第 09 天：比對檔案與版本差異

使用任何版本控管軟體的過程中，經常會需要查看歷史紀錄與比對版本之間的差異。而在使用 Git 的時候要如何進行比對，將是本文重點。

準備工作目錄

我們透過以下指令快速建立一個擁有兩個檔案與兩個版本變更紀錄的 Git 儲存庫與工作目錄：

```
mkdir git-demo
cd git-demo
git init

echo 1 > a.txt
echo 2 > b.txt
git add .
git commit -m "Initial commit"

echo 3 > a.txt
echo 4 > b.txt
git add .
git commit -m "Update a.txt and b.txt to 3 and 4"
```

關於 git diff 的基本觀念

在 Git 中比對兩個版本之間的差異，通常會用 `git diff` 命令，我們先執行一個簡單的命令，比對兩個版本之間的差異：

1. 先執行 `git log` 取得版本資訊，並取得最近兩個 commit 物件的 id
2. 我們在執行 `git diff commit1 commit2` 指令，比對兩個版本間的差異，其中 commit1 請用較舊的版本，而 commit2 則用較新的版本。

如下圖示：

```
C:\git-demo>git log
commit 8adecbb4a36710f74c64371b3d2377e58c126fb1
Author: Will <doggy.huang@gmail.com>
Date:   Tue Oct 8 17:38:20 2013 +0800

    Update a.txt and b.txt to 3 and 4

commit 53c1ffd5215203a2a1dd4902a0c4942d7943ac55
Author: Will <doggy.huang@gmail.com>
Date:   Tue Oct 8 17:38:19 2013 +0800

    Initial commit

C:\git-demo>git diff 53c1f 8adec
diff --git a/a.txt b/a.txt
index 37bcc8b..d855592 100644
--- a/a.txt
+++ b/a.txt
@@ -1 +1 @@
-1
+3
diff --git a/b.txt b/b.txt
index 8f6c1e5..8adb55b 100644
--- a/b.txt
+++ b/b.txt
@@ -1 +1 @@
-2
+4
```

我們從 `git diff` 執行的輸出結果，將可得到一個執行的結果。由於我們這兩個版本庫中有兩個檔案，而且在這兩個版本之間也都有異動，所以他會列出兩段「差異比對」的結果。

各位可以從上圖看到每一段都是以 `diff --git` 開頭，代表 git 對哪兩個檔案進行比對。

第二行的 `index 37bcc8b..d855592 100644` 則是代表 git 在做這次比對時的「標頭資訊」(Header Line)，這裡可能會有好幾行，資訊不一定只有這些。這裡會標示許多關於此次差異比對的額外資訊。例如 index 這行，後面的兩個 hash id (`37bcc8b..d855592`) 就代表在 Git 物件儲存庫(object storage)中的兩個 blob 物件 id，用來比較這兩個 blob 物件。在後面的 `100644` 則是 git 屬性，有點類似 Linux 環境下的檔案屬性，例如宣告這是個檔案、目錄、可讀、可寫、可執行之類的。以下是幾個常見的 git 屬性範例：

```
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
```

相關連結可參考以下討論串：

- How to read the mode field of git-ls-tree's output
- index-format.txt

接下來第三行的 `--- a/a.txt` 則代表兩個比對的版本中「比較舊的」那個版本。

接下來第四行的 `+++ b/a.txt` 則代表兩個比對的版本中「比較新的」那個版本。

接下來第五行的 `@@ -1 +1 @@` 則代表這個檔案在舊版的總行數與新版的總行數，-1 代表舊版只有 1 行，+1 代表新版也只有 1 行。

最後則是列出所有變更的內容，這裡有三種可能的表示法：

- 以減號 - 號開頭，代表從舊版到新版的過程中，此行被刪除了。
- 以加號 + 號開頭，代表從舊版到新版的過程中，此行是被新增上去的。
- 以空白字元開頭，則代表這一行在兩個版本中都有出現，沒有任何變更。

如此一來就完成了這兩個版本中第一個 blob 物件的差異比對，接著會顯示該版本中第二個 blob 物件的差異比對，以此類推。

在 Git 中使用 `git diff` 的時候，事實上是以 tree 物件為比較的單位，我們從【第 06 天：解析 Git 資料結構 - 物件結構】文章圖解與影片中有學到，其實每一個 commit 物件都會包括一個根目錄的 tree 物件。所以我們剛剛利用 `git diff` 比對兩個 commit 物件時，其實比對的是 commit 物件下的那個 tree 物件，而比對的過程又會遞迴的一直比下去。由此你應該可以感受到，Git 的 diff 比對機制十分強大，你可以很快速的比對出任意兩個版本之間的異動比較。

在使用 `git diff` 命令時，主要有三種 tree 物件的來源，分別是：

- 在所有的 commit graph 中存在的 tree object，也就是任意版本中任意一個 tree 物件的意思。
- 索引 (index)，代表你已經將檔案狀態送進「索引資料庫」的那些資訊，此時透過 `git add` 命令時，其實 tree 物件已經被建立。
- 你目前的工作目錄 (working directory)，雖然工作目錄的改變還沒有變成 tree 物件，但透過 `git diff` 是可以這樣用的。

四種基本的比較方式

要透過 `git diff` 命令比對任意兩個版本，通常會有以下四種指令的用法：

1. git diff

   在什麼參數都不加的使用情況，比對的是「工作目錄」與「索引」之間的差異。這是個很常用的指令，因為當你執行 `git add .` 指令之前，先透過 `git diff` 查看你自己到底改了哪些東西。

   註：事實上，在使用 Git 版本控管的過程中，在執行 `git commit` 之前，的確有可能會執行 `git add` 指令好幾次，用以確認到底哪些檔案要加入到索引之中，最後才會 commit 進版本。

2. git diff commit

   如果你只在 `git diff` 之後加上一個 commit id，比對的是「工作目錄」與「指定 commit 物件裡的那個 tree 物件」。

   最常用的指令是 `git diff HEAD` ，因為這代表你要拿「工作目錄」與「當前分支的最新版」進行比對。這種比對方法，不會去比對「索引」的狀態，所以各位必須區分清楚，你到底比對的是甚麼 tree 物件的來源。

3. git diff --cached commit

   在執行 `git commit` 之前，索引狀態應該已經都準備好了。所以如果你要比對「當前的索引狀態」與「指定 commit 物件裡的那個 tree 物件」，就可以用這個指令完成比對任務。

   最常用的指令一樣是 `git diff --cached HEAD` ，這個語法代表的是「當前的索引狀態」與「當前分支的最新版」進行比對。這種比對方法，不會去比對「工作目錄」的檔案內容，而是直接去比對「索引」與「目前最新版」之間的差異，這有助於你在執行 `git commit` 之前找出那些變更的內容，也就是你將會有哪些變更被建立版本的意思。

   註1: `git diff --cached` 與 `git diff --staged` 是完全一樣的結果， `--staged` 只是 `--cached` 的別名，讓你比較好記而已!

   註2: `git diff --cached` 與 `git diff --cached HEAD` 執行時也是完全一樣的結果，最後的 HEAD 可以省略。

4. git diff commit1 commit2

   最後一種則是透過兩個不同的版本 (commit id) 來比對其差異，這個命令可以跳過「索引」與「工作目錄」的任何變更，而是直接比對特定兩個版本。事實上 Git 是比對特定兩個版本 commit 物件內的那個 tree 物件。

   最常用的指令則是 `git diff HEAD^ HEAD` 命令，這代表你要比較【最新版的前一版】與【最新版】之間的差異。這裡的 HEAD 與 ^ 的意義，我們會在日後的文章中說明。

今日小結

今天介紹的 `git diff` 是個很常用的指令，各位應該熟練地使用它。我們最後來複習一下其常用指令的差異：

```
git diff                => 工作目錄 vs 索引
git diff HEAD           => 工作目錄 vs HEAD
git diff --cached HEAD  => 索引     vs HEAD
git diff --cached       => 索引     vs HEAD
git diff HEAD^ HEAD         => HEAD^   vs HEAD
```

我重新整理一下本日學到的 Git 指令與參數：

- git log
- git diff
- git diff HEAD
- git diff --cached
- git diff --staged
- git diff HEAD^ HEAD

參考連結

- BASIC SNAPSHOTTING
- git-diff(1) Manual Page

- HOME
- 回目錄
- 前一天：關於分支的基本觀念與使用方式
- 下一天：認識 Git 物件的絕對名稱