**การทดลองที่ 1 การใช้งาน Repository เบื้องต้น**

# วัตถุประสงค์

1. เพื่อให้ผู้เรียนรู้และเข้าใจแนวคดิ ในการใช้ Repository
2. เพื่อให้ผู้เรียนสามารถใช้ Repository (Github) เบื้องต้นได้

**ทฤษฎีก่อนการทดลอง**

# Git

**Git[[1]](#footnote-1)** เป็นระบบควบคุมเวอร์ชั่น (Version control systems) แบบ open source เป็นเครื่องมือที่ใช้บริหารจัดการการ เปลี่ยนแปลงของไฟล์ต่าง ๆ ใน project การบันทึกการแก้ไขไฟล์แต่ละครั้งจะเรียกว่ารุ่น (revision) ซึ่งแต่ละรุ่นของการ เปลี่ยนแปลงจะถูกก ากับด้วยการประทับเวลา (timestamp) และบุคคลที่ท าการเปลี่ยนแปลง ดังนั้น หากเกิดความผิดพลาดหรือ เสียหายจากการแก้ไข เราก็จะสามารถย้อนเวลากลับไปยังการแก้ไขครั้งก่อนๆ ที่สมบูรณ์ได้ตามต้องการ ถือได้ว่าระบบควบคุมเวอร์ ชั่นเป็นระบบพื้นฐานที่นิยมใช้ในการบริการจัดการ source code ของโปรแกรม ซึ่งจริง ๆ แล้ว เราสามารถใช้ระบบควบคุมเวอร์ ชั่นกับไฟล์ชนิดใดๆ หรืองานชนิดใดๆ ก็ได้ ไม่เฉพาะ source code ของโปรแกรมเท่านั้น ในปัจจุบัน มีระบบควบคุมเวอร์ชั่นให้ เลือกใช้หลากหลาย ทั้งเป็นแบบฟรีและมีค่าใช้จ่าย (เช่น Git, Mercurial, Subversion) โดย Git จะได้รับความนิยมมากกว่าชนิด อื่นๆ การท างานของ Git นั้นจะมีพื้นที่เก็บไฟล์ ซึ่งเรียกว่า ‘repositories’ ซึ่งเราสามารถติดตั้งบริการ git บน server ใดๆ ก็ได้ แต่ server บริการ git ที่ได้รับความนิยมในปัจจุบันได้แก่ Github, Gitlab, Bitbucket เป็นต้น ข้อดีของการใช้ server รวมก็คือ สามารถแบ่งปันและร่วมมือ ช่วยเหลือกันในแก้ไขโปรแกรมได้จากทุกคนทั่วโลก ลักษณะเฉพาะอย่างหนึ่งของ Git ก็คือ ใน folder ที่ชื่อ .git บนคอมพิวเตอร์ของเราจะเก็บทุกสิ่งที่เก็บบน server จึงมั่นใจได้ว่า เราสามารถท างานกับระบบควบคุมเวอร์ชันได้ทั้ง แบบออนไลน์และออฟไลน์ และหากเกิดกรณีที่ repository บน server เสียหาย เราก็สามารถน าทุกอย่างที่เก็บบนเครื่องกลับขึ้น ไปเก็บบน server ได้

# Github

Github เป็นบริษัทหนึ่ง ที่ให้บริการ Git repository บนพื้นฐานของเว็บ (web-based Git repository hosting) โดย Github จะ ให้พื้นที่เราสร้าง repository ส าหรับโปรเจค ให้บริการฟังก์ชันการท างานพื้นฐานของระบบ git เช่น การ branches, merges,

และ commits อีกทั้งยังให้พื้นที่ส าหรับแจ้งข้อผิดพลาด บัก หรือความต้องการเพิ่มเติม features ต่างๆ ตลอดจนมีความสามารถ ในการเขียนค าอธิบายแบบ wiki ใน repository นั้น ๆ ด้วย Github เป็นบริษัทที่มีมูลค่าประมาณ 2 พันล้าน USD, มีผู้ใช้ ประมาณ 20 ล้านคน มี repositories ประมาณ 40 ล้าน และในจ านวนเหล่านั้น มีโปรเจคที่ส าคัญมากรวมอยู่ด้วย เช่น kernel ของ Linux , source code ของ dotnet framework จากไมโครซอฟท์ และอื่นๆ ท าให้มีความมั่นใจในระดับหนึ่งว่าถ้า Github เกิดล่มขึ้นมา ก็จะมีเพื่อนร่วมชะตากรรมอีกไม่น้อย

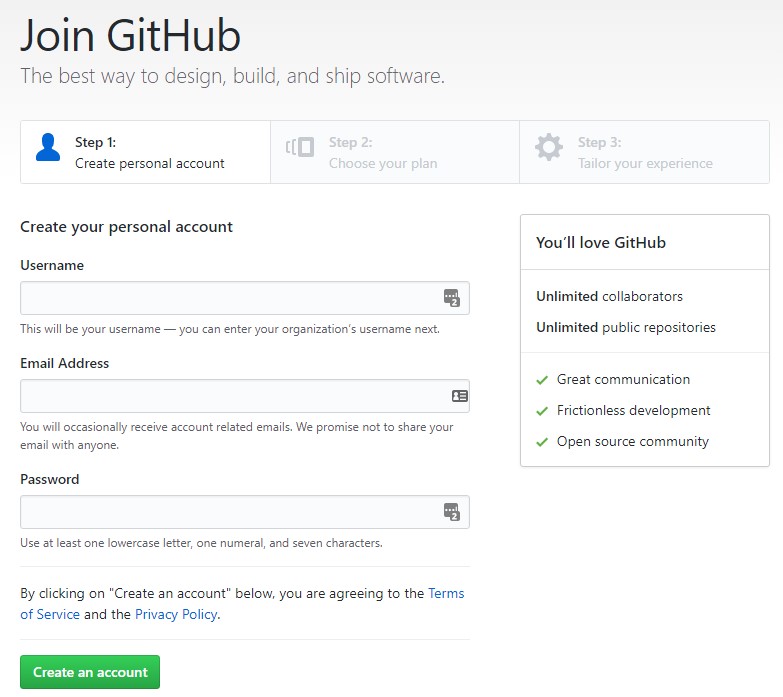
**ขั้นตอนการทดลอง**

1. **เริ่มใช้งาน** **Github**  ในการใช้งาน Github เราจะต้องมีบัญชีผู้ใช้ของ Github ซึ่งทาง Github จะให้บริการฟรีแบบไม่จ ากัดจ านวน

repository ซึ่งจะเป็นแบบ public หรือ private ก็ได้ repository แบบ public นั้น จะสามารถมองเห็นได้จากทุกคน ส่วน repository ที่เป็นแบบ private เราจะสามารถก าหนดบุคคลที่อนุญาตให้เห็น repository ของเราได้ ซึ่งจะสะดวกในการท า project ที่เป็นความลับ

**1.1 สร้างบัญชีผู้ใช้งานบน** **Github**  การสร้างบัญชีผู้ใช้บน Github ให้ไปที่<https://github.com/join>จากนั้น ให้กรอกรายละเอียด ซึ่งชื่อผู้ใช้ (User

name) จะถูกน าไปใช้ในหลายๆ ที่ ดังนั้นควรเป็นชื่อที่จ าง่ายและพิมพ์ได้สะดวก มิฉะนั้นจะเสียเวลาในการท างาน



**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 ตามค าแนะน าของโปรแกรมติดตั้ง

1.2.3 เปิดโปรแกรม Git bash จะได้หน้าต่าง terminal ที่ท างานใน text mode

**รูปที่**

**1.**

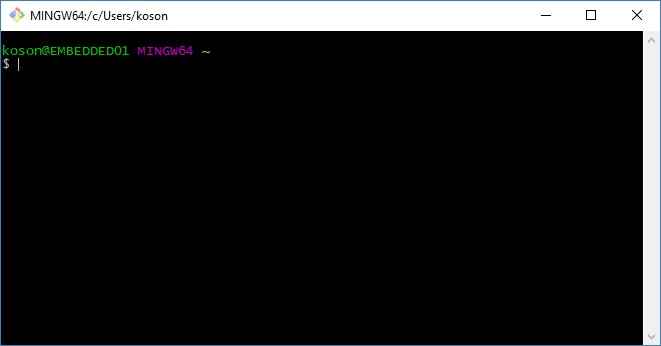
**2**

หน้าต่าง

terminal

ของ

git bash



|  |
| --- |
| ผลการทดลอง |
|  |

1.2.4 ทดสอบว่าสามารถใช้งาน Git บนเครื่องของเราได้หรือไม่ ให้พมิ พ์ค าสั่งต่อไปนี้

**$ git**

ถ้า terminal ตอบกลับมาว่าไม่รจู้ กั ค าสั่ง git แสดงว่าการติดตั้งยังไมส่ มบูรณ์ ให้กลับไปตรวจสอบขั้นตอน 1.2.2 ให้ติดตั้ง เรียบร้อย

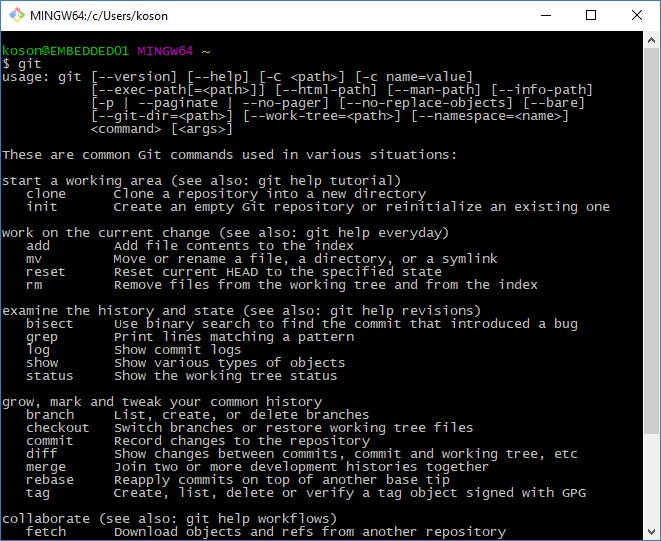
**รูปที่**

**1.**

**3**

ผลการทดลองพิมพ์ค าสงั่

git



|  |
| --- |
| ผลการทดลอง |
|  |

1.2.5 บอกให้ Git รู้จักชื่อของเรา โดยพิมพ์ค าสั่งต่อไปนี้ [[2]](#footnote-2)

**$ git config --global user.name “USER NAME”**

ในกรณีที่เราต้องการทราบชื่อผู้ใชป้ ัจจุบัน สามารถสั่งให้ Git รายงานออกมาด้วยการพิมพ์ค าสั่งต่อไปนี้

**$ git config user.name**

**รูปที่**

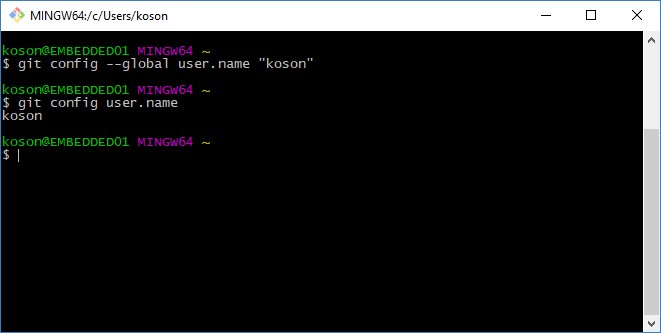
**1.**

**4**

git config

--

global user.name



|  |
| --- |
| ผลการทดลอง |
|  |

1.2.6 บอกให้ Git รู้จัก email ของเรา โดยพิมพ์ค าสั่งต่อไปนี้

**$ git config --global user.email “USER EMAIL ADDRESS”**

ในกรณีที่เราต้องการทราบชื่อผู้ใชป้ ัจจุบัน สามารถสั่งให้ Git รายงานออกมาด้วยการพิมพ์ค าสั่งต่อไปนี้

**$ git config user.email**

หมายเหตุ email ที่ใช้จะต้องตรงกับ email ที่ลงทะเบียนไว้กับ Github มิฉะนั้นจะไมส่ ามารถเขียนข้อมูลขึ้นไปบน

## server ได้

เมื่อท าในขั้นตอน 1.2.5 และ 1.2.6 เรียบร้อยแล้ว การท างานใดๆ บน Github ก็จะปรากฏชื่อและ Email ของเราก ากับ ไว้เสมอ

**รูปที่**

**1.**

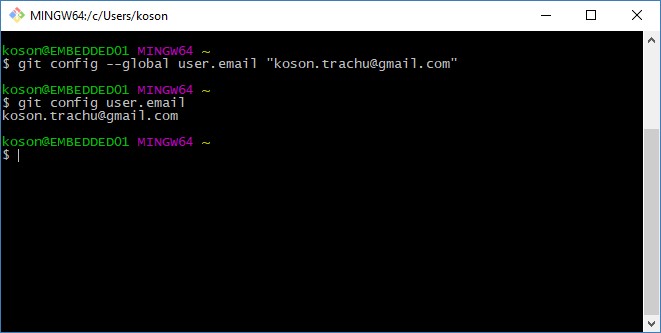
**5**

git config

--

global

user.email



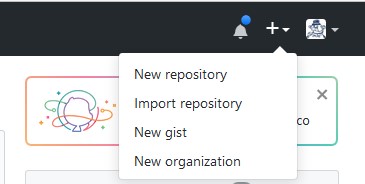
|  |
| --- |
| ผลการทดลอง |
|  |

### **1.3 สร้าง repository (บน server)**

Repository เป็นพื้นที่ส าหรับเกบ็ project ของเรา ซึ่งไม่ได้หมายความถึงเฉพาะ source code เท่านั้น repository ยัง สามารถประกอบด้วยไฟล์ทุกชนิด ไม่ว่าจะเป็น Word Document, spread sheet, presentation, เอกสารการออกวิเคราะห์และ ออกแบบซอฟต์แวร์ ไฟลม์ ีเดยี ภาพและเสียง รวมไปถึงเอกสาร Wiki ในลักษณะ html ด้วย ดังนั้น ในการท าโครงการพัฒนา ซอฟต์แวร์ เราสามารถน าทุกสิ่งทจี่ าเป็นส าหรับการท างาน มาใสไ่ วใ้ น repository และเมื่อเพื่อนร่วมทีมหรือ user ใดๆ ท าส าเนา repository ของเราไป เขาก็จะได้ทุกสิ่งทุกอย่างไปอย่างครบถ้วน ดังนั้นจึงอาจพูดได้ว่าเราสามารถใช้ repository เป็นเครื่องมือ บริหารโครงการทมี่ ีประสิทธิภาพได้เช่นกัน

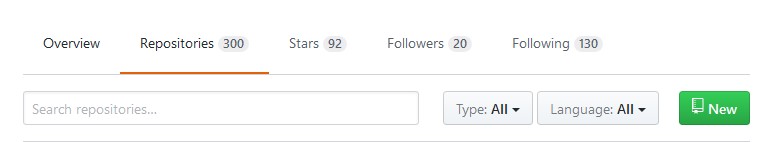
1.3.1 การสร้าง repository บน Github สามารถสรา้ งได้หลายวิธดี ้วยกัน เช่น

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



**รูปที่ 1.6** การสร้าง repositoryโดยการคลิกที่ปุ่มเครื่องหมาย “+”

(2) การสร้าง repository โดยการคลิกที่ปุ่ม New สีเขียว



**รูปที่ 1.7** การสร้าง repositoryโดยการคลิกที่ปุ่ม New

(3) การสร้าง repository โดยลิง[ค](https://github.com/new)์<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-line](https://help.github.com/articles/adding-an-existing-project-to-github-using-the-command-line/)[[3]](#footnote-3))

1.3.2 ก าหนดชื่อและชนิดของ repository

การใช้วิธีการ 3 วิธีแรก ในข้อ 1.3.1 จะไดผ้ ลอย่างเดียวกัน คือ Github จะพามาหน้าส าหรับสร้าง repository

* ในช่อง Repository name ให้ใสช่ ื่อของ repository เนื่องจากบ่อยครั้งที่เราต้องใช้งานค าสั่งต่าง ๆ บน terminal ซึ่ง ต้องพิมพ์ชื่อ repository เอง ดังนั้นชื่อของ repository จะต้องมีความหมายในตัว เข้าใจง่าย กระชับ
* ในช่อง Description (optional) เพิ่มค าอธิบายสั้นๆ เกี่ยวกับ repository เพื่อให้ชาวโลกอ่านแล้วเหน็ ภาพรวมของ repository ได้อย่างรวดเร็ว

* ชนิดของ repository นั้น ถ้าหากเป็นโปรเจคที่เป็นความลับ ไม่อาจเปิดเผยต่อชาวโลกได้ เช่นประกอบด้วยฐานข้อมูลใน งานวิจัย คะแนนแล็บของนักศึกษา ชื่อ URL, user name, password ที่เขียนลงไปใน source code เราก็อาจจะเลือก เป็น private ซึ่งอาจจะต้องมีค่าใช้จ่ายในการสมัครสมาชิกพิเศษ หรือไม่ก็ต้องเป็น academic account ในที่นี้ให้เลือก เป็น public
* ถ้าเราท าเครื่องหมาย☑ หน้าข้อความ Initialize this repository with a README เพื่อให้เราสามารถเขียนบรรยาย คร่าวๆ เกี่ยวกับ repository ได้
* เดี๋ยวก่อน…. ในขั้นตอนนี้ ยังไมต่ ้องท าเครื่องหมาย ☑เพราะเราจะทดลองสร้างโดยใช้command line tool
* เลือกว่าจะเพิ่ม .gitignore หรือ license file ด้วยหรือไม่ โดย .gitignore นี้จะบอก Git ว่าไม่ต้องสนใจที่จะติดตามไฟล์ ชนิดใดบ้าง โดย Git จะก าหนดชนิดของไฟล์ให้เบื้องต้น เช่น ถ้าเราเลือก .gitignore เป็น ภาษา C++ แล้ว Git จะเพิ่ม ชนิดของไฟล์ต่างๆ ที่เป็นผลจากการคอมไพล์ไว้ในรายการที่เพิกเฉย (เช่น ไฟล์ที่มีนามสกุล .exe) ซึ่งไฟล์เหล่านั้น มักจะ เกิดจากการคอมไพล์โปรแกรม ไม่ใช่ไฟล์ที่เราเป็นคนแก้ไข source code จึงไม่จ าเป็นที่จะต้องน าไปเก็บบน repository ให้สิ้นเปลืองพื้นที่ สามารถดูเทมเพลตของ .gitignore ได้จาก A collection of useful .gitignore templates[[4]](#footnote-4) **-** **ยังไม่ต้องเลือก**.**gitignore** **เช่นเดยี วกัน**

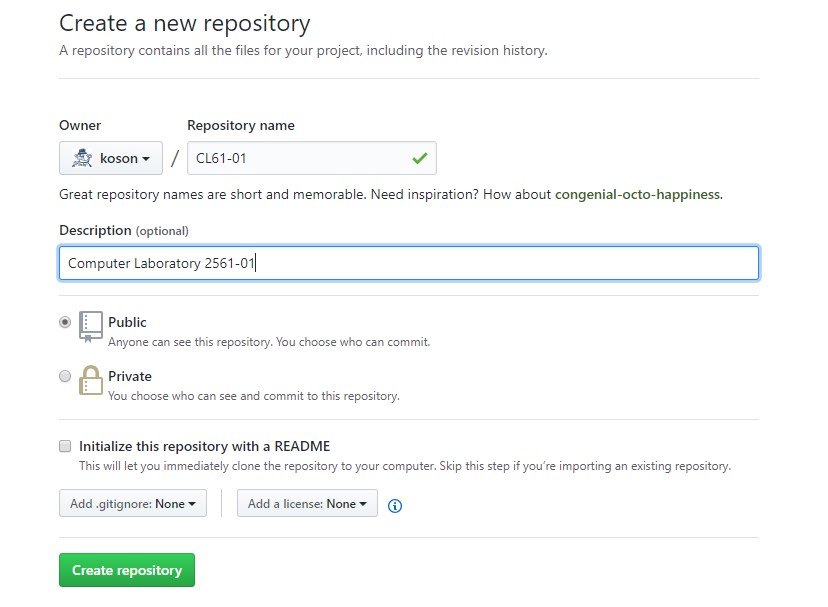
**รูปที่**

**1.**

**8**

การสร้าง

repository



|  |
| --- |
| ผลการทดลอง |
|  |

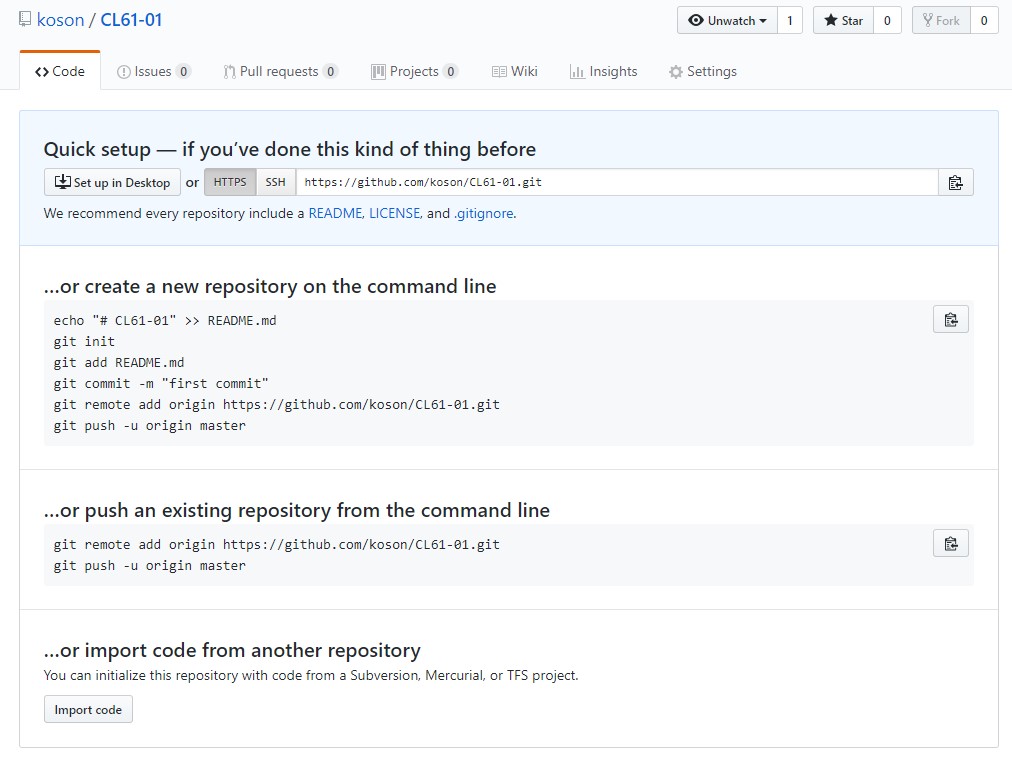
## - คลิกปุ่ม Create repository สีเขยี ว

Github

จะสร้าง

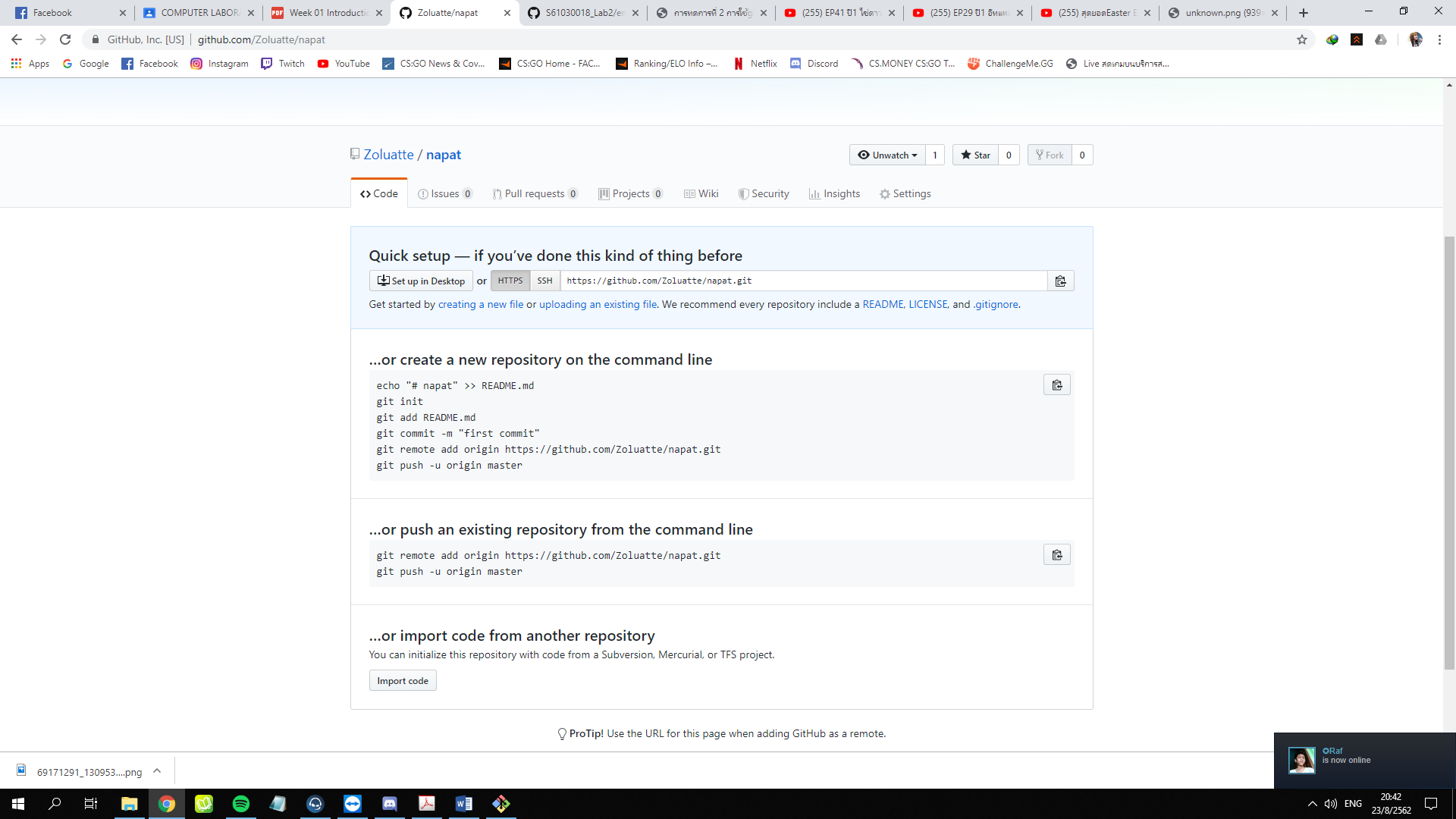
repository

ให้ตามต้องการ



**รูปที่ 1.9** repository ที่ได้จากการสรา้ งในข้อ 1.3

ผลการทดลอง



## หมายเหตุ ให้เปิดหน้าเพจนี้ค้างไว้ เพราะเราต้องมาดูผลการเปลี่ยนแปลงในภายหลัง

### **1.4 สร้าง** **git บนเครื่องคอมพิวเตอร์** **(Local)**

Repository ที่สร้างขึ้นในข้อ 1.3 นั้น เป็น repository ที่อยู่บน server ในขณะที่เราก าลังแก้ไข source code ซึ่ง มักจะเป็นการแก้ไขเล็ก ๆ น้อย ๆ การท างานของ git จะเน้นท างานที่ local เป็นหลัก ต่อเมื่อเราได้พัฒนา source code จนถึงจุด หนึ่ง ที่คิดว่าสามารถเผยแพร่ เพื่อการทดสอบหรือใช้งาน เราจึงส่งขึ้นไปเก็บบน server การท าส าเนาของ repository มาไว้บนเครื่อง (local) สามารถท าได้หลายวิธี ซึ่งเบื้องต้นนี้ เราจะศึกษาโดยการใช้งาน command line ซึ่งอาจจะพบกับความยุ่งยากบ้างในตอนแรกๆ แต่เมื่อใช้บ่อย ๆ จนช านาญจะพบว่ามีความยืดหยุ่นสูงกว่าการใช้

GUI Clients หรือเมื่อศึกษาจนเข้าใจแล้วหันไปใช้ GUI Clients ก็จะสามารถเข้าใจถึงการท างานของระบบ Git อย่างแท้จริง

**1.4.1 การ** **clone repository ด้วย** **command line (git bash)**

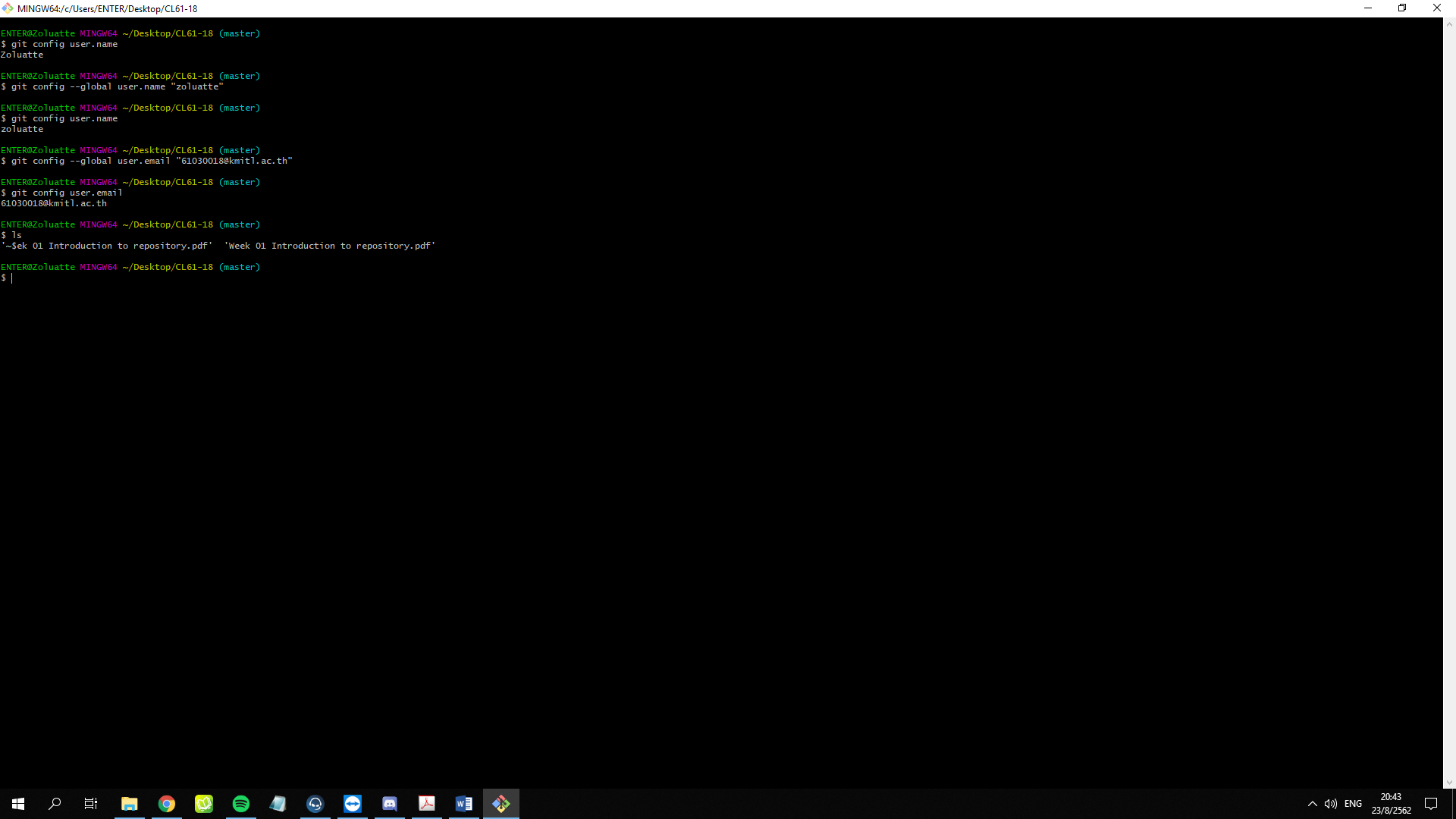
**1)** การเตรียมการเบื้องต้น

- ในหน้าต่าง git bash ให้พิมพ์ค าสงั่ list ดูรายการของไฟล์และโฟลเดอร์

**$ ls**

เราจะเห็นรายการไฟล์ถูกแสดงขึ้นมา

ผลการทดลอง



### - สร้างโฟลเดอร์ส าหรับเก็บงานในวชิ าการทดลอง (ในที่นี้ชื่อว่า CL2561 ย่อมาจาก Computer Laboratory 2561) โดย

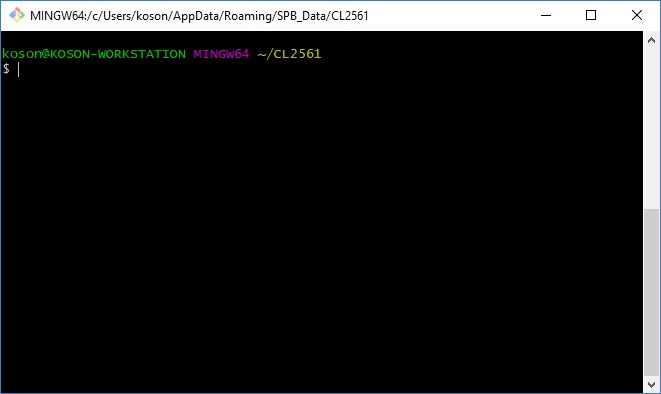
ใช้ค าสั่ง

**$ mkdir CL2561**

- ย้ายเข้าไปอยู่ในโฟลเดอร์ทสี่ ร้างขนึ้ โดยใช้ค าสั่ง

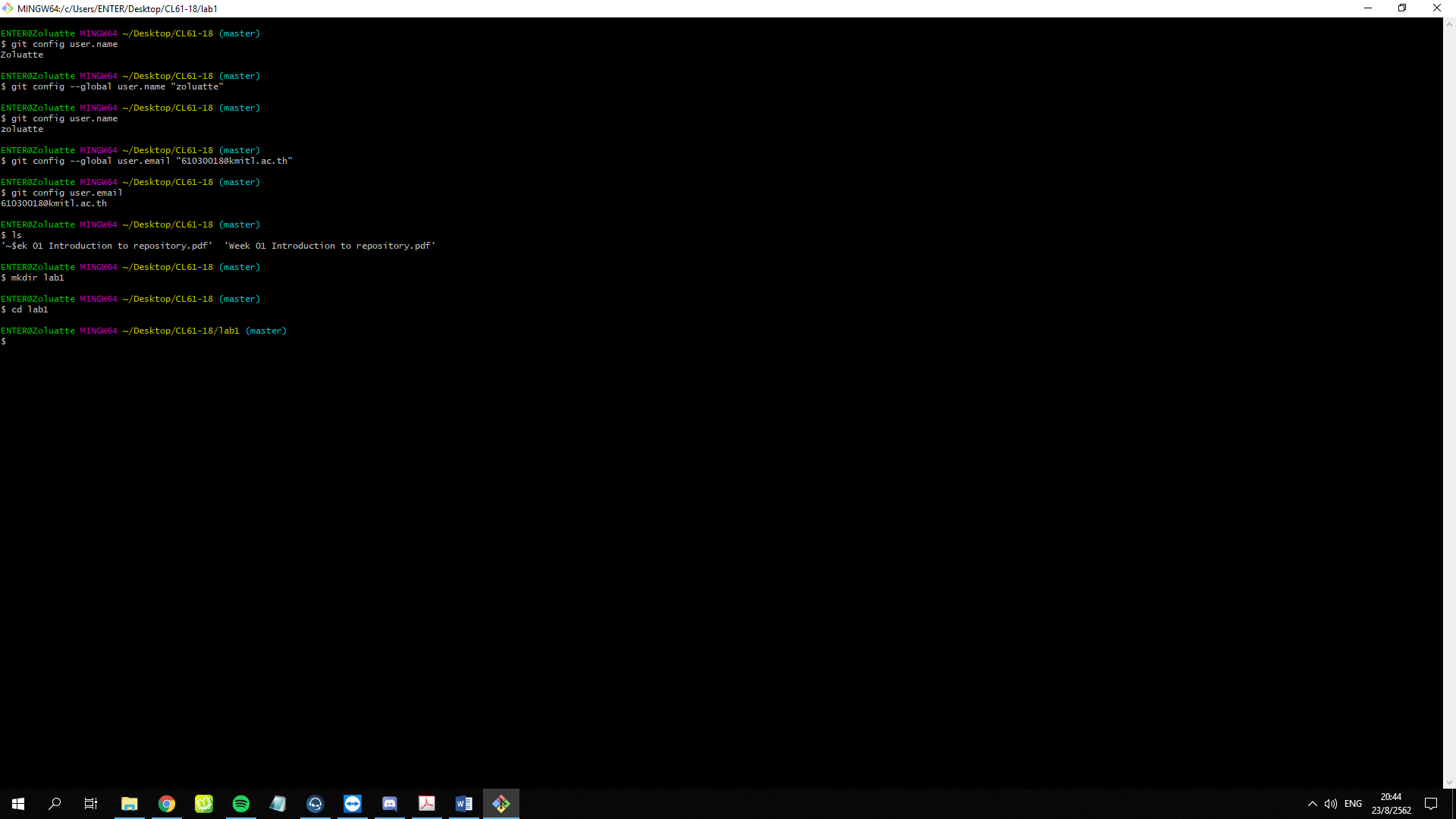
**$ cd CL2561**

สังเกตุได้จาก git bash จะแสดงชื่อของโฟลเดอร์ปัจจุบันเป็นดังรูปที่ 1.10



**รูปที่ 1.10** หน้าต่าง terminal ของ git bash เตรียมพร้อมส าหรับการ clone

ผลการทดลอง



## 2) การท าส าเนา repository มาไว้บนเครื่องโดยการ clone

- ท าส าเนา repository มาไว้บนเครื่องโดยใช้ค าสั่งที่มีรปู แบบดังต่อไปนี้

**$ git clone https://github.com/[YOUR USERNAME]/[YOUR REPOSITORY NAME]**

[YOUR USERNAME] คือ username ของเราบน github

### [YOUR REPOSITORY NAME]คือชื่อ repository ของเราที่สร้างในข้อ 1.3

ถ้าจ าไมไ่ ด้ ก็ไม่เป็นไร ให้เข้าไปที่ repository ที่เพิ่งสร้างบน Github (ดูรูปที่ 1.9) จะเห็นว่ามี URL ของ repository ส าหรับการ โคลน ดังรูปที่ 1.11 ให้เรากดปมุ่ copy ที่อยู่ด้านขวามือของ url

**รูปที่**

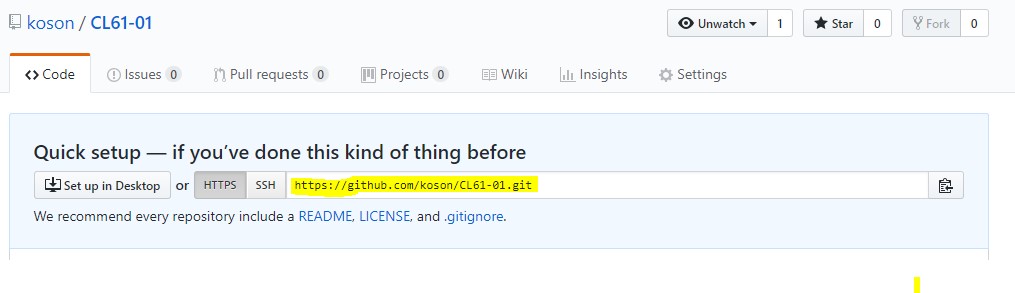
**1.**

**11**

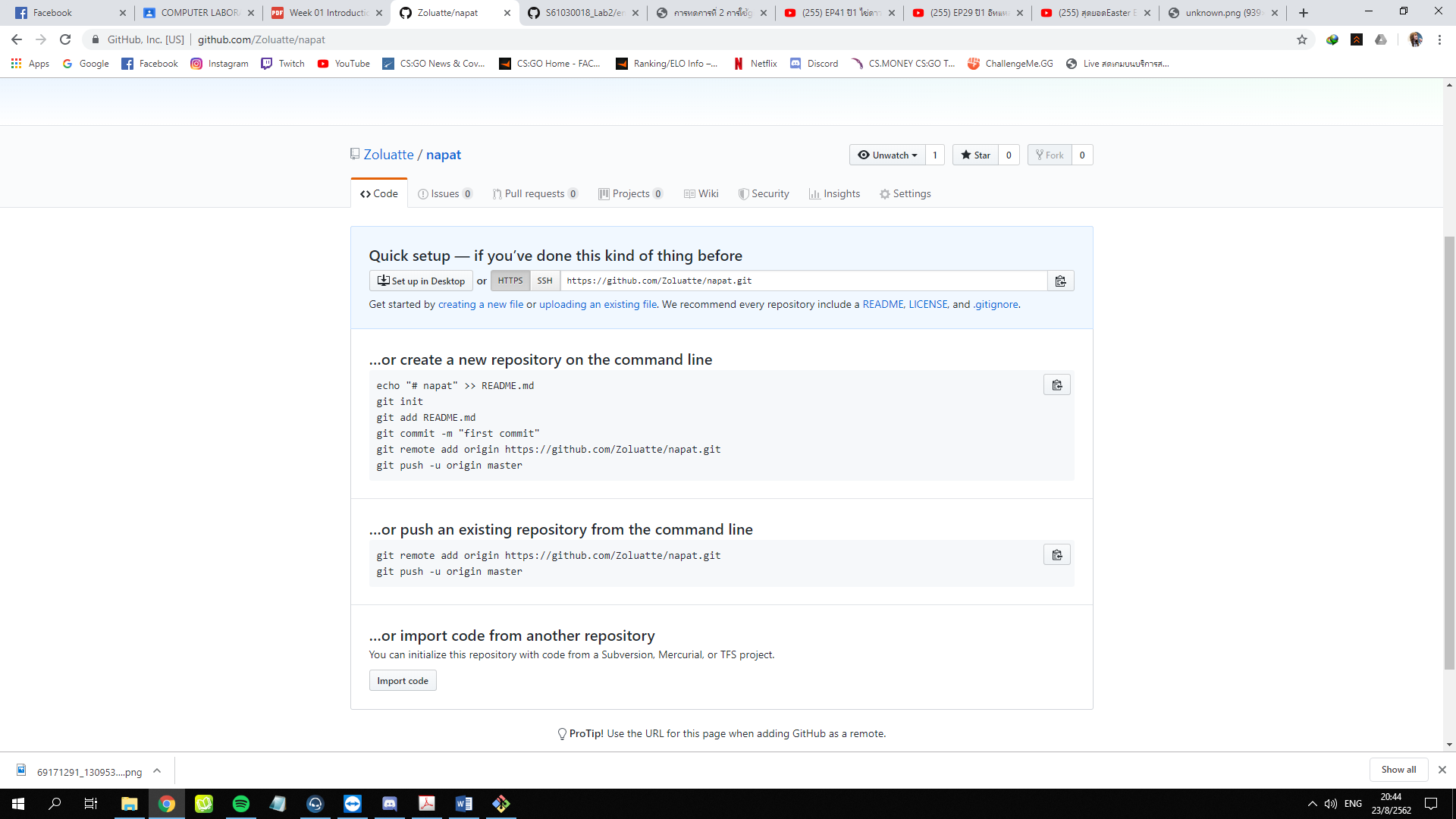
URL

ส าหรับการ

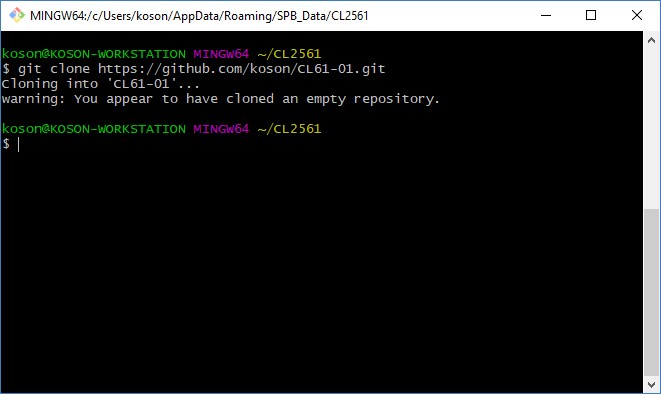
clone repository



ผลการทดลอง



* ใน git bash ให้พิมพ์ค าสั่ง git clone ตามด้วย URL ที่คัดลอกมา - เมื่อท าการ clone เรียบร้อย จะได้ผลดังรูปที่ 1.12



**รูปที่ 1.12** ผลการ clone repository

|  |
| --- |
| ผลการทดลอง |
|  |

* เรียกดรู ายการโฟลเดอร์ (ด้วยค าสงั่ ls) และเปลี่ยนโฟลเดอร์ (ด้วยคา สั่ง change directory :cd)

|  |
| --- |
| ผลการทดลอง |
|  |

ตอนแรกจะพบว่ามโี ฟลเดอร์ชื่อ CL61-01 ซึ่งถูก clone มาจาก server จึงย้ายเข้าไปในโฟลเดอร์นั้น แล้วจึงสั่ง ls เพื่อดูรายการ ไฟล์ พบว่า repository ของเราจะยังว่างเปล่า ดังรูปที่ 1.13



### **รูปที่ 1.13** ไฟล์ที่ถูก clone มาจาก repository

|  |
| --- |
| ผลการทดลอง |
|  |

* ให้พิมค าสั่งต่อไปนี้ ครั้งละบรรทดั (พิมพ์ให้ครบบรรทัดแล้วเคาะ enter)

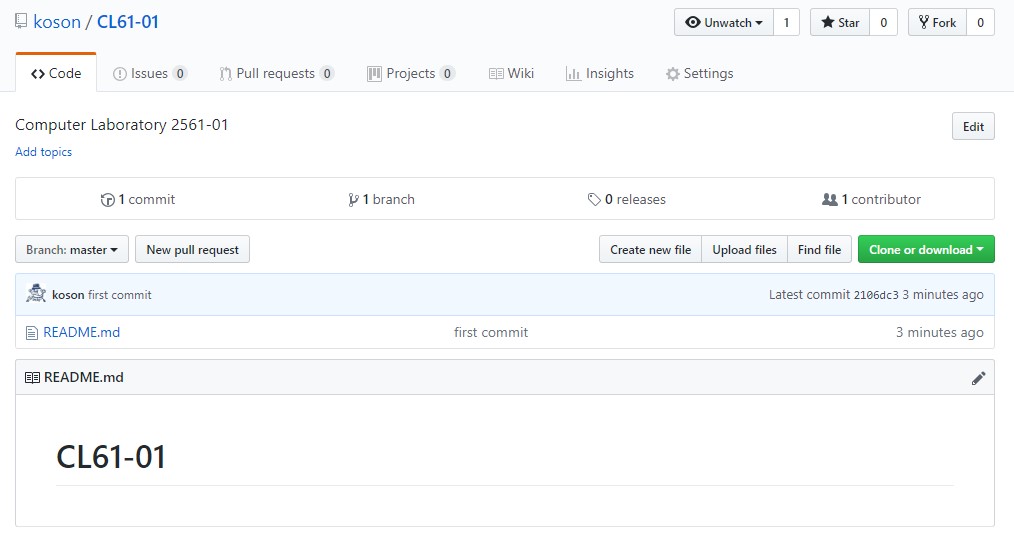


**รูปที่ 1.14** การเพิ่มไฟล์ README.md ให้กับ repository

|  |
| --- |
| ผลการทดลอง |
|  |

จากรูปที่ 1.14 จะได้ผลการท างานดังรูปที่ 1.15 ซึ่งจะเห็นว่า บางค าสั่งอาจจะมี error เกิดขึ้น เนื่องจากมี repository อยู่บน server แล้ว แต่ก็ให้ท าให้ครบทุกขั้นตอนไปก่อน เพราะ ในกรณีนี้ error เหล่านั้นไม่ส่งผลกระทบร้ายแรงต่อการท างาน

* ให้กลับไปที่ browser และกด refresh 1 ครั้ง จะเห็นว่าหน้า repository ที่เราเพิ่งสร้าง จะเปลี่ยนไป



**รูปที่ 1.15** หน้าเพจ repository ที่เปลี่ยนไปหลังจากเพิ่มไฟล์ README.md

|  |
| --- |
| ผลการทดลอง |
|  |

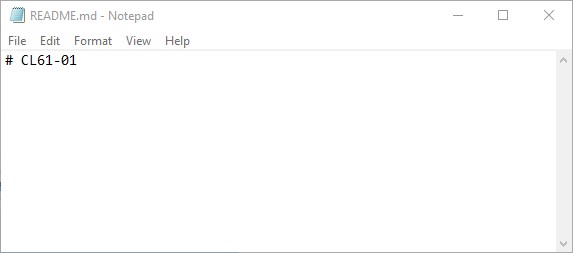
**1.5 การแก้ไขงานและบันทึกการเปลี่ยนแปลงบน local computer**  ถึงตอนนี้ เนื้อหาในไฟล์ README.md บน server และ local computer จะเหมือนกันทุกประการ เนื่องจากเป็นการ

clone มาและยังไม่ได้ท าการแก้ไขใดๆ อีกทั้งเรามั่นใจว่าไม่มีผู้ใช้คนอื่นๆ ก าลังแก้ไขงานของเราบน server (ซึ่งการแก้ไขงาน ร่วมกันบน server จะอยู่ในการทดลองถัดไป) เราสามารถแก้ไขและท า revision ของเอกสารได้ตามต้องการ โดยการเปลี่ยนแปลง ต่างๆ จะเกิดขึ้นบนเครื่อง local computer เท่านั้น **1.5.1 ทดลองแก้ไขไฟล์ README.md**  โดยส่วนใหญ่ ในการเขียนโปรแกรม มักจะกระท าบนโปรแกรม Integrated development environment หรือเรียก สั้นๆ ว่า IDE[[5]](#footnote-5) แต่ในการทดลองนี้ จะใช้โปรแกรมแกไ้ ขเอกสารอย่างงา่ ยๆ นั่นคือโปรแกรม Notepad.exe

* ให้พิมพ์ค าสั่งต่อไปนลี้ งใน git bash

**$ notepad README.md**

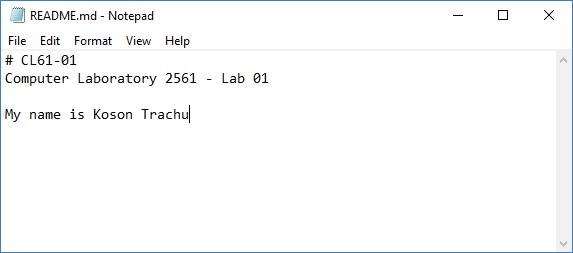
ระบบจะเปิด text editor ที่มากบั ระบบปฏบิ ัติการ Windows ดังรูปที่ 16



**รูปที่ 1.16** การใช้โปรแกรม notepad.exe แก้ไขไฟล์ README.md

|  |
| --- |
| ผลการทดลอง |
|  |

### - แก้ไขไฟล์ README.md ใน notepad โดยเพิ่มข้อความลงไปดังตวั อย่าง (ให้นักศึกษาใส่ชื่อตนเอง)



**รูปที่ 1.17** แก้ไขไฟล์ README.md โดยเพิ่มบรรทดั ต่อท้ายเข้าไป

|  |
| --- |
| ผลการทดลอง |
|  |

* บันทึกและปิดโปรแกรม notepad.exe
* ตรวจสอบการเปลี่ยนแปลงใน git bash โดยพมิ พ์ค าสั่ง git status แล้วสังเกตผุ ลทไี่ ด้จากการรันค าสงั่

## $ git status



**รูปที่ 1.18** การตรวจสอบสถานะของ git

จะพบว่า git ได้ท าการติดตามการเปลยี่ นแปลง (tracking) ของไฟล์ต่างๆ ใน repository ของเราอยู่เสมอ ถึงแม้จะ เป็น local computer ก็ตาม (ไมน่ ับไฟล์ใน .gitignore)

|  |
| --- |
| ผลการทดลอง |
|  |

**1.5.2 บันทึกการเปลี่ยนแปลงบน** **local computer** ถึงตรงนี้ ถ้าเราต้องการจะแกไ้ ขตอ่ ก็สามารถท าได้ แต่การเปลี่ยนแปลงต่างๆ จะไม่สามารถถูกติดตามโดย git ถ้าต้องการ ให้ git บันทึก (หรือนับ) การเปลี่ยนแปลงเป็นรุ่นหนึ่งๆ ของ source code สามารถท าได้โดยการ commit การเปลี่ยนแปลงลงใน local repository ซึ่งการใช้งานเบื้องต้นจะมี 2 ค าสั่งคือ git add และ git commit

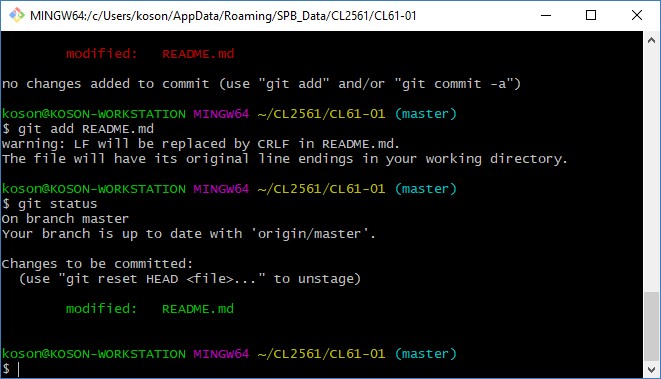
- เพิ่มไฟล์ทเี่ ปลี่ยนแปลง เข้าสรู่ ายการ commit โดยใช้ค าสั่งต่อไปนี้

## $ git add README.md

ตรวจสอบการเปลี่ยนแปลงใน git bash โดยพิมพ์คา สั่ง git status แล้วสังเกตผุ ลทไี่ ด้จากการรันค าสงั่

|  |
| --- |
| ผลการทดลอง |
|  |

## $ git status



**รูปที่ 1.19** การตรวจสอบสถานะของ git /ผลจากการท าคา สั่ง git add

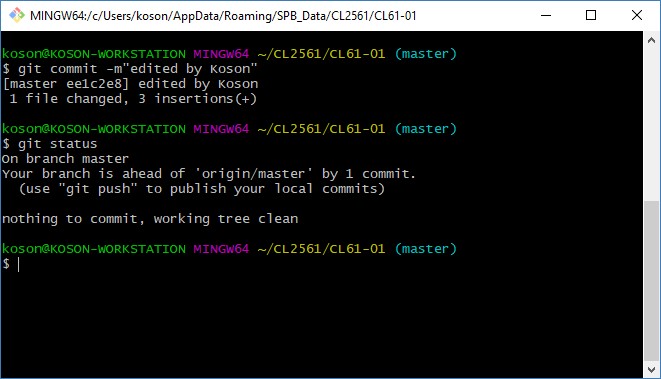
**หมายเหตุ** หากมีการแก้ไขหลายๆ ไฟล์ เราอาจใช้ค าสั่ง git add --allแทนการใช้ชื่อไฟล์ได้ - Commit ไฟล์ที่เปลี่ยนแปลง เข้าสู่ repository โดยใช้ค าสั่งต่อไปนี้

**$ git commit -m “Edited by Koson”**

ตามด้วยการตรวจสอบสถานะของ repository

## $ git status

จะไดผ้ ลดังนี้



**รูปที่ 1.20** ผลจากการท า git commit

|  |
| --- |
| ผลการทดลอง |
|  |

**หมายเหตุ** รูปแบบของการ commitประกอบด้วย ค าสั่ง git commit -m “THIS IS A COMMIT MESSAGE”โดยที่

commit messageควรเป็นข้อความที่สื่อความหมาย มีความยาวไม่มากนัก แต่ไม่สั้นจนเกินไป **ควรหลีกเลี่ยงค าทีไม่สื่อ ความหมาย เช่น** “**1”, “2”** **หรือ** “**a”** **ถึงแม้ว่า git** **จะอนุญาตให้ใช้ก็ตาม** เนื่องจากเมื่อพัฒนาไปหลายๆ รุ่น จะไม่สามารถท า ความเข้าใจเหตุผลที่แก้ไข source codeนั้น ๆ ได้ และในการเปลี่ยนแปลงแต่ละครั้ง gitจะน า commit messageนี้ไปใช้ร่วมกับ การเปลี่ยนแปลงเสมอ

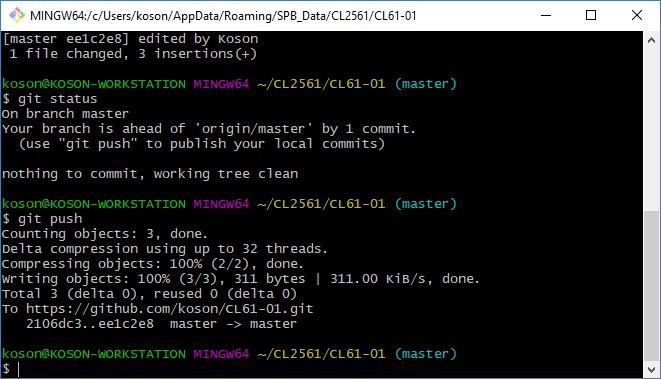
**1.6 การซิงค์การเปลี่ยนแปลงระหว่าง** **local computer และ** **server**  หลังจากที่เราได้ท าการ clone repository มาที่ local computer แล้ว การแก้ไขงานทั้งหมด สามารถท าได้บน local computer ได้โดยไม่ต้องเชื่อมต่อกับ server แต่ในบางครั้งที่มีการท างานร่วมกันเป็นทีม จะต้องปรับปรุง source code ให้เป็น ปัจจุบันอยู่เสมอ 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 เรามักจะใช้ค าสั่ง 3 ค าสั่งควบคู่กันคือ (1) git add --all, (2) git commit -m “Commit message” และ (3) git push แต่ในการทดลองที่ผ่านมา เราท าใน (1) และ (2) ไปแล้ว ดังนั้น ให้พิมพ์ค าสั่งต่อไปนี้ เพื่อ push repository ขึ้น server

## $ git push

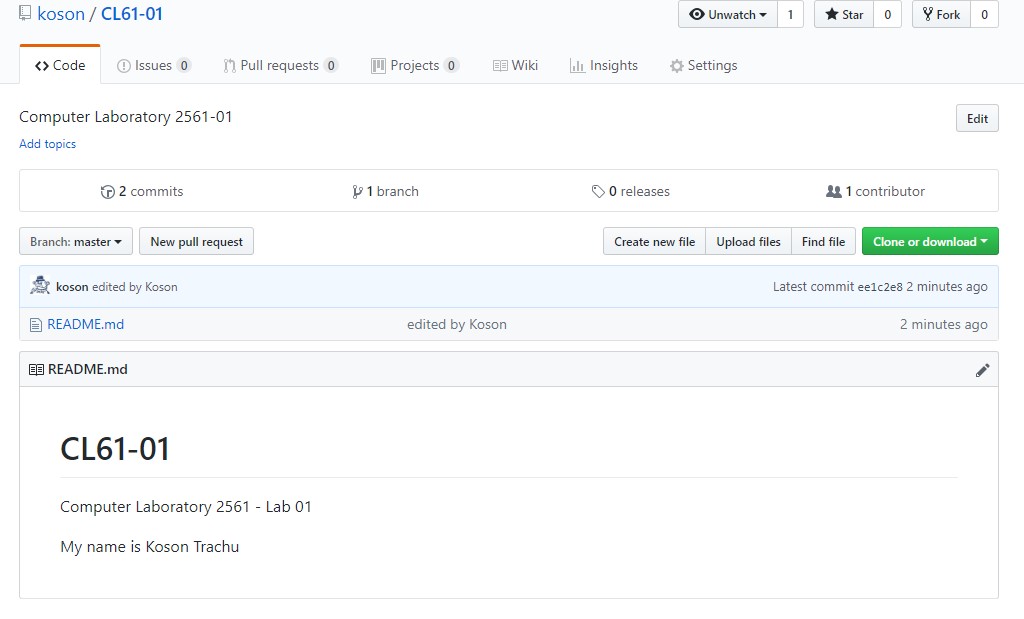
จะไดผ้ ลลัพธ์คล้ายตัวอย่างในรูปที่ 20



**รูปที่ 1.21** ผลจากการท าคา สั่ง git push

|  |
| --- |
| ผลการทดลอง |
|  |

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

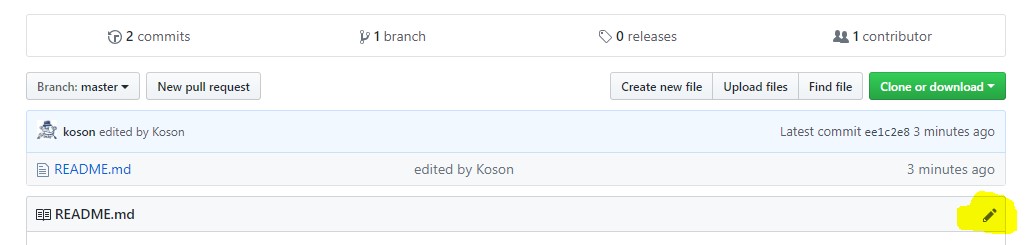


**รูปที่ 1.22** การเปลี่ยนแปลงที่เกิดขึ้นบน server

|  |
| --- |
| ผลการทดลอง |
|  |

**1.6.2 การ pull มาจาก server**

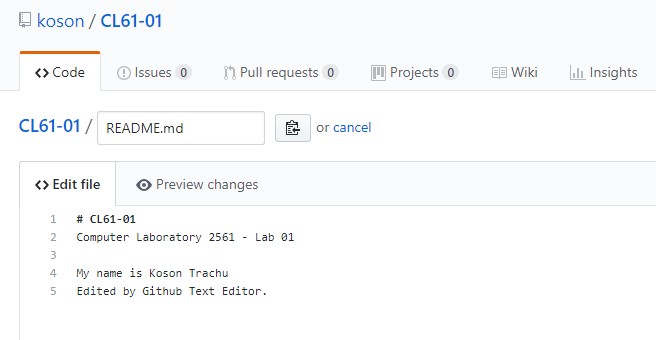
* การเปลี่ยนแปลงใดๆ ที่เกิดขึ้นบน local computer จะถูกส่งขึ้นมาเก็บด้วยค าสั่ง git push และถ้ามีการแก้ไขไฟล์ใด ๆ เกิดขึ้นบน server เราก็สามารถที่จะดึงกลับไปท างานที่ local computer ได้เช่นกัน
* ให้แก้ไขไฟล์ README.md โดยการคลิกที่ชื่อไฟล์ และปมุ่ ปากกาบริเวณด้านขวามือ



**รูปที่ 1.23** เข้าสู่โหมดการแกไ้ ขไฟล์ด้วย Github Text Editor

|  |
| --- |
| ผลการทดลอง |
|  |

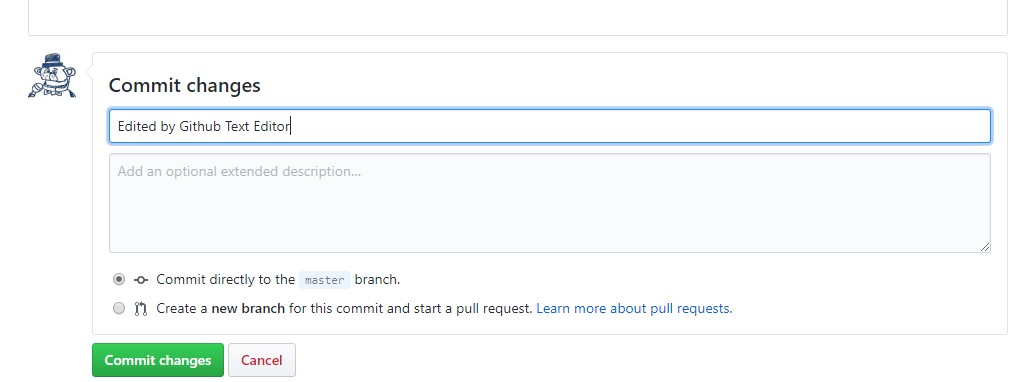
- เพิ่มข้อความที่บรรทัดล่างสุดดังตัวอย่าง



**รูปที่ 1.24** เพิ่มข้อความบางอย่างใน Github Text Editor

|  |
| --- |
| ผลการทดลอง |
|  |

* เพิ่มข้อความในช่อง Commit changes และกดปุ่ม Commit changes สีเขียว



**รูปที่ 1.25** เพิ่มข้อความ Commit changes

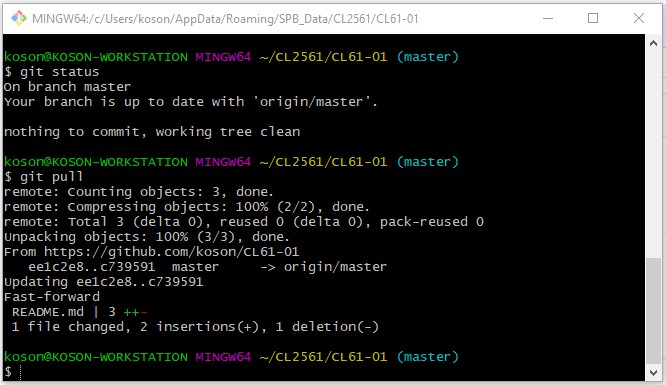
|  |
| --- |
| ผลการทดลอง |
|  |

* กลับมาที่ git bash พิมพ์คา สั่ง git status สังเกตุผลการท างาน

## $ git status

- ที่ git bash พิมพ์ค าสั่ง git pull

## $ git pull

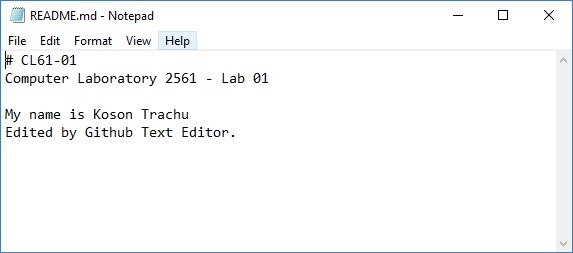


**รูปที่ 1.26** การใช้ค าสั่ง git pull

|  |
| --- |
| ผลการทดลอง |
|  |

- ดูการเปลี่ยนแปลงในไฟล์ README.md

## $ notepad.exe README.md

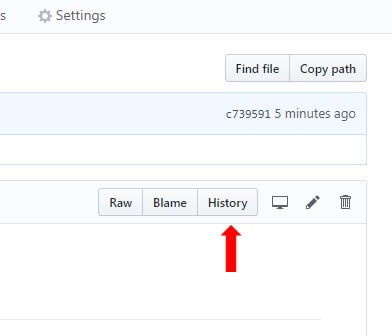
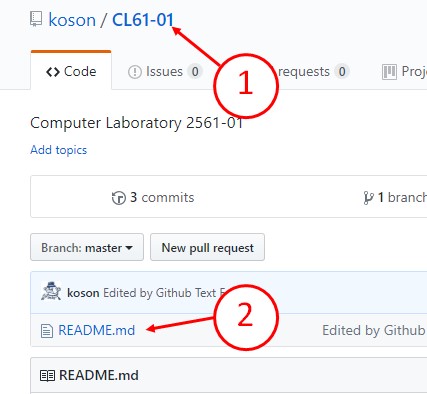


**รูปที่** **1.27** การเปลี่ยนแปลงในไฟล์เอกสาร README.md

|  |
| --- |
| ผลการทดลอง |
|  |

**1.7 การตรวจสอบประวัติการเปลี่ยนแปลงของไฟล์**

### - กลับไปที่ web browser (1) คลิกที่ชื่อ repository, (2) คลิกที่ชื่อ repository

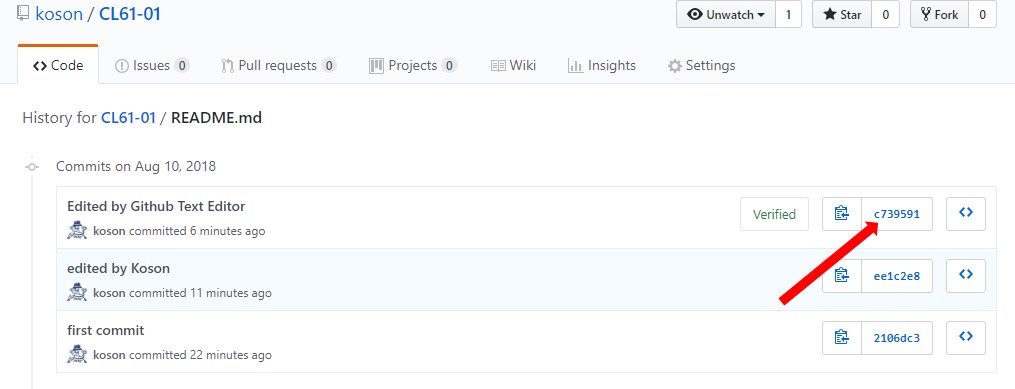


(ก) เลือกชื่อ repository, ชื่อไฟล์ (ข) คลิกที่ปุ่ม History

**รูปที่ 1.28** การเข้าถึงประวัติของไฟล์

### เมื่อคลิกดูประวตั ิไฟล์ จะพบว่า ไม่ว่าเราจะแก้ไขไฟล์ที่ไหน แต่ Git จะติดตามและบันทึกการเปลี่ยนแปลงทุกครั้งที่เรา ท าการ commit

|  |
| --- |
| ผลการทดลอง |
|  |



**รูปที่ 1.29** รายการประวัติการแกไ้ ขไฟล์

|  |
| --- |
| ผลการทดลอง |
| 0 |

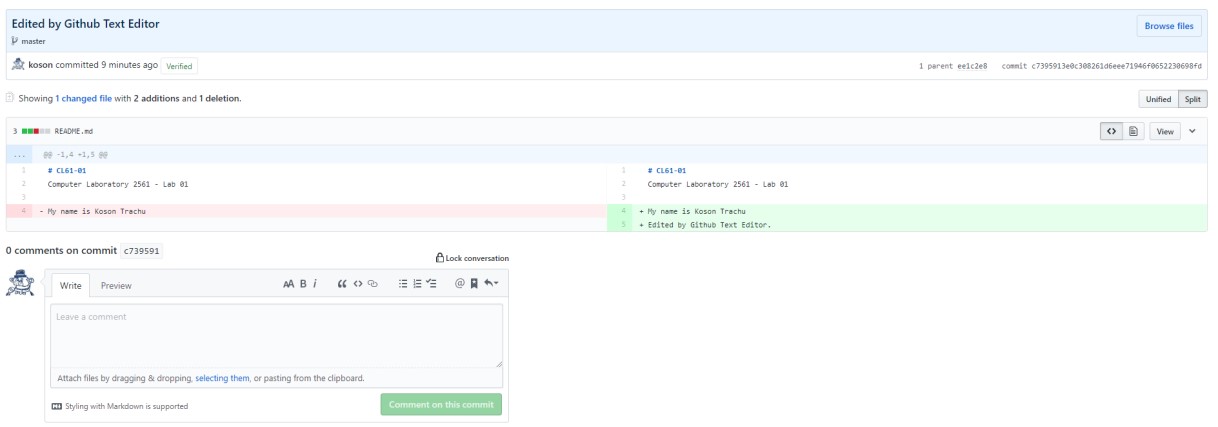
ให้คลิกปุ่มที่มีเลขฐาน 16 ก ากับ (เป็นชื่อรหัสก ากับการแก้ไข ที่ทีมพัฒนาจะใช้อ้างอิงถึง) ตามลูกศรสีแดงในรูปทีี่ 29 เรา จะเห็นประวัติการแก้ไขไฟล์ ดังรูปที่ 30

**รูปที่**

**1.**

**30**

ประวัติการแก้ไขไฟล์



**แบบฝึกหัด**

1. ให้นักศึกษาทดลองเพิ่มไฟล์ชื่อ student.txt ลงใน repository แล้วเพิ่มรายชื่อเพื่อนในห้อง โดยเพิ่มบน notepad จ านวนครึ่งหนึ่ง และท าบน github text editor จนครบ โดยใหเ้ ขียน commit message ด้วยว่าเพิ่มจากที่ใด
2. ให้นักศึกษาทดลองแก้ไขไฟล์ README.md ตามตารางต่อไปนี้ แล้วท ารายงานประวตั ิไฟล์มาส่ง

|  |  |  |
| --- | --- | --- |
| ล าดับที่ | สถานที่แก้ไข | สิ่งที่กระท า |
| 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** การท าแต่ละขั้น ให้ local และ server ซิงค์กันเสมอ (ตอ้ ง push, pull, commit, add )

**ค าถาม**

1. จากภาพที่ 29 ถ้าหากนักศึกษาคลิกตามปมุ่ ที่มีเลขฐานสิบหกก ากบั อยู่ ทุกปุ่ม จะไดผ้ ลอย่างไรบา้ ง ให้อธิบายสิ่งที่พบ เห็น
2. ให้บอกประโยชน์ของ repository ตามที่นักศึกษาเข้าใจ
3. ให้บอกแนวทางการน า repository ไปใช้ในการเรียนหรือชีวิตประจา วันของนักศึกษา

1. **"Git · GitHub." Accessed August 10, 2017.** [**https://github.com/git.**](https://github.com/git) [↑](#footnote-ref-1)
2. "Setting your username in Git - User Documentation - GitHub Help." Accessed August 10, 2017. [https://help.github.com/articles/setting-your-username-in-git/.](https://help.github.com/articles/setting-your-username-in-git/) [↑](#footnote-ref-2)
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/) [↑](#footnote-ref-3)
4. "A collection of useful .gitignore templates ...." Accessed August 11, 2017. [https://github.com/github/gitignore.](https://github.com/github/gitignore) [↑](#footnote-ref-4)
5. "Integrated development environment - Wikipedia." Accessed August 11, 2017. [https://en.wikipedia.org/wiki/Integrated\_development\_environment.](https://en.wikipedia.org/wiki/Integrated_development_environment) [↑](#footnote-ref-5)