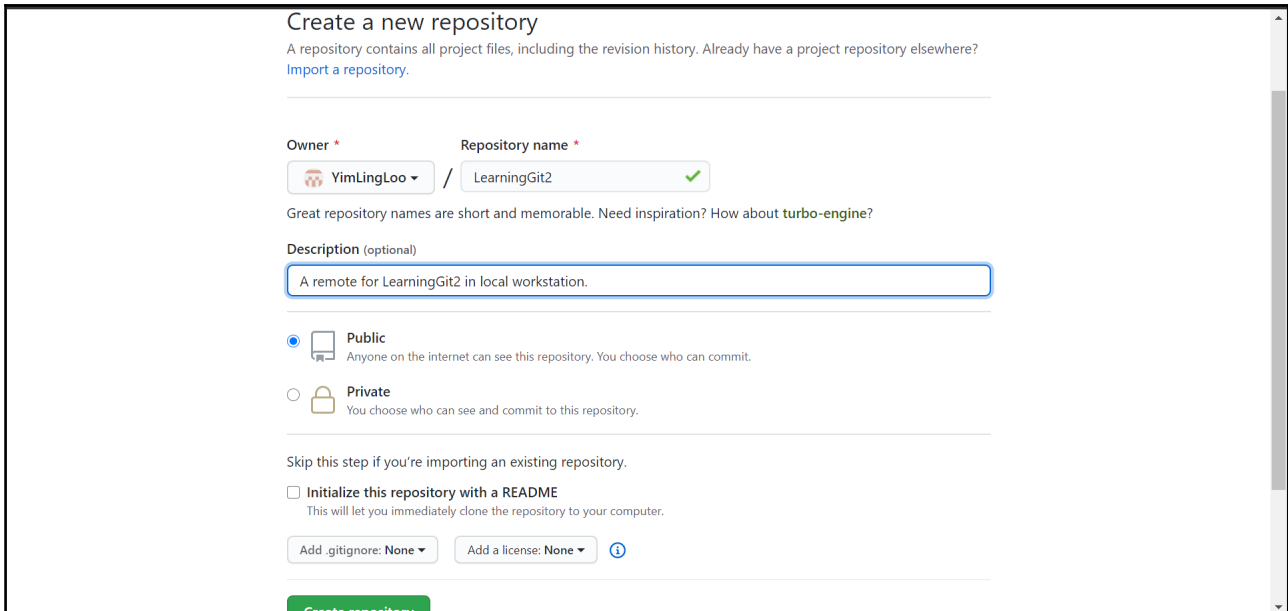
Figure 21: Existing files committed into repo.

UECS2363 SOFTWARE CONSTRUCTION AND CONFIGURATION

Keeping the repo in local workstation will not do good unless the developer has no intention for collaboration. Thus, one may need to create a remote server to enable collaborators to find the repo and start contributing.

Add Remote Control.

First, one need to create an empty repo in Github that reflects the same repo name as the repo in local workstation as shown in Figure 22.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * YimLingLoo / Repository name * LearningGit2 ✓

Great repository names are short and memorable. Need inspiration? How about [turbo-engine](#)?

Description (optional)

☒ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

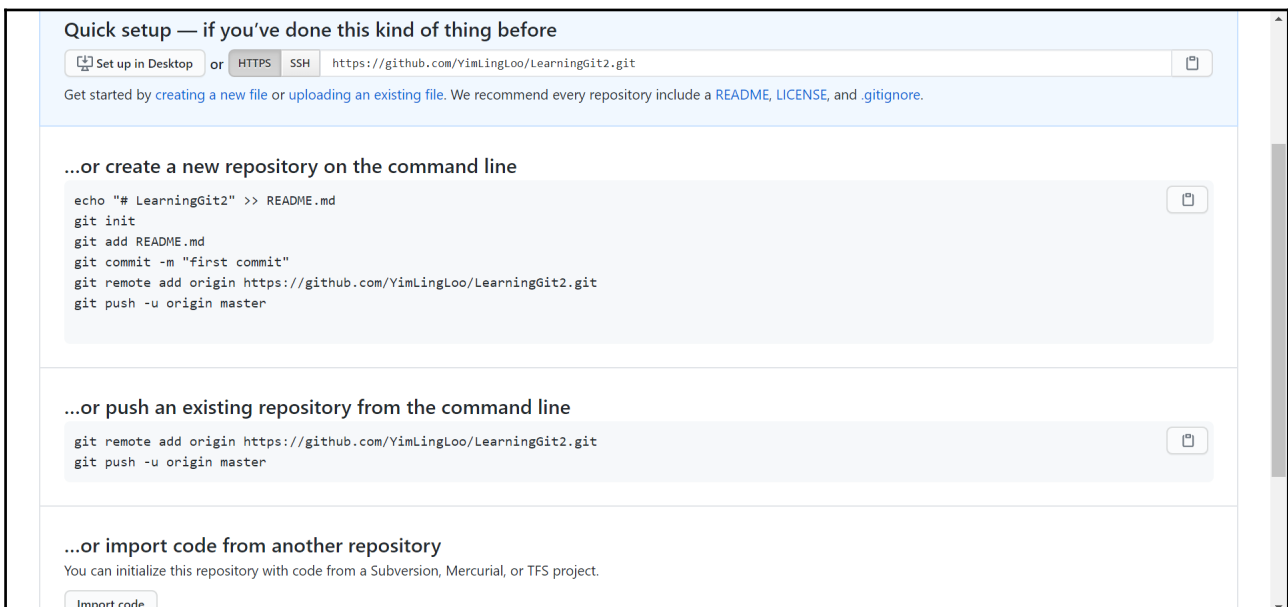
Skip this step if you're importing an existing repository.

☐ Initialize this repository with a README
This will let you immediately clone the repository to your computer.

Add .gitignore: None Add a license: None ⓘ

[Create repository](#)

Figure 22: Create new empty repo in remote server (Github).



Quick setup — if you've done this kind of thing before

[Set up in Desktop](#) or [HTTPS](#) [SSH](#) <https://github.com/YimLingLoo/LearningGit2.git>

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# LearningGit2" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/YimLingLoo/LearningGit2.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/YimLingLoo/LearningGit2.git
git push -u origin master
```

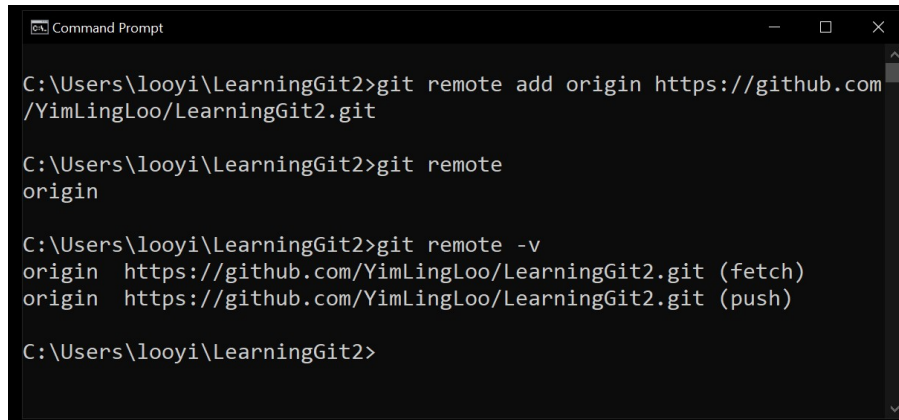
...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)

Figure 23: Remote server suggestion upon empty repo creation.

Extract the https location of the empty repo in remote server so that a remote can be added to the local workstation repo to this location as shown in Figure 24.



```

C:\Users\looyi\LearningGit2>git remote add origin https://github.com/YimLingLoo/LearningGit2.git

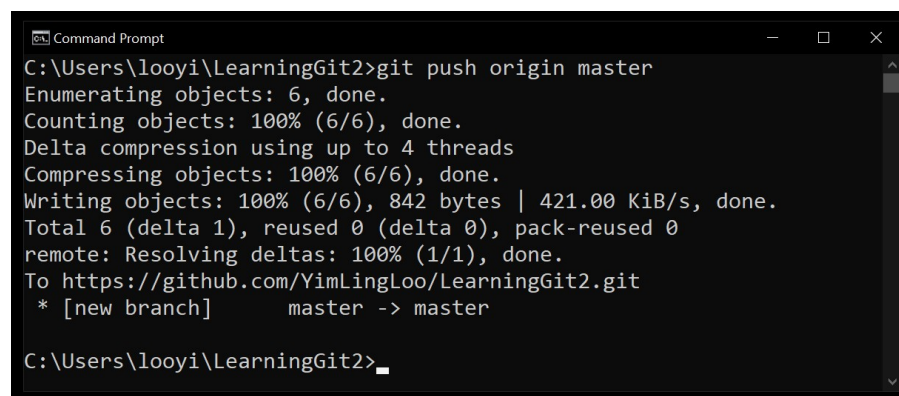
C:\Users\looyi\LearningGit2>git remote origin

C:\Users\looyi\LearningGit2>git remote -v
origin  https://github.com/YimLingLoo/LearningGit2.git (fetch)
origin  https://github.com/YimLingLoo/LearningGit2.git (push)

C:\Users\looyi\LearningGit2>
    
```

Figure 24: Adding remote to local repo.

Since remote is setup, in order to allow others to see the files that were committed into the local workstation repo previously, one may execute push request again as shown in Figure 25.



```

C:\Users\looyi\LearningGit2>git push origin master
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 842 bytes | 421.00 KiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/YimLingLoo/LearningGit2.git
 * [new branch]      master -> master

C:\Users\looyi\LearningGit2>
    
```

Figure 25: Push changes to repo in remote server.

Now, commit some changes in the repo of remote server (Github); simulating that one committed changes/contributed to the software project in the remote server as shown in Figure 26.

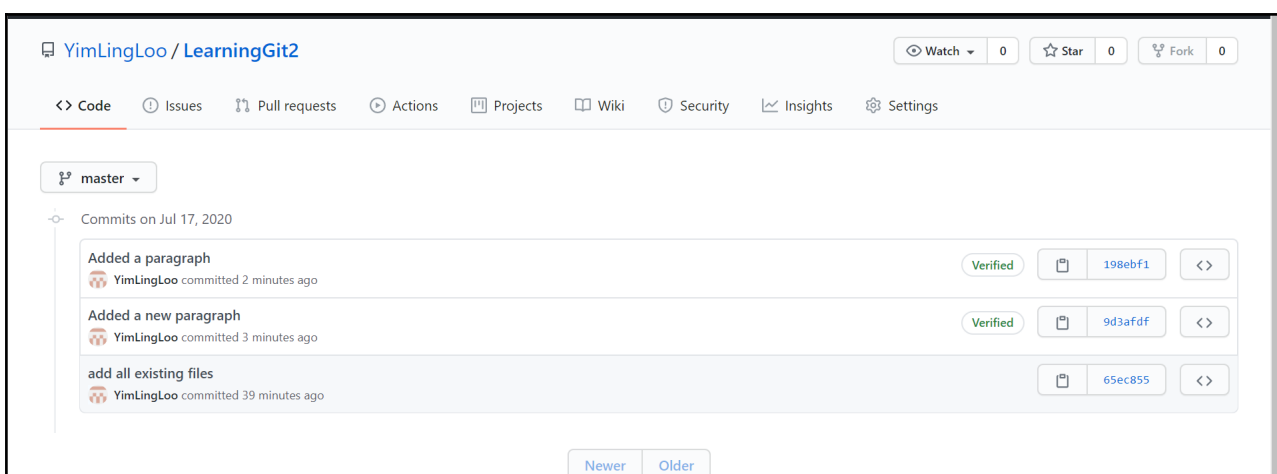
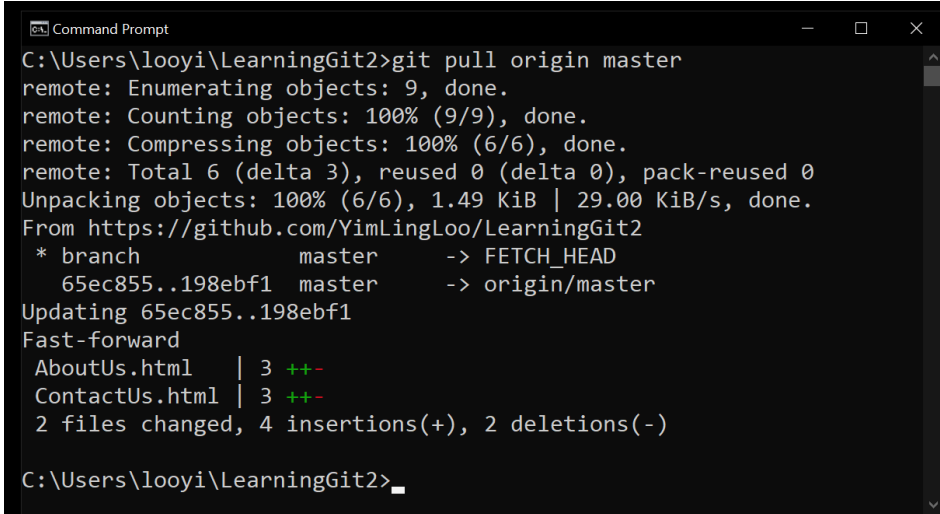


Figure 26: Commit changes to files in repo of remote server.

Now, what happens if the developer wants to see the changes/update the local repo with the contribution that was done in remote server?

A Pull Request.

Developer may always update the local workstation's repo through pulling the changes from remote server. Git command "git pull origin master" will enable one to update the repo with changes made in remote server as shown in Figure 27.



```

C:\Users\looyi\LearningGit2>git pull origin master
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 6 (delta 3), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (6/6), 1.49 KiB | 29.00 KiB/s, done.
From https://github.com/YimLingLoo/LearningGit2
* branch          master       -> FETCH_HEAD
   65ec855..198ebf1 master     -> origin/master
Updating 65ec855..198ebf1
Fast-forward
 AboutUs.html      | 3 ++-
 ContactUs.html    | 3 ++-
 2 files changed, 4 insertions(+), 2 deletions(-)

C:\Users\looyi\LearningGit2>

```

Figure 27: Update local repo with changes committed in remote server repo.

Summary

1. Git is a **version control software** that helps developer to keep track of the different versions of software project files created and edited. Git enables collaboration of many developers on a software project development.
2. Github is a web server with a website that runs Git software on the platform to enable developers to work on a software project and keep track of the versions.
3. **Repository** of files can be made for a software project in Github and made "public" so that other developers can **collaborate** in the development process.
4. "**Clone**" a repo from Github to local workstation so that one may work locally if preferred. In this way, Github becomes "**remote**" server while local workstation becomes local server.
5. Making changes to file(s) in the working directory of repo does not register the change in Git. A change need to be "**commit**"-ed in order to save the changes in Git.
6. Developer can choose to "**merge**" changes made in local server back into the original repo in remote server.
7. Merging involves "**push request**" which will push the registered changes from local server to original repo in remote server.
8. Developer may also "**init**"-ialize a fresh repo of software project files in local workstation using Git.
9. Existing files in project folder made repo need to be "**add**"-ed to be committed as files to exist in the repo.
10. A remote server (Github) can be added to local repo in order to enable distributed SCM.
11. Merging of changes in repo of remote server into repo of local server involves "**pull request**".