msysGit

Git for Windows

5484218541881

* Git為分散式版本控制系統，是為了更好管理Linux內核而開發的。

Git可以把檔案的狀態作為更新歷史記錄保存起來。因此可以把編輯過的檔案復原到以前的狀態，也可以顯示編輯過內容的差異。

而且，當有人想將編輯過的舊檔案上傳到伺服器、覆蓋其他人的最新檔案時，系統會發出警告，因此可以避免在無意中覆蓋他人的編輯內容。

在Git管理檔案的話，更新歷史會保存在Git。所以不需複製備用的檔案

* 使用數據庫管理歷史記錄

數據庫 (Repository) 是記錄檔案或目錄狀態的地方，儲存內容的修改歷史記錄。在數據庫的管理下除了儲存修改歷史記錄外，還可以追蹤內容的狀態和版本喔。

* 遠端數據庫和本地端數據庫

首先，Git的數據庫分為遠端數據庫和本地端數據庫。

• 遠端數據庫: 配有專用的伺服器，為了讓多人共享而建立的數據庫。

• 本地端數據庫：為了方便用戶個人使用，在自己的機器上配置的數據庫。

您可以在本地端數據庫上使用像是還原更改、跟踪更改等所有Git版本控制功能。

不過，如果想要公開在本地端數據庫的修改內容，就需要將內容上傳到遠端數據庫了。另外，透過遠端數據庫還可以取得其他人修改的內容。

* 建立本地端數據庫的方法有兩種：一種是重新建數據庫。另一種是複製遠端數據庫。

* 記錄修改的提交

若想把變更與新增的檔案/目錄儲存到數據庫中，您需要執行提交（Commit）。

執行提交後，數據庫裡會產生上次提交的狀態與現在狀態的差異記錄（也被稱為Revision）。

如下圖，提交是以時間順序排列狀態被儲存到數據庫中的。憑藉該提交和最新的檔案狀態，就可以知道過去的修改記錄以及內容。

為了區分提交，系統會產生一組識別碼來給提交命名，識別碼會根據提交修改的內容計算出不重複的40位英文數字。指定識別碼，就可以在數據庫中找到對應的提交。

* Tips（小撇步 ）

像錯誤修復或功能添加之類不同含義的更改，要盡量分開來提交。這樣可以方便事後從歷史記錄裡找出特定的修改內容。

執行提交時，系統會要求輸入提交訊息。請務必輸入提交訊息，因為在訊息空白的狀態下執行提交是會失敗的。

* Tips（小撇步 ）

提交訊息是查看其他人提交的修改內容或自己檢查歷史記錄時重要的資料。所以要用心填寫讓人容易理解的提交訊息。

Git的標準提交訊息：

第1行：提交時修改內容的摘要

第2行：空行

第3行以後：修改的理由

* 工作目錄與索引

工作目錄（Working Tree）是保存您目前正在處理檔案的目錄，Git 相關的操作都會在這個目錄下完成。

索引(Index )位於工作目錄和數據庫之間，是為了向數據庫提交作準備的暫存區域。

所以在工作目錄上做的任何變更並不會直接提交到數據庫的。Git在執行提交的時候，不是直接將工作目錄的狀態儲存到數據庫，而是將索引的狀態儲存到數據庫。因此，要提交變更，首先必需要把變更內容加入到索引中。

索引的存在可以排除工作目錄裡不必要的檔案提交，還可以只將檔案變更內容的一部分加入索引並提交。

* 開發分版觀念：Master、Branch和Merge

主幹（Master）與分支（Branch）是稱呼專案的主要版本和分支版本。

在GitHub上，通常第一個建立的專案版本會稱為Master版本。

但對Git而言，開發專案的版本並無主幹（Master）與分支（Branch）的分別，每個分支都是等價關係，而開發者自行調整版本命名來區分。

一般習慣將穩定版本稱主幹，其餘的變動、開發中版本則都稱作分支。

Branch也是一個建立分支的Git指令，可將某個歷史版本複製一份，獨立成為另一個新的分支版本，而合併（Merge）指令剛好相反，則是把兩個不同的分支版本，合併到其中一個分支上。

* 版本線圖（commit graph）

Git則提供了一個視覺化的版本線圖（commit graph），來呈現出主幹與各分支連結形成的樹狀架構。不過隨著專案越來越大，版本線圖會變得越來越複雜。

* 專案分版手法：Clone和Fork

複製（Clone）指令是把專案在遠端儲存庫上的所有內容複製到本地，建立起本機儲存庫及工作目錄，而叉（Fork）則是把別人專案的遠端儲存庫內容複製一份到自己的遠端儲存庫

* Pull & Push

者可以執行Pull取得其他人開放授權的遠端儲存庫上的程式碼，也可以將自己修改的程式碼Push到可存取的第三者遠端儲存庫上。透過推（Push）跟拉（Pull）兩個指令，開發者就能互相分享原始碼。

* Pull Request

Pull Request則是更主動地要求第三方開發者納入自己開發的程式，將本地端儲存庫上的程式碼，整併到對方的儲存庫上。例如A成員可透過Pull Request要求B成員，將A成員修改後的程式碼併入B成員的開發專案中。

GitHub是以Git為核心的線上協作平臺

使用議題（Issue）、新開分支（Branch）、維基（Wiki）、拉取要求（Pull Request）等功能

**設定 Git 執行環境的三個層級** ( **註**: 執行 **git config --help** 可以取得完整指令說明 )

* **系統層級** (System-level configuration) (設定於整台電腦，所有使用者的預設值)
  + git config --system --list
  + 預設路徑: **C:\Program Files (x86)\Git\etc\gitconfig**
  + 或是路徑: **%LOCALAPPDATA%\VirtualStore\Program Files (x86)\Git\etc\gitconfig   
    補充參考**: [VirtualStore - Inside Windows Vista User Account Control](http://technet.microsoft.com/zh-tw/magazine/2007.06.uac(en-us).aspx" \t "_blank)
* **使用者層級** (User-level configuration) (設定於目前登入的使用者)
  + git config --global --list
  + 預設路徑: C:\Users\<使用者帳號>\.gitconfig      (這是個檔案)
  + 常用指令
    - 最常見的設定就是 user 區段下的 name 與 email 參數，第一次用 Git 一定要設定!
      * git config --global user.name "John Doe"
      * git config --global user.email "johndoe@example.com"
    - 在 Linux 底下可以指定編輯器
      * git config --global core.editor vim
    - 在Windows底下，最好打開 core.autocrlf 選項，讓 commit 的檔案沒有 CR 字元   
      ( 純 Windows 開發環境，可以設定為 false；跨平台協同開發，建議設定為 true )
      * git config --global core.autocrlf true
    - 自動訂正打錯的參數，例如 git statsu 會自動修正為 git status 讓你不用重打一次
      * git config --global help.autocorrect 1
* **儲存區層級** (Repository-level configuration)
  + git config --list
  + 預設路徑: <Git儲存區>\.git\config

