**TortoiseGit教學**

**1.下載TortoiseGit**

若要使用TortoiseGit，請先至下列的網站安裝msysgit。

<https://git-for-windows.github.io/>

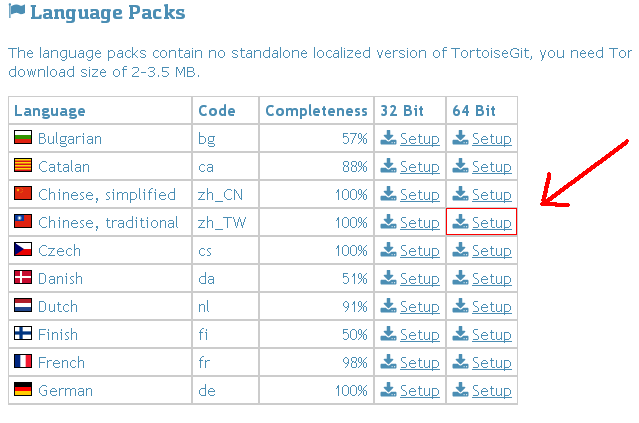
接著到TortoiseGit官網下載程式

<https://tortoisegit.org/>

若需要中文化或他國語言，請在TortoiseGit download頁面尋找”Language Packs”，並依照所需的語言以及位元數下載所需要的語言包。

以下載繁體中文 64位元語言包為例

請依照下圖點選紅色框框的Setup來下載語言包

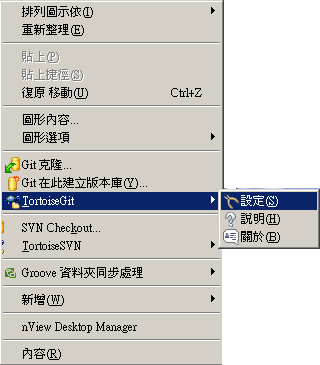


在安裝過程中如果沒有需要或者不清楚內容的話請使用預設的選項。

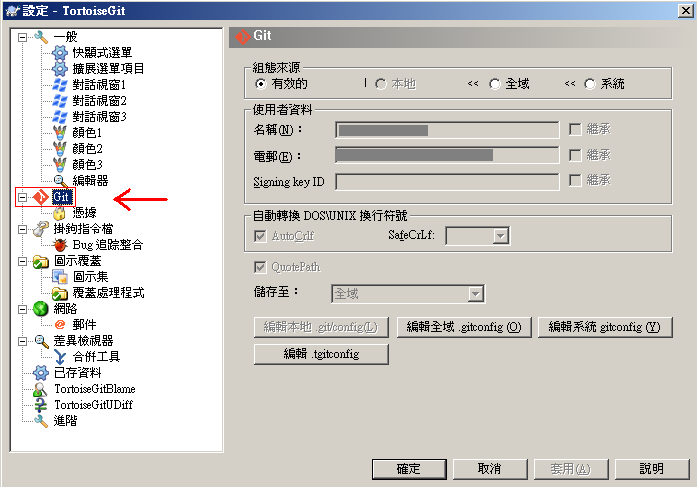
P.S : 最新版本並不支援Windows Vista 或 XP之前的版本，如果有需要請使用1.8.16.0版本

**2.初始設定Git**

安裝完成後，第一次使用需要進行些許的設定，請在桌面或其他任意的地方點擊右鍵



會像上圖一樣看到裡面多出了許多的選項，請點選TortoiseGit 🡪 設定。



請在左邊選單點選Git，並點選全域按鈕，然後輸入名稱及電子郵件信箱。

現在我們設定完成了，就可以開始使用TortoiseGit了！

**3.開始使用Git**

**3.1數據庫**

在正式開始使用Git之前，我們必須了解何謂數據庫(Repository)。

數據庫是紀錄檔案或目錄狀態的地方，可以儲存內容的修改歷史紀錄，並且追蹤內容的狀態及版本。

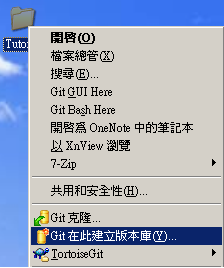
數據庫分為”遠端數據庫”以及”本地端數據庫”，遠端數據庫配有專用的伺服器，可以讓多人使用並共享；本地端數據庫為方便個人使用，而在自己的機器上配置的數據庫。

你可以在本地端數據庫進行資料的更改、還原或追蹤等Git上擁有的功能，但是倘若你想要公開本地端數據庫的修改內容，就需要上傳到遠端的數據庫了。另外，透過遠端數據庫還能取得其他人修改的內容。

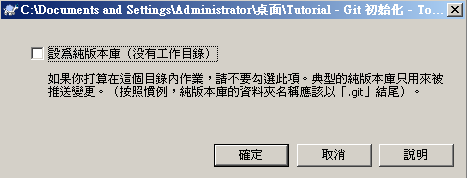
**3.2 建立數據庫**

現在我們對數據庫有了基本的了解了，接下來我們將會介紹建立數據庫的方法。

首先我們要先建立一個本地端的數據庫，請在桌面或任何地方建立一個資料夾”Tutorial”，並對其點擊右鍵，選擇”Git在此建立版本庫”選項。



接著會顯示下列的視窗，請不要點選設為純版本庫並按下確定。



看到下列的視窗就表示建立數據庫成功了。



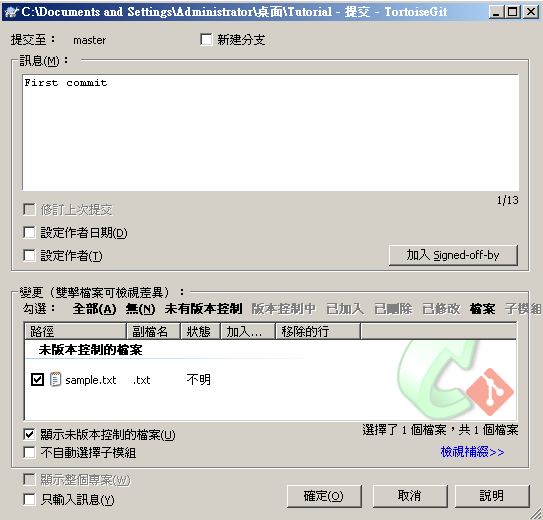
而你的資料夾圖示應該要變成下列所示

(若沒變更請重新整理)

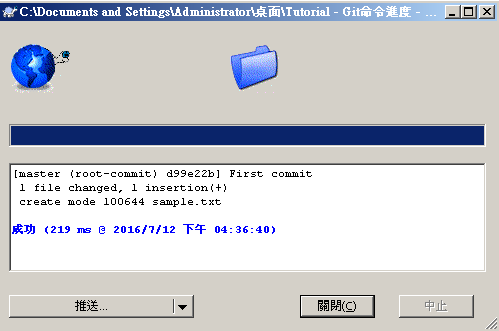


接下來我們會試著演示數據庫的使用以及運作方式，首先請在Tutorial資料夾中新增一個純文字檔Sample.txt，並隨意在內部輸入內容。

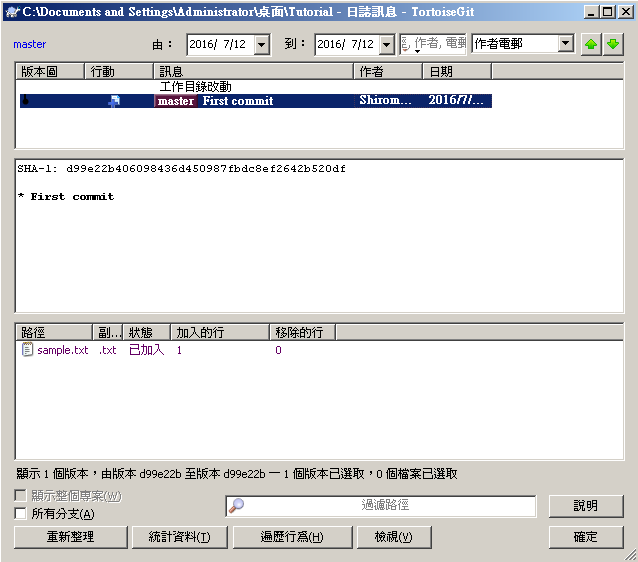
接著打開tutorial目錄，對著資料夾內部點擊右鍵，並點擊其中的”Git提交”按鍵就會出現下列視窗。



如果作業成功就會出現以下畫面。

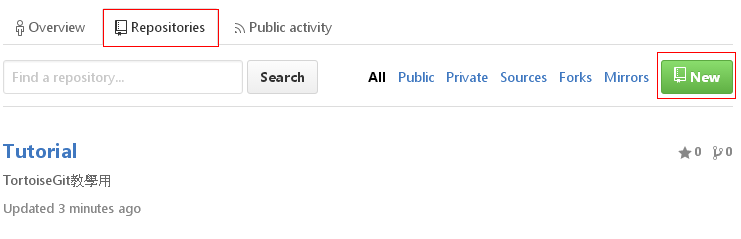


點擊右鍵開啟選單TortoiseGit🡪顯示日誌，剛才的提交就會如同下圖一般顯示在日誌訊息裡了。

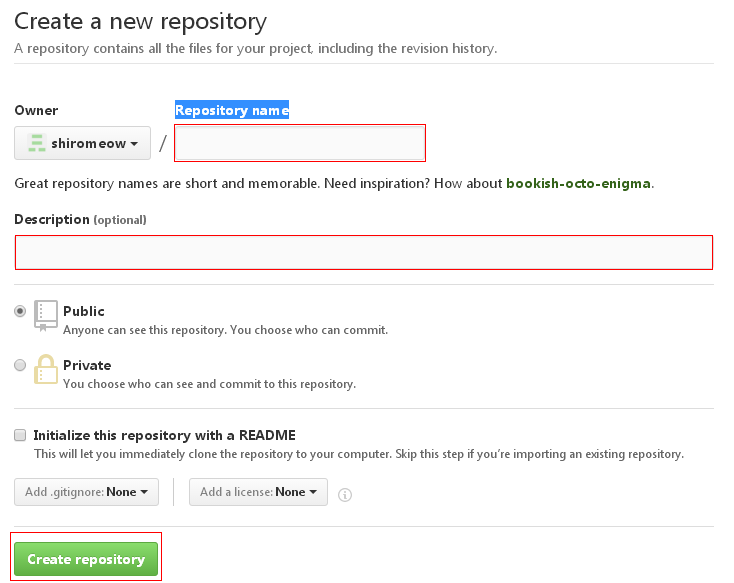


接著我們要試著上傳到遠端的數據庫了，在本次的教學中，我們選用GitHub的平台來建立遠端數據庫。

首先先在GitHub平台註冊一個帳號，然後在右上角的頭像點選your profile，在頁面中會發現Repositories的選項，然後點選右邊綠色的”NEW”按鈕。



Repository name請輸入tutorial，Description請隨意輸入，在GitHub中，將數據庫設為Public(也就是所有人都能看到)是免費的，如果想要私人的數據庫的話則需要付錢給網站，接著點選下面綠色的”create repository”。



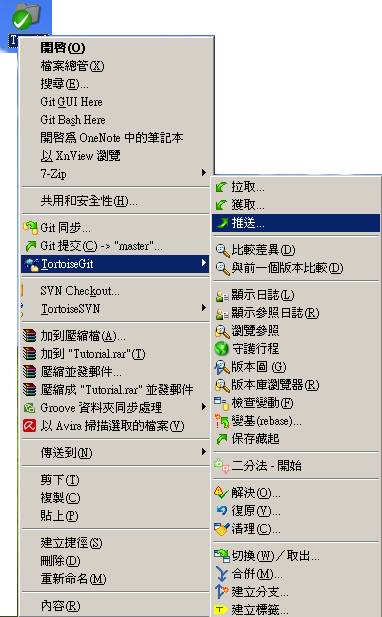
這樣一來遠端的數據庫就建立完成了。

**3.3 推送(PUSH)**

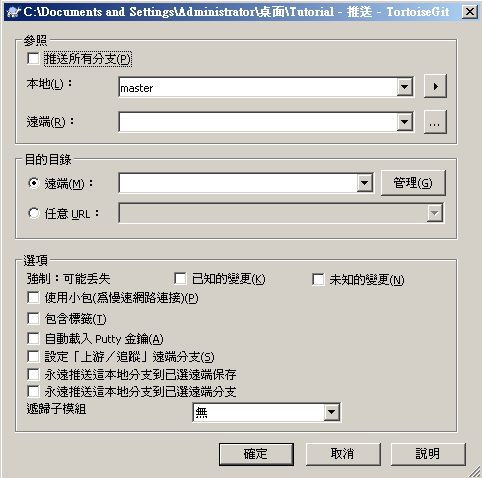
建立好數據庫之後，該來了解一下TortoiseGit之中的功能了，首先要介紹的是**推送(push)**指令，push是將本地端數據庫的修改歷史共享到遠端數據庫的操作，執行push之後，本地端的修改歷史就會上傳到遠端的數據庫，使兩個數據庫的修改歷史保持同步。

在前面我們已經建立好一個本地端數據庫了，現在來嘗試一下將tutorial資料夾的內容push到遠端數據庫吧。

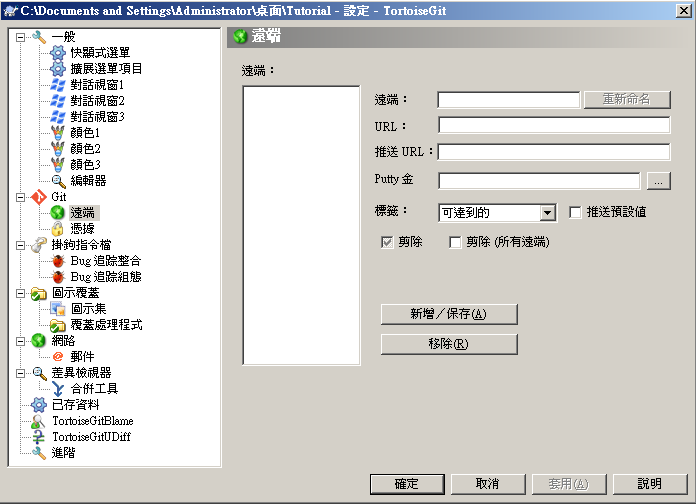
首先，請先右鍵點擊Tutorial資料夾選取Tortoise目錄並點擊”推送”。



顯示以下畫面後選擇”管理”

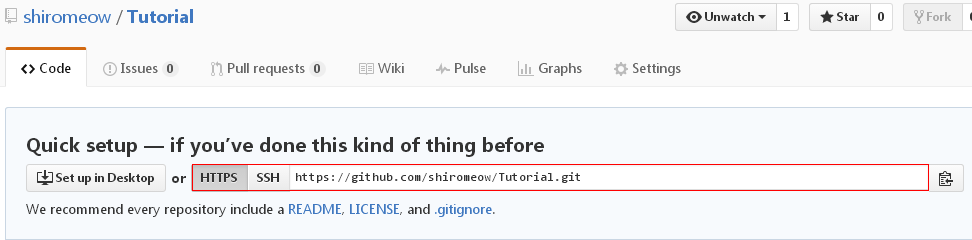


會出現以下畫面



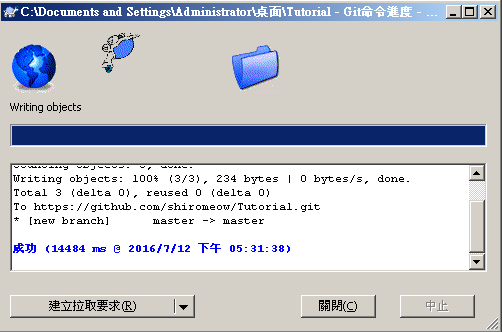
在”遠端”中輸入遠端數據庫的名稱。

在”URL”中輸入之前在遠端數據庫中出現的網址。

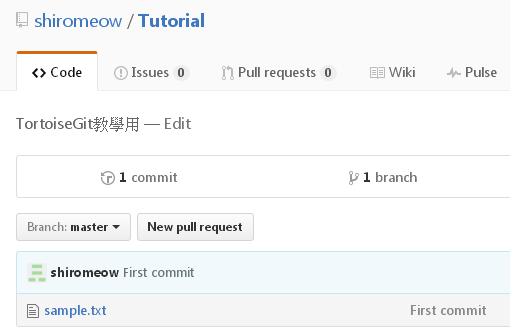


接下來會要求輸入用戶名稱及密碼，請輸入你註冊時用的用戶名稱及密碼。

成功後就會看到下列的視窗。



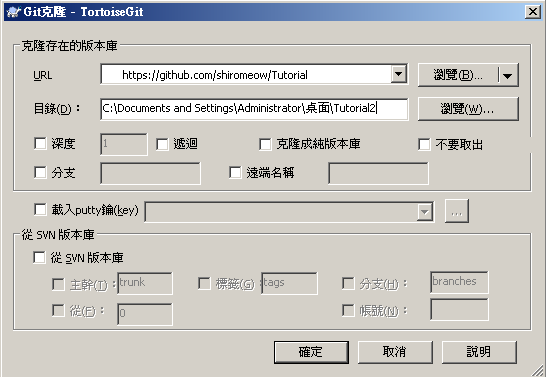
而我們的檔案也出現在遠端數據庫中了！



**3.4 複製(Clone)**

現在檔案已經在遠端數據庫了，讓我們試著將檔案複製下來，來看看檔案是否真的完整的上傳上去了，這個時候我們就會用到**複製(clone)**指令。

首先我們在桌面上點擊右鍵開啟選單，點擊Git克隆選項，出現的視窗會如下圖所示，並輸入遠端數據庫的URL(剛剛在GitHub上複製的URL)，然後目錄請輸入想放的本地路徑(本次於桌面的tutorial2資料夾)。



最後進到複製的目錄看看，文字檔中是不是自己所打的那一串字呢。

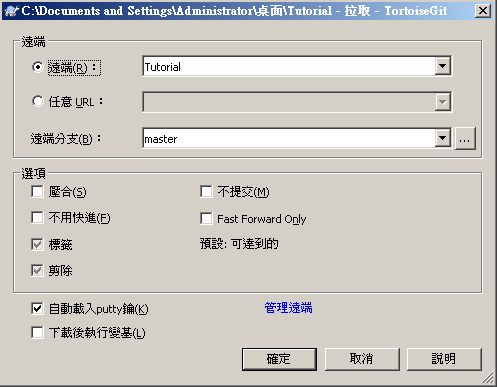
**3.5 拉取(Pull)**

接下來我們要嘗試的指令是**拉取(pull)**，拉取指令的作用在於將遠端數據庫的修改歷史同步到本地的資料庫，在多人使用數據庫的情況下，利用拉取同步其他人的修改內容到本地是有其必要性的。

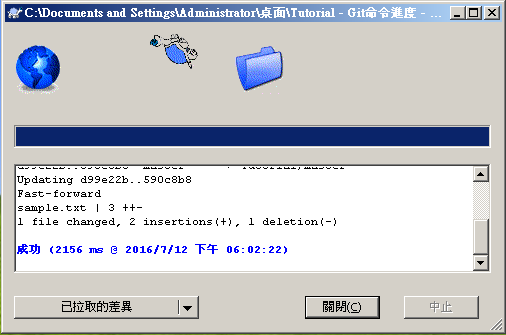
要嘗試pull指令，首先我們要稍微更改遠端數據庫的內容，所以請在tutorial2(或是你剛剛clone的任意名字的資料夾)內的文字檔更改裡面的內容，並像前面所講執行Push的操作。

現在你的遠端數據庫以及你的tutorial資料夾的內容已經不同步了，為了使他們同步，我們會需要用到pull指令來將遠端數據庫的內容同步到tutorial資料夾中。

請點擊右鍵，並在TortoiseGit選單中選擇”拉取”，就會出現以下的視窗。



確定後出現以下視窗就成功了。



**4.分支**

**4.1 分支的介紹**

在使用Git進行版本控制的時候，我們會需要用到分支的功能，分支是一個十分重要的功能，可以說Git的大部分精隨都藏在這裡了，在下面我們會簡要的介紹分支的內容。

分支是為了將修改紀錄的整體流程分開保存，使得分開的分支不受其他分支更改的影響，使你在同一個數據庫可以進行不同的修改，讓你有機會在錯誤的修改後，可以還原到原本沒問題的地方，或是需要進行緊急修正的時候，可以先將hotfix做發布的動作，在進行其他的作業等等。

**4.2 分支的種類**

基本上，分支在Git中可以分為三類

* Master分支
* Integration分支
* Topic分支
* **Master分支**

在數據庫進行最初的提交後，Git會建立一個名為master的分支。之後的提交在切換分支之前都會增加在master分支裡。

* **Integration分支**

保持Integration分支的穩定是很重要的，因為新的分支會建立在它的上面，而且Integration分支是為了可以隨時建立發布版本的分支。

通常，大家會將master分支當作Integration分支使用。

* **Topic分支**

Topic分支是為了開發功能或修復錯誤之類的任務所建立的分支。若同時進行多個任務時，您必須建立多個的Topic分支。

所以，如果要進行更改的話，最好先建立Topic分支並在上面做操作，像是使用Jenkins之類的CI工具進行自動建立或測驗時，也是會在Topic分支裡進行。

Topic分支是從穩定的Integration分支上建立的，完成作業後，要將Topic分支再合併到Integration分支。

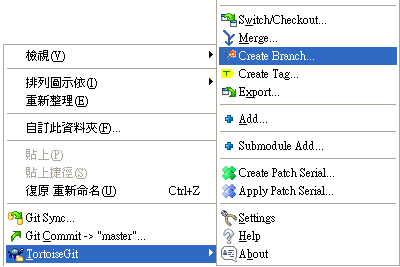
**4.3分支的建立與切換**

學會了分支的種類後，現在就應該學習分支要如何切換了，但是在那之前，我們要對一些東西進行簡短的介紹。

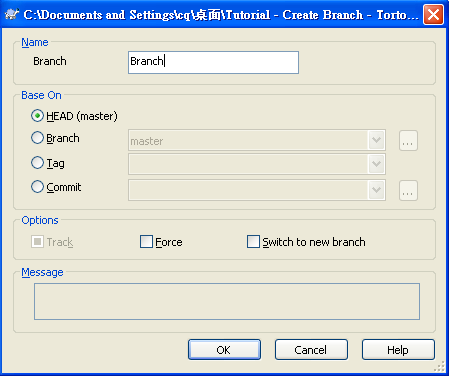
首先，在Git中，**Head**代表當前分支的最新提交名稱，在建立新的資料庫的時候，Git會自動將Head指向master分支，藉由更改Head的指向，我們可以更新正在使用的分支。

**Stash**是暫時儲存檔案修改內容的區域。Stash可以暫時儲存工作目錄還沒提交的修改內容，您可以在事後再取出暫時儲存的修改，應用到原先的分支或者其他的分支中。

想要使用分支的功能，首先我們必須創建一個分支，請在tutorial資料夾中點選右鍵--> TortoiseGit --> Create Branch

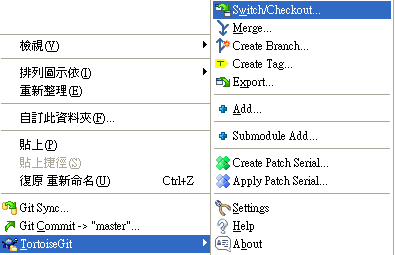


就會出現以下的畫面

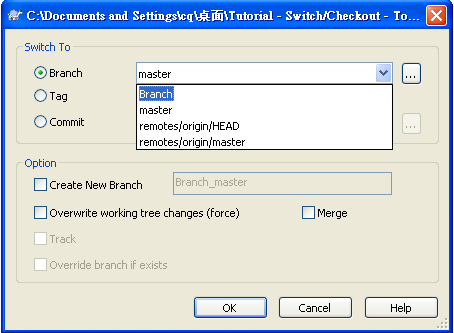


請在Branch欄位打上自己想要取的名字，下面的Base On是指你想要從哪裡建立分支，在這裡因為我們只有Master分支，所以我們是建立在Master分支之上，倘若以後有更多的分支而你又有需要在分支上做額外的改動，你可以改選Branch並選擇你要在上建立的分支。

在建立分支後，就要學會如何去切換它了，想要切換分支請在tutorial資料夾中點選右鍵 🡪 TortoiseGit 🡪 Switch/Checkout



就會看到以下的畫面



你會在Branch的欄位看到選單，請選擇自己取的分支名稱，並且點選確定，就能切換到你建立的新分支了。

**4.4分支的合併**

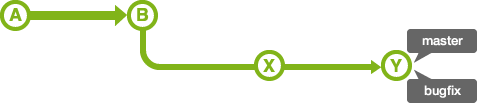
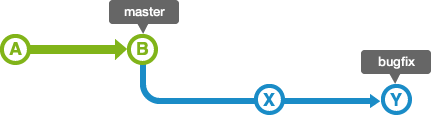
學會建立與切換分支後，該是來學習如何合併分支了，在以下我們要介紹合併分支的兩種方法”Merge”和”Rebase”。

* **Merge**

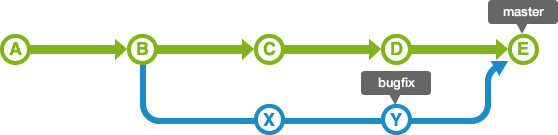
使用 merge，可以合併多個歷史記錄。

如下圖所示 bugfix 分支是從 master 分支分開出來的。

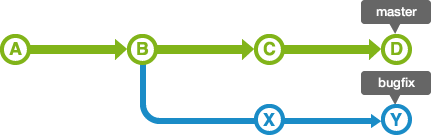
合併 bugfix 分支到 master 分支時， 如果master 分支的狀態是沒有更改過的話，那麼這個合併是非常簡單的。 bugfix 分支的歷史記錄包含了 master 分支的歷史記錄，所以只要把bugfix 移動到 master 分支就可以導入 bugfix 分支的內容。這樣的合併被稱為 fast-forward（快轉）合併。



但是，master 分支的歷史記錄有可能在 bugfix 分支分開後有新的修改。這時候，要把 master 分支的修改內容和 bugfix 分支的修改內容匯合起來。



匯合兩個修改時會產生一個名為「合併提交」的提交。Master的位置會被更新到新建立的合併提交上。

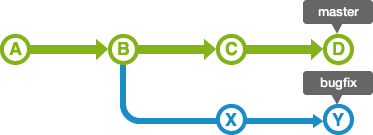


* **Rebase**

和 merge 的例子一樣，如下圖所示 bugfix 分支是從 master 分支分開出來的。

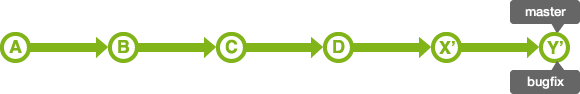


使用 rebase 進行分支合併的話會出現下圖所顯示的歷史記錄。現在來簡單的講解一下合併的流程吧。

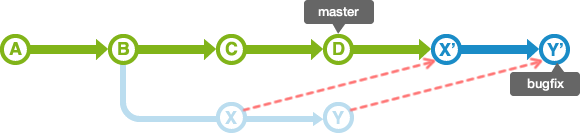


首先，rebase bugfix 分支到 master 分支。bugfix 分支的歷史記錄會增加在 master 分支的後面。因此，如圖所示歷史記錄會被統一，形成簡單的一條線。

移動提交 X 和 Y 有可能會發生衝突，所以需要修改各自提交時發生衝突的部分。



執行Rebase 時， master 的位置不變，因此，待 master 分支合併 bugfix 分支後，master 的HEAD會移動到 bugfix 的HEAD這裡。



* **Note**

Merge 和 rebase 都是合併歷史記錄，但是結果不同。

* **Merge**  
  修改內容的歷史記錄會維持原狀，但是合併後的歷史紀錄會變得更複雜。
* **Rebase**  
  修改內容的歷史記錄會接在要合併的分支後面，合併後的歷史記錄會比較清楚簡單，但是，會比使用 merge 更容易發生衝突。

您可以根據團隊的需求分別使用merge 和 rebase 。例如，您想要簡化歷史記錄，您可以試試看以下的操作：

* 當您想要從integration分支導入最新變更時，您可以在topic分支上使用 rebase 。
* 當您想要合併topic分支上的變更到integration分支時，您可以先在topic分支上使用

rebase ，接著再將topic分支上的變更合併到integration分支 。