

การทดลองที่ 1 การใช้งาน Repository เปื้องตัว

วัตถุประสงค์

- เพื่อให้ผู้เรียนรู้และเข้าใจแนวคิดในการใช้ Repository
- เพื่อให้ผู้เรียนสามารถใช้ Repository (Github) เปื้องตัวได้

ทฤษฎีก่อนการทดลอง

Git

Git¹ เป็นระบบควบคุมเวอร์ชัน (Version control systems) แบบ open source เป็นเครื่องมือที่ใช้บริหารจัดการการเปลี่ยนแปลงของไฟล์ต่าง ๆ ใน project การบันทึกการแก้ไขไฟล์แต่ละครั้งจะเรียกว่ารุ่น (revision) ซึ่งแต่ละรุ่นของการเปลี่ยนแปลงจะถูกกำหนดด้วยการประทับเวลา (timestamp) และบุคคลที่ทำการเปลี่ยนแปลง ดังนั้น หากเกิดความผิดพลาดหรือเสียหายจากการแก้ไข เรา ก็จะสามารถย้อนเวลากลับไปยังการแก้ไขครั้งก่อนๆ ที่สมบูรณ์ได้ตามต้องการ ถ้าได้ว่าระบบควบคุมเวอร์ชันเป็นระบบพื้นฐานที่นิยมใช้ในการบริการจัดการ source code ของโปรแกรม ซึ่งจริง ๆ แล้ว เราสามารถใช้ระบบควบคุมเวอร์ชันกับไฟล์ชนิดใดๆ หรืองานชนิดใดๆ ก็ได้ ไม่เฉพาะ source code ของโปรแกรมเท่านั้น ในปัจจุบัน มีระบบควบคุมเวอร์ชันให้เลือกใช้หลากหลาย ทั้งเป็นแบบฟรีและมีค่าใช้จ่าย (เช่น Git, Mercurial, Subversion) โดย Git จะได้รับความนิยมมากกว่าชนิดอื่นๆ การทำงานของ Git นั้นจะมีพื้นที่เก็บไฟล์ ซึ่งเรียกว่า ‘repositories’ ซึ่งเราสามารถติดตั้งบริการ git บน server ได้ ก็ได้แต่ server บริการ git ที่ได้รับความนิยมในปัจจุบันได้แก่ Github, Gitlab, Bitbucket เป็นต้น ข้อดีของการใช้ server รวมก็คือสามารถแบ่งปันและร่วมมือ ช่วยเหลือกันในแก้ไขโปรแกรมได้จากทุกคนทั่วโลก ลักษณะเฉพาะอย่างหนึ่งของ Git ก็คือ ใน folder ที่ชื่อ .git บนคอมพิวเตอร์ของเราจะเก็บทุกสิ่งที่เก็บบน server จึงมั่นใจได้ว่า เราสามารถทำงานกับระบบควบคุมเวอร์ชันได้ทั้งแบบออนไลน์และออฟไลน์ และหากเกิดกรณีที่ repository บน server เสียหาย เรา ก็สามารถนำทุกอย่างที่เก็บบนเครื่องกลับขึ้นไปเก็บบน server ได้

Github

Github เป็นบริษัทหนึ่ง ที่ให้บริการ Git repository บนพื้นฐานของเว็บ (web-based Git repository hosting) โดย Github จะให้พื้นที่เราสร้าง repository สำหรับโปรเจค ให้บริการฟังก์ชันการทำงานพื้นฐานของระบบ git เช่น การ branches, merges,

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

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 มีโปรเจคที่สำคัญมากว่ายุคด้วย เช่น kernel ของ Linux , source code ของ dotnet framework จากไมโครซอฟท์ และอื่นๆ ทำให้มีความมั่นใจในระดับหนึ่งว่าถ้า Github เกิดล่มขึ้นมา ก็จะมีเพื่อนร่วมชาตารอมอีกไม่น้อย

ขั้นตอนการทดลอง

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 ของเรารได้ ซึ่งจะสะดวกในการทำ project ที่เป็นความลับ

1.1 สร้างบัญชีผู้ใช้งานบน Github

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

The screenshot shows the 'Join GitHub' page with the heading 'The best way to design, build, and ship software.' Below it, there are three steps: 'Step 1: Create personal account', 'Step 2: Choose your plan', and 'Step 3: Tailor your experience'. The first step is active, showing fields for 'Username' (with placeholder 'This will be your username — you can enter your organization's username next.'), 'Email Address' (with placeholder 'You will occasionally receive account related emails. We promise not to share your email with anyone.'), and 'Password' (with placeholder 'Use at least one lowercase letter, one numeral, and seven characters.'). To the right, a sidebar lists 'You'll love GitHub' features: 'Unlimited collaborators', 'Unlimited public repositories', 'Great communication', 'Frictionless development', and 'Open source community'. At the bottom, a note states: 'By clicking on "Create an account" below, you are agreeing to the [Terms of Service](#) and the [Privacy Policy](#)'. A green 'Create an account' button is at the very bottom.

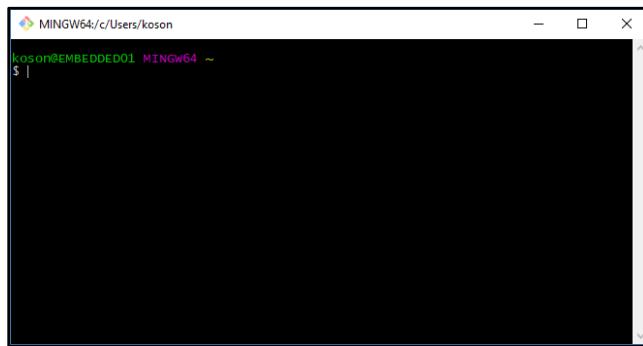
รูปที่ 1.1 การสร้างบัญชี Github

1.2 ติดตั้งโปรแกรม Git

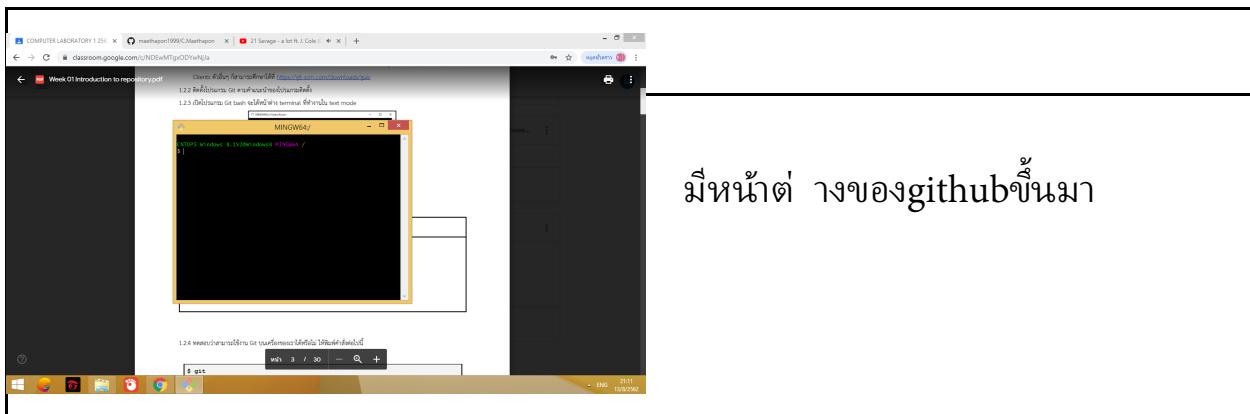
1.2.1 ดาวน์โหลดโปรแกรม Git จาก <https://git-scm.com/downloads> โดยเลือกโปรแกรมติดตั้งให้ตรงกับระบบปฏิบัติการที่ใช้ โปรแกรมที่ดาวน์โหลดมา จะมี GUI ให้เราใช้งานด้วยซึ่งมีชื่อเรียกว่า Github desktop แต่ถ้าหากสนใจที่จะใช้ Git GUI Clients ตัวอื่นๆ ก็สามารถศึกษาได้ที่ <https://git-scm.com/downloads/guis>

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



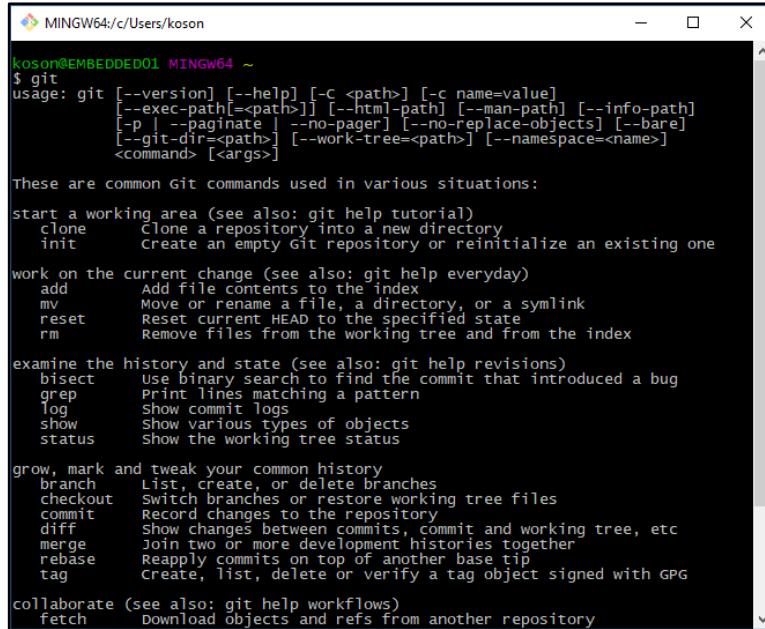
มีหน้าต่าง ทางของgithubขึ้นมา

1.2.4 ทดสอบว่าสามารถใช้งาน Git บนเครื่องของเราได้หรือไม่ ให้พิมพ์คำสั่งต่อไปนี้

```
$ git
```

ถ้า terminal ตอบกลับมาว่าไม่รู้จักคำสั่ง git แสดงว่าการติดตั้งยังไม่สมบูรณ์ ให้กลับไปตรวจสอบขั้นตอน 1.2.2 ให้ติดตั้ง

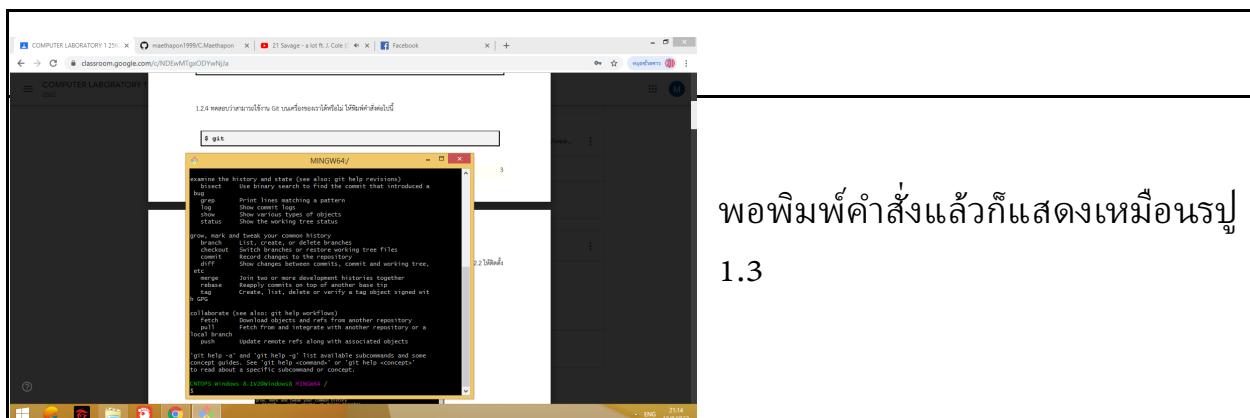
เรียบร้อย



```
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>]
           <commands> [<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



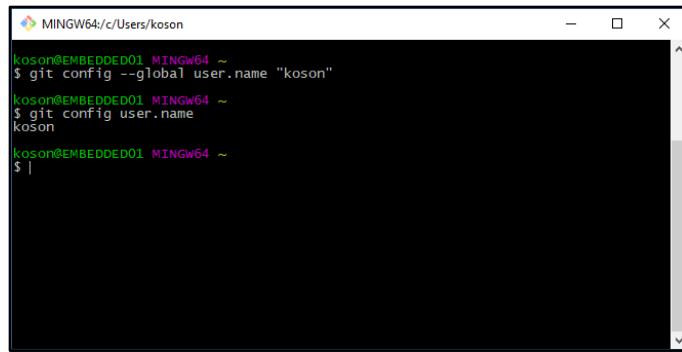
1.2.5 บอกให้ Git รู้จักชื่อของเรา โดยพิมพ์คำสั่งต่อไปนี้²

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

ในกรณีที่เราต้องการทราบชื่อผู้ใช้ปัจจุบัน สามารถสั่งให้ Git รายงานออกมาด้วยการพิมพ์คำสั่งต่อไปนี้

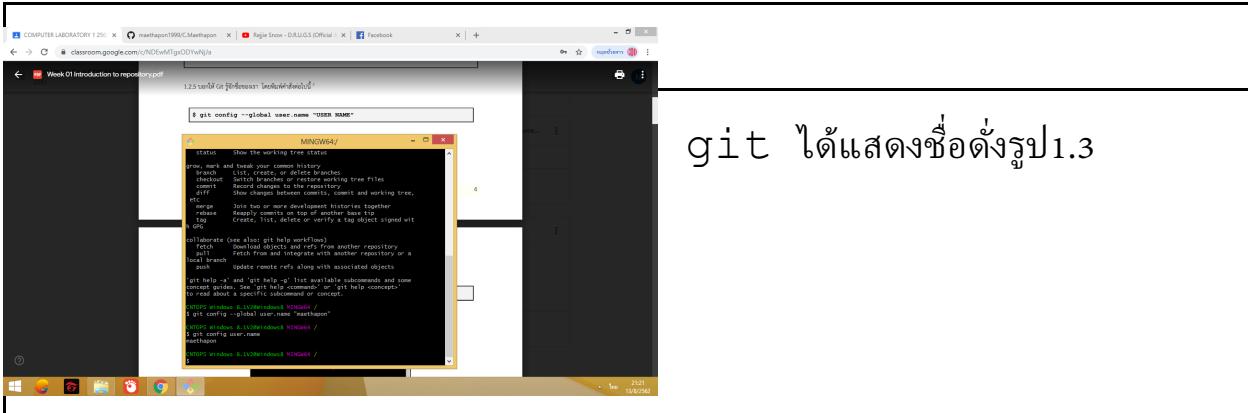
² "Setting your username in Git - User Documentation - GitHub Help." Accessed August 10, 2017. <https://help.github.com/articles/setting-your-username-in-git/>.

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



```
MINGW64:/c/Users/koson
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



1.2.6 บอกให้ Git รู้จัก email ของเรา โดยพิมพ์คำสั่งต่อไปนี้

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

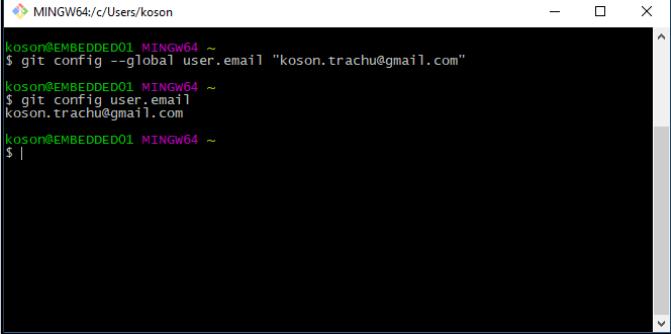
ในกรณีที่เราต้องการทราบข้อมูลผู้ใช้งานบน สามารถสั่งให้ Git รายงานอีเมลได้โดยการพิมพ์คำสั่งต่อไปนี้

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

หมายเหตุ email ที่ใช้จะต้องตรงกับ email ที่ลงทะเบียนไว้กับ Github มิฉะนั้นจะไม่สามารถเขียนข้อมูลขึ้นไปบน server ได้

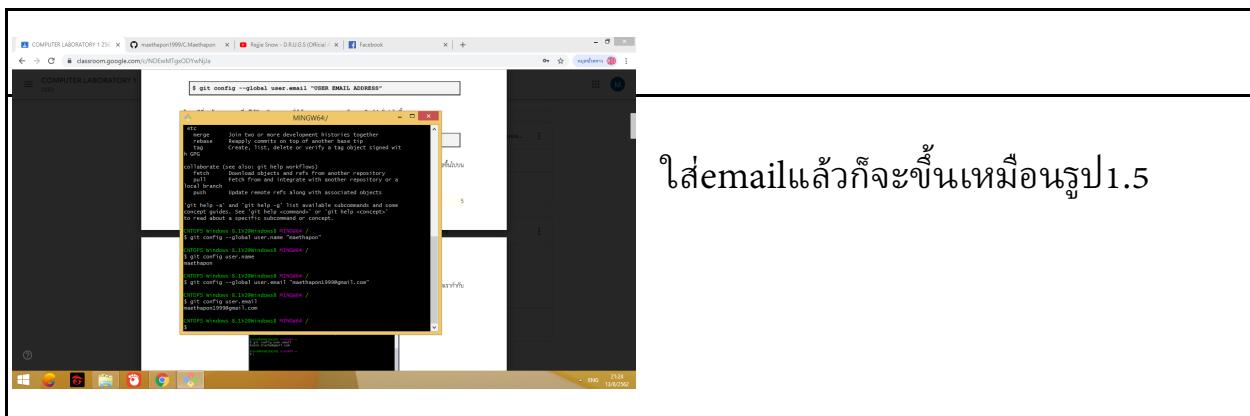
เมื่อทำในขั้นตอน 1.2.5 และ 1.2.6 เรียบร้อยแล้ว การทำงานใดๆ บน Github ก็จะประภูมิชื่อและ Email ของเรา kab กับ

ไฟล์ config



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

รูปที่ 1.5 git config --global user.email

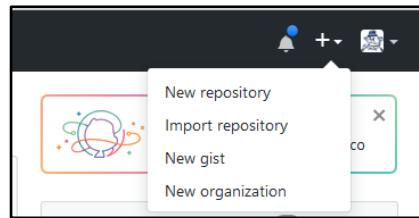


1.3 สร้าง repository (บน server)

Repository เป็นพื้นที่สำหรับเก็บ project ของเรา ซึ่งไม่ได้มายความถึงเฉพาะ source code เท่านั้น repository ยังสามารถประกอบด้วยไฟล์ทุกชนิด ไม่ว่าจะเป็น Word Document, spread sheet, presentation, เอกสารการอภิเคราะห์และออกแบบซอฟต์แวร์ ไฟล์มีเดียวภาพและเสียง รวมไปถึงเอกสาร Wiki ในลักษณะ html ด้วย ดังนั้น ในการทำโครงการพัฒนาซอฟต์แวร์ เราสามารถนำทุกสิ่งที่จำเป็นสำหรับการทำงาน มาใส่ไว้ใน repository และเมื่อเพื่อนร่วมทีมหรือ user ได้ฯ ทำสำเนา repository ของเราไป เขายังจะได้ทุกสิ่งทุกอย่างไปอย่างครบถ้วน ดังนั้นจึงอาจพูดได้ว่าเราสามารถใช้ repository เป็นเครื่องมือบริหารโครงการที่มีประสิทธิภาพได้เช่นกัน

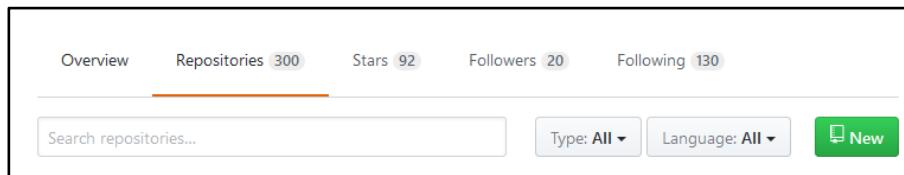
1.3.1 การสร้าง repository บน Github สามารถสร้างได้หลายวิธีด้วยกัน เช่น

(1) การสร้าง repository โดยการคลิกที่ปุ่มเครื่องหมาย “+” ที่ด้านบนขวาของหน้าจอ Github และเลือก 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 ซึ่งจะพาเราไปสร้าง repository บนเว็บแล้ว เรายังสามารถสร้าง repository โดยใช้ command line บน terminal (ศึกษาได้จาก [adding-an-existing-project-to-github-using-the-command-line³](#))

1.3.2 กำหนดชื่อและชนิดของ repository

การใช้วิธีการ 3 วิธีแรก ในข้อ 1.3.1 จะได้ผลอย่างเดียวกัน คือ Github จะพามาหน้าสำหรับสร้าง repository

- ในช่อง **Repository name** ให้ใส่ชื่อของ repository เนื่องจากบอยครั้งที่เราต้องใช้งานคำสั่งต่าง ๆ บน terminal ซึ่งต้องพิมพ์ชื่อ repository เอง ดังนั้นชื่อของ repository จะต้องมีความหมายในตัว เช่นใจจ่าย 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/>

- ขั้นตอนของ **repository** นั้น ถ้าหากเป็นโปรเจคที่เป็นความลับ ไม่อาจเปิดเผยต่อชาวโลกได้ เช่นประกอบด้วยฐานข้อมูลในงานวิจัย คะแนนแล็บของนักศึกษา ชื่อ URL, user name, password ที่เขียนลงไปใน source code เราอาจจะเลือกเป็น private ซึ่งอาจจะต้องมีค่าใช้จ่ายในการสมัครสมาชิกพิเศษ หรือไม่ก็ต้องเป็น academic account ในที่นี้ให้เลือกเป็น public
- ถ้าเราทำเครื่องหมาย หน้าข้อความ Initialize this repository with a README เพื่อให้เราสามารถเขียนบรรยายคร่าวๆ เกี่ยวกับ repository ได;
 - **เดียวก่อน.... 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++ และ Git จะเพิ่มชนิดของไฟล์ต่างๆ ที่เป็นผลจากการคอมไพล์ไว้ในรายการที่เก็บราย (เช่น ไฟล์ที่มีนามสกุล .exe) ซึ่งไฟล์เหล่านั้น มักจะเกิดจากการคอมไпал์โปรแกรม ไม่ใช่ไฟล์ที่เราเป็นคนแก้ไข source code จึงไม่จำเป็นที่จะต้องนำไปเก็บบน repository ให้เสียเปลืองพื้นที่ สามารถดูที่เมนูเพลตของ .gitignore ได้จาก A collection of useful .gitignore templates⁴
 - **ยังไม่ต้องเลือก .gitignore เช่นเดียวกัน**

Create a new repository

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

Owner: koson / Repository name: CL61-01

Great repository names are short and memorable. Need inspiration? How about *congenial-octo-happiness*.

Description (optional): Computer Laboratory 2561-01

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

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

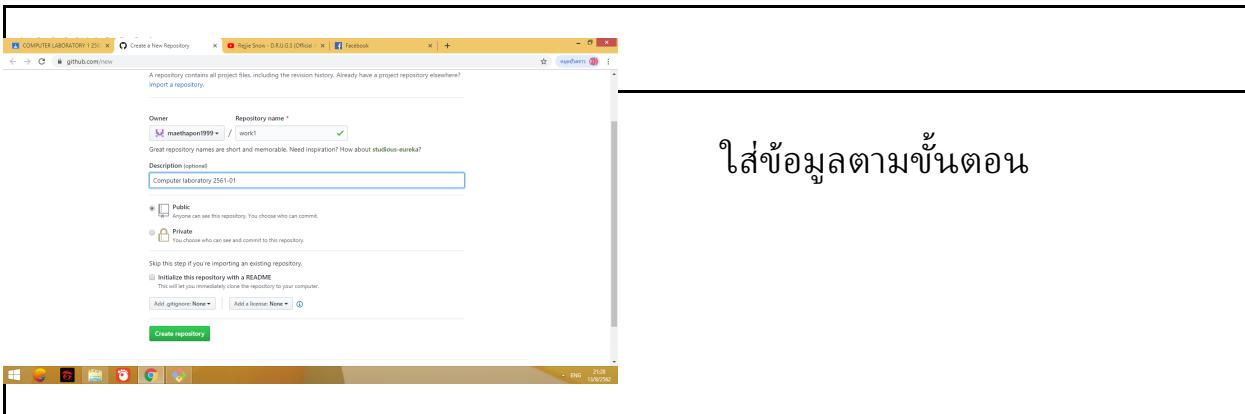
Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None

Create repository

รูปที่ 1.8 การสร้าง repository

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



- คลิกปุ่ม Create repository สีเขียว
Github จะสร้าง repository ให้ตามต้องการ

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

Set up in Desktop or HTTPS SSH <https://github.com/koson/CL61-01.git>

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

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

```
echo "# CL61-01" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/koson/CL61-01.git
git push -u origin master
```

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

```
git remote add origin https://github.com/koson/CL61-01.git
git push -u origin master
```

...or import code from another repository

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

[Import code](#)

รูปที่ 1.9 repository ที่ได้จากการสร้างในข้อ 1.3

ผลการทดลอง

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 ซึ่งเป็นต้นนี้ เราจะศึกษาโดยการใช้งาน command line ซึ่งอาจจะพบกับความยุ่งยากบ้างในตอนแรกๆ แต่เมื่อใช่ป้อย ๆ จนชำนาญจะพบว่ามีความยืดหยุ่นสูงกว่าการใช้ GUI Clients หรือเมื่อศึกษาจนเข้าใจแล้วหันไปใช้ GUI Clients ก็จะสามารถเข้าใจถึงการทำงานของระบบ Git อย่างแท้จริง

1.4.1 การ clone repository ด้วย command line (git bash)

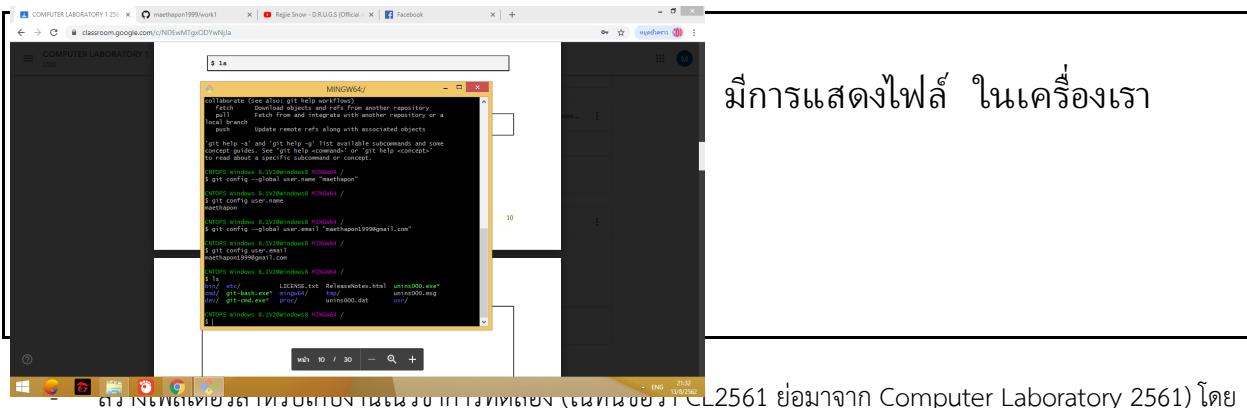
1) การเตรียมการเบื้องต้น

- ในหน้าต่าง git bash ให้พิมพ์คำสั่ง list ดูรายการของไฟล์และโฟลเดอร์

```
$ ls
```

เราจะเห็นรายการไฟล์ๆ กุ้กแสดงขึ้นมา

ผลการทดลอง



มีการแสดงไฟล์ ในเครื่องเรา

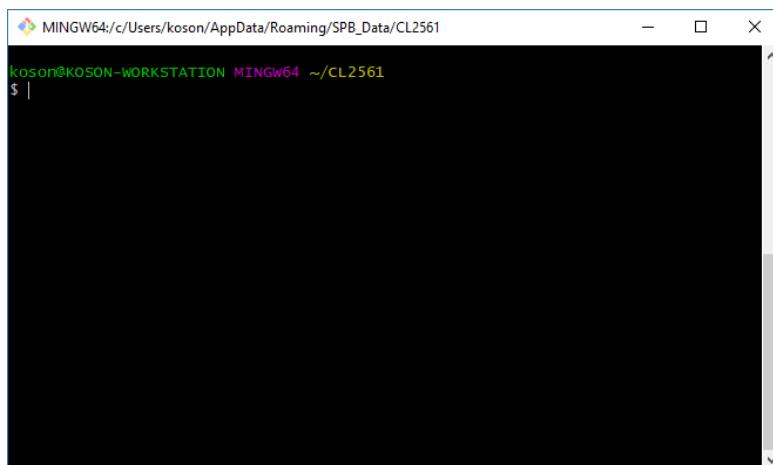
จาก เนื้อหาเรียนรู้ที่อบรมโดยอาจารย์ ดร. วิภาดา ธรรมชาติ夷งค์ (เนื้อหาเรียนรู้ CL2561 ย่อมาจาก Computer Laboratory 2561) โดยใช้คำสั่ง

\$ mkdir CL2561

- ย้ายเข้าไปอยู่ในโฟลเดอร์ที่สร้างขึ้น โดยใช้คำสั่ง

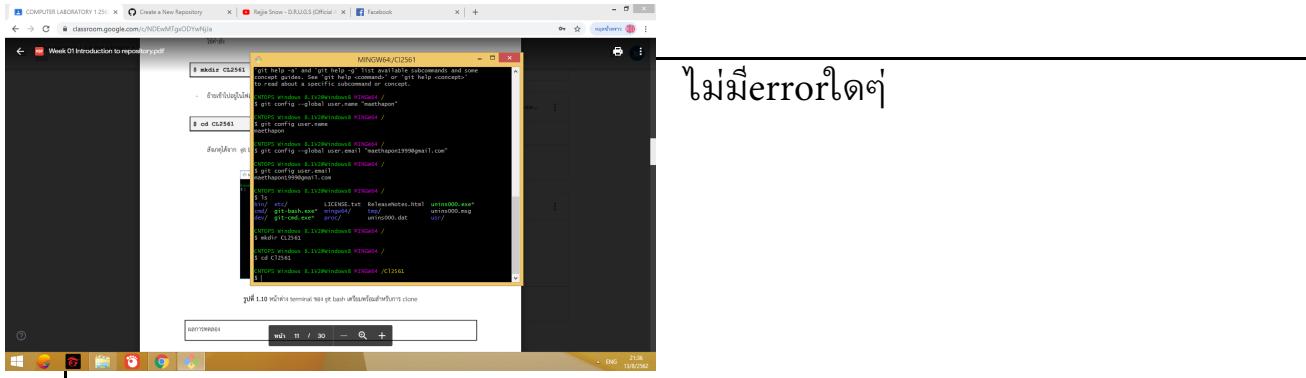
\$ cd CL2561

สังเกตุได้จาก git bash จะแสดงชื่อของโฟลเดอร์ปัจจุบันเป็นดังรูปที่ 1.10



รูปที่ 1.10 หน้าต่าง terminal ของ git bash เตรียมพร้อมสำหรับการ clone

ผลการทดลอง



2) การทำสำเนา repository มาไว้บนเครื่องโดยการ clone

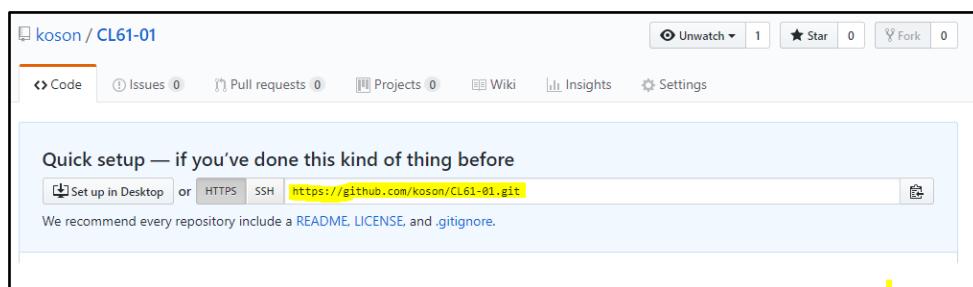
- ทำสำเนา repository มาไว้บนเครื่องโดยใช้คำสั่งที่มีรูปแบบดังต่อไปนี้

```
$ qit clone https://qithub.com/[YOUR USERNAME]/[YOUR REPOSITORY NAME]
```

[YOUR USERNAME] คือ username ของเราบน github

[YOUR REPOSITORY NAME] គឺជាឪីកូវិក repository ទូទៅនៃការងារ 1.3

ถ้าจำไม่ได้ ก็ไม่เป็นไร ให้เข้าไปที่ repository ที่เพิ่งสร้างบน Github (ดูรูปที่ 1.9) จะเห็นว่ามี URL ของ repository สำหรับการโคลน ดังรูปที่ 1.11 ให้ரากดปุ่ม copy ที่อยู่ด้านขวาเมื่อของ url



รูปที่ 1.11 URL สำหรับการ clone repository

ผลการทดสอบ

```

$ git clone https://github.com/[YOUR USERNAME]/[YOUR REPOSITORY NAME]
MINGW64UC2561
$ git config --global user.name "earthaporn"
$ git config --global user.email "earthaporn199@gmail.com"
$ git clone https://github.com/[YOUR USERNAME]/[YOUR REPOSITORY NAME]
Cloning into '[YOUR REPOSITORY NAME]'...
warning: You appear to have cloned an empty repository.

$ ls
LICENSE.txt  ReleaseNotes.html  uniso000.exe
cmd  git-sh-wash.exe  nsgd4d/  uniso000.dat  uniso000.msg
$ cd [YOUR REPOSITORY NAME]
$ dir
CL2561
$ git clone https://github.com/[YOUR USERNAME]/[YOUR REPOSITORY NAME]
Cloning into '[YOUR REPOSITORY NAME]'...
warning: You appear to have cloned an empty repository.

$ ls

```

Codeing ตามขั้นตอนและแสดงผลการทำงานของ Git

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

```

koson@KOSON-WORKSTATION MINGW64 ~/CL2561
$ git clone https://github.com/koson/CL61-01.git
Cloning into 'CL61-01'...
warning: You appear to have cloned an empty repository.

koson@KOSON-WORKSTATION MINGW64 ~/CL2561
$ |

```

รูปที่ 1.12 ผลการ clone repository

```

$ git clone https://github.com/[YOUR USERNAME]/[YOUR REPOSITORY NAME]
MINGW64UC2561
$ git config --global user.name "earthaporn"
$ git config --global user.email "earthaporn199@gmail.com"
$ git clone https://github.com/[YOUR USERNAME]/[YOUR REPOSITORY NAME]
Cloning into '[YOUR REPOSITORY NAME]'...
warning: You appear to have cloned an empty repository.

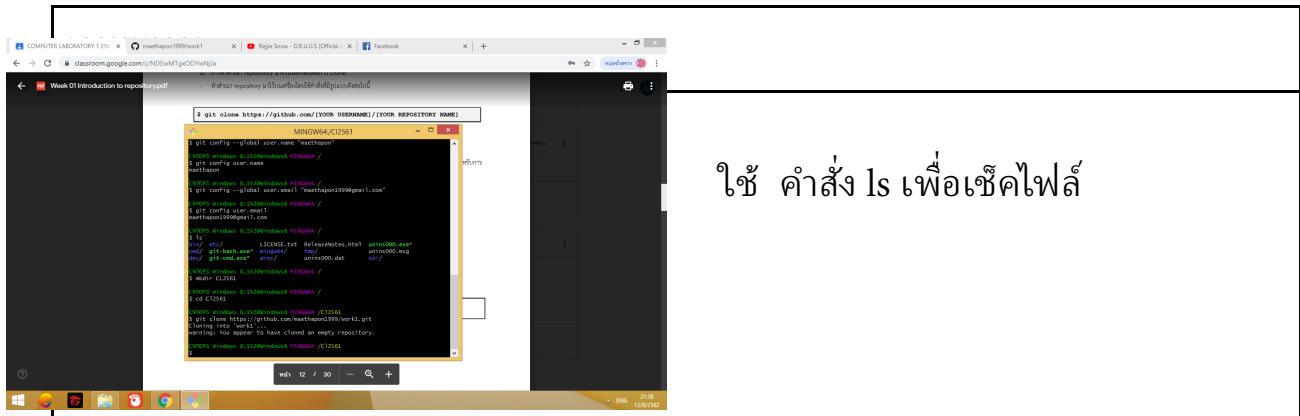
$ ls
LICENSE.txt  ReleaseNotes.html  uniso000.exe
cmd  git-sh-wash.exe  nsgd4d/  uniso000.dat  uniso000.msg
$ cd [YOUR REPOSITORY NAME]
$ dir
CL2561
$ git clone https://github.com/[YOUR USERNAME]/[YOUR REPOSITORY NAME]
Cloning into '[YOUR REPOSITORY NAME]'...
warning: You appear to have cloned an empty repository.

$ ls

```

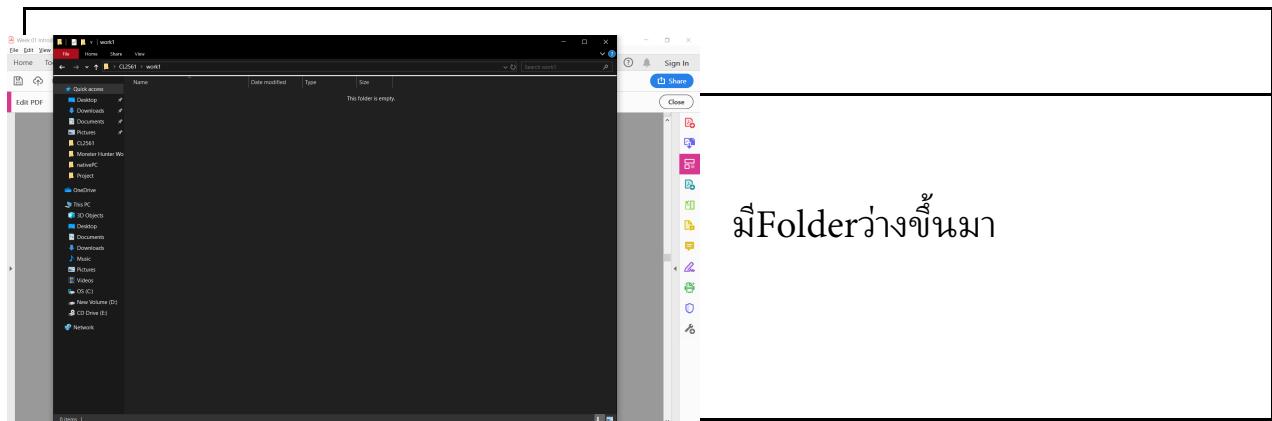
ขึ้น code ตามรายงานและไม่สามารถ Code ชี้ 2 ได้

- เรียกดูรายการไฟล์เดอร์ (ด้วยคำสั่ง ls) และเปลี่ยนไฟล์เดอร์ (ด้วยคำสั่ง change directory :cd)



ตอนแรกจะพบว่ามีไฟล์เดอร์ชื่อ CL61-01 ซึ่งถูก clone มาจาก server จึงย้ายเข้าไปในไฟล์เดอร์นั้นแล้วจึงสั่ง ls เพื่อดูรายการไฟล์ พบว่า repository ของเราจะยังว่างเปล่า ดังรูปที่ 1.13

รูปที่ 1.13 ไฟล์ที่ถูก clone มาจาก repository

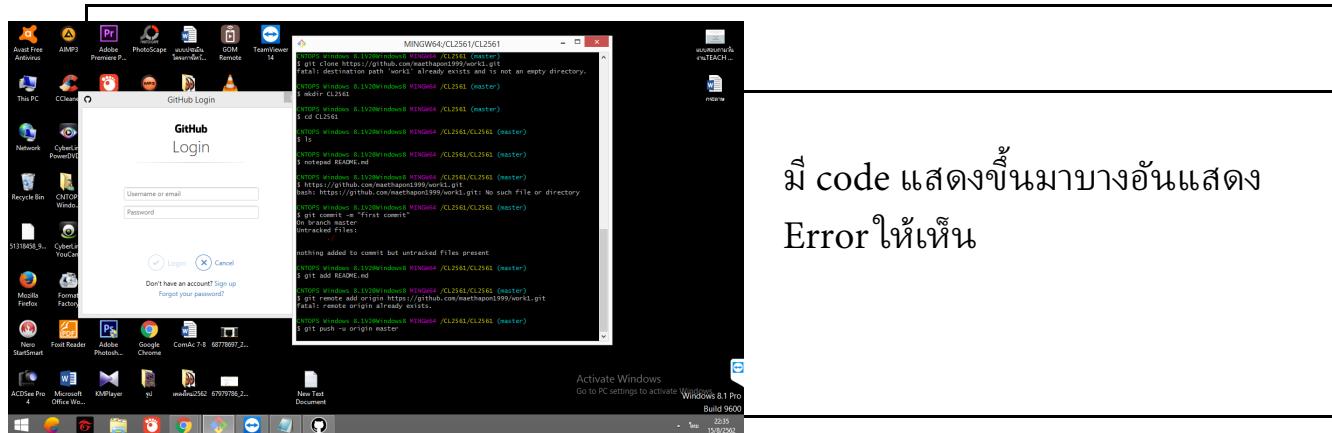


- ให้พิมคำสั่งต่อไปนี้ ครั้งละบรรทัด (พิมพ์ให้ครบบรรทัดแล้วเคาะ enter)

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

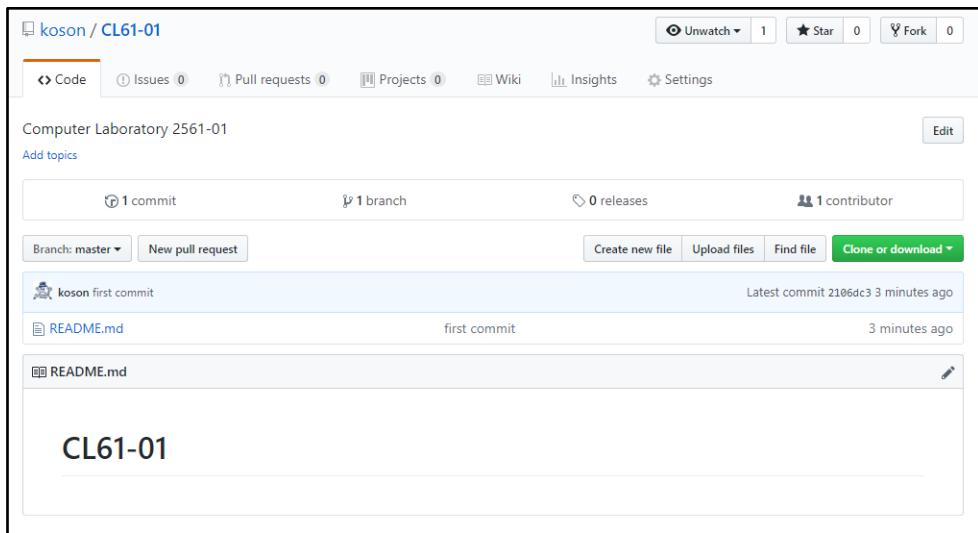
```
echo "# CL61-01" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/koson/CL61-01.git
git push -u origin master
```

รูปที่ 1.14 การเพิ่มไฟล์ README.md ให้กับ repository

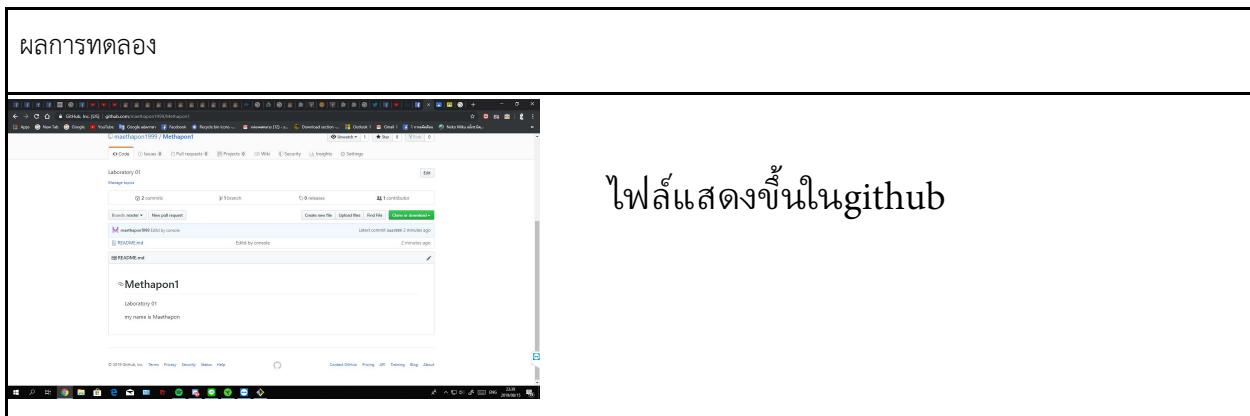


จากรูปที่ 1.14 จะได้ผลการทำงานดังรูปที่ 1.15 ซึ่งจะเห็นว่า บางคำสั่งอาจจะมี error เกิดขึ้น เนื่องจากมี repository อยู่บน server และ แต่ก็ให้ทำให้ครบถ้วนไปก่อน เพราะ ในกรณีนี้ error เหล่านั้นไม่ส่งผลกระทบร้ายแรงต่อการทำงาน

- ให้กลับไปที่ browser และกด refresh 1 ครั้ง จะเห็นว่าหน้า repository ที่เราเพิ่งสร้าง จะเปลี่ยนไป



รูปที่ 1.15 หน้าจอ repository ที่เปลี่ยนไปหลังจากเพิ่มไฟล์ README.md



1.5 การแก้ไขงานและบันทึกการเปลี่ยนแปลงบน local computer

ถึงตอนนี้ เนื้อหาในไฟล์ README.md บน server และ local computer จะเหมือนกันทุกประการ เนื่องจากเป็นการ clone มาและยังไม่ได้ทำการแก้ไขใดๆ อีกทั้งเรามั่นใจว่าไม่มีผู้เขียนอื่นๆ กำลังแก้ไขงานของเรานบน server (ซึ่งการแก้ไขงานร่วมกันบน server จะอยู่ในการทดลองถัดไป) เราสามารถแก้ไขและทำ revision ของเอกสารได้ตามต้องการ โดยการเปลี่ยนแปลงต่างๆ จะเกิดขึ้นบนเครื่อง local computer เท่านั้น

1.5.1 ทดลองแก้ไขไฟล์ README.md

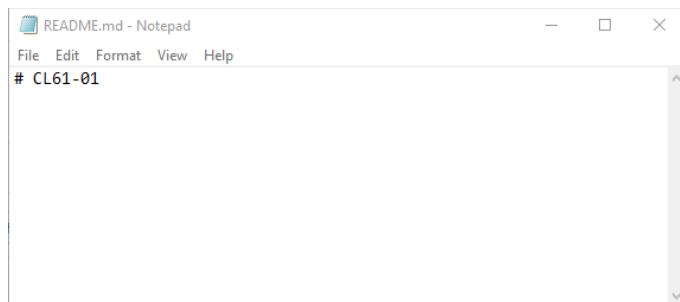
โดยส่วนใหญ่ในการเขียนโปรแกรม มักจะกระทำบนโปรแกรม Integrated development environment หรือเรียกว่า IDE⁵ แต่ในการทดลองนี้ จะใช้โปรแกรมแก้ไขเอกสารอย่างง่ายๆ นั่นคือโปรแกรม Notepad.exe

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

- ให้พิมพ์คำสั่งต่อไปนี้ลงใน git bash

```
$ notepad README.md
```

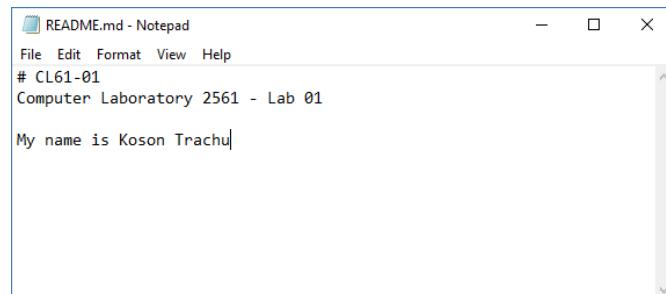
ระบบจะเปิด text editor ที่มากับระบบปฏิบัติการ Windows ดังรูปที่ 16



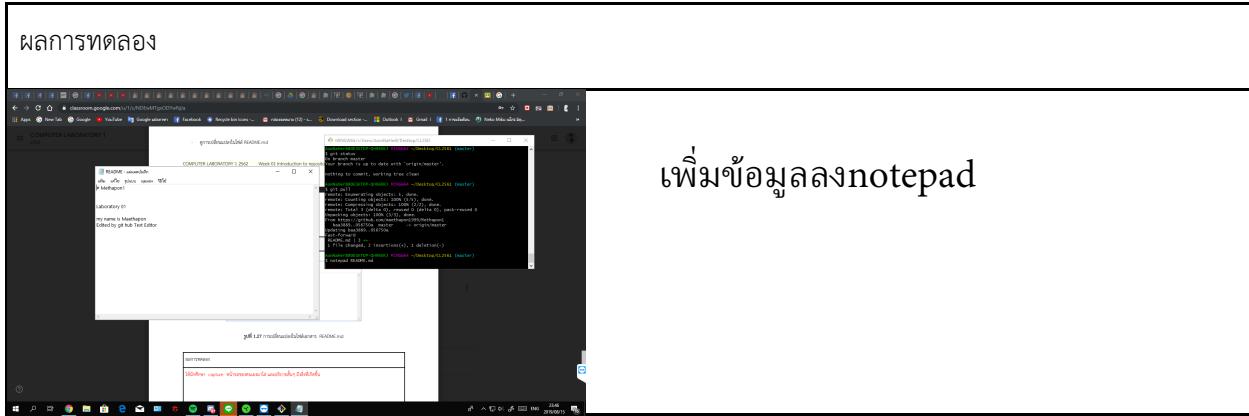
รูปที่ 1.16 การใช้โปรแกรม notepad.exe แก้ไขไฟล์ README.md



- แก้ไขไฟล์ README.md ใน notepad โดยเพิ่มข้อความลงไปดังตัวอย่าง (ให้นักศึกษาใส่ชื่อตนเอง)



รูปที่ 1.17 แก้ไขไฟล์ README.md โดยเพิ่มบรรทัดต่อท้ายเข้าไป



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

\$ git status

```
MINGW64:/c/Users/koson/AppData/Roaming/SPB_Data/CL2561/CL61-01
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/koson/CL61-01.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

koson@KOSON-WORKSTATION MINGW64 ~/CL2561/CL61-01 (master)
$ notepad README.md

koson@KOSON-WORKSTATION MINGW64 ~/CL2561/CL61-01 (master)
$ git status
on branch master
Your branch is up to date with 'origin/master'.

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

    modified:   README.md

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

koson@KOSON-WORKSTATION MINGW64 ~/CL2561/CL61-01 (master)
$
```

รูปที่ 1.18 การตรวจสอบสถานะของ git

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

ผลการทดลอง

The screenshot shows a PDF editor interface with a terminal window open. The terminal window displays the following command and its output:

```
git add README.md
On branch master
Changes staged to date with "origin/master".
 1 file changed, 1 insertion(+)
.gitignore captured 为已修改的文件并将其添加到本地仓库
```

จาก Code แสดงถึงการเปลี่ยนแปลงข้อมูลของ README.Md

1.5.2 บันทึกการเปลี่ยนแปลงบน local computer

ถึงตรงนี้ ถ้าเราต้องการจะแก้ไขต่อ ก็สามารถทำได้ แต่การเปลี่ยนแปลงต่างๆ จะไม่สามารถถูกติดตามโดย git ถ้าต้องการให้ git บันทึก (หรือบัน) การเปลี่ยนแปลงเป็นรุ่นหนึ่งๆ 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 และสังเกตผลที่ได้จากการรันคำสั่ง

ผลการทดลอง

The screenshot shows a PDF editor interface with a terminal window open. The terminal window displays the following command and its output:

```
git add README.md
On branch master
Changes to be committed:
  (use "git add --interactive" to select changes to stage)
    new file: README.md
.gitignore captured 为已修改的文件并将其添加到本地仓库
```

มีการเปลี่ยนแปลง จาก Modified เป็นสีแดง เปลี่ยนเป็นสีเขียว

```
$ git status
```

```

MINGW64:/c/Users/koson/AppData/Roaming/SPB_Data/CL2561/CL61-01
modified: README.md
no changes added to commit (use "git add" and/or "git commit -a")
koson@KOSON-WORKSTATION MINGW64 ~/CL2561/CL61-01 (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.

koson@KOSON-WORKSTATION MINGW64 ~/CL2561/CL61-01 (master)
$ git status
on branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified: README.md

koson@KOSON-WORKSTATION MINGW64 ~/CL2561/CL61-01 (master)
$ |

```

รูปที่ 1.19 การตรวจสอบสถานะของ git / ผลจากการทำคำสั่ง git add

หมายเหตุ หากมีการแก้ไขหลายไฟล์ เราอาจใช้คำสั่ง git add --all แทนการใช้ชื่อไฟล์ได้

- Commit ไฟล์ที่เปลี่ยนแปลง เข้าสู่ repository โดยใช้คำสั่งต่อไปนี้

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

ตามด้วยการตรวจสอบสถานะของ repository

```
$ git status
```

จะได้ผลดังนี้

```

MINGW64:/c/Users/koson/AppData/Roaming/SPB_Data/CL2561/CL61-01
koson@KOSON-WORKSTATION MINGW64 ~/CL2561/CL61-01 (master)
$ git commit -m"edited by Koson"
[master ee1c2e8] edited by Koson
 1 file changed, 3 insertions(+)

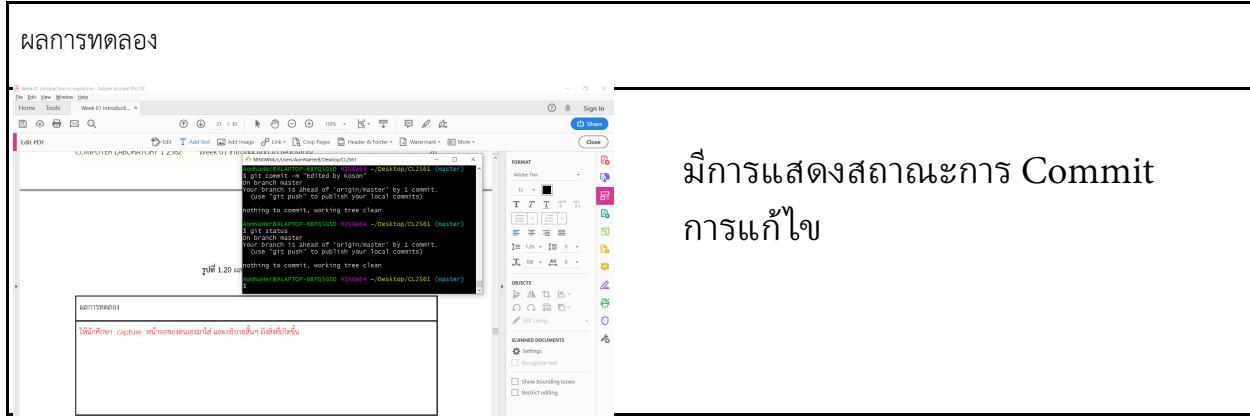
koson@KOSON-WORKSTATION MINGW64 ~/CL2561/CL61-01 (master)
$ git status
on branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

koson@KOSON-WORKSTATION MINGW64 ~/CL2561/CL61-01 (master)
$ |

```

รูปที่ 1.20 ผลจากการทำ git commit



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

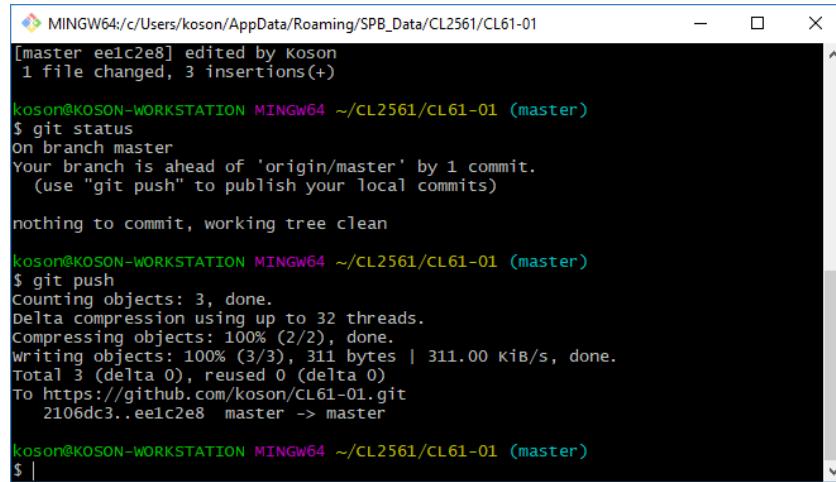
หลังจากที่เราได้ทำการ clone repository มาที่ local computer และ การแก้ไขงานทั้งหมด สามารถทำได้บน local computer ได้โดยไม่ต้องเชื่อมต่อ กับ server และในบางครั้งที่มีการทำงานร่วมกันเป็นทีม จะต้องปรับปรุง source code ให้เป็นปัจจุบันอยู่เสมอ จะต้องมีการ sync กับ server ได้แก่การ upload การเปลี่ยนแปลงขึ้นสู่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

1.6.1 การ push ขึ้นสู่ server

โดยทั่วไป การที่จะ push ขึ้นสู่ server เราอาจจะใช้คำสั่ง 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:/c/Users/koson/AppData/Roaming/SPB_Data/CL2561/CL61-01
[master ee1c2e8] edited by Koson
 1 file changed, 3 insertions(+)

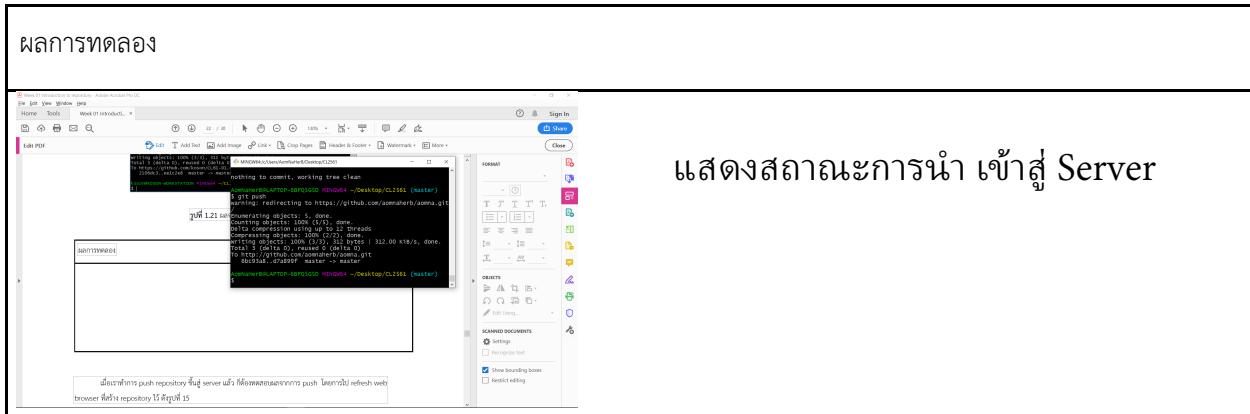
koson@KOSON-WORKSTATION MINGW64 ~/CL2561/CL61-01 (master)
$ git status
on branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

koson@KOSON-WORKSTATION MINGW64 ~/CL2561/CL61-01 (master)
$ git push
Counting objects: 3, done.
Delta compression using up to 32 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 311 bytes | 311.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/koson/CL61-01.git
  2106dc3..ee1c2e8  master -> master

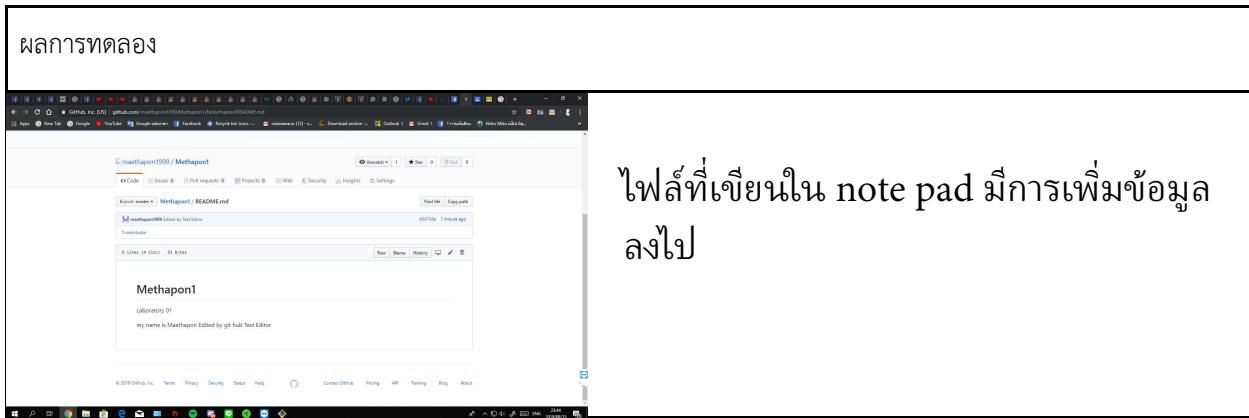
koson@KOSON-WORKSTATION MINGW64 ~/CL2561/CL61-01 (master)
$ |
```

รูปที่ 1.21 ผลจากการทำคำสั่ง git push



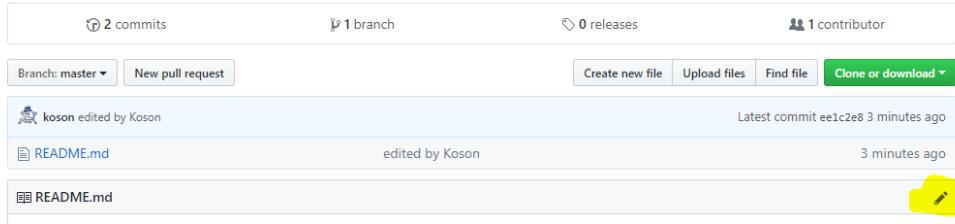
เมื่อเราทำการ push repository ขึ้นสู่ server และ ก็ต้องทดสอบรายการ push โดยการไป refresh web browser ที่สร้าง repository ไว้ ดังรูปที่ 15

รูปที่ 1.22 การเปลี่ยนแปลงที่เกิดขึ้นบน server

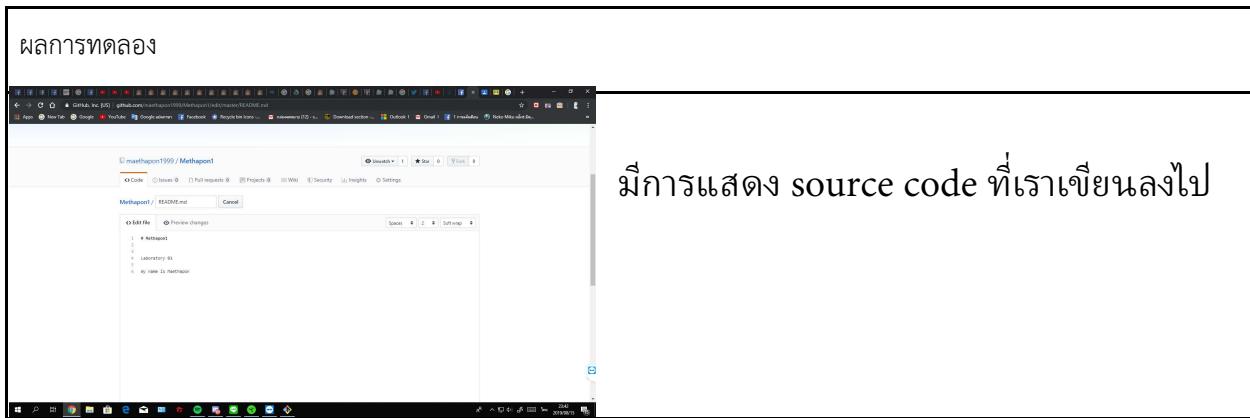


1.6.2 การ pull มาจาก server

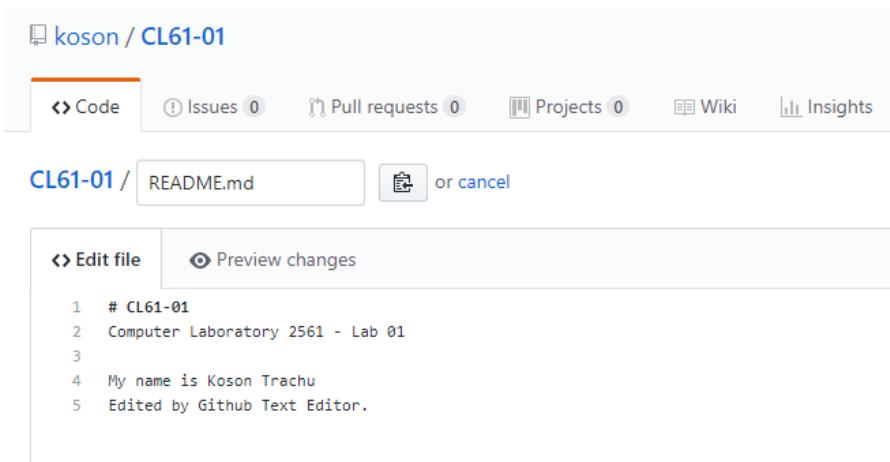
- การเปลี่ยนแปลงใดๆ ที่เกิดขึ้นบน local computer จะถูกส่งขึ้นมาเก็บด้วยคำสั่ง git push และถ้ามีการแก้ไขไฟล์ใดๆ ก็ต้องทำการ pull ที่จะดึงกลับไปทำงานที่ local computer ได้ เช่นกัน
- ให้แก้ไขไฟล์ README.md โดยการคลิกที่ชื่อไฟล์ และปุ่มปักกาบริเวณด้านขวาเมื่อ



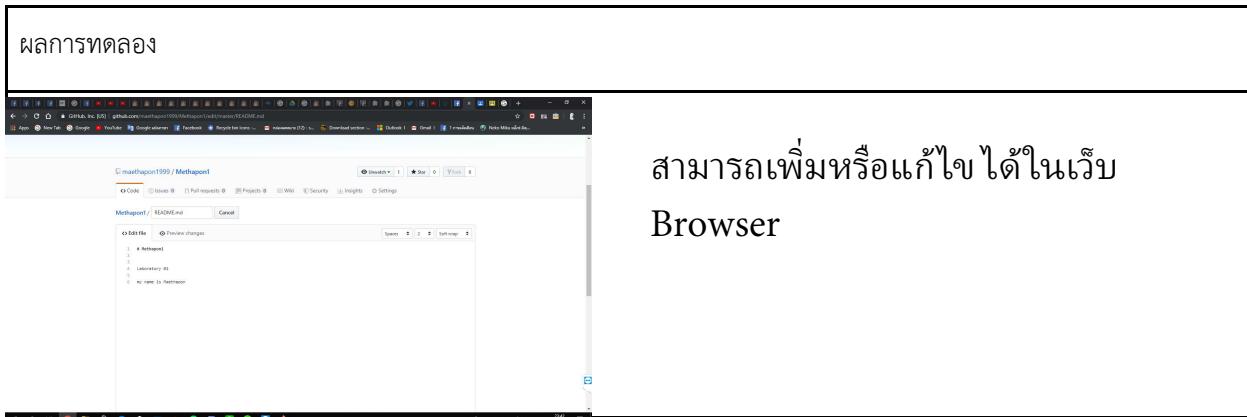
รูปที่ 1.23 เข้าสู่โหมดการแก้ไขไฟล์ด้วย Github Text Editor



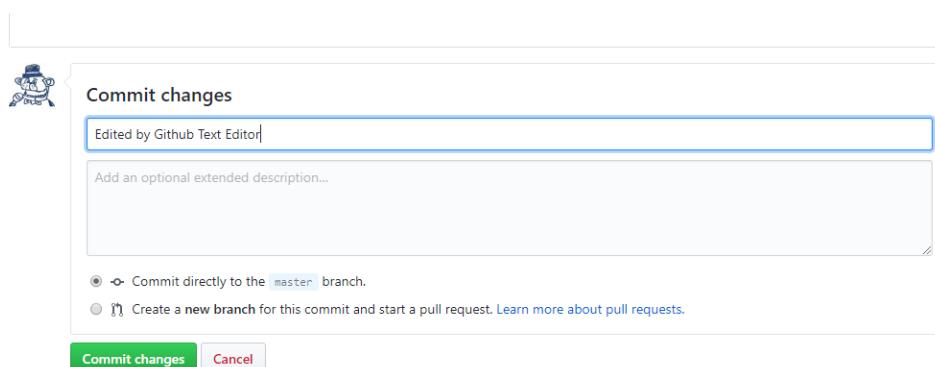
- เพิ่มข้อความที่บรรยายถึงสุดยอดตัวอย่าง



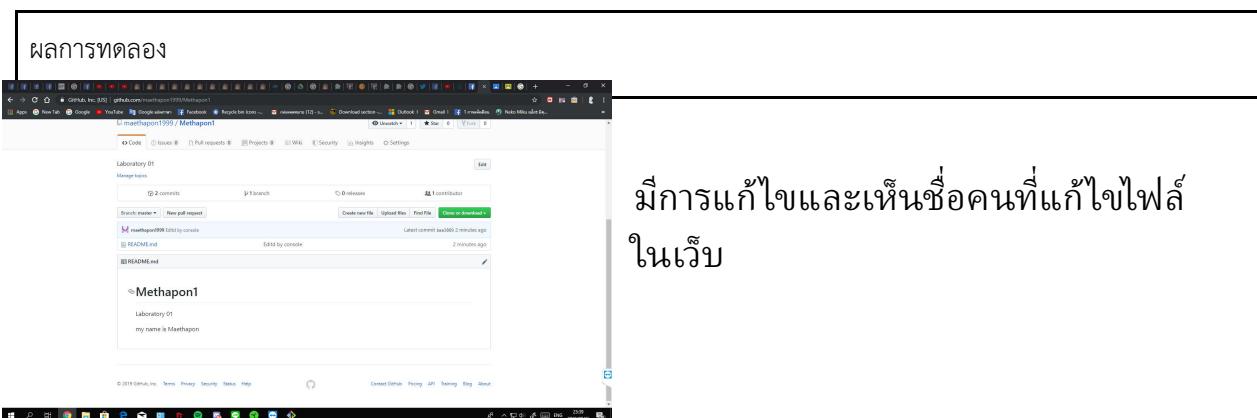
รูปที่ 1.24 เพิ่มข้อความบางอย่างใน Github Text Editor



- เพิ่มข้อความในช่อง Commit changes และกดปุ่ม Commit changes สีเขียว



รูปที่ 1.25 เพิ่มข้อความ Commit changes



- กลับมาที่ git bash พิมพ์คำสั่ง git status สังเกตผลการทำงาน

```
$ git status
```

- ที่ git bash พิมพ์คำสั่ง git pull

```
$ git pull
```

```
MINGW64:/c/Users/koson/AppData/Roaming/SPB_Data/CL2561/CL61-01
koson@KOSON-WORKSTATION MINGW64 ~/CL2561/CL61-01 (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean

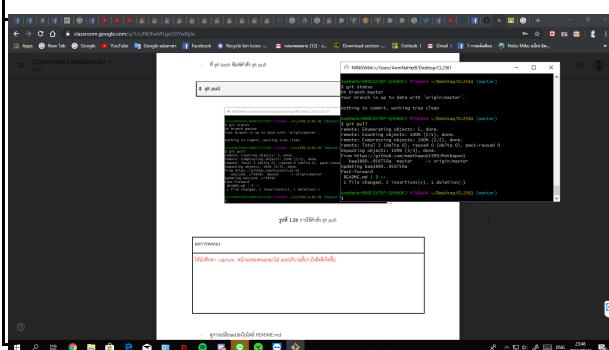
koson@KOSON-WORKSTATION MINGW64 ~/CL2561/CL61-01 (master)
$ git pull
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/koson/CL61-01
      ee1c2e8..c739591  master      -> origin/master
Updating ee1c2e8..c739591
Fast-forward
 README.md | 3 +++
 1 file changed, 2 insertions(+), 1 deletion(-)

koson@KOSON-WORKSTATION MINGW64 ~/CL2561/CL61-01 (master)
$
```

รูปที่ 1.26 การใช้คำสั่ง git pull

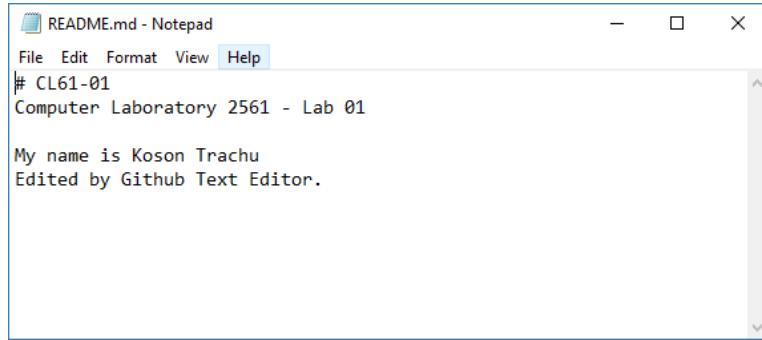
ผลการทดลอง

มี code แสดงถึงการดึงไฟล์มาจาก Github
ของเรา



- ดูการเปลี่ยนแปลงในไฟล์ README.md

\$ notepad.exe README.md



รูปที่ 1.27 การเปลี่ยนแปลงในไฟล์เอกสาร README.md

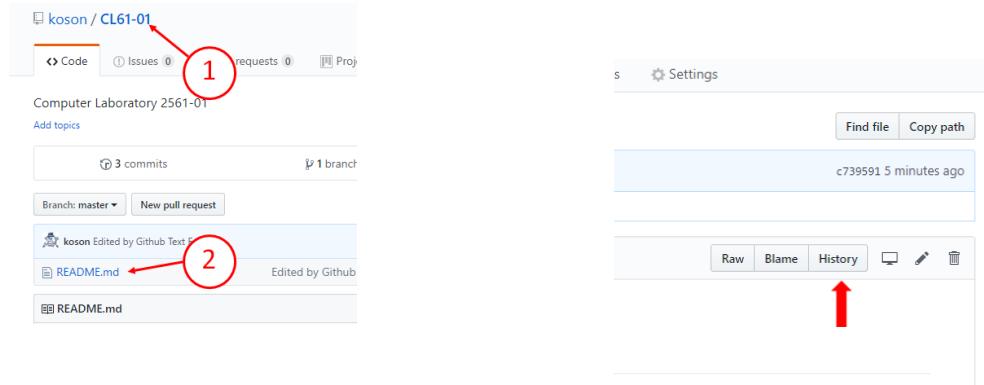
ผลการทดลอง

A screenshot of the Adobe Acrobat DC software interface. The main window shows a PDF document titled "README.pdf" which contains the same text as the Notepad file: "# CL61-01", "Computer Laboratory 2561 - Lab 01", "My name is Keson Trachu", and "Edited by Github Text Editor.". The right side of the screen has a large text area with the overlaid text: "จ้างในไฟล์มีการแก้ไขเมื่อนกับที่ แก้ไข เว็บไซต์".

จ้างในไฟล์มีการแก้ไขเมื่อนกับที่ แก้ไข เว็บไซต์

1.7 การตรวจสอบประวัติการเปลี่ยนแปลงของไฟล์

- กดลับไปที่ web browser (1) คลิกที่ชื่อ repository, (2) คลิกที่ชื่อ repository



(ก) เลือกชื่อ repository, ชื่อไฟล์

(ข) คลิกที่ปุ่ม History

รูปที่ 1.28 การเข้าถึงประวัติของไฟล์

เมื่อคลิกดูประวัติไฟล์ จะพบว่า ไม่ว่าเราจะแก้ไขไฟล์ที่ไหน แต่ Git จะติดตามและบันทึกการเปลี่ยนแปลงทุกครั้งที่เราทำการ commit

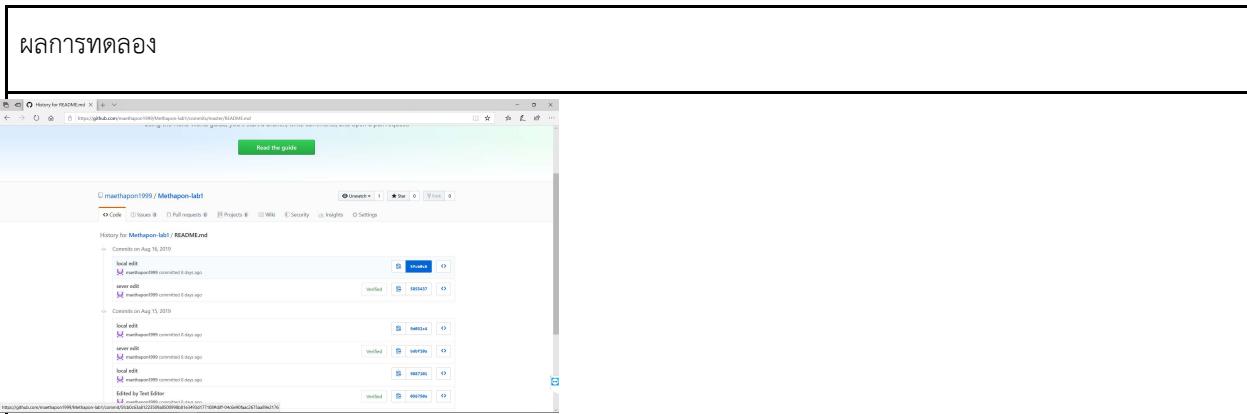
ผลการทดลอง

แสดงประวัติการ commit ของเราว่ามี
ใครได้แก้ไขหรือไม่

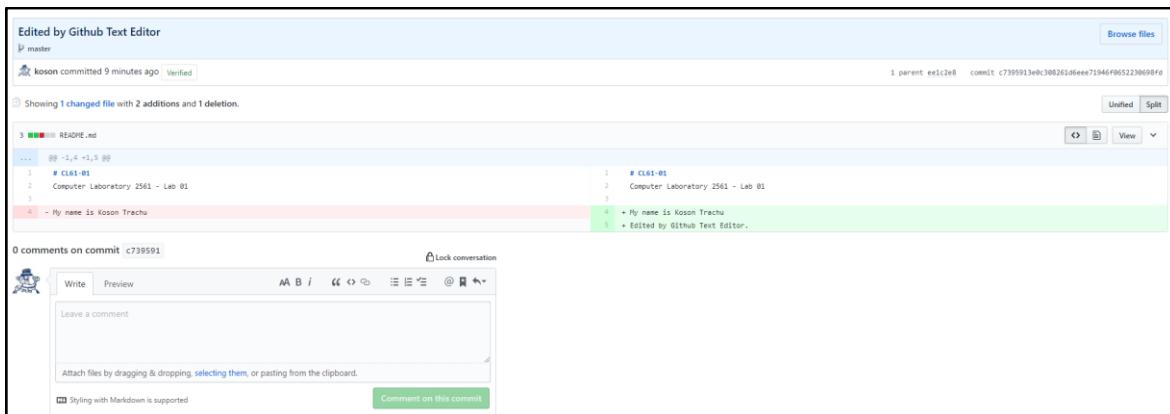
History for CL61-01 / README.md

แสดงประวัติการ commit ของเราว่ามี
ใครได้แก้ไขหรือไม่

รูปที่ 1.29 รายการประวัติการแก้ไขไฟล์



ให้คลิกปุ่มที่มีเลขฐาน 16 กำกับ (เป็นชื่อรหัสกำกับการแก้ไข ที่ทีมพัฒนาจะใช้อ้างอิงถึง) ตามลูกศรสีแดงในรูปที่ 29 เวลาเท่านั้นประวัติการแก้ไขไฟล์ ดังรูปที่ 30



รูปที่ 1.30 ประวัติการแก้ไขไฟล์

แบบฝึกหัด

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

ลำดับที่	สถานที่แก้ไข	สิ่งที่กระทำ
1	Local (Notepad)	ลบเนื้อหาเดิมออกทั้งหมด
2	Server (Github Text Editor)	#include < stdio.h > main() { printf ("hello, world\n"); }
3	Local (Notepad)	เปลี่ยน printf("hello, world\n"); เป็น printf("hello, [ชื่อนักศึกษา]\n");
4	Server (Github Text Editor)	#include <stdio.h> int main () { char yourname[100]; printf("What is your name?\t"); scanf("%s",yourname); printf("hello, %s\n", yourname); }
5	Local (Notepad)	เพิ่ม printf("Goodbye\n"); ใต้ printf("hello, %s\n", yourname);

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

คำถาม

- จากภาพที่ 29 ถ้าหากนักศึกษาคลิกตามปุ่ม ที่มีเลขอารบิกกำกับอยู่ ทุกปุ่ม จะได้ผลอย่างไรบ้าง ให้อธิบายสิ่งที่พบรหัส = จะแสดงประวัติการแก้ไขข้อมูลของเรา
- ให้บอกรายละเอียดของ repository ตามที่นักศึกษาเข้าใจ = เราสามารถนำไฟล์ของผู้อื่นมาแก้ไขและต่อยอดได้และสามารถนำไฟล์ให้ผู้อื่นช่วยแก้ไขได้
- ให้บอกรแนวทางการนำ repository ไปใช้ในการเรียนหรือชีวิตประจำวันของนักศึกษา = เราสามารถนำไฟล์มาพัฒนาและต่อยอดให้ดีขึ้นได้และมีโฉดหลากหลายให้เลือก