Lab #22. git and GitHub

- HGU CSEE Standard on assignments:
  - Submitting assignments or program codes written by others or acquired from the internet without explicit approval of the professor is regarded as cheating.
  - Showing or lending one's own homework to other student is also considered cheating that disturbs fair evaluation and hinders the academic achievement of the other student.
  - It is regarded as cheating if two or more students conduct their homework together and submit it individually when the homework is not a group assignment.
- When you report the lab results to the TA, s/he may ask your answers to the following questions.

Task 0. Preparation.

- Open the following references.
  - Git Tutorial (W3 Schools): https://www.w3schools.com/git/default.asp?remote=github
  - Git Cheat Sheet: https://education.github.com/git-cheat-sheet-education.pdf
  - GitHub Docs: https://docs.github.com/en

- Make sure you have a git client on your machine.
  - On Linux and MacOS, git is installed by default, *e.g.*,

```
$ git --version
git version 2.32.1 (Apple Git-133)
```

  - On Windows, open a command line prompt and type "`git --version`"
    - If your system has git, you will see a message like below:

```
git version 2.32.1.windows.1
```

    - If you see the below message, your system does not have git. You will need to visit https://git-scm.com/download/win, download and install a proper version of the software.

```
'git' is not recognized as an internal command,
operable program or batch file.
```

- Configuration: Add your identity.
  - Give the following commands (you only need to do this once):

```
$ git config --global user.name "YOUR NAME"
$ git config --global user.email ACCOUNT@DOMAIN.COM
```

- Create an account on GitHub.com.

Task 1. Creating a repository – EACH TEAM MEMBER SHOULD DO THIS TASK.

a) Required theory (quoted from *Li. A Tutorial for Git and GitHub. University of Zurich*).
  - Repository is your top-level working directory contains everything about your project.
    - The working directory probably contains many subdirectories – source code, binaries, documentation, data files, *etc*.
    - One of these subdirectories, named `.git`, is your repository.
  - At any time, you can take a "snapshot" of everything (or selected things) in your project directory, and put it in your repository
    - This "snapshot" is called a commit object.

  - The commit object contains (1) a set of files, (2) references to the "parents" of the commit object, and (3) a unique "SHA1" name.
    - *C.f.*, SHA1 hash function online - http://www.sha1-online.com/
  - Commit objects do not require huge amounts of memory.
- You can work as much as you like in your working directory, but the repository is not updated until you commit something.

b) Turn back to your initial Make project. Get your `main.c`, `mylib.c`, `mylib.h`, and `makefile`.

[main.c]

```
#include <stdio.h>
#include "mylib.h"

int main(void){

  int a = 3, b = 5;
  printf("(initial) a=%d, b=%d\n", a, b);

  swap(&a, &b);
  printf("(swapped) a=%d, b=%d\n", a, b);

  return 0;
}
```

[mylib.c]

```
#include "mylib.h"

void swap(int* a, int* b){
  int tmp = *a;
  *a = *b;
  *b = tmp;
}
```

[mylib.h]

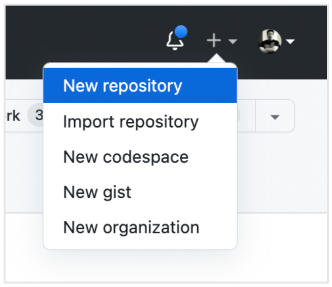
```
#ifndef  _MYLIB_H_
#define  _MYLIB_H_

void swap(int*, int*);

#endif
```

[makefile]

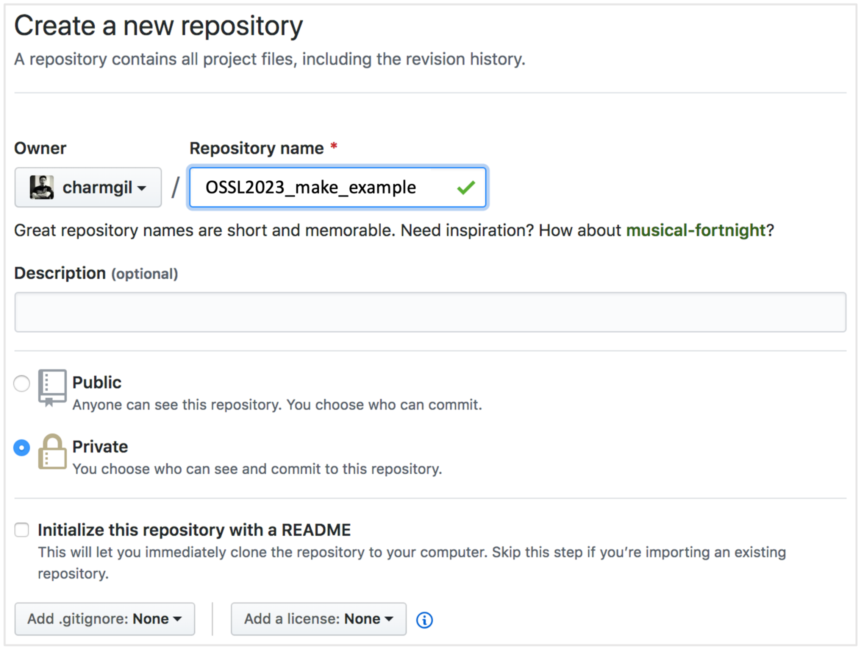
```
a.out: mylib.o main.o
  gcc -o a.out mylib.o main.o

mylib.o: mylib.c
  gcc -c -o mylib.o mylib.c

main.o: main.c
  gcc -c -o main.o main.c

clean:
  rm a.out mylib.o main.o
```

c) Create a new remote repository.

- On github.com, create a new repository by clicking on the + button (upper-right corner) and select "New repository".



- Give a basic information to the new repository. Name it on your own (it does not have to be OSSL2023_blah).



Create a new repository

A repository contains all project files, including the revision history.

Owner: charmgil / Repository name *: OSSL2023_make_example

Great repository names are short and memorable. Need inspiration? How about musical-fortnight?

Description (optional)

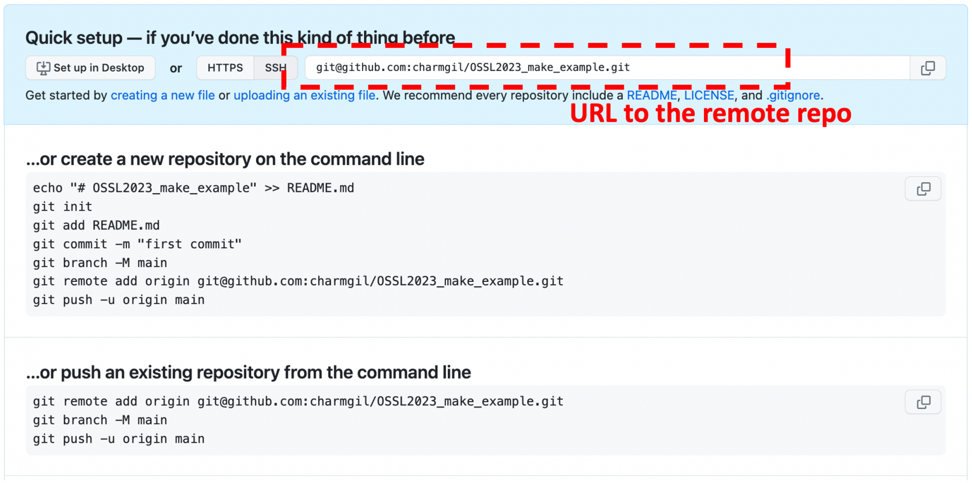
Public — Anyone can see this repository. You choose who can commit.

[✓ Private] — You choose who can see and commit to this repository.

Initialize this repository with a README — This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None

- Keep the URL to the remote repo.



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

`git@github.com:charmgil/OSSL2023_make_example.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 "# OSSL2023_make_example" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:charmgil/OSSL2023_make_example.git
git push -u origin main
```

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

```
git remote add origin git@github.com:charmgil/OSSL2023_make_example.git
git branch -M main
git push -u origin main
```

d) Init a local git repository.

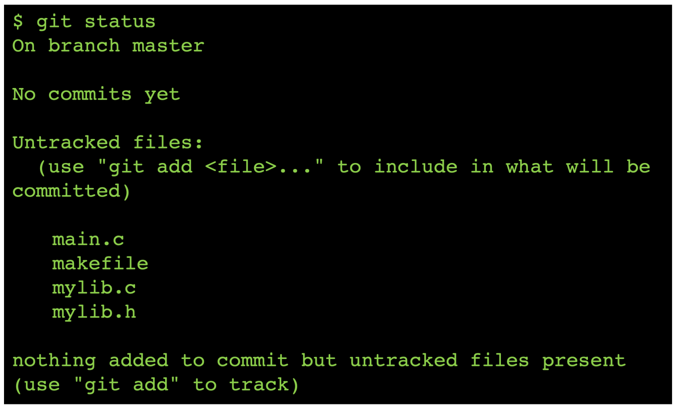
- When you said `git init` in your project directory, or when you cloned an existing project, you created a repository.
  - The repository is a subdirectory named `.git` containing various files.
  - The dot indicates a "hidden" directory.
  - You do NOT work directly with the contents of the `.git` directory; various git commands do that for you
- Go to the make project directory, command `git init` at the project directory.

```
$ ls
main.c    makefile    mylib.c    mylib.h
$ git init
```

```
Initialized empty Git repository in
/Users/charmgil/Projects/CODE.c/github_example/.git/
```

e) Check the status.

- Command `git status` to check the status of repository. Try to figure out what the status message means.



```
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be
committed)

    main.c
    makefile
    mylib.c
    mylib.h

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

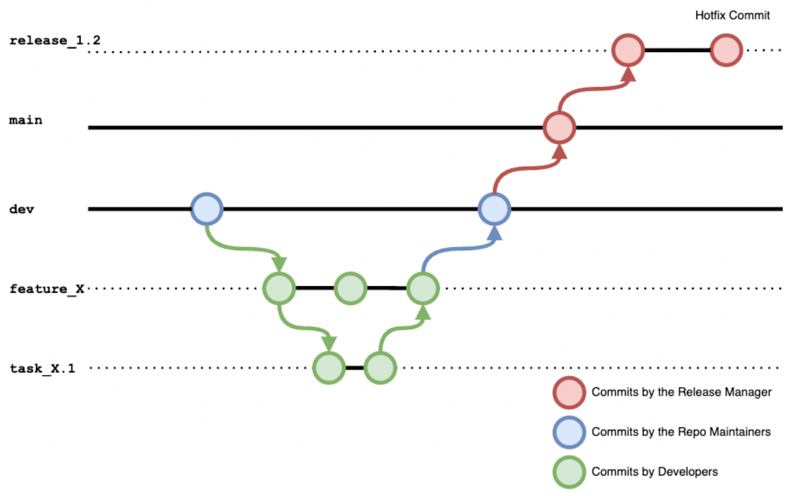
f) Make the first commit.

- You do your work in your project directory, as usual.
- If you create new files and/or folders, they are not tracked by git unless you ask it to do so.
  - The git command to start tracking working files:
    `git add newFile1 newFolder1 newFolder2 newFile2`
- Committing makes a "snapshot" of everything being tracked into your repository.
- A message telling what you have done is required.
  - Example: `git commit –m "fixed the incorrect precision issue"`
  - If one simply types in `git commit`, an editor is open for you to enter a message. To complete the commit, one needs to enter a message, save and close the editor.

- First, you need to add the project files to let git keeps track of your files.
  - Command `git add -A` or `git add .`; Then, all files at the current directory (`.`) are staged for git commit.
  - In case you want to unstage files, use `git reset`. For example,
    - `git reset -- README` unstages a file named README.
    - `git reset` unstages all your files.

- Commit the changes to your local git repository.
  - `git commit -m "first commit"`
  - When you commit your change to git, it creates a commit object.
    - A commit object represents the complete state of the project, including all the files in the project.
    - The very first commit object has no "parents".
    - Usually, you take some commit object, make some changes, and create a new commit object; the original commit object is the parent of the new commit object.
      - Hence, most commit objects have a single parent.
    - Sometimes, you need to merge two commit objects to form a new one.
      - Then, the new commit object has two parents.

- Consequently, commit objects forms a directed graph.
  - Git is all about using and manipulating this graph.



g) Associate the repositories.

- So far, we have created two repositories (one is on the local machine; and another is on GitHub.com). Now we would like to associate them.

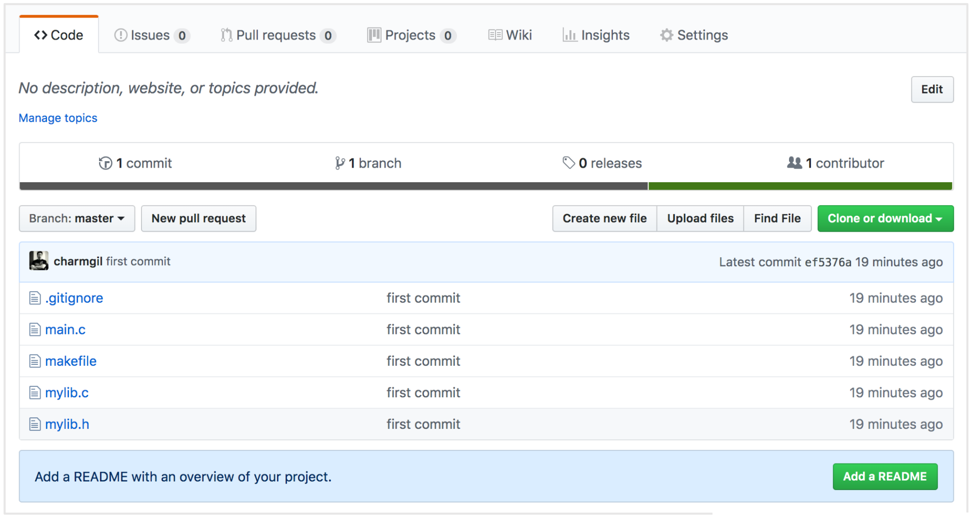
```
$ git remote add origin https://github.com/charmgil/remote_repo.git
```
→ This URL comes from step c.

- Then, push the content of your local repository (the `.git` directory) to the remote repository (the one on GitHub).

```
$ git push -u origin master
Counting objects: 7, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (7/7), 701 bytes | 701.00 KiB/s, done.
Total 7 (delta 0), reused 0 (delta 0)
To https://github.com/charmgil/OSS2020_make_example.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

h) You have completed Task 1: Your project is now on GitHub.

- Check out the remote repository on GitHub.com



No description, website, or topics provided.

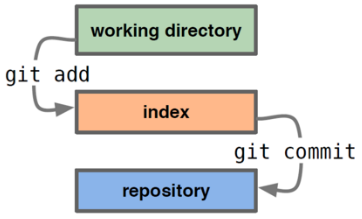
1 commit | 1 branch | 0 releases | 1 contributor

charmgil first commit — Latest commit ef5376a 19 minutes ago

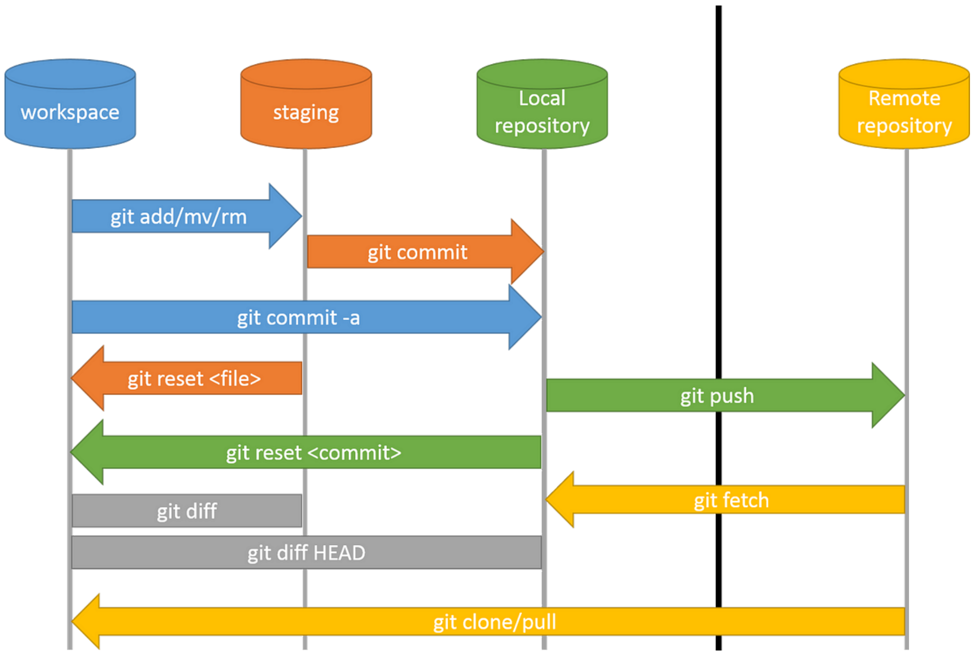
.gitignore	first commit	19 minutes ago
main.c	first commit	19 minutes ago
makefile	first commit	19 minutes ago
mylib.c	first commit	19 minutes ago
mylib.h	first commit	19 minutes ago

Add a README with an overview of your project.

- The steps and commands that you have work along with can be summarized as follows:
  - `git status`
  - (*Work on your files*)
  - `git add your editfiles`
  - `git commit –m “What I did”`

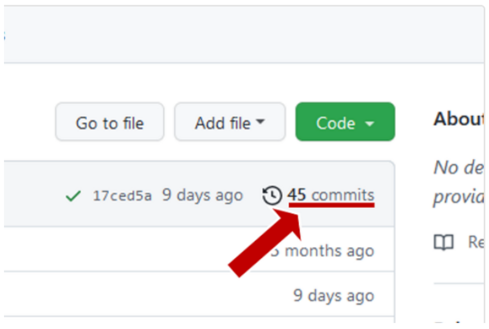


- For your reference, below diagram is a more detailed layout of the states and commands:



Task 2. Co-working via a GitHub repository – TEAM WORK ON ONE OF THE INDIVIDUAL REPOS.

a) Now each team has two individual repositories. On Task 2, both you and your teammate work as the internal team members in the project.
- Choose one of the two individual repositories that you created from Task 1, and add the other member as a contributor to the project.
  - How to: https://docs.github.com/en/account-and-profile/setting-up-and-managing-your-personal-account-on-github/managing-access-to-your-personal-repositories/inviting-collaborators-to-a-personal-repository (Give the **Write** permission to the teammate.)
- Extend the project and add new features to the project. More specifically,
  - **Student A**: Add new functions `int add(int a, int b)` and `int subtract(int a, int b)` to the project. You may want to implement this in `mylib.h` and `mylib.c`. Write a driver routine, for example,
```
    printf(“(added) a + b = %d + %d = %d\n”, a, b, add(a, b));
    printf(“(subtracted) a - b = %d - %d = %d\n”, a, b, subtract(a, b));
```
  in `main.c`. Finally, update `makefile`, accordingly.
  - **Student B**: Add new functions `int mult(int a, int b)` and `int div(int a, int b)` to the project. You may want to implement this in `mylib.h` and `mylib.c`. Make sure to properly handle the outcome datatype and avoid the divided-by-0 issue. Write a driver routine, for example,
```
    printf(“(multiplied) a x b = %d x %d = %d\n”, a, b, mult(a, b));
    printf(“(divided) a / b = %d / %d = %d\n”, a, b, div(a, b));
```
  in `main.c`. Finally, update `makefile`, accordingly.
- Combine all the progress from both students using GitHub. You are supposed to use the `git commit` and `git push` commands many times.
  - In git, commits are cheap. Do commits often.
  - One can check the commit history. Click on the “## commits” button as shown in below:



Task 3. Contributing to a GitHub repository – TEAM WORK ON ANOTHER INDIVIDUAL REPO.

a) On Task 3, you will contribute to the project as if you are not the internal member of the project.
- Work on the remaining repository (other than the one you worked with on Task 2).
- <u>Use the account that does not own the repository</u>.
- Fork the project to the account. (Again, use the account other than the project owner.)
  - How to: https://docs.github.com/en/get-started/quickstart/fork-a-repo
- Clone the forked project to your local machine.
  - How to: https://docs.github.com/en/repositories/creating-and-managing-repositories/cloning-a-repository
- **Both students**: Work together to receive the values of variables `a` and `b` from the user (in `main.c`).
- Commit and push the local changes to the remote repository (the forked one).

- **Student A** (owner of the forked project): Create a pull request to the original project.
  - How to: https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/creating-a-pull-request
- **Student B** (owner of the original project): Check out the pull request by Student A and merge the pull request.
  - How to: https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/incorporating-changes-from-a-pull-request/merging-a-pull-request

* On the post-lab tutor session on Tuesday, 23 May, each team is supposed to demonstrate:
- Both of (both students') the project repositories on GitHub.
- The final version of the project outcomes from Tasks 2 and 3.
  (One should be able to show and run the code.)
- The commit, pull request, and merge histories, using the Web interface of GitHub.