GIT報告

1. git常用指令

* git config：使用者設定
* git --version：查看當前git版本
* git init：初始化git環境
* git status：查詢當前目錄的狀態
* git add <file name>：新增檔案至git追蹤清單
  + .：將所有Untracked files新增至追蹤清單
  + -u：更新追蹤清單中的檔案
* git commit -m “”：提交變更，並提交變更紀錄
  + --amend：重新提交
* git log：查看commit的歷史紀錄
* git rm <file name>：刪除檔案
* git reset HEAD：將檔案退回Untracked files
* git checkout -- <file name>：將檔案復原成上個版本
  + <branchname>：切換至分支
  + -b <branchname>：建立並切換分支
* git clean：清空未追蹤的檔案
  + –n：列出將被清空的檔案清單
* git diff：不同版本的變更比較
  + HEAD：比較最新版與目前工作目錄檔案的差異
  + HEAD^..HEAD：比較前一版與最新版的差異
* git mv <file name> <new file name>：更改檔名
* git branch：顯示分支清單
  + <branchname>：建立分支
  + –d <branchname>：刪除分支
* git merge <branchname>：合併分支
* git clone <url>：複製遠端儲存庫
* git remote add <name> <url>：新增遠端儲存庫
* git push：將修改內容推至分支
  + --delete <repository> <branchname>：刪除遠端儲存庫的分支
  + --tags：將本地端儲存庫所有的標籤添加到遠端儲存庫
* git pull：將遠端儲存庫的修改內容合併到本端儲存庫
* git fetch：查看遠端儲存庫的修改內容
* git stash：暫時儲存現狀
  + list：顯示暫存清單
  + pop：恢復最新暫存內容
  + drop：刪除最新的暫存操作
  + clear：刪除所有暫存操作
* git tag：顯示標籤清單
  + -n：顯示標籤註解
  + <tagname>：建立標籤
  + -a <tagname>：建立含註解的標籤
  + -d <tagname>：刪除標籤

1. git best practice
2. commit的原則：乾淨且目的單一

寫程式時常在做特定事情時陷入不小心做太多事的困境，例如當你只想修一個bug，卻在修的過程中發現了另一個，你無法阻止自己想把它修好的慾望，然後又發現一個，結果就像滾雪球，在這次的commit做了太多的更動，這是非常有問題的，所以每次commit越小越集中越好，這有幾項好處：

* 讓團隊的其他成員更容易知道你做了什麼更動，在閱讀程式能更有效率。
* 如果要退回commit，這樣做容易得多。

1. commit message必須要有意義

這應該是最容易達成的，能使其他人和你自己將來的工作更加輕鬆。

1. 頻繁地commit

當你頻繁地commit是效果最好且對你有利的，而不是每次commit都要追求完美，更好的辦法是每完成一小塊就commit一次，當你的code不斷保持最小限度的更新，就能避免衝突。

1. 不要修改發布記錄

一旦commit合併回預設分支，就盡量不要修改記錄，因為這對每個有權訪問儲存庫的人都是個問題，雖然git-rebase是好用的功能，但應該只用在自己的分支上，當然，偶爾還是會有無法避免，必須修改記錄的情況，但必須格外小心。

1. 不要commit自動生成的檔案

一般來說，只有手動創建的檔案需要commit，那些自動生成的檔案可以隨時重新生成，可以利用.gitignore將你不想追蹤的檔案略過。

參考資料：

<https://blog.miniasp.com/post/2013/08/18/Learning-Git-Part-1-Installation-Options-Tool-Usage-on-Local>

<https://deepsource.io/blog/git-best-practices/>

<https://www.slideshare.net/alantsai2007/git-48778340>