

การทดลองที่ 1 การใช้งาน Repository เบื้องต้น

วัตถุประสงค์

1. เพื่อให้ผู้เรียนรู้และเข้าใจแนวคิดในการใช้ Repository
2. เพื่อให้ผู้เรียนสามารถใช้ Repository (Github) เบื้องต้นได้

ทฤษฎีก่อนการทดลอง

Git

Git¹ เป็นระบบควบคุมเวอร์ชัน (Version control systems) เป็นเครื่องมือที่ใช้บริหารจัดการการเปลี่ยนแปลงของไฟล์ต่าง ๆ ใน project การบันทึกการแก้ไขไฟล์แต่ละครั้งจะเรียกว่ารุ่น (revision) ซึ่งแต่ละรุ่นของการเปลี่ยนแปลงจะถูกกำกับด้วยการประทับเวลา (timestamp) และบุคคลที่ทำการเปลี่ยนแปลง ดังนั้น หากเกิดความผิดพลาดหรือเสียหายจากการแก้ไข เราก็จะสามารถย้อนเวลากลับไปยังการแก้ไขครั้งก่อนๆ ที่สมบูรณ์ได้ตามต้องการ ถือได้ว่าระบบควบคุมเวอร์ชันเป็นระบบพื้นฐานที่นิยมใช้ในการบริหารจัดการ source code ของโปรแกรม ซึ่งจริง ๆ แล้ว เราสามารถใช้ระบบควบคุมเวอร์ชันกับไฟล์ชนิดใดๆ หรืองานชนิดใดๆ ก็ได้ ไม่เฉพาะ source code ของโปรแกรมเท่านั้น ในปัจจุบัน มีระบบควบคุมเวอร์ชันให้เลือกใช้หลากหลาย ทั้งเป็นแบบฟรีและมีค่าใช้จ่าย (เช่น Git, Mercurial, Subversion) โดย Git จะได้รับความนิยมมากกว่าชนิดอื่น ๆ การทำงานของ Git นั้นจะมีพื้นที่เก็บไฟล์ ซึ่งเรียกว่า ‘repositories’ ซึ่งเราสามารถติดตั้งบริการ git บน server ใดๆ ก็ได้ แต่ server บริการ git ที่ได้รับความนิยมในปัจจุบันได้แก่ Github, Gitlab, Bitbucket เป็นต้น ข้อดีของการใช้ server รวมก็คือ สามารถแบ่งปันและร่วมมือช่วยเหลือกันในการแก้ไขโปรแกรมได้จากทุกคนทั่วโลก ลักษณะเฉพาะอย่างหนึ่งของ Git ก็คือ ใน folder ที่ชื่อ .git บนคอมพิวเตอร์ของเราจะเก็บทุกสิ่งที่เก็บบน server จึงมั่นใจได้ว่า เราสามารถทำงานกับระบบควบคุมเวอร์ชันได้ทั้งแบบออนไลน์และออฟไลน์ และหากเกิดกรณีที่ repository บน server เสียหาย เราก็สามารถนำทุกอย่างที่เก็บบนเครื่องกลับขึ้นไปเก็บบน server ได้

Github

Github เป็นบริษัทหนึ่ง ที่ให้บริการ Git repository บนพื้นฐานของเว็บ (web-based Git repository hosting) โดย Github จะให้พื้นที่เราสร้าง repository สำหรับโปรเจกต์ ให้บริการฟังก์ชันการทำงานพื้นฐานของระบบ git เช่น การ branches, merges, และ commits อีกทั้งยังให้พื้นที่สำหรับแจ้งข้อผิดพลาด บัก หรือความต้องการเพิ่มเติม features ต่างๆ ตลอดจนมีความสามารถในการเขียนคำอธิบายแบบ wiki ใน repository นั้น ๆ ด้วย Github เป็นบริษัทที่มีมูลค่าประมาณ 2 พันล้าน USD, มีผู้ใช้ประมาณ 20 ล้านคน มี repositories ประมาณ 40 ล้าน และในจำนวนเหล่านั้น มีโปรเจกต์ที่สำคัญรวมอยู่ด้วย เช่น kernel ของ Linux , source code ของ dotnet framework จากไมโครซอฟท์ และอื่นๆ ทำให้มีความมั่นใจในระดับหนึ่งว่าถ้า Github เกิดล่มขึ้นมา ก็จะมีเพื่อนร่วมชะตากรรมอีกไม่น้อย

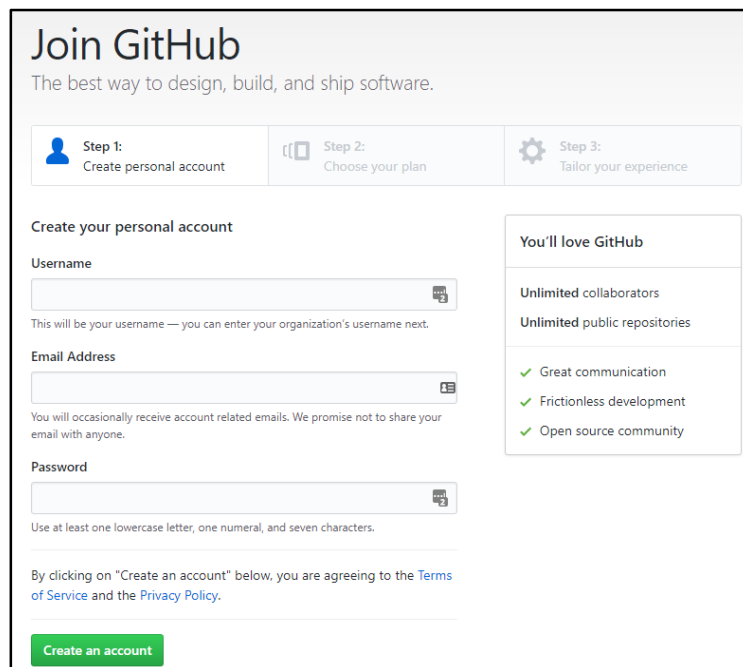
¹ "Git · GitHub." Accessed August 10, 2017. <https://github.com/git>.

1. เริ่มใช้งาน Github

ในการใช้งาน Github เราจะต้องมีบัญชีผู้ใช้ของ Github ซึ่งทาง Github จะให้บริการฟรีแบบไม่จำกัดจำนวน repository ซึ่งจะเป็นแบบ public หรือ private ก็ได้ repository แบบ public นั้น จะสามารถมองเห็นได้จากทุกคน ส่วน repository ที่เป็นแบบ private เราจะสามารถกำหนดบุคคลที่อนุญาตให้เห็น repository ของเราได้ ซึ่งจะสะดวกในการทำ project ที่เป็นความลับ

1.1 สร้างบัญชีผู้ใช้งานบน Github

การสร้างบัญชีผู้ใช้งานบน Github ให้ไปที่ <https://github.com/join> จากนั้น ให้กรอกรายละเอียด ซึ่งชื่อผู้ใช้ (User name) จะถูกนำไปใช้ในหลายๆ ที่ ดังนั้นควรเป็นชื่อที่จำง่ายและพิมพ์ได้สะดวก มิฉะนั้นจะเสียเวลาในการทำงาน



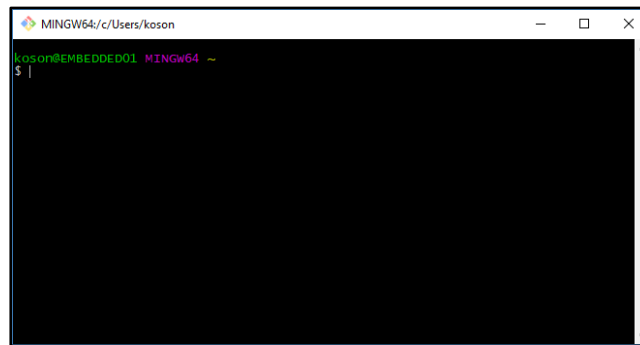
รูปที่ 1.1 การสร้างบัญชี Github

1.2 ติดตั้งโปรแกรม Git

1.2.1 ดาวน์โหลดโปรแกรม Git จาก <https://git-scm.com/downloads> โดยเลือกโปรแกรมติดตั้งให้ตรงกับระบบปฏิบัติการที่ใช้ โปรแกรมที่ดาวน์โหลดมา จะมี GUI ให้เราใช้งานด้วยซึ่งมีชื่อเรียกว่า Github desktop แต่ถ้าหากสนใจที่จะใช้ Git GUI Clients ตัวอื่นๆ ก็สามารถศึกษาได้ที่ <https://git-scm.com/downloads/guis>

1.2.2 ติดตั้งโปรแกรม Git ตามคำแนะนำของโปรแกรมติดตั้ง

1.2.3 เปิดโปรแกรม Git bash จะได้หน้าต่าง terminal ที่ทำงานใน text mode

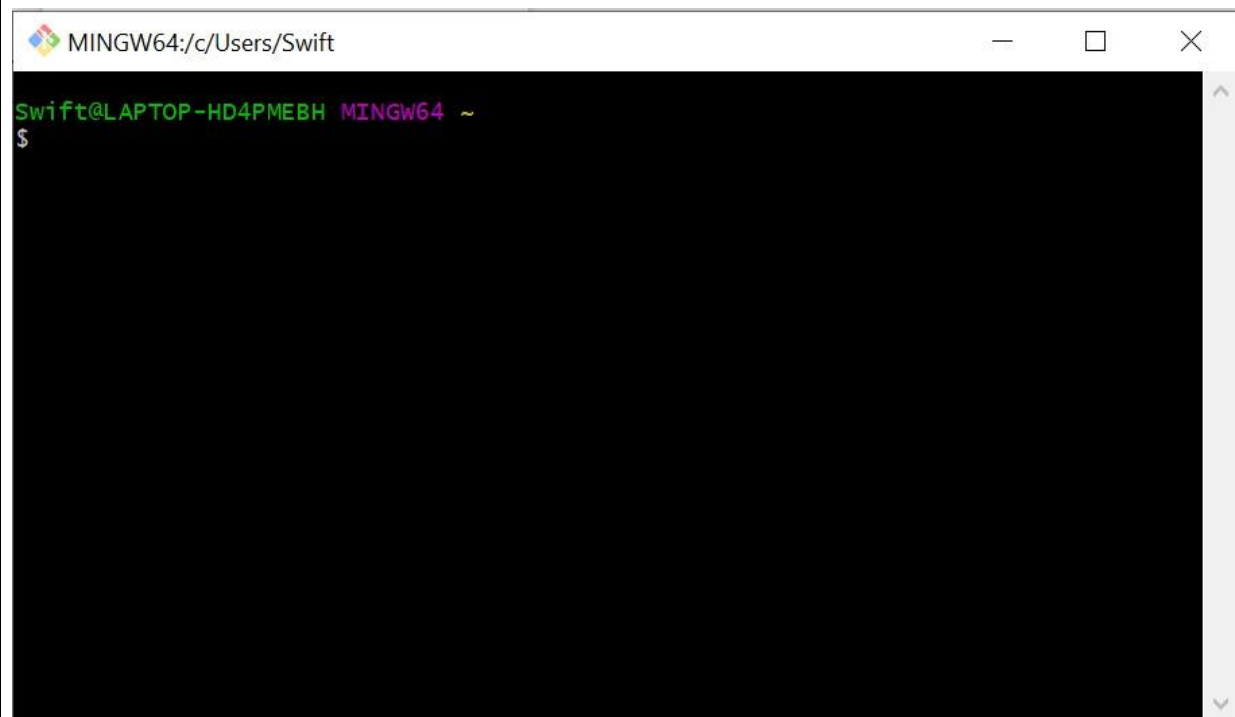


รูปที่ 1.2 หน้าต่าง terminal ของ git bash

ผลการทดลอง

ให้นักศึกษา capture หน้าจอของตนเองมาใส่ และอธิบายสั้นๆ ถึงสิ่งที่เกิดขึ้น

- เมื่อกดเปิดโปรแกรมจะขึ้นหน้าต่าง terminal ของ git bash



1.2.4 ทดสอบว่าสามารถใช้งาน Git บนเครื่องของเราได้หรือไม่ ให้พิมพ์คำสั่งต่อไปนี้

```
$ git
```

ถ้า terminal ตอบกลับมาว่าไม่รู้จักคำสั่ง git แสดงว่าการติดตั้งยังไม่สมบูรณ์ ให้กลับไปตรวจสอบขั้นตอน 1.2.2 ให้ติดตั้งเรียบร้อย

```
MINGW64/c/Users/koson
koson@EMBEDDED01 MINGW64 ~
$ git
usage: git [--version] [--help] [-c <path>] [-c name=value]
        [--exec-path=<path>] [--html-path] [--man-path] [--info-path]
        [-p | --paginate | --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
  init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  reset      Reset current HEAD to the specified state
  rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
  bisect     Use binary search to find the commit that introduced a bug
  grep       Print lines matching a pattern
  log        Show commit logs
  show       Show various types of objects
  status     Show the working tree status

grow, mark and tweak your common history
  branch     List, create, or delete branches
  checkout   Switch branches or restore working tree files
  commit     Record changes to the repository
  diff       Show changes between commits, commit and working tree, etc
  merge      Join two or more development histories together
  rebase     Reapply commits on top of another base tip
  tag        Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
  fetch      Download objects and refs from another repository
```

รูปที่ 1.3 ผลการทดลองพิมพ์คำสั่ง git

ผลการทดลอง

ให้นักศึกษา capture หน้าจอของตนเองมาใส่ และอธิบายสั้นๆ ถึงสิ่งที่เกิดขึ้น

- เมื่อใส่คำสั่ง git ลงไป จะแสดงข้อมูลเกี่ยวกับคำสั่งต่าง ๆ ที่ตัวโปรแกรม git สามารถทำได้

```
MINGW64/c/Users/Swift
Swift@LAPTOP-HD4PMEBH MINGW64 ~
$ 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>]
        [--super-prefix=<path>] [--config-env=<name>=<envvar>]
        <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
  init       Create an empty Git repository or reinitialize an existing
one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  restore    Restore working tree files
  rm         Remove files from the working tree and from the index
  sparse-checkout  Initialize and modify the sparse-checkout

examine the history and state (see also: git help revisions)
  bisect     Use binary search to find the commit that introduced a bug
  diff       Show changes between commits, commit and working tree, etc
  grep       Print lines matching a pattern
  log        Show commit logs
  show       Show various types of objects
  status     Show the working tree status

grow, mark and tweak your common history
  branch     List, create, or delete branches
  commit     Record changes to the repository
  merge      Join two or more development histories together
  rebase     Reapply commits on top of another base tip
  reset      Reset current HEAD to the specified state
  switch     Switch branches
  tag        Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
  fetch      Download objects and refs from another repository
  pull       Fetch from and integrate with another repository or a local
branch
  push       Update remote refs along with associated objects

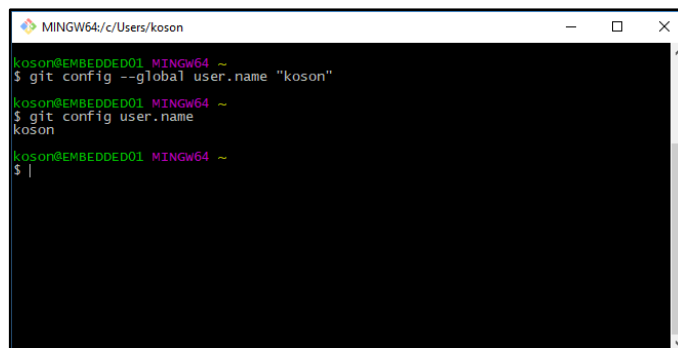
'git help -a' and 'git help -p' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'.
```

1.2.5 บอกให้ Git รู้จักชื่อของเรา โดยพิมพ์คำสั่งต่อไปนี้²

```
$ git config --global user.name "USER NAME"
```

ในกรณีที่เรต้องการทราบชื่อผู้ใช้ปัจจุบัน สามารถสั่งให้ Git รายงานออกมาด้วยการพิมพ์คำสั่งต่อไปนี้

```
$ git config user.name
```



A screenshot of a terminal window titled 'MINGW64: c:/Users/koson'. The terminal shows the following commands and output:

```
koson@EMBEDDED01 MINGW64 ~$ git config --global user.name "koson"
koson@EMBEDDED01 MINGW64 ~$ git config user.name
koson
koson@EMBEDDED01 MINGW64 ~$
```

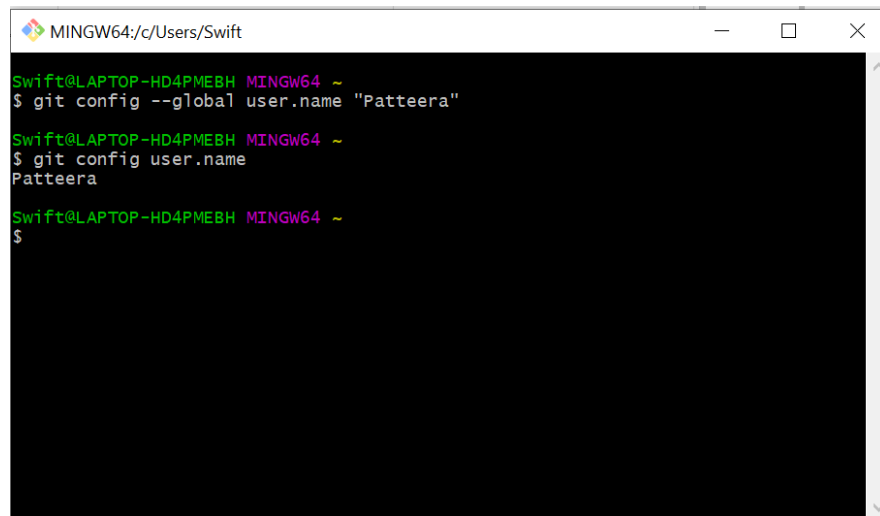
รูปที่ 1.4 git config --global user.name

ผลการทดลอง

² "Setting your username in Git - User Documentation - GitHub Help." Accessed August 10, 2017. <https://help.github.com/articles/setting-your-username-in-git/>.

ให้นักศึกษา capture หน้าจอของตนเองมาใส่ และอธิบายสั้นๆ ถึงสิ่งที่เกิดขึ้น

- เมื่อพิมพ์คำสั่งตามภาพแล้ว ตัว git จะจดจำชื่อที่เราใส่ลงไป ใน command line เมื่อใช้คำสั่ง git config user.name จะแสดงชื่อที่เราได้ทำการตั้งไว้ก่อนหน้าดังภาพ



```
MINGW64:/c/Users/Swift
Swift@LAPTOP-HD4PMEBH MINGW64 ~
$ git config --global user.name "Patteera"
Swift@LAPTOP-HD4PMEBH MINGW64 ~
$ git config user.name
Patteera
Swift@LAPTOP-HD4PMEBH MINGW64 ~
$
```

1.2.6 บอกให้ Git รู้จัก email ของเรา โดยพิมพ์คำสั่งต่อไปนี้

```
$ git config --global user.email "USER EMAIL ADDRESS"
```

ในกรณีที่เรต้องการทราบชื่อผู้ใช้ปัจจุบัน สามารถสั่งให้ Git รายงานออกมาด้วยการพิมพ์คำสั่งต่อไปนี้

```
$ git config user.email
```

หมายเหตุ email ที่ใช้จะต้องตรงกับ email ที่ลงทะเบียนไว้กับ Github มิฉะนั้นจะไม่สามารถเขียนข้อมูลขึ้นไปบน server ได้

เมื่อทำในขั้นตอน 1.2.5 และ 1.2.6 เรียบร้อยแล้ว การทำงานใดๆ บน Github ก็จะสามารถชื่อและ Email ของเรากับไว้เสมอ

```
MINGW64/c/Users/koson
koson@EMBEDDED01 MINGW64 ~
$ git config --global user.email "koson.trachu@gmail.com"
koson@EMBEDDED01 MINGW64 ~
$ git config user.email
koson.trachu@gmail.com
koson@EMBEDDED01 MINGW64 ~
$ |
```

รูปที่ 1.5 git config --global user.email

ผลการทดลอง

ให้นักศึกษา capture หน้าจอของตนเองมาใส่ และอธิบายสั้นๆ ถึงสิ่งที่เกิดขึ้น

- เมื่อพิมพ์คำสั่งตามภาพ ตัว git จะทำการจดจำชื่อ และ email ของเรา เพื่อนำไปใช้เวลาที่เรานำข้อมูลขึ้นบน server

```
MINGW64/c/Users/Swift
Swift@LAPTOP-HD4PMEBH MINGW64 ~
$ git config --global user.email "63030063@kmitl.ac.th"
Swift@LAPTOP-HD4PMEBH MINGW64 ~
$ git config user.email
63030063@kmitl.ac.th
Swift@LAPTOP-HD4PMEBH MINGW64 ~
$ |
```

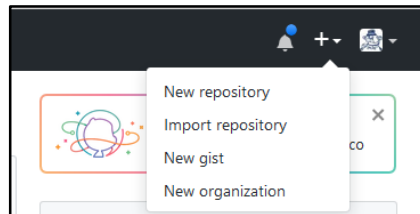
1.3 สร้าง repository (บน server)

Repository เป็นพื้นที่สำหรับเก็บ project ของเรา ซึ่งไม่ได้หมายความว่าเฉพาะ source code เท่านั้น repository ยังสามารถประกอบด้วยไฟล์ทุกชนิด ไม่ว่าจะเป็น Word Document, spread sheet, presentation, เอกสารการออกวิเคราะห์และออกแบบซอฟต์แวร์ ไฟล์มีเดียภาพและเสียง รวมไปถึงเอกสาร Wiki ในลักษณะ html ด้วย ดังนั้น ในการทำโครงการพัฒนาซอฟต์แวร์ เราสามารถนำทุกสิ่งที่เป็นจำเป็นสำหรับการทำงาน มาใส่ไว้ใน repository และเมื่อเพื่อนร่วมทีมหรือ user ใดๆ ทำสำเนา

repository ของเราไป เขาก็จะได้ทุกสิ่งทุกอย่างไปอย่างครบถ้วน ดังนั้นจึงอาจพูดได้ว่าเราสามารถใช่ repository เป็นเครื่องมือบริหารโครงการที่มีประสิทธิภาพได้เช่นกัน

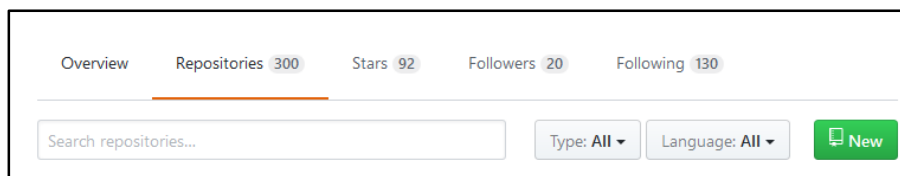
1.3.1 การสร้าง repository บน Github สามารถสร้างได้หลายวิธีด้วยกัน เช่น

(1) การสร้าง repository โดยการคลิกที่ปุ่มเครื่องหมาย “+” ที่ด้านบนขวาของหน้าเพจ Github แล้วเลือก new repository



รูปที่ 1.6 การสร้าง repository โดยการคลิกที่ปุ่มเครื่องหมาย “+”

(2) การสร้าง repository โดยการคลิกที่ปุ่ม New สีเขียว



รูปที่ 1.7 การสร้าง repository โดยการคลิกที่ปุ่ม New

(3) การสร้าง repository โดยลิงค์ <https://github.com/new>

นอกจาก 3 วิธีข้างต้น ซึ่งเราจะไปสร้าง repository บนเว็บแล้ว เรายังสามารถสร้าง repository โดยใช้ command line บน terminal (ศึกษาได้จาก [adding-an-existing-project-to-github-using-the-command-line](#)³)

1.3.2 กำหนดชื่อและชนิดของ repository

การใช้วิธีการ 3 วิธีแรก ในข้อ 1.3.1 จะได้ผลอย่างเดียวกัน คือ Github จะพามาหน้าสำหรับสร้าง repository

- ในช่อง **Repository name** ให้ใส่ชื่อของ repository เนื่องจากบ่อยครั้งที่เราต้องใช้งานคำสั่งต่าง ๆ บน terminal ซึ่งต้องพิมพ์ชื่อ repository เอง ดังนั้นชื่อของ repository จะต้องมีความหมายในตัว เข้าใจง่าย กระชับ
- ในช่อง **Description (optional)** เพิ่มคำอธิบายสั้นๆ เกี่ยวกับ repository เพื่อให้ชาวโลกอ่านแล้วเห็นภาพรวมของ repository ได้อย่างรวดเร็ว

³["Adding an existing project to GitHub using" Accessed August 11, 2017.
https://help.github.com/articles/adding-an-existing-project-to-github-using-the-command-line/](https://help.github.com/articles/adding-an-existing-project-to-github-using-the-command-line/)

- ชนิดของ **repository** นั้น ถ้าหากเป็นโปรเจกต์ที่เป็นความลับ ไม่อาจเปิดเผยต่อชาวโลกได้ เช่นประกอบด้วยฐานข้อมูลในงานวิจัย คณะนแล็บของนักศึกษา ชื่อ URL, user name, password ที่เขียนลงไป source code เราก็อาจจะเลือกเป็น private ซึ่งอาจจะต้องมีค่าใช้จ่ายในการสมัครสมาชิกพิเศษ หรือไม่ก็ต้องเป็น academic account ในที่นี้ให้เลือกเป็น public
- ถ้าเราทำเครื่องหมาย ☐ หน้าข้อความ Initialize this repository with a README เพื่อให้เราสามารถเขียนบรรยายคร่าวๆ เกี่ยวกับ repository ได้
 - **เดี๋ยวก่อน..... ในขั้นตอนนี้ ยังไม่ต้องทำเครื่องหมาย ☐ เพราะเราจะทดลองสร้างโดยใช้ command line tool**
- เลือกว่าจะเพิ่ม .gitignore หรือ license file ด้วยหรือไม่ โดย .gitignore นี้จะบอก Git ว่าไม่ต้องสนใจที่จะติดตามไฟล์ชนิดใดบ้าง โดย Git จะกำหนดชนิดของไฟล์ให้เบื้องต้น เช่น ถ้าเราเลือก .gitignore เป็น ภาษา C++ แล้ว Git จะเพิ่มชนิดของไฟล์ต่างๆ ที่เป็นผลจากการคอมไพล์ไว้ในรายการที่เพิกเฉย (เช่น ไฟล์ที่มีนามสกุล .exe) ซึ่งไฟล์เหล่านั้น มักจะเกิดจากการคอมไพล์โปรแกรม ไม่ใช่ไฟล์ที่เราเป็นคนแก้ไข source code จึงไม่จำเป็นที่จะต้องนำไปเก็บบน repository ให้สิ้นเปลืองพื้นที่ สามารถดูเทมเพลตของ .gitignore ได้จาก A collection of useful .gitignore templates⁴
 - **ยังไม่ต้องเลือก.gitignore เช่นเดียวกัน**

Create a new repository

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

Repository template

Start your repository with a template repository's contents.

No template ▾

Owner * Repository name *

koston / CL64-01 ✓

Great repository names are short and memorable. Need inspiration? How about **potential-palm-tree**?

Description (optional)

Computer Laboratory 2564-01

☒ 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.

Initialize this repository with:

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

☐ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

☐ Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

☐ Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

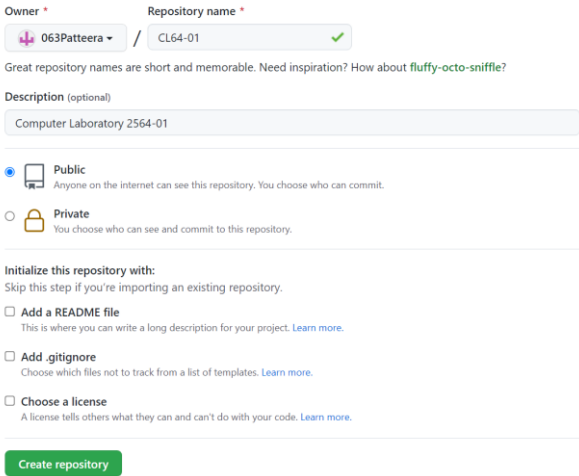
⁴"A collection of useful .gitignore templates" Accessed August 11, 2017.
<https://github.com/github/gitignore>.

รูปที่ 1.8 การสร้าง repository

ผลการทดลอง

ให้นักศึกษา capture หน้าจอของตนเองมาใส่ และอธิบายสั้นๆ ถึงสิ่งที่เกิดขึ้น

- เมื่อทำตามภาพจะขึ้นหน้าต่างเว็บให้เรากำหนดชื่อและกำหนดค่าต่าง ๆ ของ repository



Owner * Repository name *

063Patteera / CL64-01

Great repository names are short and memorable. Need inspiration? How about fluffy-octo-sniffle?

Description (optional)

Computer Laboratory 2564-01

☒ 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.

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

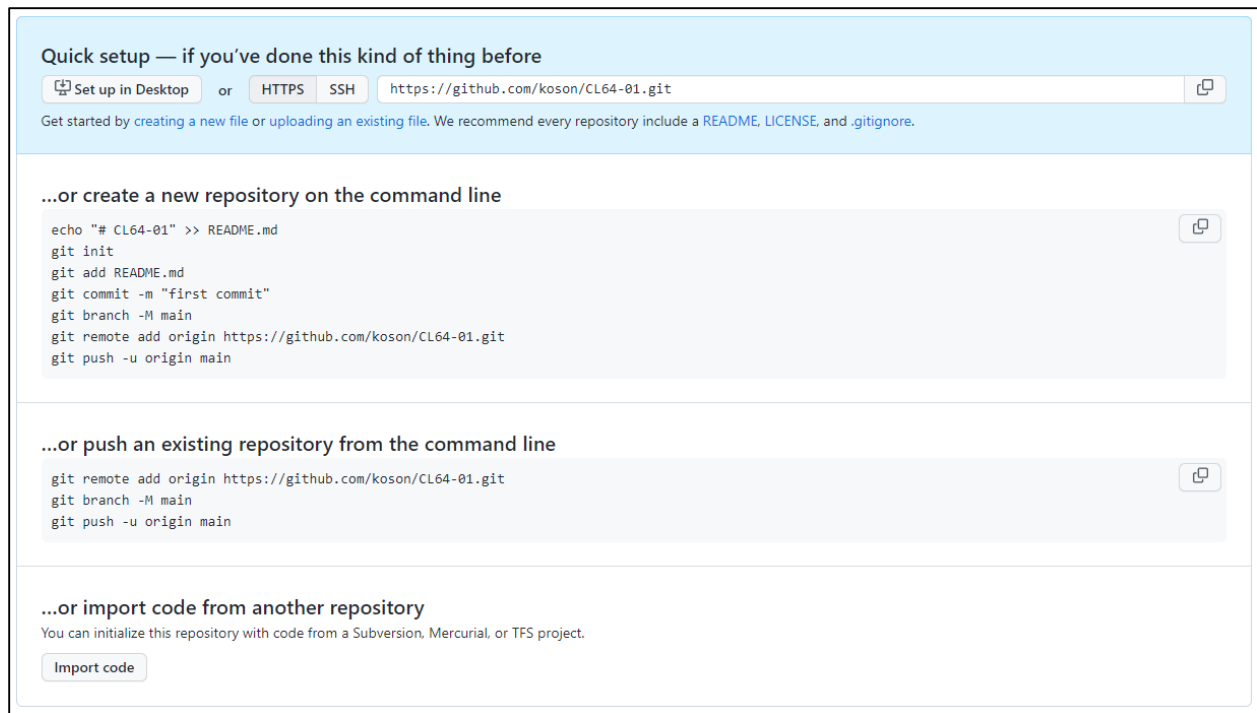
☐ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

☐ Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

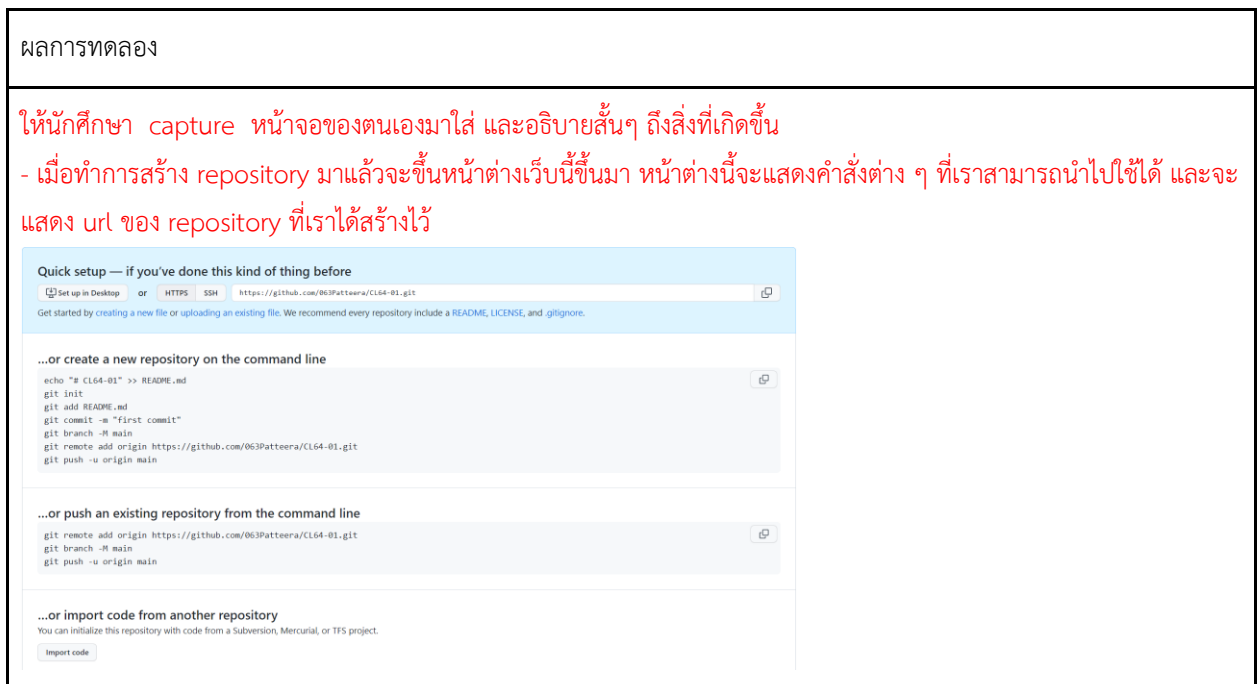
☐ Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

- คลิปุ่ม Create repository สีเขียว
Github จะสร้าง repository ให้ตามต้องการ



รูปที่ 1.9 repository ที่ได้จากการสร้างในข้อ 1.3



หมายเหตุ ให้เปิดหน้าเพจนี้ค้างไว้ เพราะเราต้องมาดูผลการเปลี่ยนแปลงในภายหลัง

1.4 สร้าง git บนเครื่องคอมพิวเตอร์ (Local)

Repository ที่สร้างขึ้นในข้อ 1.3 นั้น เป็น repository ที่อยู่บน server ในขณะที่เรากำลังแก้ไข source code ซึ่งมักจะเป็นการแก้ไขเล็ก ๆ น้อย ๆ การทำงานของ git จะเน้นทำงานที่ local เป็นหลัก ต่อเมื่อเราได้พัฒนา source code จนถึงจุดหนึ่ง ที่คิดว่าสามารถเผยแพร่ เพื่อการทดสอบหรือใช้งาน เราจึงส่งขึ้นไปเก็บบน server

การทำสำเนาของ repository มาไว้บนเครื่อง (local) สามารถทำได้หลายวิธี ซึ่งเบื้องต้นนี้ เราจะศึกษาโดยการใช้งาน command line ซึ่งอาจจะพบกับความยุ่งยากบ้างในตอนแรกๆ แต่เมื่อใช้บ่อย ๆ จนชำนาญจะพบว่ามีความยืดหยุ่นสูงกว่าการใช้ GUI Clients หรือเมื่อศึกษาจนเข้าใจแล้วหันไปใช้ GUI Clients ก็จะสามารถเข้าใจถึงการทำงานของระบบ Git อย่างแท้จริง

1.4.1 การ clone repository ด้วย command line (git bash)

1) การเตรียมการเบื้องต้น

- ในหน้าต่าง git bash ให้พิมพ์คำสั่ง list ดูรายการของไฟล์และโฟลเดอร์

```
$ ls
```

เราจะได้เห็นรายการไฟล์ถูกแสดงขึ้นมา

ผลการทดลอง

ให้นักศึกษา capture หน้าจอของตนเองมาใส่ และอธิบายสั้นๆ ถึงสิ่งที่เกิดขึ้น
- มีไฟล์ต่างๆอยู่ในโฟลเดอร์

```
MINGW64:/c/Users/Swift

Swift@LAPTOP-HD4PMEBH MINGW64 ~
$ ls
AppData/
Desktop/
Documents/
Downloads/
Favorites/
IntelGraphicsProfiles/
NTUSER.DAT
NTUSER.DAT{3e8241a2-b003-11eb-98e8-f327502e606c}.TM.b1f
NTUSER.DAT{3e8241a2-b003-11eb-98e8-f327502e606c}.TMContainer000000000000000000
1.regtrans-ms
NTUSER.DAT{3e8241a2-b003-11eb-98e8-f327502e606c}.TMContainer000000000000000000
2.regtrans-ms
Pictures/
'Saved Games'/
Videos/
ntuser.dat.LOG1
ntuser.dat.LOG2
ntuser.ini

Swift@LAPTOP-HD4PMEBH MINGW64 ~
$
```

- สร้างโฟลเดอร์สำหรับเก็บงานในวิชาการทดลอง (ในที่นี้ชื่อว่า CL2564 ย่อมาจาก Computer Laboratory 2564) โดยใช้คำสั่ง

```
$ mkdir CL2564
```

- ย้ายเข้าไปอยู่ในโฟลเดอร์ที่สร้างขึ้น โดยใช้คำสั่ง

```
$ cd CL2564
```

สังเกตได้จาก git bash จะแสดงชื่อของโฟลเดอร์ปัจจุบันเป็นดังรูปที่ 1.10

```
MINGW64~/f/GitHubRepos/Computerlab-2564/CL2564
kson@DESKTOP-CPUNNOU MINGW64 /f/GitHubRepos/Computerlab-2564
$ mkdir CL2564
kson@DESKTOP-CPUNNOU MINGW64 /f/GitHubRepos/Computerlab-2564
$ cd CL2564
kson@DESKTOP-CPUNNOU MINGW64 /f/GitHubRepos/Computerlab-2564/CL2564
$ |
```

รูปที่ 1.10 หน้าต่าง terminal ของ git bash เตรียมพร้อมสำหรับการ clone

ผลการทดลอง

ให้นักศึกษา capture หน้าจอของตนเองมาใส่ และอธิบายสั้นๆ ถึงสิ่งที่เกิดขึ้น

- ทำการสร้างโฟลเดอร์ไว้ในโฟลเดอร์ Git ชื่อว่า CL2564 และทำการเปิด path ของโฟลเดอร์ CL2564

```
MINGW64:/c/Users/Swift/CL2564
Swift@LAPTOP-HD4PMEBH MINGW64 ~
$ mkdir CL2564
Swift@LAPTOP-HD4PMEBH MINGW64 ~
$ cd CL2564
Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564
$
```

2) การทำสำเนา repository มาไว้บนเครื่องโดยการ clone

- ทำสำเนา repository มาไว้บนเครื่องโดยใช้คำสั่งที่มีรูปแบบดังต่อไปนี้

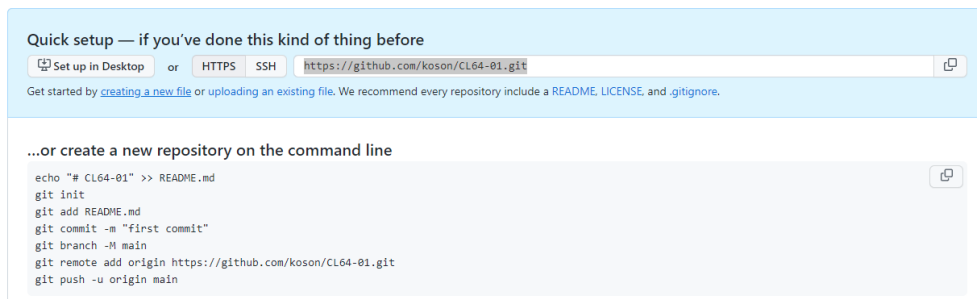
```
$ git clone https://github.com/YOUR_USERNAME/YOUR_REPOSITORY_NAME
```

[YOUR USERNAME] คือ username ของเราบน github

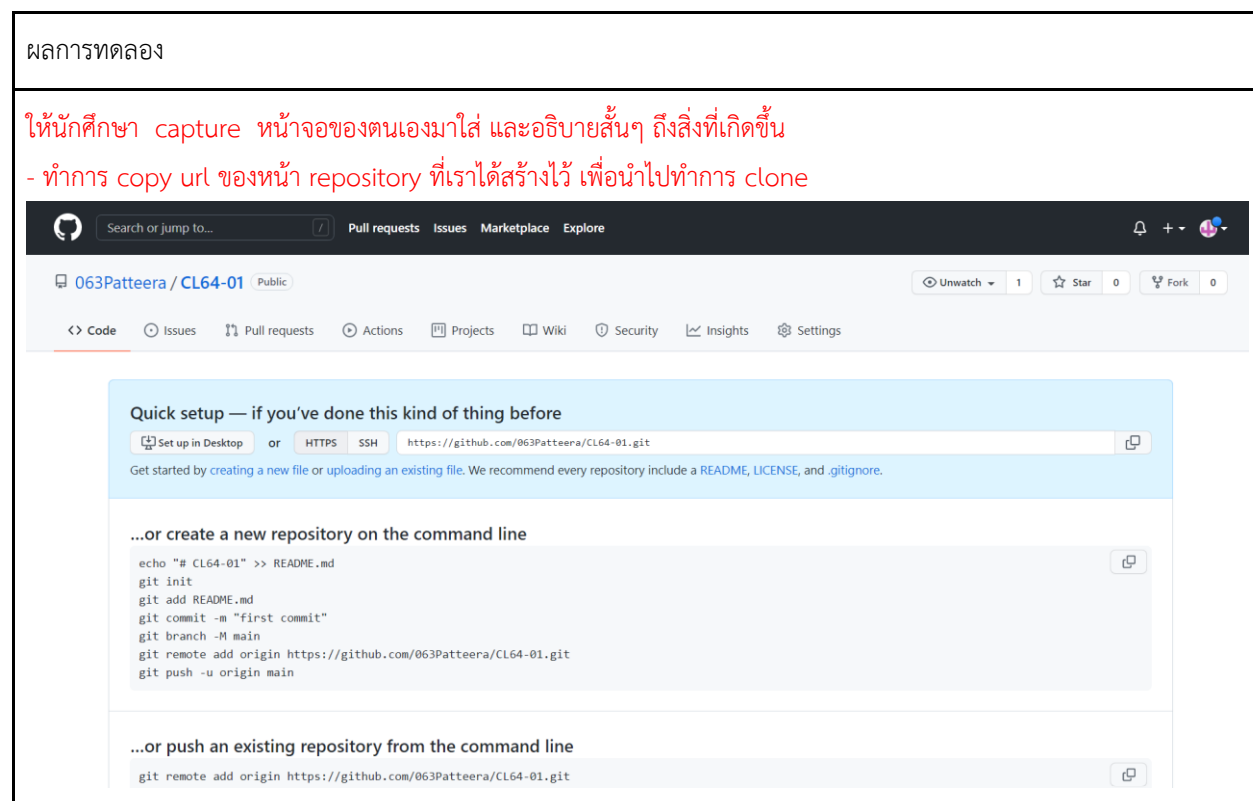
[YOUR REPOSITORY NAME] คือชื่อ repository ของเราที่สร้างในข้อ 1.3

ถ้าจำไม่ได้ ก็ไม่เป็นไร ให้เข้าไปที่ repository ที่เพิ่งสร้างบน Github (ดูรูปที่ 1.9) จะเห็นว่า URL ของ repository สำหรับการ

โคลน ดังรูปที่ 1.11 ให้เรากดปุ่ม copy ที่อยู่ด้านขวามือของ url



รูปที่ 1.11 URL สำหรับการ clone repository



- ใน git bash ให้พิมพ์คำสั่ง git clone ตามด้วย URL ที่คัดลอกมา
- เมื่อทำการ clone เรียบร้อย จะได้ผลดังรูปที่ 1.12

```
MINGW64:/f/GitHubRepos/Computerlab-2564/CL2564
koson@DESKTOP-CPUNNOU MINGW64 /f/GitHubRepos/Computerlab-2564
$ mkdir CL2564

koson@DESKTOP-CPUNNOU MINGW64 /f/GitHubRepos/Computerlab-2564
$ cd CL2564

koson@DESKTOP-CPUNNOU MINGW64 /f/GitHubRepos/Computerlab-2564/CL2564
$ git clone https://github.com/koson/CL64-01.git
Cloning into 'CL64-01'...
warning: You appear to have cloned an empty repository.

koson@DESKTOP-CPUNNOU MINGW64 /f/GitHubRepos/Computerlab-2564/CL2564
$ |
```

รูปที่ 1.12 ผลการ clone repository

ผลการทดลอง

ให้นักศึกษา capture หน้าจอของตนเองมาใส่ และอธิบายสั้นๆ ถึงสิ่งที่เกิดขึ้น

- ทำการ clone ข้อมูลที่อยู่ใน repository ลงในโฟลเดอร์เครื่องของเรา

```
MINGW64:/c/Users/Swift/CL2564

Swift@LAPTOP-HD4PMEBH MINGW64 ~
$ git clone https://github.com/063Patteera/CL64-01.git
Cloning into 'CL64-01'...
warning: You appear to have cloned an empty repository.

Swift@LAPTOP-HD4PMEBH MINGW64 ~
$ cd CL2564

Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564
$
```

- เรียกดูรายการโฟลเดอร์ (ด้วยคำสั่ง ls) และเปลี่ยนโฟลเดอร์ (ด้วยคำสั่ง change directory :cd)

ผลการทดลอง

ให้นักศึกษา capture หน้าจอของตนเองมาใส่ และอธิบายสั้นๆ ถึงสิ่งที่เกิดขึ้น

- ใช้คำสั่ง ls เพื่อดูข้อมูลที่อยู่ในโฟลเดอร์ CL2564 ซึ่งจะเจอโฟลเดอร์ CL64-01 ที่ทำการ clone มา ให้ใช้คำสั่ง cd CL64-01 เพื่อเข้าถึง path ของโฟลเดอร์ CL64-01

MINGW64:/c/Users/Swift/CL2564/CL64-01

```
Swift@LAPTOP-HD4PMEBH MINGW64 ~  
$ cd CL2564  
  
Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564  
$ git clone https://github.com/063Patteera/CL64-01.git  
Cloning into 'CL64-01'...  
warning: You appear to have cloned an empty repository.  
  
Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564  
$ ls  
CL64-01/  
  
Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564  
$ cd CL64-01  
  
Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564/CL64-01 (master)  
$ |
```

ตอนแรกจะพบว่าไม่มีโฟลเดอร์ชื่อ CL64-01 ซึ่งถูก clone มาจาก server จึงย้ายเข้าไปในโฟลเดอร์นั้น แล้วจึงสั่ง ls เพื่อดูรายการไฟล์ พบว่า repository ของเรายังว่างเปล่า ดังรูปที่ 1.13

รูปที่ 1.13 ไฟล์ที่ถูก clone มาจาก repository

ผลการทดลอง

ให้นักศึกษา capture หน้าจอของตนเองมาใส่ และอธิบายสั้นๆ ถึงสิ่งที่เกิดขึ้น

- ทำการดูข้อมูลในโฟลเดอร์ CL64-01

MINGW64:/c/Users/Swift/CL2564/CL64-01

```
Swift@LAPTOP-HD4PMEBH MINGW64 ~  
$ cd CL2564  
  
Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564  
$ git clone https://github.com/063Patteera/CL64-01.git  
Cloning into 'CL64-01'...  
warning: You appear to have cloned an empty repository.  
  
Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564  
$ ls  
CL64-01/  
  
Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564  
$ cd CL64-01  
  
Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564/CL64-01 (master)  
$ ls  
  
Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564/CL64-01 (master)  
$ |
```

- ให้พิมพ์คำสั่งต่อไปนี้ ครั้งละบรรทัด (พิมพ์ให้ครบบรรทัดแล้วเคาะ enter)

```
echo "# CL64-01" >> README.md  
git init  
git add README.md  
git commit -m "first commit"  
git branch -M main  
git remote add origin https://github.com/koson/CL64-01.git  
git push -u origin main
```

รูปที่ 1.14 การเพิ่มไฟล์ README.md ให้กับ repository

ผลการทดลอง

ให้นักศึกษา capture หน้าจอของตนเองมาใส่ และอธิบายสั้นๆ ถึงสิ่งที่เกิดขึ้น
- ทำการเพิ่มไฟล์ README.md ในโฟลเดอร์ CL64-01 และทำการตั้งค่าต่าง ๆ

MINGW64:/c/Users/Swift

```
Swift@LAPTOP-HD4PMEBH MINGW64 ~
$ echo "# CL64-01" >> README.md

Swift@LAPTOP-HD4PMEBH MINGW64 ~
$ git init
Initialized empty Git repository in C:/Users/Swift/.git/

Swift@LAPTOP-HD4PMEBH MINGW64 ~ (master)
$ git add README.md
warning: LF will be replaced by CRLF in README.md.
The file will have its original line endings in your working directory

Swift@LAPTOP-HD4PMEBH MINGW64 ~ (master)
$ git commit -m "first commit"
[master (root-commit) f9fb210] first commit
1 file changed, 1 insertion(+)
create mode 100644 README.md

Swift@LAPTOP-HD4PMEBH MINGW64 ~ (master)
$ git branch -M main

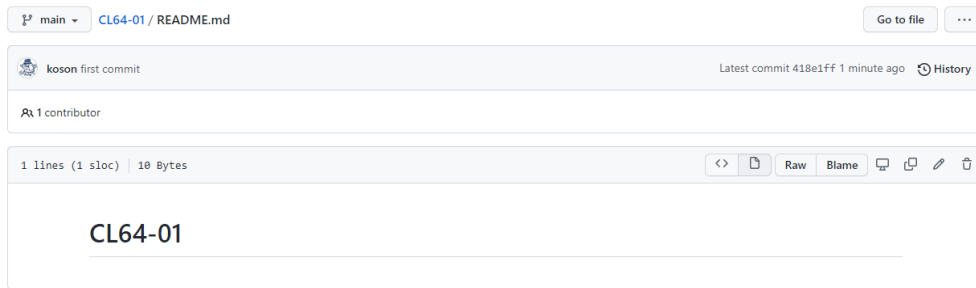
Swift@LAPTOP-HD4PMEBH MINGW64 ~ (main)
$ git remote add origin https://github.com/063Patteera/CL64-01.git

Swift@LAPTOP-HD4PMEBH MINGW64 ~ (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 219 bytes | 43.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/063Patteera/CL64-01.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.

Swift@LAPTOP-HD4PMEBH MINGW64 ~ (main)
$ |
```

จากรูปที่ 1.14 จะได้ผลการทำงานดังรูปที่ 1.15 ซึ่งจะเห็นว่า บางคำสั่งอาจจะมี error เกิดขึ้น เนื่องจากมี repository อยู่บน server แล้ว แต่ก็ให้ทำให้ครบทุกขั้นตอนไปก่อน เพราะ ในกรณีนี้ error เหล่านั้นไม่ส่งผลกระทบร้ายแรงต่อการทำงาน

- ให้กลับไป browser และกด refresh 1 ครั้ง จะเห็นว่าหน้า repository ที่เราเพิ่งสร้าง จะเปลี่ยนไป



รูปที่ 1.15 หน้าเพจ repository ที่เปลี่ยนไปหลังจากเพิ่มไฟล์ README.md

ผลการทดลอง

ให้นักศึกษา capture หน้าจอของตนเองมาใส่ และอธิบายสั้นๆ ถึงสิ่งที่เกิดขึ้น

- เมื่อตั้งค่าต่าง ๆ เสร็จแล้ว ข้อมูลของหน้าเว็บ repository จะเปลี่ยนไปตามข้อมูลที่เราระบุค่า

1.5 การแก้ไขงานและบันทึกการเปลี่ยนแปลงบน local computer

ถึงตอนนี้ เนื้อหาในไฟล์ README.md บน server และ local computer จะเหมือนกันทุกประการ เนื่องจากการ clone มาและยังไม่ได้ทำการแก้ไขใดๆ อีกทั้งเรามั่นใจว่าไม่มีผู้อื่นใช้คนอื่นๆ กำลังแก้ไขงานของเราบน server (ซึ่งการแก้ไขงานร่วมกันบน server จะอยู่ในการทดลองถัดไป) เราสามารถแก้ไขและทำ revision ของเอกสารได้ตามต้องการ โดยการเปลี่ยนแปลงต่างๆ จะเกิดขึ้นบนเครื่อง local computer เท่านั้น

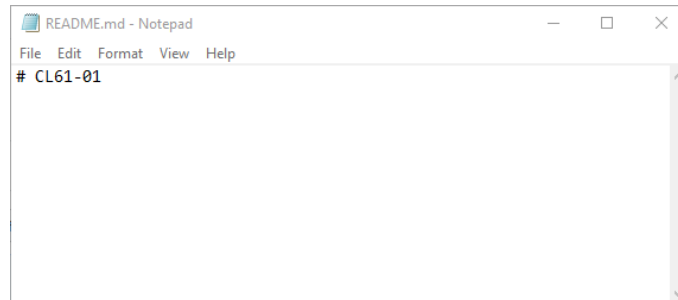
1.5.1 ทดลองแก้ไขไฟล์ README.md

โดยส่วนใหญ่ ในการเขียนโปรแกรม มักจะกระทำบนโปรแกรม Integrated development environment หรือเรียกสั้นๆ ว่า IDE⁵ แต่ในการทดลองนี้ จะใช้โปรแกรมแก้ไขเอกสารอย่างง่ายๆ นั่นคือโปรแกรม Notepad.exe

- ให้พิมพ์คำสั่งต่อไปนี้ลงใน git bash

```
$ notepad README.md
```

ระบบจะเปิด text editor ที่มากับระบบปฏิบัติการ Windows ดังรูปที่ 16

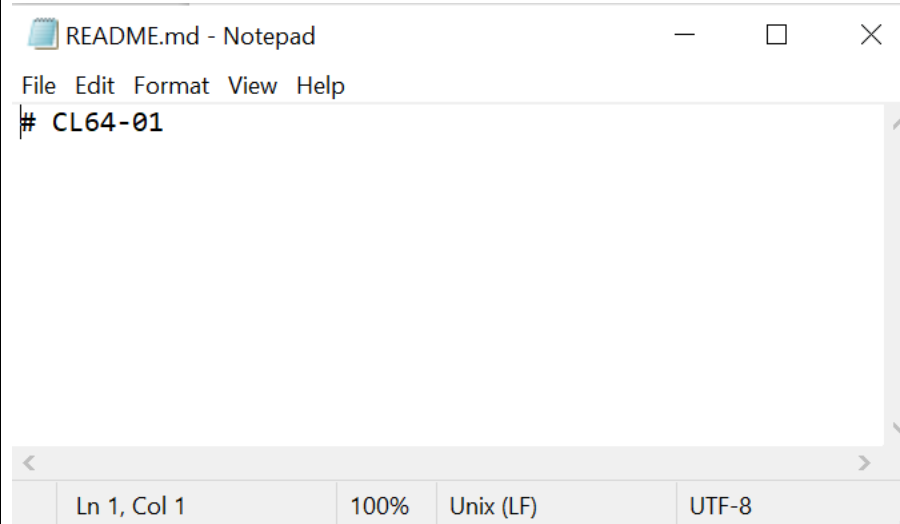


รูปที่ 1.16 การใช้โปรแกรม notepad.exe แก้ไขไฟล์ README.md

ผลการทดลอง

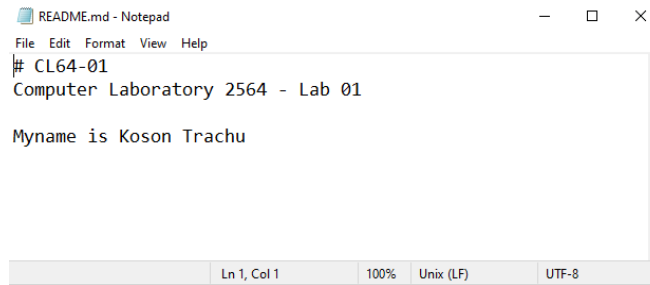
ให้นักศึกษา capture หน้าจอของตนเองมาใส่ และอธิบายสั้นๆ ถึงสิ่งที่เกิดขึ้น

- ข้อมูลใน README.md จะแสดงตามข้อมูลที่เราได้ตั้งค่าไว้



⁵ "Integrated development environment - Wikipedia." Accessed August 11, 2017.
https://en.wikipedia.org/wiki/Integrated_development_environment.

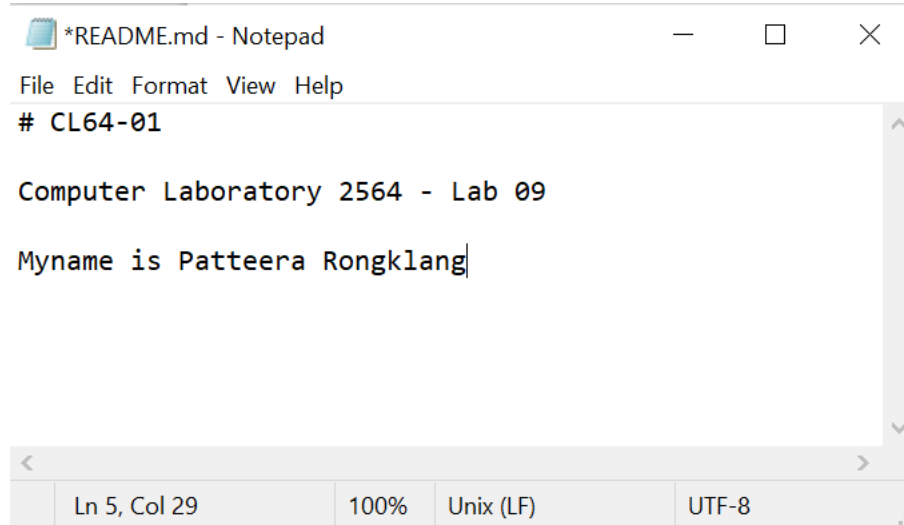
- แก้ไขไฟล์ README.md ใน notepad โดยเพิ่มข้อความลงไปดังตัวอย่าง (ให้นักศึกษาใส่ชื่อตนเอง)



รูปที่ 1.17 แก้ไขไฟล์ README.md โดยเพิ่มบรรทัดต่อท้ายเข้าไป

ผลการทดลอง

ให้นักศึกษา capture หน้าจอของตนเองมาใส่ และอธิบายสั้นๆ ถึงสิ่งที่เกิดขึ้น
- ทำการเพิ่มข้อมูลลงไป README.md



- บันทึกและปิดโปรแกรม notepad.exe
- ตรวจสอบการเปลี่ยนแปลงใน git bash โดยพิมพ์คำสั่ง git status แล้วสังเกตผลที่ได้จากการรันคำสั่ง

```
$ git status
```

```
MINGW64:~/f/GitHubRepos/Computerlab-2564/CL2564
* [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.

koston@DESKTOP-CPUNNOU MINGW64 /f/GitHubRepos/Computerlab-2564/CL2564 (main)
$ notepad readme.md

koston@DESKTOP-CPUNNOU MINGW64 /f/GitHubRepos/Computerlab-2564/CL2564 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        CL64-01/

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

koston@DESKTOP-CPUNNOU MINGW64 /f/GitHubRepos/Computerlab-2564/CL2564 (main)
$ |
```

รูปที่ 1.18 การตรวจสอบสถานะของ git

จะพบว่า git ได้ทำการติดตามการเปลี่ยนแปลง (tracking) ของไฟล์ต่างๆ ใน repository ของเราอยู่เสมอ ถึงแม้จะเป็น local computer ก็ตาม (ไม่นับไฟล์ใน .gitignore)

ผลการทดลอง

ให้นักศึกษา capture หน้าจอของตนเองมาใส่ และอธิบายสั้นๆ ถึงสิ่งที่เกิดขึ้น
- ทำการตรวจสอบสถานะปัจจุบันของ git

```
MINGW64:/c/Users/Swift/CL2564/CL64-01

Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564/CL64-01 (main)
$ |

Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564/CL64-01 (main)
$

Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564/CL64-01 (main)
$ notepad README.md

Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564/CL64-01 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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")

Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564/CL64-01 (main)
$
```

1.5.2 บันทึกการเปลี่ยนแปลงบน local computer

ถึงตรงนี้ ถ้าเราต้องการจะแก้ไขต่อ ก็ทำได้ แต่การเปลี่ยนแปลงต่างๆ จะไม่สามารถถูกติดตามโดย git ถ้าต้องการให้ git บันทึก (หรือนับ) การเปลี่ยนแปลงเป็นรุ่นต่างๆ ของ source code สามารถทำได้โดยการ commit การเปลี่ยนแปลงลงใน local repository ซึ่งการใช้งานเบื้องต้นจะมี 2 คำสั่งคือ git add และ git commit

- เพิ่มไฟล์ที่เปลี่ยนแปลง เข้าสู่รายการ commit โดยใช้คำสั่งต่อไปนี้

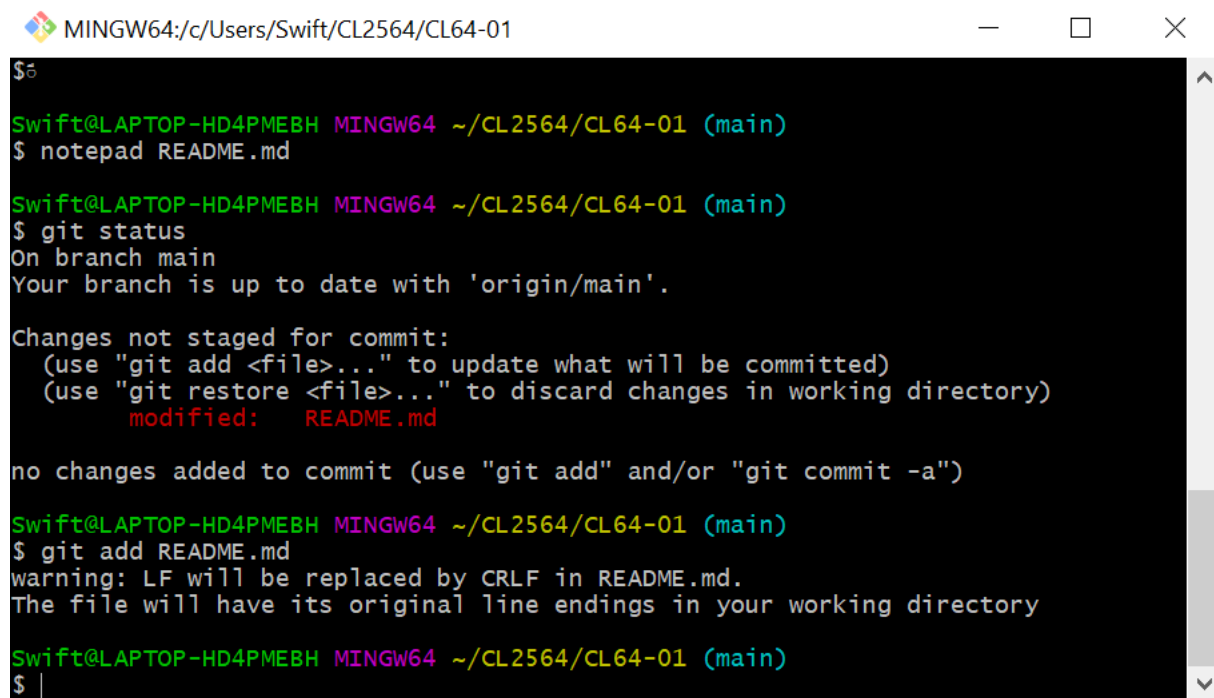
```
$ git add README.md
```

ตรวจสอบการเปลี่ยนแปลงใน git bash โดยพิมพ์คำสั่ง git status แล้วสังเกตผลที่ได้จากการรันคำสั่ง

ผลการทดลอง

ให้นักศึกษา capture หน้าจอของตนเองมาใส่ และอธิบายสั้นๆ ถึงสิ่งที่เกิดขึ้น

- ทำการเพิ่มข้อมูล README.md ลงไปใน git และทำการตรวจสอบสถานะของ git



```
MINGW64:/c/Users/Swift/CL2564/CL64-01
$
Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564/CL64-01 (main)
$ notepad README.md

Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564/CL64-01 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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")

Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564/CL64-01 (main)
$ git add README.md
warning: LF will be replaced by CRLF in README.md.
The file will have its original line endings in your working directory

Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564/CL64-01 (main)
$
```

```
$ git status
```



```
MINGW64~/f/GitHubRepos/Computerlab-2564/CL2564
no changes added to commit (use "git add" and/or "git commit -a")

koson@DESKTOP-CPUNNOU MINGW64 /f/GitHubRepos/Computerlab-2564/CL2564 (main)
$ git add README.md
warning: LF will be replaced by CRLF in README.md.
The file will have its original line endings in your working directory

koson@DESKTOP-CPUNNOU MINGW64 /f/GitHubRepos/Computerlab-2564/CL2564 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        CL64-01/

koson@DESKTOP-CPUNNOU MINGW64 /f/GitHubRepos/Computerlab-2564/CL2564 (main)
$ |
```

รูปที่ 1.19 การตรวจสอบสถานะของ git / ผลจากการทำคำสั่ง git add

หมายเหตุ หากมีการแก้ไขหลายๆ ไฟล์ เราอาจใช้คำสั่ง git add --all แทนการใช้ชื่อไฟล์ได้

- Commit ไฟล์ที่เปลี่ยนแปลง เข้าสู่ repository โดยใช้คำสั่งต่อไปนี้

```
$ git commit -m "Edited by Koson"
```

ตามด้วยการตรวจสอบสถานะของ repository

```
$ git status
```

จะได้ผลดังนี้

```
MINGW64~/f/GitHubRepos/Computerlab-2564/CL2564
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        CL64-01/

koson@DESKTOP-CPUNNOU MINGW64 /f/GitHubRepos/Computerlab-2564/CL2564 (main)
$ git commit -m "Edit by Koson"
[main 09c839b] Edit by Koson
1 file changed, 3 insertions(+)

koson@DESKTOP-CPUNNOU MINGW64 /f/GitHubRepos/Computerlab-2564/CL2564 (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        CL64-01/

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

koson@DESKTOP-CPUNNOU MINGW64 /f/GitHubRepos/Computerlab-2564/CL2564 (main)
$ |
```

รูปที่ 1.20 ผลจากการทำ git commit

ผลการทดลอง

ให้นักศึกษา capture หน้าจอของตนเองมาใส่ และอธิบายสั้นๆ ถึงสิ่งที่เกิดขึ้น
- ใช้คำสั่ง `git commit -m` เพื่อทำการเพิ่มไฟล์ที่ถูกเปลี่ยนแปลงลงไป repository



```
MINGW64:/c/Users/Swift/CL2564/CL64-01

Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564/CL64-01 (main)
$ git commit -m "Edited by Patteera"
error: pathspec 'by' did not match any file(s) known to git
error: pathspec 'Patteera"' did not match any file(s) known to git

Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564/CL64-01 (main)
$ git add README.md

Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564/CL64-01 (main)
$ git commit -m "Edited by Patteera"
[main f370a38] Edited by Patteera
1 file changed, 4 insertions(+)

Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564/CL64-01 (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean

Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564/CL64-01 (main)
$ |
```

หมายเหตุ รูปแบบของการ commit ประกอบด้วย คำสั่ง `git commit -m "THIS IS A COMMIT MESSAGE"` โดยที่ commit message ควรเป็นข้อความที่สื่อความหมาย มีความยาวไม่มากนัก แต่ไม่สั้นจนเกินไป ควรหลีกเลี่ยงคำที่ไม่สื่อความหมาย เช่น "1", "2" หรือ "a" ถึงแม้ว่า git จะอนุญาตให้ใช้ก็ตาม เนื่องจากเมื่อพัฒนาไปหลายๆ รุ่น จะไม่สามารถทำความเข้าใจเหตุผลที่แก้ไข source code นั้น ๆ ได้ และในการเปลี่ยนแปลงแต่ละครั้ง git จะนำ commit message นี้ไปใช้ร่วมกับการเปลี่ยนแปลงเสมอ

1.6 การซิงค์การเปลี่ยนแปลงระหว่าง local computer และ server

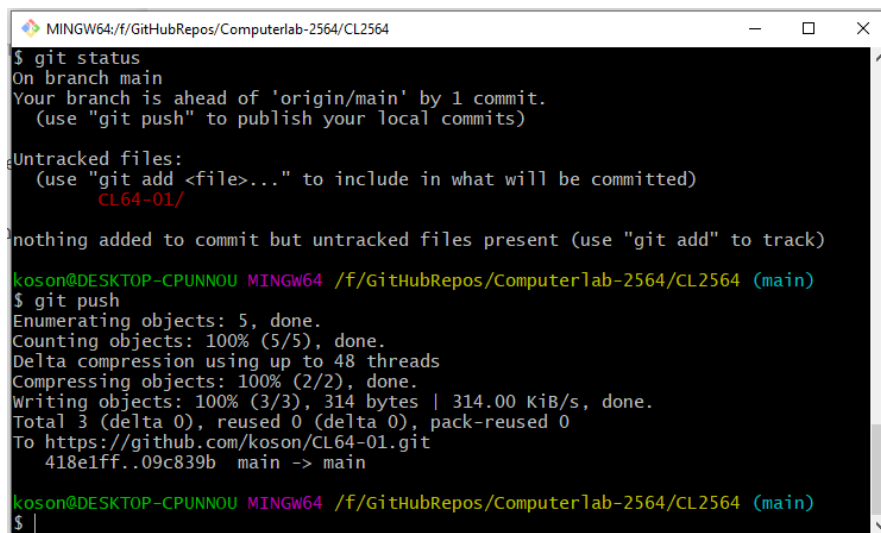
หลังจากที่เราได้ทำการ clone repository มาที่ local computer แล้ว การแก้ไขงานทั้งหมด สามารถทำได้บน local computer ได้โดยไม่ต้องเชื่อมต่อกับ server แต่ในบางครั้งที่มีการทำงานร่วมกันเป็นทีม จะต้องปรับปรุง source code ให้เป็นปัจจุบันอยู่เสมอ จะต้องมีการ sync กับ server ได้แก่การ upload การเปลี่ยนแปลงขึ้นสู่ server (เรียกว่าการ push) และการ download การเปลี่ยนแปลงมาจาก server (เรียกว่าการ pull)

1.6.1 การ push ขึ้นสู่ server

โดยทั่วไป การที่จะ push ขึ้นสู่ server เรามักจะใช้คำสั่ง 3 คำสั่งควบคู่กันคือ (1) git add --all, (2) git commit -m “Commit message” และ (3) git push แต่ในการทดลองที่ผ่านมา เราทำใน (1) และ (2) ไปแล้ว ดังนั้น ให้พิมพ์คำสั่งต่อไปนี้ เพื่อ push repository ขึ้น server

```
$ git push
```

จะได้ผลลัพธ์คล้ายตัวอย่างในรูปที่ 20



```
MINGW64:/f/GitHubRepos/Computerlab-2564/CL2564
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  CL64-01/

nothing added to commit but untracked files present (use "git add" to track)
koson@DESKTOP-CPUNNOU MINGW64 /f/GitHubRepos/Computerlab-2564/CL2564 (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 48 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 314 bytes | 314.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/koson/CL64-01.git
  418e1ff..09c839b  main -> main
koson@DESKTOP-CPUNNOU MINGW64 /f/GitHubRepos/Computerlab-2564/CL2564 (main)
$
```

รูปที่ 1.21 ผลจากการทำคำสั่ง git push

ผลการทดลอง

ให้นักศึกษา capture หน้าจอของตนเองมาใส่ และอธิบายสั้นๆ ถึงสิ่งที่เกิดขึ้น
- ใช้คำสั่ง git push เพื่อนำข้อมูลที่ถูกละเปลี่ยนแปลงขึ้น server

```
MINGW64:/c/Users/Swift/CL2564/CL64-01
[main f370a38] Edited by Patteera
1 file changed, 4 insertions(+)

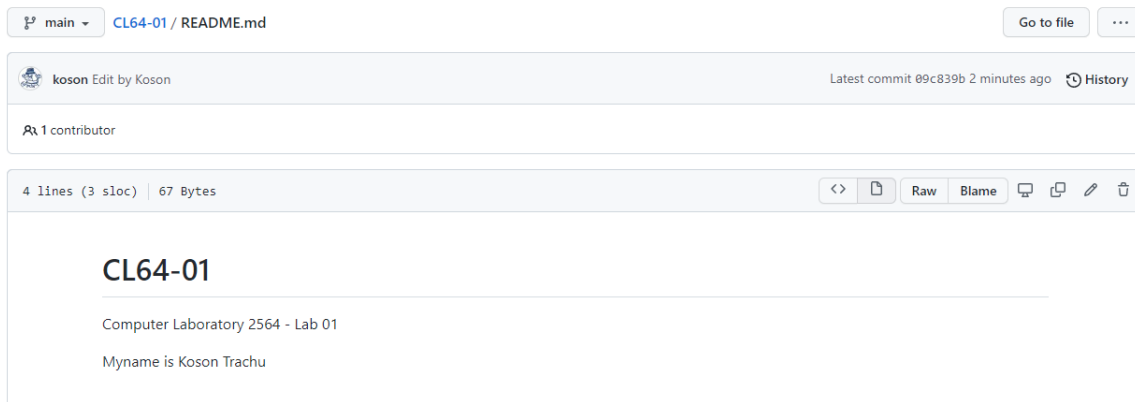
Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564/CL64-01 (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean

Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564/CL64-01 (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 312 bytes | 156.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/063Patteera/CL64-01.git
bf6e3e2..f370a38  main -> main

Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564/CL64-01 (main)
$
```

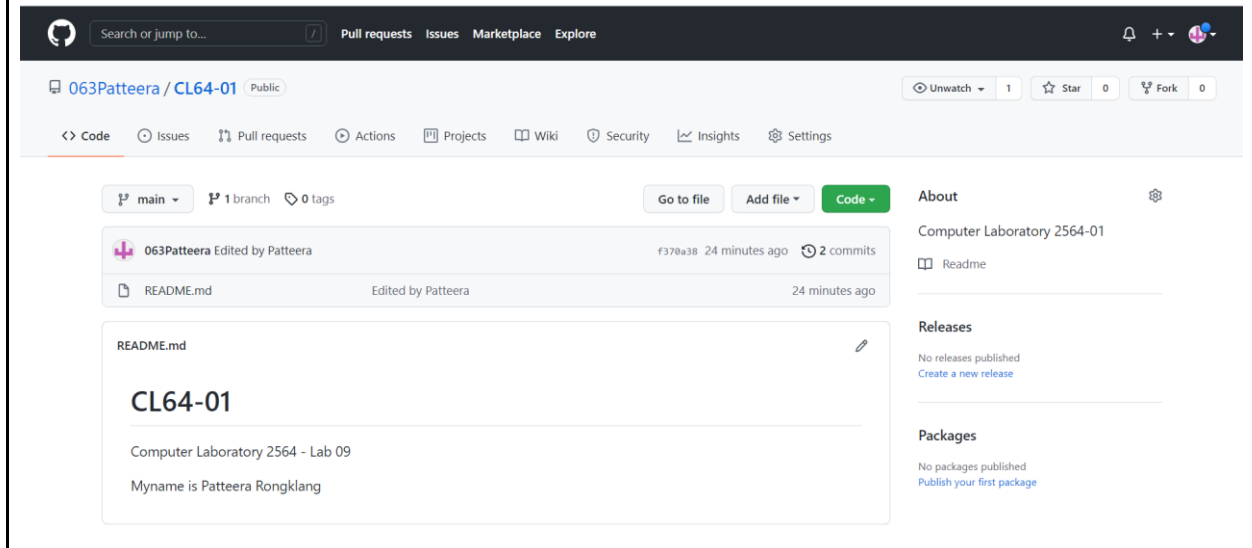
เมื่อเราทำการ push repository ขึ้นสู่ server แล้ว ก็ต้องทดสอบผลจากการ push โดยการไป refresh web browser ที่สร้าง repository ไว้ ดังรูปที่ 15



รูปที่ 1.22 การเปลี่ยนแปลงที่เกิดขึ้นบน server

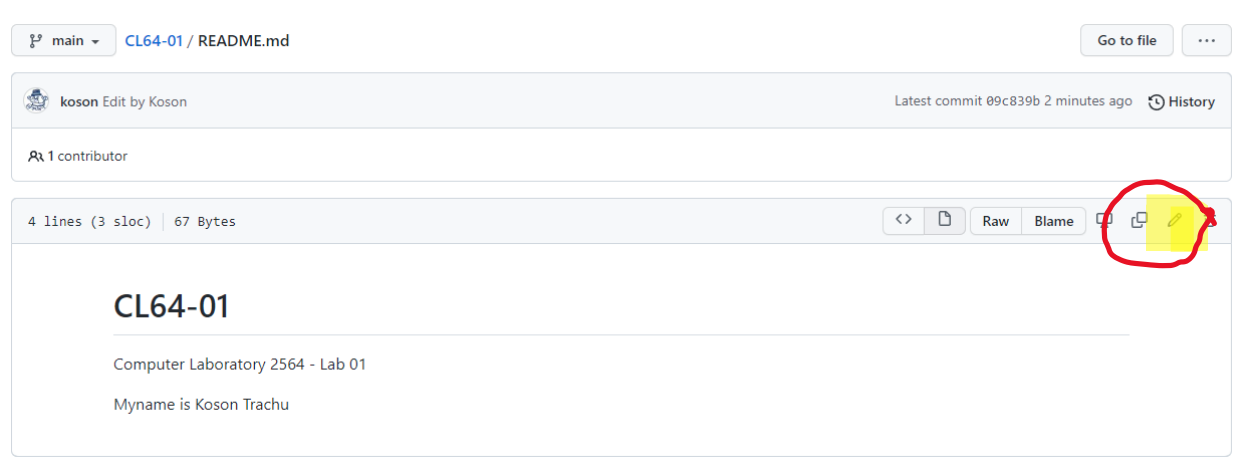
ผลการทดลอง

ให้นักศึกษา capture หน้าจอของตนเองมาใส่ และอธิบายสั้นๆ ถึงสิ่งที่เกิดขึ้น
- ผลจากการนำข้อมูลขึ้น server จะเห็นได้ว่าข้อมูลของเว็บจะมีการเปลี่ยนแปลง



1.6.2 การ pull มาจาก server

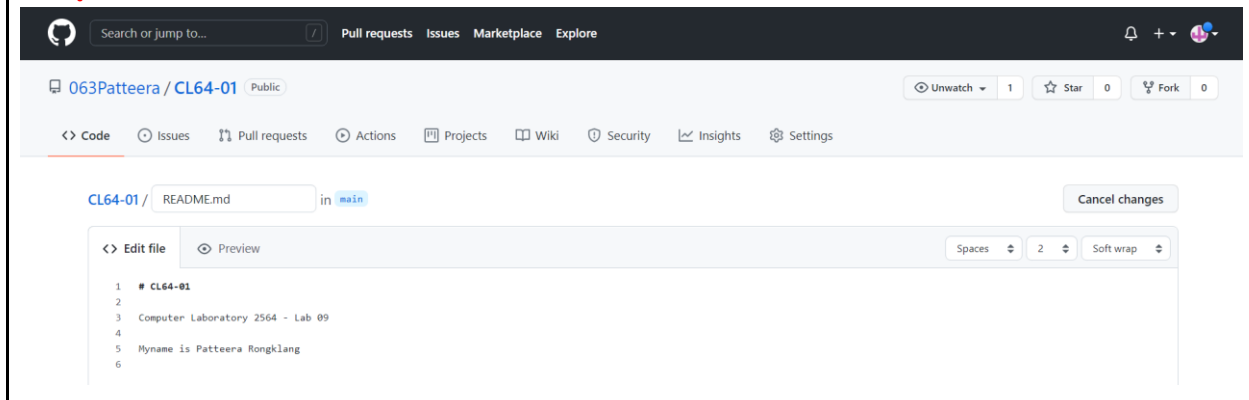
- การเปลี่ยนแปลงใดๆ ที่เกิดขึ้นบน local computer จะถูกส่งขึ้นมาเก็บด้วยคำสั่ง git push และถ้ามีการแก้ไขไฟล์ใด ๆ เกิดขึ้นบน server เราก็สามารถที่จะดึงกลับไปทำงานที่ local computer ได้เช่นกัน
- ให้แก้ไขไฟล์ README.md โดยการคลิกที่ชื่อไฟล์ และปุ่มปากกาบริเวณด้านขวามือ



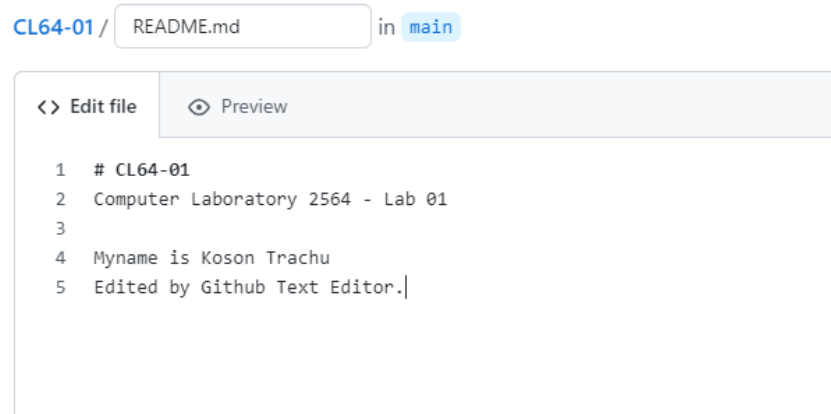
รูปที่ 1.23 เข้าสู่โหมดการแก้ไขไฟล์ด้วย Github Text Editor

ผลการทดลอง

ให้นักศึกษา capture หน้าจอของตนเองมาใส่ และอธิบายสั้นๆ ถึงสิ่งที่เกิดขึ้น
- เข้าสู่หน้าแก้ไขไฟล์ของ Github



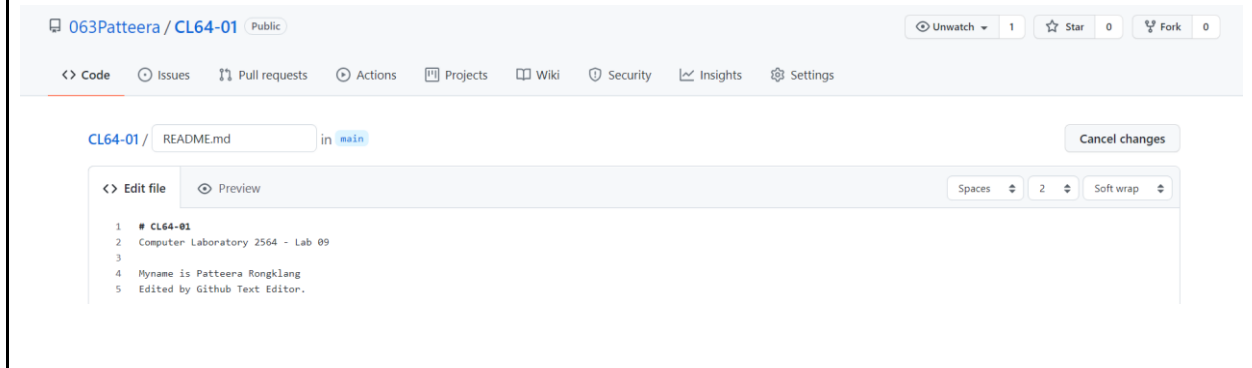
- เพิ่มข้อความที่บรรทัดล่างสุดดังตัวอย่าง



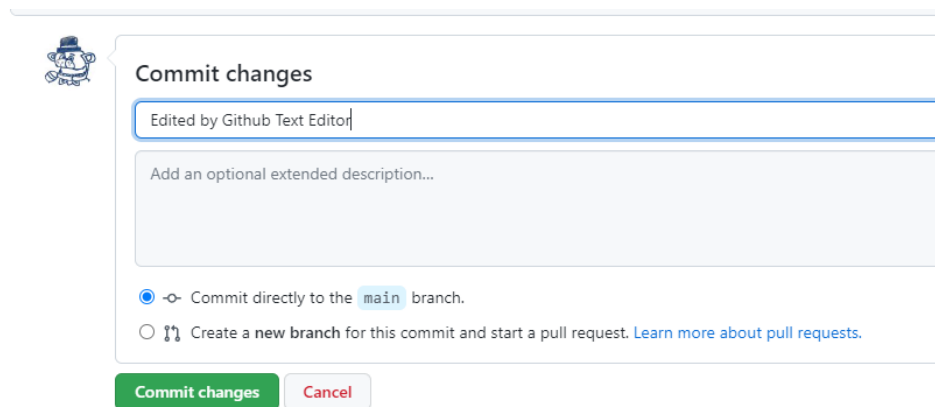
รูปที่ 1.24 เพิ่มข้อความบางอย่างใน Github Text Editor

ผลการทดลอง

ให้นักศึกษา capture หน้าจอของตนเองมาใส่ และอธิบายสั้นๆ ถึงสิ่งที่เกิดขึ้น
- ทำการเปลี่ยนข้อมูลตามภาพตัวอย่าง



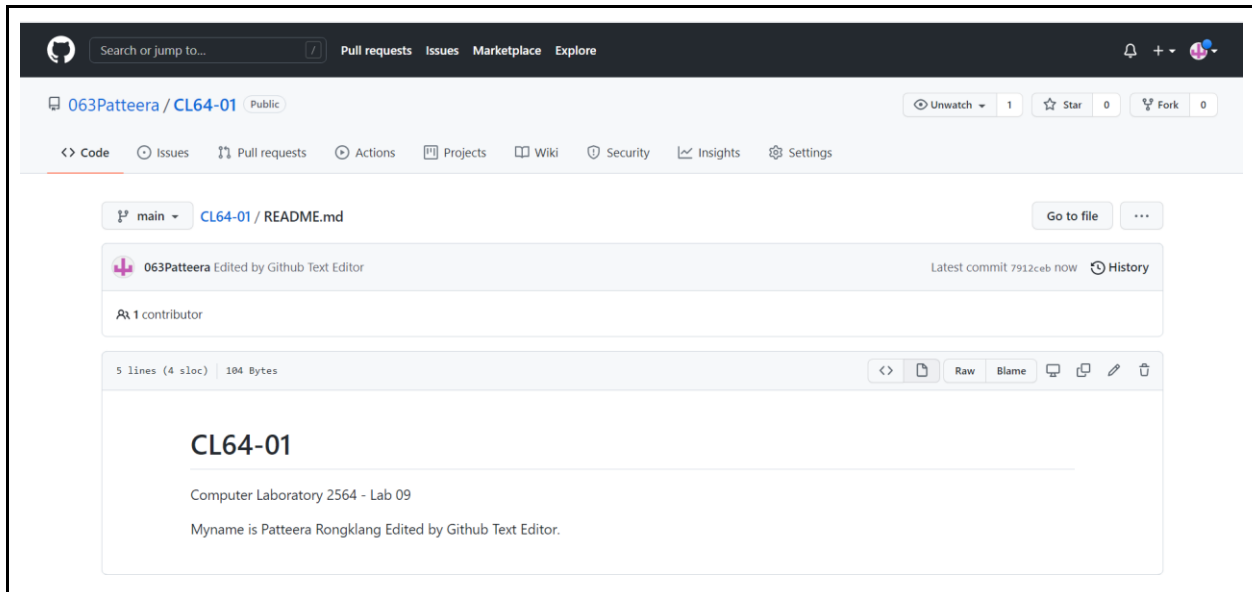
- เพิ่มข้อความในช่อง Commit changes และกดปุ่ม Commit changes สีเขียว



รูปที่ 1.25 เพิ่มข้อความ Commit changes

ผลการทดลอง

ให้นักศึกษา capture หน้าจอของตนเองมาใส่ และอธิบายสั้นๆ ถึงสิ่งที่เกิดขึ้น
- เมื่อทำการเปลี่ยนข้อมูลแล้ว จะเห็นได้ว่าข้อมูลหน้าเว็บจะมีการเปลี่ยนแปลงตามสิ่งที่เราเปลี่ยนไปก่อนหน้า



- กลับมาที่ git bash พิมพ์คำสั่ง git status สังเกตผลการทำงาน

```
$ git status
```

- ที่ git bash พิมพ์คำสั่ง git pull

```
$ git pull
```

```

MINGW64:/f/GitHubRepos/Computerlab-2564/CL2564
nothing added to commit but untracked files present (use "git add" to track)
koson@DESKTOP-CPUNNOU MINGW64 /f/GitHubRepos/Computerlab-2564/CL2564 (main)
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 713 bytes | 12.00 KiB/s, done.
From https://github.com/koson/CL64-01
   09c839b..ee99611  main       -> origin/main
Updating 09c839b..ee99611
Fast-forward
 README.md | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
koson@DESKTOP-CPUNNOU MINGW64 /f/GitHubRepos/Computerlab-2564/CL2564 (main)
$

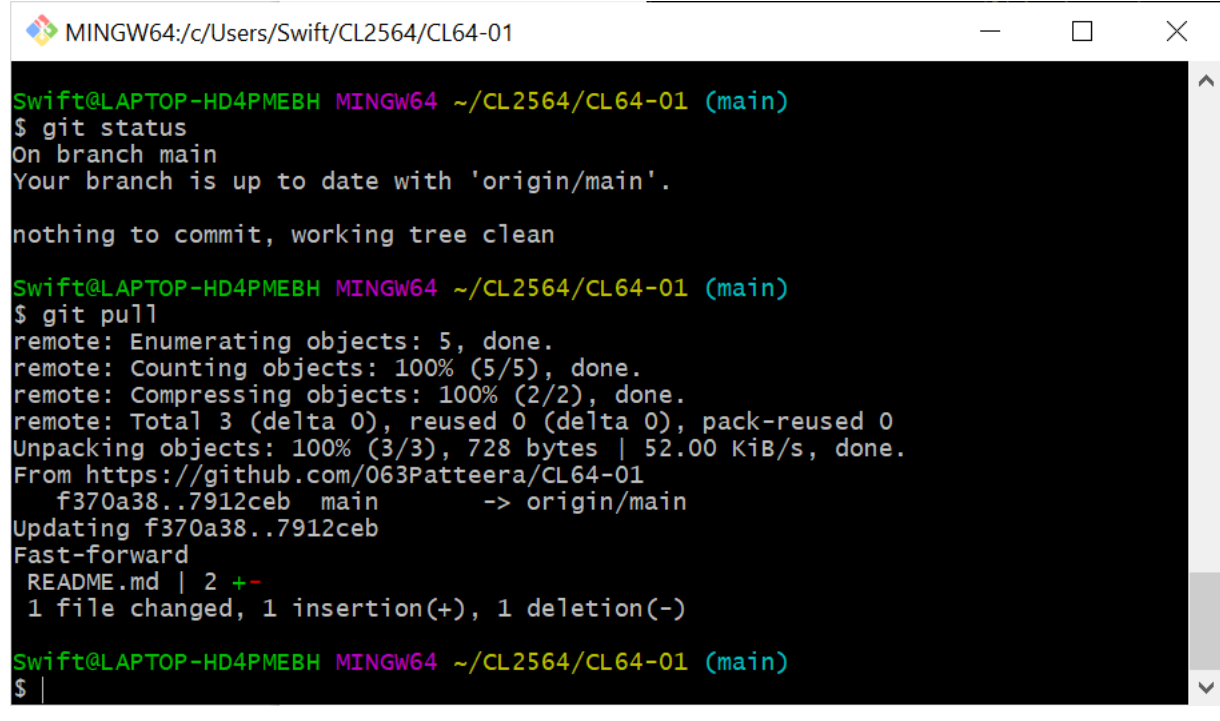
```

รูปที่ 1.26 การใช้คำสั่ง git pull

ผลการทดลอง

ให้นักศึกษา capture หน้าจอของตนเองมาใส่ และอธิบายสั้นๆ ถึงสิ่งที่เกิดขึ้น

- ทำการดึงข้อมูลจาก server ของเว็บ repository ที่เราเปลี่ยนแปลงไปมาอยู่ในเครื่อง



```
MINGW64:/c/Users/Swift/CL2564/CL64-01

Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564/CL64-01 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

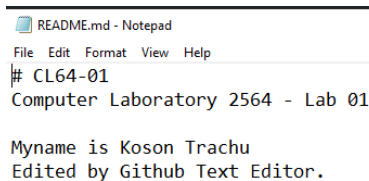
nothing to commit, working tree clean

Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564/CL64-01 (main)
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 728 bytes | 52.00 KiB/s, done.
From https://github.com/063Patteera/CL64-01
   f370a38..7912ceb  main       -> origin/main
Updating f370a38..7912ceb
Fast-forward
 README.md | 2 + -
 1 file changed, 1 insertion(+), 1 deletion(-)

Swift@LAPTOP-HD4PMEBH MINGW64 ~/CL2564/CL64-01 (main)
$
```

- ดูการเปลี่ยนแปลงในไฟล์ README.md

```
$ notepad.exe README.md
```



```
README.md - Notepad
File Edit Format View Help
# CL64-01
Computer Laboratory 2564 - Lab 01

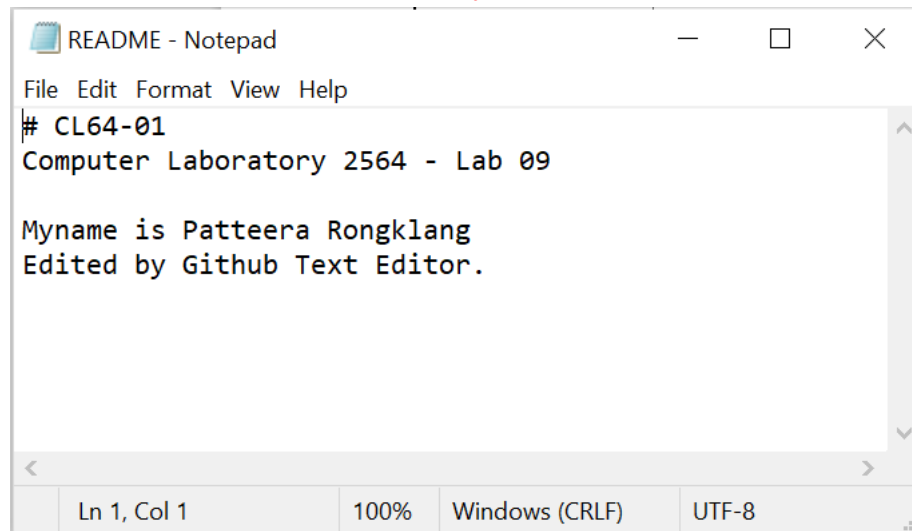
Myname is Koson Trachu
Edited by Github Text Editor.
```

รูปที่ 1.27 การเปลี่ยนแปลงในไฟล์เอกสาร README.md

ผลการทดลอง

ให้นักศึกษา capture หน้าจอของตนเองมาใส่ และอธิบายสั้นๆ ถึงสิ่งที่เกิดขึ้น

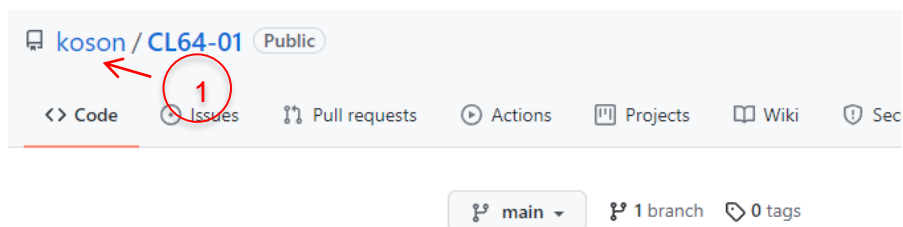
- ทำการเปิดไฟล์ README.md จะเห็นได้ว่าข้อมูลมีการเปลี่ยนแปลงตามที่เราได้เปลี่ยนใน Github



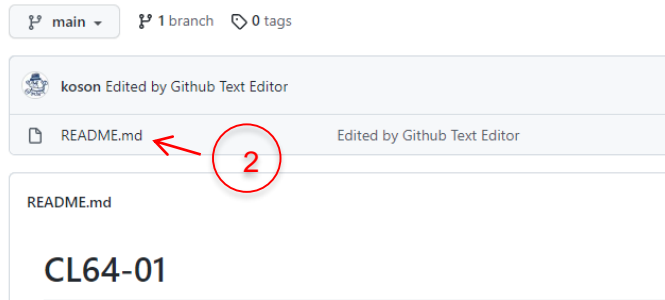
1.7 การตรวจสอบประวัติการเปลี่ยนแปลงของไฟล์

- กลับไปที่ web browser (1) คลิกที่ชื่อ repository, (2) คลิกที่ชื่อไฟล์ README.md (3) คลิกปุ่ม History

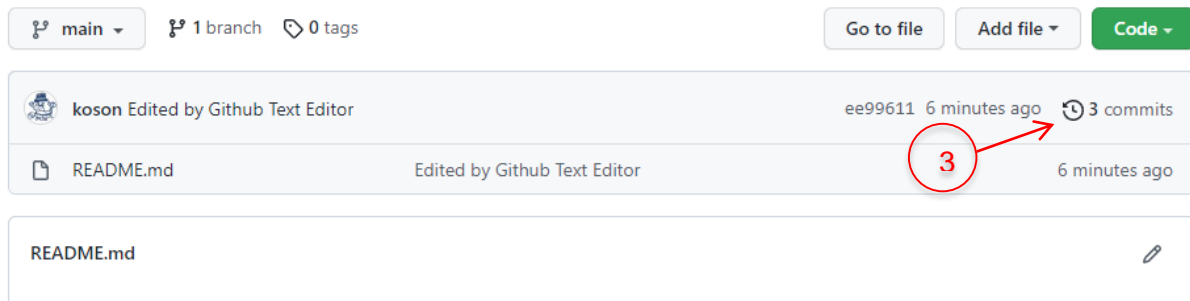
1



2.



3.



รูปที่ 1.28 การเข้าถึงประวัติของไฟล์

เมื่อคลิกดูประวัติไฟล์ จะพบว่า ไม่ว่าเราจะแก้ไขไฟล์ที่ไหน แต่ Git จะติดตามและบันทึกการเปลี่ยนแปลงทุกครั้งที่เราทำการ commit

ผลการทดลอง

ให้นักศึกษา capture หน้าจอของตนเองมาใส่ และอธิบายสั้นๆ ถึงสิ่งที่เกิดขึ้น

- เมื่อกดมาที่ CL64-01 จะขึ้นข้อมูลเกี่ยวกับประวัติต่าง ๆ ที่เราได้ทำการเปลี่ยนแปลงไป

main

1 branch

0 tags

Go to file

Add file

Code

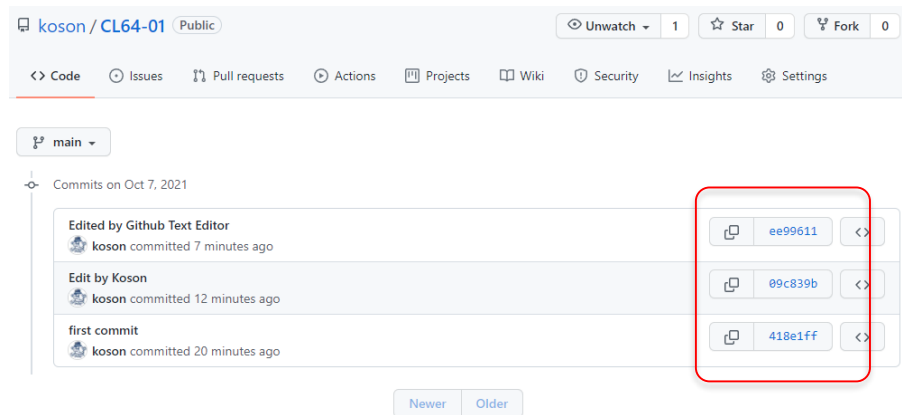
063Patteera Edited by Github Text Editor

7912ceb 18 minutes ago 3 commits

README.md

Edited by Github Text Editor

18 minutes ago



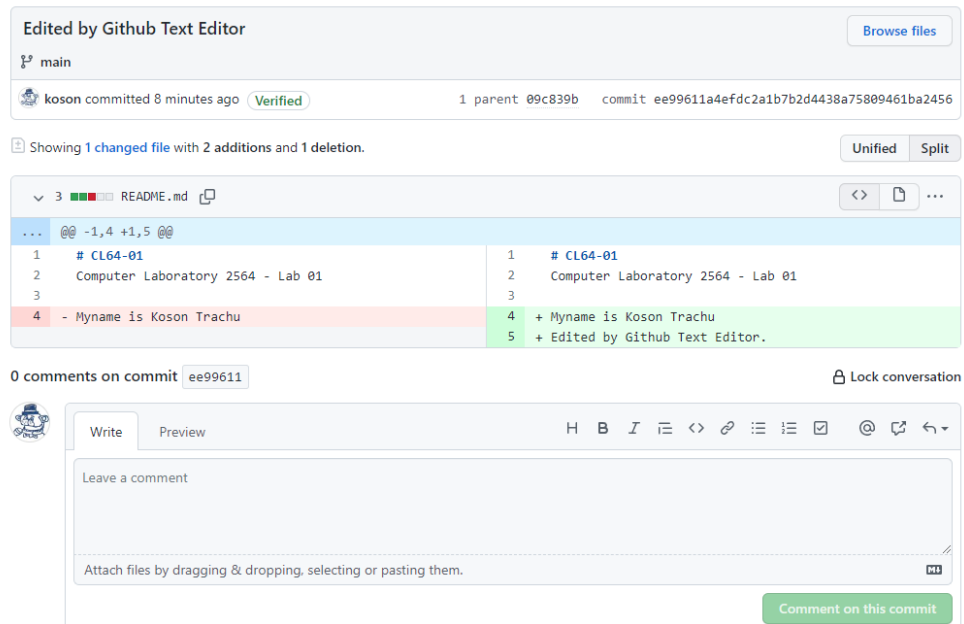
รูปที่ 1.29 รายการประวัติการแก้ไขไฟล์

ผลการทดลอง

ให้นักศึกษา capture หน้าจอของตนเองมาใส่ และอธิบายสั้นๆ ถึงสิ่งที่เกิดขึ้น

- เมื่อกดดูประวัติ จะขึ้นประวัติข้อมูลทั้งหมด โดยเราสามารถดูได้ว่าเราได้ทำการเปลี่ยนแปลงส่วนไหนไปบ้าง

ให้คลิกปุ่มที่มีเลขฐาน 16 กำกับ (เป็นชื่อรหัสกำกับกับการแก้ไข ที่ทีมพัฒนาจะใช้อ้างอิงถึง) ตามลูกศรสีแดงในรูปที่ 29 เรา
จะเห็นประวัติการแก้ไขไฟล์ ดังรูปที่ 30



รูปที่ 1.30 ประวัติการแก้ไขไฟล์

แบบฝึกหัด

- ให้นักศึกษาทดลองเพิ่มไฟล์ชื่อ student.txt ลงใน repository แล้วเพิ่มรายชื่อเพื่อนในห้อง โดยเพิ่มบน notepad จำนวนครึ่งหนึ่ง และทำบน github text editor จนครบ โดยให้เขียน commit message ด้วยว่าเพิ่มจากที่ได้
- ให้นักศึกษาทดลองแก้ไขไฟล์ README.md ตามตารางต่อไปนี้ แล้วทำรายงานประวัติไฟล์มาส่ง

ลำดับที่	สถานที่แก้ไข	สิ่งที่กระทำ
1	Local (Notepad)	ลบเนื้อหาเดิมออกทั้งหมด
2	Server (Github Text Editor)	<pre>#include <stdio.h> main() { printf("hello, world\n"); }</pre>
3	Local (Notepad)	เปลี่ยน printf("hello, world\n"); เป็น printf("hello, [ชื่อนักศึกษา]\n");
4	Server (Github Text Editor)	<pre>#include <stdio.h> int main () {</pre>

		<pre>char yourname[100]; printf("What is your name?\t"); scanf("%s",yourname); printf("hello, %s\n", yourname); }</pre>
5	Local (Notepad)	<pre>เพิ่ม printf("Goodbye\n"); ได้ printf("hello, %s\n", yourname);</pre>

หมายเหตุ การทำแต่ละขั้น ให้ local และ server ซิงค์กันเสมอ (ต้อง push, pull, commit, add)

คำถาม

- จากภาพที่ 29 ถ้าหากนักศึกษาคลิกตามปุ่ม ที่มีเลขฐานสิบหกกำกับอยู่ ทุกปุ่ม จะได้ผลอย่างไรบ้าง ให้อธิบายสิ่งที่พบเห็น
= จะเป็นการดูประวัติการแก้ไขของแต่ละ commit ว่าเราได้แก้ไข เพิ่ม หรือลบส่วนไหนไปแล้วบ้าง
- ให้บอกประโยชน์ของ repository ตามที่นักศึกษาเข้าใจ
= ช่วยให้โปรแกรมการเขียนโปรแกรมทำได้ง่ายและรวดเร็วมากขึ้น เพราะเราสามารถกลับมาดูประวัติที่เราทำการเขียนก่อนหน้าได้ตลอดเวลา ดูสิ่งที่เราเพิ่ม หรือลบก่อนหน้าได้ และในกรณีที่เลวร้ายที่สุดหากเราแก้ไขโค้ดในเครื่องแล้วเกิด error เราก็สามารถไปดูโค้ด หรือไปดึงโค้ดก่อนหน้าที่เราเขียนก่อนที่จะเกิด error มาใช้ได้
- ให้บอกแนวทางการนำ repository ไปใช้ในการเรียนหรือชีวิตประจำวันของนักศึกษา
= นำ repository มาใช้ในโปรแกรมการเขียนโปรแกรมที่กำลังทำอยู่ จากประสบการณ์ส่วนตัว เวลาที่แก้ไขโค้ดมักจะไม่มีการสำรองข้อมูลของโค้ดไว้ ทำให้บางครั้งเพิ่มโค้ดแล้วทำให้เกิด error ต้องมานั่งแก้โค้ดส่วนที่ error ให้กลับไปเป็นเหมือนเดิม แต่ถ้าเกิดนำ repository ของ github มาใช้ สามารถช่วยแก้ปัญหาส่วนนี้ได้ เพราะจะช่วยให้สามารถดูประวัติการเขียนโค้ดที่เขียนไว้ได้