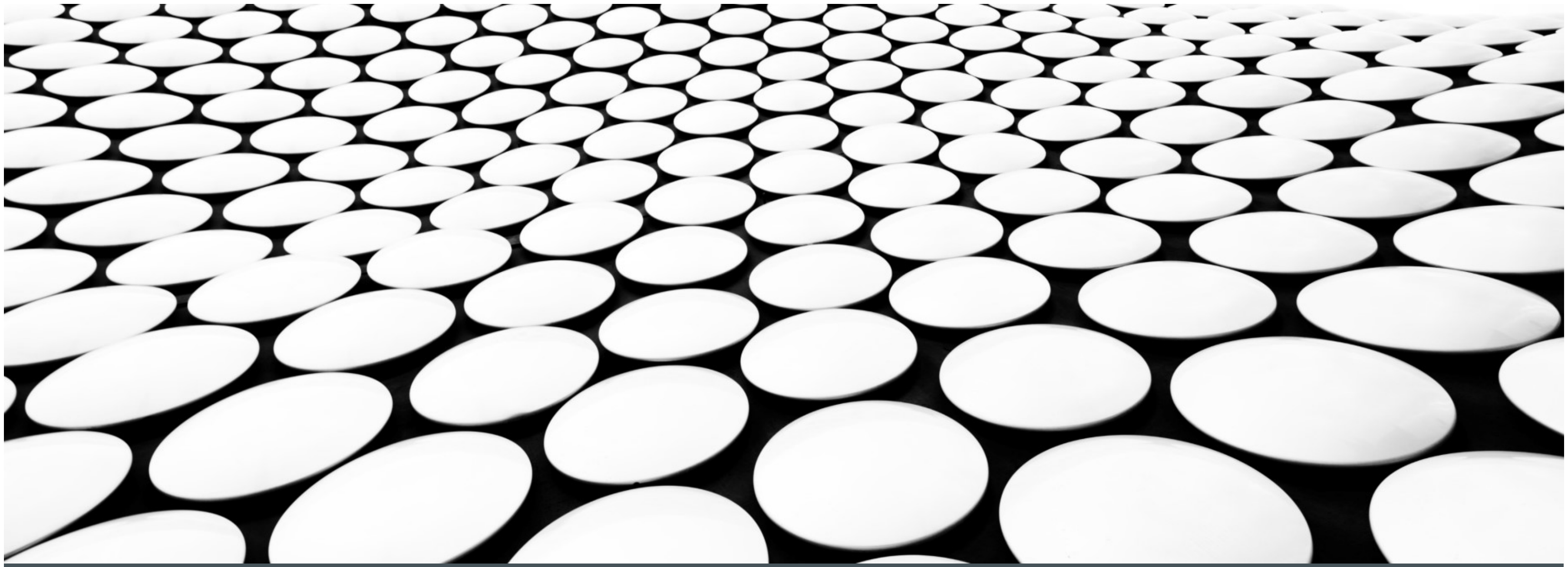
GOBLOCK 실무 프로젝트 교육과정

블럭체인 SW프로젝트의 '**체계적 협업 및 개발**'을 위한

GIT TRAINING



SYSCORE
System Security & Computer Engineering Research Lab.

GOAL OF THIS LECTURE NOTE

- **Git Training**

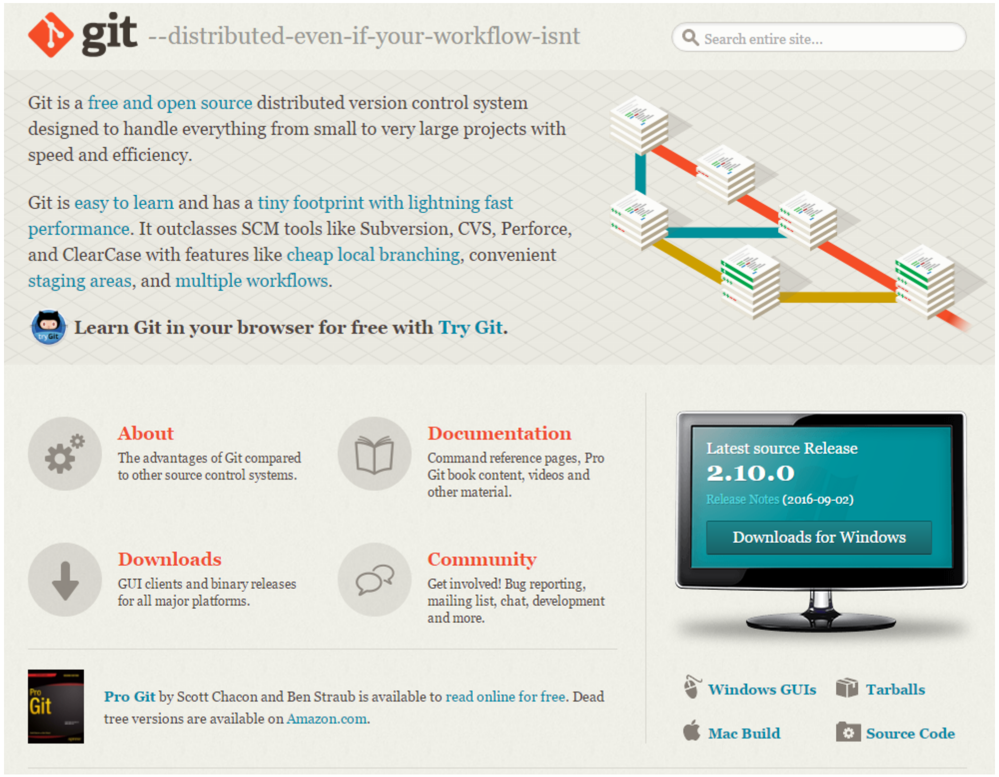
1. Motivation of version control system
2. Git History
3. Main Features of Git
4. SW Development Scenario with Git

SYSCORE
System Security & Computer Engineering Research Lab.

PREPARATION FOR GIT PRACTICE

1. Install git-scm

- https://git-scm.com/
- https://git-scm.com/downloads



git --distributed-even-if-your-workflow-isnt

Search entire site...

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

Learn Git in your browser for free with Try Git.

About The advantages of Git compared to other source control systems.

Documentation Command reference pages, Pro Git book content, videos and other material.

Downloads GUI clients and binary releases for all major platforms.

Community Get involved! Bug reporting, mailing list, chat, development and more.

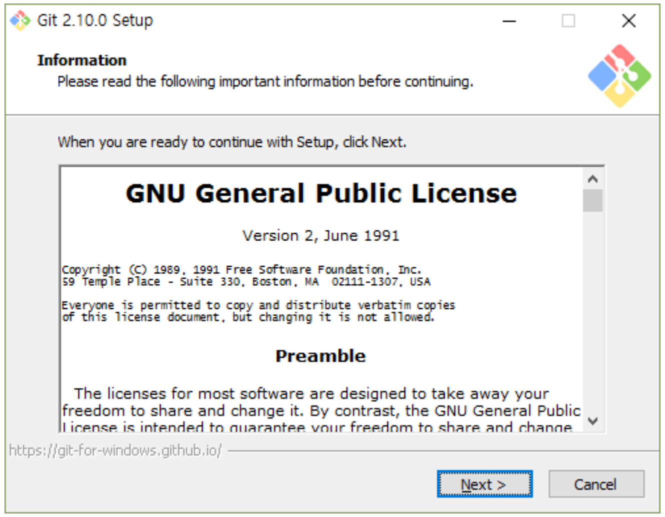
Latest source Release 2.10.0 Release Notes (2016-09-02)

Downloads for Windows

Windows GUIs · Tarballs · Mac Build · Source Code

Pro Git by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

INSTALLING GIT



Git 2.10.0 Setup

Information
Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

GNU General Public License

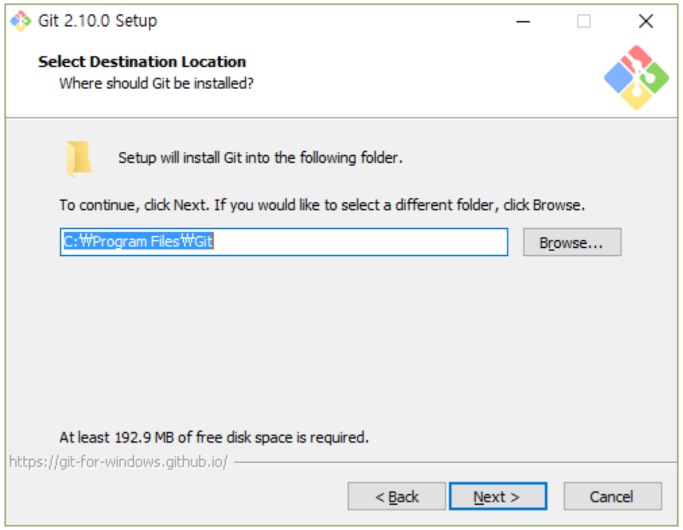
Version 2, June 1991

```
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA  02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
```

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change

https://git-for-windows.github.io/



Git 2.10.0 Setup

Select Destination Location
Where should Git be installed?

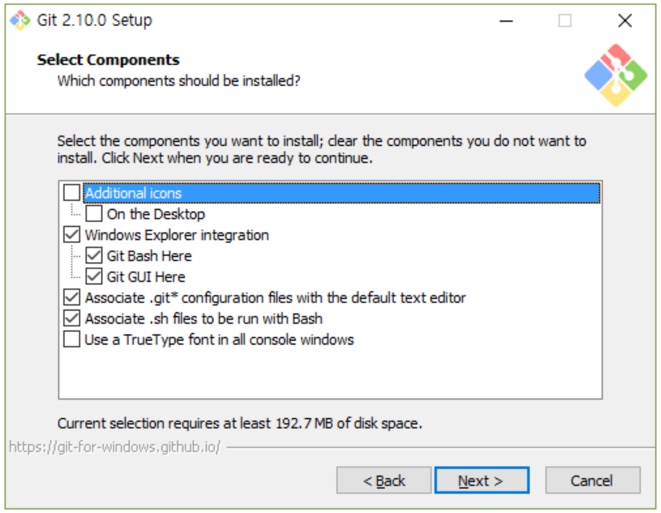
Setup will install Git into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:₩Program Files₩Git

At least 192.9 MB of free disk space is required.

https://git-for-windows.github.io/



Git 2.10.0 Setup

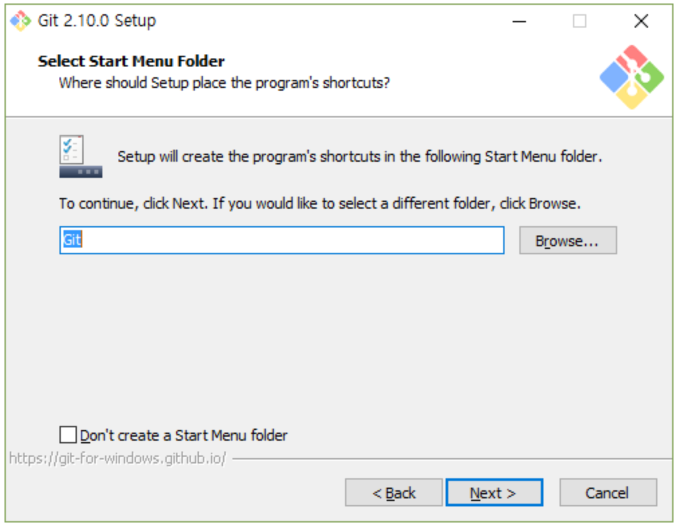
Select Components
Which components should be installed?

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- [ ] Additional icons
  - [ ] On the Desktop
- [x] Windows Explorer integration
  - [x] Git Bash Here
  - [x] Git GUI Here
- [x] Associate .git* configuration files with the default text editor
- [x] Associate .sh files to be run with Bash
- [ ] Use a TrueType font in all console windows

Current selection requires at least 192.7 MB of disk space.

https://git-for-windows.github.io/



Git 2.10.0 Setup

Select Start Menu Folder
Where should Setup place the program's shortcuts?

Setup will create the program's shortcuts in the following Start Menu folder.

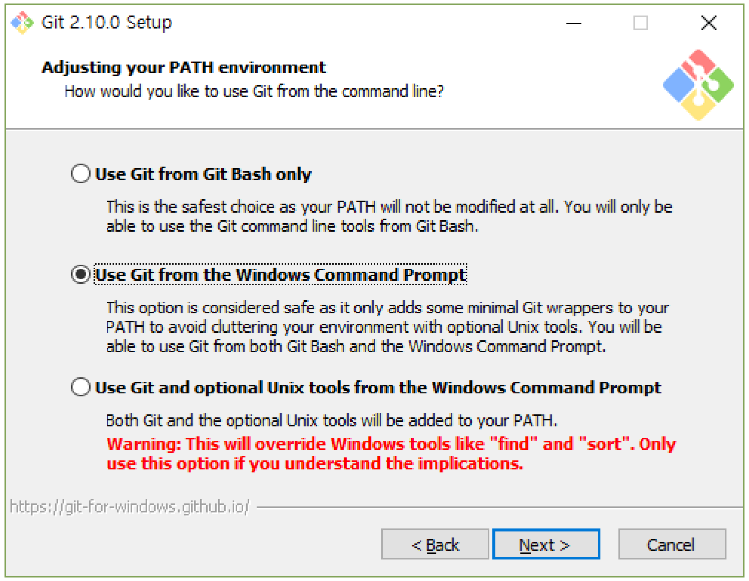
To continue, click Next. If you would like to select a different folder, click Browse.

Git

- [ ] Don't create a Start Menu folder

https://git-for-windows.github.io/

INSTALLING GIT

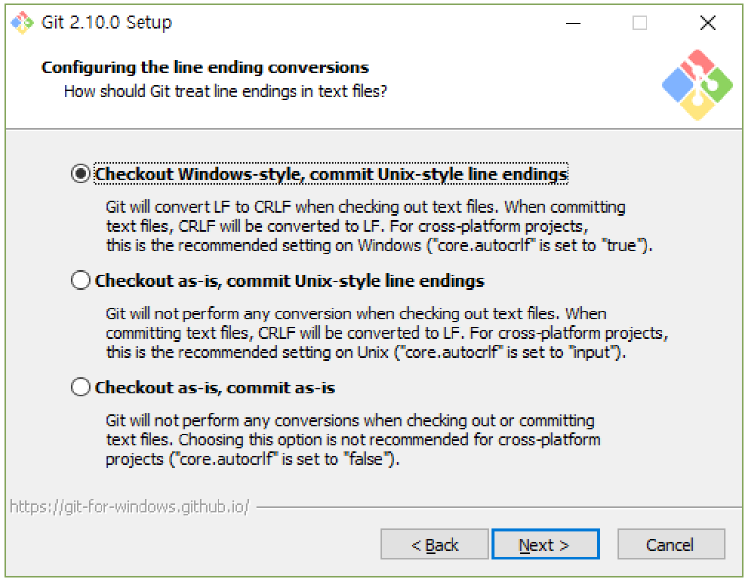

Git 2.10.0 Setup

Adjusting your PATH environment
How would you like to use Git from the command line?

( ) **Use Git from Git Bash only**
This is the safest choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.

[✓ **Use Git from the Windows Command Prompt**]
This option is considered safe as it only adds some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from both Git Bash and the Windows Command Prompt.

( ) **Use Git and optional Unix tools from the Windows Command Prompt**
Both Git and the optional Unix tools will be added to your PATH.
Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.

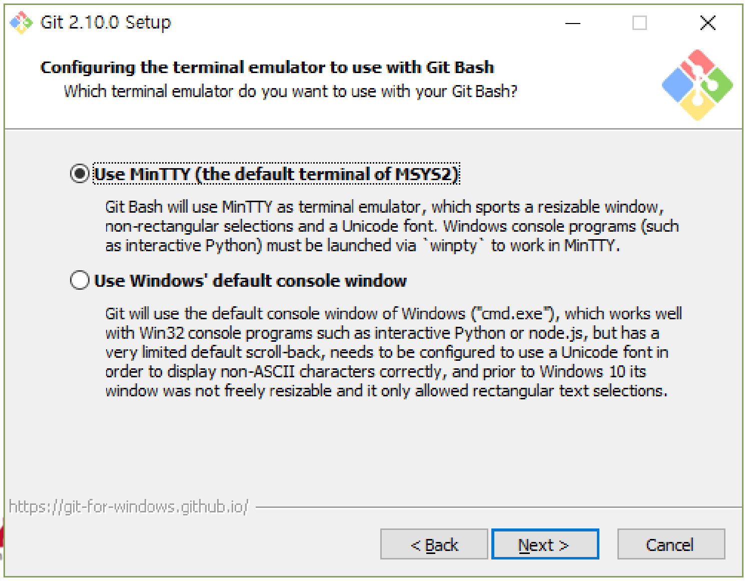

Git 2.10.0 Setup

Configuring the line ending conversions
How should Git treat line endings in text files?

[✓ **Checkout Windows-style, commit Unix-style line endings**]
Git will convert LF to CRLF when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Windows ("core.autocrlf" is set to "true").

( ) **Checkout as-is, commit Unix-style line endings**
Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").

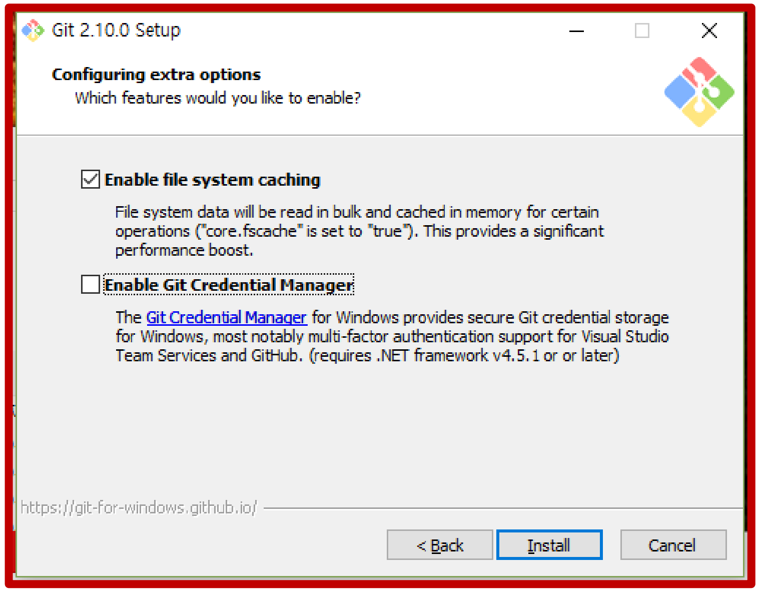
( ) **Checkout as-is, commit as-is**
Git will not perform any conversions when checking out or committing text files. Choosing this option is not recommended for cross-platform projects ("core.autocrlf" is set to "false").


Git 2.10.0 Setup

Configuring the terminal emulator to use with Git Bash
Which terminal emulator do you want to use with your Git Bash?

[✓ **Use MinTTY (the default terminal of MSYS2)**]
Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.

( ) **Use Windows' default console window**
Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.


Git 2.10.0 Setup

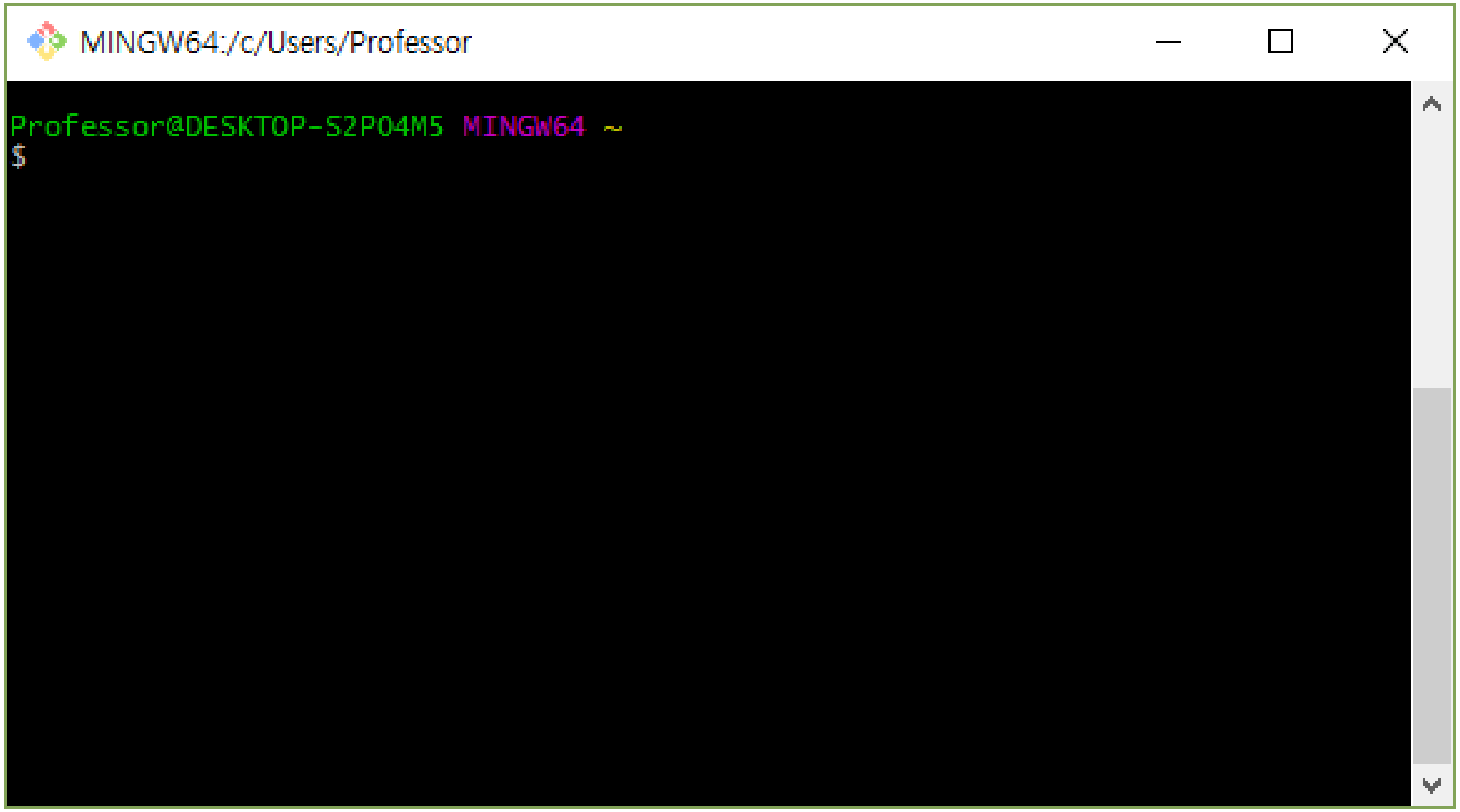
Configuring extra options
Which features would you like to enable?

[✓ **Enable file system caching**]
File system data will be read in bulk and cached in memory for certain operations ("core.fscache" is set to "true"). This provides a significant performance boost.

☐ **Enable Git Credential Manager**
The Git Credential Manager for Windows provides secure Git credential storage for Windows, most notably multi-factor authentication support for Visual Studio Team Services and GitHub. (requires .NET framework v4.5.1 or or later)

AFTER THE INSTALLATION OF GIT

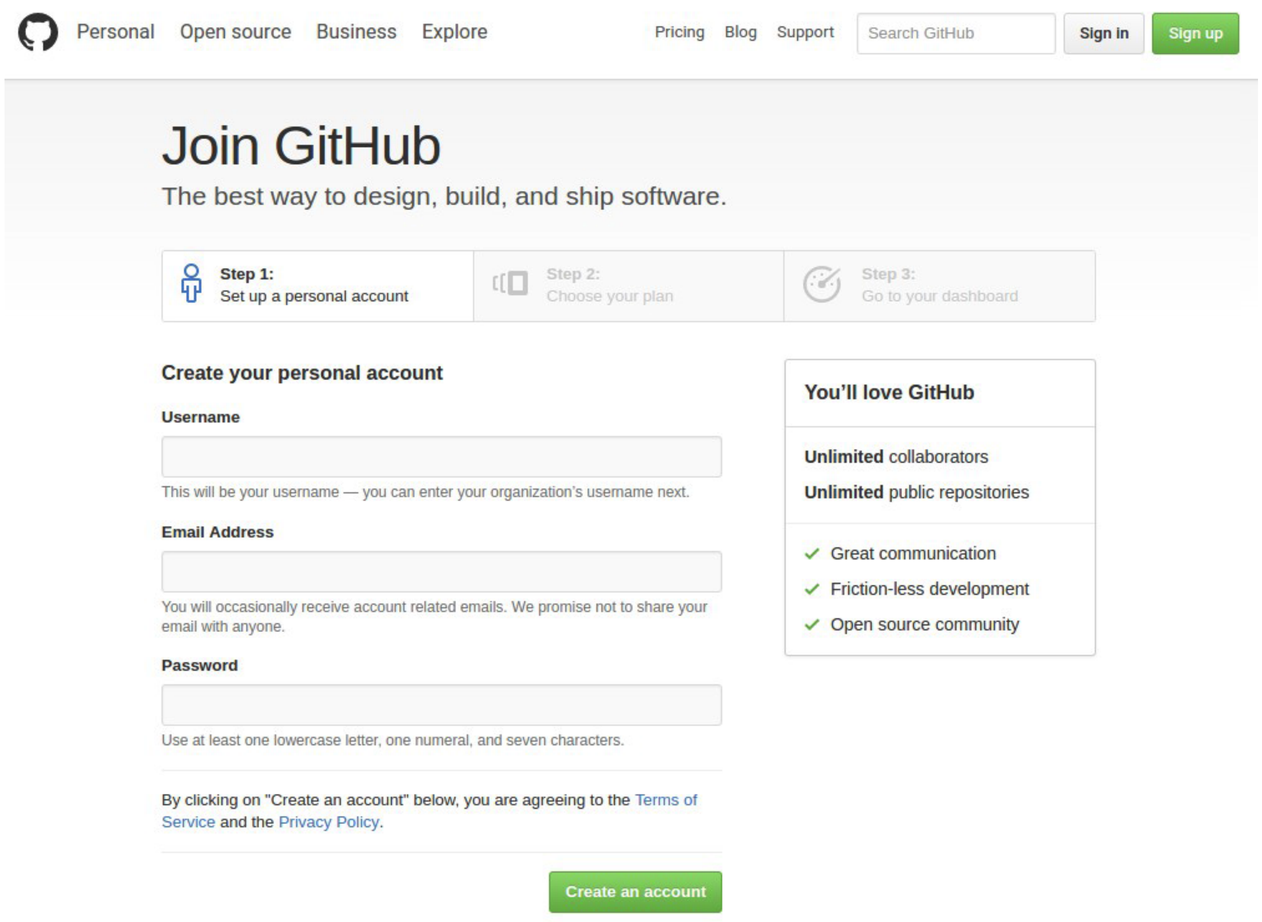
- Execution of git-bash



```
Professor@DESKTOP-S2P04M5 MINGW64 ~
$
```

PREPARATION FOR GIT PRACTICE

2. Make an account on GitHub



Join GitHub

The best way to design, build, and ship software.

Step 1: Set up a personal account

Step 2: Choose your plan

Step 3: Go to your dashboard

Create your personal account

Username

This will be your username — you can enter your organization's username next.

Email Address

You will occasionally receive account related emails. We promise not to share your email with anyone.

Password

Use at least one lowercase letter, one numeral, and seven characters.

By clicking on "Create an account" below, you are agreeing to the Terms of Service and the Privacy Policy.

Create an account

You'll love GitHub

Unlimited collaborators

Unlimited public repositories

✓ Great communication

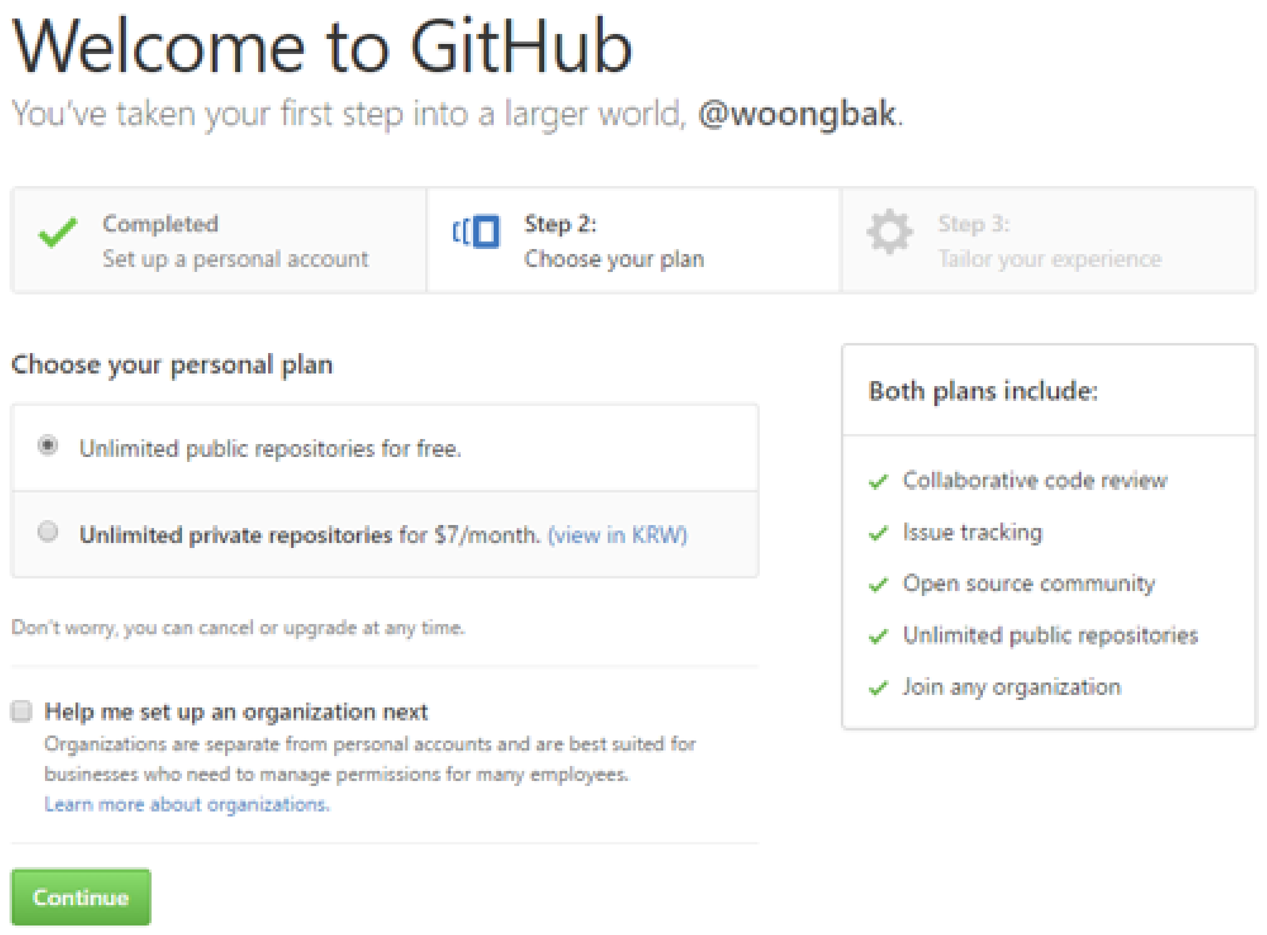
✓ Friction-less development

✓ Open source community

SYSCORE System Security & Computer Engineering Research Lab.

PREPARATION FOR GIT PRACTICE

2. Make an account on GitHub



Welcome to GitHub

You've taken your first step into a larger world, **@woongbak**.

✓ Completed
Set up a personal account

Step 2:
Choose your plan

Step 3:
Tailor your experience

Choose your personal plan

● Unlimited public repositories for free.

○ **Unlimited private repositories** for $7/month. (view in KRW)

Don't worry, you can cancel or upgrade at any time.

☐ **Help me set up an organization next**
Organizations are separate from personal accounts and are best suited for businesses who need to manage permissions for many employees.
Learn more about organizations.

Continue

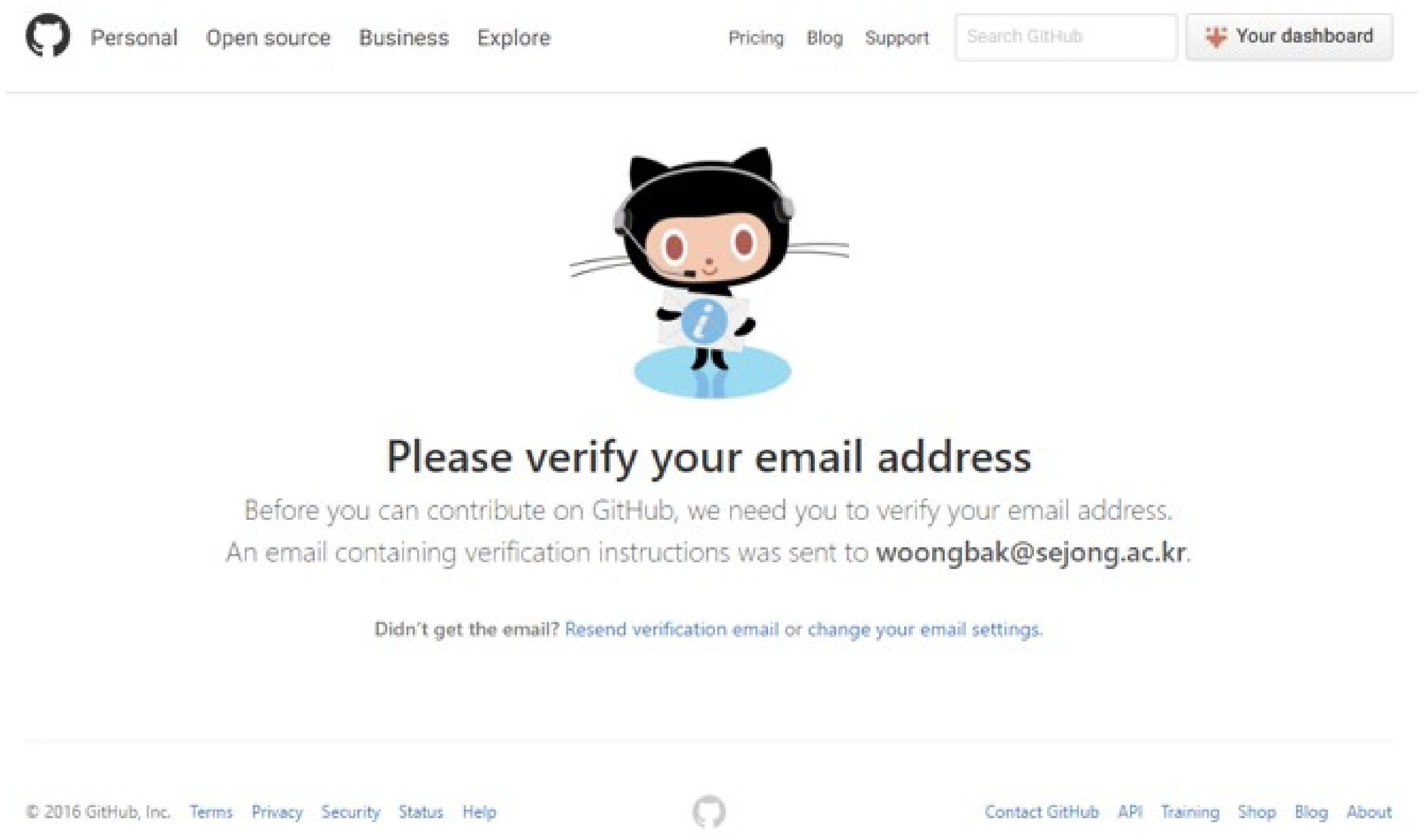
Both plans include:

- ✓ Collaborative code review
- ✓ Issue tracking
- ✓ Open source community
- ✓ Unlimited public repositories
- ✓ Join any organization

SYSCORE
System Security & Computer Engineering Research Lab.

PREPARATION FOR GIT PRACTICE

2. Make an account on GitHub



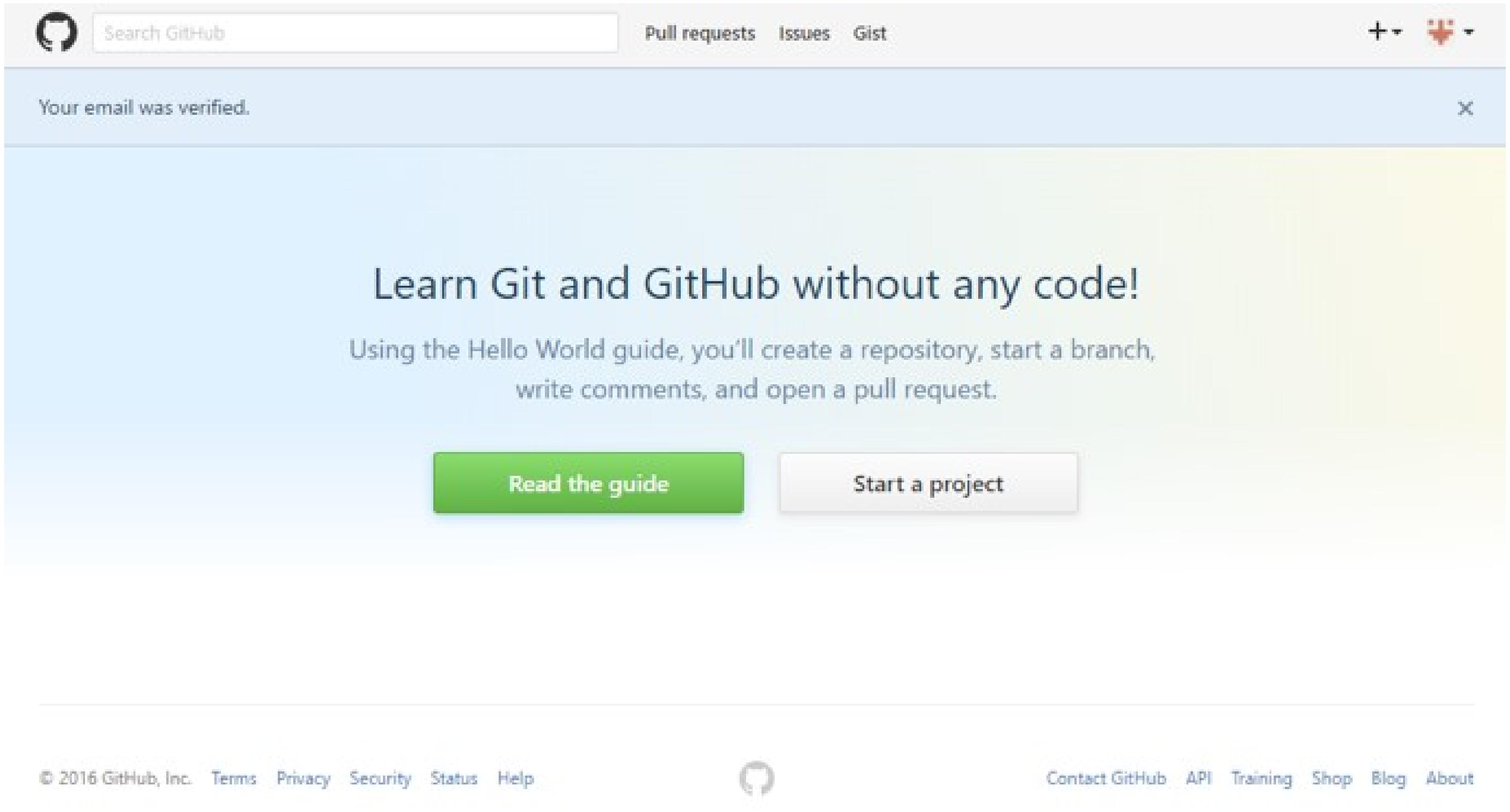
Please verify your email address

Before you can contribute on GitHub, we need you to verify your email address.
An email containing verification instructions was sent to **woongbak@sejong.ac.kr**.

Didn't get the email? Resend verification email or change your email settings.

PREPARATION FOR GIT PRACTICE

2. Make an account on GitHub



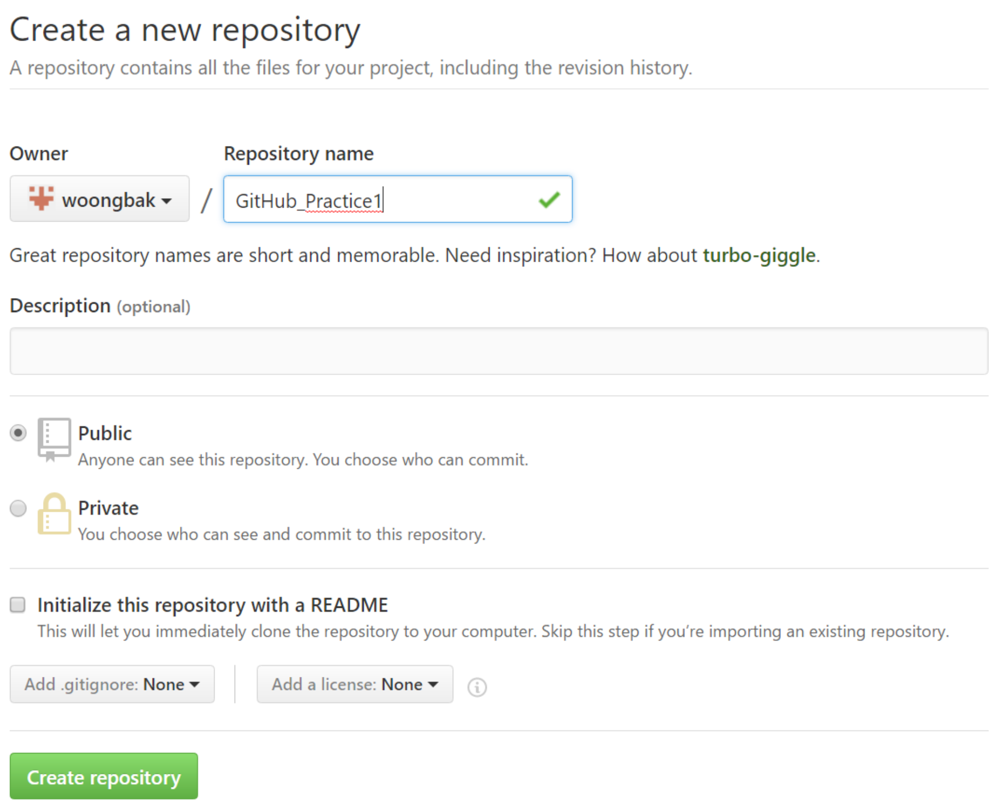
Your email was verified.

Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

PREPARATION FOR GIT PRACTICE

3. Create your own remote repository



Create a new repository

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

Owner: woongbak / Repository name: GitHub_Practice1 ✓

Great repository names are short and memorable. Need inspiration? How about **turbo-giggle**.

Description (optional)

[● Public] Anyone can see this repository. You choose who can commit.

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

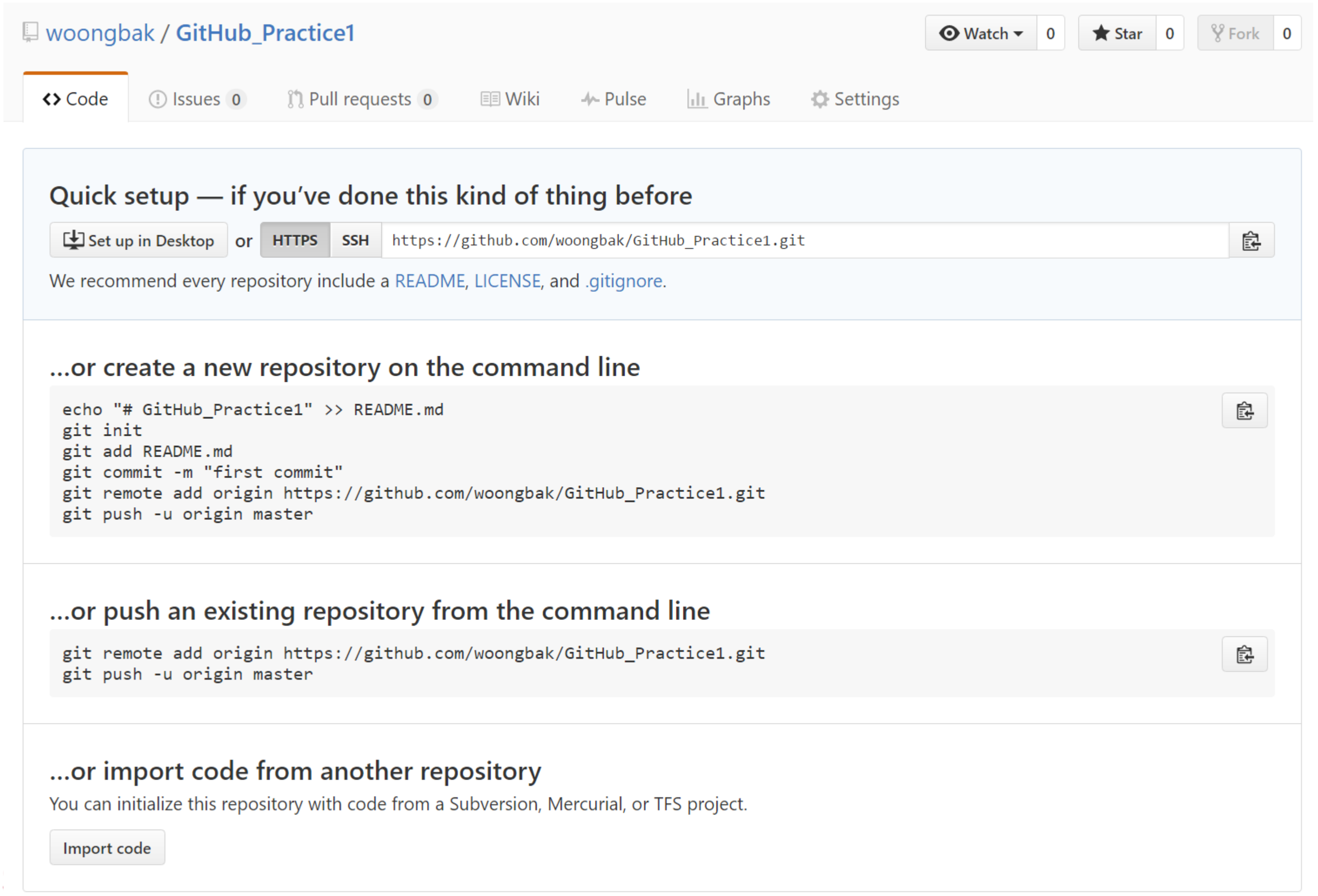
☐ Initialize this repository with a README

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

PREPARATION FOR GIT PRACTICE

3. Create your own remote repository



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

https://github.com/woongbak/GitHub_Practice1.git

We recommend every repository include a README, LICENSE, and .gitignore.

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

```
echo "# GitHub_Practice1" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/woongbak/GitHub_Practice1.git
git push -u origin master
```

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

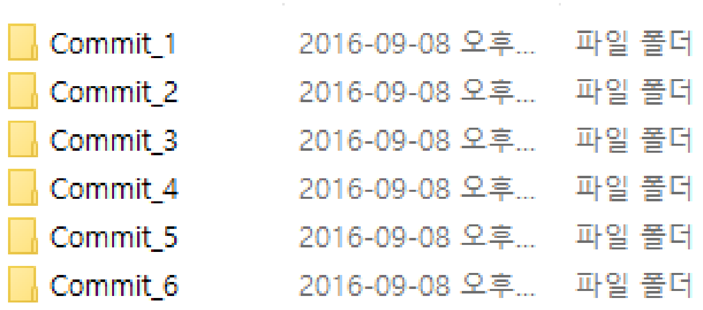
```
git remote add origin https://github.com/woongbak/GitHub_Practice1.git
git push -u origin master
```

...or import code from another repository

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

GET READY TO USE GIT!

1. **Download "git-practice1.zip" from kakaotalk**
2. **Unzip the file into a certain directory**
3. **Then you will see following folders**



Commit_1	2016-09-08 오후...	파일 폴더
Commit_2	2016-09-08 오후...	파일 폴더
Commit_3	2016-09-08 오후...	파일 폴더
Commit_4	2016-09-08 오후...	파일 폴더
Commit_5	2016-09-08 오후...	파일 폴더
Commit_6	2016-09-08 오후...	파일 폴더

4. **Coding: You can copy(ctrl+c) and paste(ctrl+v) file s in above directory into your git repository.**

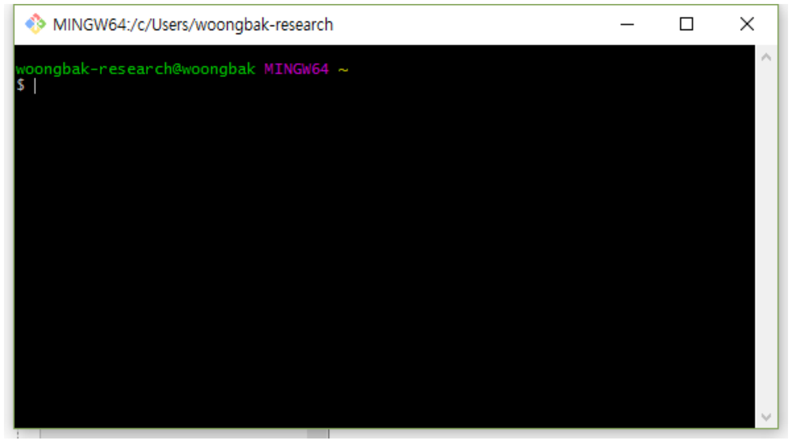
   But you need to type git-commands (next slide) using your key board.

GIT COMMANDS

command	description
git clone ***url [dir]***	copy a git repository so you can add to it
git add ***files***	adds file contents to the staging area
git commit	records a snapshot of the staging area
git status	view the status of your files in the working directory and staging area
git diff	shows diff of what is staged and what is modified but unstaged
git help ***[command]***	get help info about a particular command
git pull	fetch from a remote repo and try to merge into the current branch
git push	push your new branches and data to a remote repository
others: init, reset, branch, checkout, merge, log, tag	

BASIC GIT PRACTICE

5. Execute Git-bash


MINGW64:/c/Users/woongbak-research

```
woongbak-research@woongbak MINGW64 ~
$ |
```

* Default Location: C:₩users₩your_windows_account₩

6. Set the name and email for Git to use when you commit:

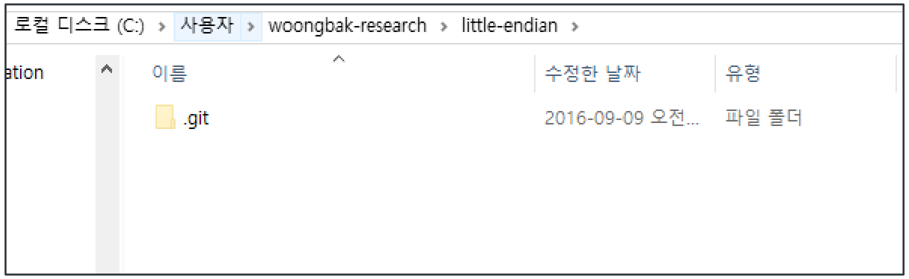
```
$ git config --global user.name “Ki-Woong Park”
$ git config --global user.email woongbak@sejong.ac.kr
```

- You can call **`git config --list`** to verify these are set.
- These will be set globally for all Git projects you work with.
- You can also set variables on a project-only basis by not using the **`--global`** flag.

SYSCORE System Security & Computer Engineering Research Lab.

BASIC GIT PRACTICE

Working directory


로컬 디스크 (C:) › 사용자 › woongbak-research › little-endian ›

이름	수정한 날짜	유형
.git	2016-09-09 오전...	파일 폴더

7. Creating folder for practice

```
$ mkdir little-endian
$ cd little-endian
```

8. To create a Git repo in your current directory:

```
$ git init
```

- This will create a .git directory in your current directory.

9. Copy “ReadMe.pdf” in “Commit-1” folder into “little-endian” folder

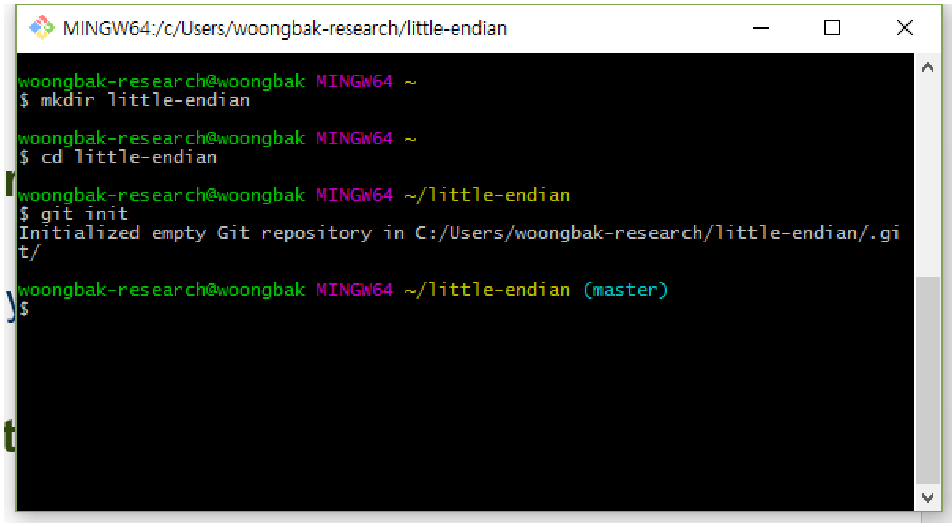
10. Commit the file in that directory into the repo:

```
$ git add ReadMe.pdf
$ git commit –m “little-endian: Add ReadMe file”
```

To check current condition

```
$ git show
$ git log
$ git diff
$ git status
```

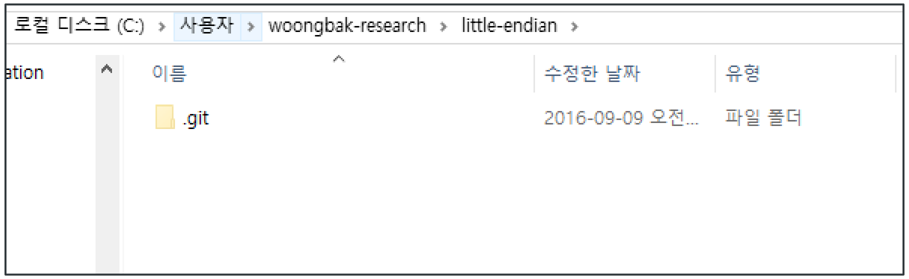
SYSCORE System Security & Computer Engineering Research Lab.



MINGW64:/c/Users/woongbak-research/little-endian

```
woongbak-research@woongbak MINGW64 ~
$ mkdir little-endian

woongbak-research@woongbak MINGW64 ~
$ cd little-endian

woongbak-research@woongbak MINGW64 ~/little-endian
$ git init
Initialized empty Git repository in C:/Users/woongbak-research/little-endian/.gi
t/

woongbak-research@woongbak MINGW64 ~/little-endian (master)
$
```

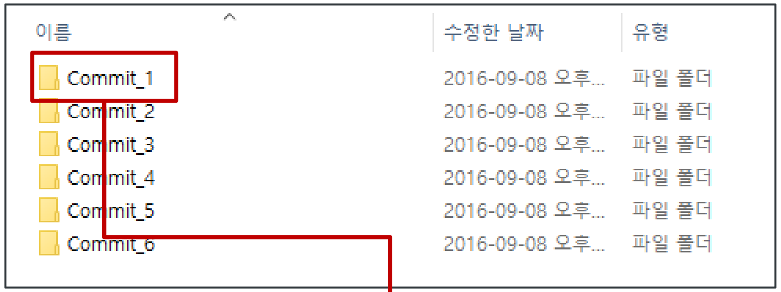
Working directory



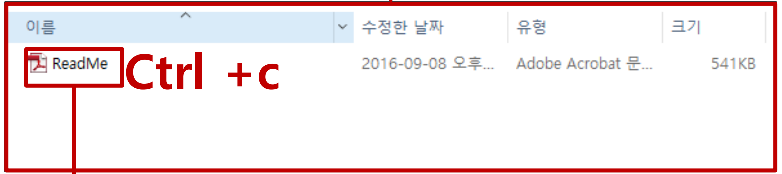
로컬 디스크 (C:) › 사용자 › woongbak-research › little-endian ›

이름	수정한 날짜	유형
.git	2016-09-09 오전...	파일 폴더

Unzipped directory from git-practice1.zip

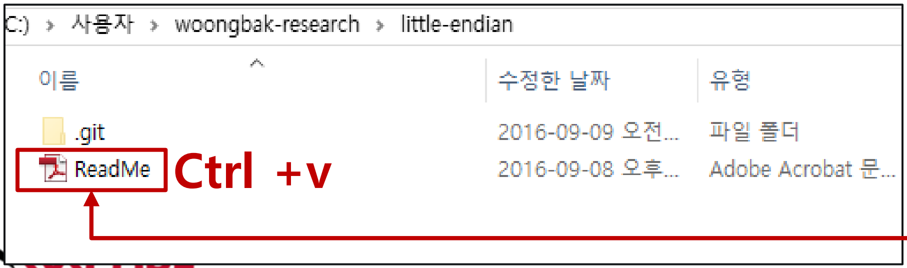


이름	수정한 날짜	유형
Commit_1	2016-09-08 오후...	파일 폴더
Commit_2	2016-09-08 오후...	파일 폴더
Commit_3	2016-09-08 오후...	파일 폴더
Commit_4	2016-09-08 오후...	파일 폴더
Commit_5	2016-09-08 오후...	파일 폴더
Commit_6	2016-09-08 오후...	파일 폴더



이름	수정한 날짜	유형	크기
ReadMe	2016-09-08 오후...	Adobe Acrobat 문...	541KB

Ctrl +c

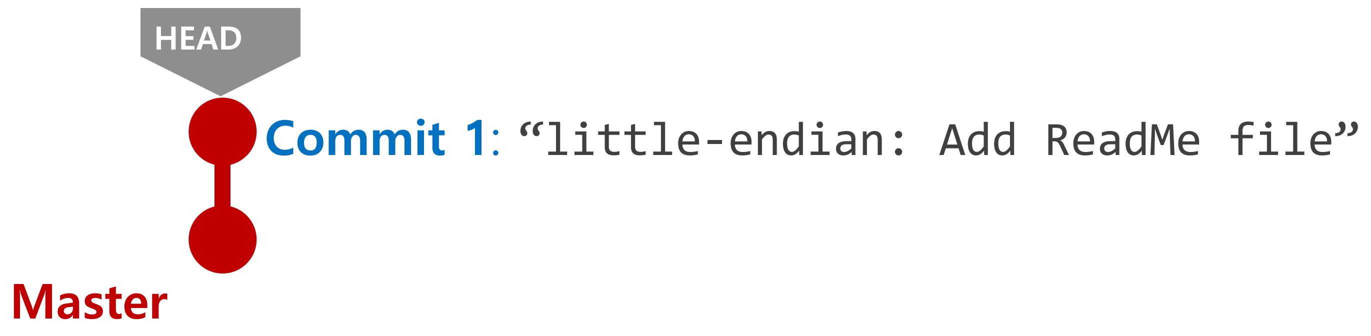


C:) › 사용자 › woongbak-research › little-endian

이름	수정한 날짜	유형
.git	2016-09-09 오전...	파일 폴더
ReadMe	2016-09-08 오후...	Adobe Acrobat 문...

Ctrl +v

OPERATIONS BOARD



<FOOTNOTE> LOCAL REPOSITORY

- **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
  - Commit objects do *not* require huge amounts of memory
- **You can work as much as you like in your working directory, but the repository isn't updated until you `commit` something**

<FOOTNOTE> INIT AND THE .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

<FOOTNOTE> COMMITTING FILES

- The first time we ask a file to be tracked, *and* every time before we commit a file we must add it to the staging area:

```
$ git add ReadMe.pdf
```

This takes a snapshot of these files at this point in time and adds it to the staging area.

- To move staged changes into the repo we commit:

```
$ git commit -m “little-endian: Add ReadMe file”
```

Note: To unstage a change on a file before you have committed it:

$ git reset HEAD *filename*

Note: To unmodify a modified file:

$ git checkout *filename*

Note: These commands are just acting on your local version of repo.

<FOOTNOTE> STATUS AND DIFF

- To view the status of your files in the working directory and st aging area:

```
$ git status
```
or
```
$ git status -s
```
(`-s` shows a short one line version)

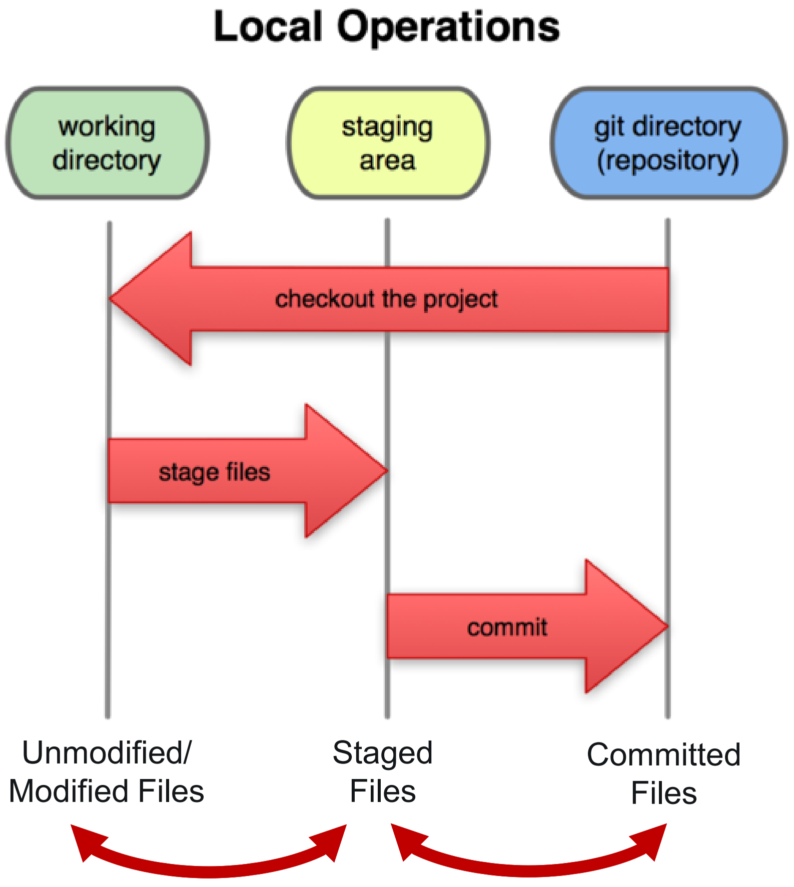
- To see what is modified but unstaged:

```
$ git diff
```

- To see staged changes:

```
$ git diff --cached
```

<FOOTNOTE> DIFF



```
$ git diff    $ git diff --cached
```

<FOOTNOTE> FOR EXAMPLE, AFTER EDITING A FILE…

```
$ vi git_test.txt
$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   git_test.txt
#
no changes added to commit (use "git add" and/or "git commit -a")
$ git status -s
 M git_test.txt                 ← Note: M is in second column = "working  tree"
$ git diff                      ← Shows modifications that have not been staged.
diff --git a/git_test.txt b/git_test.txt
index 66b293d..90b65fd 100644
--- a/git_test.txt
+++ b/git_test.txt
@@ -1,2 +1,4 @@
 Here is git_test's file.
+
+One new line added.
$ git diff --cached             ← Shows nothing, no modifications have been staged yet.
```

<FOOTNOTE> AFTER ADDING FILE TO STAGING AREA…

```
$ git add git_test.txt
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   git_test.txt
#
$ git status -s
M  git_test.txt           ← Note: M is in first column = "staging area"
$ git diff            ← Note: Shows nothing, no modifications that have not been staged.
$ git diff --cached           ← Note: Shows staged modifications.
diff --git a/git_test.txt b/git_test.txt
index 66b293d..90b65fd 100644
--- a/git_test.txt
+++ b/git_test.txt
@@ -1,2 +1,4 @@
 Here is git_test's file.
+
+One new line added.
```

SYSCORE
System Security & Computer Engineering Research Lab.

<FOOTNOTE> VIEWING LOGS

To see a log of all changes in your local repo:

- **`$ git log`** or
- **`$ git log --oneline`** (to show a shorter version)

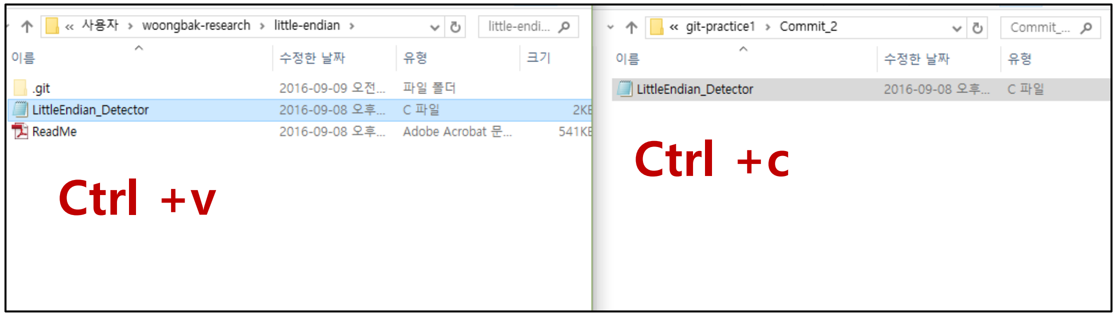
```
1677b2d Edited first line of readme
258efa7 Added line to readme
0e52da7 Initial commit
```

- **`git log -5`** **(to show only the 5 most recent updates, etc.)**

Note: changes will be listed by commitID #, (SHA-1 hash)

BASIC GIT PRACTICE

11. Copy “LittleEndian_Detector.c” in “Commit-2” folder into “little-endian” folder


« 사용자 › woongbak-research › little-endian ›
little-endi...
« git-practice1 › Commit_2
Commit_...
Ctrl +v
Ctrl +c

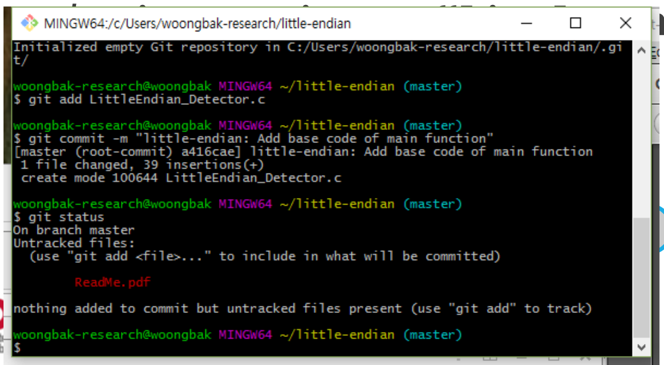
이름	수정한 날짜	유형	크기
.git	2016-09-09 오전...	파일 폴더	
LittleEndian_Detector	2016-09-08 오후...	C 파일	2KB
ReadMe	2016-09-08 오후...	Adobe Acrobat 문...	541KB

이름	수정한 날짜	유형
LittleEndian_Detector	2016-09-08 오후...	C 파일

12. Commit the file in that directory into the repo:

```
$ git add LittleEndian_Detector.c
```

```
ndian: Add base code of
```

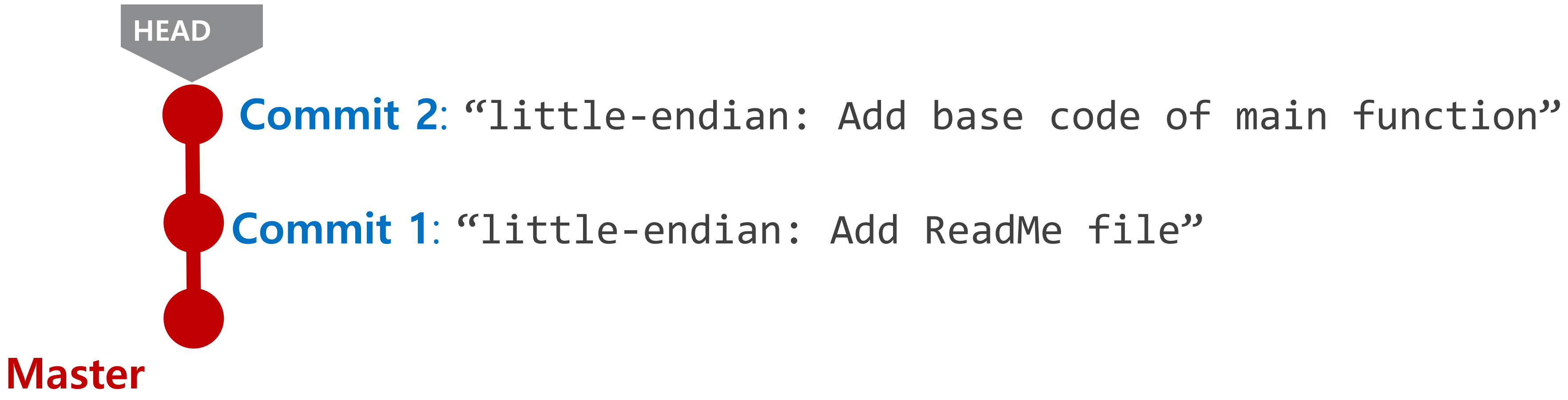


```
Initialized empty Git repository in C:/Users/woongbak-research/little-endian/.gi
t/

woongbak-research@woongbak MINGW64 ~/little-endian (master)
$ git add LittleEndian_Detector.c

woongbak-research@woongbak MINGW64 ~/little-endian (master)
$ git commit -m "little-endian: Add base code of main function"
[master (root-commit) a416cae] little-endian: Add base code of main function
 1 file changed, 39 insertions(+)
 create mode 100644 LittleEndian_Detector.c

woongbak-research@woongbak MINGW64 ~/little-endian (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        ReadMe.pdf

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

woongbak-research@woongbak MINGW64 ~/little-endian (master)
$
```

To check current condition

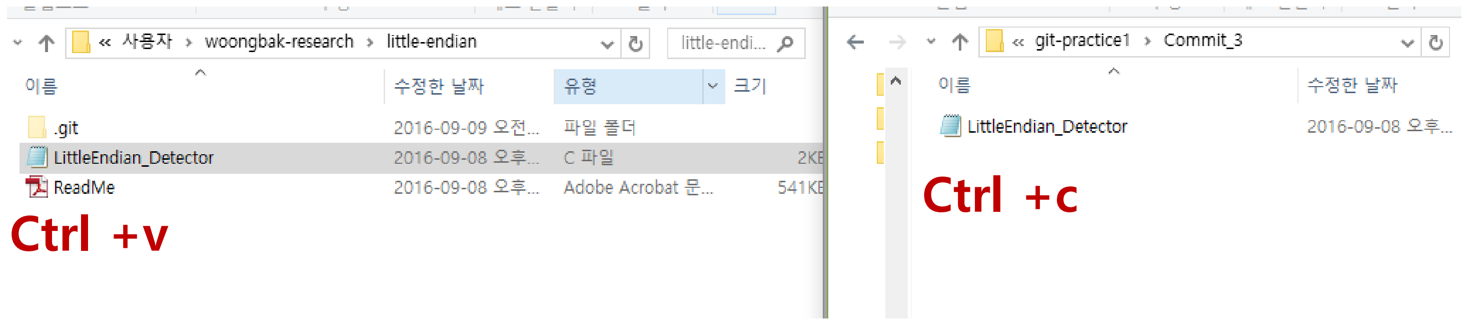
```
$ git show
$ git log
$ git diff
$ git status
```

OPERATIONS BOARD


HEAD
Commit 2: “little-endian: Add base code of main function”
Commit 1: “little-endian: Add ReadMe file”
Master

BASIC GIT PRACTICE

13. Overwrite “LittleEndian_Detector.c” in “Commit-3” folder into “LittleEndian_Detector.c” of “little-endian” folder



« 사용자 › woongbak-research › little-endian

이름	수정한 날짜	유형	크기
.git	2016-09-09 오전...	파일 폴더	
LittleEndian_Detector	2016-09-08 오후...	C 파일	2KB
ReadMe	2016-09-08 오후...	Adobe Acrobat 문...	541KB

Ctrl +v

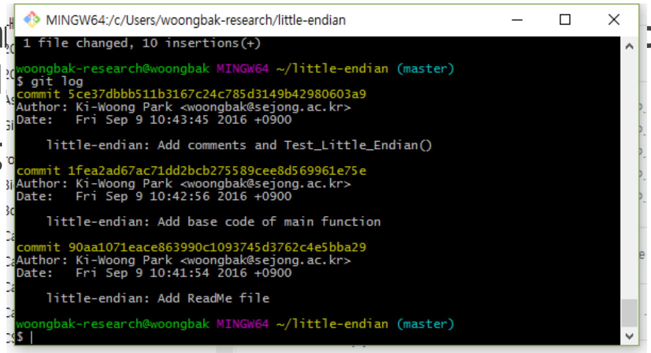
« git-practice1 › Commit_3

이름	수정한 날짜
LittleEndian_Detector	2016-09-08 오후...

Ctrl +c

14. Commit the file in that directory into the repo:

```
$ git diff
$ git add LittleEndian_Detector.c
$ git com                                   comments and
Test_Little_End
$ git log
```

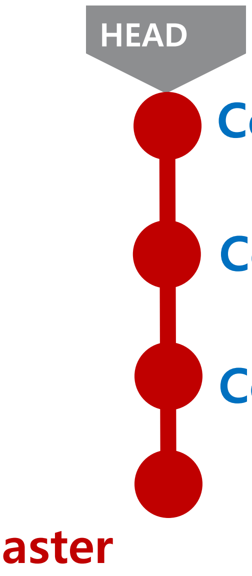


```
 1 file changed, 10 insertions(+)

woongbak-research@woongbak MINGW64 ~/little-endian (master)
$ git log
commit 5ce37dbbb511b3167c24c785d3149b42980603a9
Author: Ki-Woong Park <woongbak@sejong.ac.kr>
Date:   Fri Sep 9 10:43:45 2016 +0900

    little-endian: Add comments and Test_Little_Endian()

commit 1fea2ad67ac71dd2bcb275589cee8d569961e75e
Author: Ki-Woong Park <woongbak@sejong.ac.kr>
Date:   Fri Sep 9 10:42:56 2016 +0900

    little-endian: Add base code of main function

commit 90aa1071eace863990c1093745d3762c4e5bba29
Author: Ki-Woong Park <woongbak@sejong.ac.kr>
Date:   Fri Sep 9 10:41:54 2016 +0900

    little-endian: Add ReadMe file

woongbak-research@woongbak MINGW64 ~/little-endian (master)
$ |
```

3 commits

SYSCORE System Security & Computer Engineering Research Lab.

OPERATIONS BOARD



Commit 3: “`little-endian: Add comments and Test_Little_Endian()`”

Commit 2: “`little-endian: Add base code of main function`”

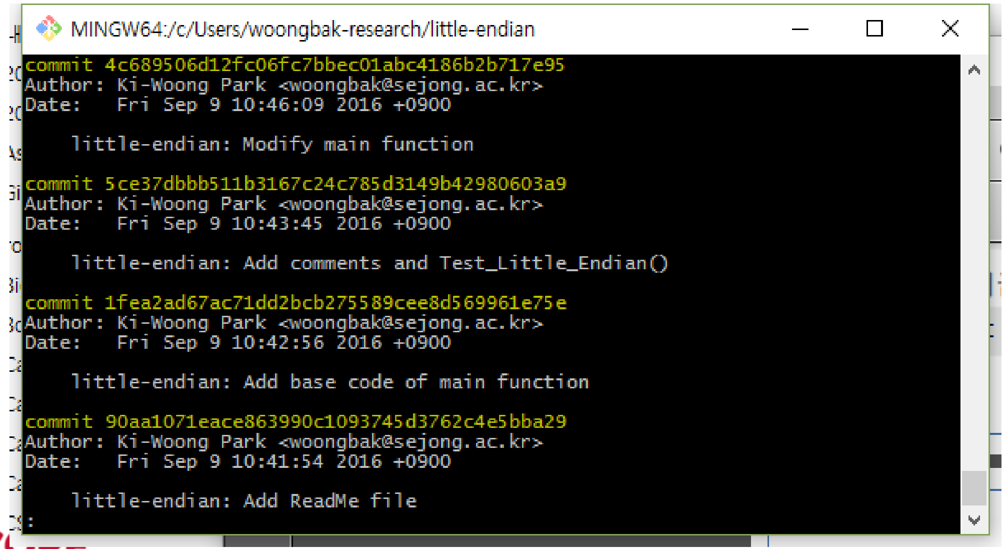
Commit 1: “`little-endian: Add ReadMe file`”

BASIC GIT PRACTICE

15. Overwrite “LittleEndian_Detector.c” in “Commit-4” folder into “LittleEndian_Detector.c” of “little-endian” folder

16. Commit the file in that directory into the repo:

```
$ git diff
$ git add LittleEndian_Detector.c
$ git commit –m “little-endian: Modify main function”
```

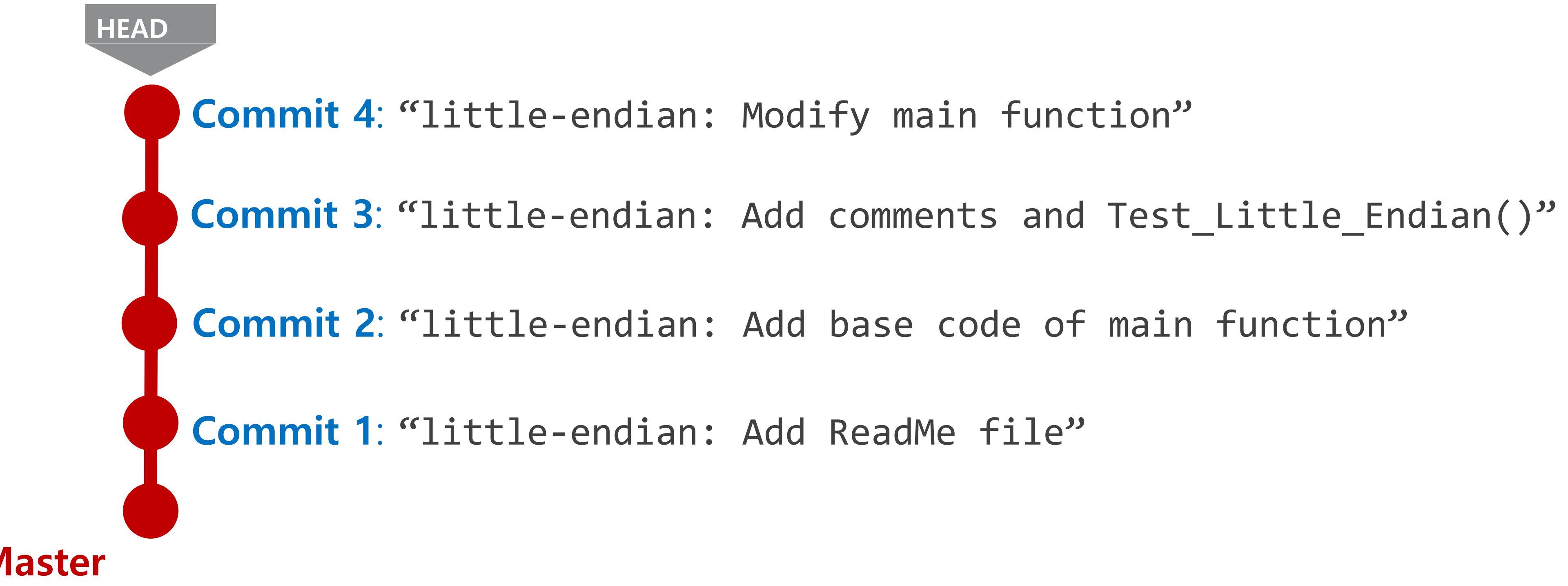


MINGW64:/c/Users/woongbak-research/little-endian

```
commit 4c689506d12fc06fc7bbec01abc4186b2b717e95
Author: Ki-Woong Park <woongbak@sejong.ac.kr>
Date:   Fri Sep 9 10:46:09 2016 +0900

    little-endian: Modify main function

commit 5ce37dbbb511b3167c24c785d3149b42980603a9
Author: Ki-Woong Park <woongbak@sejong.ac.kr>
Date:   Fri Sep 9 10:43:45 2016 +0900

    little-endian: Add comments and Test_Little_Endian()

commit 1fea2ad67ac71dd2bcb275589cee8d569961e75e
Author: Ki-Woong Park <woongbak@sejong.ac.kr>
Date:   Fri Sep 9 10:42:56 2016 +0900

    little-endian: Add base code of main function

commit 90aa1071eace863990c1093745d3762c4e5bba29
Author: Ki-Woong Park <woongbak@sejong.ac.kr>
Date:   Fri Sep 9 10:41:54 2016 +0900

    little-endian: Add ReadMe file
:
```

4 commits

OPERATIONS BOARD



Commit 4: “little-endian: Modify main function”

Commit 3: “little-endian: Add comments and Test_Little_Endian()”

Commit 2: “little-endian: Add base code of main function”

Commit 1: “little-endian: Add ReadMe file”

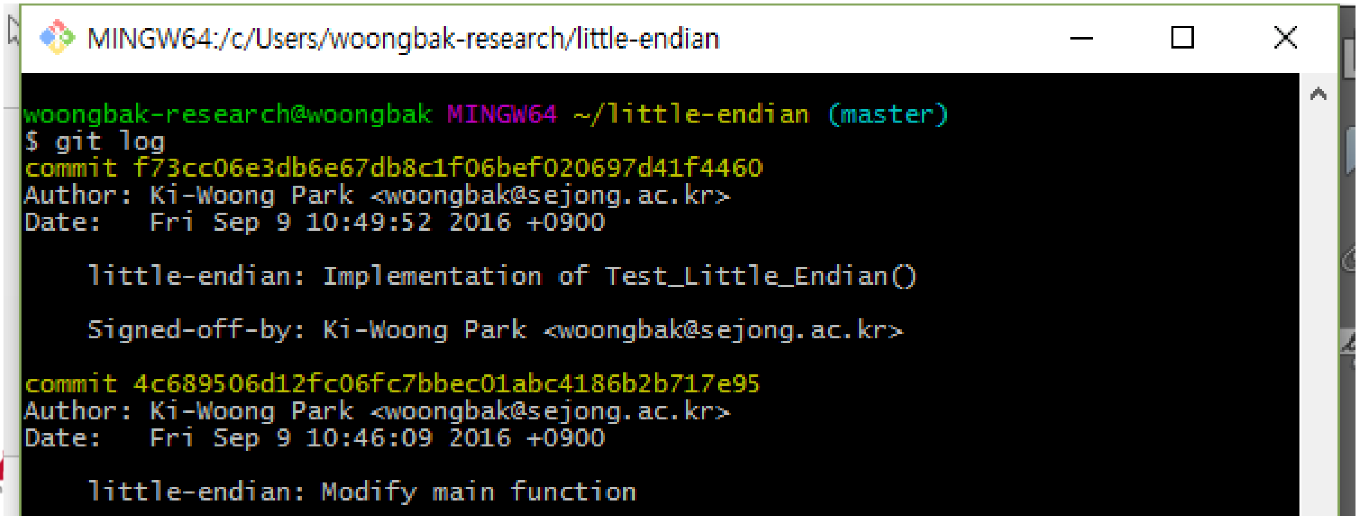
BASIC GIT PRACTICE

17. Overwrite “LittleEndian_Detector.c” in “Commit-5” folder into “LittleEndian_Detector.c” of “little-endian” folder

18. Commit the file in that directory into the repo:

```
$ git diff
$ git add LittleEndian_Detector.c
$ git commit -sm “little-endian: Implementation of Test_Little_Endian()”
```

-sm : commit with your signature



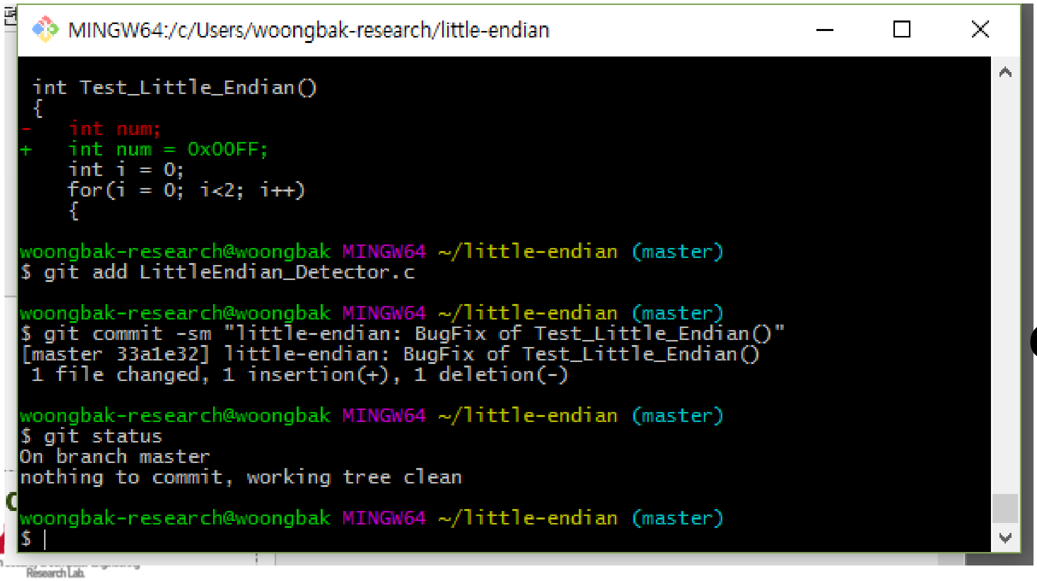
```
woongbak-research@woongbak MINGW64 ~/little-endian (master)
$ git log
commit f73cc06e3db6e67db8c1f06bef020697d41f4460
Author: Ki-Woong Park <woongbak@sejong.ac.kr>
Date:   Fri Sep 9 10:49:52 2016 +0900

    little-endian: Implementation of Test_Little_Endian()

    Signed-off-by: Ki-Woong Park <woongbak@sejong.ac.kr>

commit 4c689506d12fc06fc7bbec01abc4186b2b717e95
Author: Ki-Woong Park <woongbak@sejong.ac.kr>
Date:   Fri Sep 9 10:46:09 2016 +0900

    little-endian: Modify main function
```

BASIC GIT PRACTICE

19. Overwrite "LittleEndian_Detector.c" in "Commit-6" folder into "LittleEndian_Detector.c" of "little-endian" folder

20. Commit the file in that directory into the repo:

```
$ git diff
$ git add LittleEndian_Detector.c
$ git commit –sm "little-endian: BugFix of Test_Little_Endian()"
```



MINGW64:/c/Users/woongbak-research/little-endian

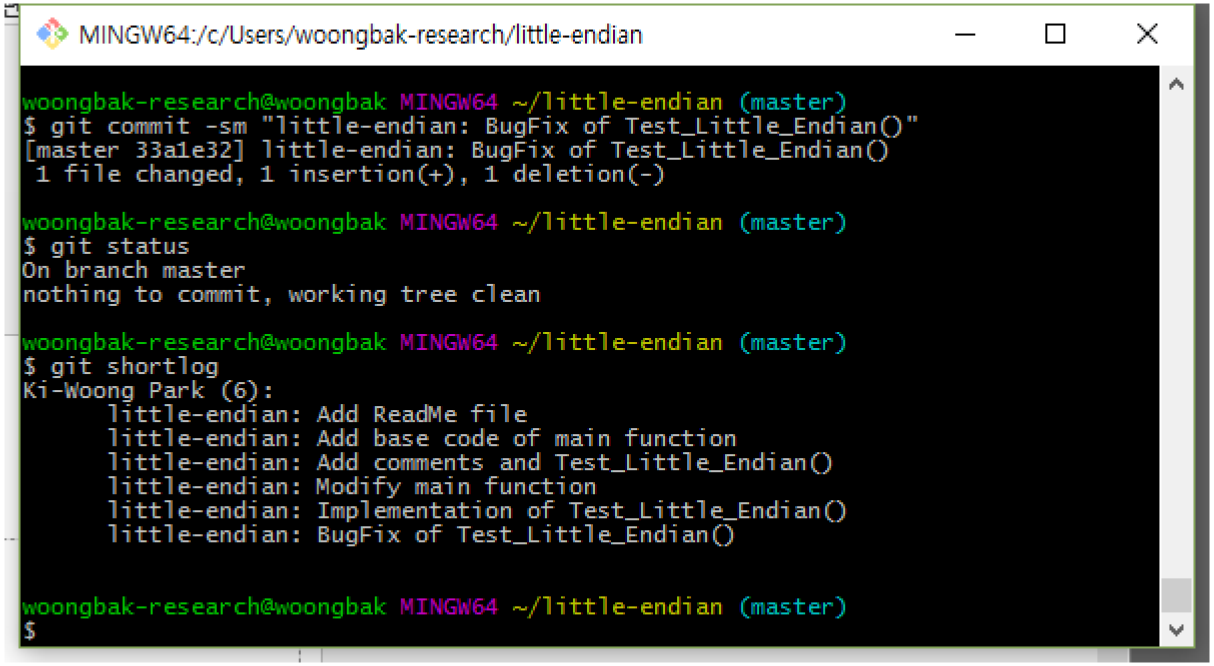
```
 int Test_Little_Endian()
 {
-    int num;
+    int num = 0x00FF;
     int i = 0;
     for(i = 0; i<2; i++)
     {

woongbak-research@woongbak MINGW64 ~/little-endian (master)
$ git add LittleEndian_Detector.c

woongbak-research@woongbak MINGW64 ~/little-endian (master)
$ git commit -sm "little-endian: BugFix of Test_Little_Endian()"
[master 33a1e32] little-endian: BugFix of Test_Little_Endian()
 1 file changed, 1 insertion(+), 1 deletion(-)

woongbak-research@woongbak MINGW64 ~/little-endian (master)
$ git status
On branch master
nothing to commit, working tree clean

woongbak-research@woongbak MINGW64 ~/little-endian (master)
$ |
```

Current Branch: master

BASIC GIT PRACTICE

21. Check six commits we have done

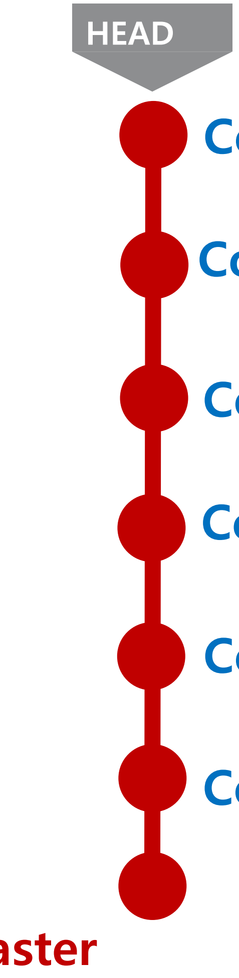
```
$ git shortlog
```



MINGW64:/c/Users/woongbak-research/little-endian

```
woongbak-research@woongbak MINGW64 ~/little-endian (master)
$ git commit -sm "little-endian: BugFix of Test_Little_Endian()"
[master 33a1e32] little-endian: BugFix of Test_Little_Endian()
 1 file changed, 1 insertion(+), 1 deletion(-)

woongbak-research@woongbak MINGW64 ~/little-endian (master)
$ git status
On branch master
nothing to commit, working tree clean

woongbak-research@woongbak MINGW64 ~/little-endian (master)
$ git shortlog
Ki-Woong Park (6):
      little-endian: Add ReadMe file
      little-endian: Add base code of main function
      little-endian: Add comments and Test_Little_Endian()
      little-endian: Modify main function
      little-endian: Implementation of Test_Little_Endian()
      little-endian: BugFix of Test_Little_Endian()


woongbak-research@woongbak MINGW64 ~/little-endian (master)
$
```

OPERATIONS BOARD



Commit 6: “`little-endian: BugFix of Test_Little_Endian()`”

Commit 5: “`little-endian: Implementation of Test_Little_Endian()`”

Commit 4: “`little-endian: Modify main function`”

Commit 3: “`little-endian: Add comments and Test_Little_Endian()`”

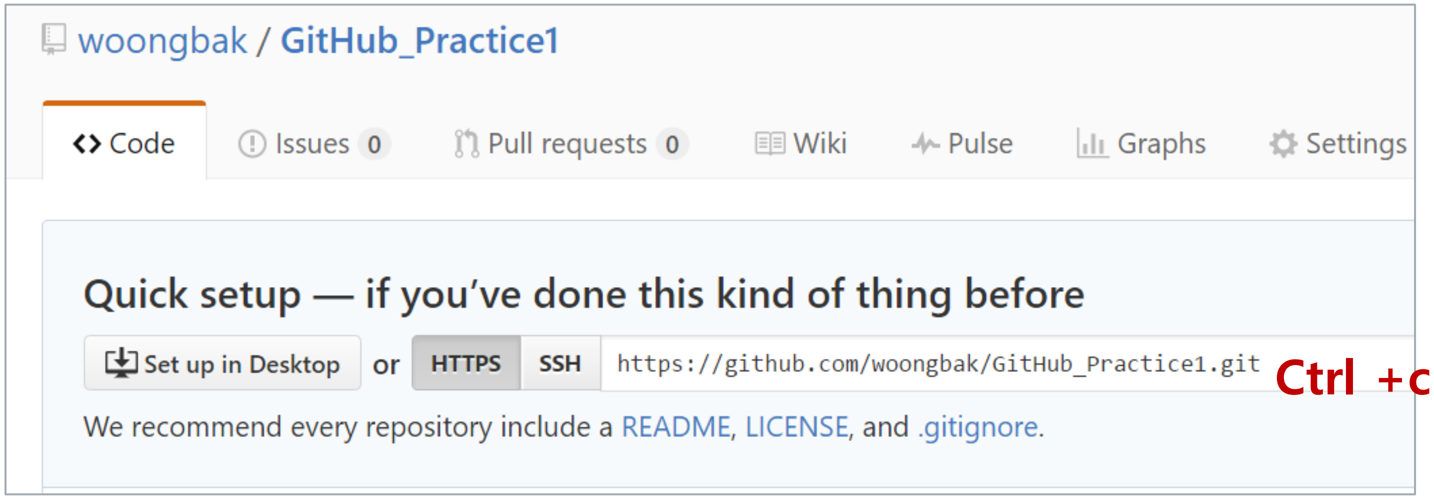
Commit 2: “`little-endian: Add base code of main function`”

Commit 1: “`little-endian: Add ReadMe file`”

BASIC GIT PRACTICE

We will push commits to remote repository in GitHub

22. Register remote repository of GitHub



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

https://github.com/woongbak/GitHub_Practice1.git

We recommend every repository include a README, LICENSE, and .gitignore.

Ctrl +c

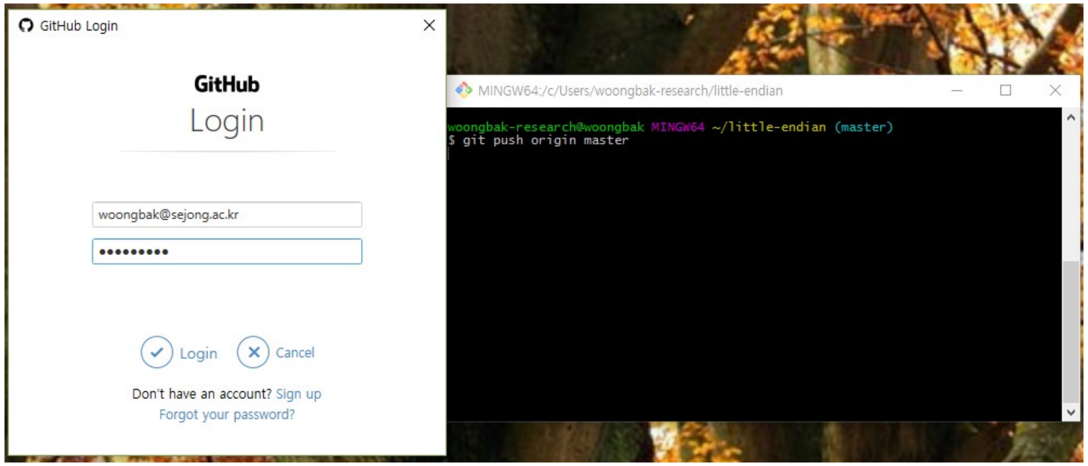
origin = alias for remote repository

```
# git remote add origin https://github.com/woongbak/GitHub_Practice1.git
```

BASIC GIT PRACTICE

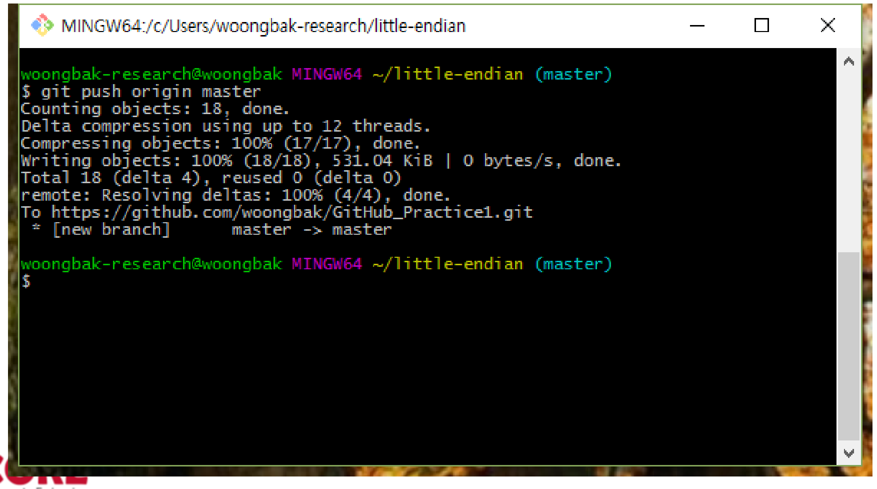
23. Push six commits into origin (GitHub remote repository)

```
$ git push origin master
```


GitHub Login
GitHub
Login
Login
Cancel
Don't have an account? Sign up
Forgot your password?
MINGW64:/c/Users/woongbak-research/little-endian

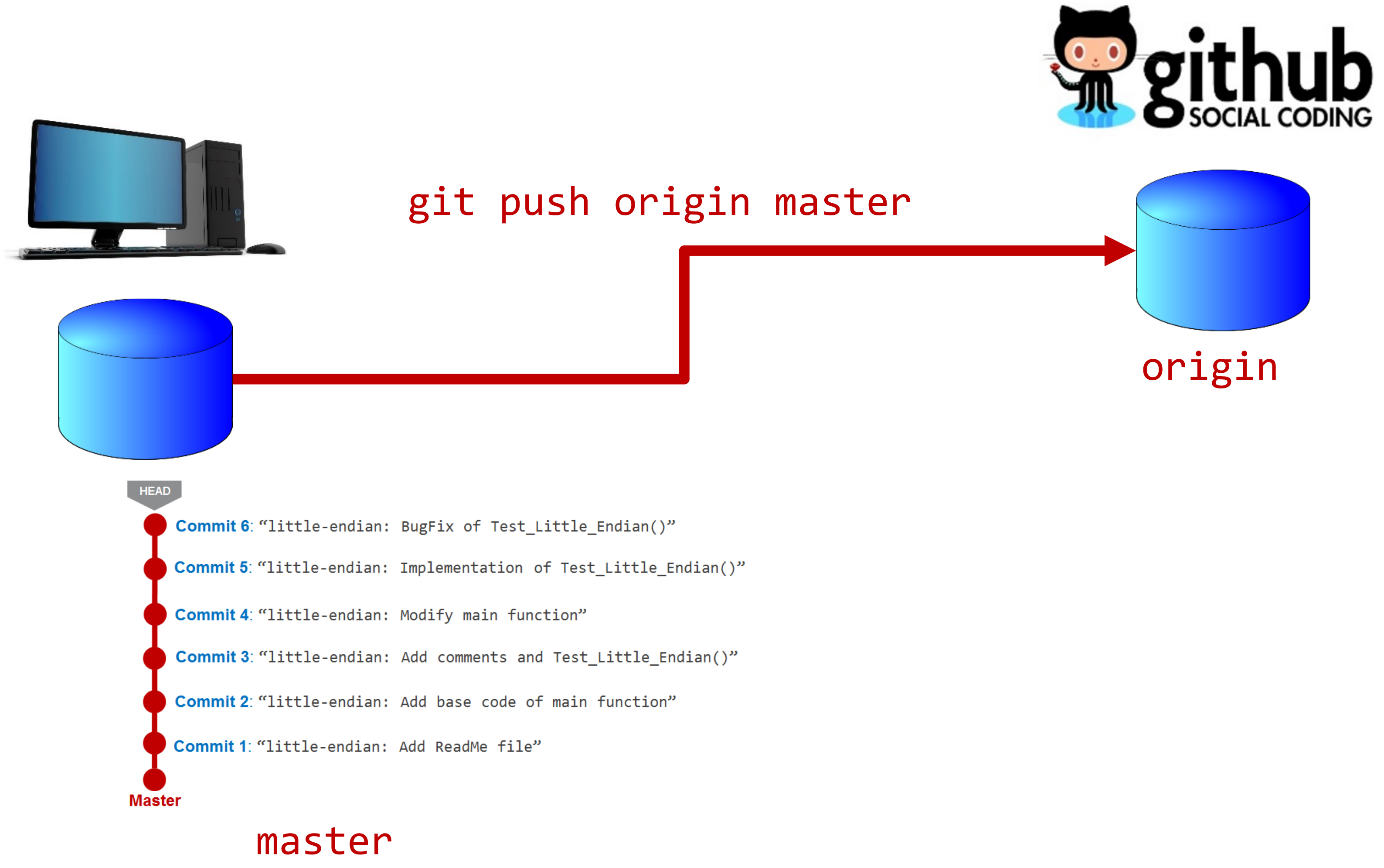
woongbak@sejong.ac.kr

```
woongbak-research@woongbak MINGW64 ~/little-endian (master)
$ git push origin master
```



```
woongbak-research@woongbak MINGW64 ~/little-endian (master)
$ git push origin master
Counting objects: 18, done.
Delta compression using up to 12 threads.
Compressing objects: 100% (17/17), done.
Writing objects: 100% (18/18), 531.04 KiB | 0 bytes/s, done.
Total 18 (delta 4), reused 0 (delta 0)
remote: Resolving deltas: 100% (4/4), done.
To https://github.com/woongbak/GitHub_Practice1.git
 * [new branch]      master -> master

woongbak-research@woongbak MINGW64 ~/little-endian (master)
$
```

OPERATIONS BOARD



Commit 6: "little-endian: BugFix of Test_Little_Endian()"

Commit 5: "little-endian: Implementation of Test_Little_Endian()"

Commit 4: "little-endian: Modify main function"

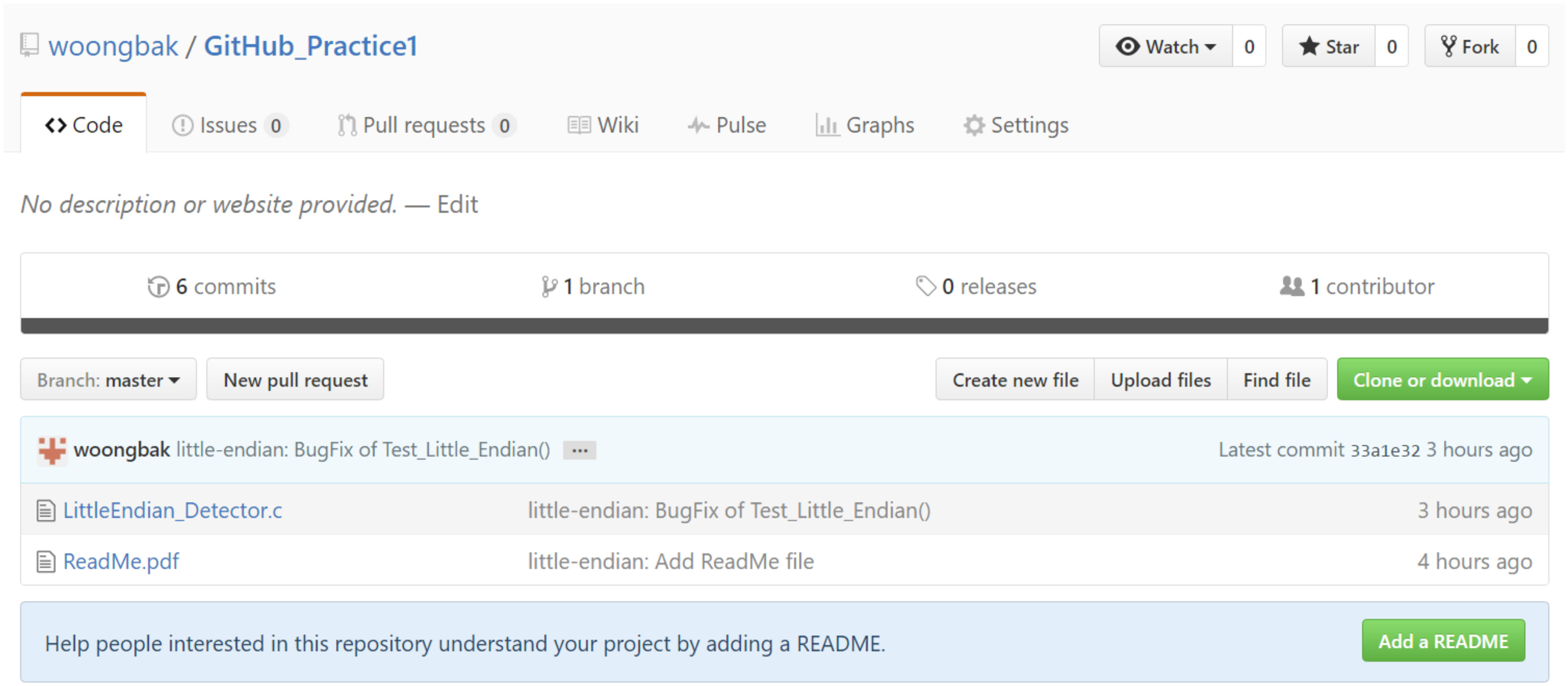
Commit 3: "little-endian: Add comments and Test_Little_Endian()"

Commit 2: "little-endian: Add base code of main function"

Commit 1: "little-endian: Add ReadMe file"

BASIC GIT PRACTICE

24. Check your GitHub Repository


woongbak / GitHub_Practice1

Watch 0 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Wiki | Pulse | Graphs | Settings

No description or website provided. — Edit

6 commits | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

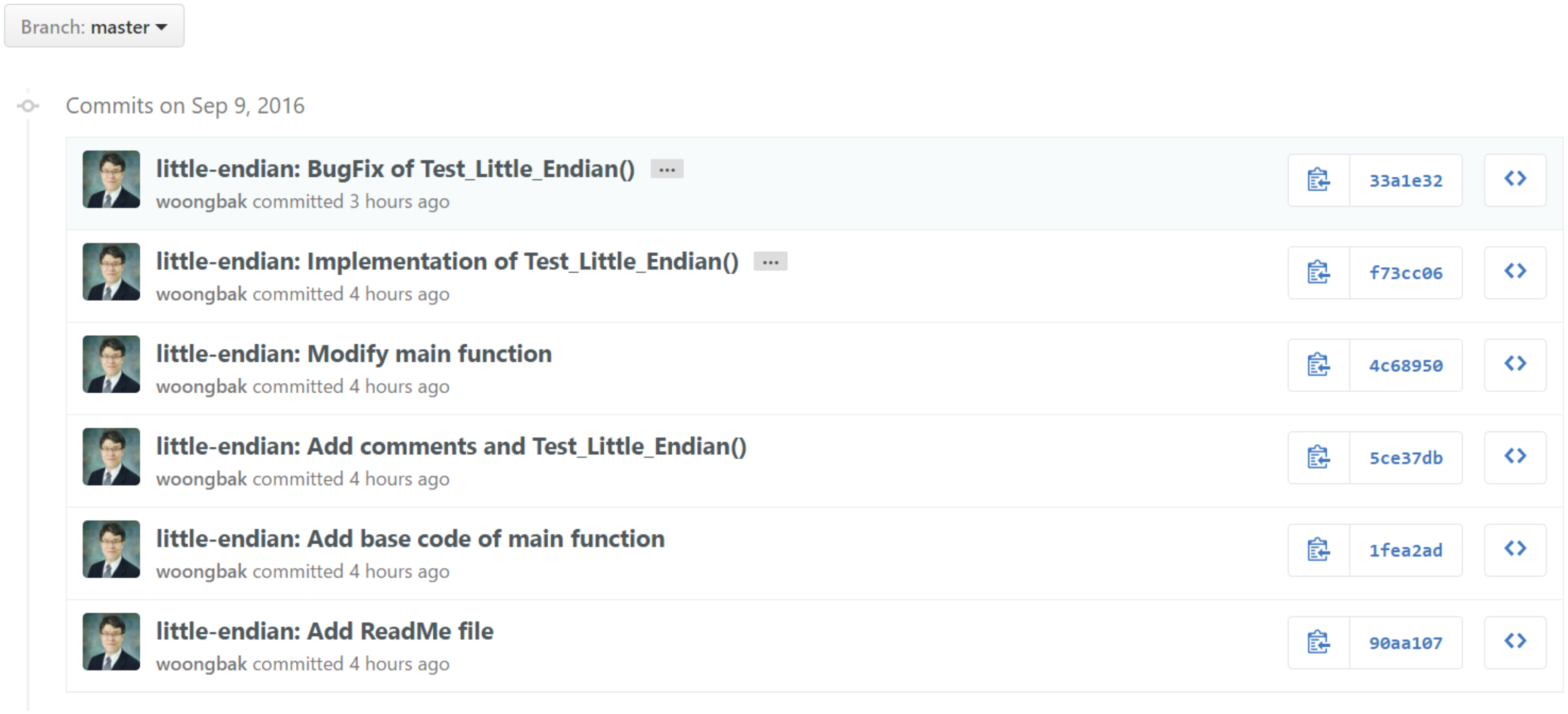
woongbak little-endian: BugFix of Test_Little_Endian() … Latest commit 33a1e32 3 hours ago

LittleEndian_Detector.c	little-endian: BugFix of Test_Little_Endian()	3 hours ago
ReadMe.pdf	little-endian: Add ReadMe file	4 hours ago

Help people interested in this repository understand your project by adding a README. Add a README

BASIC GIT PRACTICE

25. Check your commits



Branch: master

Commits on Sep 9, 2016

little-endian: BugFix of Test_Little_Endian() … woongbak committed 3 hours ago	33a1e32
little-endian: Implementation of Test_Little_Endian() … woongbak committed 4 hours ago	f73cc06
little-endian: Modify main function woongbak committed 4 hours ago	4c68950
little-endian: Add comments and Test_Little_Endian() woongbak committed 4 hours ago	5ce37db
little-endian: Add base code of main function woongbak committed 4 hours ago	1fea2ad
little-endian: Add ReadMe file woongbak committed 4 hours ago	90aa107

SUMMARY

- Describe what you have done

FROM NOW, ADVANCED GIT PRACTICE

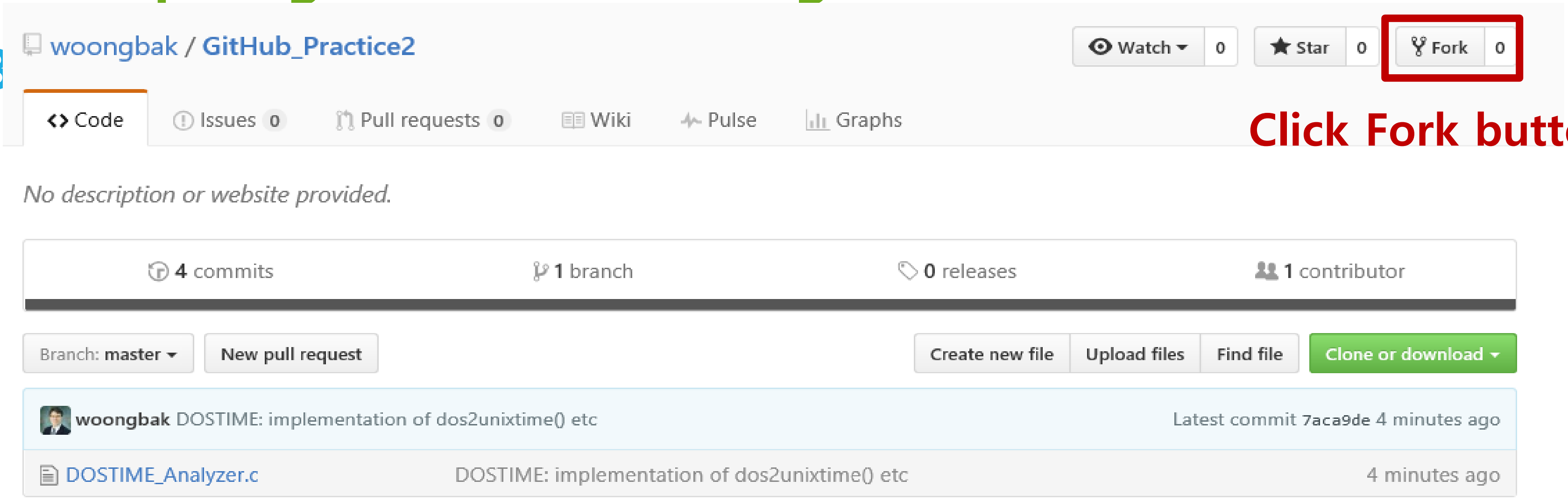
Goal: Contributing to original project

After your done with your changes and you want these changes to the original project you have create a pull request

FROM NOW, ADVANCED GIT PRACTICE

Forking project into your github account

1. Login into your github account that you have created.
2. Go to https://github.com/woongbak/GitHub_Practice2
3



No description or website provided.

woongbak DOSTIME: implementation of dos2unixtime() etc — Latest commit 7aca9de 4 minutes ago

DOSTIME_Analyzer.c — DOSTIME: implementation of dos2unixtime() etc — 4 minutes ago

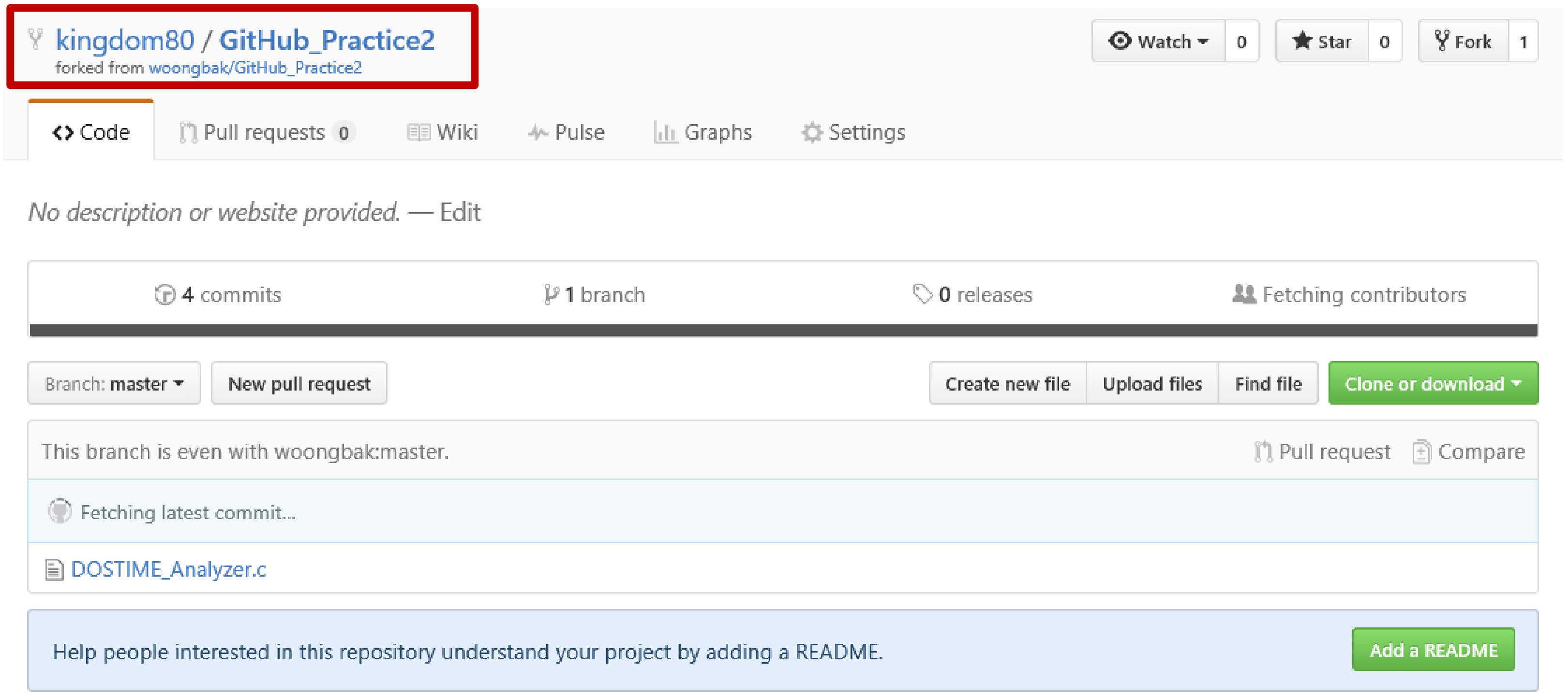
Now the source code will be available to your Account.
You can see that by checking your repositories

ADVANCED GIT PRACTICE

Now the source code will be available to your Account.

You can see that by checking your repositories

4. If your GitHub ID = kingdom80, you will see



kingdom80 / GitHub_Practice2
forked from woongbak/GitHub_Practice2

No description or website provided. — Edit

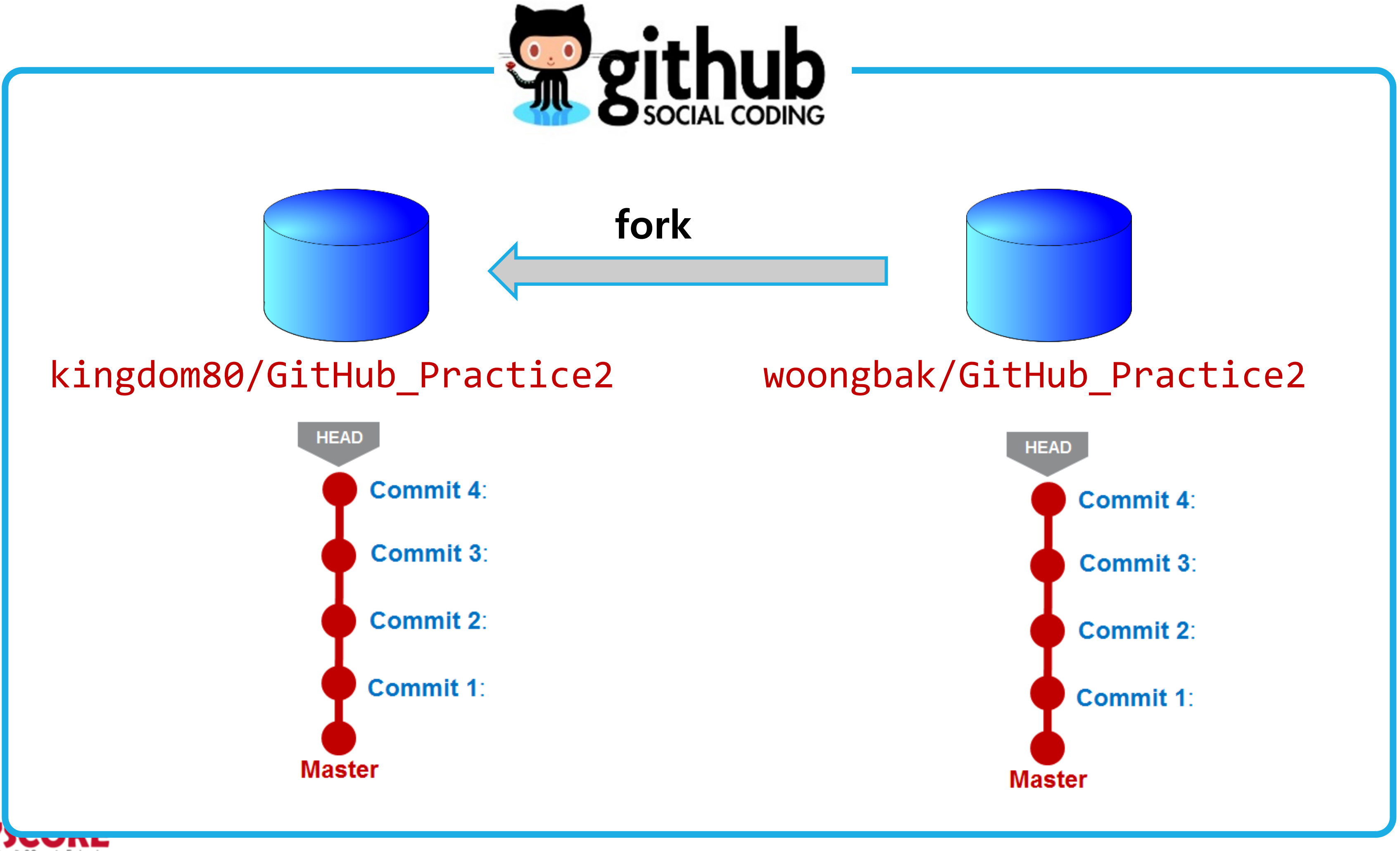
This branch is even with woongbak:master.

Fetching latest commit…

DOSTIME_Analyzer.c

Help people interested in this repository understand your project by adding a README.

OPERATIONS BOARD



ADVANCED GIT PRACTICE

5. In your git-bash,

```
$ cd /Users/yourPC_ID/
$ git clone https://github.com/kingdom80/GitHub_Practice2.git
```

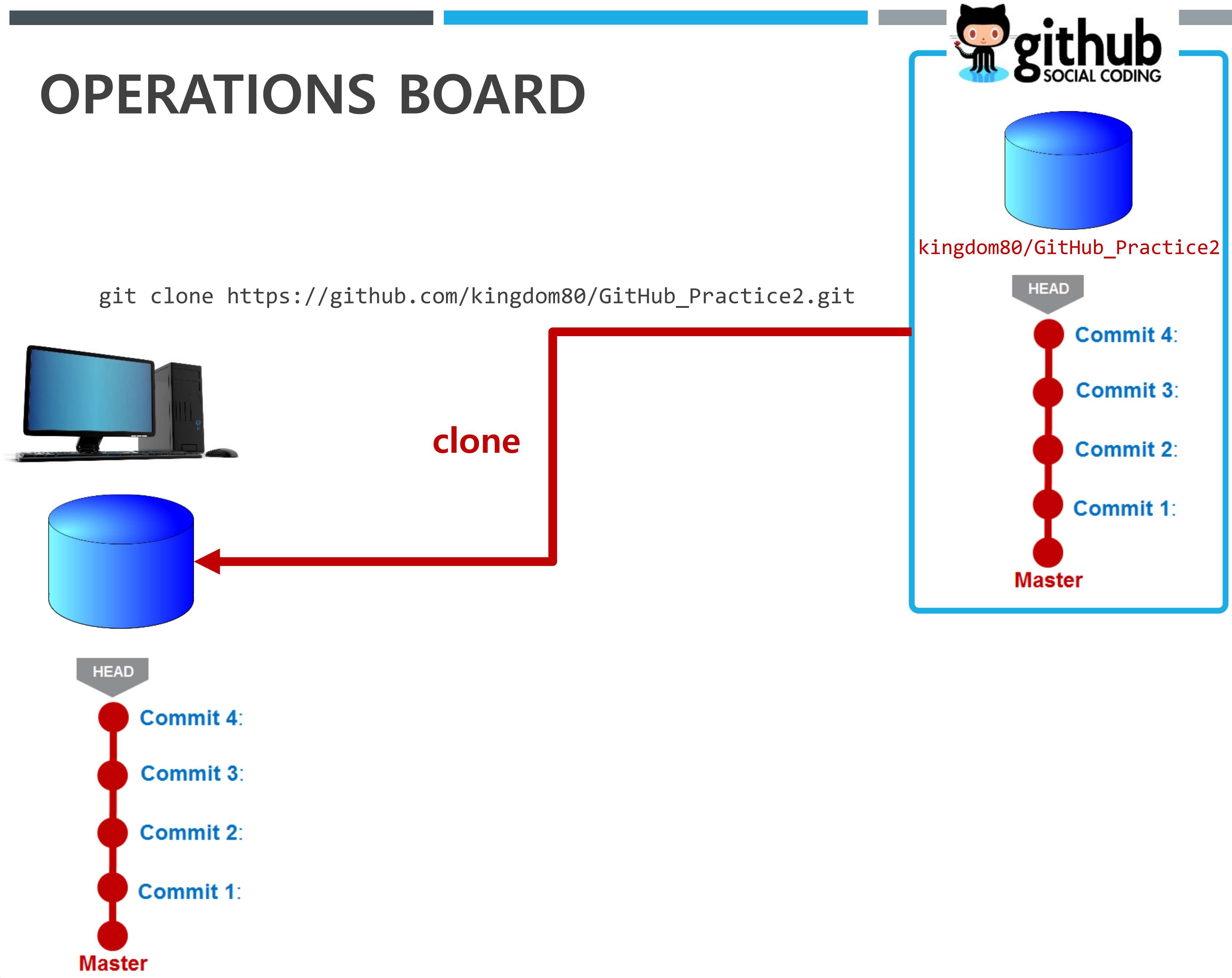
git clone https://github.com/<username>/<repository name>.git
Note: <username> is your github username and <repository name>
These make an exact copy of the repository at the given URL

6. Now you will have a folder named 'GitHub_Practice2'.
Go to the directory

```
$ cd GitHub_practice2
```

OPERATIONS BOARD

```
git clone https://github.com/kingdom80/GitHub_Practice2.git
```



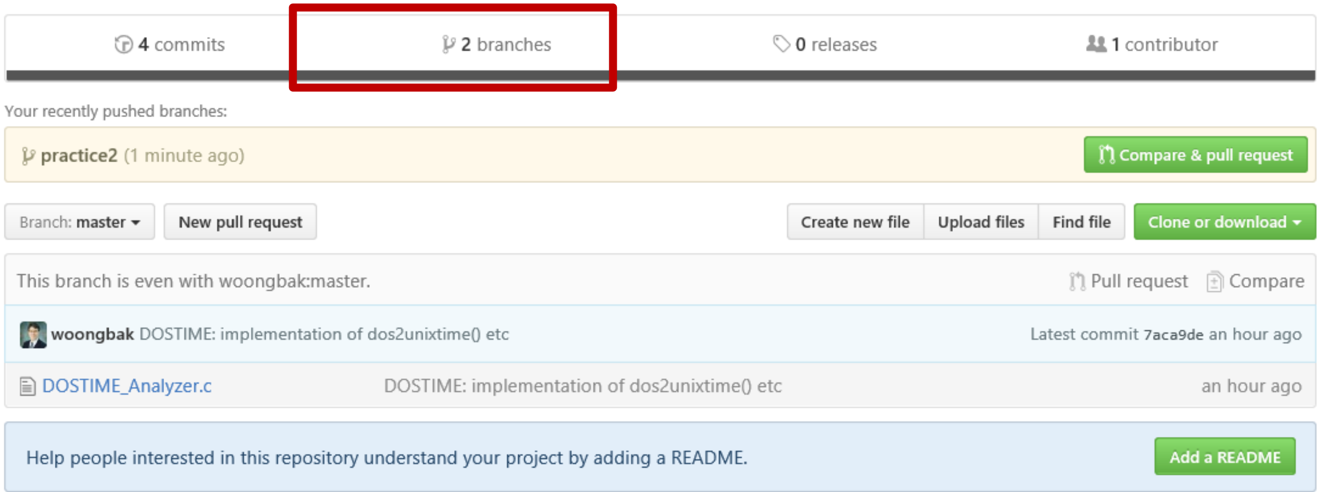
ADVANCED GIT PRACTICE

6. Make a branch (practice2) and Add/Commit a file

```
$ git checkout –b practice2
$ touch test_code.c
$ git add test_code.c
$ git commit –m “Add test_code by kingdom80”
```

7. Push the changes into your remote repository

```
$ git
```



4 commits | 2 branches | 0 releases | 1 contributor

Your recently pushed branches:

practice2 (1 minute ago) — Compare & pull request

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

This branch is even with woongbak:master. Pull request Compare

woongbak DOSTIME: implementation of dos2unixtime() etc — Latest commit 7aca9de an hour ago

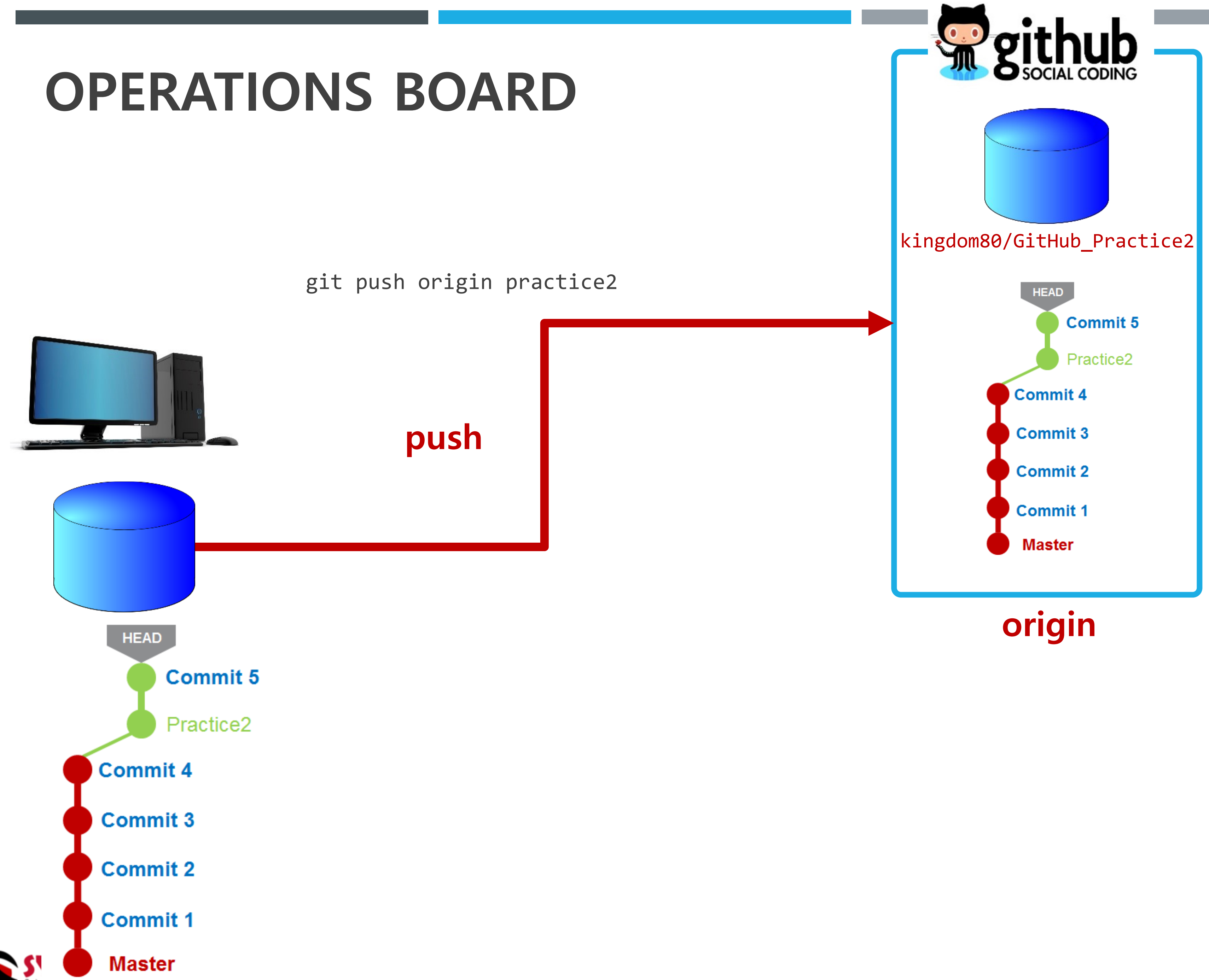
DOSTIME_Analyzer.c	DOSTIME: implementation of dos2unixtime() etc	an hour ago

Help people interested in this repository understand your project by adding a README. Add a README

SYSCORI System Security & Computer Engineering Research Lab.

OPERATIONS BOARD

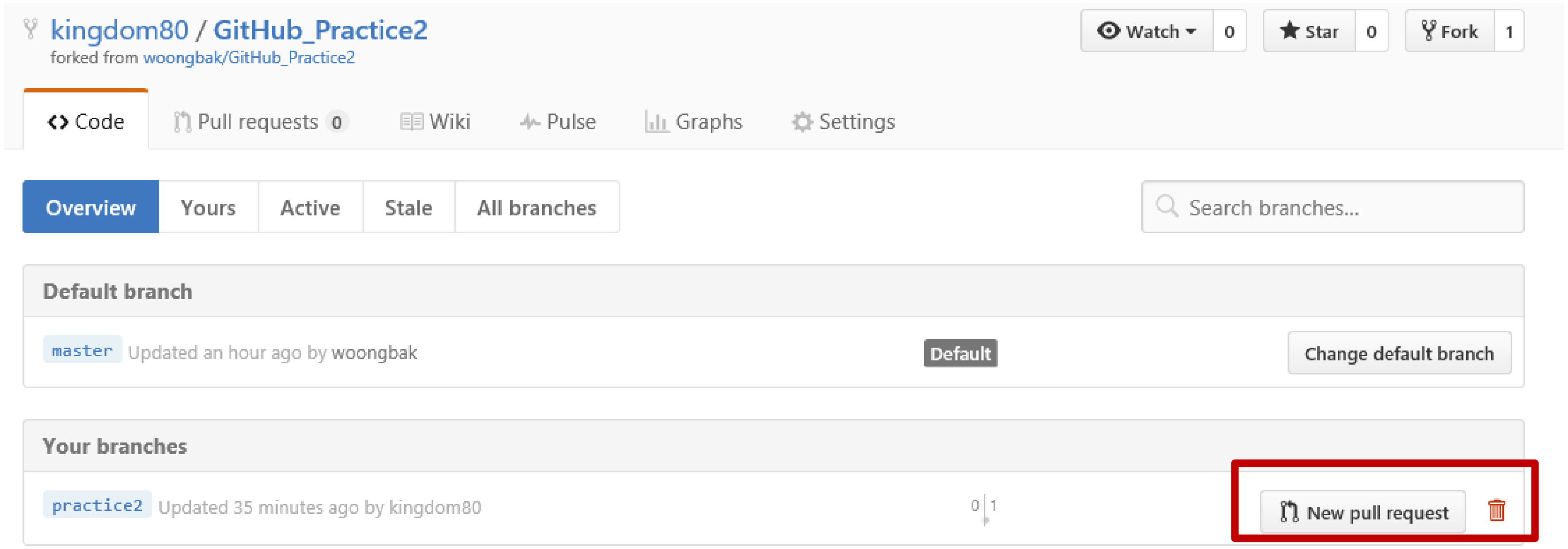
```
git push origin practice2
```



ADVANCED GIT PRACTICE

After your done with your changes and you want these changes to the original project you have create a pull request

8. Press 'New pull request'


kingdom80 / GitHub_Practice2
forked from woongbak/GitHub_Practice2

Default branch

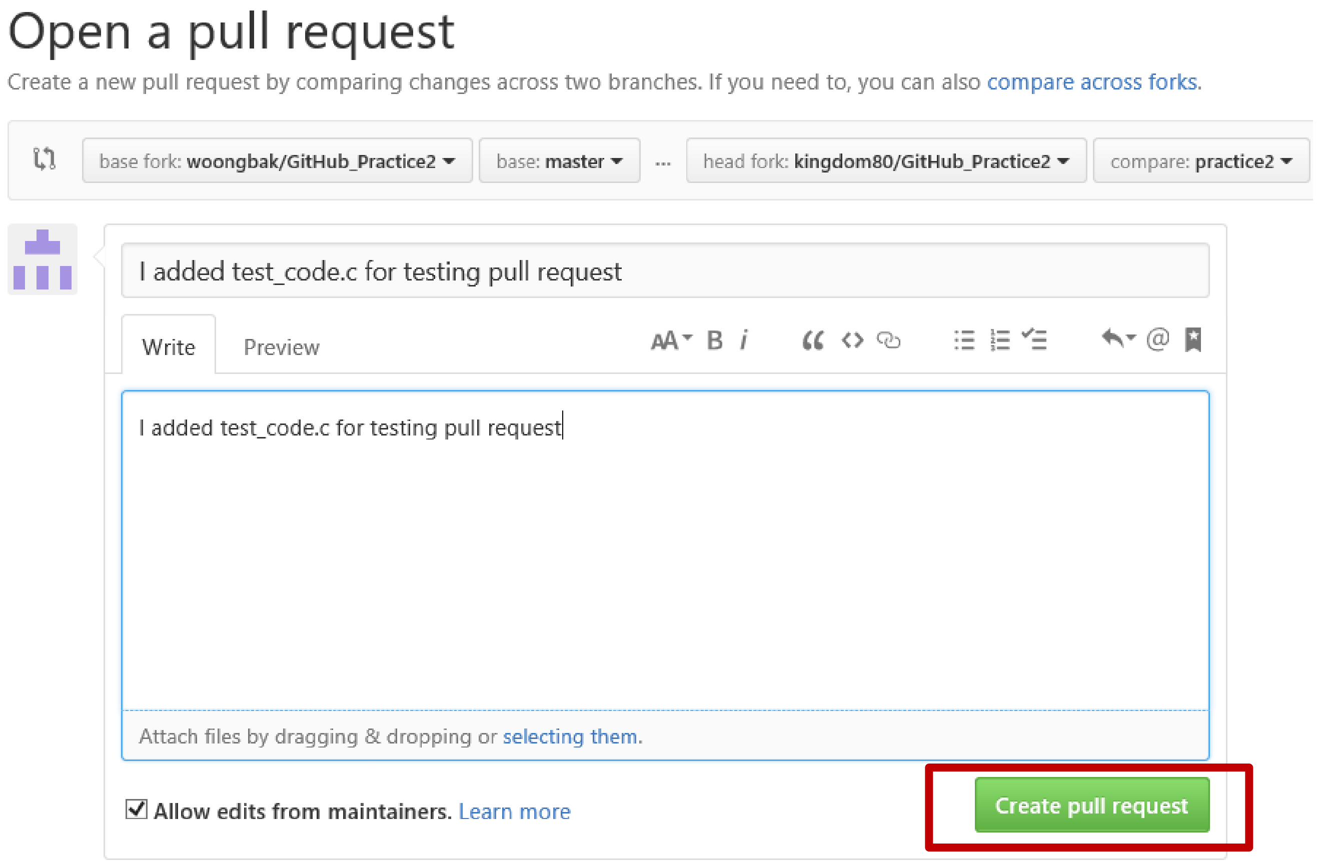
master Updated an hour ago by woongbak Default Change default branch

Your branches

practice2 Updated 35 minutes ago by kingdom80 0 | 1 New pull request

ADVANCED GIT PRACTICE

9. Describe pull request message and create a pull request


Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

base fork: woongbak/GitHub_Practice2 ▾ base: master ▾ ... head fork: kingdom80/GitHub_Practice2 ▾ compare: practice2 ▾

I added test_code.c for testing pull request

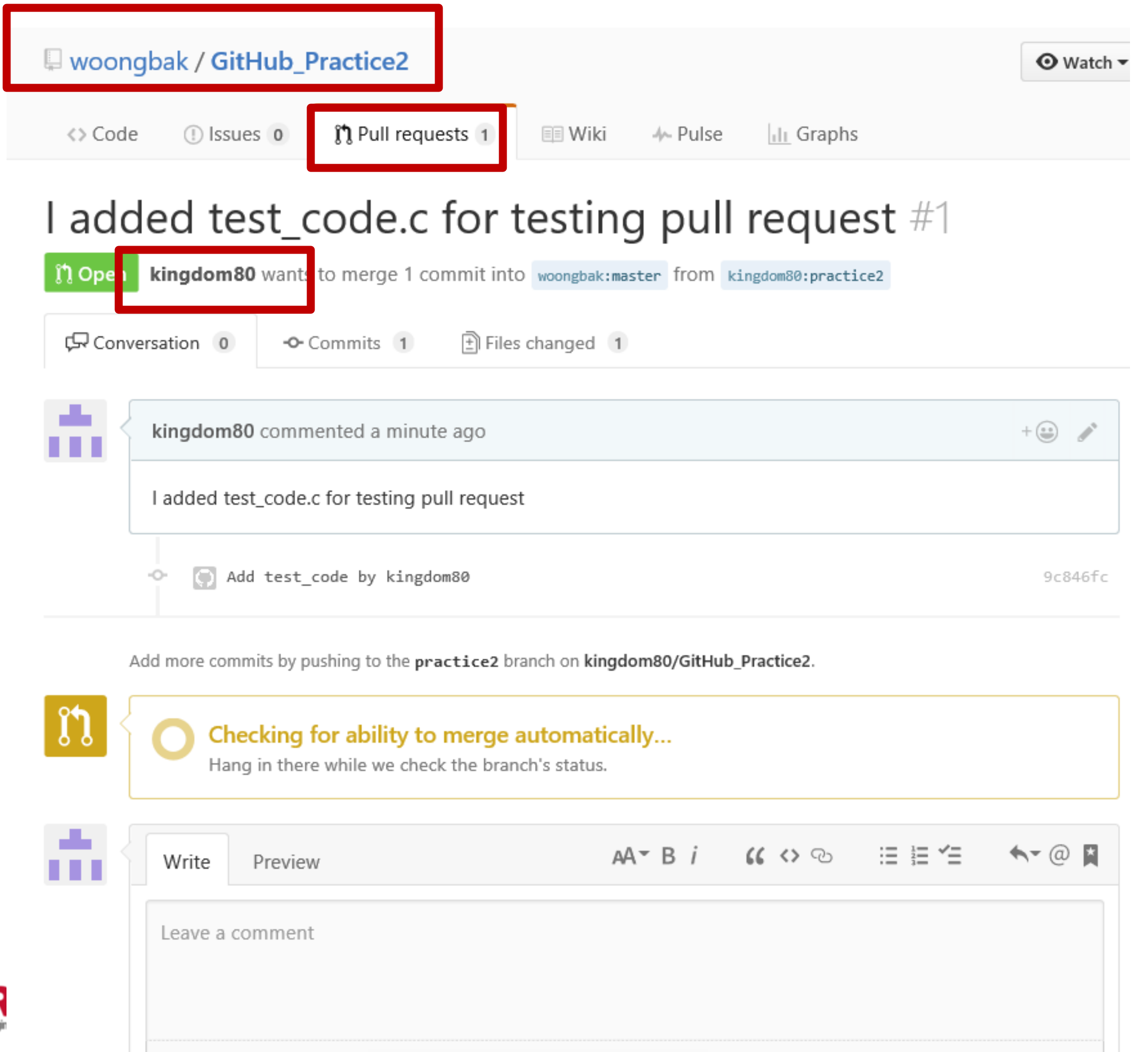
I added test_code.c for testing pull request

Attach files by dragging & dropping or selecting them.

☑ Allow edits from maintainers. Learn more

ADVANCED GIT PRACTICE

10. Checking the created pull request by you


woongbak / GitHub_Practice2

I added test_code.c for testing pull request #1

Open kingdom80 wants to merge 1 commit into `woongbak:master` from `kingdom80:practice2`

kingdom80 commented a minute ago

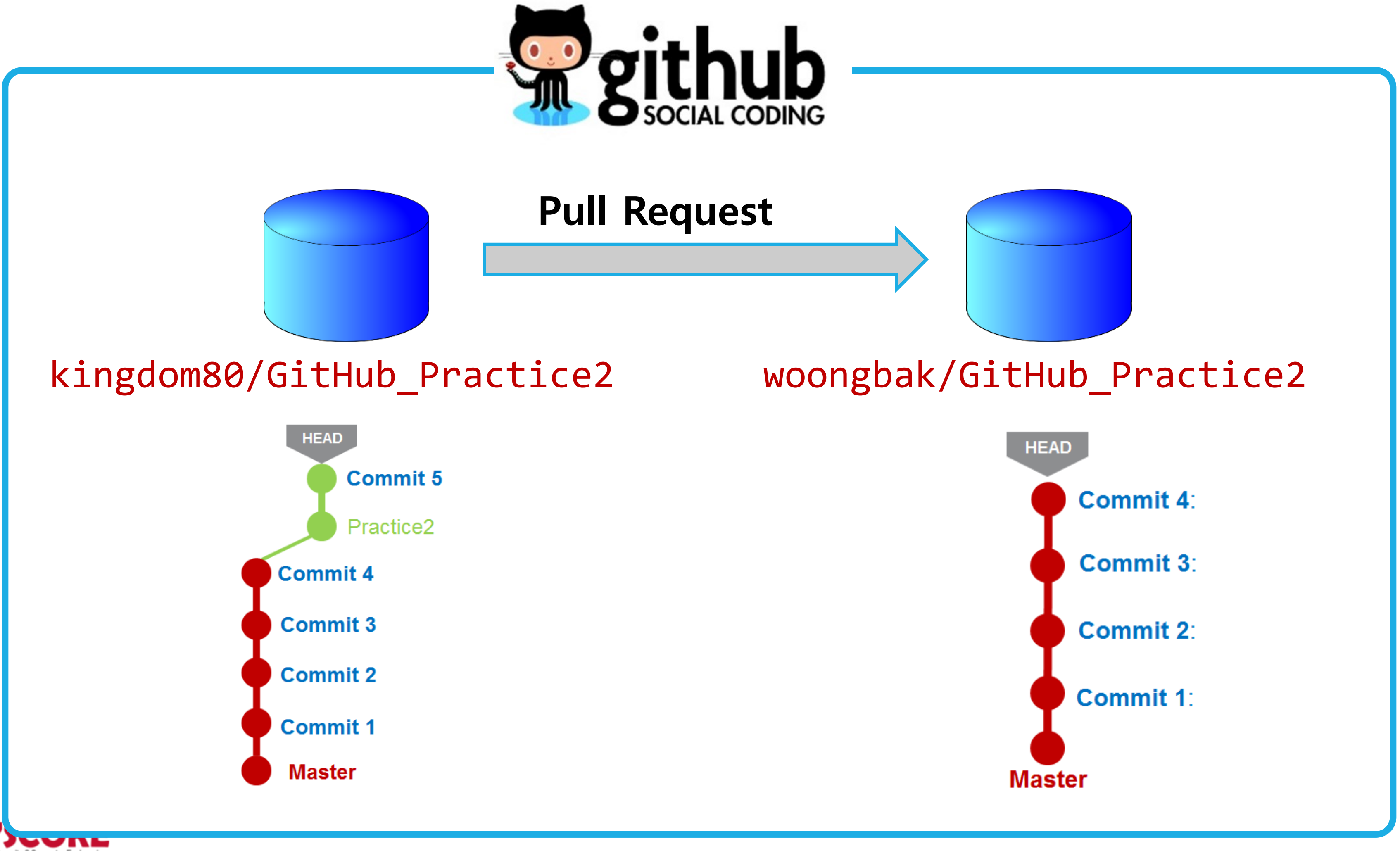
I added test_code.c for testing pull request

```
Add test_code by kingdom80                                    9c846fc
```

Add more commits by pushing to the **practice2** branch on **kingdom80/GitHub_Practice2**.

Checking for ability to merge automatically...
Hang in there while we check the branch's status.

OPERATIONS BOARD



<FOOTNOTE> BRANCHING

- **To create a branch called experimental**
  - `$ git branch experimental`
- **To list all branches:** (* shows which one you are currently on)
  - `$ git branch`
- **To switch to the experimental branch:**
  - `$ git checkout experimental`
- **Later on, changes between the two branches differ, to merge changes from experimental into the master:**
  - `$ git checkout master`
  - `$ git merge experimental`

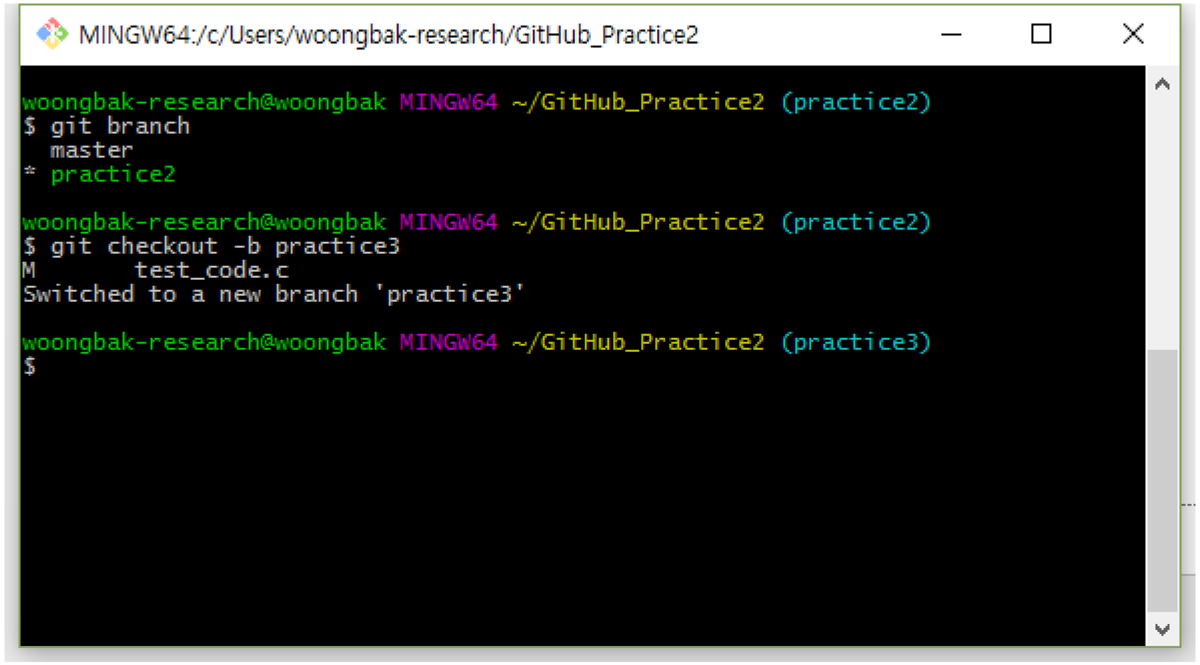
Note: `git log --graph` **can be useful for showing branches.**

Note: These branches are in *your local repo*!

SYSCORE System Security & Computer Engineering Research Lab.

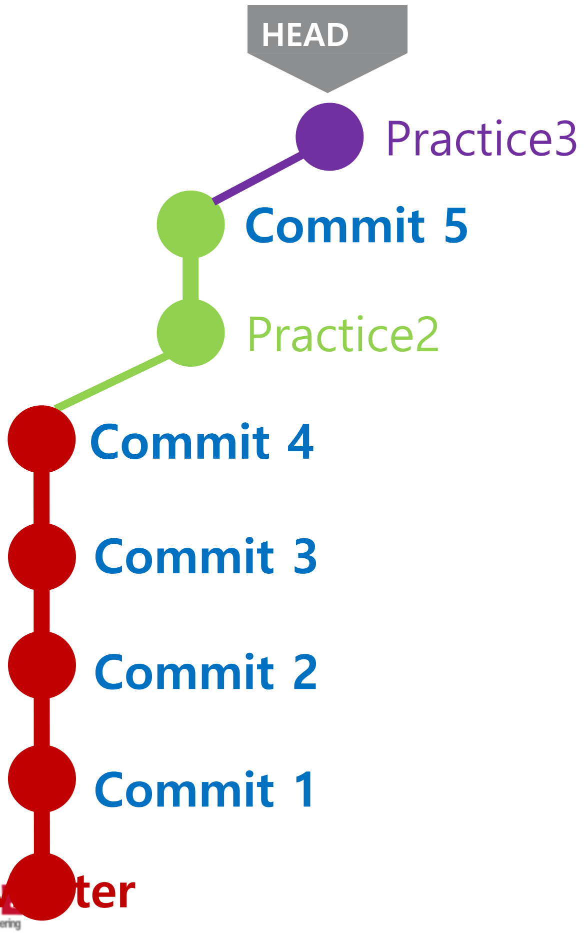
ADVANCED GIT PRACTICE

```
$ git branch
$ git checkout -b practice3
```


MINGW64:/c/Users/woongbak-research/GitHub_Practice2

```
woongbak-research@woongbak MINGW64 ~/GitHub_Practice2 (practice2)
$ git branch
  master
* practice2

woongbak-research@woongbak MINGW64 ~/GitHub_Practice2 (practice2)
$ git checkout -b practice3
M       test_code.c
Switched to a new branch 'practice3'

woongbak-research@woongbak MINGW64 ~/GitHub_Practice2 (practice3)
$
```

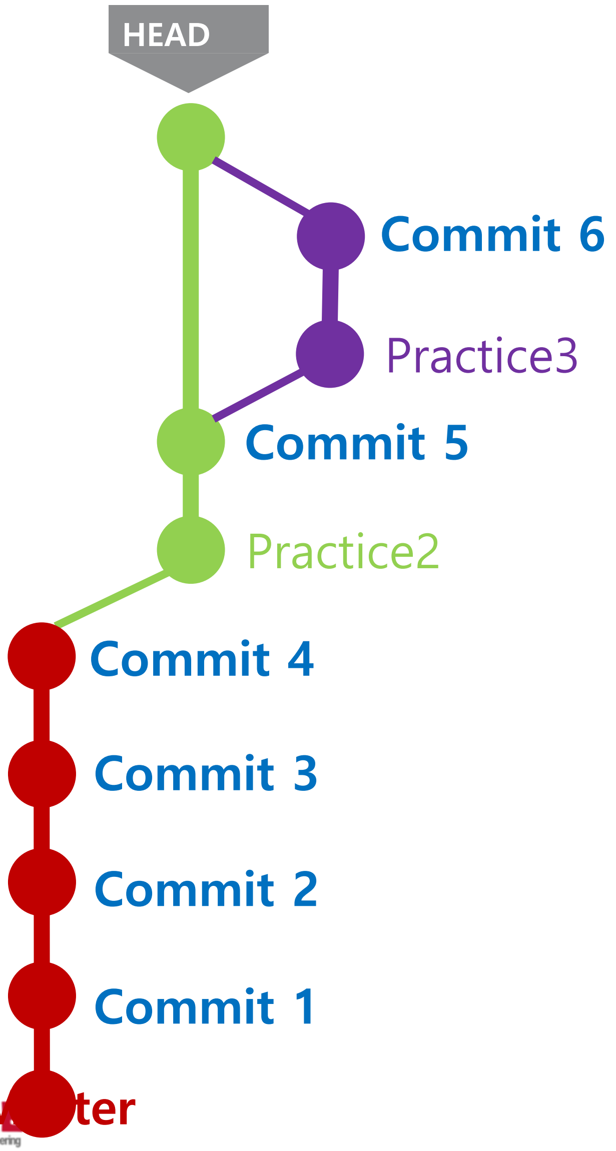
OPERATIONS BOARD



ADVANCED GIT PRACTICE

```
$ touch test_code2.c
$ git add test_code2.c
$ git commit –m “Added test_code2.c”
$ git checkout practice2
$ git status
$ git merge practice3
```

OPERATIONS BOARD



PULLING AND PUSHING

- **Good practice:**
  1. Add and Commit your changes to your local repo
  2. Pull from remote repo to get most recent changes (fix conflicts if necessary, add and commit them to your local repo)
  3. Push your changes to the remote repo

To fetch the most recent updates from the remote repo into your local repo, and put them into your working directory:

```
$ git pull origin master
```

To push your changes from your local repo to the remote repo:

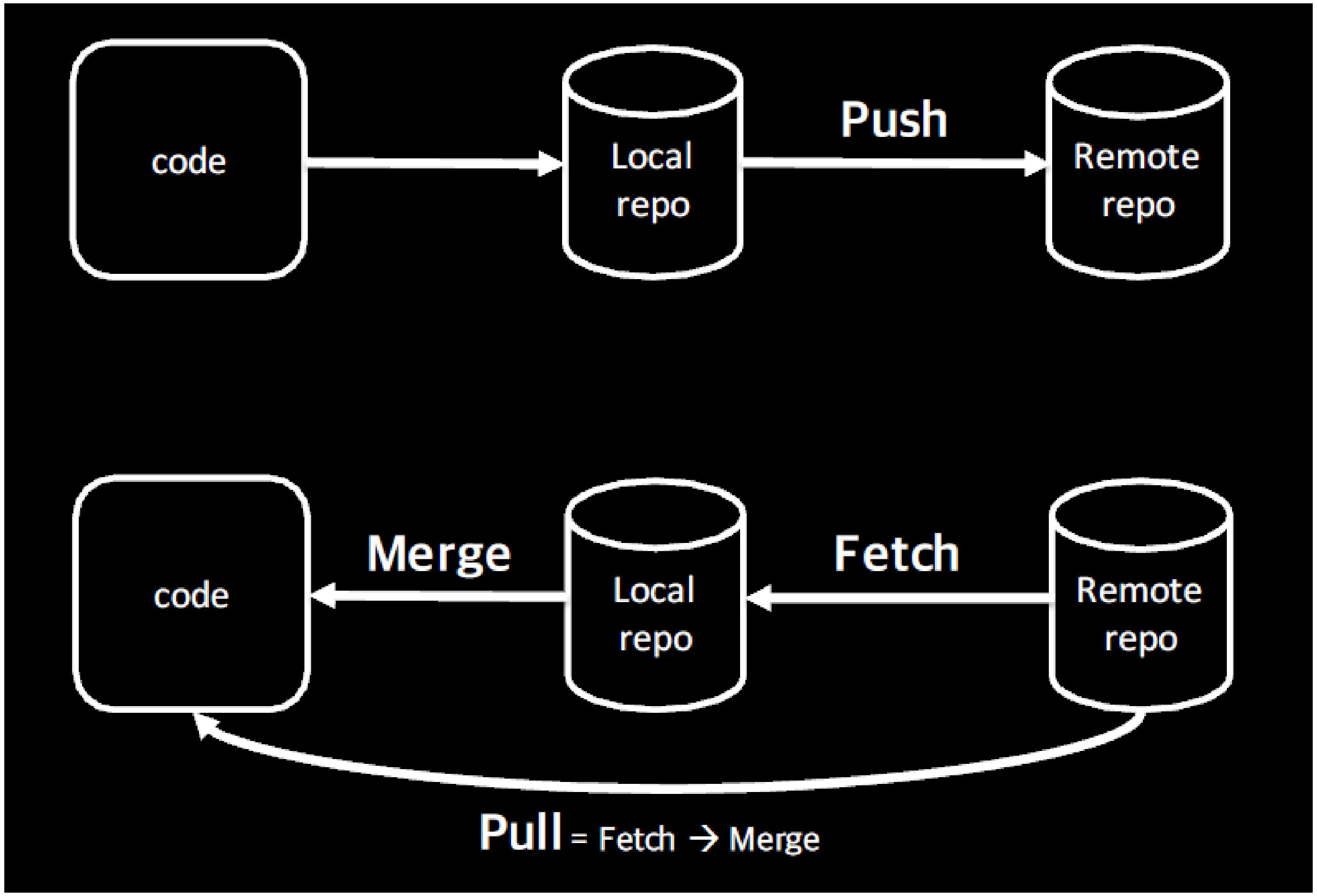
```
$ git push origin master
```

Notes: `origin` = an alias for the URL you cloned from

`master` = the local branch

SYSCORE System Security & Computer Engineering Research Lab.

PULLING AND PUSHING



GIT PRACTICE (REVIEW)

1. $ git config --global user.name “Your Name”
2. $ git config --global user.email youremail@whatever.com
3. $ git clone https://github.com/yourID/GitHub_Practice1.git
4. $ cd GitHub_Practice1

Then try:

1. $ git log, $ git log --oneline
2. Create a file named Student*ID_name*.txt
3. $ git status, $ git status -s
4. Add the file: $ git add StudentID_name.txt
5. $ git status, $ git status -s
6. Commit the file to your local repo:
   $ git commit -m “added StudentID_name.txt file”
7. $ git status, $ git status -s

Then try:

1. Pull from remote repo: $git pull origin master
2. Push to remote repo: $git push origin master

USEFUL LINKS

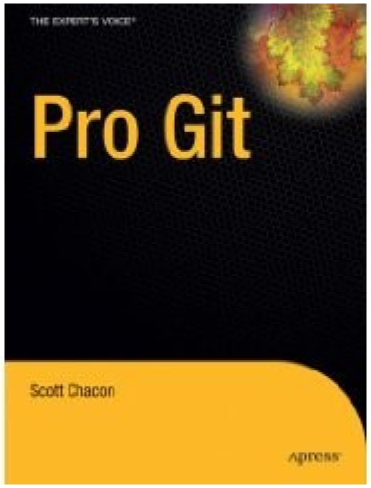
- https://try.github.io/

SUMMARY

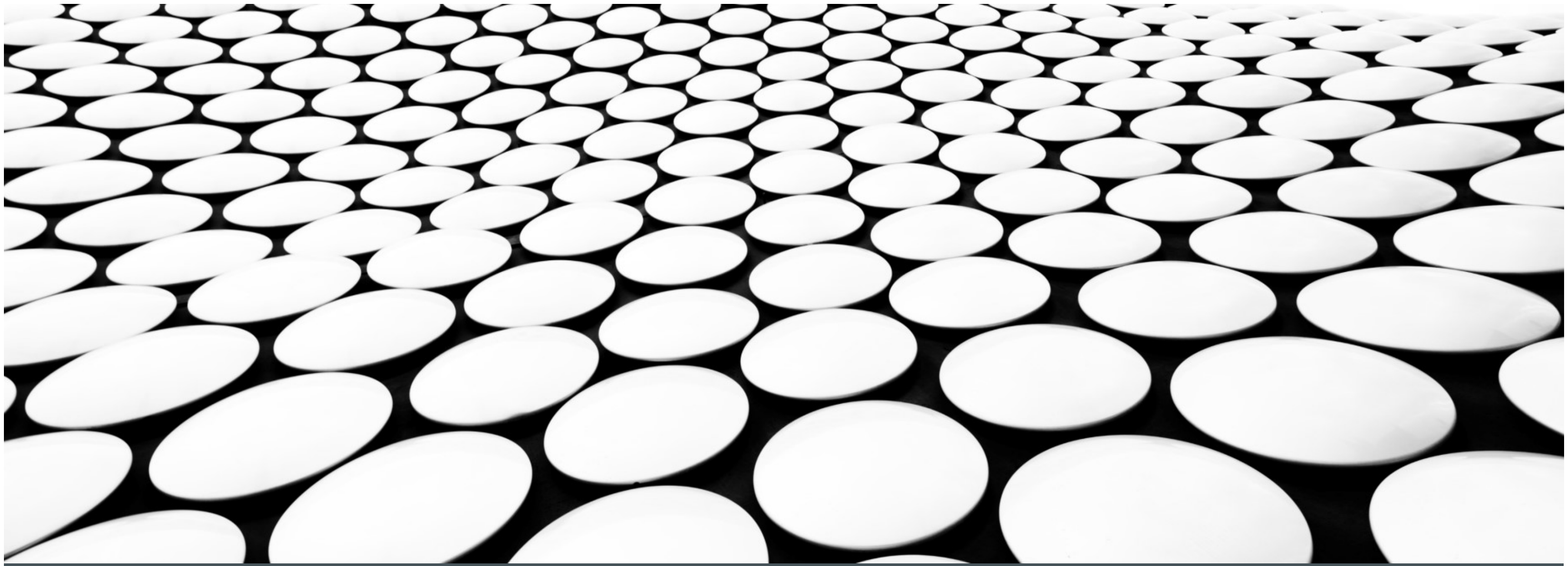
- **We covered fundamentals of Git**
  - Three trees of git
    - HEAD, INDEX and working directory
  - Basic work flow
    - Modify, stage and commit cycle
  - Branching and merging
    - Branch and merge
  - Remote
    - Add remote, push, pull, fetch
  - Other commands
    - Revert change, history view

SUMMARY

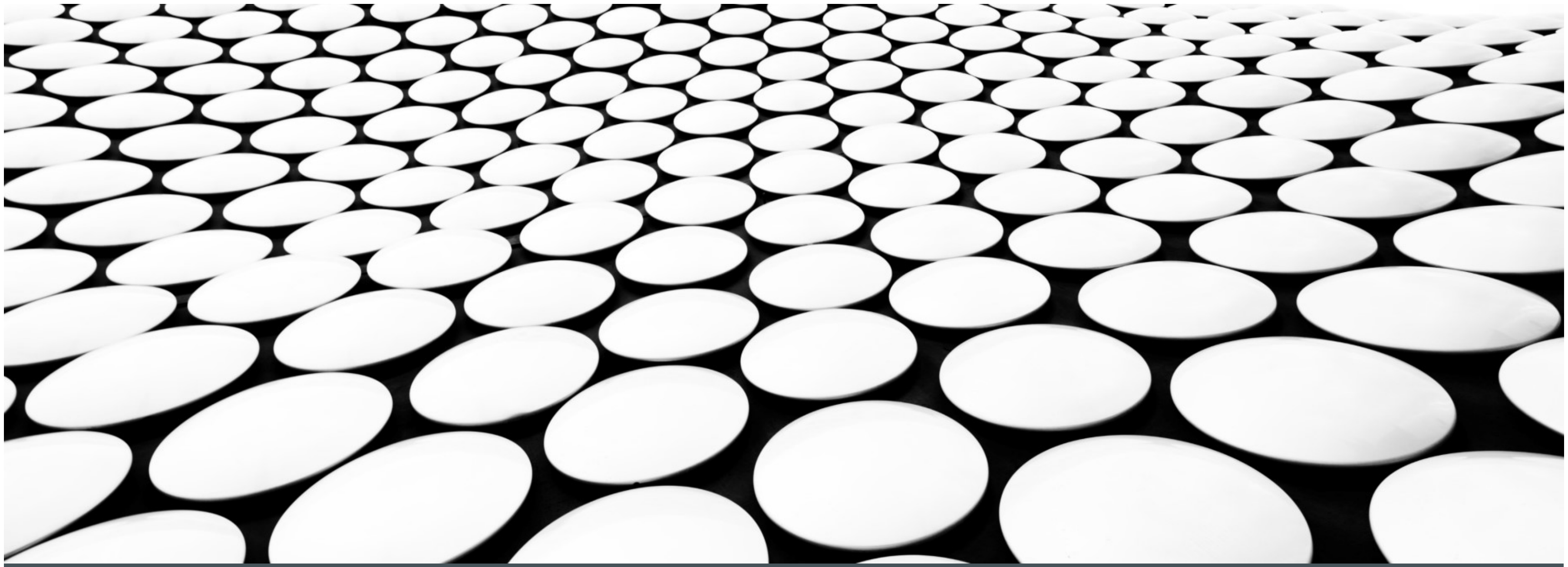
- **However, this is by no means a complete portray of git, some advanced topics are skipped:**
  - Rebasing
  - Commit amend
  - Distributed workflow
- **For more information, consult**
  - Official document
  - Pro Git
    - Free book available at http://progit.org/book/



GIT MISSION #1 (스테가노그래피)



GIT MISSION #2 (GIT)



프롤로그

- 스테가노그래피의 신기한 경험을 한 나는, 스테가노그래피는 도데체 어떻게 구현이 되었을까 호기심이 발동하였다.
- 인터넷에서 구한 스테가노그래피의 소스코드를 분석해 보며..
- 무릎을 딱 하고 치게 되는데....

미션을 수행하기 위한 준비

과제 파일 다운로드

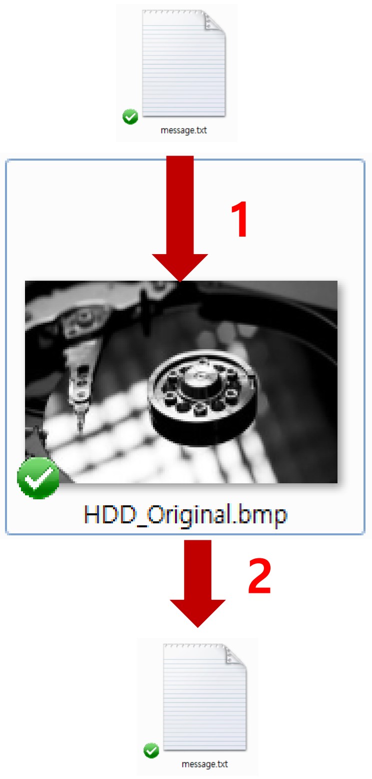
- https://github.com/woongbak/Steganography.git
- Github에 접속하여, 자신의 계정으로 Fork
  - Fork 및 Clone수행 후, 과제 수행 결과(주석)는 Pull Request를 통해 제출

파일 내용

- Steganography 폴더 (스테카노그래피 도구 소스코드)

미션 수행 순서

- 컴파일 및 툴 활용해 보기
  - Hiding a file into an image
- 컴파일 된 Steganography 툴을 이용한 메시지 추출
  - Extracting the file from the image
- 원리 분석 → Report, GitHub Pull Request



Q&A

- **1. 제공된 스테가노그래피 툴을 컴파일 하고 도구를 활용해 보자. (이미지에 데이터를 숨기고, 이를 다시 추출해 보자.) 정상적으로 동작하였는가? (화면 캡쳐를 하고 동작을 설명하시오.)**

- **2. 이번 미션에서 수행한 프로그램으로 데이터를 숨기고, 미션 1에서 사용한 툴로 데이터를 추출해 보자. 그리고 미션 1에서 사용한 툴을 이용하여 데이터를 숨기고, 이번 미션의 프로그램을 이용하여 데이터를 추출해보자. 두 가지의 경우에 대해서 각각 추출이 되는가? 그 이유는 각각 무엇인가?**

- **3. 소스코드에 대한 주석을 달고 주석이 달린 소스코드를 Pull Request 하시오. Pull Request 결과를 캡처하시오.**
  - (Hint: SteganographyHelper.cs 파일)
    - public static Bitmap embedText(string text, Bitmap bmp) → 숨기기 위한 메소드
    - Public static string extractText(Bitmap bmp) → 추출하기 위한 메소드

Q&A

- **4. 이번 미션에서 사용한 툴이 이미지에 데이터를 숨기는 원리를 소스코드를 분석하여 정리하고 이를 설명하시오.**
  - 구동원리
  - 3번에서 수행 결과(주석)와 연계하여 분석 수행

숨기는 원리

- **각 이미지의 픽셀을 루프로 돌며 RGB값을 얻는다.**
  - 예(R: 130, G: 4, B: 100)
- **각 RGB 값의 LSB를 0으로 세팅한다. → 세팅된 부분은 값을 숨기는데 활용**
- **숨길 문자를 정수로 변환한 후, 변환된 값을 해당하는 픽셀(R1, G1, B1, R2, G2, B2, R3, G3)에 인코딩 한다.**
- **하나의 문자(8 bits)가 처리되면, 다음 문자로 이동하여 위의 인코딩을 반복한다. (숨기려는 문자가 끝날 때 까지..)**
- **숨긴 데이터가 끝났다는 위치를 알려주기 위해 8bits를 0로 세팅한다.**

추출하는 원리

- 추출은 숨김의 반대~ ☺