「Git」儲存庫的運作，是將工作目錄裡的變化，透過更新索引的方式，將資料

寫入成 Git 物件。

「物件」用來保存版本庫中所有檔案與版本紀錄。

「索引」則是用來保存當下要進版本庫之前的目錄狀態。

## git command

====================================================================

git --version (查看版本)

git help XXX (查找XXX指令)

git gc (Git封裝-改善執行的速度)

git fsck (Git維護檔案系統是否完整)

git clone 儲存庫Url (將儲存庫整各複製下來)

git show id (查看各object id 資訊)

查看狀態：=================================

git status (查看目前狀態,查看目前最新版與索引檔的差異!!)

git status –s (查看目前狀態精簡版)

查看記錄：=================================

git log (查看記錄)

git log –n (n=>任意數字，可查看近期幾筆的記錄)

git log --pretty=oneline (查看記錄說明及完整commitid)

git log --pretty=oneline --abbrev-commit (僅查看記錄說明)

建立git 儲存庫：=================================

git init (建立儲存庫)

add到索引檔(staged or cached)：=================================

git add . (將工作目錄中所有檔加入索引檔)

git add app (將app的檔案加入索引檔…)

git add \*.txt (將副檔名為txt的檔案加入索引檔)

git add -u (只將工作目錄下有異動的檔案加入索引檔裡)

放置儲存庫(Depository)：=================================

git commit -m "註記" (索引檔與儲存庫比對差異後，將差異提交成ㄧ個commit物件)

刪除檔案操作：=================================

git rm '\*.txt' (工作目錄下移除該檔file，)

git rm --cached 檔案 (工作目錄下不移除該檔file，移除索引檔內的該檔file

變更檔名稱操作：=================================

git mv 'oldname' 'newname' (變更檔案名稱)

還原操作：=================================

git reset (將git 不論甚麼狀態都還原回來，but mv 或者 rm 都沒辦法還原回來)

git reset --hard (一次性全部還原儲存庫目前最新狀態)

git ls-files (列出目前已儲存在索引檔的檔案路徑)

分支操作：=================================

git checkout -b [branch\_name] (-b=>branch 建立新的分支並且切換此分支上!!)

git checkout [branch\_name] (切換此分支)

git checkout master  'filename' (將儲存庫的某個檔案還原到工作目錄下)

git checkout objects內的檔案 (此為讓工作目錄為哪一階段的，但無detached HEAD，因此commit會錯誤!!，必須再建一個分支出來才可操作!!!!!，不建議這麼作……)

git branch (查看已存在分支)

git branch [branch\_name] (建立此新的分支，但未切換!!)

git branch -d [branch\_name] (移除此分支，但須不在此分支下操作!!)

查看commitid內容：=================================

git cat-file -p [ref or object\_id] (查看commitid資訊)(:id至少要4碼以上)

比對差異性：=======================================

git diff (工作目錄 vs 索引)

git diff commitid (工作目錄 vs commit id)

git diff HEAD (工作目錄 vs HEAD ,HEAD:此分支最新的tree物件)

git diff --cached commitid(索引 VS 指定的commitid)

git diff --cached HEAD (索引 vs HEAD)(--cached 也可寫為--staged)

git diff --cached (索引 vs HEAD)(--cached 也可寫為--staged)

git diff HEAD^ HEAD (HEAD^ vs HEAD , ^:前ㄧ個版本)

git diff commit1 commit2 (commit1 vs commit2)(:其中commit1為較舊版本 commit2為較新

版本)

建立一般參照：=================================

git update-ref ㄧ般參照名稱 object\_id (將object\_id建立一般參照名稱)

git update-ref refs\ㄧ般參照名稱 [object\_id] (正式的參照名稱)

git update-ref -d ㄧ般參照名稱 (移除此一般參照名稱)

建立符號參照：=================================

git symbolic-ref 符號參照 一般參照 (符號參照)

git show-ref(將參照顯示出來)

「符號參照」會指向另一個「一般參照」，而「一般參照」則是指向一個 Git 物件的「絕對名稱」。

暫存操作：=================================

git stash 會將所有已列入追蹤(tracked)的檔案建立暫存版，將

此分支還原最新的HEAD，也可寫成 git stash save

git stash -u 會包括所有tracked或untracked的檔案，全部都建立成暫存版，將

此分支還原最新的HEAD，也可寫成 git stash save -u

(兩者執行後，會再.git\refs\資料夾下建立stash參考名稱<一般參考>)

git stash list 查看所有暫存檔

git stash pop 重新合併(會取最近的暫存檔版本還原並且刪除此暫存檔)

git stash pop “stash@{id}” (取id版暫存檔版本還原並且刪除此暫存檔)

git stash apply重新合併(會取最近的暫存檔版本還原但不會刪除此暫存檔)

git stash apply “stash@{id}” 其中od 為查看你有幾個版本的暫存檔(還原第id版但不會刪除)

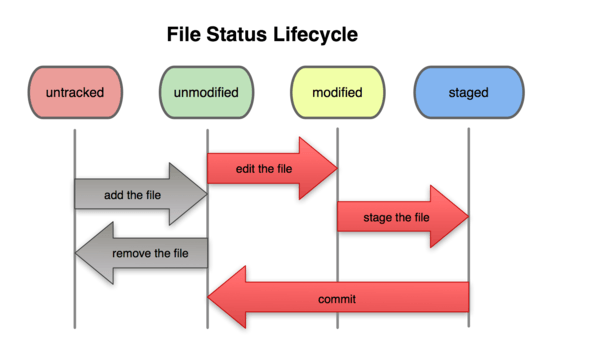
git stash drop “stash@{id}” 刪除id版本的暫存檔

git stash clear 刪除所有暫存檔

git fetch (取回所有遠端儲存庫的物件)

===================================================================

## 索引檔說明



untracked (未追蹤的，代表尚未被加入 Git 儲存庫的檔案狀態)  
unmodified (未修改的，代表檔案第一次被加入，或是檔案內容與

HEAD 內容一致的狀態)  
modified (已修改的，代表檔案已經被編輯過，或是檔案內容與 HEAD

內容不一致的狀態)  
staged (等待被 commit 的，代表下次執行 git commit 會將這些檔案

全部送入版本庫)

git status 為目前最新版(Depository)與索引之間的差異!!!

git add 為將目前工作目錄的變更寫入到索引檔裡

git rm filename: 移除工作目錄下的檔案及索引檔下的檔案

git rm --cached filename 僅移除索引檔下的檔案

git mv oldfilename newfilename 變更檔案名稱，並且同時更新索引檔與工作

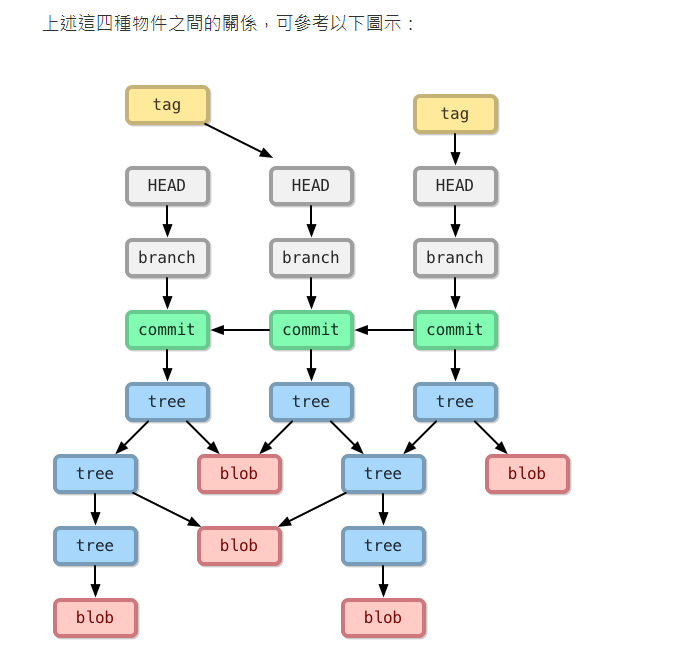
目錄下的實體檔

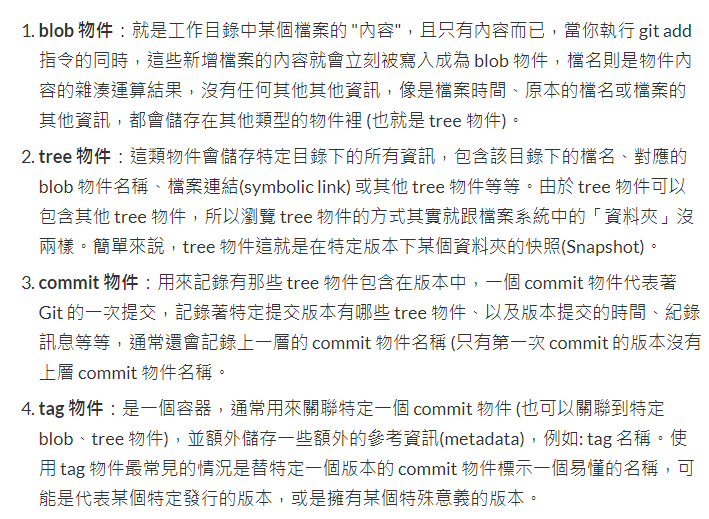
git commit 比對目前最新版與索引檔的差異，將差異部分建立一個

commit物件

git ls-files 可列出所有已存在索引檔的檔案路徑

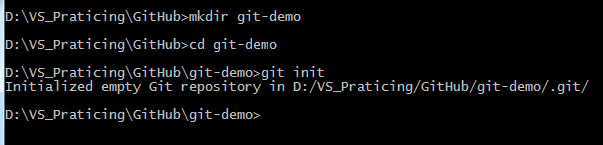
## 物件各分類說明



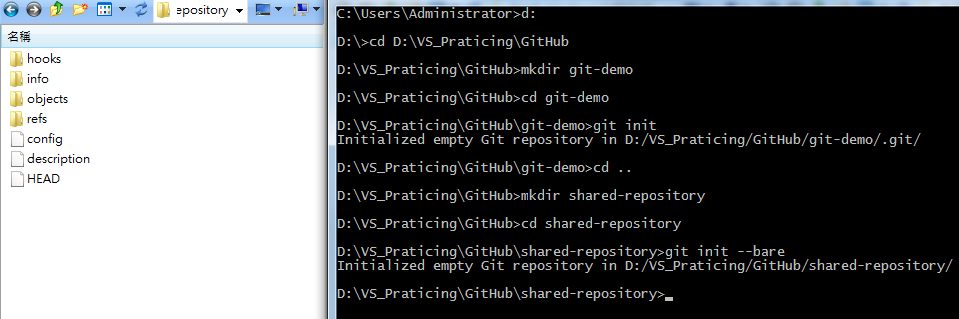


## 重要圖示解說

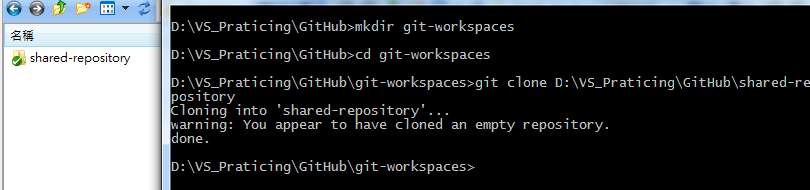
建立工作目錄及儲存庫



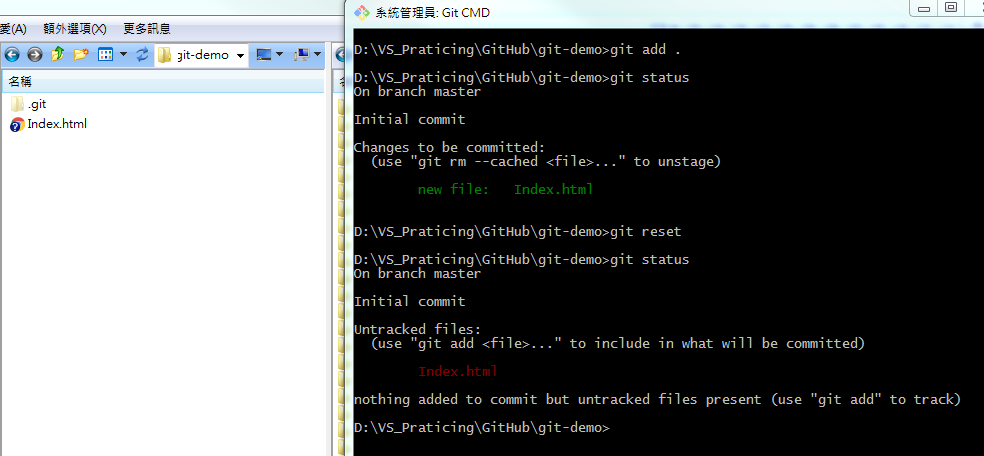
建立共用儲存庫(shared-repository)



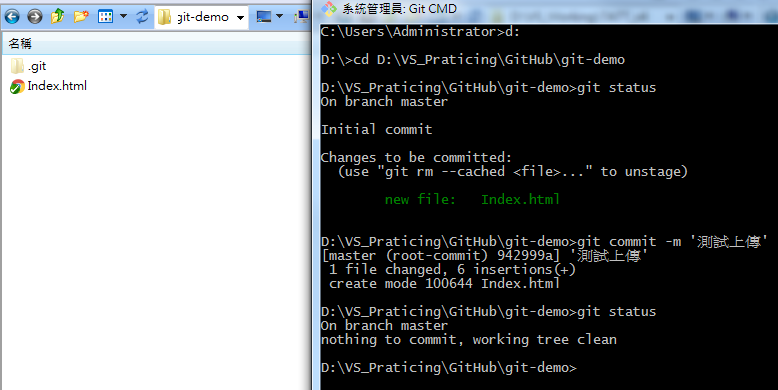
如何將共用儲存庫下載至工作目錄下



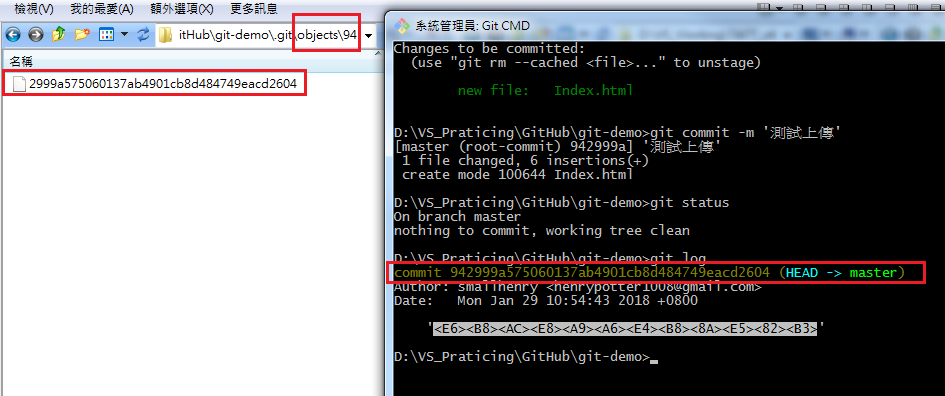
如何將檔案從工作目錄載入索引



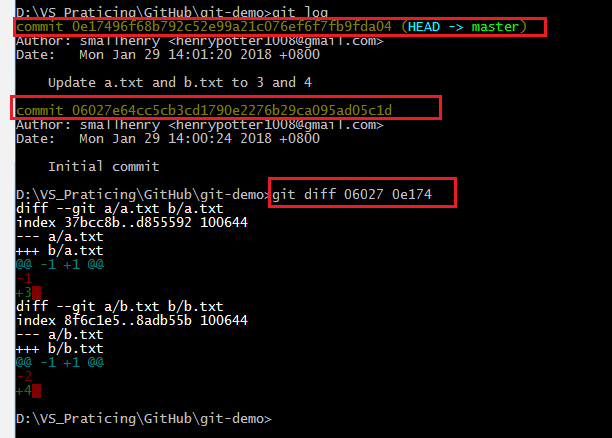
如何將檔案從工作目錄底下的檔案放置儲存庫



查看上傳記錄(注意分支)



比對差異性



--git 屬性

====================================================================

0100000000000000 (040000): Directory

1000000110100100 (100644): Regular non-executable file

1000000110110100 (100664): Regular non-executable group-writeable file

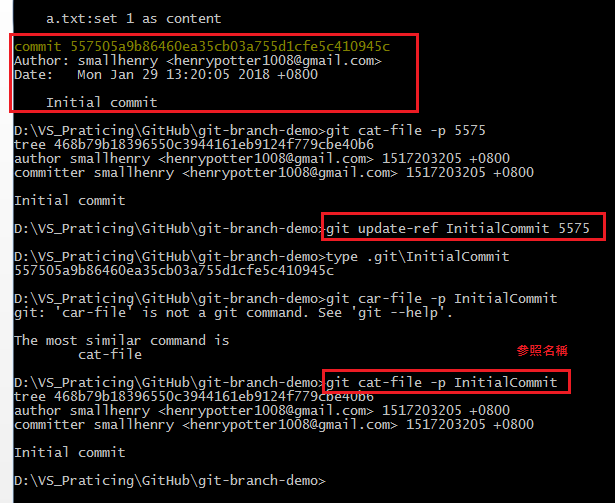
1000000111101101 (100755): Regular executable file

1010000000000000 (120000): Symbolic link

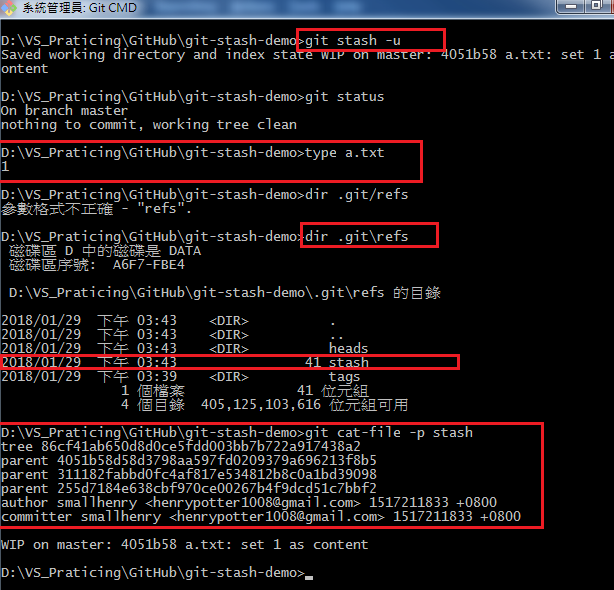
1110000000000000 (160000): Gitlink

====================================================================

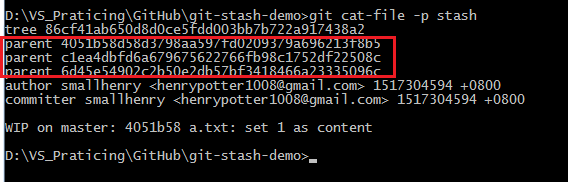
如何建立一般參照名稱

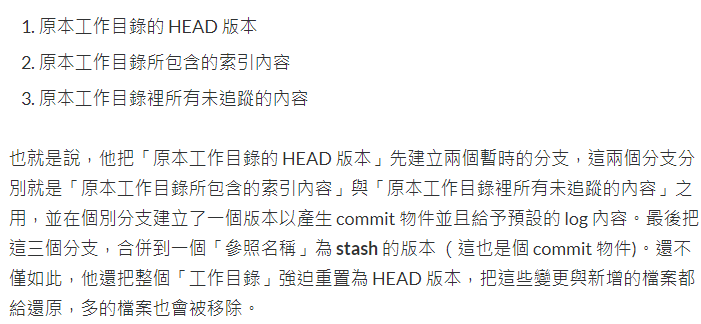


建立暫存檔，並且將工作目錄還原還原此分支最新commit物件

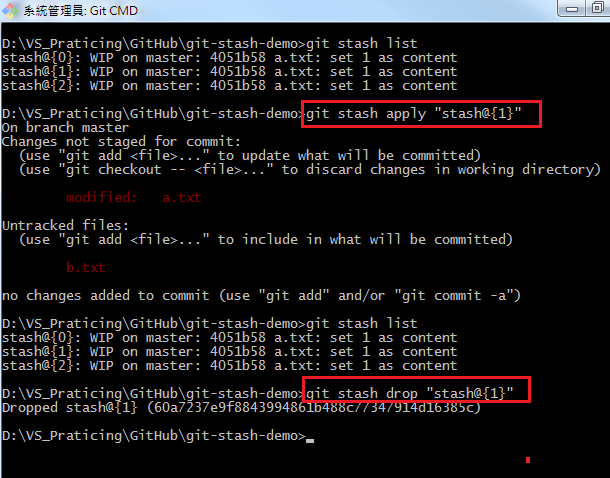


建立暫存檔，會有三個commitID組成一個一般參照stash….





暫存檔操作



## 常見對應詞

Depository: 儲存庫

Working directory: 工作目錄

Index : 索引

Tree: 目錄資訊

Blob: 檔案內容

Object storage(objects): 物件儲存區

\* Cache (快取)

\* Directory cache (目錄快取)

\* Current directory cache (當前目錄快取)

\* Staging area (等待被 commit 的地方)

\* Staged files (等待被 commit 的檔案)

\* Branch (分支)

絕對名稱(commit\_id、tree\_id、blob\_id….)

參照名稱(refs)

符號參照名稱(symref)(ex: HEAD為該分支最新版的commit物件)

====================================================================

## git實體檔

====================================================================

objects : blob與tree物件存放區域(將檔案內容透過SHA1轉為二進位編碼，前

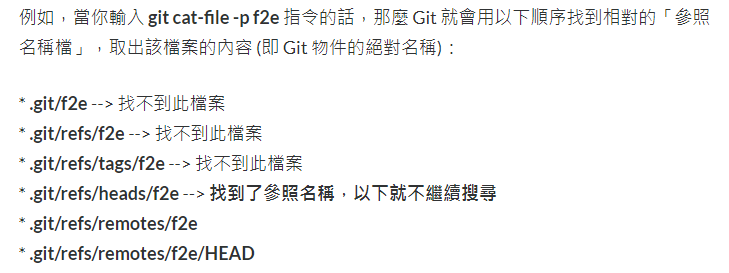
兩碼為資料夾名稱，其餘為檔案名稱，兩者為此物件的id(絕對名稱))

refs : 使用參照名稱(ex: branch\_name )

\* 本地分支：\*\*.git/refs/heads/\*\*

\* 遠端分支：\*\*.git/refs/remotes/\*\*

\* 標 籤：\*\*.git/refs/tags/\*\*



符號參照名稱 (symref)



index : 索引檔

====================================================================