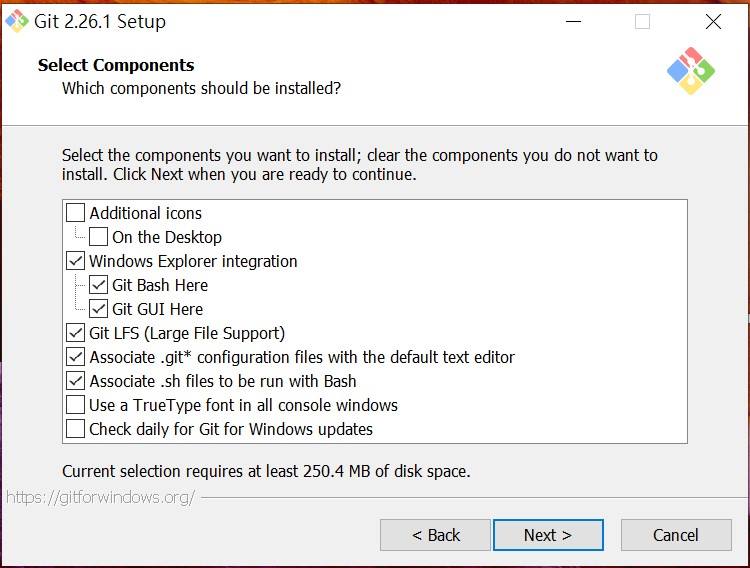
下載軟體(先載scm，再載fork)：

https://git-scm.com/

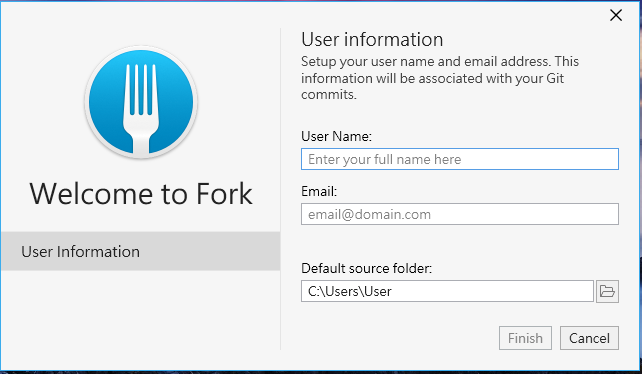
https://git-fork.com/ (版本為1.48.0.0)



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:

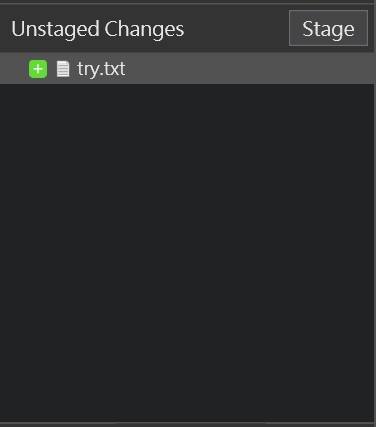
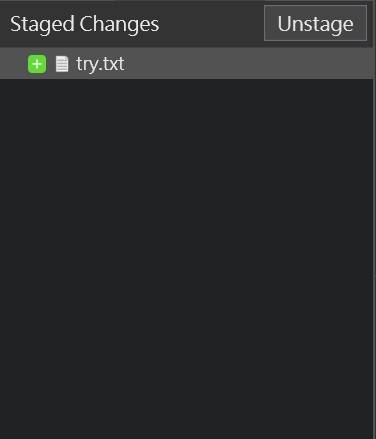
桌面建立資料夾(資料夾名為test) -> 在資料夾中建立txt檔(檔名為try)

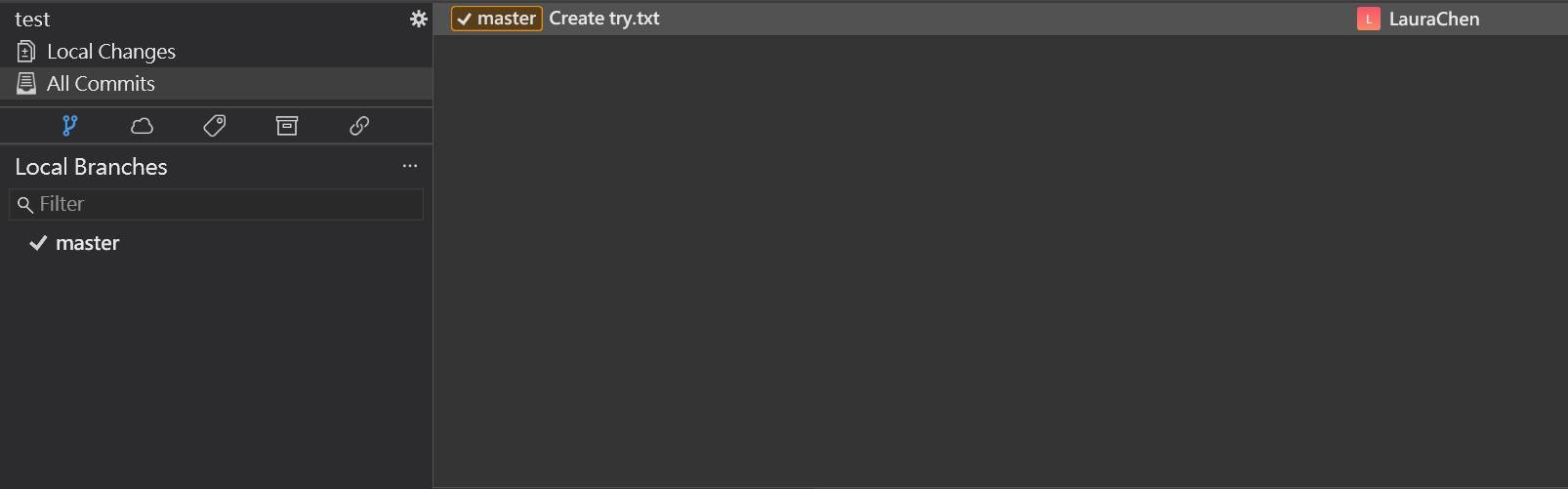
補充：

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



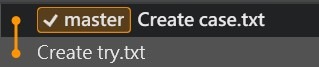


快照：其意思同拍照般可以凍結當時的一瞬間，對著資料庫拍照，就是對資料庫的某一個時間點，凍結記錄內容並完整的紀錄下來

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

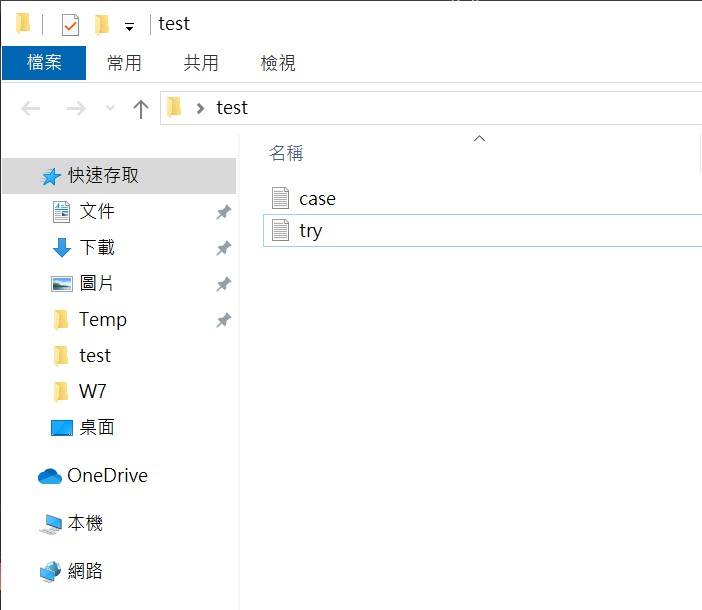
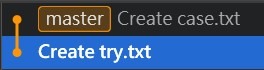
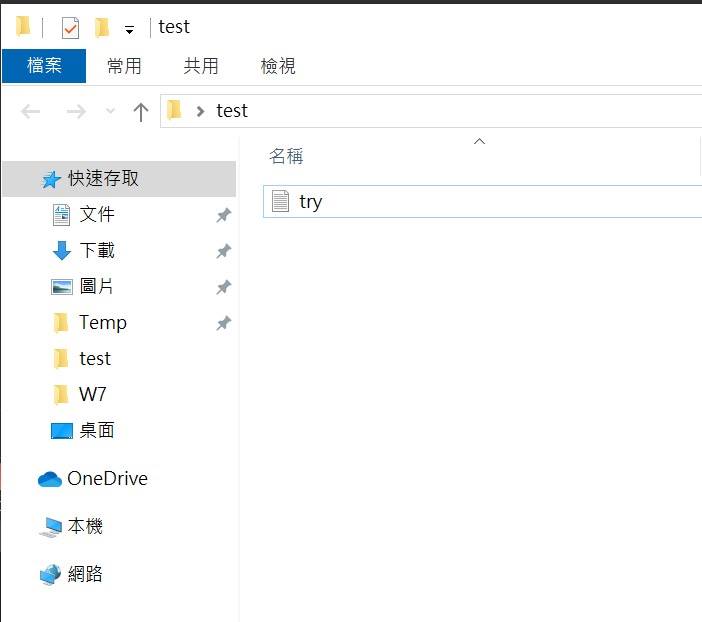


圖2



* 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。

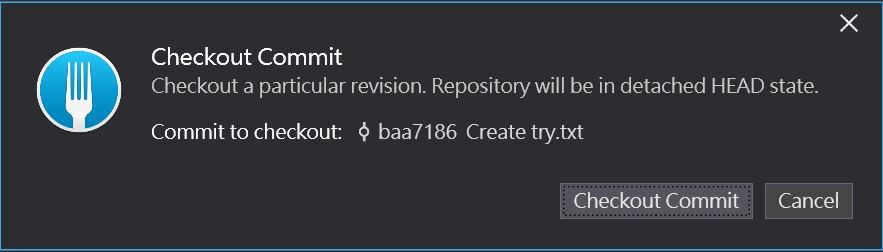
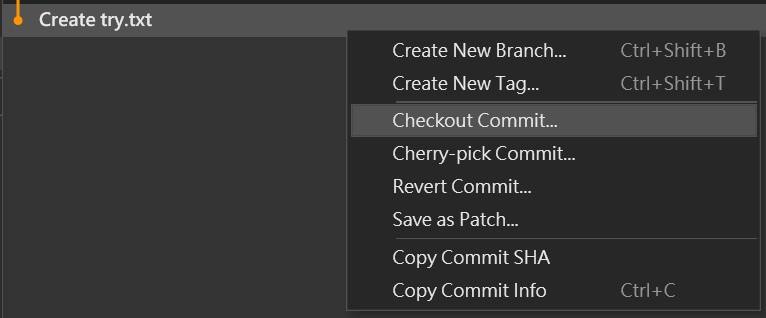
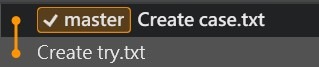
若要返回在All Commits中Create case.txt點兩下即可回復如圖2。

圖5

圖6

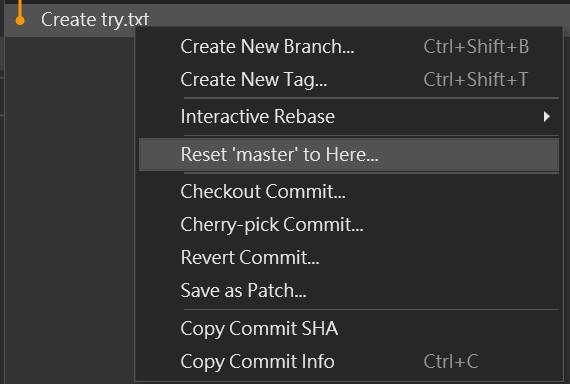
刪除節點紀錄(reset)：選擇要回復的地方按右鍵Reset 'master' to here -> 選擇要刪除的選項(避免不要，除非有把握)

預設情況：刪除case.txt，未操作前為圖7

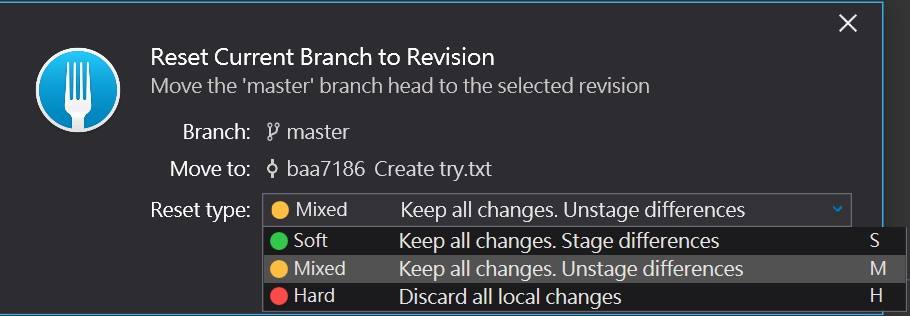


* 圖7

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)

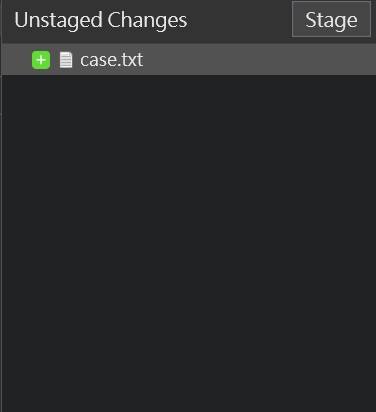


* 圖8

圖9



* 圖10



* 圖11

補充：

soft:紀錄到Change staged stage處

Mixed:記錄到Change Unstaged stage處

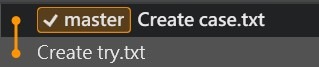
hard:完全刪除

補充：

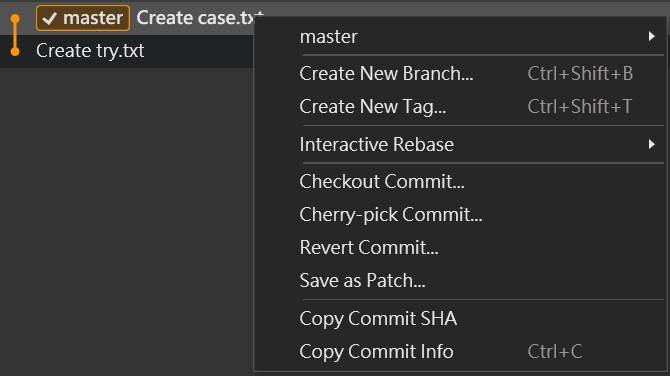
當有跟其他人共同使用一樣的資料夾，怕reset檔案救不回來，所以用revert可以做類似備份的動作，就算之後要使用也好回復，不會有其他問題(revert意思為取消這個節點所做的動作) 「建議都用revert」

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



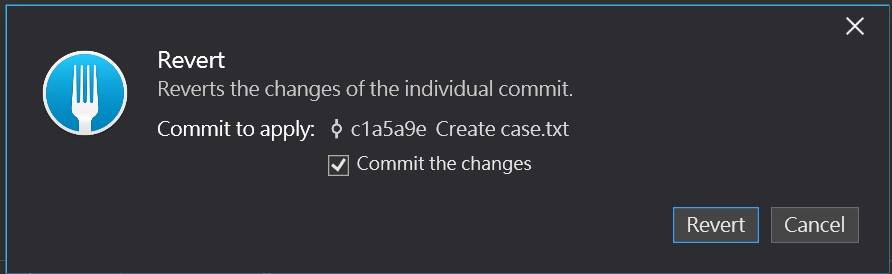
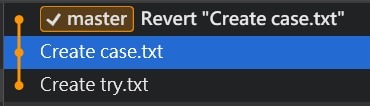
圖13

圖14



* 圖15

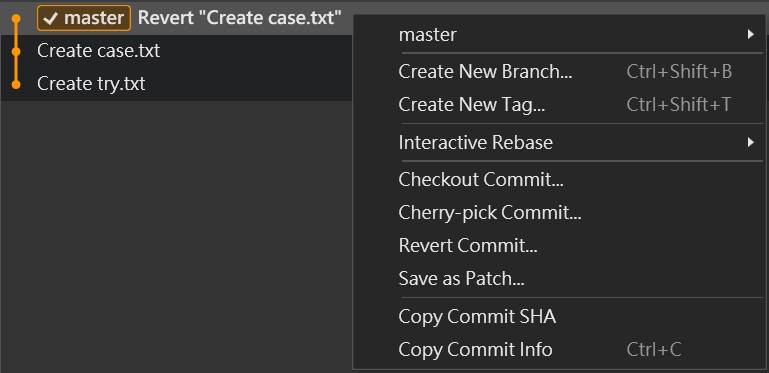
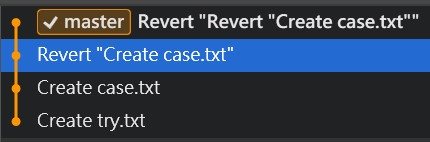


圖16

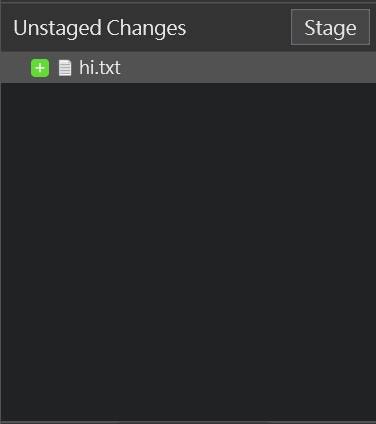


* 圖17

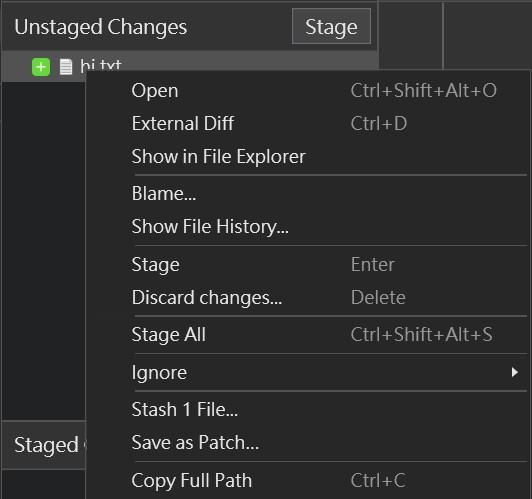
在資料夾中建立(或輸入)錯誤的檔案刪除：在Change Unstaged stage選擇要回復的檔案 -> 右鍵選Discard Changes -> 選Discard -> done

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

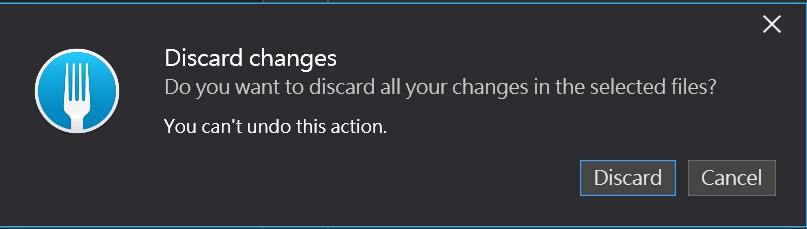


圖20

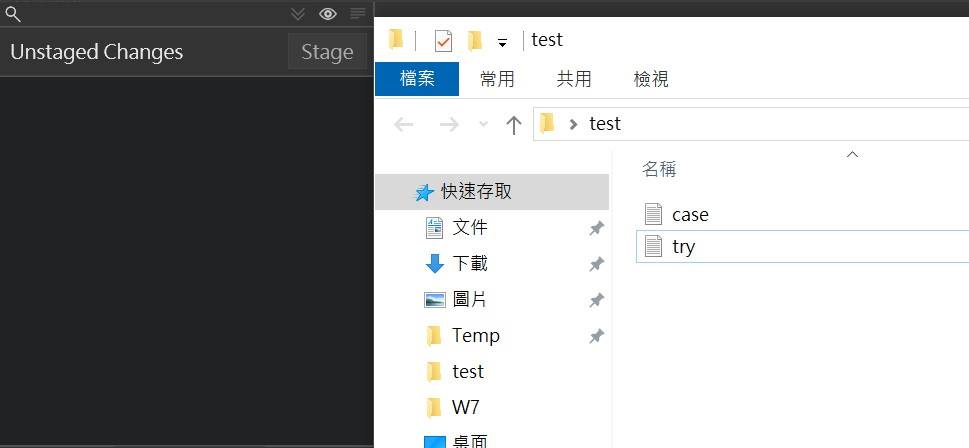


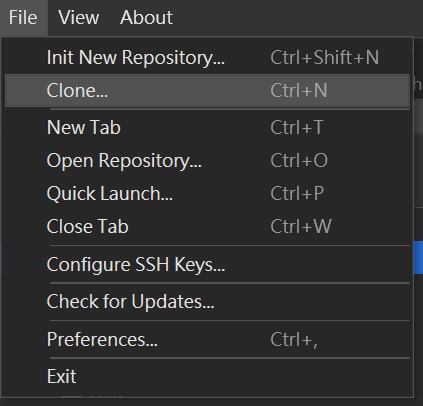
圖21

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

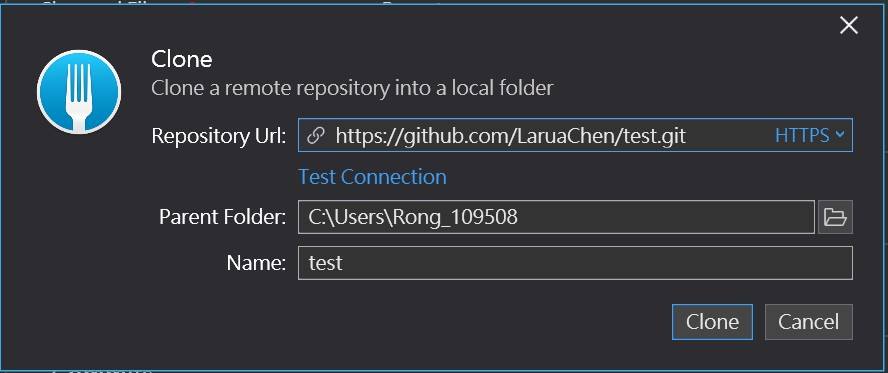
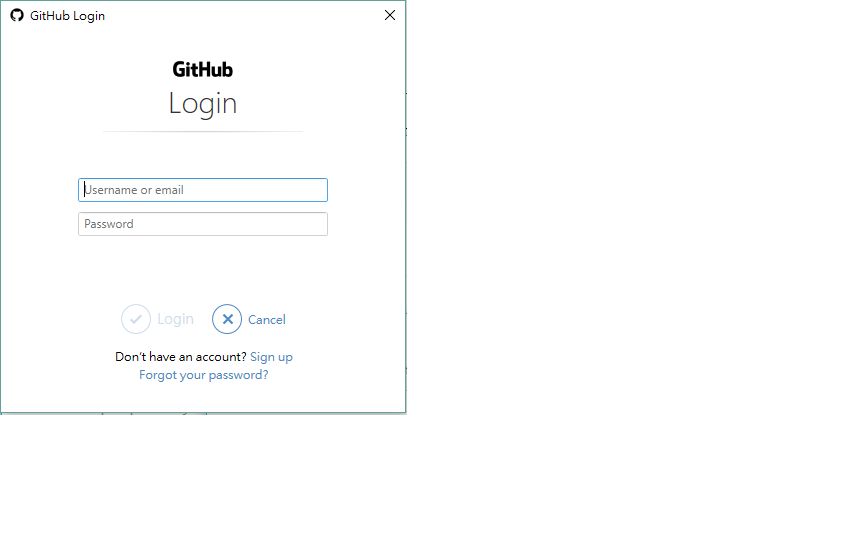


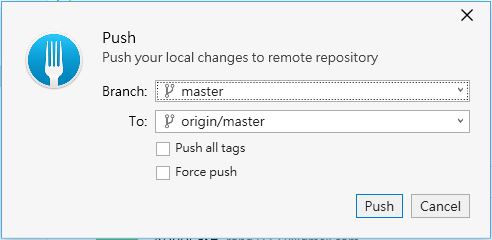
圖23



* 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

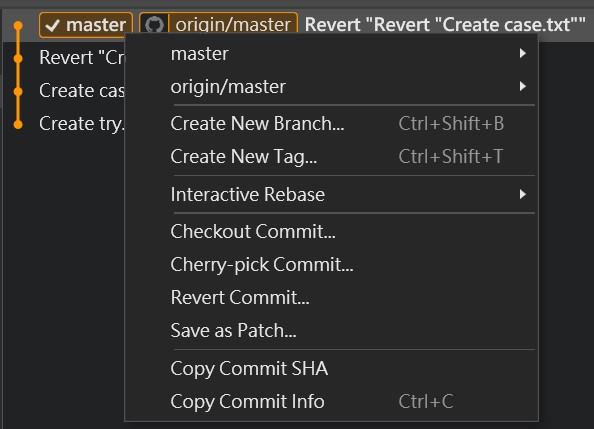


->圖25

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

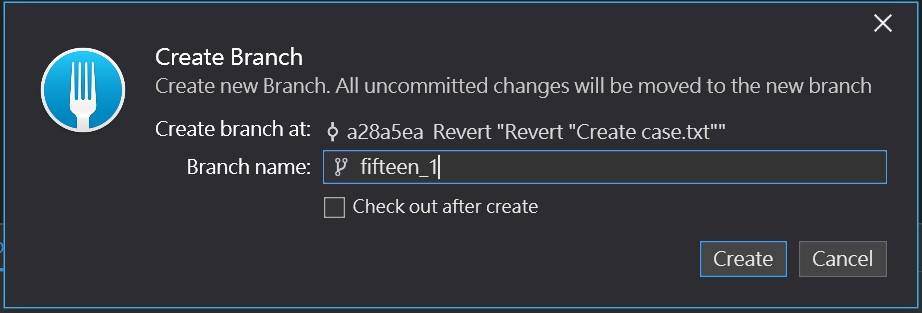


圖27

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

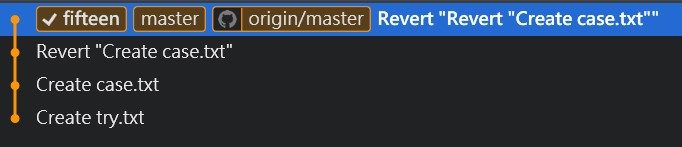
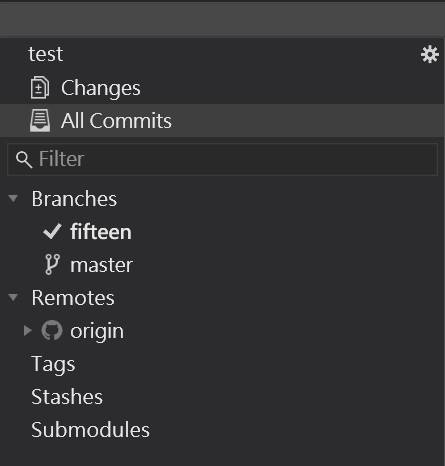


圖28



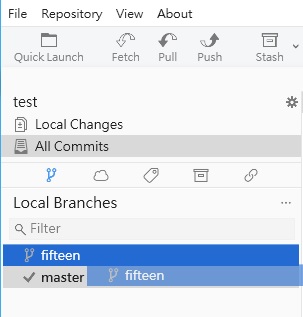
->圖29

補充：

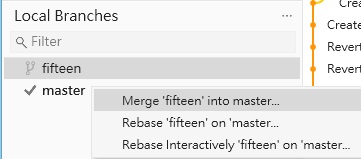
多人在共用同個資料夾存放資料時，分支主要是在區分不同的功能，或是用來測試功能，確保master是可以的檔案是可以使用的。若有多人同時使用同個檔案修改上傳到github並合併到master會產生衝突，後面有解決方法

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):

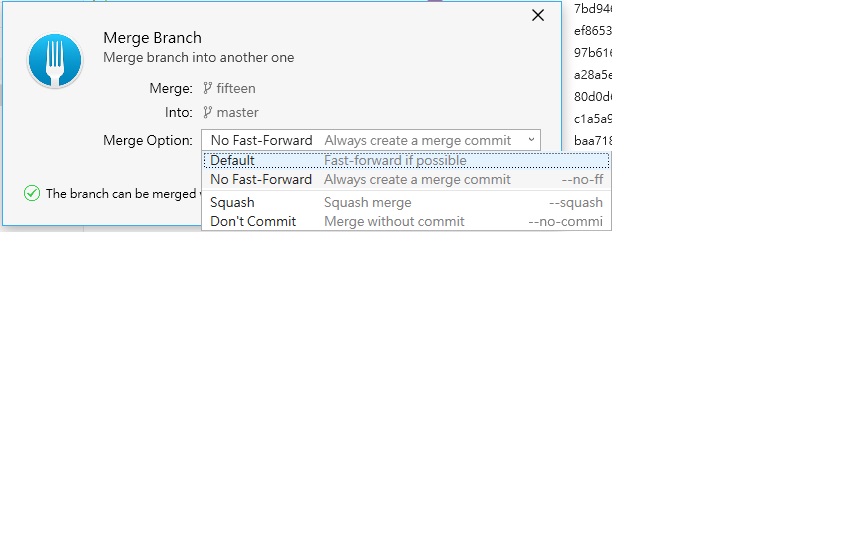
將分支按著不放拖曳在master上(圖30) -> merge '分支名稱' into master(圖31) -> 選No Fast-Forward(圖32) -> 按merge(圖33)



->圖30

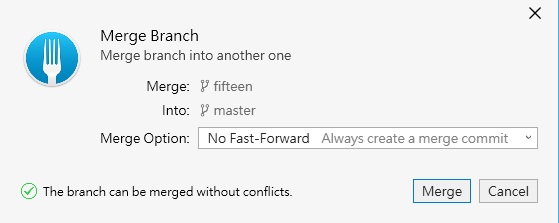


->圖31



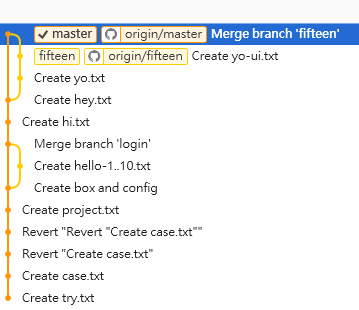
->圖32

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

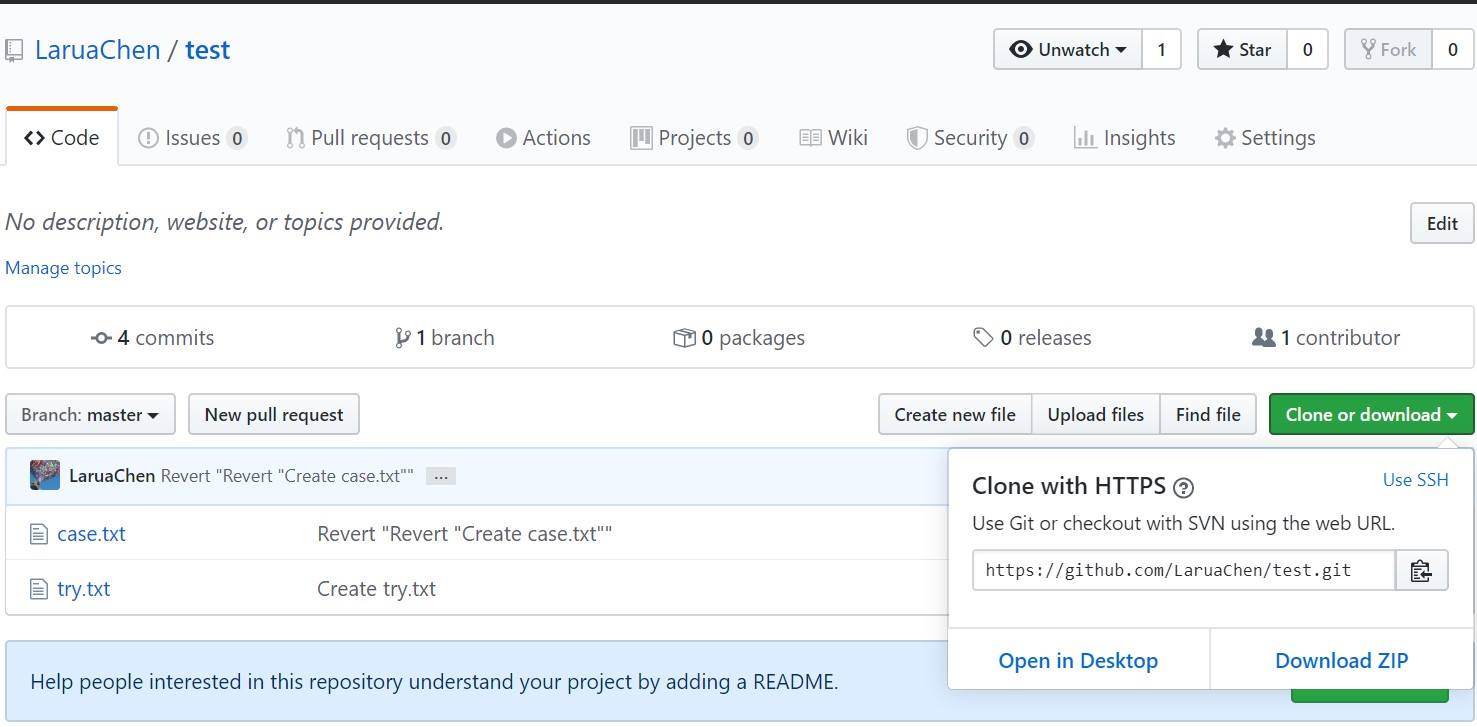
將雲端檔案下載回本地端步驟：

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)



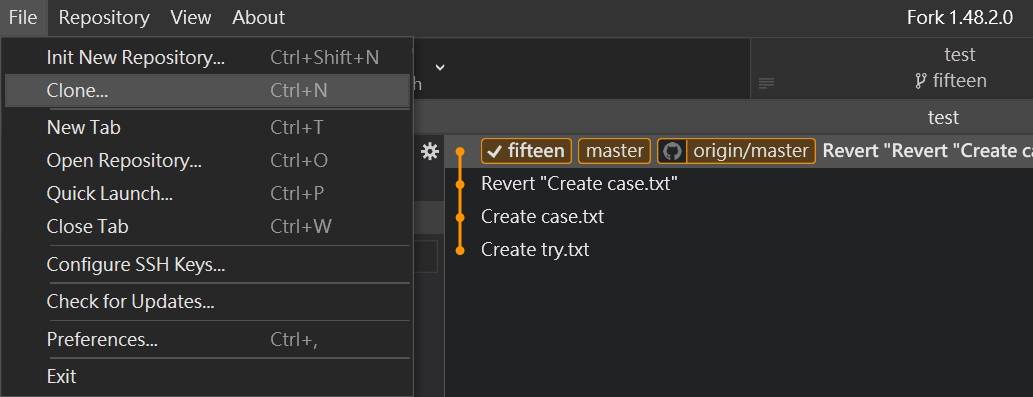
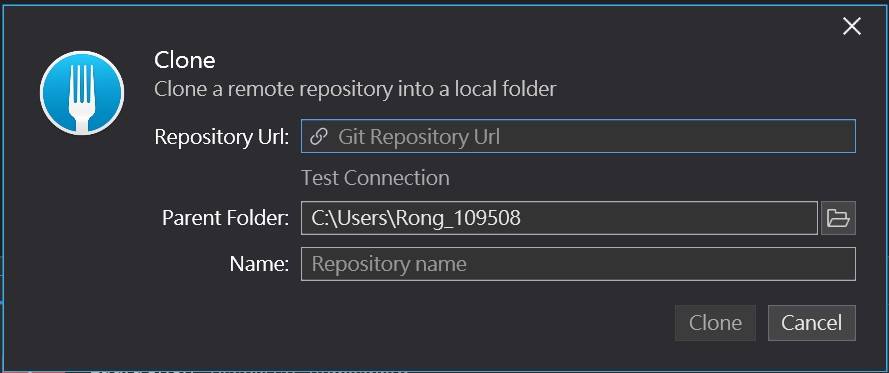
圖35

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) -> compase 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)

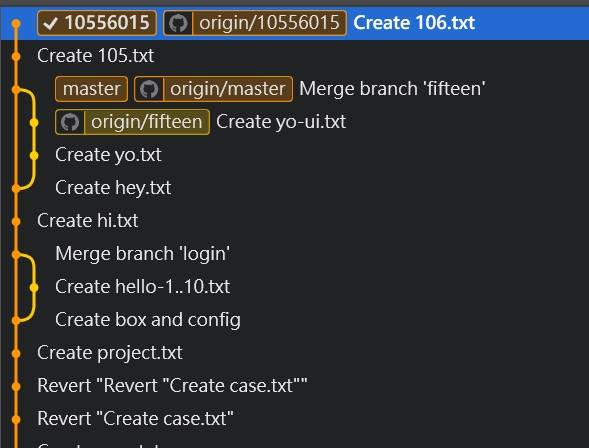
假設情況：要將10556015分支及檔案合併到master分支

圖38

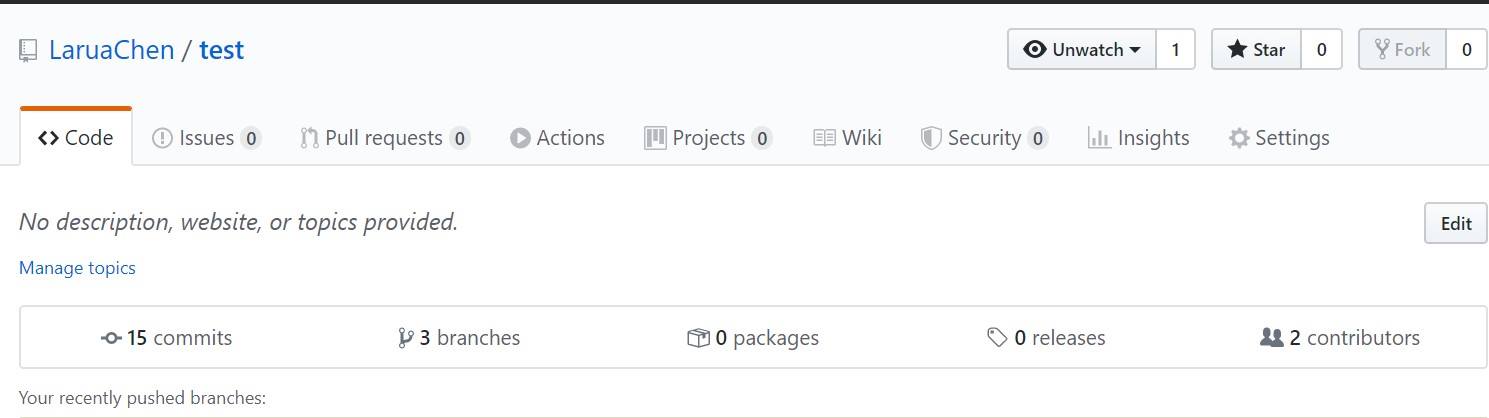


圖39

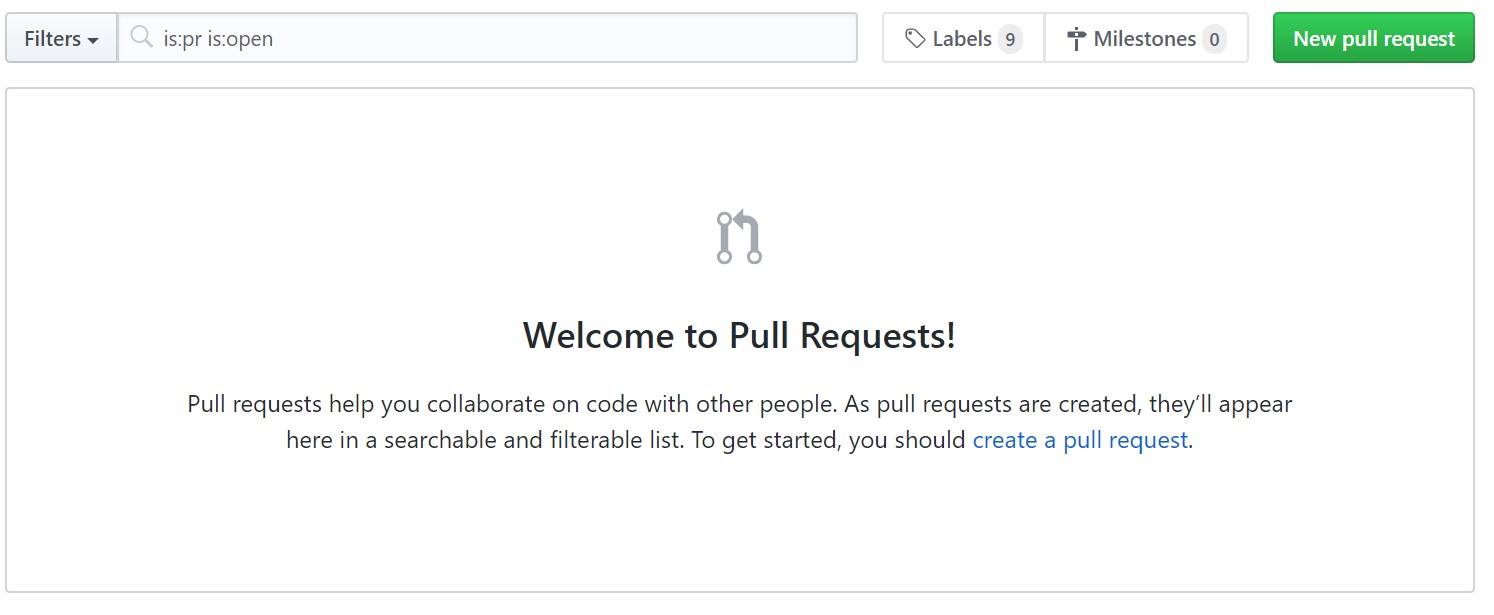


圖40

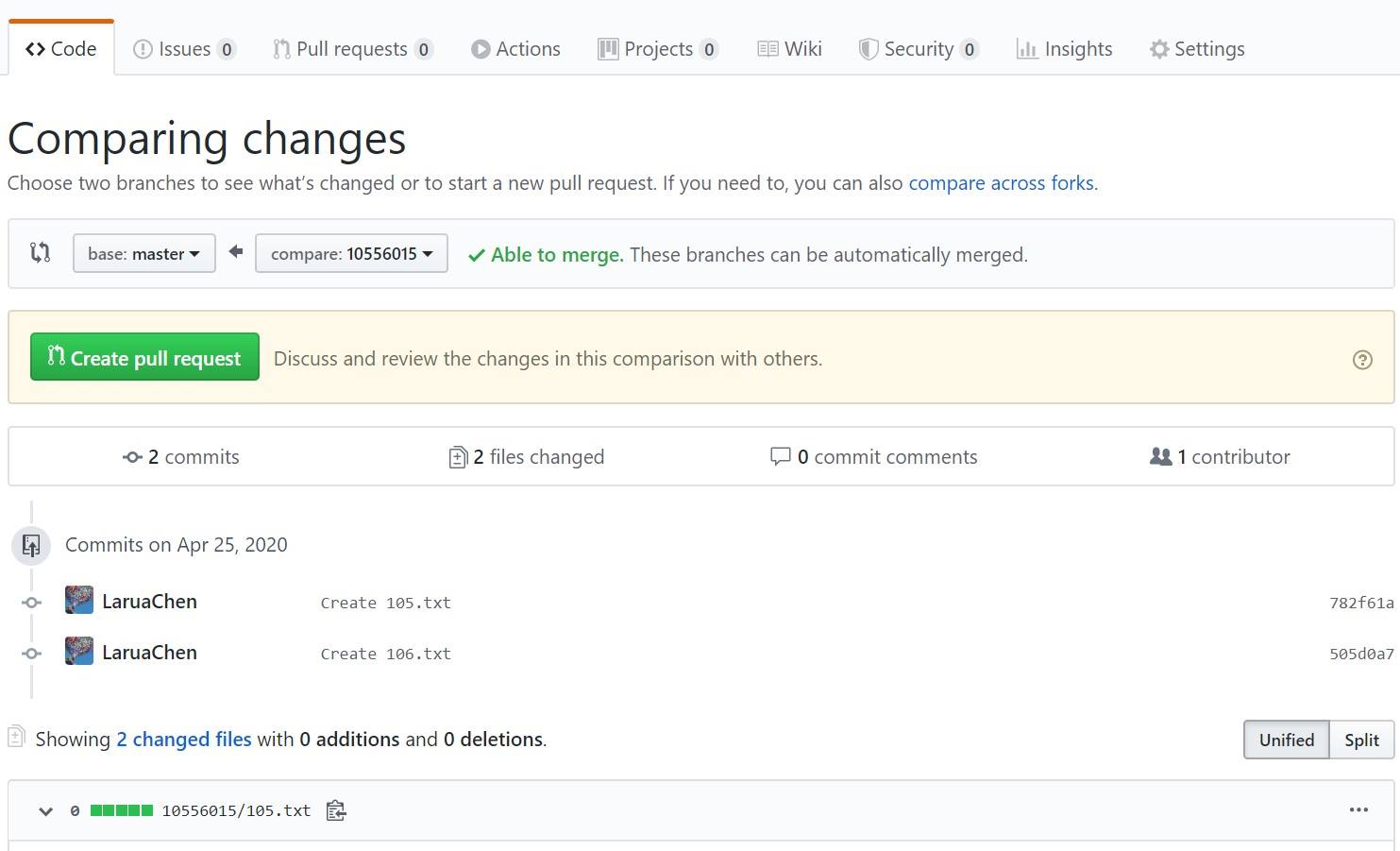
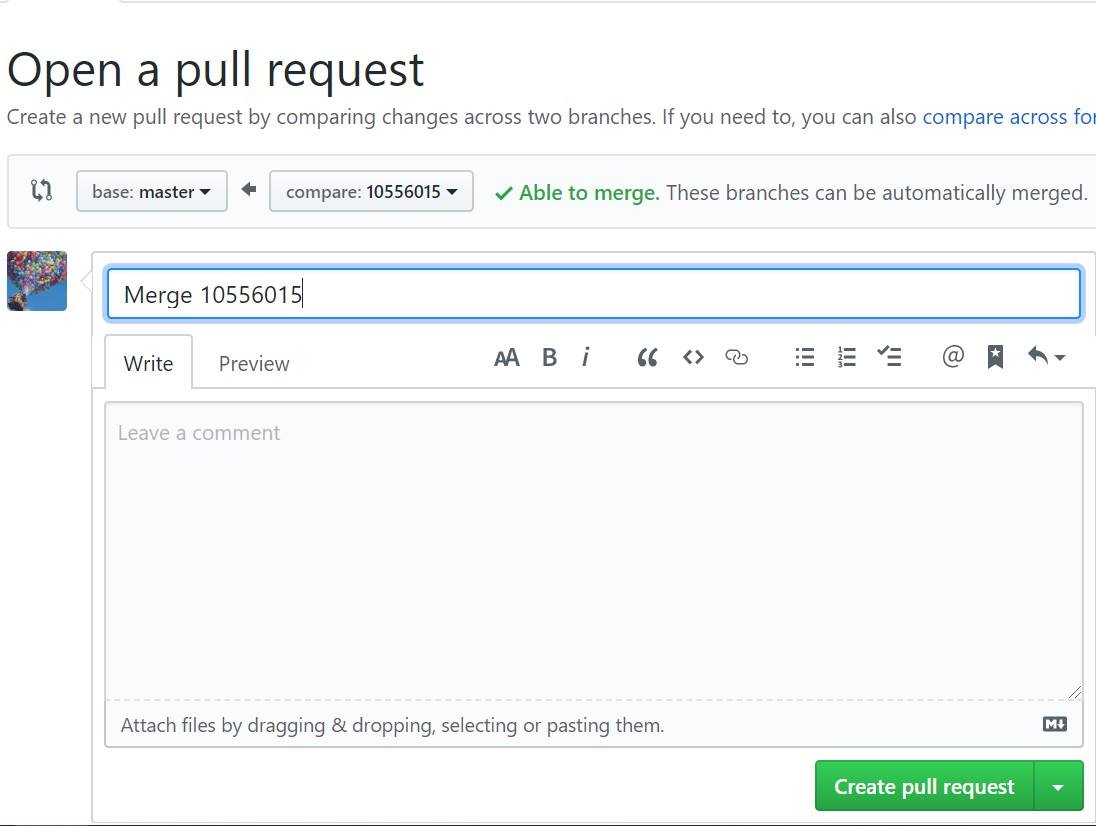
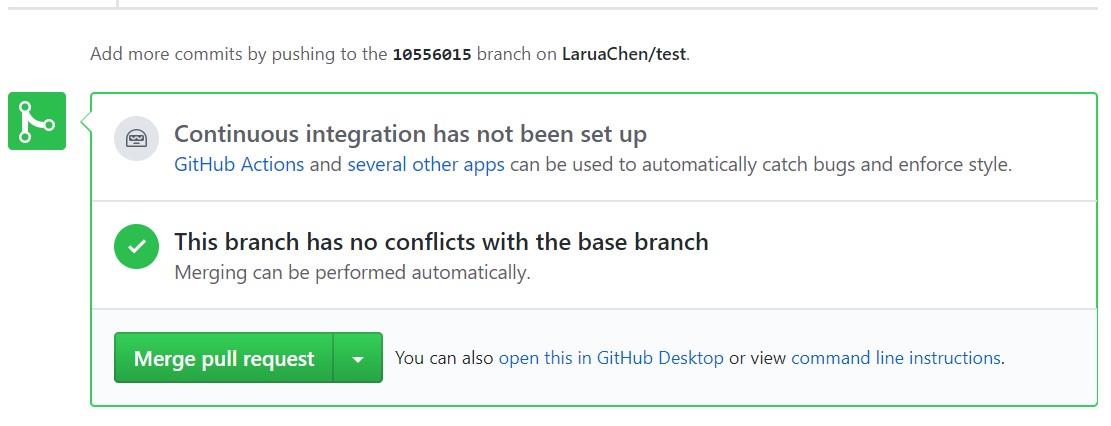


圖41

圖42



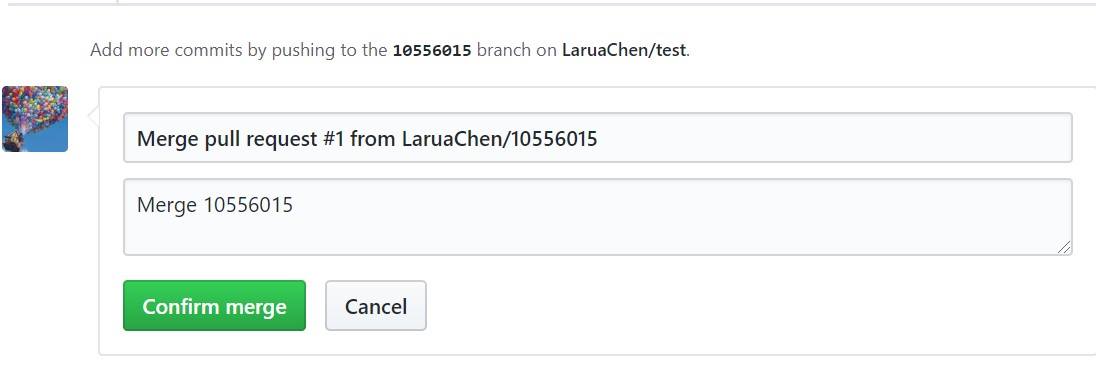
圖43

圖44

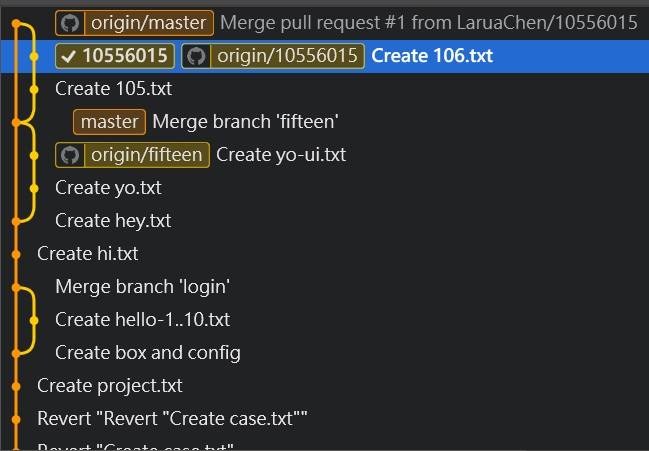


圖45(這圖是10556015分支合併到master分支之後的狀況)

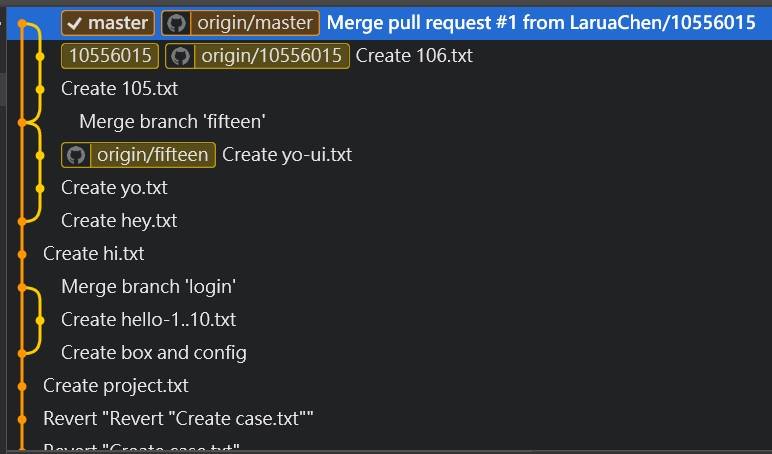
在圖45中若要把本地端的master更新到與雲端上的(origin master)相同紀錄，在fork介面要選擇本地端的master並按下Pull就能成功。(圖46)

圖46

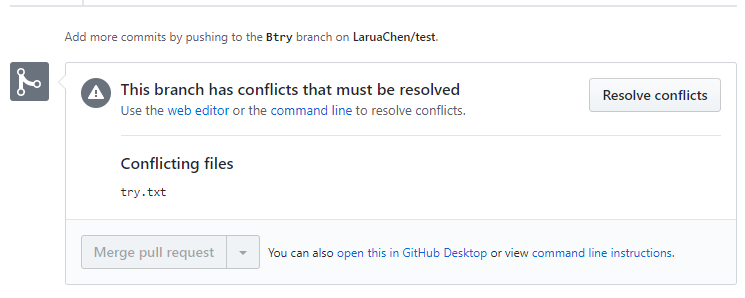
解決兩人上傳相同檔案的衝突：

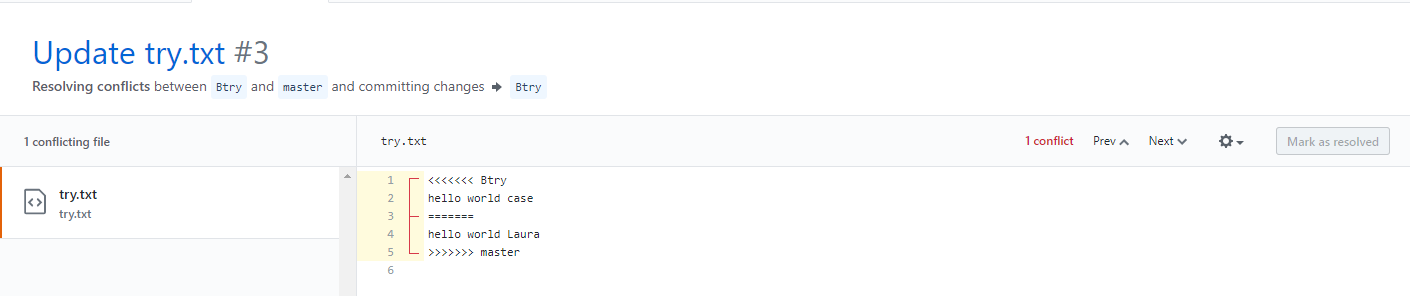
假設情況：在同個分支下更動同個檔案，並且有同時兩人在自己建立的分支上傳同個檔案到github，有一位已經合併好到master裡，以至於另外一位不能合併到master裡

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)

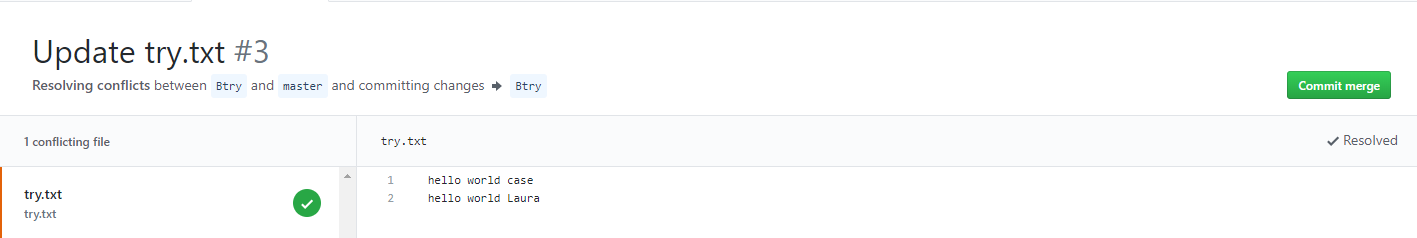
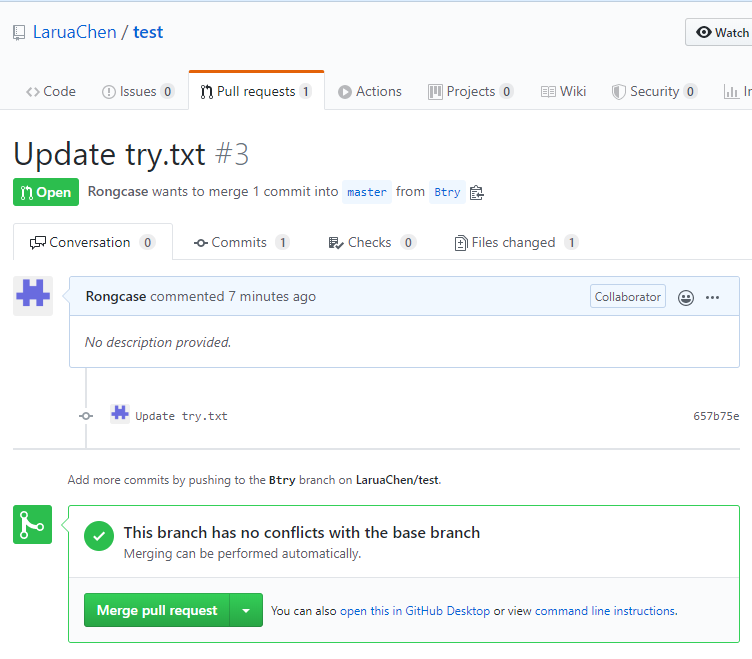
圖48(紅框處必須刪除，其餘的看要留下A或是B或者都留下，此例情況先都留下)

圖49

圖50

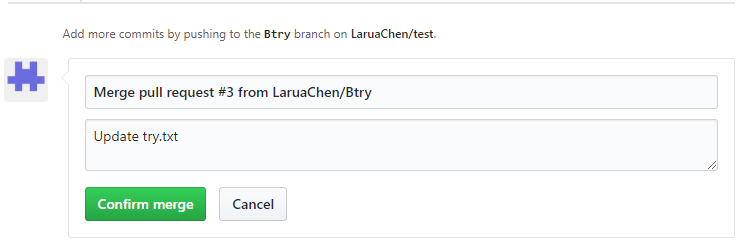
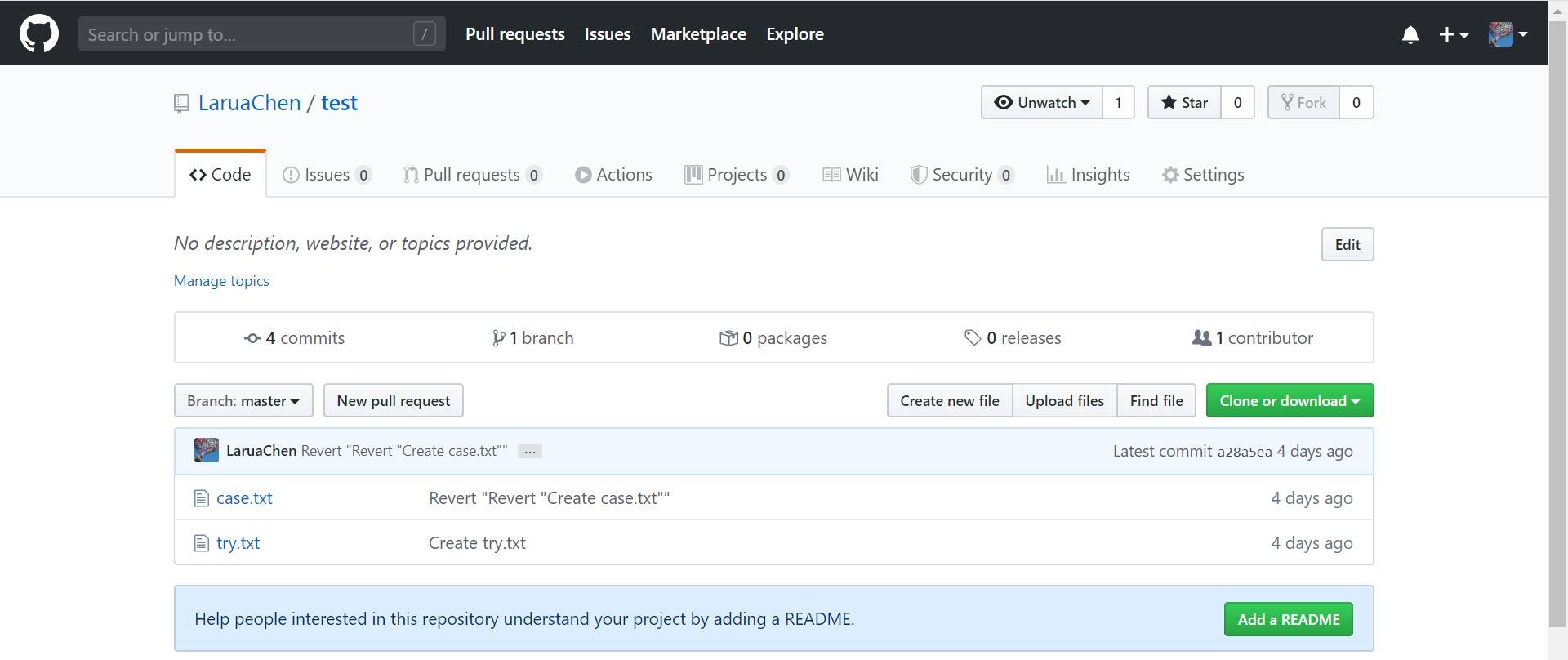
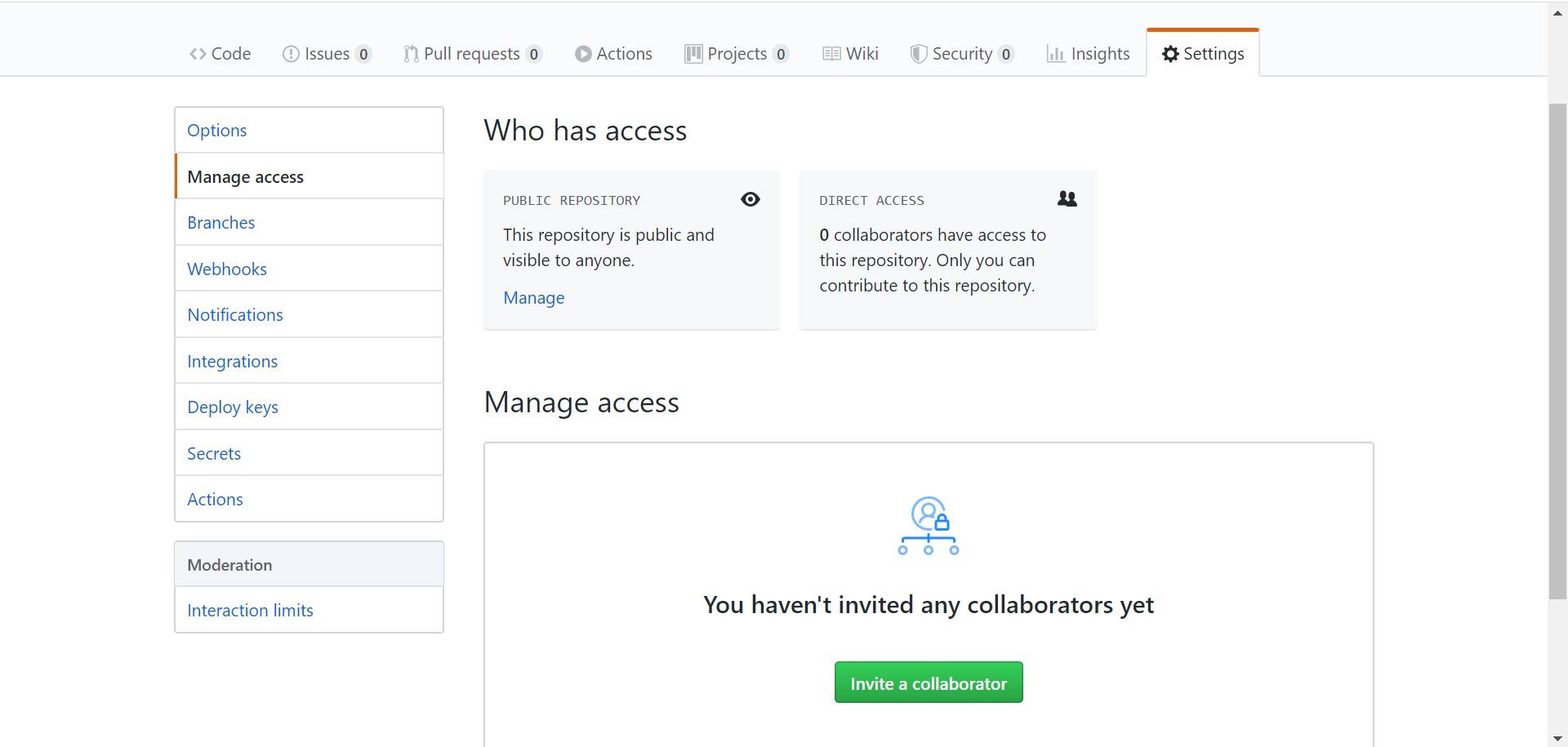


圖51

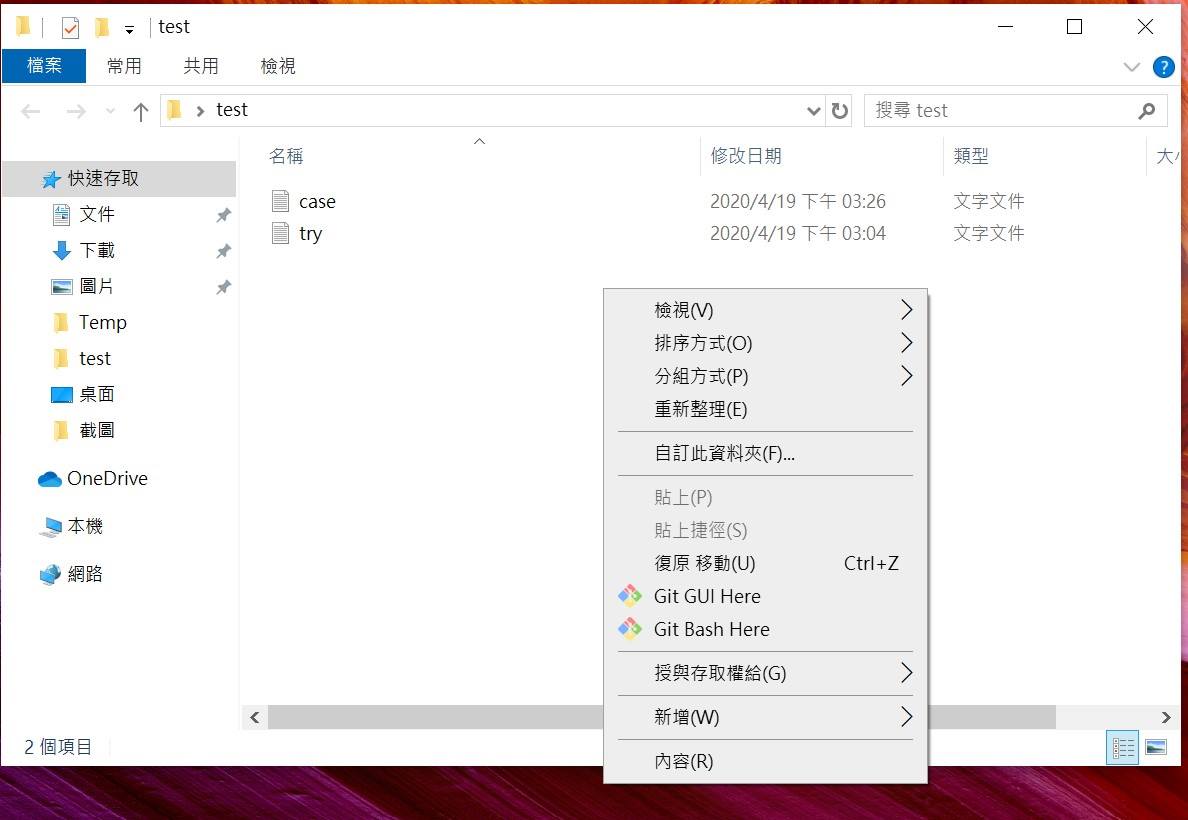
開協同合作:

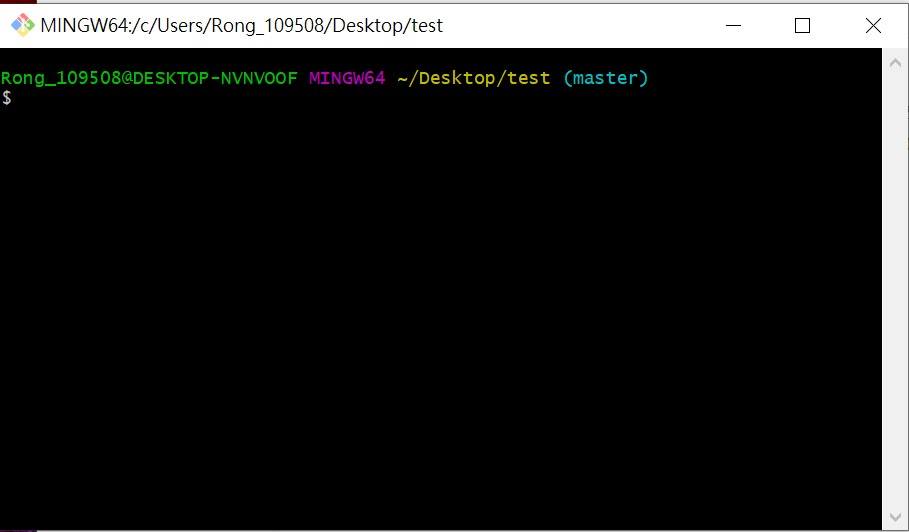
在github要共享的資料夾首頁按setting -> Manage access -> Invite a collaborators ->輸入要協同的使用者github名稱

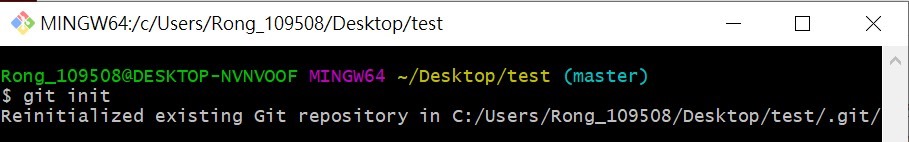
被邀請的人要在信箱裡面去確認看有沒有邀請信

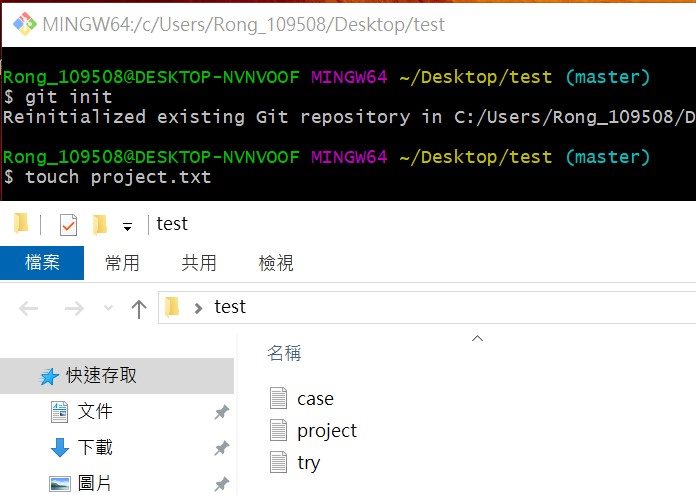
Github 指令：

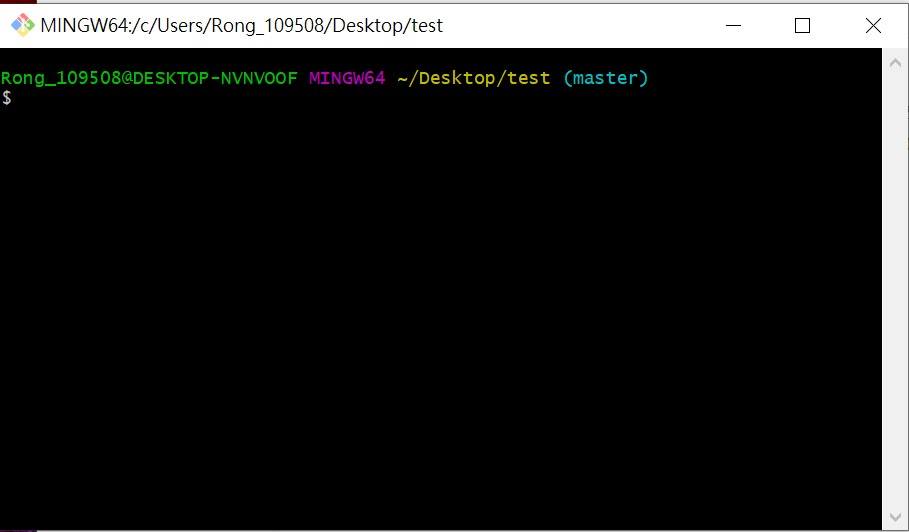
指令網址：http://0102.tw/md

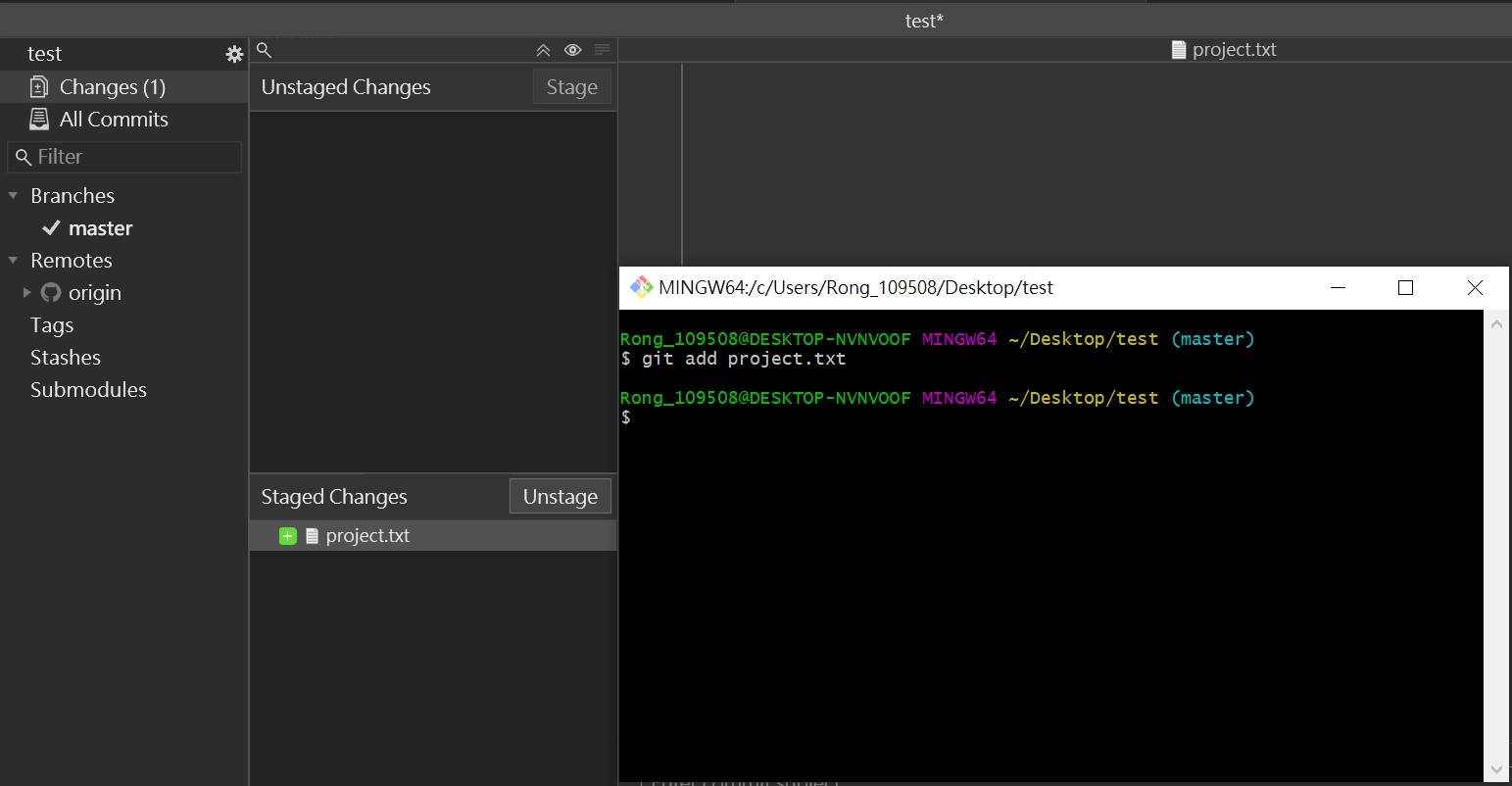
開啟指令視窗:在資料夾內按右鍵 -> git bash here

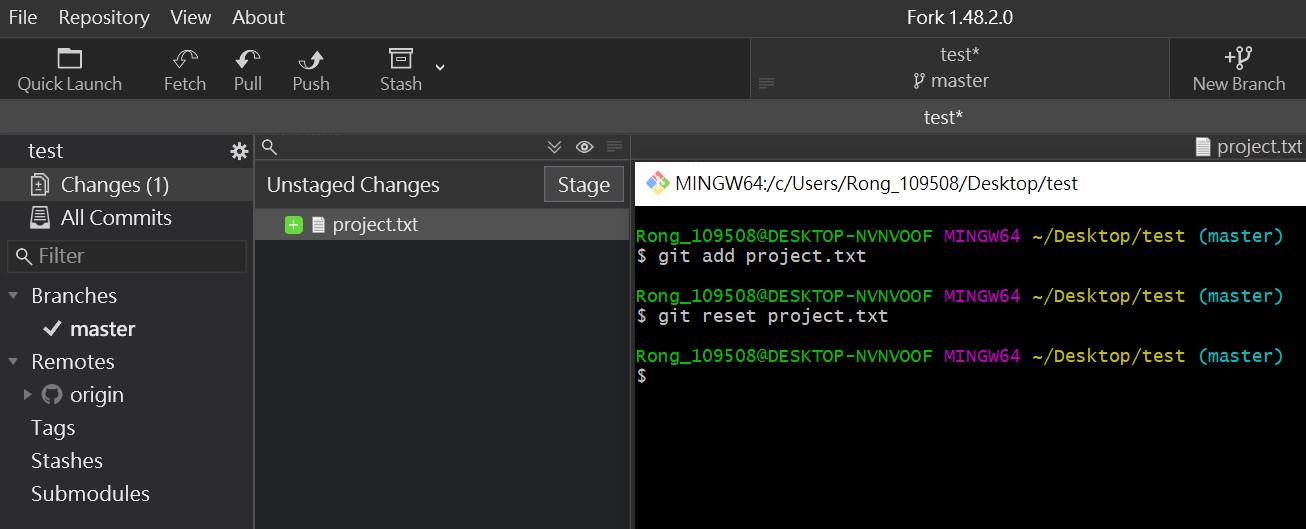


git init:開啟資料夾路徑

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

ctrl + L：清空所有指令

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)

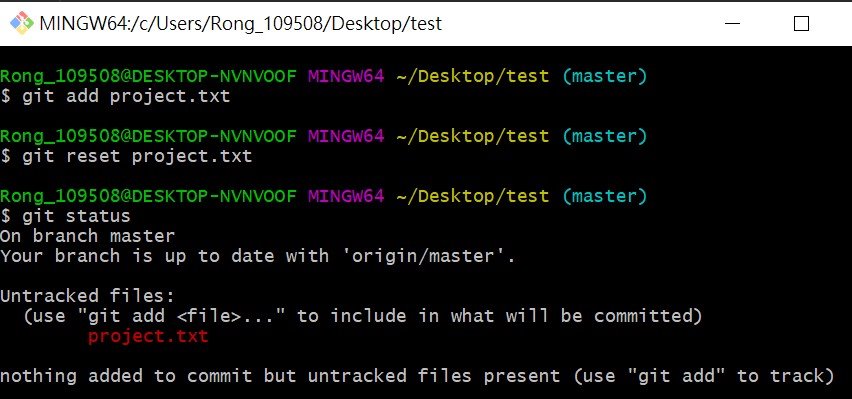
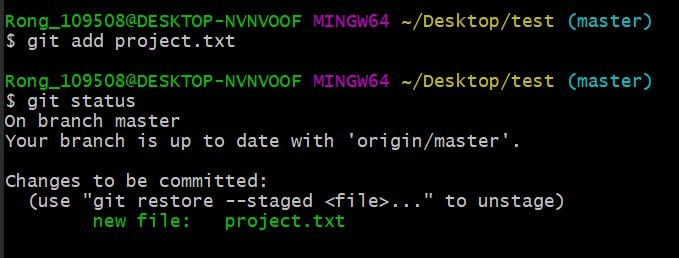
git reset project.txt: 跑到UnStaged Changes

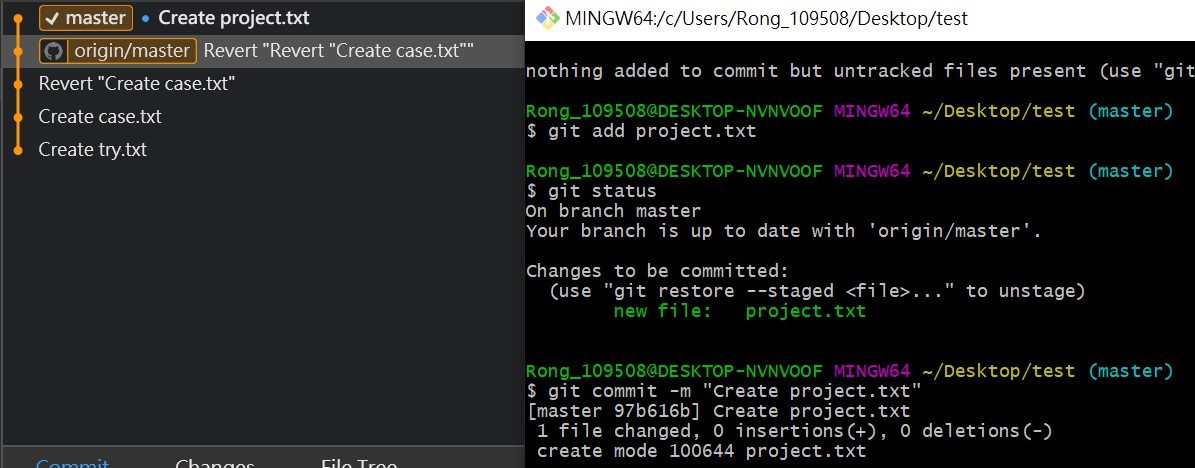
補充：

add選取一個檔案

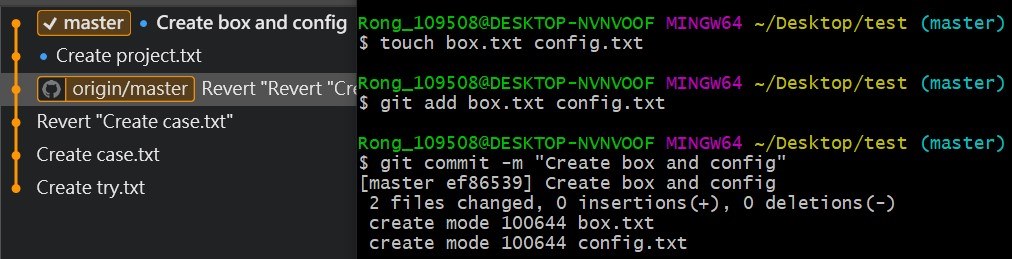
Create建立新檔案

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

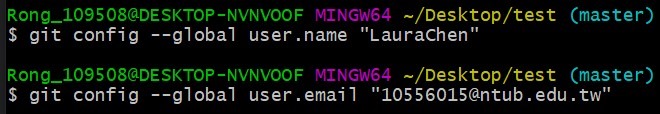
git commit -m "Create project":下節點

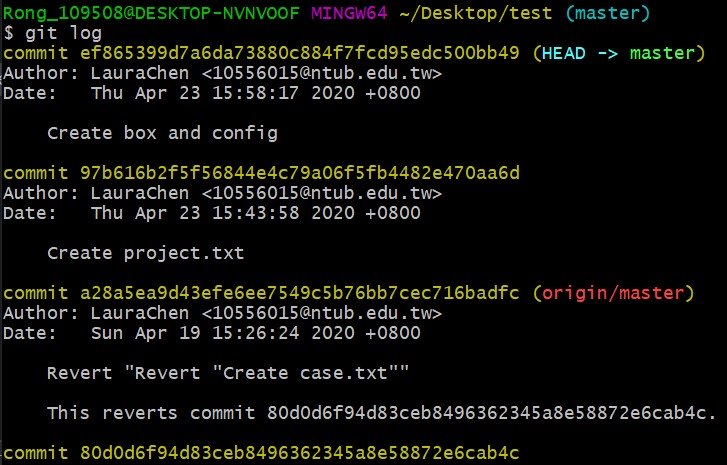
git commit -m "Create box and config":下節點(同時選取box和config檔)

以下的圖是新增兩個檔案去建立節點的流程

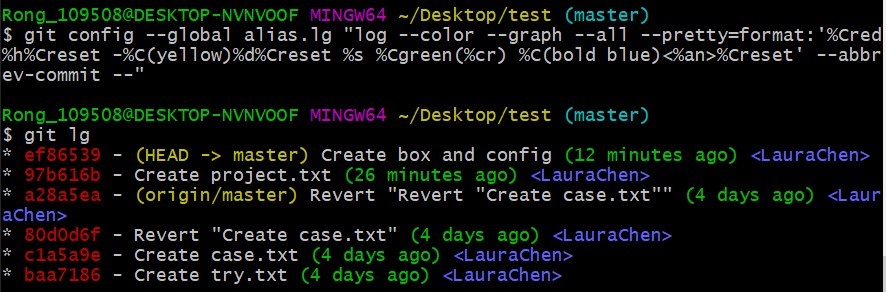
輸入你的github user name 和github email：

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

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

git log:查詢節點紀錄

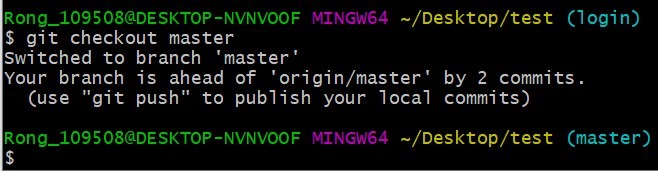
git config --global alias.lg "log --color --graph --all --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit --"

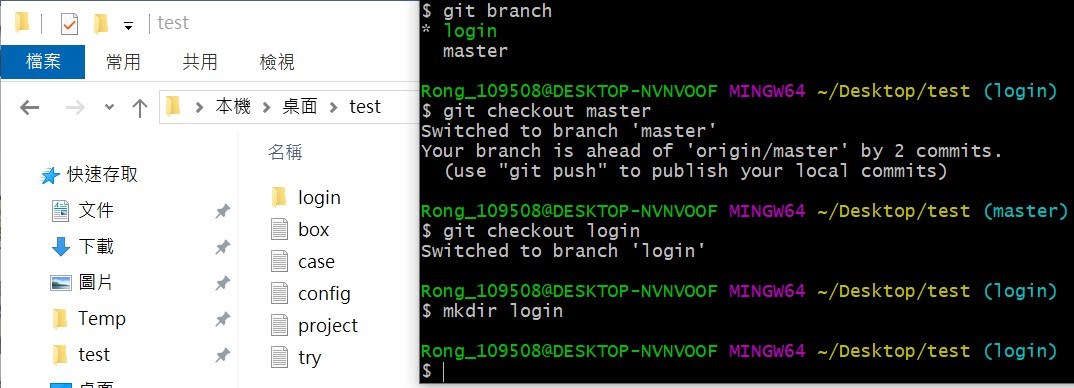
git lg:查詢節點紀錄，格式比較好看(前提是要先輸入「git config --global alias.lg "log --color --graph --all --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit --"」)

新增「login」分支: git checkout -b 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

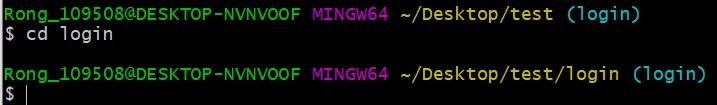
選擇「master」分支: git checkout master

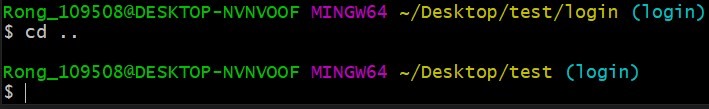
新增「login」資料夾: mkdir login

指令介面：

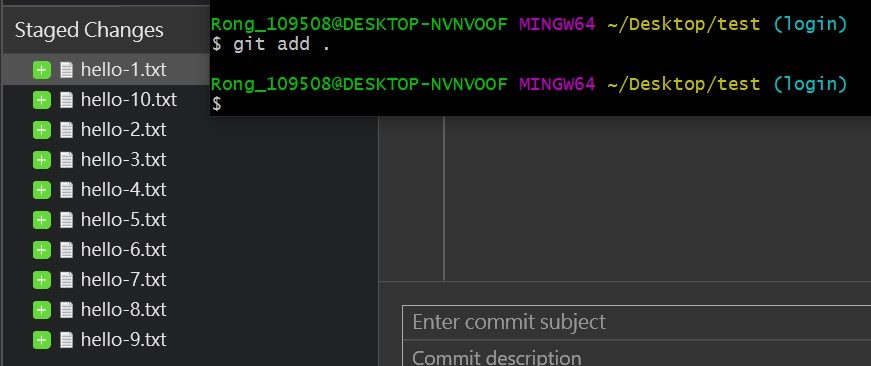
ctrl + insert：複製(在指令介面上按右鍵也可以)

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)

cd .. 返回上一層(d和.之間要加空白)

touch hello-{1..10}.txt 建立十個檔案

git add . 選取全部資料(d和.之間要加空白)

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