

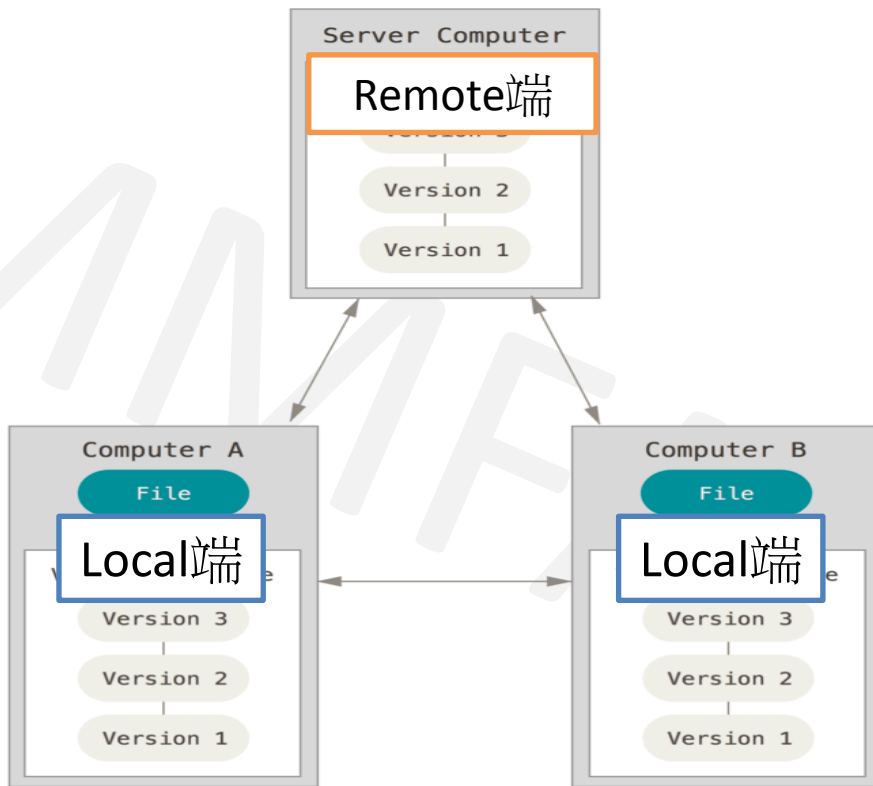
A blue triangle graphic is located on the left side of the slide, pointing towards the right. It is a solid blue color with a slight gradient.

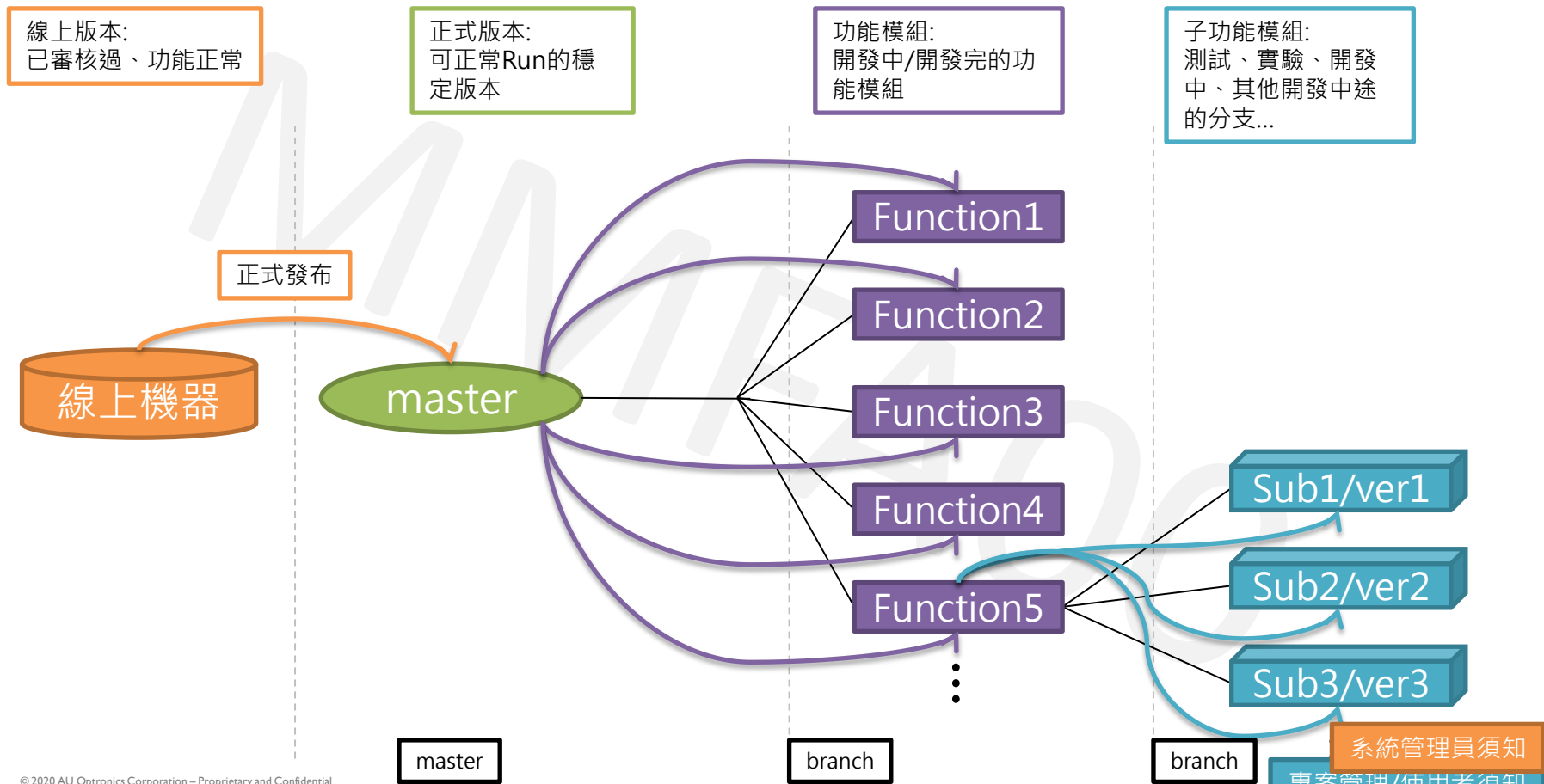
版本控管 使用說明

用Git追蹤你的程式碼

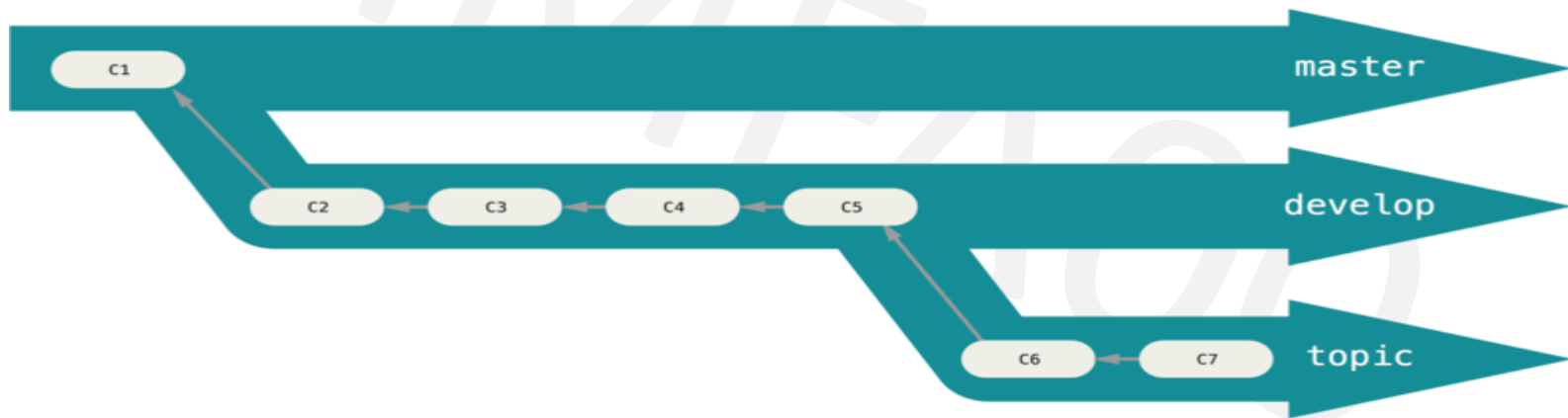
使用概念、架構

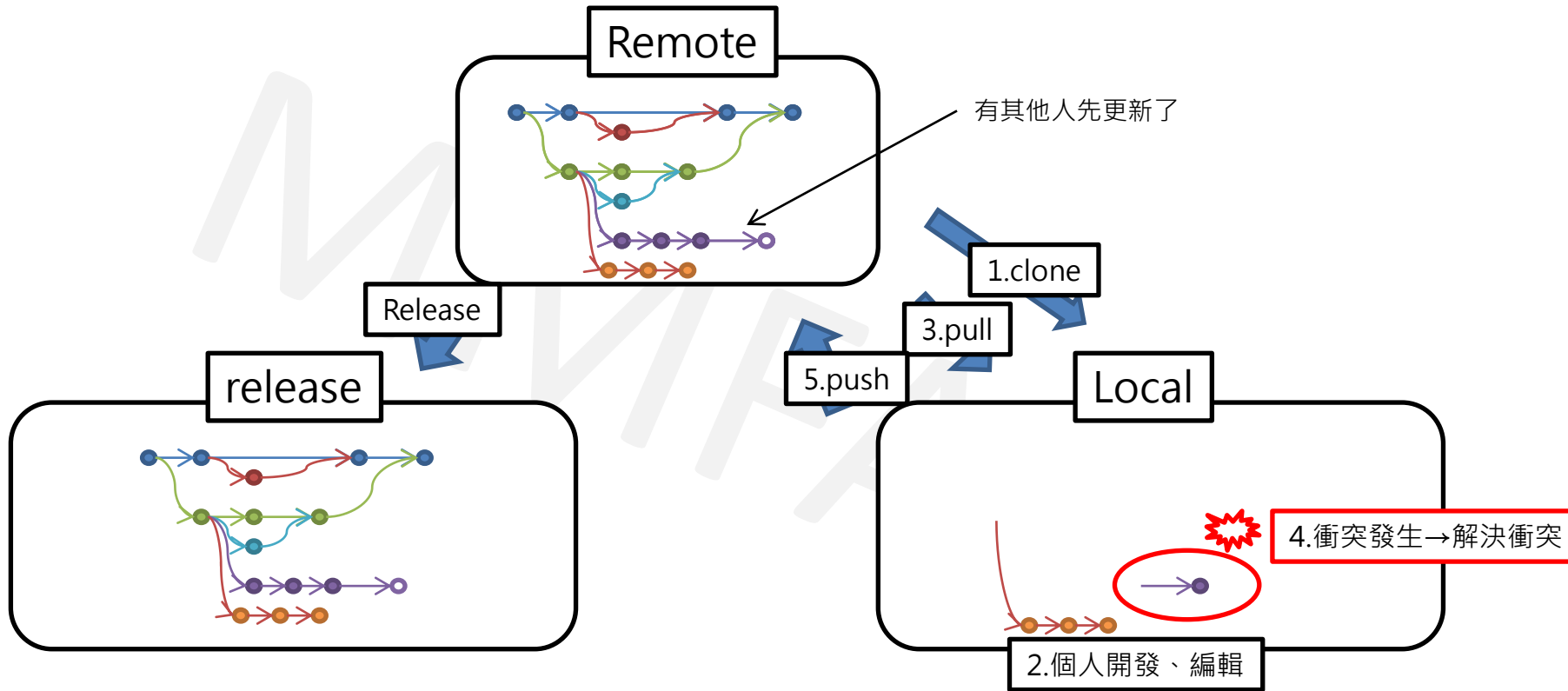
概念

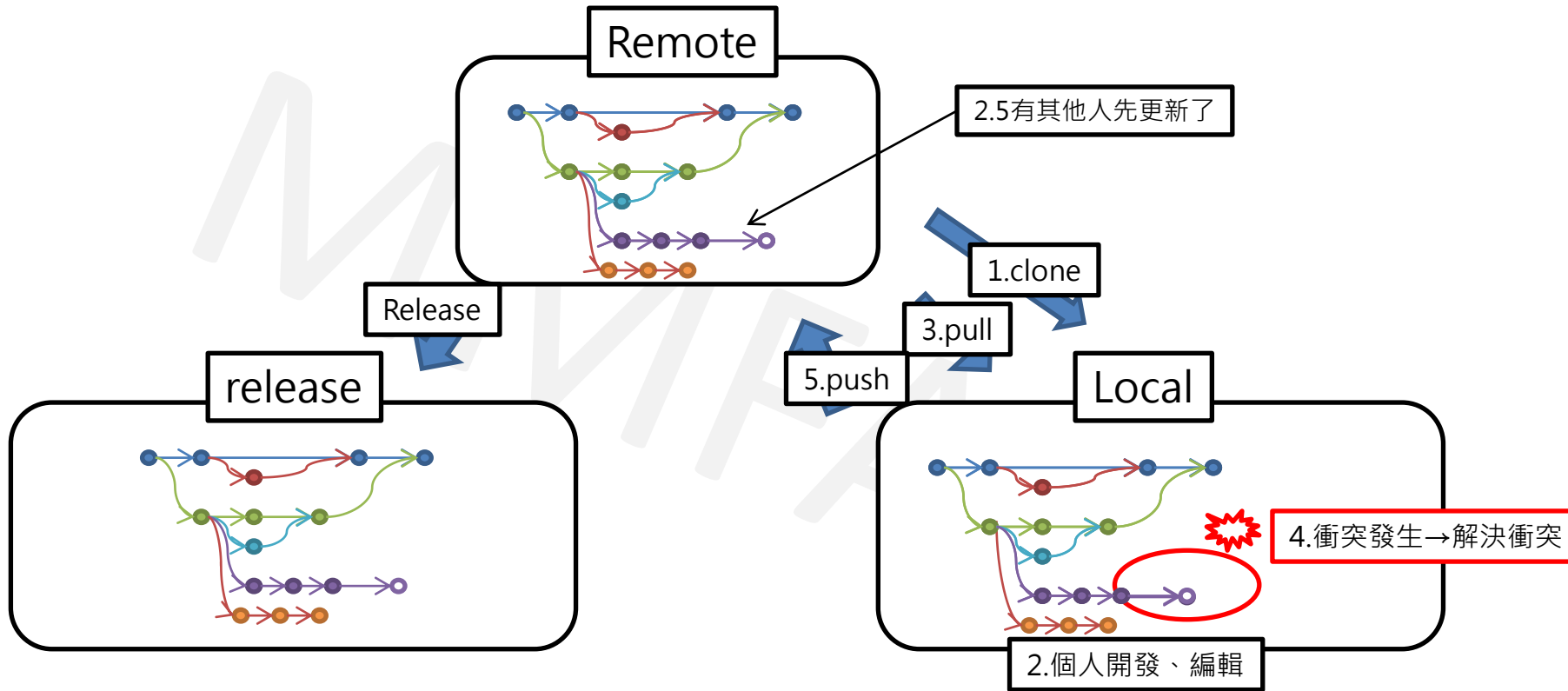




開發分支(Branch)







透過TurtoiseGit快速上手 (其他的軟體操作方式雷同)

操作

(第一次)從已知專案獲取專案檔案

Step1. 獲取你的專案連結 gitURL(http URL)

E.g. `http://tcaigitlab.corpnet.auo.com/pmo/**GroupName**/**ProjectName**.git`

Step2. 設定使用者資訊

E.g. `http://NT帳號:NT密碼@tcaigitlab.corpnet.auo.com/pmo/**GroupName**/**ProjectName**.git`

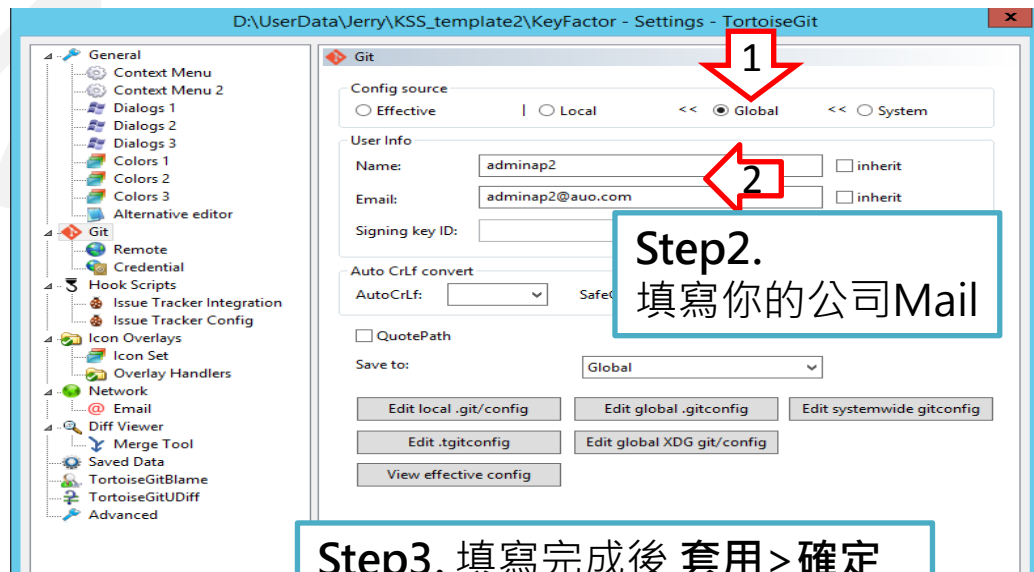
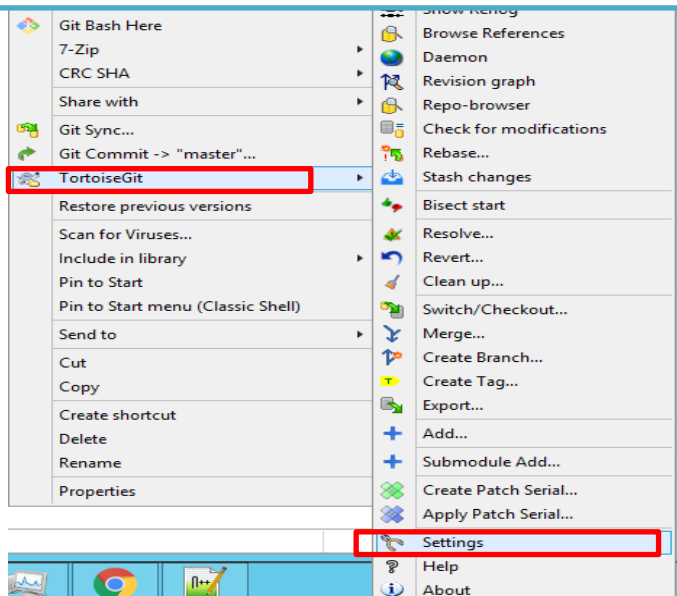
Step3. 打開Cmd視窗 (or Git Bash) , 並切換到要放至檔案的位置(路徑底下)

Step4. 鍵入(複製) 下方指令、貼上 Step3. gitURL , Enter執行

```
git clone http://NT帳號:NT密碼@tcaigitlab.corpnet.auo.com/pmo/**GroupName**/**ProjectName**.git
```

設定使用者Email

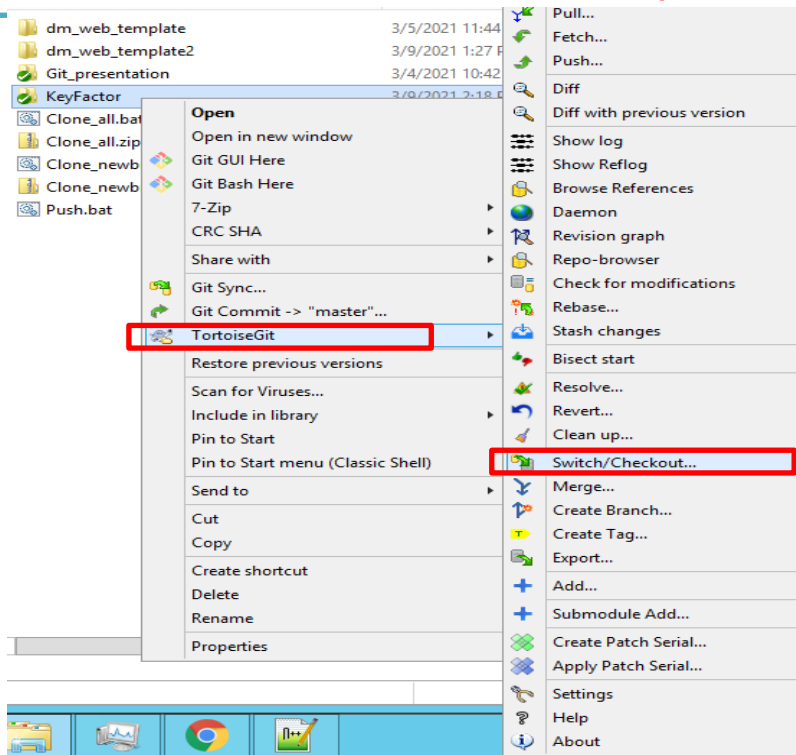
Step1. 對剛剛複製(Clone)下來的資料夾
滑鼠 **右鍵** >> **TortoiseGit** >> **Settings**



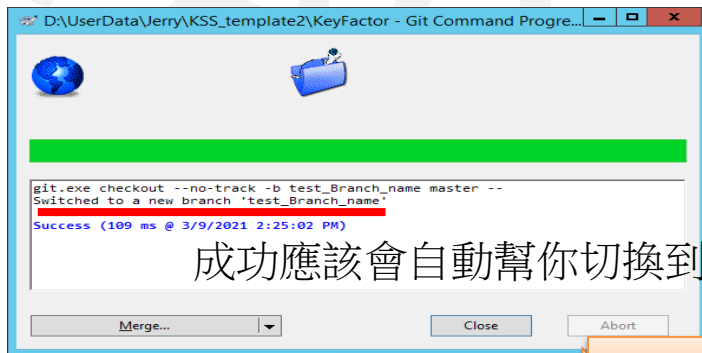
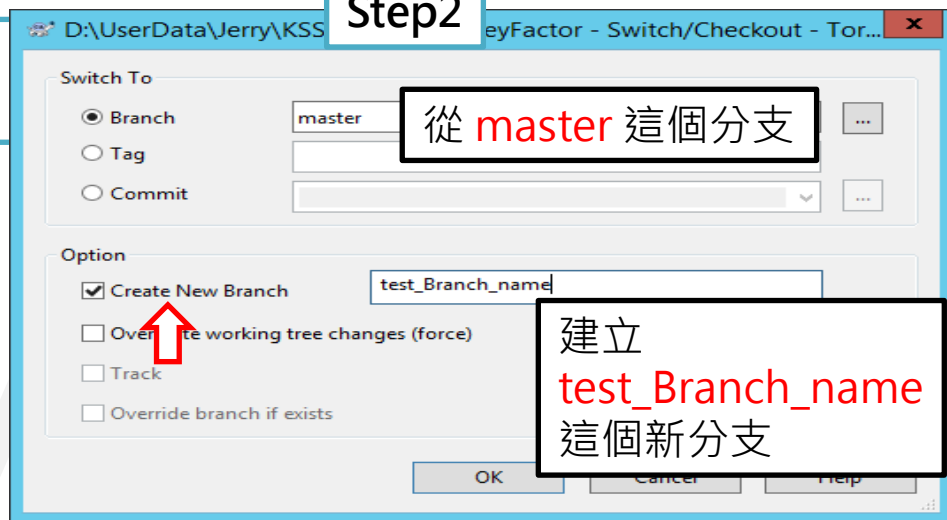
Step3. 填寫完成後 套用 > 確定

建立新開發分支(Branch)

Step1 對剛剛複製(Clone)下來的資料夾
滑鼠右鍵 >> TortoiseGit >> Switch/Checkout



Step2

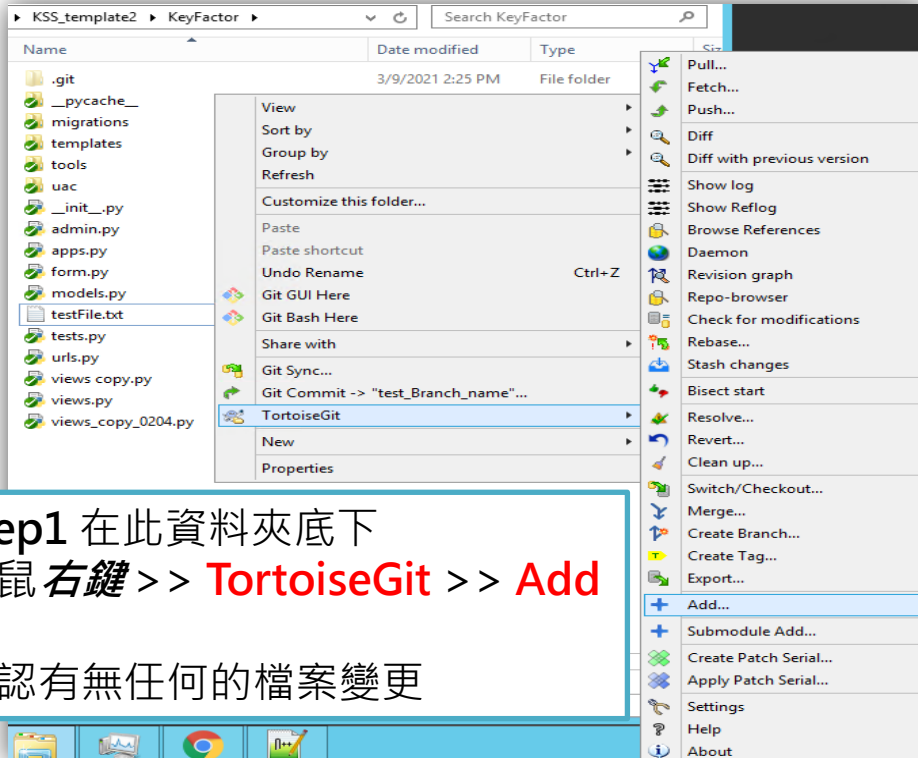
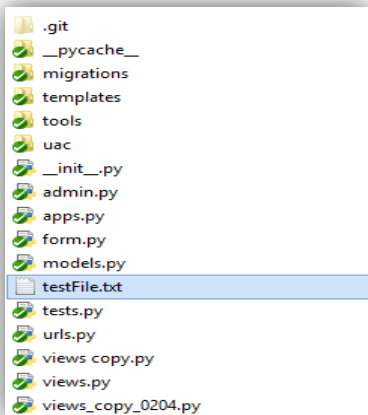


成功應該會自動幫你切換到新分支上

開始進行你的開發！

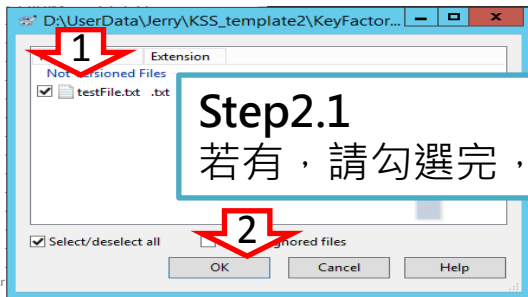
開發段落

情境：今天**新增/刪除/修改**testFile.txt 這份檔案



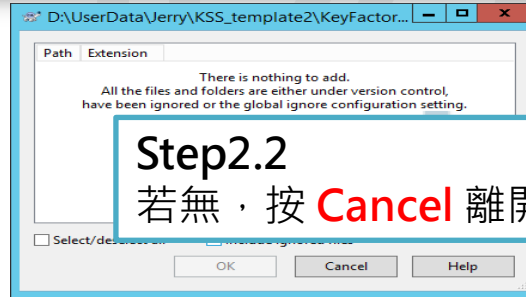
Step1 在此資料夾底下
滑鼠**右鍵**>> **TortoiseGit** >> **Add**

確認有無任何的檔案變更



Step2.1

若有，請勾選完，按 **OK** 新增



Step2.2

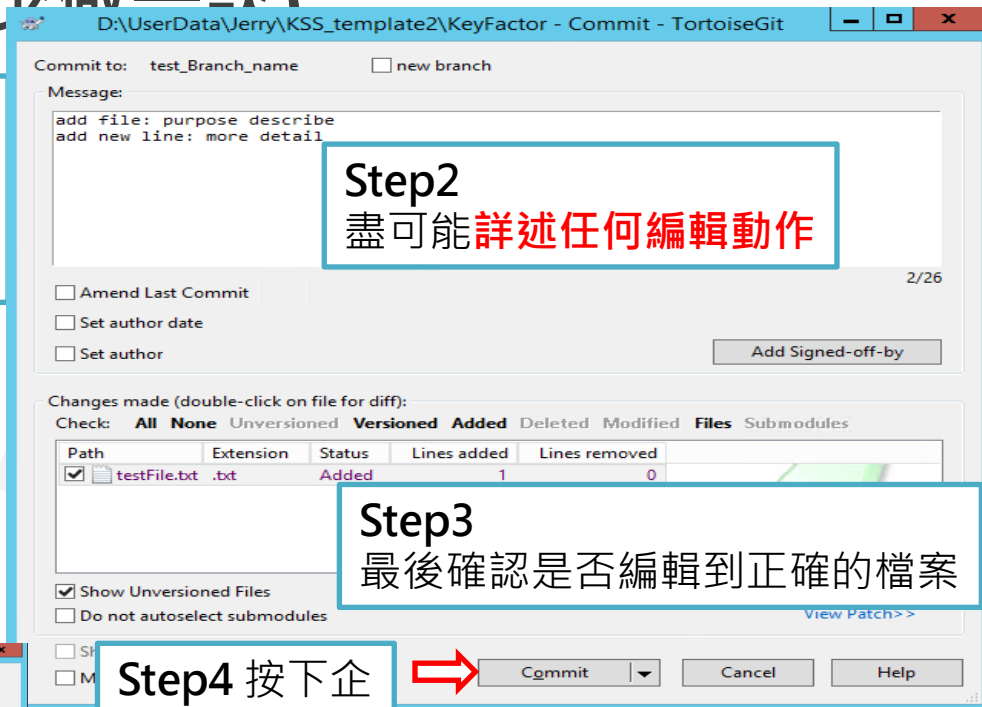
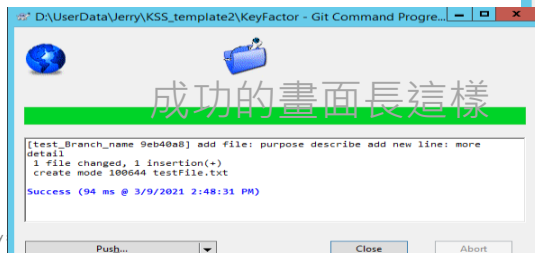
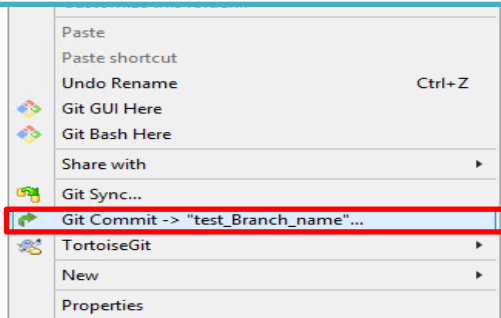
若無，按 **Cancel** 離開視窗

Commit自己的變更 (工作日誌，每天建議至少做一次)

Step1 在此資料夾底下

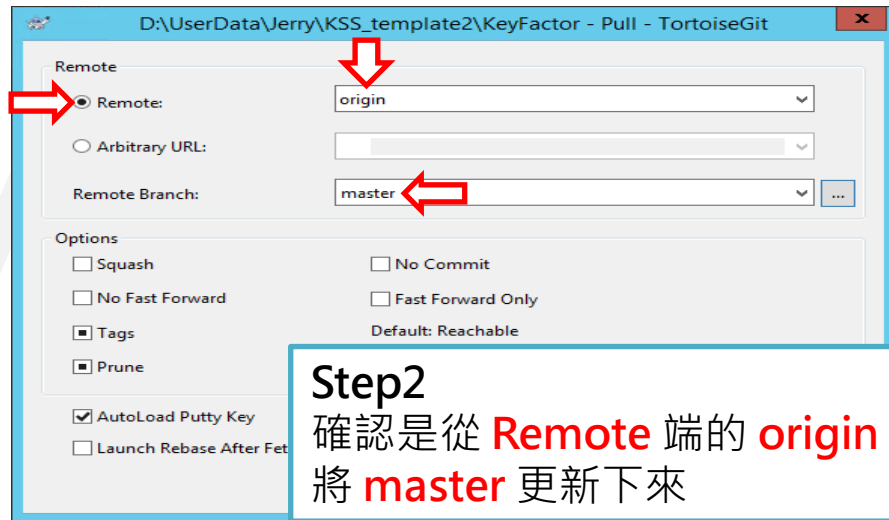
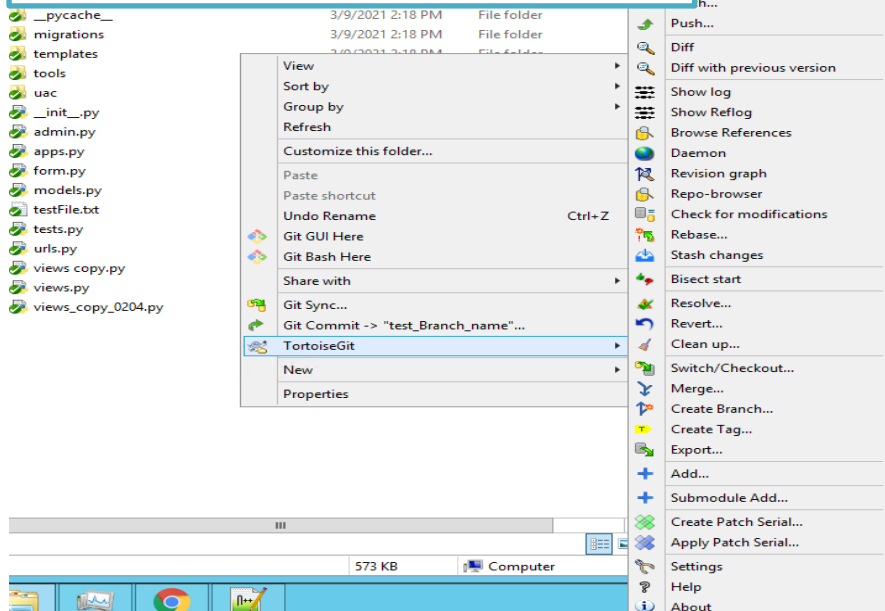
滑鼠右鍵 >> **Git Commit ->** “目前的分支”

※請務必確認是在分支上面開發，而非master



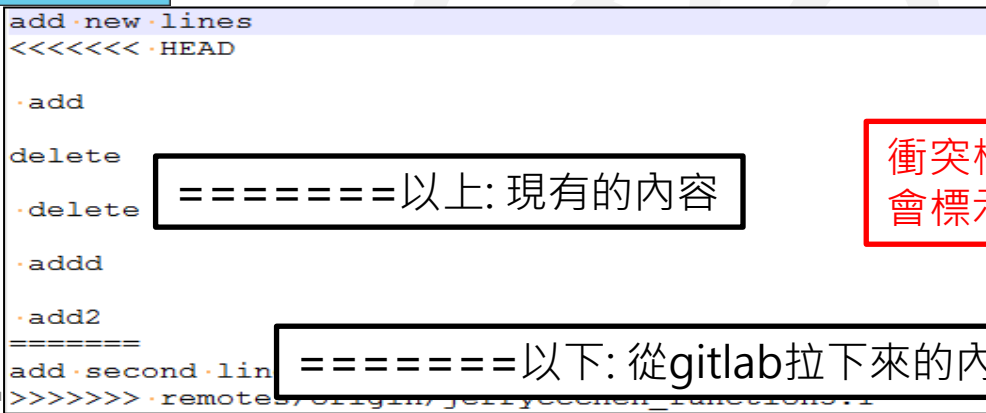
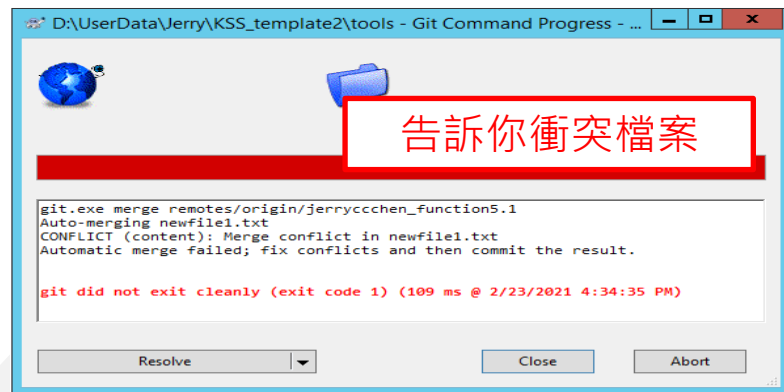
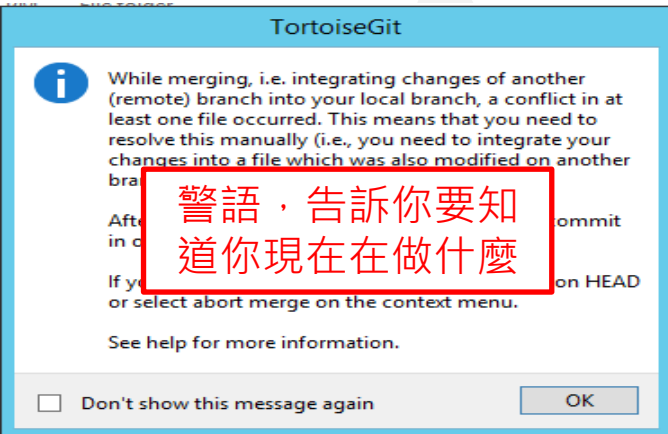
開發完畢/段落 確認伺服器端master有無更新(Pull)

Step1 在此資料夾底下
滑鼠右鍵 >> TortoiseGit >> Pull



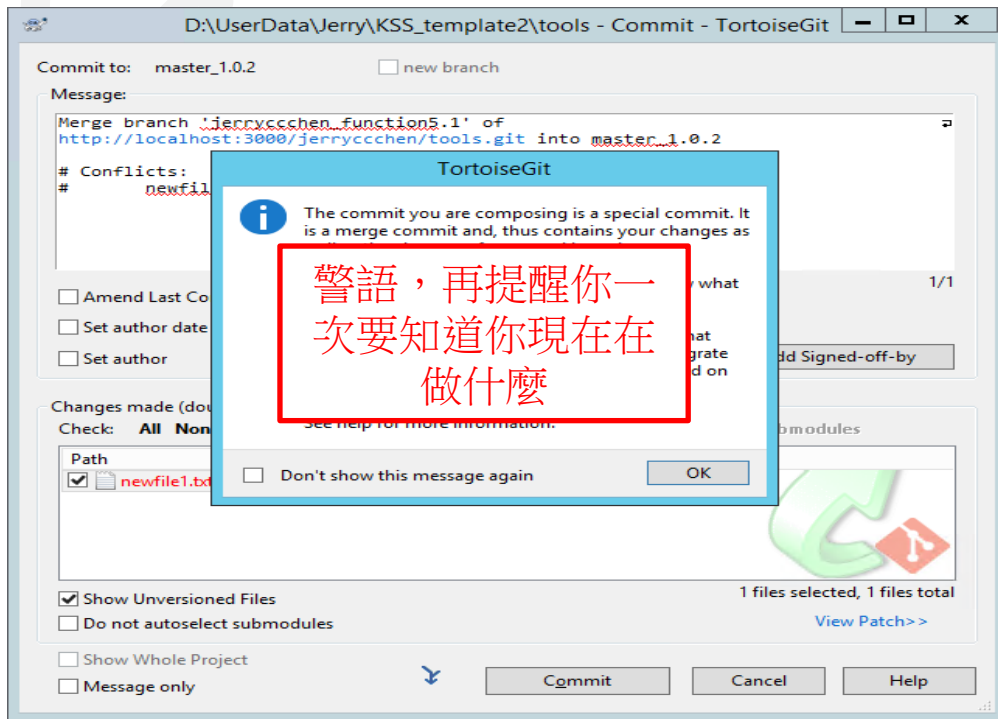
Step2
確認是從 Remote 端的 origin，
將 master 更新下來

注意！有可能Pull下來會出現編輯衝突 請就提示的指定檔案處理相異處



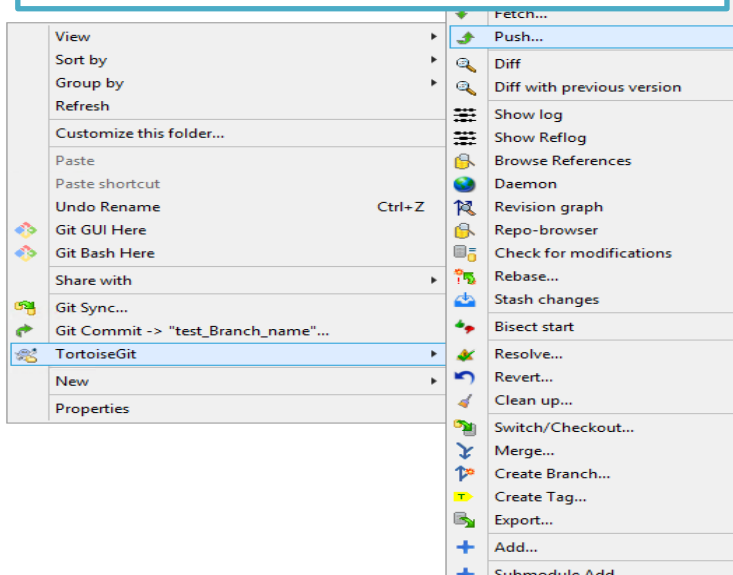
衝突檔案內，
會標示出有衝突的地方在哪

注意！處理完Pull產生的衝突後
請記得在Commit，並敘述修改了哪一份檔案的什麼
衝突(至少說改了哪幾行)



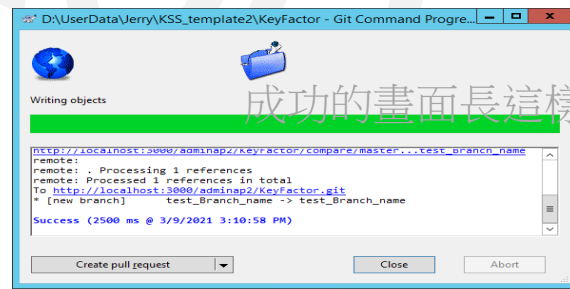
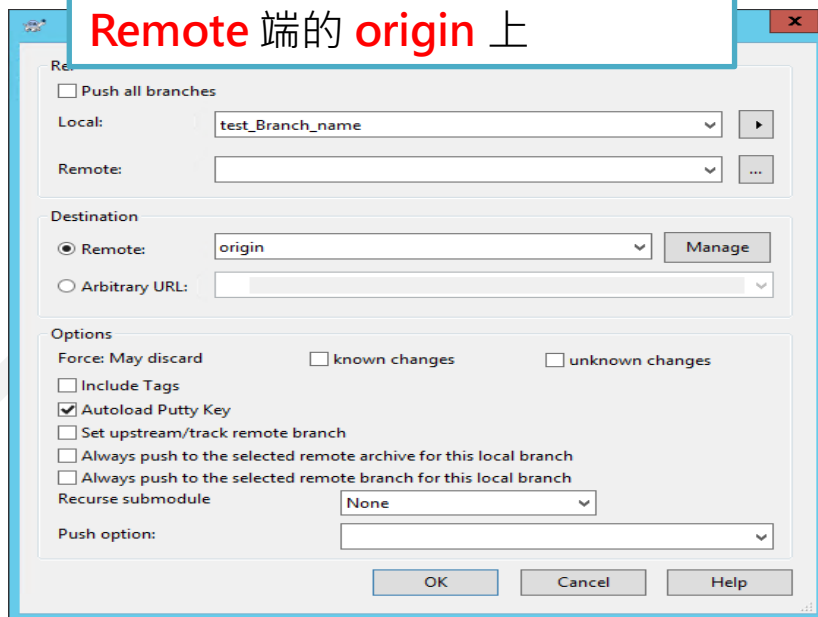
推送(Push)

Step1 在此資料夾底下
滑鼠 **右鍵** >> **TortoiseGit** >> **Push**



Step2

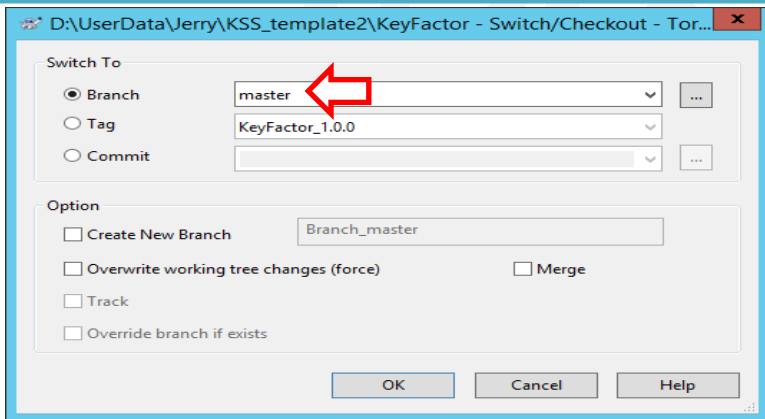
確認是將 “目前的分支” 推送到
Remote 端的 **origin** 上



[延伸閱讀]合併(Merge)

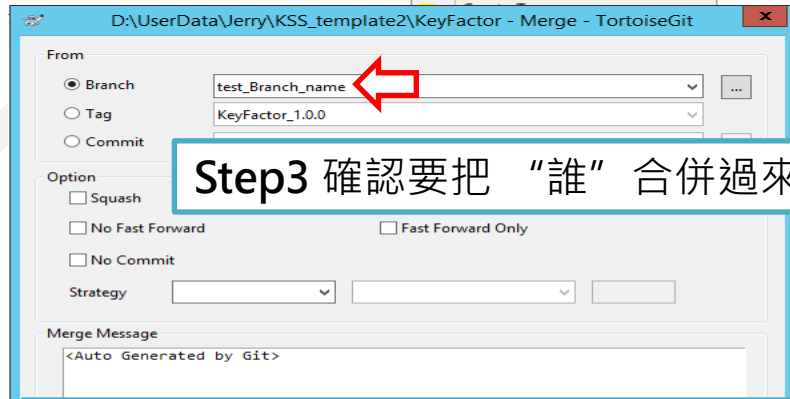
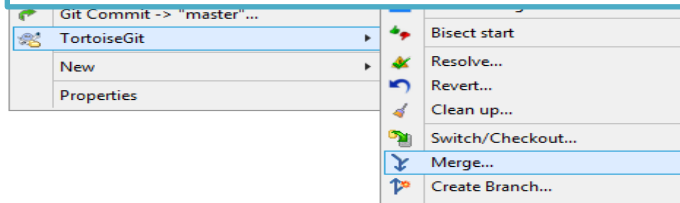
Step1 在此資料夾底下
切換回主線(master)

滑鼠 **右鍵** >> **TortoiseGit** >> **Switch/Checkout**



Step2 在此資料夾底下
打開 Merge 功能視窗

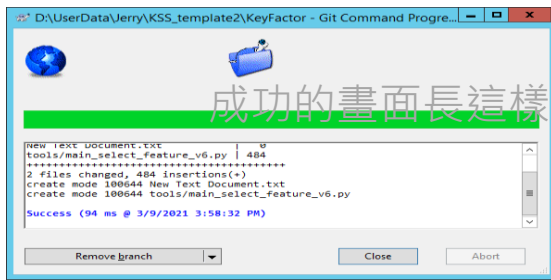
滑鼠 **右鍵** >> **TortoiseGit** >> **Merge**

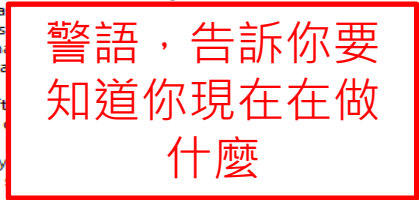


Step3 確認要把 “誰” 合併過來

Step4

填寫合併訊息(至少註明把誰合併到主線)



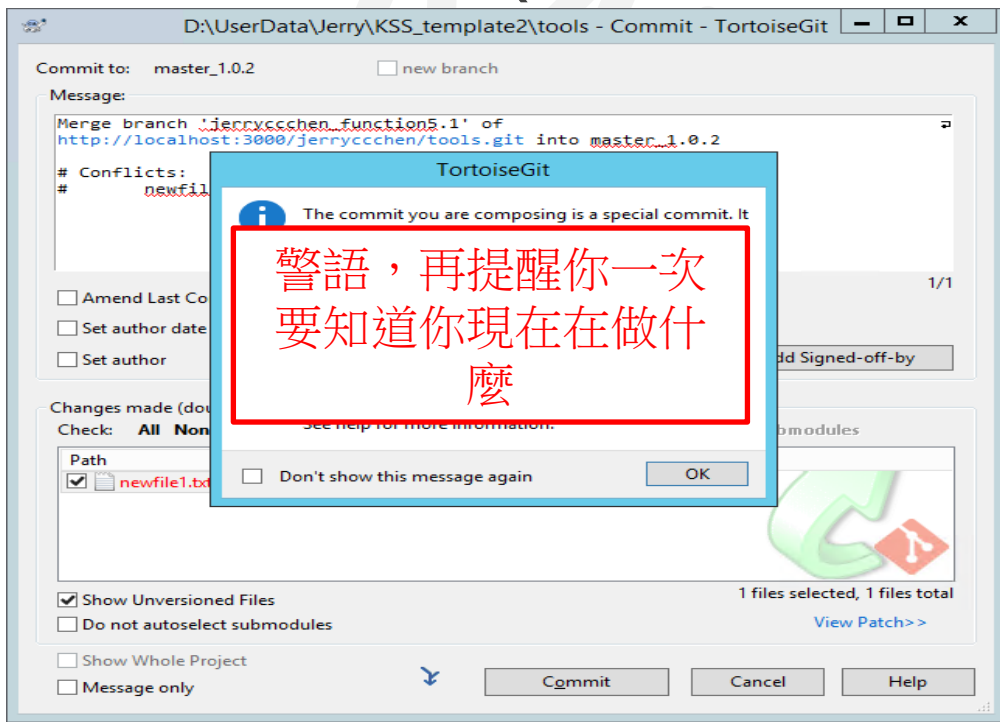


=====以上: 現有的分支的內容

=====以下: 要合併過來分支的內容

衝突檔案內，
會標示出有衝突的地方
在哪

注意！處理完Merge產生的衝突後
請記得在Commit，並敘述修改了哪一份檔案
的什麼衝突(至少說改了哪幾行)



參考格式E.g.：

動作:[新增|修改|刪除]

從[Jerry]合併回[dev]

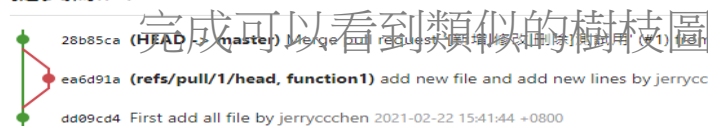
標題:測試用

變更人員:Jerryccchen

變更內容:增加檔案，同時新增5行內容

目的:測試功能

提交線圖



檔案原則

Script

已上線：按照系統架構放置，於文件中提出架構圖

開發中：只保留最近開發項目，舊有檔案一律放入Archive資料夾

Data

設定檔：按照系統架構放置

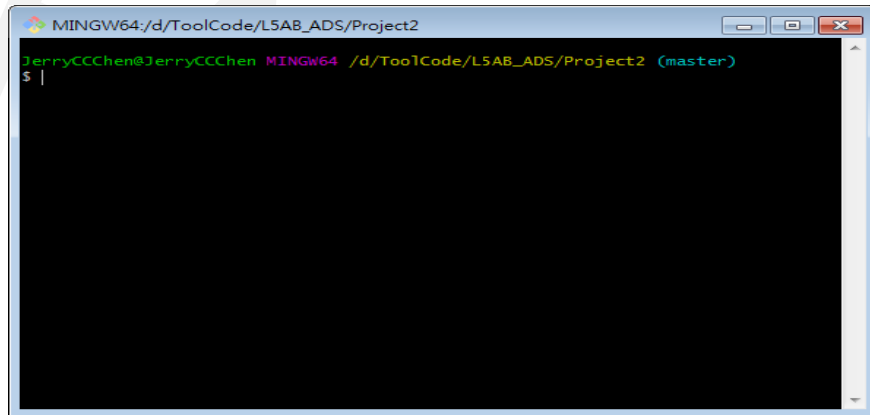
其他：一律壓縮(.zip/.7z)，放入DATA資料夾

圖片

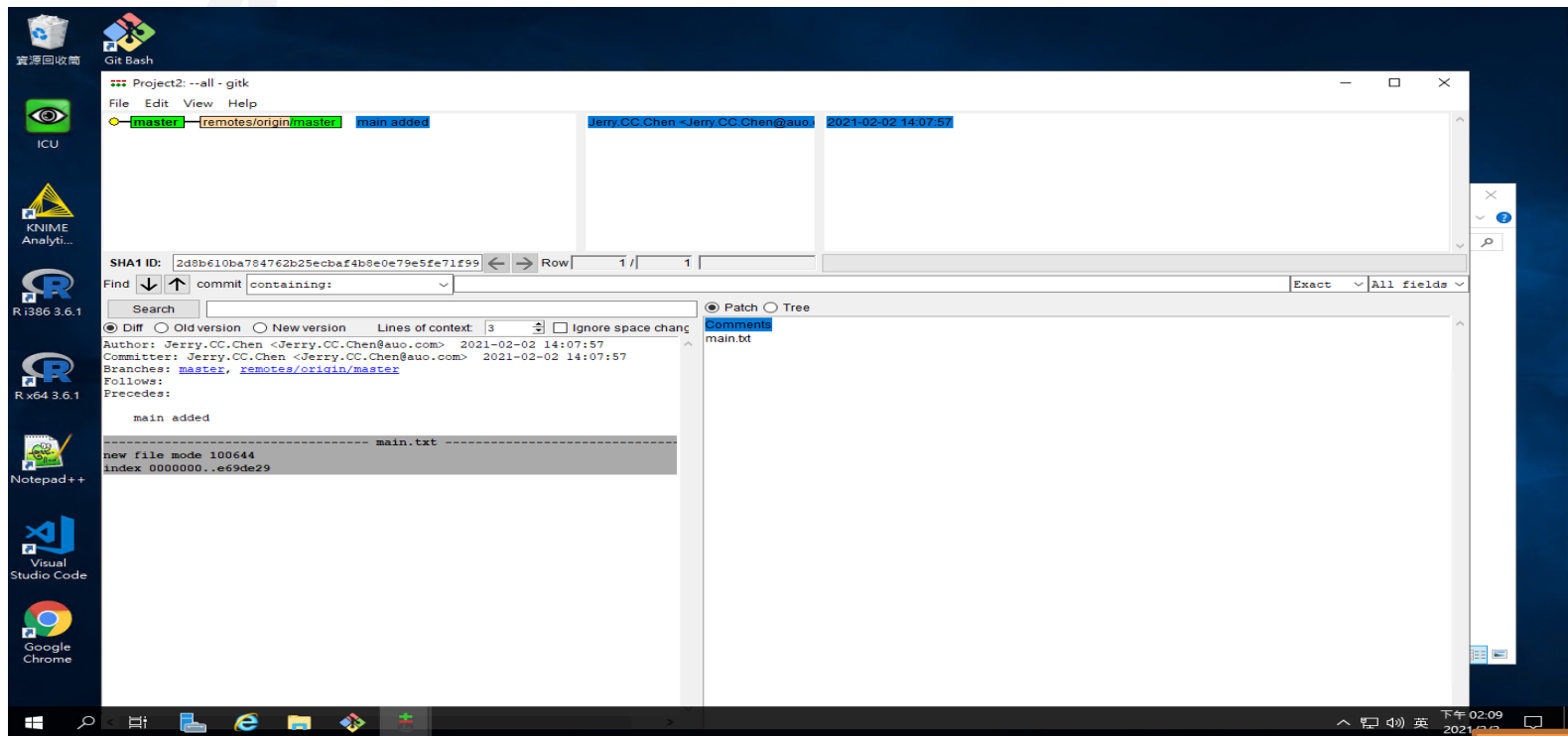
除系統需求外(按照系統架構放置)，其餘一律禁止放入

Git Bash

於clone回來的資料夾內，滑鼠右鍵>>Git Bash Here
執行 git 相關指令的視窗 (建議使用，直接用command視窗也可以)



Git GUI



系統管理員須知

專案管理/使用者須知

功能、操作細項 & 須知

操作說明(COMMAND版)

常用功能

指令集

新增測試版本(建議)

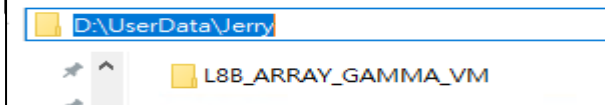
建立測試版本(分支)，開發使用

```
git checkout -b dev  
$ git checkout -b dev  
Switched to a new branch 'dev'
```

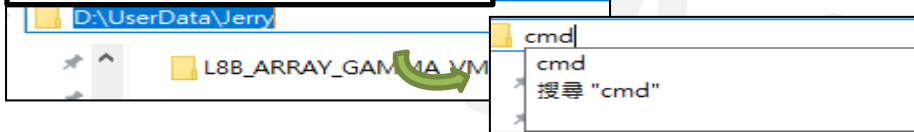
※建議由專案管理者執行此動作，其餘開發人員一律使用dev作為主幹。
待測試完dev的版本後，再由專案管理者進行merge到master的工作。
除了可以多一層保護(備份、錯誤)之外，也可避免上線版本(master)內容過於混雜。

初始化

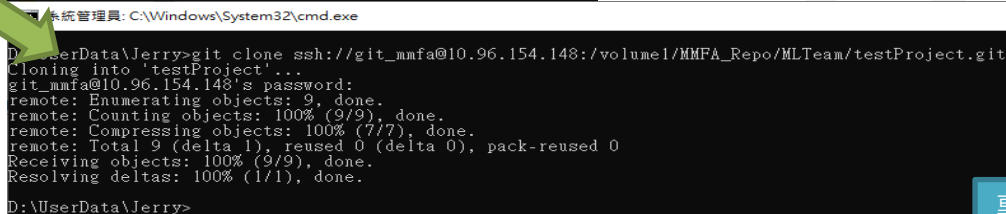
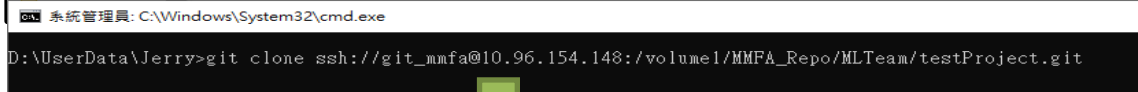
1. 開啟要放置專案的資料夾



2. 從路徑列呼叫 CMD



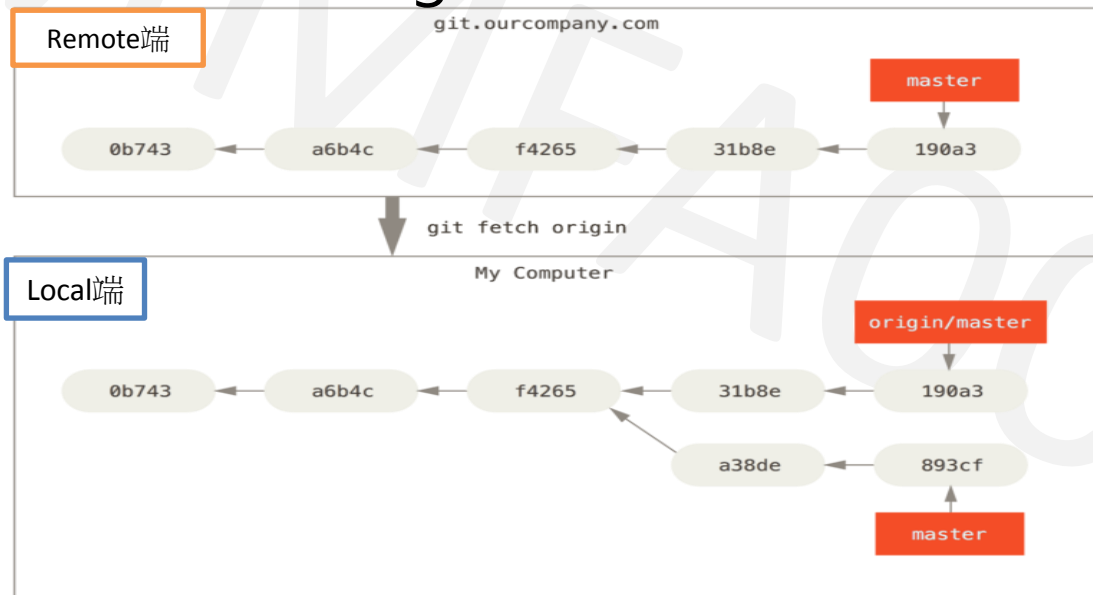
3. 輸入 git clone [Repository 位置]



同步

git fetch

Updates remote-tracking branches.



Branch(分支)類

```
git checkout -b [branchname]
```

從現有的branch新增指定的branch(分支)。

```
git checkout [branchname]
```

切換至指定的branch(分支)

推送

```
git push origin [--delete] [branchname]
```

將開發完成品推送至git server中指定的branch(分支)。
加上--delete則刪除。

Branch(分支)類

git branch [--all]

列出所有branch(分支)並指出當下的branch(Local)。加上--all則列出所有。

git branch -v

列出所有branch(分支)，並顯示各自最新一次的commit訊息。

```
jerryccchen@Tw100043939 MINGW64 /d:/UserData/Jerry/testProject (master)
$ git branch
dev
dev2
function1
function2
function3
* master

jerryccchen@Tw100043939 MINGW64 /d:/UserData/Jerry/testProject (master)
$ git branch --all
dev
function1
function2
function3
* master
remotes/main/master
remotes/origin/dev
remotes/origin/dev2
remotes/origin/function1
remotes/origin/function2
remotes/origin/function3
```

```
jerryccchen@Tw100043939 MINGW64 /d:/UserData/Jerry/testProject (master)
$ git branch -v
dev          c769be1 merge function3 to dev
dev2         e33db4b test message
function1    7fb97b0 add function1
function2    c93622b add function2
function3    e4fd0fa add function3
* master     c769be1 merge function3 to dev
```

Branch(分支)類

```
git branch --move 舊名字 新名字
```

修改branch名稱。(Local)

```
git push --set-upstream origin 新名字
```

修改完名字後，重新push上去。

利用 *git branch --all* 檢查，無誤後用 *git push origin --delete* 舊名字刪除舊的branch。

Branch(分支)類 [注意]

```
git branch --move master main
```

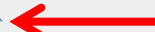
Local 的master消失，改名為main；但remote端的master還在。

在刪除remote端的master前請確認：

1. 任何與此案相關的專案、scripts都必續重新確認/設定。
2. 與此專案的相關人員諮詢過
3. 所有參考這個branch都要更新
4. 關閉所有與此branch相關的需求

確認完上述項目後，得以刪除remote端的master。

```
git branch --all
* main
remotes/origin/HEAD -> origin/master
remotes/origin/main
remotes/origin/master
```



Branch(分支)類

git merge *[branchname]*

從當下的branch(分支)合併到指定的branchname。

※專案管理者要上線時(合併至master)，請先切換到 master，再執行merge的動作。

git mergetool

透過GUI解決衝突區塊。

```
<<<<<< HEAD:index.html
<div id=
=====
<div id=
please
</div>
>>>>>> iss53:index.html
```

當下的Branch內容

Merge過來的Branch內容

下載最新版本 (從server)

```
git pull origin [branchname]
```

從git server中將指定的branch(分支)拉到本地端

開發中使用

```
git add .
```

新增編輯/開發產生的變更

```
git commit -m '[message]'
```

本地端確認編輯/開發變更的動作，**務必填寫編輯/開發動作
(英文)**

https://backlog.com/git-tutorial/tw/stepup/stepup3_2.html

<https://www.atlassian.com/git/tutorials/using-branches/git-branch>

<https://www.atlassian.com/git/tutorials/using-branches/git-checkout>

<https://www.atlassian.com/git/tutorials/using-branches/git-merge>