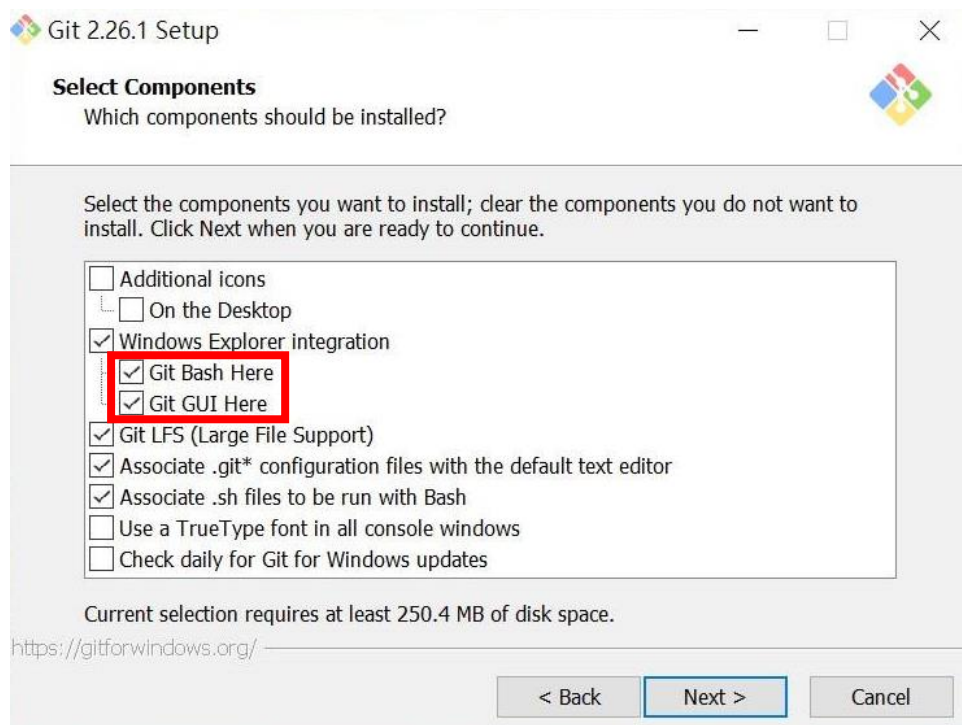


下載軟體(先載 scm，再載 fork)：

<https://git-scm.com/>

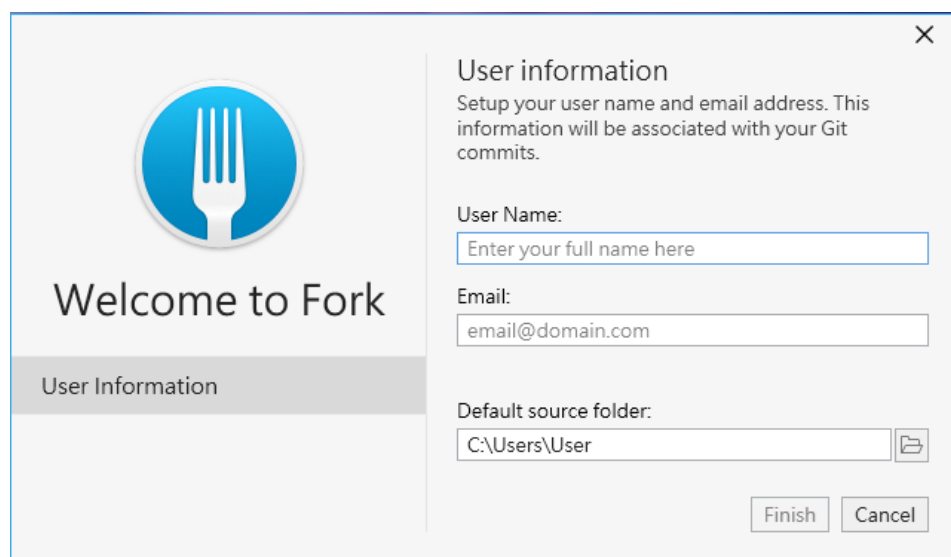
<https://git-fork.com/> (版本為 1.48.0.0)



scm 安裝頁面：

要勾 Git Bash Here、Git GUI Here

其餘不要更動



fork UserName: Github 使用者名稱

fork email 輸入：Github 信箱帳號

fork GUI 操作:

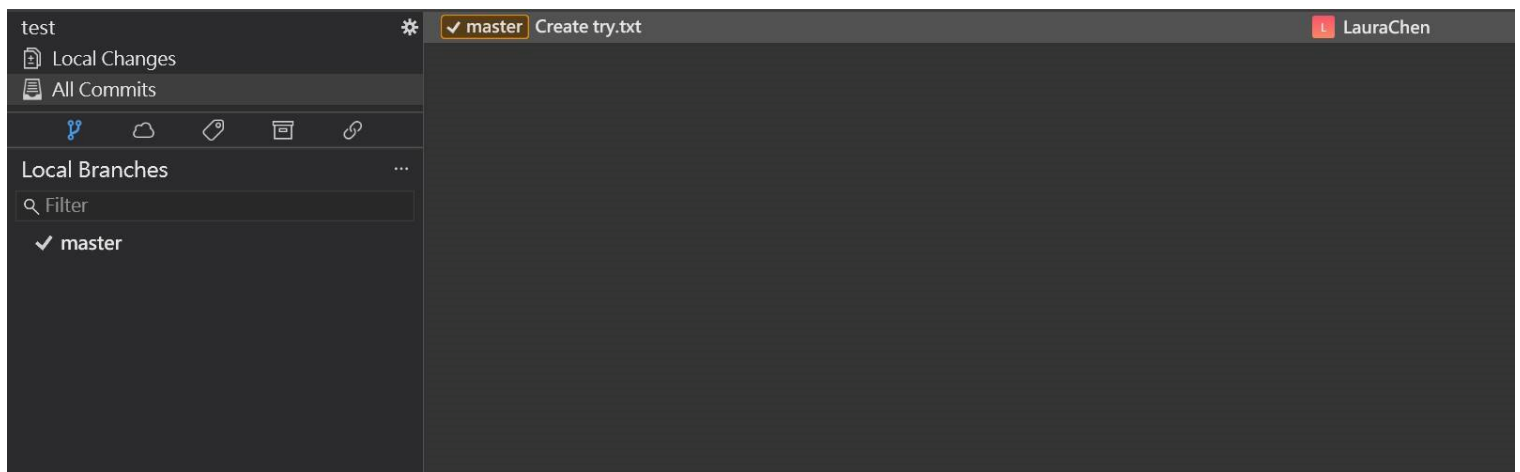
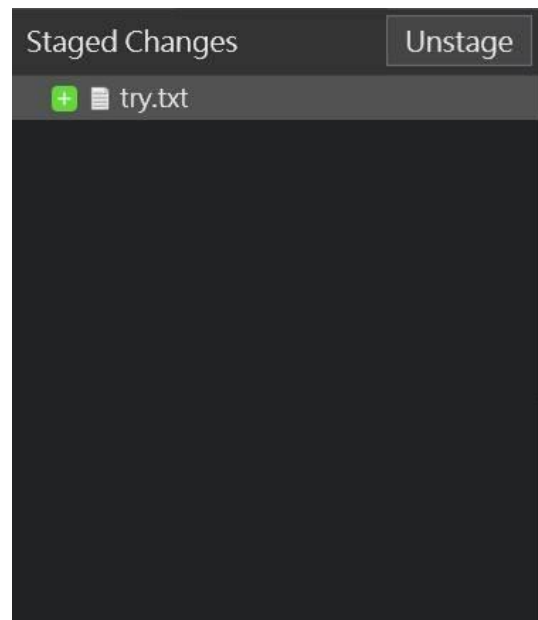
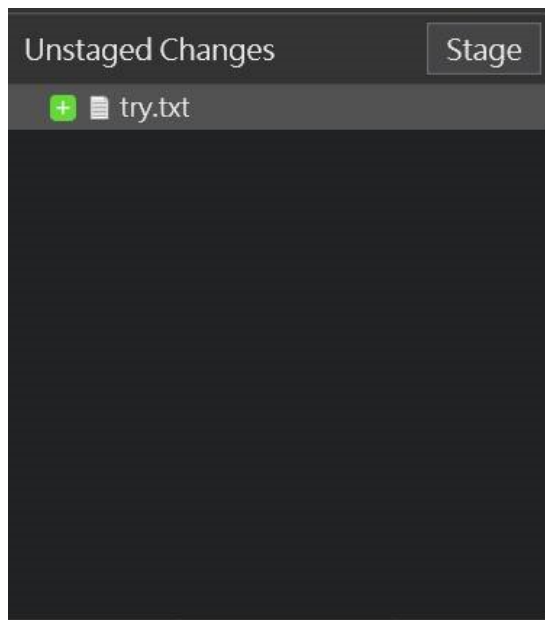
桌面建立資料夾(資料夾名為 **test**) -> 在資料夾中建立 **txt** 檔(檔名為 **try**)

補充：

建立此原因是要記錄每個檔案的詳細資料,多個檔案紀錄時也不會有衝突的問題

建立檔案節點的流程(以資料夾 **test**，檔案為 **try** 為例)：

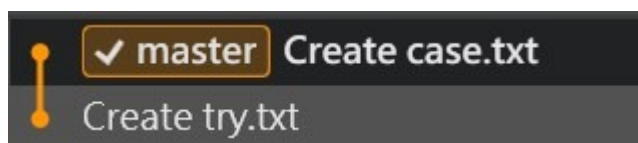
file -> init new repository -> 選擇要建立 github 的資料夾(test 資料夾) -> 選 Local Changes -> 選要快照的檔案(try 檔案)再 Unstaged Changes 處按兩下(或是選取後按 stage，之後就會新增在 Staged Changes 處) -> 在 Enter commit subject 處輸入 Create try.txt，點選 Commit 1 File -> 在 All Commits 裡就會有檔案的紀錄且建立節點



快照：其意思同拍照般可以凍結當時的一瞬間，對著資料庫拍照，就是對資料庫的某一個時間點，凍結記錄內容並完整的紀錄下來

回復前一步：選擇前一個節點按兩下 **Checkout commit** -> 在資料夾的地方就會將最新的檔案隱藏

預設情況：當有兩個以上的節點(包含兩個)時，以 **case.txt** 和 **try.txt** 為例。
以圖一來看最新的是 **case.txt**，要把 **case.txt** 隱藏起來。(圖 2 為未更動之前的情況)



→ 圖 1

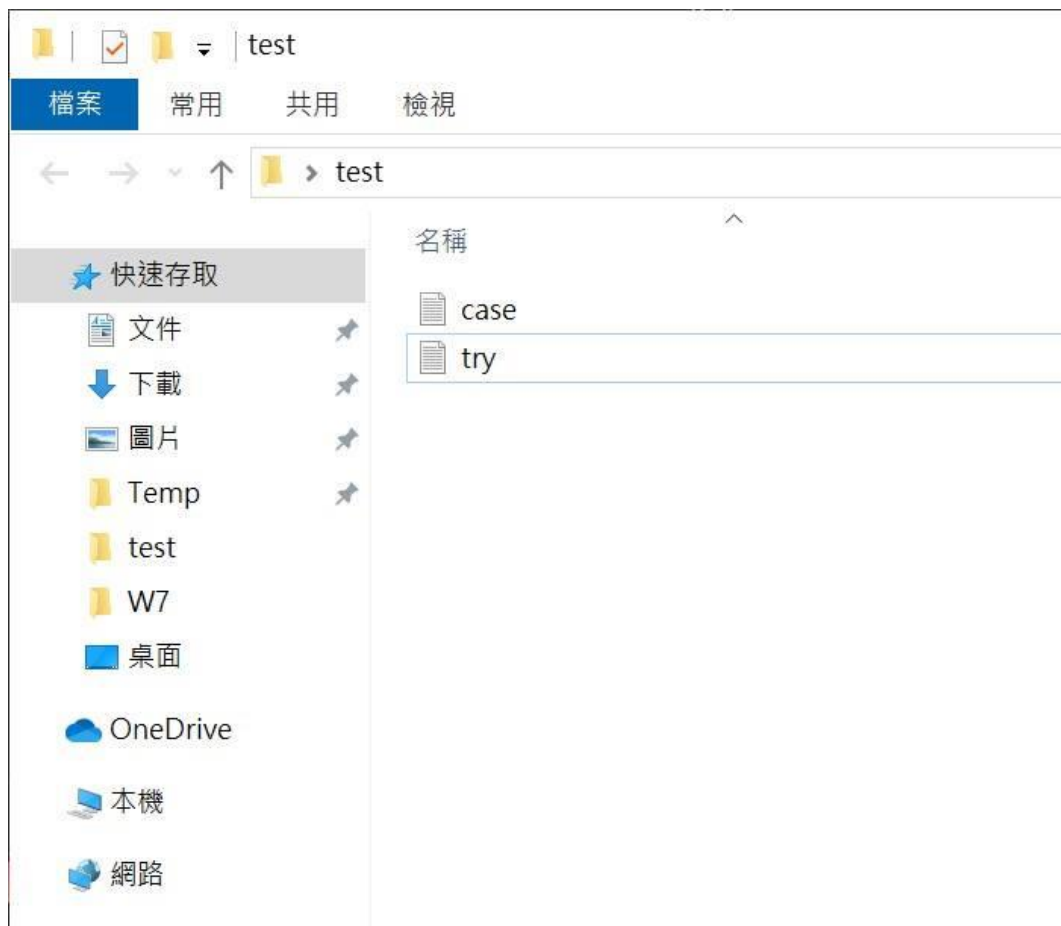
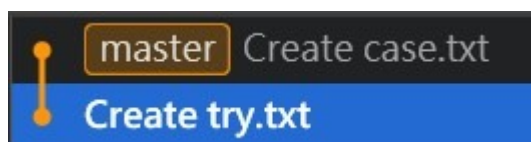


圖 2



→ 圖 3

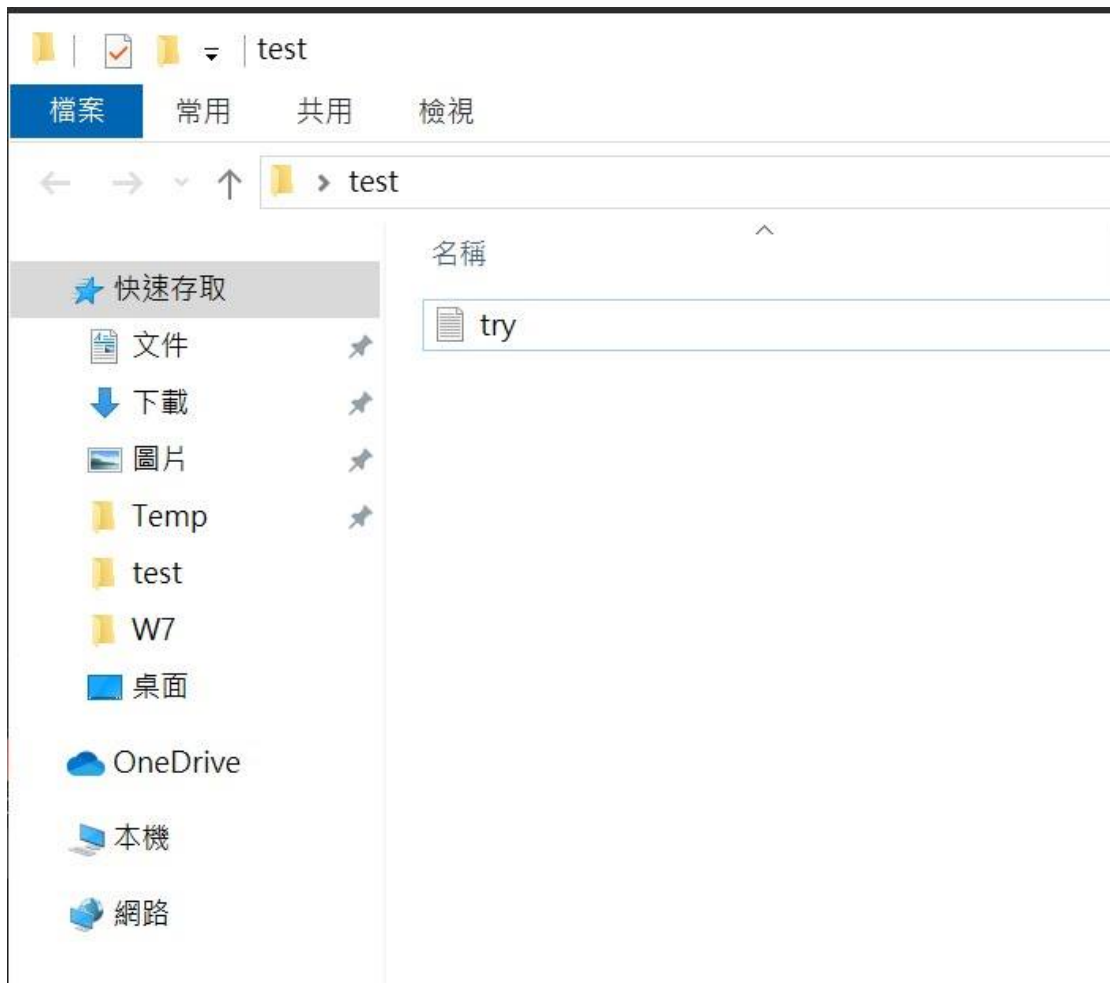


圖 4

隱藏的步驟：

All Commits 中在 create try.txt 點兩下(或是在 create try.txt 按右鍵選 checkout commit，如圖 6)就會跳出圖 5，按下 checkout commit 後，在資料夾中圖 4 就會將 case.txt 隱藏，在 All Commits 的情況為圖 3。

若要返回在 All Commits 中 Create case.txt 點兩下即可回復如圖 2。

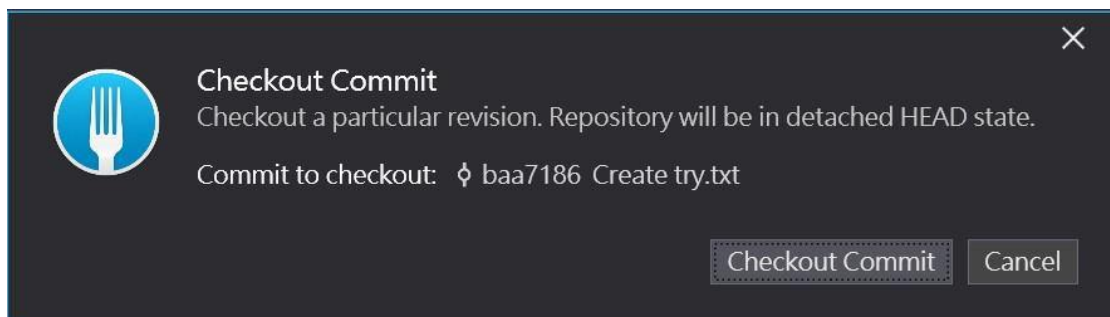


圖 5

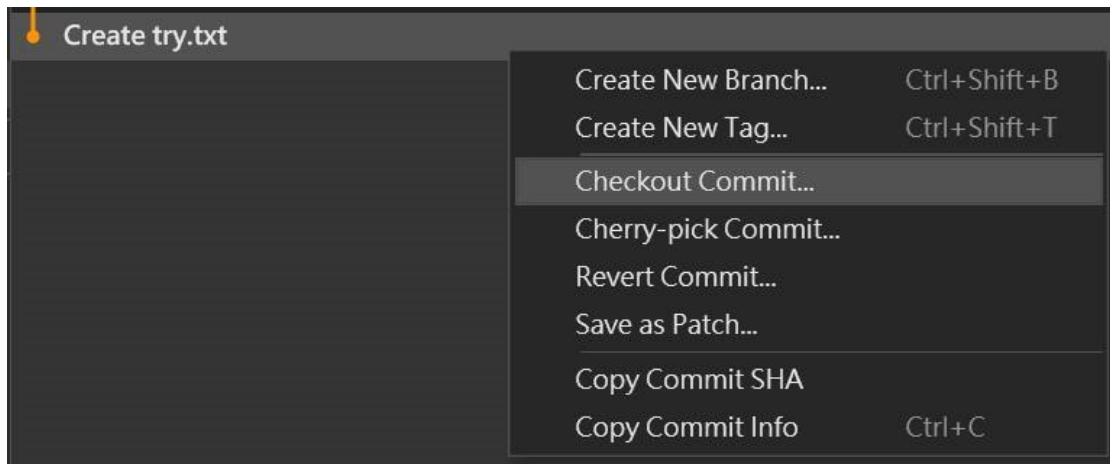
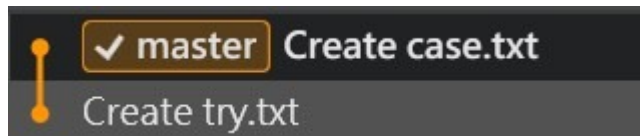


圖 6

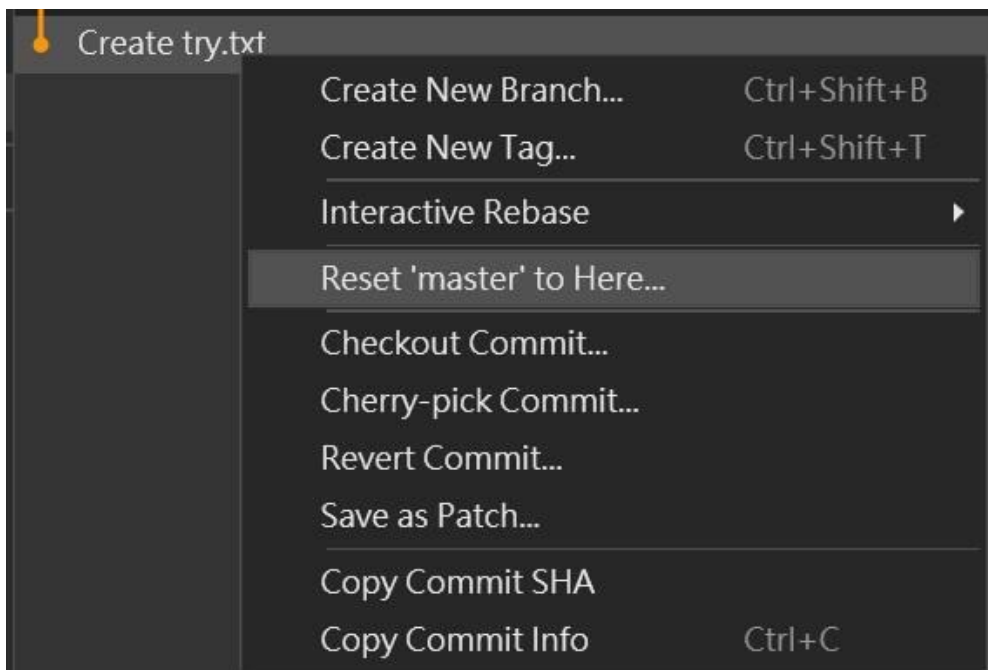
刪除節點紀錄(reset)：選擇要回復的地方按右鍵 Reset 'master' to here -> 選擇要刪除的選項(避免不要，除非有把握)

預設情況：刪除 case.txt，未操作前為圖 7



→ 圖 7

在 Create try.txt 按右鍵選 Reset 'master' to here(如圖 8)，就會彈出一個視窗(如圖 9，選 Mixed 即可)，按下 reset 後就成功刪除 case.txt 的節點(如圖 10)。Case.txt 就會返回到 Unstaged Changes 處(如圖 11)



→ 圖 8

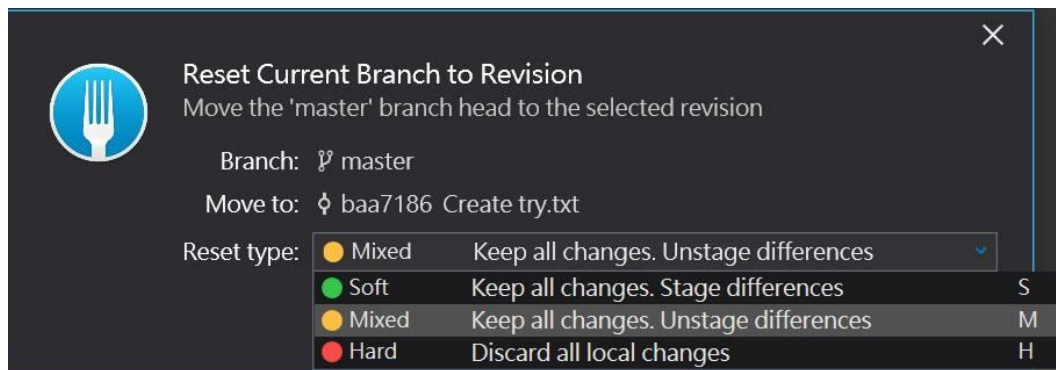
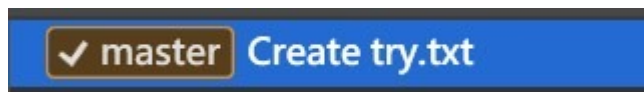
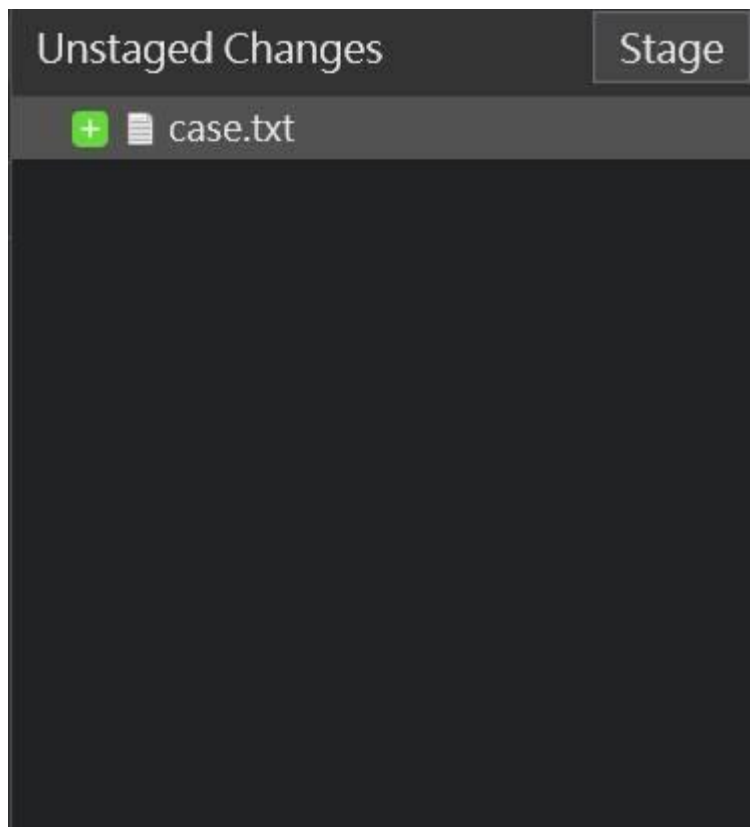


圖 9



→ 圖 10



→ 圖 11

補充：

soft:紀錄到 Change staged stage 處

Mixed:紀錄到 Change Unstaged stage 處

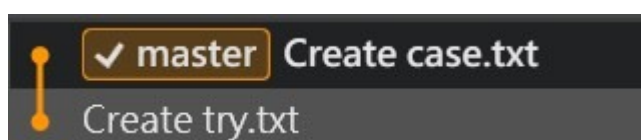
hard:完全刪除

補充：

當有跟其他人共同使用一樣的資料夾，怕 **reset** 檔案救不回來，所以用 **revert** 可以做類似備份的動作，就算之後要使用也好回復，不會有其他問題(**revert** 意思為取消這個節點所做的動作) 「建議都用 **revert**」

不強制刪除節點，可以回復(**Revert**)流程。預設情況回復 **case.txt**：

圖 12 為未 **revert** 之前的情況，在 **Create case.txt** 按右鍵選 **Revert Commit...**(圖 13)後，就會彈出圖 14 的視窗按下 **Revert** 後，在 **All Commits** 就是圖 15 之情況，在 **test** 的資料夾也不會出現 **case.txt** 的檔案。要回復回去就是在 **Revert"Create case.txt"**的節點按右鍵(圖 16)後，也會彈出圖 14 的視窗按下 **Revert** 後，即可回復回來(圖 17 是 **Revert** 之結果)。(在 **test** 資料夾中也會出現)



→ 圖 12

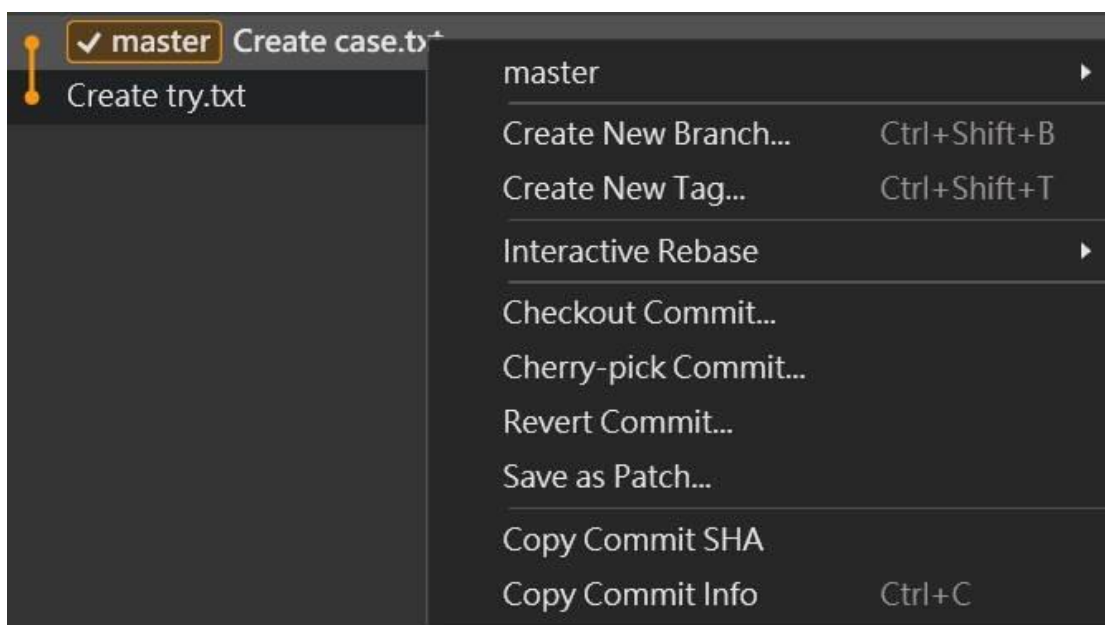


圖 13

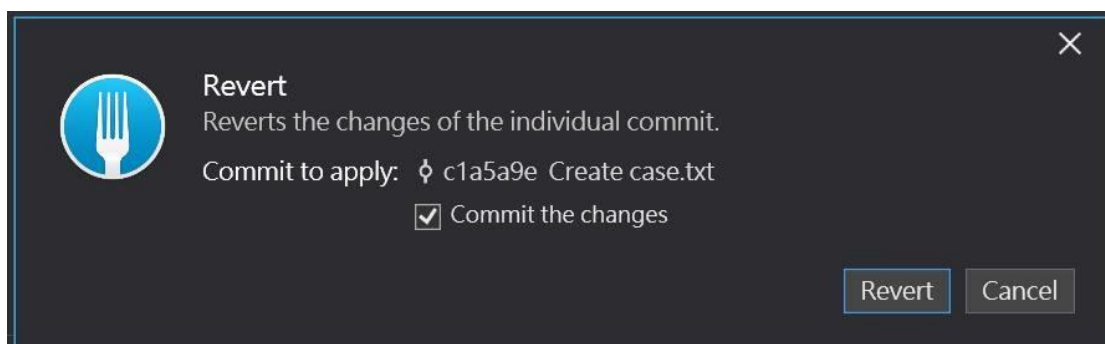
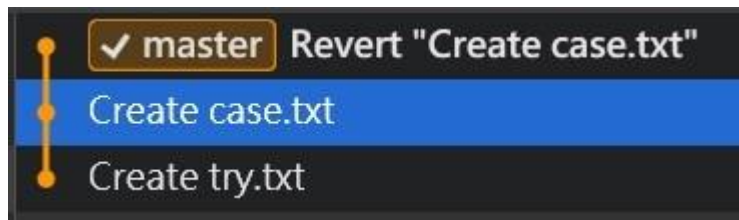


圖 14



→ 圖 15

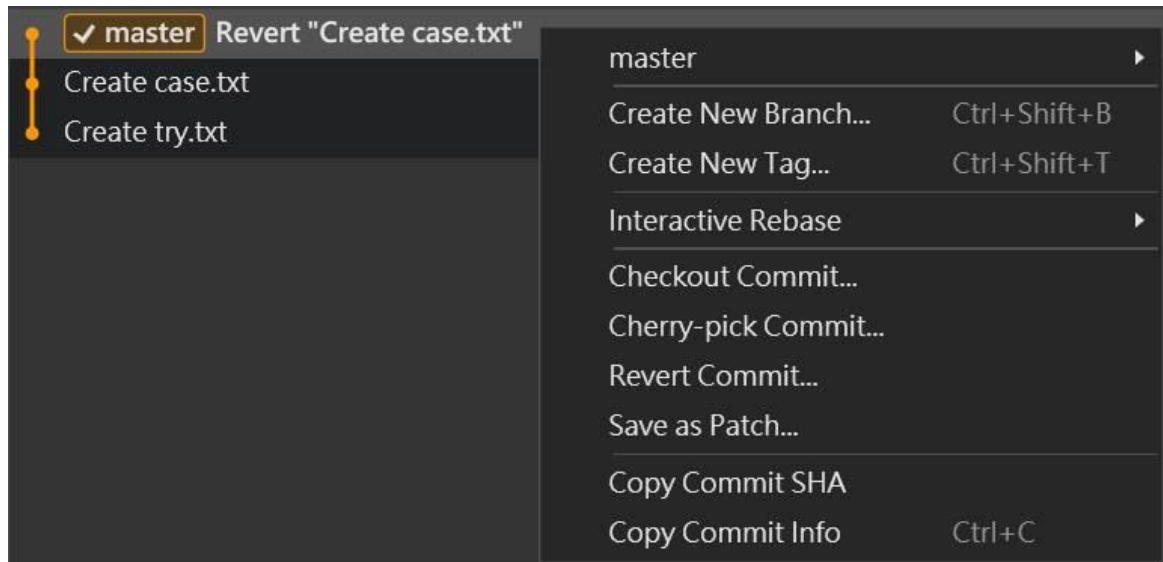
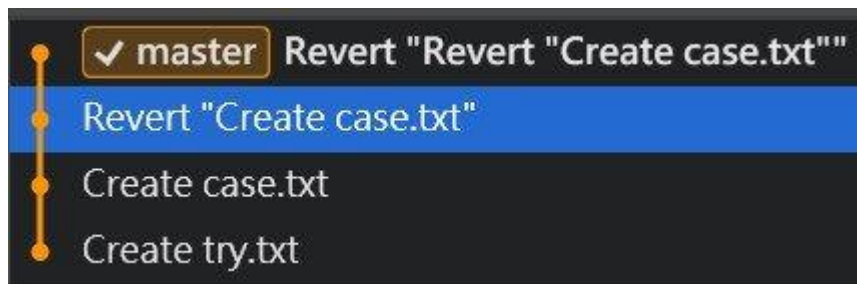


圖 16

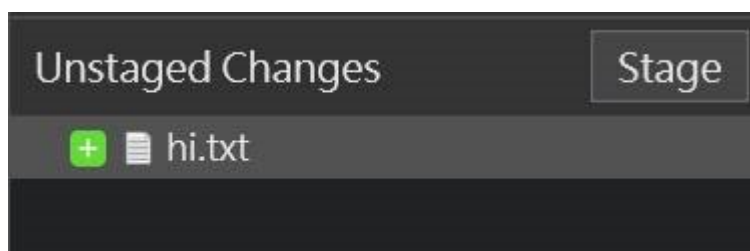


→ 圖 17

在資料夾中建立(或輸入)錯誤的檔案刪除：在 Change Unstaged stage 選擇要回復的檔案 -> 右鍵選 Discard Changes -> 選 Discard -> done

預設情況：刪除 hi.txt

在圖 18 中的 hi.txt 按右鍵就會出現圖 19 的清單，選擇 Discard changes 後，立即彈出圖 20 的視窗按下 Discard 後，如圖 21 在 Unstaged Changes 和 test 資料夾就不會有 hi.txt 的檔案。



→ 圖 18

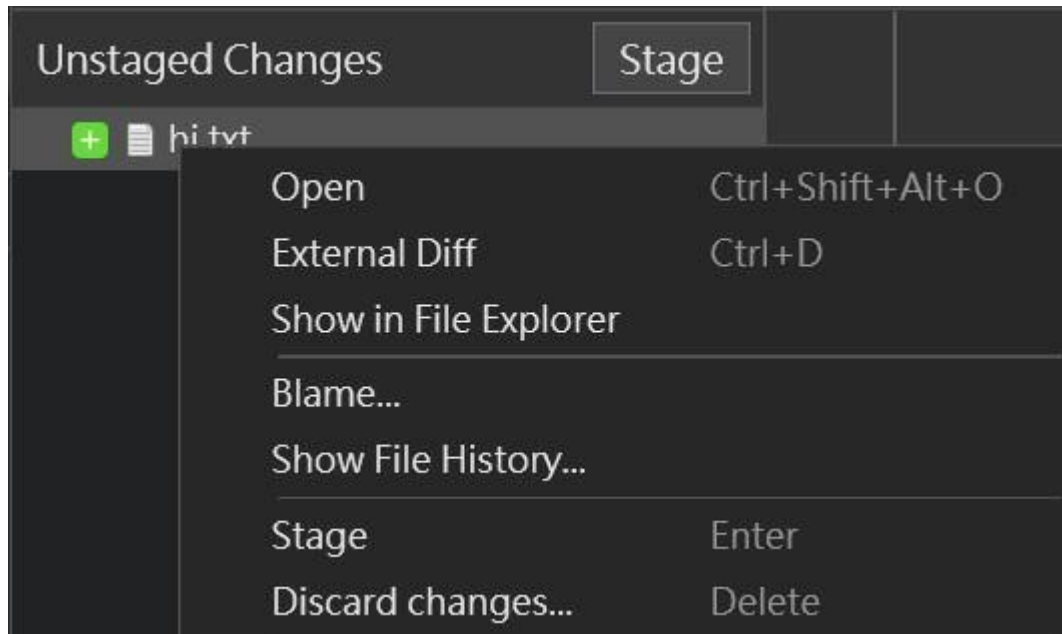


圖 19

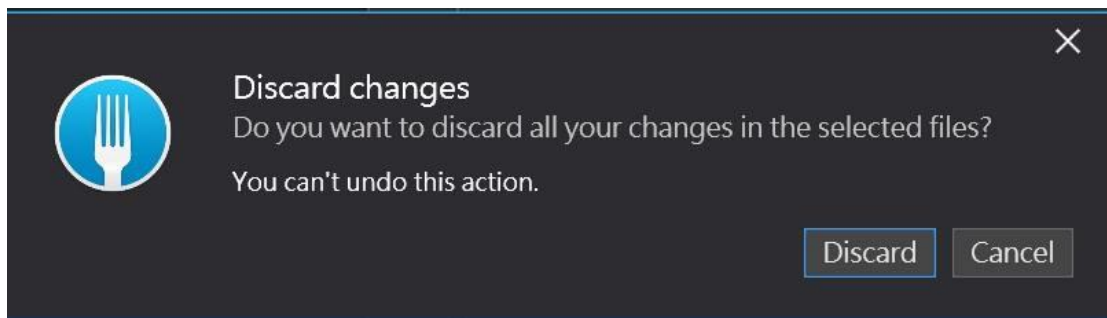


圖 20

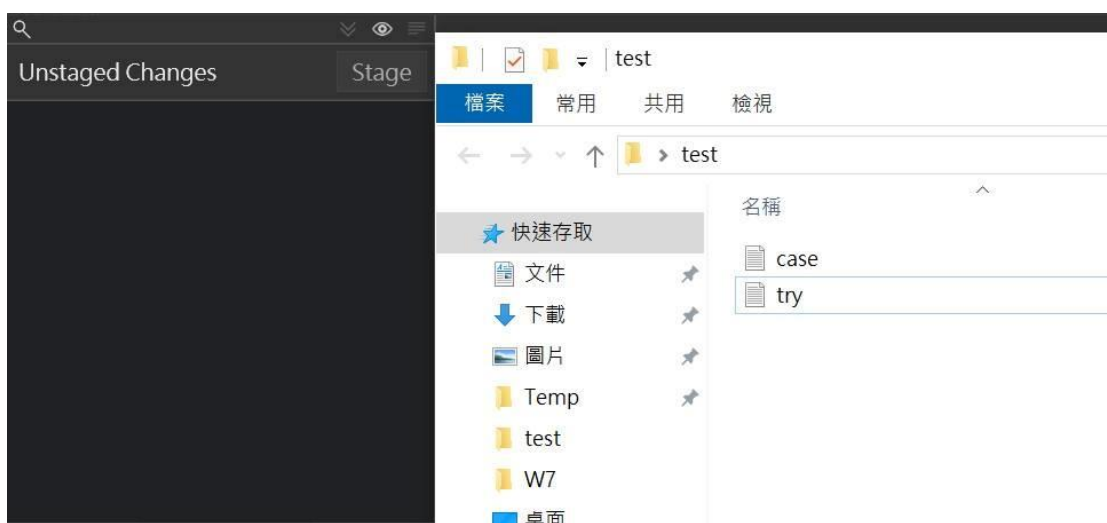


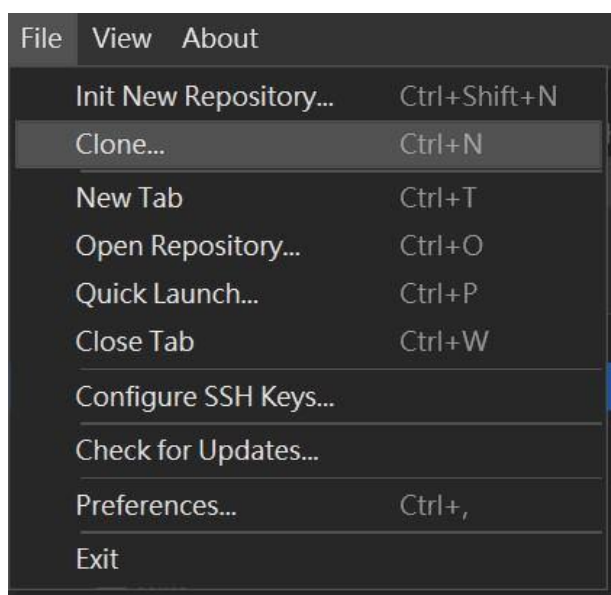
圖 21

上傳到 github(當第一次建立的時候):

在 Github 網站按右上角的「+」 -> 在 Github 網站按 new repository -> 輸入檔名要跟上傳的檔名相同 -> 按 enter 建立(或按 Create repository) -> 複製 github 上方連結 -> 在 fork 的 Remotes 按右鍵 -> 選 add new remotes -> 在 Repository Url 貼上 github 網址 -> 按 Add new remote -> 按 Push -> 再按 Push -> 彈出視窗登入 github -> 在 github 重整後就能看到了

組員下載(一定要在 github 登入的情況下做) :

複製共享資料夾的連結 -> 在 fork 的 File 選 clone (圖 22)-> 在 Repository Url 貼上 github 共享網址(圖 23) -> Parent Folder 選擇資料夾儲存的路徑 -> Name 不要動 ->按 clone -> 登入 github(圖 24) -> 按 login



→ 圖 22

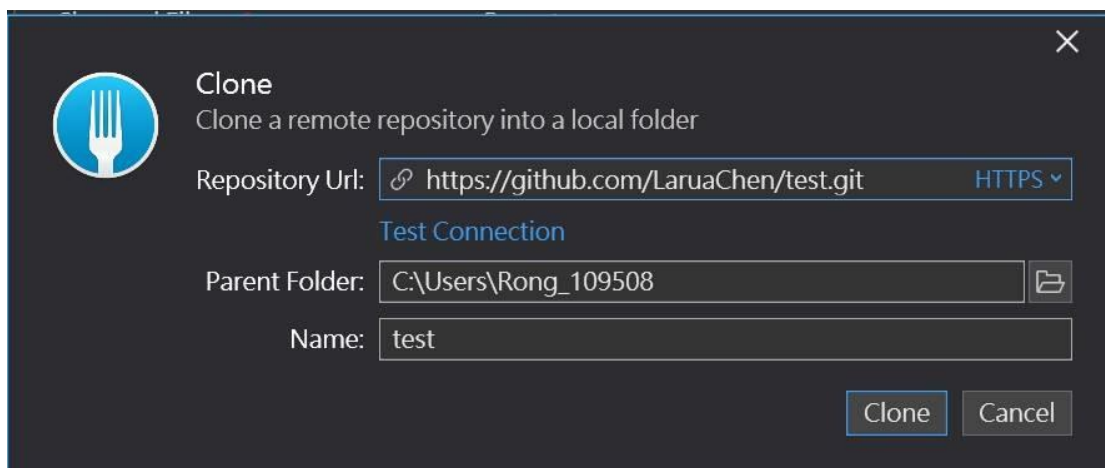
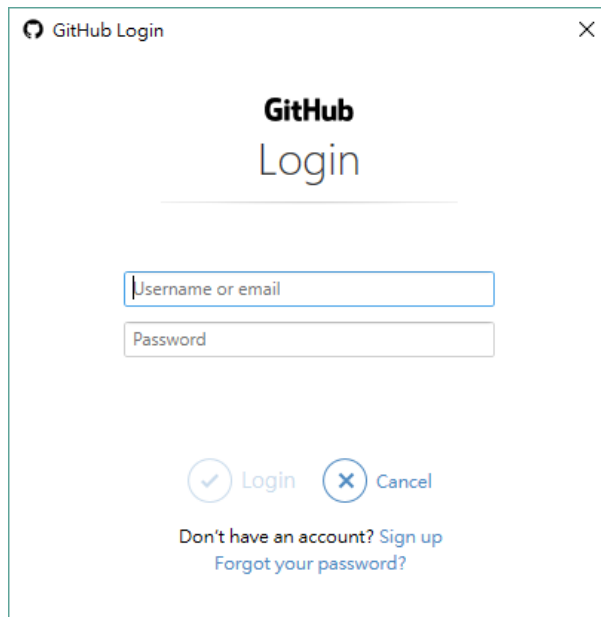
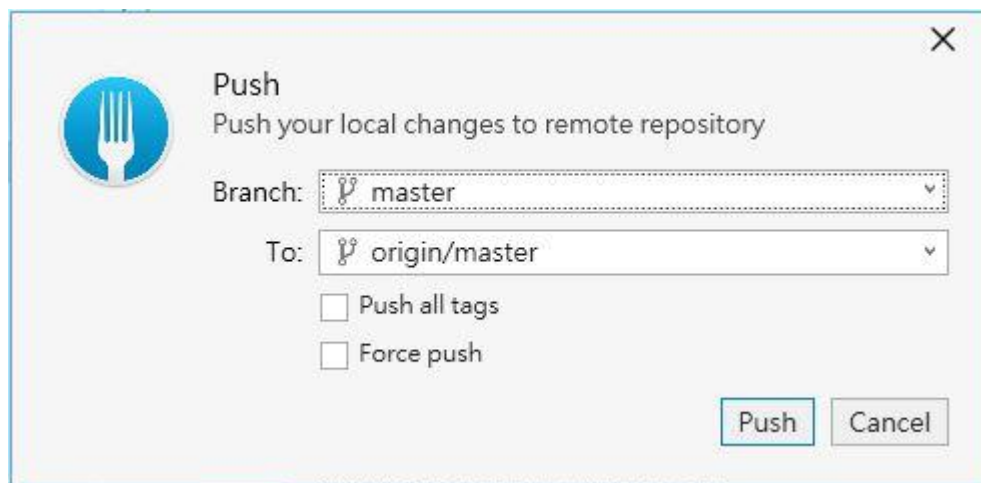


圖 23



→ 圖 24

組員上傳到 github(評分重要環節，一定要在 github 登入的情況下做):
按 Push(會出現圖 25) -> 再按 Push -> 彈出視窗登入 github(若已登入就不會出現) -> 在 github 重整後就能看到了



->圖 25

force push:將本地端的檔案上傳當最終版，而且將雲端上原有的東西弄消失，除非有把握不然不要做

建立分支(將 master 單獨運作):

在最新的 master 上按右鍵 create new branch(圖 26) -> 輸入分支名稱(圖 27) -> create

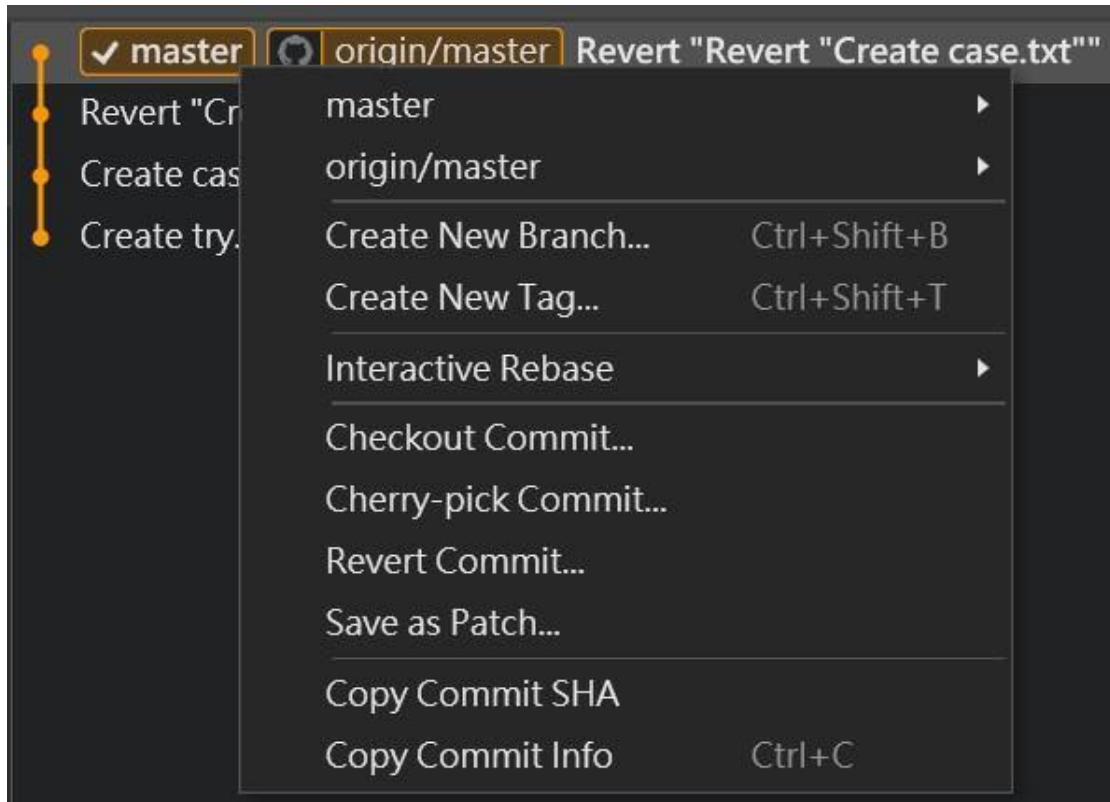


圖 26

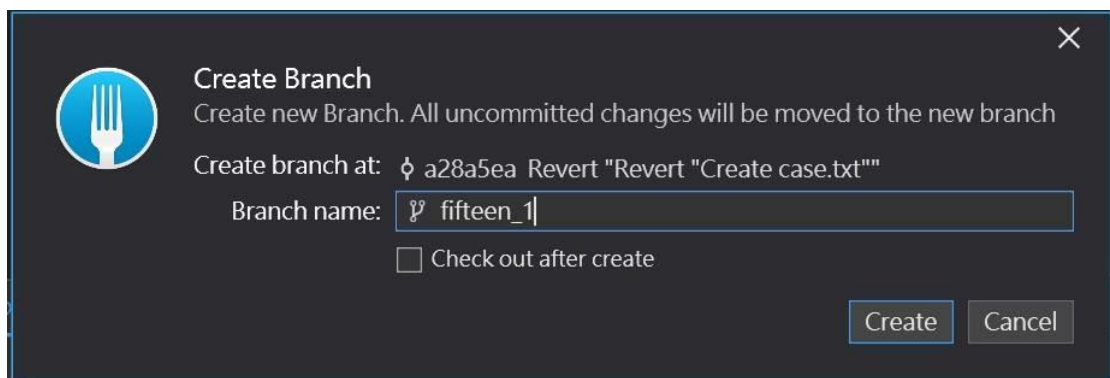
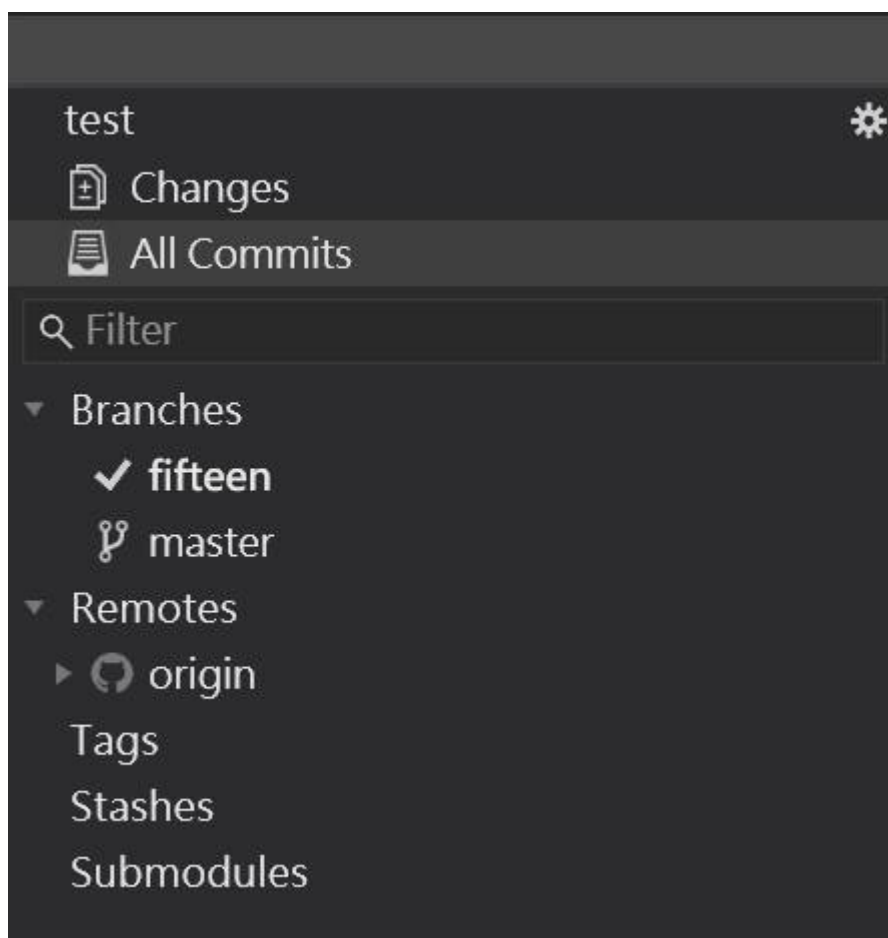


圖 27

選擇分支：在 **fifteen** 上點兩下選擇到這個分支(圖 28)或是在旁邊的 Branches(圖 29)也可以選



圖 28

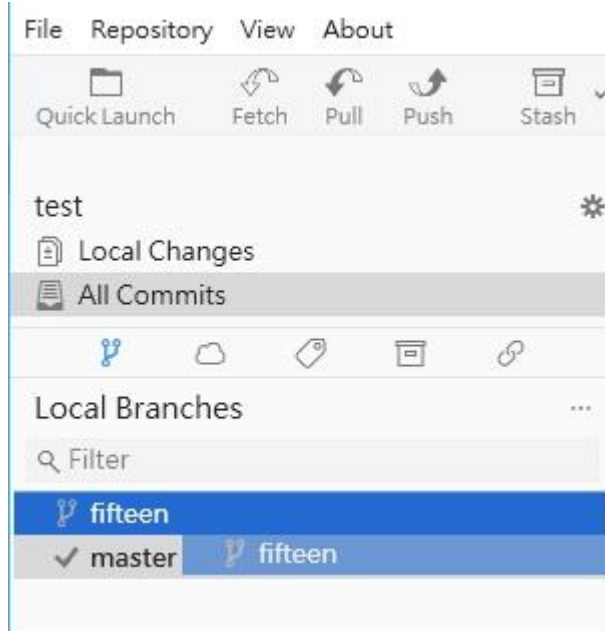


->圖 29

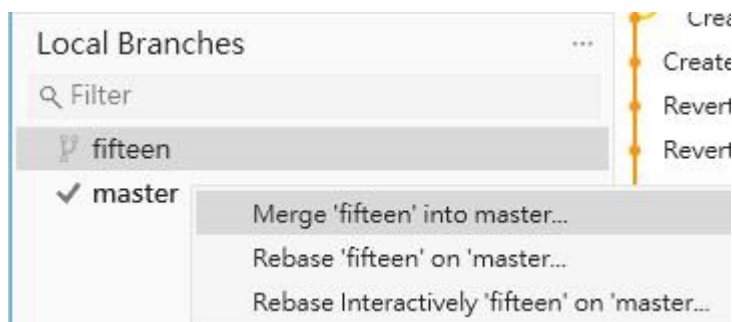
補充：

多人在共用同個資料夾存放資料時，分支主要是在區分不同的功能，或是用來測試功能，確保 **master** 是可以的檔案是可以使用的。若有多人同時使用同個檔案修改上傳到 **github** 並合併到 **master** 會產生衝突，後面有解決方法

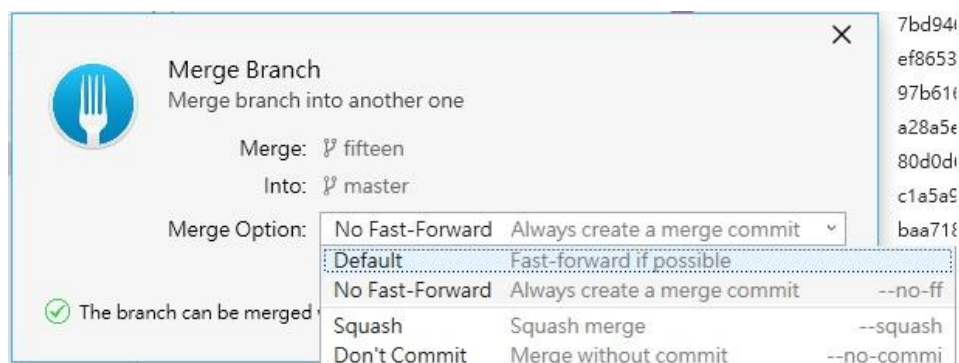
將分支結合到 master 上傳 github(為了安全起見請使用後面的 github 合併方法):
 將分支按著不放拖曳在 master 上(圖 30) -> merge '分支名稱' into master(圖 31) ->
 選 No Fast-Forward(圖 32) -> 按 merge(圖 33)



->圖 30



->圖 31



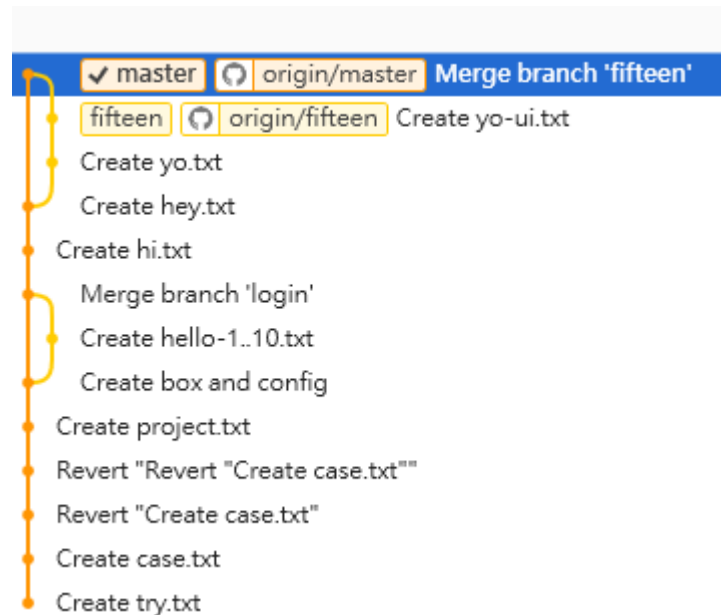
->圖 32



圖 33

建議:

上傳到 github，當有 master 分支和其餘兩個分支需要整合在 master 上，要依序整合不能同時整合。(下面的圖 34 是正確的整合)



→ 圖 34

建立組織: 在 github 網站的右上角按+ -> 建立 new organization(要付費)

上傳其餘分支到 github(為了要讓 master 分支可以正常運作)：

假設情況：上傳 fifteen 的分支

在 fork 建立「fifteen」分支，並在此分支中建立檔案後，在 fork 介面按 Push，上傳完後就能在 Github 上看到 fifteen 分支，在 fork 介面也會看到 origin/fifteen 的分支(這是遠端的分支)

為什麼不能 pull 下來

將雲端檔案下載回本地端步驟：

在 github 複製 clone 的網址(在 Clone or download 的綠紐，圖 35) -> 在 fork 裡選 File -> clone(圖 36) -> 在 Repository Url 貼上 clone 的網址(圖 37) -> 在 Parent Folder 選擇你要放置資料夾的路徑 -> 再按 clone 即可

確認雲端和本地端是否相同：看 master 和有 github/master 是否在 All Commits 上最新的紀錄且在同一行紀錄上(圖 36 紅框處)

fetch:下載看遠端的狀況

pull:下載來看遠端的狀況和分支也一併下來(pull = fetch + merge)

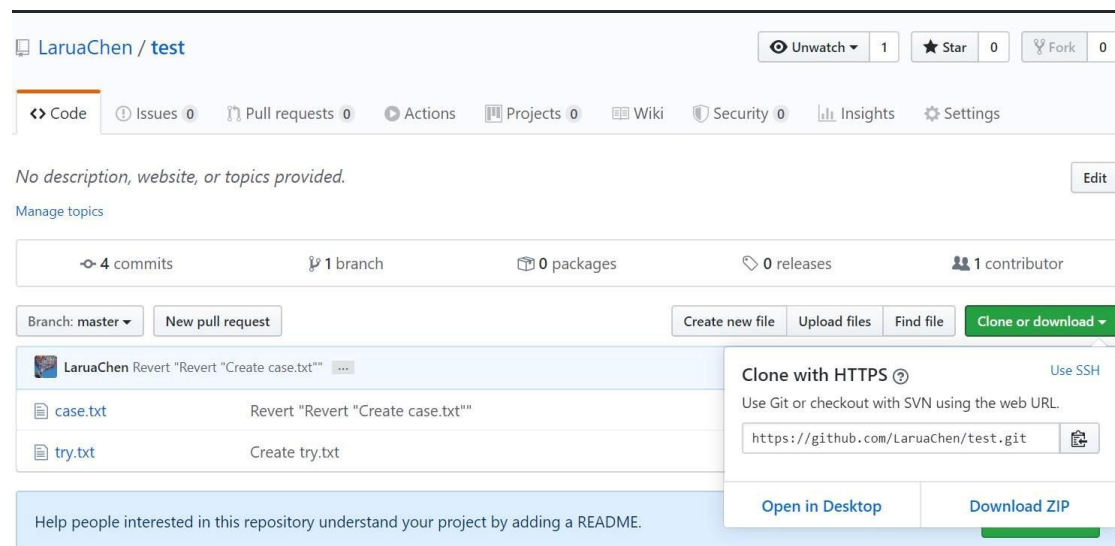


圖 35

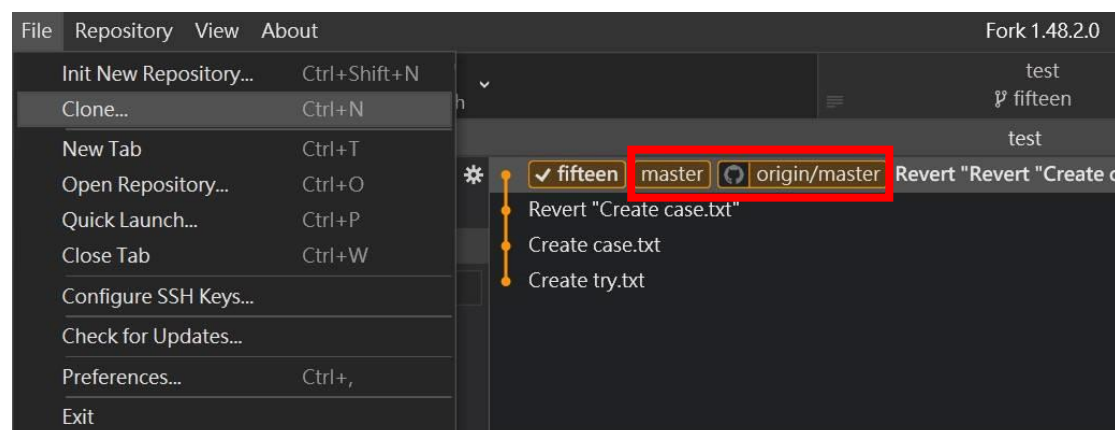


圖 36

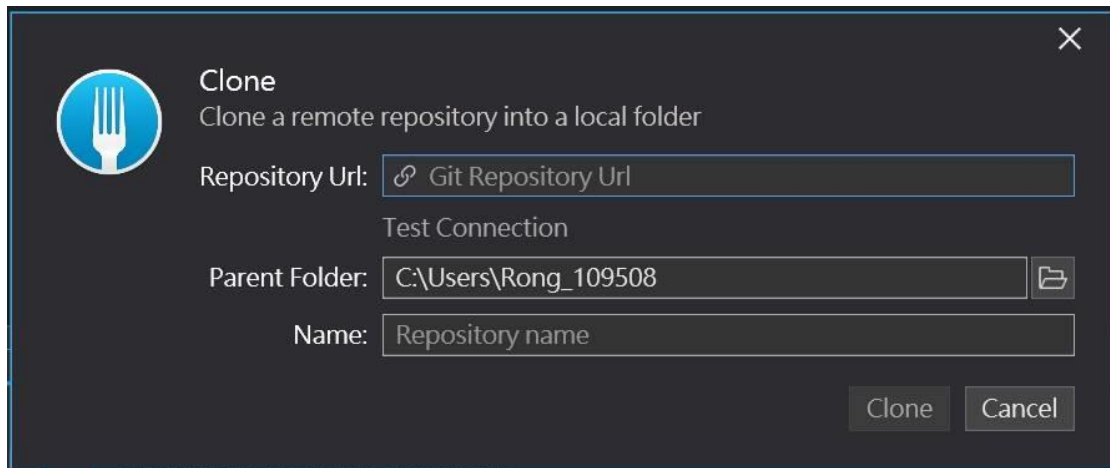


圖 37

用 github 合併分支(因用 fork merge 會容易出錯，**合併分支一次合併一個**)：
 要先確認分支是否上傳到 github 上(圖 38 紅框處) -> github 選 Pull requests(圖 39)
 -> new pull request(圖 40) -> compare 選要 merge 的分支(圖 41) -> create pull
 request(圖 41) -> 名稱改成「Merge 分支名稱」(圖 42) -> Ceate pull request (圖 42)
 -> 成功 merge 後變成綠燈(代表沒有衝突，圖 43) -> 按 Merge pull request(圖 43)
 -> Confirm merge(做「Confirm merge」動作要通知群組) -> fetch(看雲端的狀況)

假設情況：要將 10556015 分支及檔案合併到 master 分支

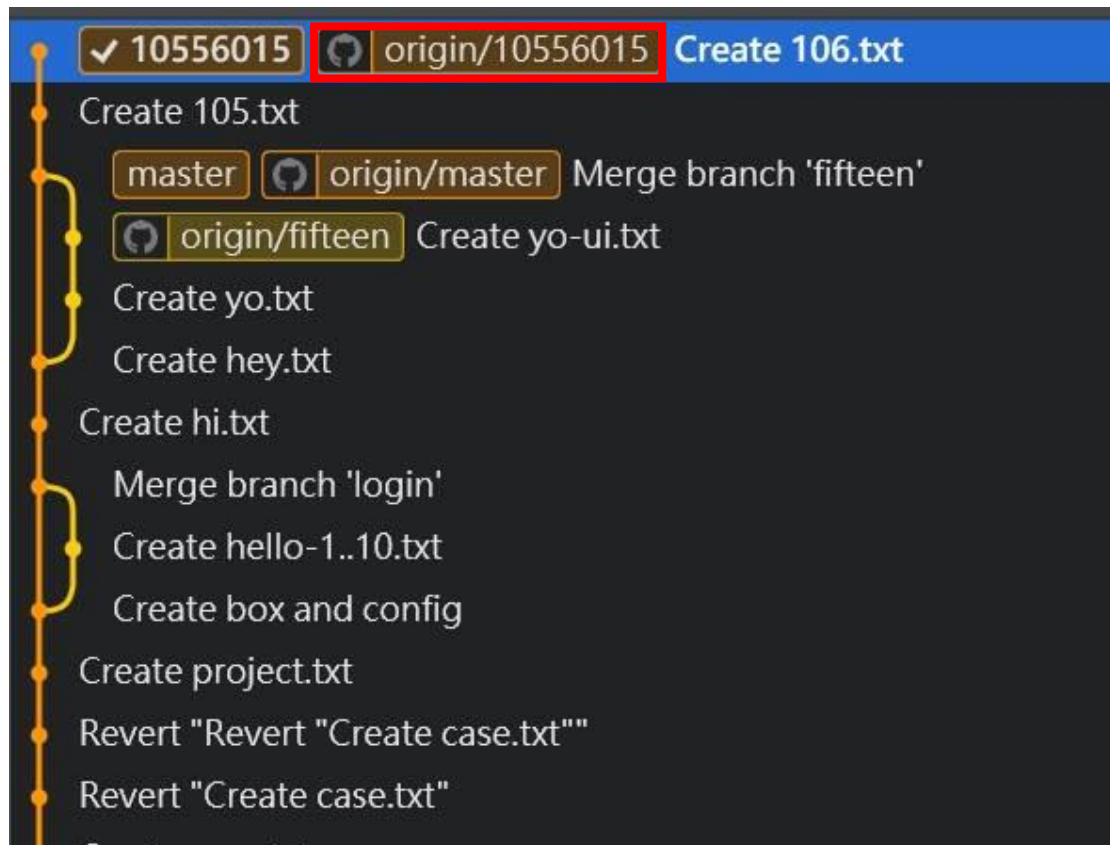


圖 38

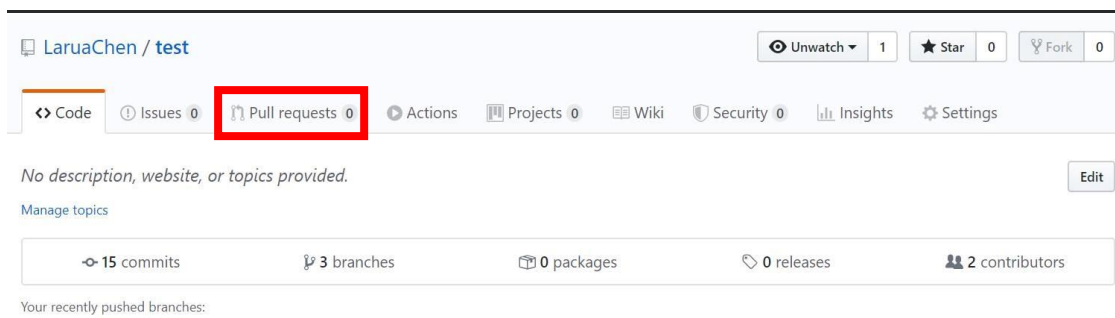


圖 39

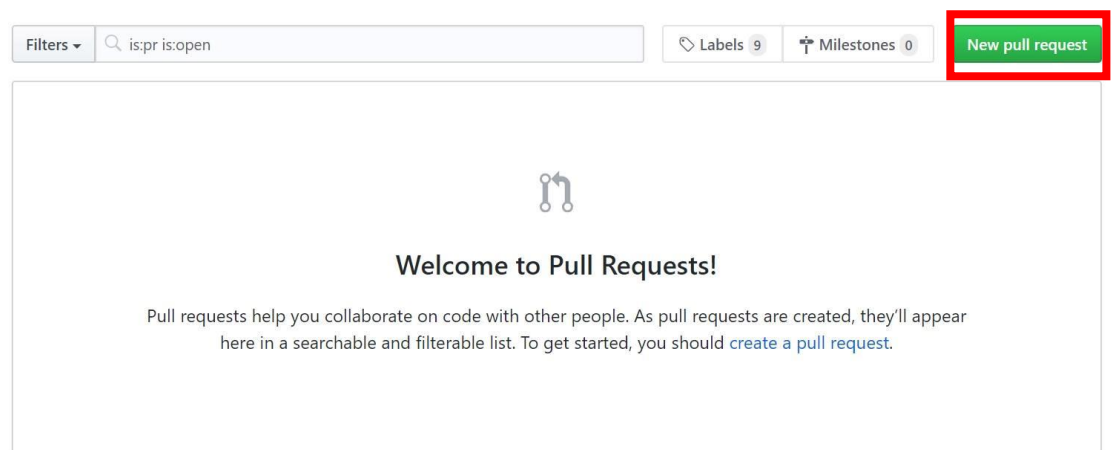


圖 40

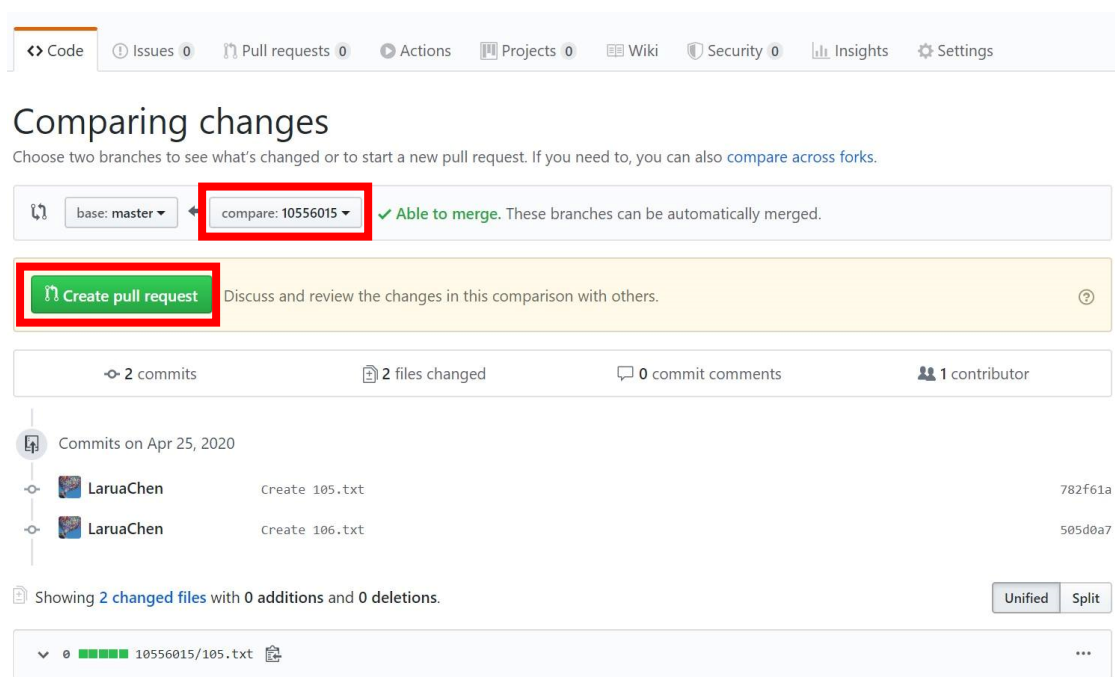




圖 41

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across for](#)

 base: master compare: 10556015 ✓ Able to merge. These branches can be automatically merged.



Write

Preview

AA B i “ <> 🔗 ☰ ☷ ✓ @ ★ ↶

Leave a comment


Attach files by dragging & dropping, selecting or pasting them.


MS


Create pull request

圖 42

Add more commits by pushing to the **10556015** branch on **LaruaChen/test**.



 Continuous integration has not been set up
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

 This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request

You can also open this in GitHub Desktop or view command line instructions.

圖 43

Add more commits by pushing to the **10556015** branch on **LaruaChen/test**.



Merge pull request #1 from LaruaChen/10556015

Merge 10556015

Confirm merge

Cancel

圖 44

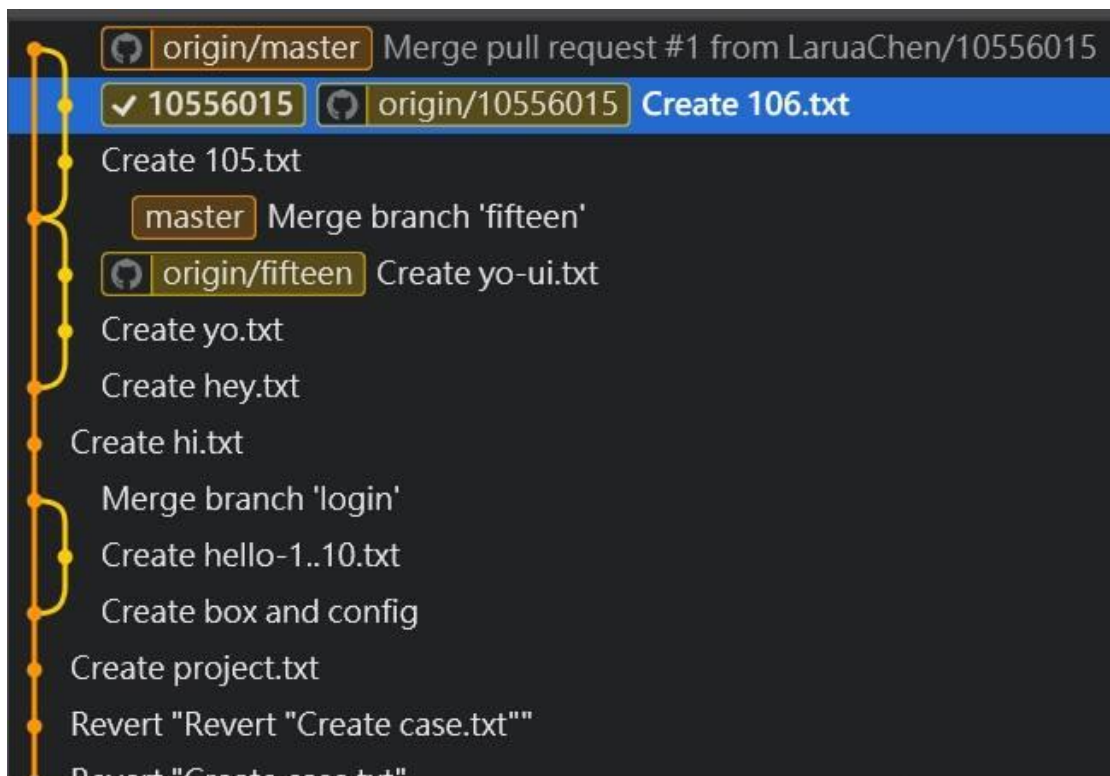


圖 45(這圖是 10556015 分支合併到 master 分支之後的狀況)

在圖 45 中若要把本地端的 master 更新到與雲端上的(origin master)相同紀錄，在 fork 介面要選擇本地端的 master 並按下 Pull 就能成功。(圖 46)

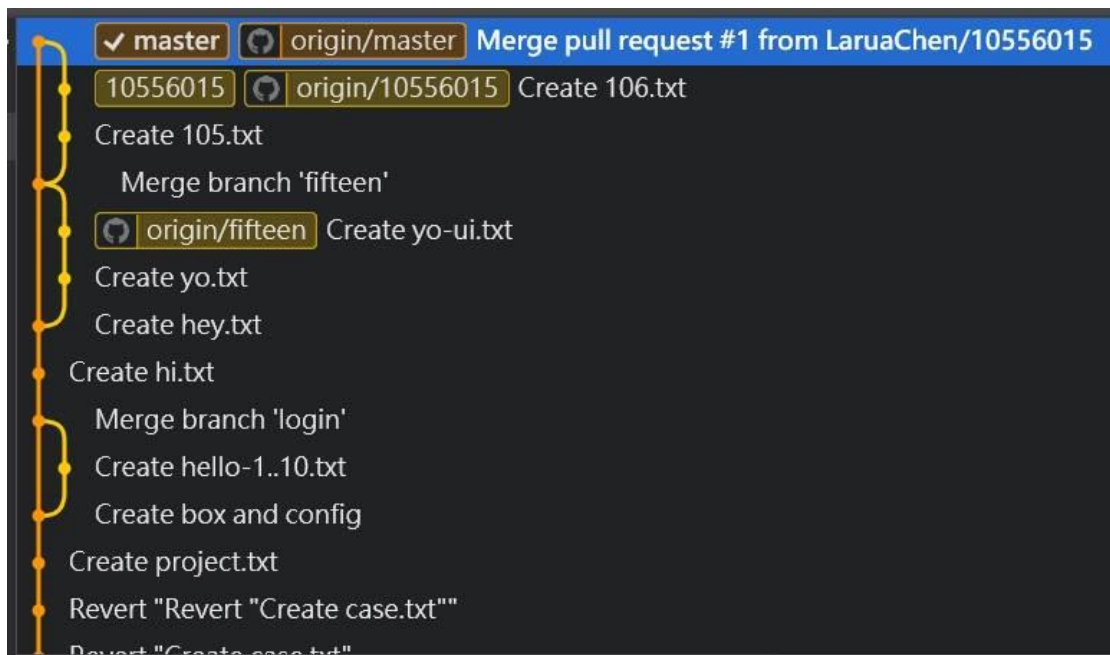


圖 46

解決兩人上傳相同檔案的衝突：

假設情況：在同個分支下更動同個檔案，並且有同時兩人在自己建立的分支上傳同個檔案到 **github**，有一位已經合併好到 **master** 裡，以至於另外一位不能合併到 **master** 裡

舉例：有一位 **A** 和一位 **B** 同時在修改 **master** 分支下的 **try.txt**，並在各自新增一個分支把 **try.txt** 新增節點在這個新分支中，兩位也同時上傳到 **github** 上要合併到 **master** 裡，**A** 已經合併好了，但 **B** 因為 **A** 所合併的檔案而產生衝突不能合併

解決辦法：

Resolve conflicts(圖 47) -> 將留下資料之外的東西都需要刪除(圖 48) -> Mark as resolved(在圖 48 的右上角紅框處) -> commit merge(圖 49) -> Merge pull request(圖 50) -> confirm merge(圖 51)

可以此影片：Screen Recording 2020-01-15 at 4.21.50 PM

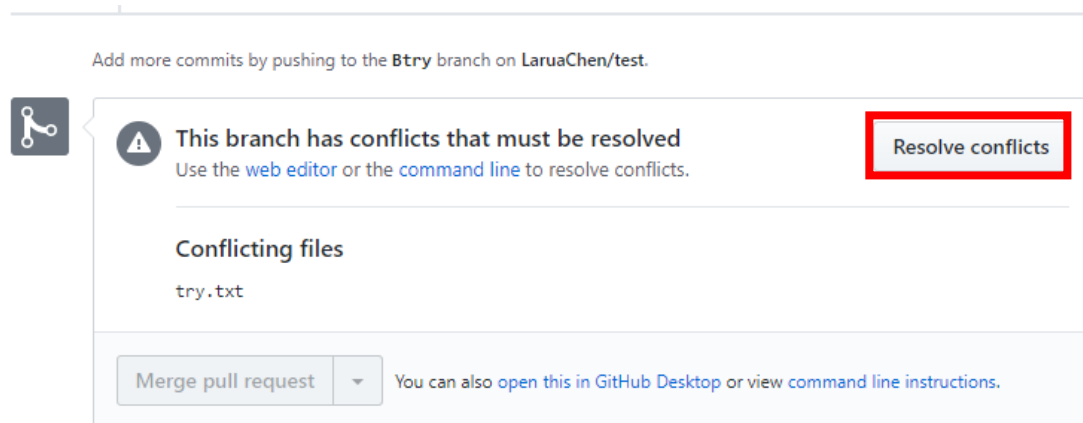


圖 47(產生衝突)

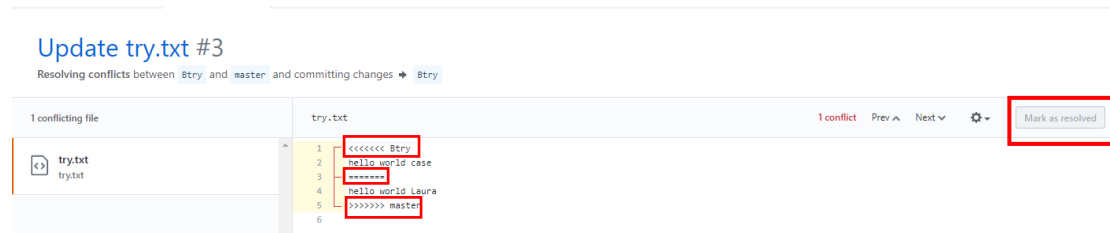


圖 48(紅框處必須刪除，其餘的看要留下 **A** 或是 **B** 或者都留下，此例情況先都留下)

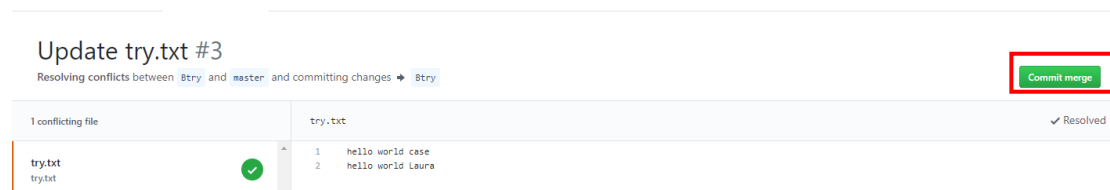


圖 49

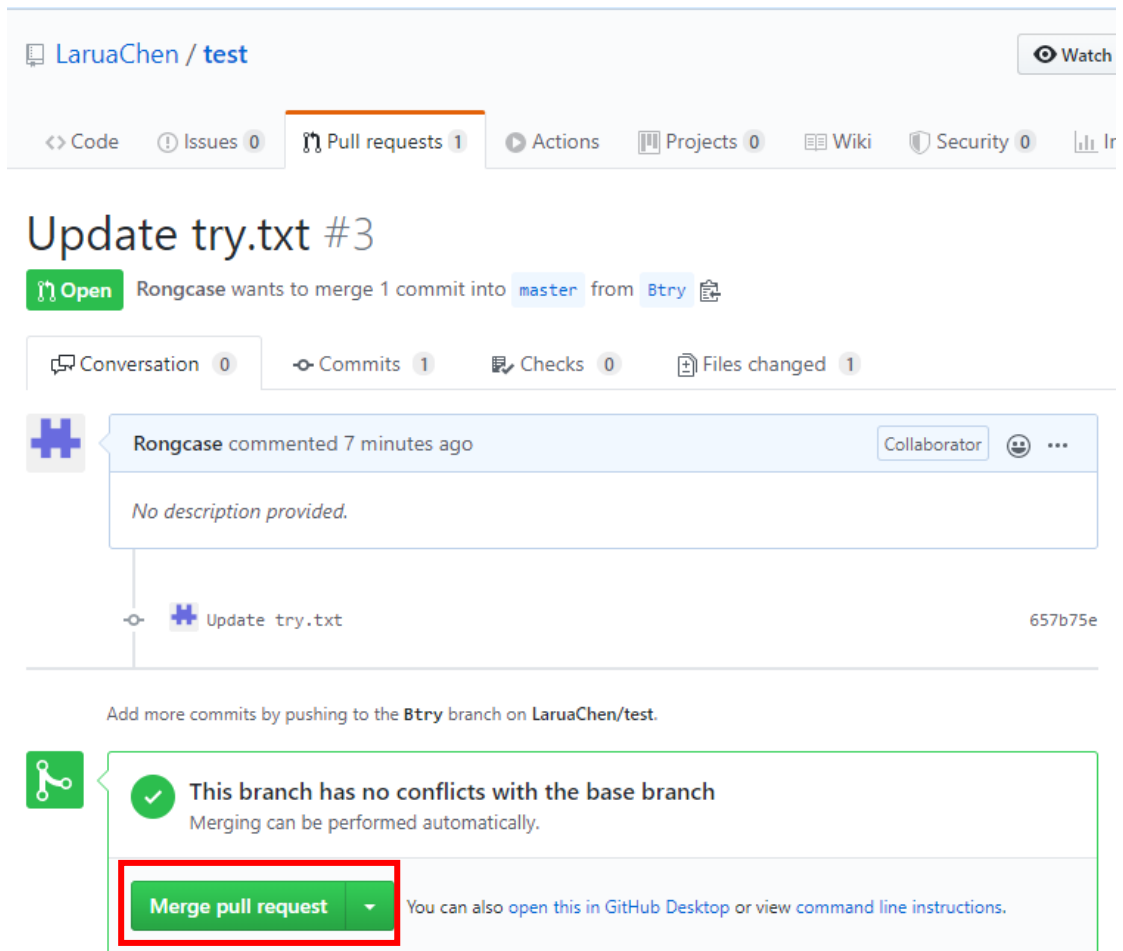


圖 50

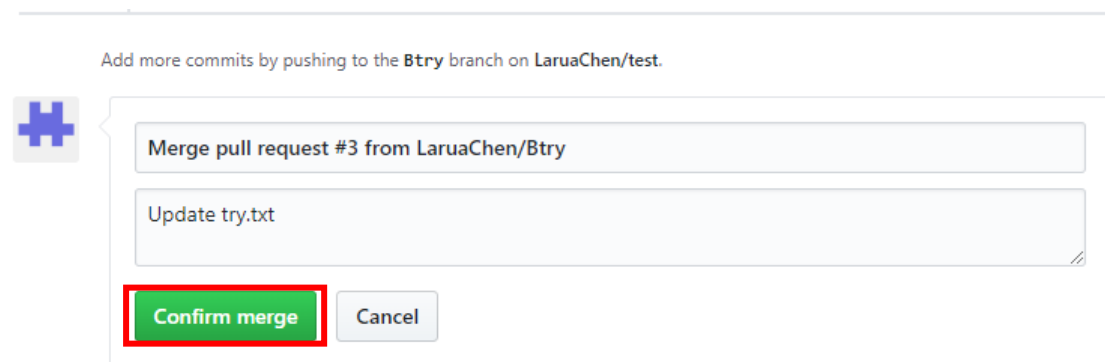
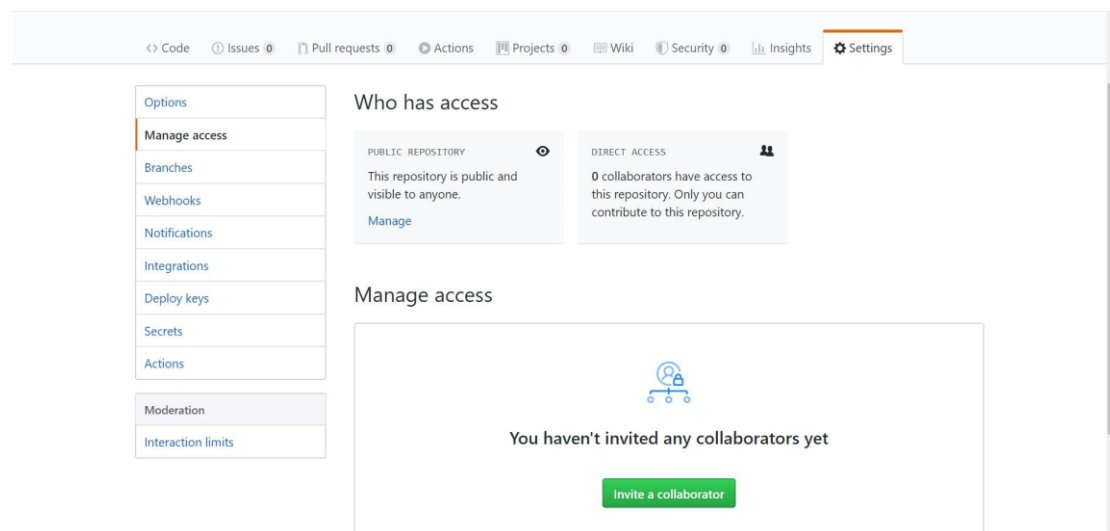
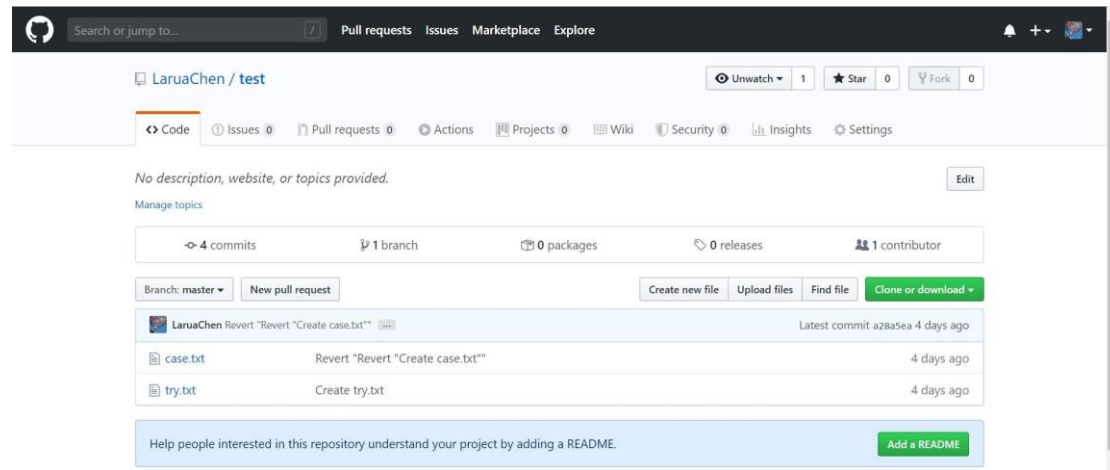


圖 51

開協同合作:

在 github 要共享的資料夾首頁按 setting -> Manage access -> Invite a collaborators
->輸入要協同的使用者 github 名稱

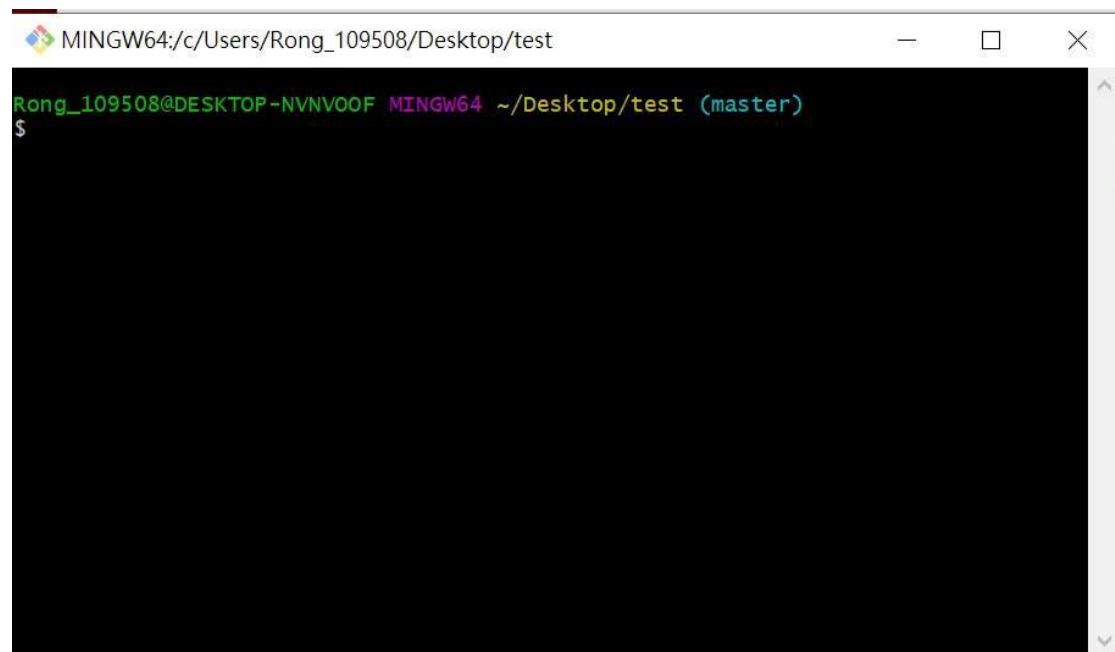
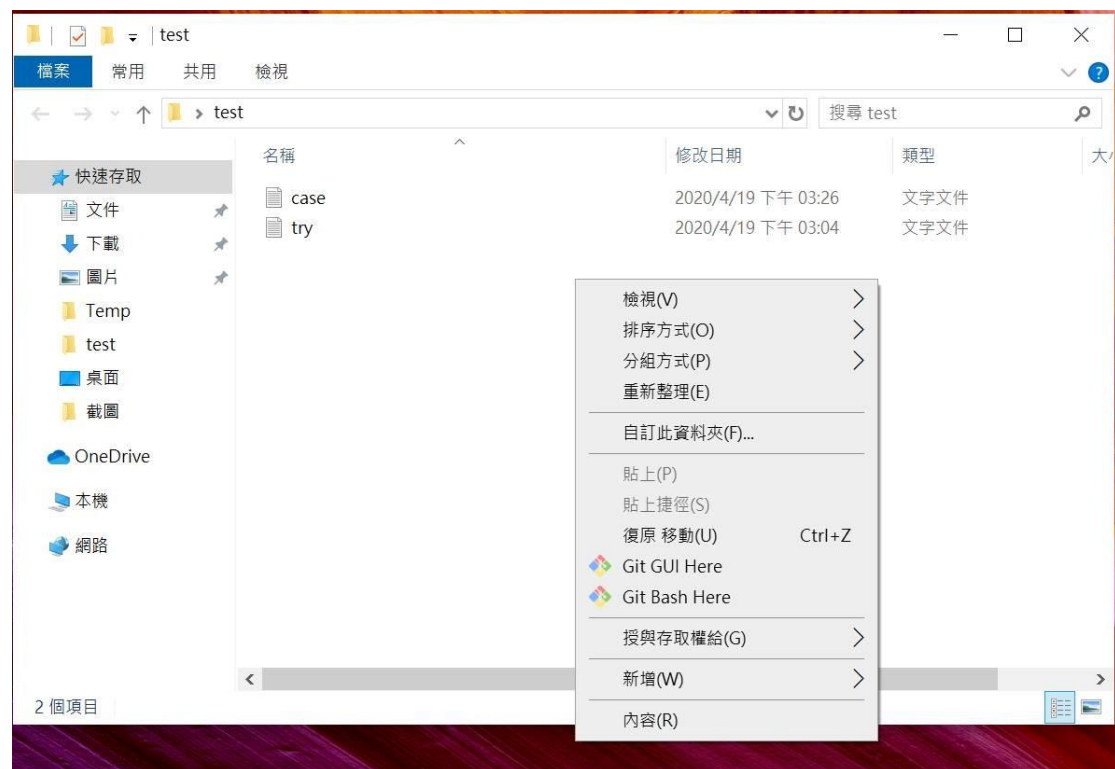


被邀請的人要在信箱裡面去確認看有沒有邀請信

Github 指令：

指令網址：<http://0102.tw/md>

開啟指令視窗:在資料夾內按右鍵 -> git bash here

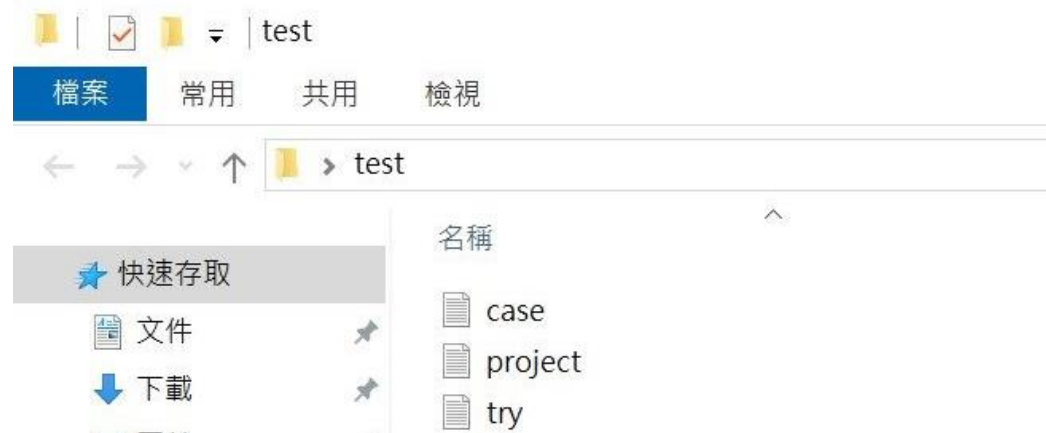


git init:開啟資料夾路徑

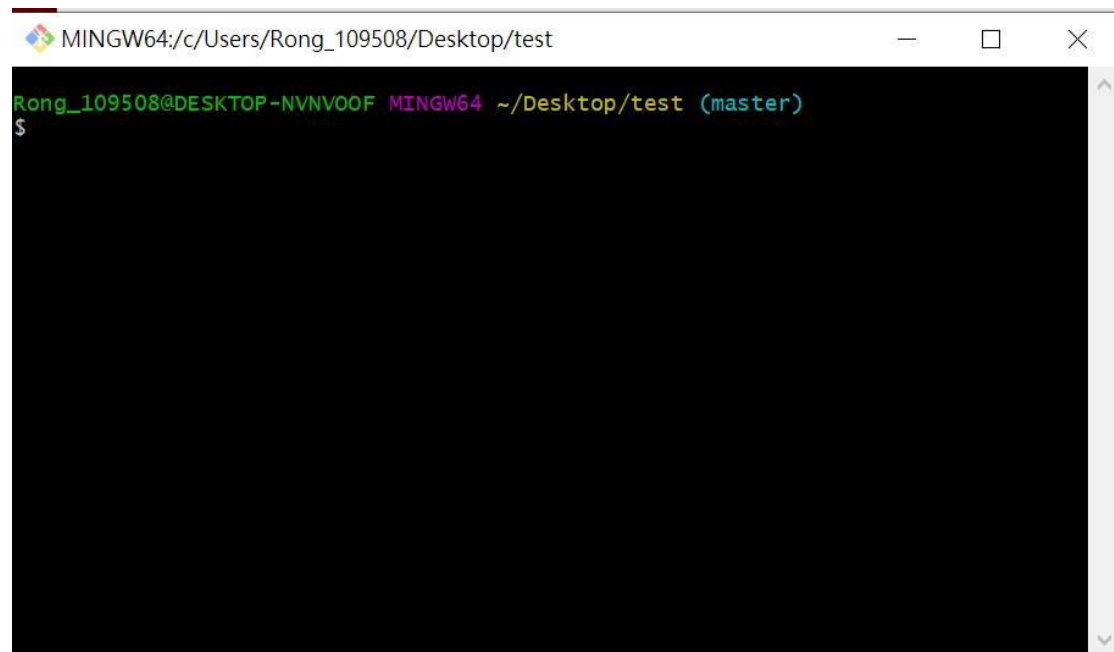
```
MINGW64:/c/Users/Rong_109508/Desktop/test
Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test (master)
$ git init
Reinitialized existing Git repository in C:/Users/Rong_109508/Desktop/test/.git/
```

touch project.txt：在 master 分支中建立「project.txt」檔案

```
Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test (master)
$ touch project.txt
```

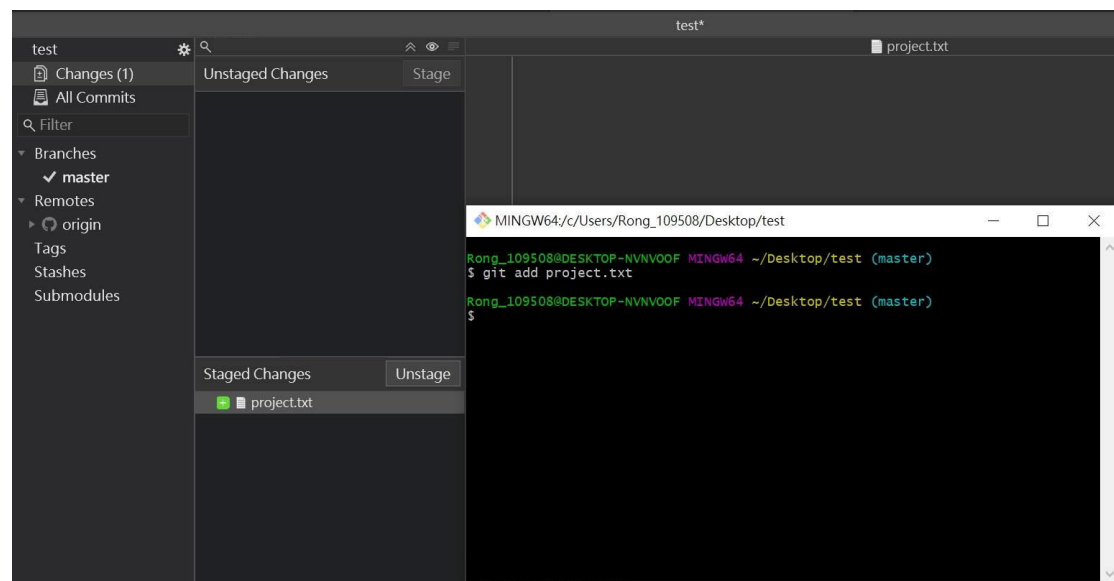


ctrl + L : 清空所有指令

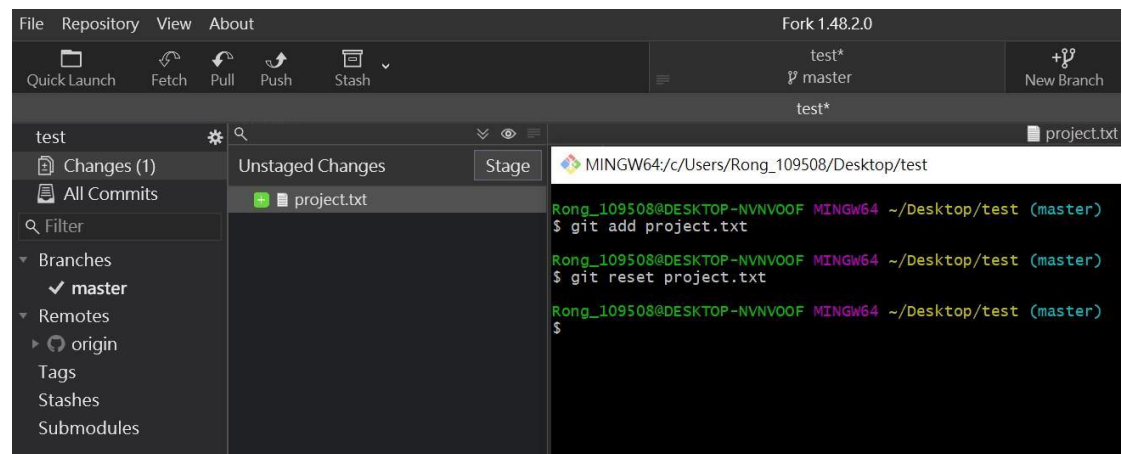


```
MINGW64:/c/Users/Rong_109508/Desktop/test
Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test (master)
$
```

git add project.txt : 跑到 Staged Changes(在指令介面中放進資料夾的檔案需要 add 進去)



git reset project.txt: 跑到 UnStaged Changes



補充：

add 選取一個檔案

Create 建立新檔案

update 更新檔案

git status: 顯示紅色是未選取(reset)，綠色是以選取(add)，顯示資料夾的狀態

```
Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test (master)
$ git reset project.txt

Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

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

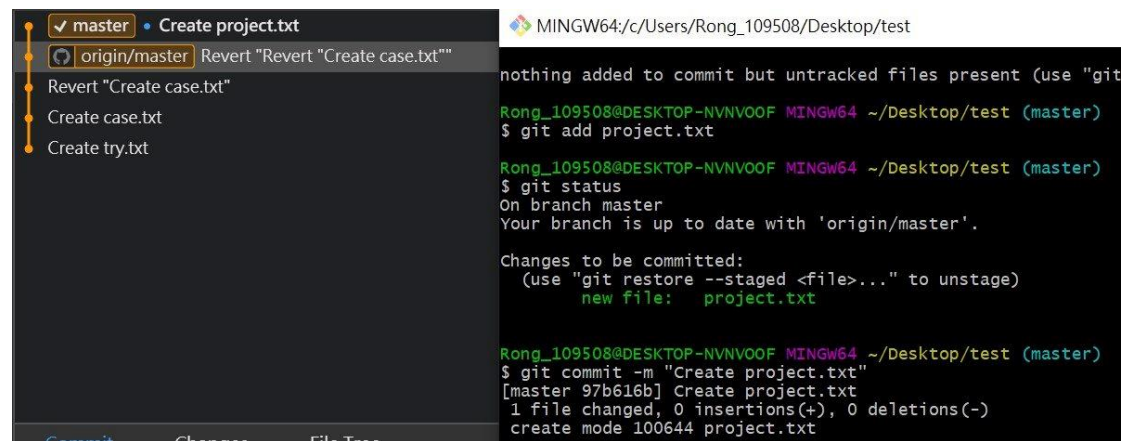
nothing added to commit but untracked files present (use "git add" to track)
```

```
Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test (master)
$ git add project.txt

Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
  new file:   project.txt
```

git commit -m "Create project":下節點



The screenshot shows a Git GUI on the left and a terminal on the right. The GUI shows a commit history with 'Create project.txt' as the latest commit. The terminal shows the following commands and output:

```
MINGW64:/c/Users/Rong_109508/Desktop/test

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

Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test (master)
$ git add project.txt

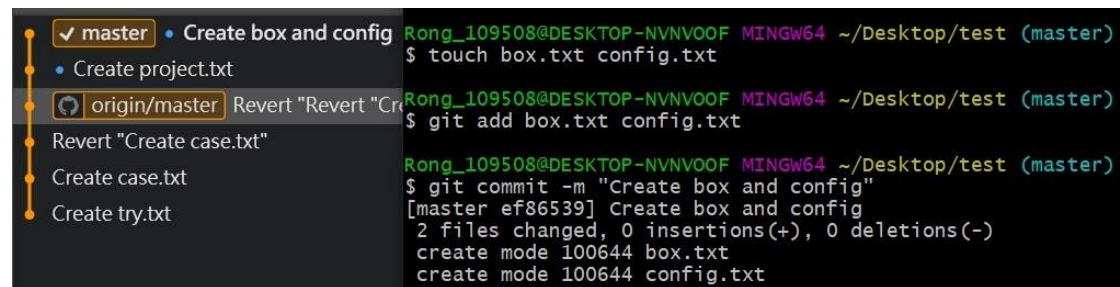
Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   project.txt

Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test (master)
$ git commit -m "Create project.txt"
[master 97b616b] Create project.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 project.txt
```

git commit -m "Create box and config":下節點(同時選取 box 和 config 檔)

以下的圖是新增兩個檔案去建立節點的流程



The screenshot shows a Git GUI on the left and a terminal on the right. The GUI shows a commit history with 'Create box and config' as the latest commit. The terminal shows the following commands and output:

```
Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test (master)
$ touch box.txt config.txt

Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test (master)
$ git add box.txt config.txt

Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test (master)
$ git commit -m "Create box and config"
[master ef86539] Create box and config
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 box.txt
create mode 100644 config.txt
```

輸入你的 github user name 和 github email :

git config --global user.name "Your Name"

git config --global user.email "you@example.com"



The screenshot shows a terminal with the following commands and output:

```
Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test (master)
$ git config --global user.name "LauraChen"

Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test (master)
$ git config --global user.email "10556015@ntub.edu.tw"
```

git log:查詢節點紀錄

```
Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test (master)
$ git log
commit ef865399d7a6da73880c884f7fcd95edc500bb49 (HEAD -> master)
Author: LauraChen <10556015@ntub.edu.tw>
Date: Thu Apr 23 15:58:17 2020 +0800

    Create box and config

commit 97b616b2f5f56844e4c79a06f5fb4482e470aa6d
Author: LauraChen <10556015@ntub.edu.tw>
Date: Thu Apr 23 15:43:58 2020 +0800

    Create project.txt

commit a28a5ea9d43efe6ee7549c5b76bb7cec716badfc (origin/master)
Author: LauraChen <10556015@ntub.edu.tw>
Date: Sun Apr 19 15:26:24 2020 +0800

    Revert "Revert "Create case.txt""

    This reverts commit 80d0d6f94d83ceb8496362345a8e58872e6cab4c.

commit 80d0d6f94d83ceb8496362345a8e58872e6cab4c
```

git config --global alias.lg "log --color --graph --all --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit --"

git lg:查詢節點紀錄，格式比較好看(前提是要先輸入「git config --global alias.lg "log --color --graph --all --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit --"」)

```
Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test (master)
$ git config --global alias.lg "log --color --graph --all --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit --"

Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test (master)
$ git lg
* ef86539 - (HEAD -> master) Create box and config (12 minutes ago) <LauraChen>
* 97b616b - Create project.txt (26 minutes ago) <LauraChen>
* a28a5ea - (origin/master) Revert "Revert "Create case.txt"" (4 days ago) <LauraChen>
* 80d0d6f - Revert "Create case.txt" (4 days ago) <LauraChen>
* c1a5a9e - Create case.txt (4 days ago) <LauraChen>
* baa7186 - Create try.txt (4 days ago) <LauraChen>
```

新增「login」分支: `git checkout -b login`

```
Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test (master)
$ git checkout -b login
Switched to a new branch 'login'
```

列出所有分支: `git branch`(綠色和星號來表示目前所選的分支名稱)
(login):淺藍色部分是目前選的分支

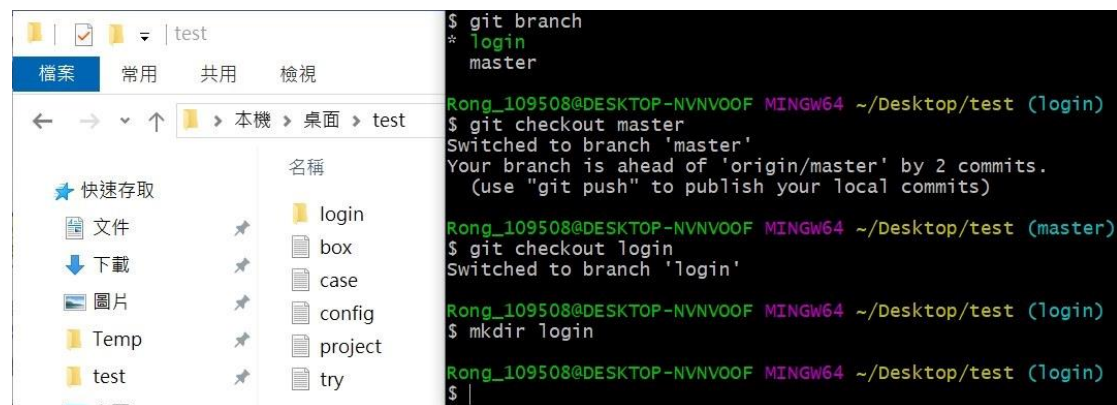
```
Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test (login)
$ git branch
* login
master
```

選擇「master」分支: `git checkout master`

```
Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test (login)
$ git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 2 commits.
(use "git push" to publish your local commits)

Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test (master)
$
```

新增「login」資料夾: `mkdir login`



指令介面:

`ctrl + insert`: 複製(在指令介面上按右鍵也可以)

`shift + insert`: 貼上(在指令介面上按右鍵也可以)

切換到 login 資料夾：cd login/ (切換工作目錄)

```
Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test (login)
$ cd login

Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test/login (login)
$ |
```

cd .. 返回上一層(d 和.之間要加空白)

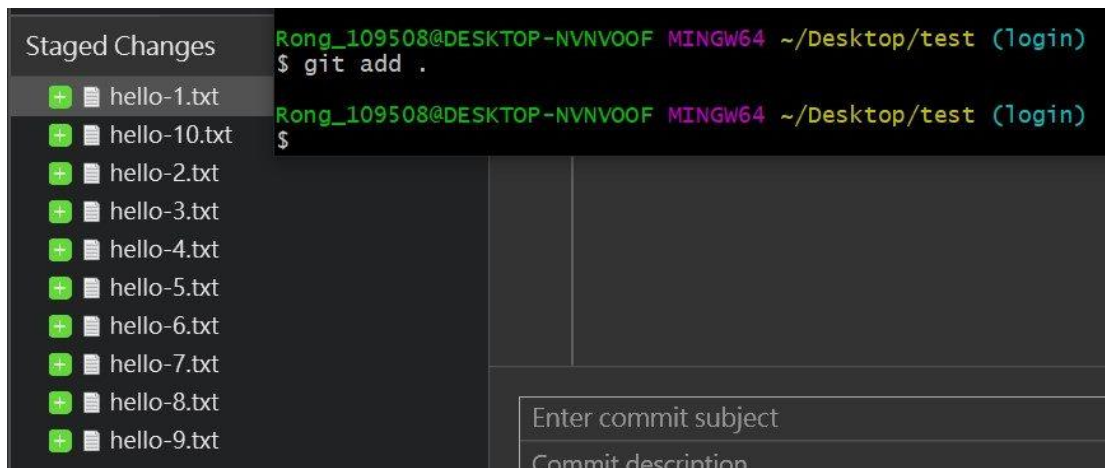
```
Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test/login (login)
$ cd ..

Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test (login)
$ |
```

touch hello-{1..10}.txt 建立十個檔案

```
Rong_109508@DESKTOP-NVNVOOF MINGW64 ~/Desktop/test (login)
$ touch hello-{1..10}.txt
```

git add . 選取全部資料(d 和.之間要加空白)



The screenshot shows a terminal window with the command `git add .` being executed. The output shows the command prompt returning. To the left of the terminal, a file explorer window is open, displaying a list of files under 'Staged Changes'. The files listed are hello-1.txt through hello-9.txt, each with a green plus icon indicating they are staged for commit. Below the file list, there are input fields for 'Enter commit subject' and 'Commit description'.

git merge login --no-ff :將 login 分支建立節點到 merge 分支(在 master 輸入會進入到 vim 編輯器)

退出 vim 編輯器：按 esc -> :wq(保存並退出)

git config --global core.editor notepad：關閉 vim 編輯器

git merge login --no-ff --no-edit:將 login 分支建立節點到 merge 分支(不會進入編輯器)

\$ git remote add origin https://github.com/LaruaChen/git-cli-startup.git: 建立
remote 連線

git push -u origin master:上傳 master 分支
*merge 上傳後不要再做其他事

git push:將全部記錄上傳

git commit -m "Update project" : commit 更新

git pull: 下載

git diff: 顯示異動的檔案

git blame : 顯示指令清單

感謝作者: LaruaChen 05.01.2020