Pull 下載更新

上個章節我們介紹了如何把東西推上 GitHub，接下來我們看看怎麼把東西拉回來更新。

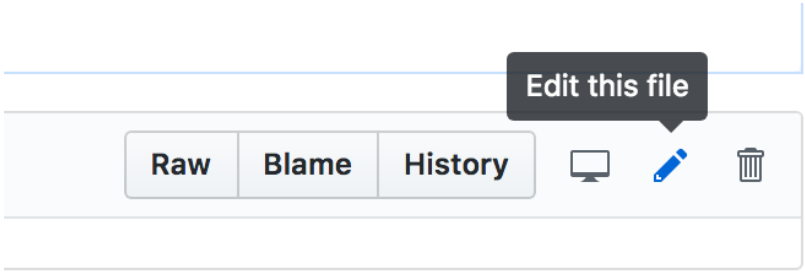
跟 Push 指令相反，Pull 指令是拉回本機更新。但要介紹 Pull 之前，需要先介紹一下 Fetch 這個指令。

Fetch 指令才是把東西拉回來的主角

以上個章節的例子來說（網址：https://github.com/kaochenlong/practice-git)，我們試著執行這個指令：

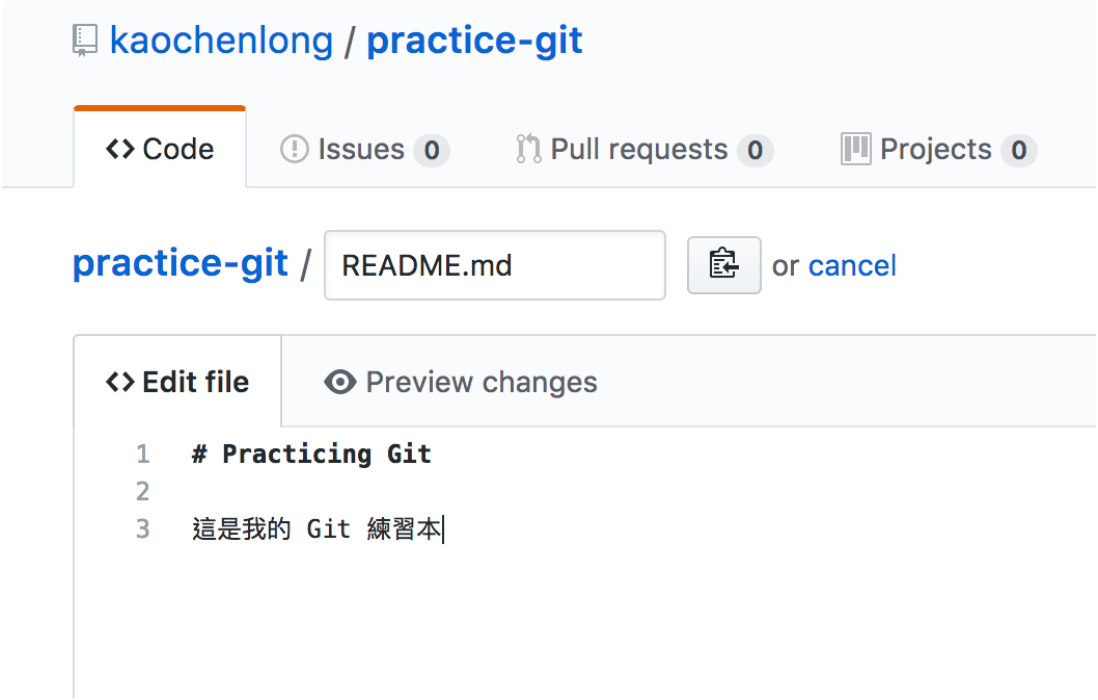
```
$ git fetch
```

你會發現沒有任何訊息，那是因為現在我們的進度跟線上版本是一樣的（廢話，因為就只有我自己一個人啊）。為了營造「有不同進度」的效果，我們可以到 GitHub 網站上，直接在線上編輯某個檔案。例如我點進 `README.md` 檔案，在右上角會有個編輯的按鈕：



edit file

編輯內容如下：

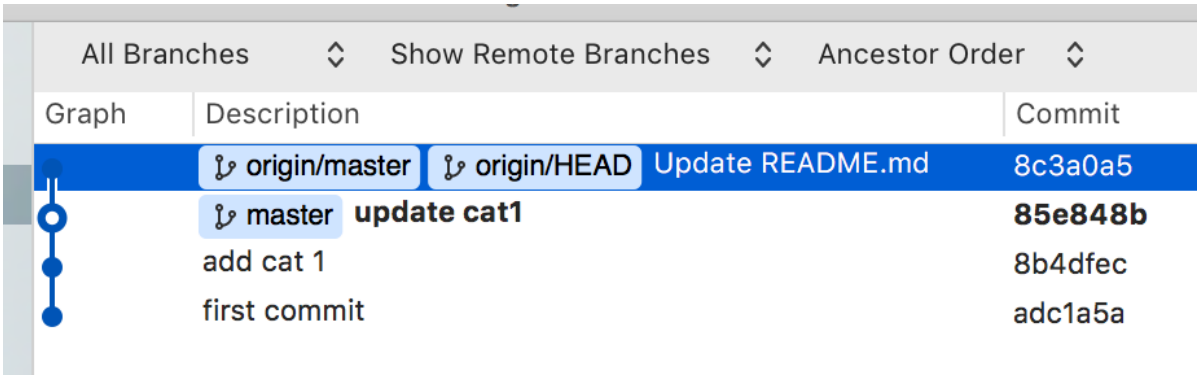


edit file

按下下方的「Commit changes」即可進行存檔並新增一次 Commit，這樣一來線上版本的 Commit 數就領先本機一次了。此時再次執行 Fetch 指令：

```
$ git fetch
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From github.com:kaochenlong/practice-git
   85e848b..8c3a0a5  master     -> origin/master
```

就可以看得出來有拉東西回來了。這時先看一下狀態：

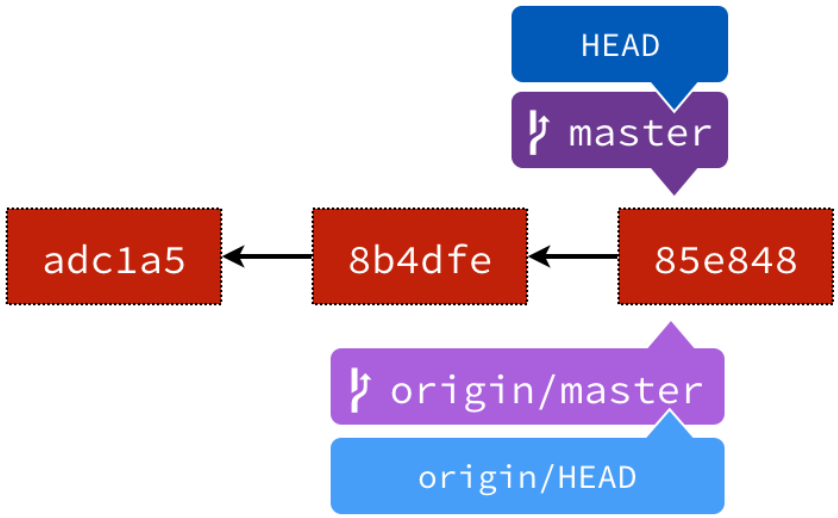


edit file

會發現在 `master` 分支前面，有兩個奇怪的分支分別是 `origin/master` 跟 `origin/HEAD` .. Fetch 的過程到底發生了什麼事呢？

Fetch 過程發生了什麼事？

讓我用畫面來說明一下，這是在 Fetch 前的樣子，HEAD 跟 `master` 分支都不意外的乖乖待在它們該在的位置：

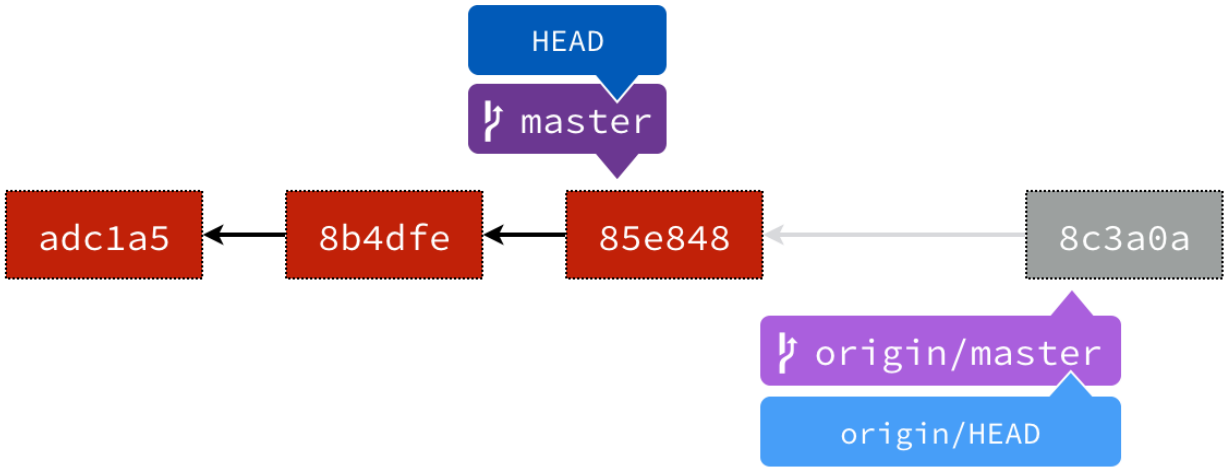


edit file

因為現在這個專案之前有推送東西到 Server 上，所以遠端分支也會記錄一份在本機上，一樣也是有 HEAD 跟 `master` 分支，但會在前面加註遠端節點 `origin`，變成 `origin/HEAD` 跟 `origin/master`。

如果你還記得，我們在第一次推的時候有使用了 `-u` 參數設定 upstream，所以目前這個 `origin/master` 分支其實就是本地 `master` 分支的 upstream 喔。

接下來，當執行 Fetch 指令，Git 看了一下線上版本的內容後，把你目前線上有但你這邊沒有的內容抓了一份下來，同時移動 `origin` 相關的分支：



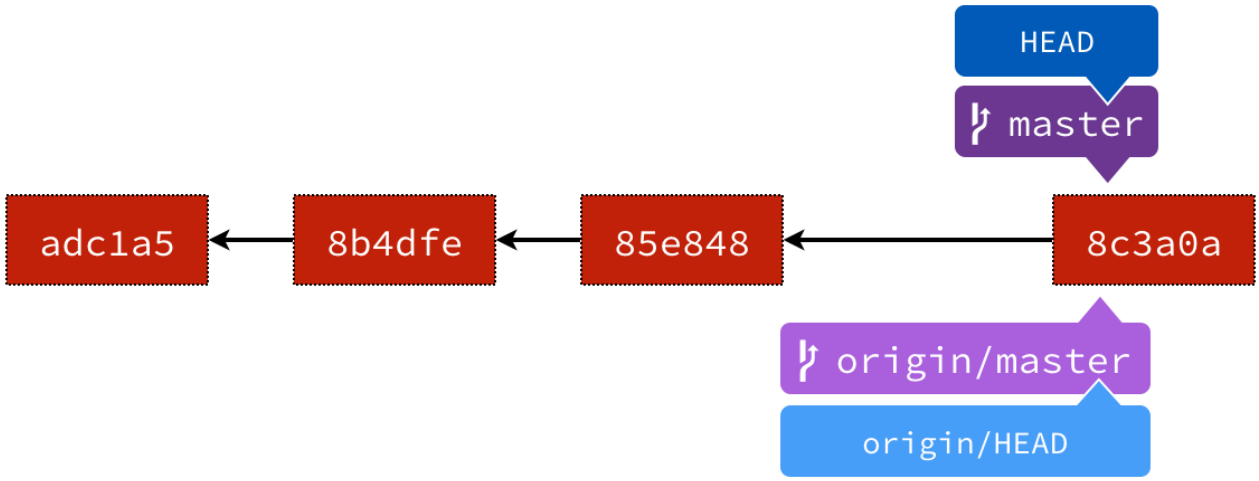
edit file

先不管 `origin/master` 這分支名字有點奇怪，也不用管它是本地還是遠端的分支，對 Git 來說，它就是一個從原本 `master` 分支分出去的分支而已。

既然這個分支是從 `master` 分支分出去而且進度還比 `master` 分支還要新，如果 `master` 分支想要跟上它，這個情境對大家來說不知道會不會有點熟悉？是的，就是合併（merge）啦：

```
$ git merge origin/master
Updating 85e848b..8c3a0a5
Fast-forward
 README.md | 2 ++
 1 file changed, 2 insertions(+)
```

因為 `origin/master` 分支跟 `master` 分支本是同根生，所以在上面合併的過程可以看到是使用快轉模式（Fast Forward）方式進行。關於合併的說明，可參考「合併分支」章節介紹。現在的狀態就會變這樣：



edit file

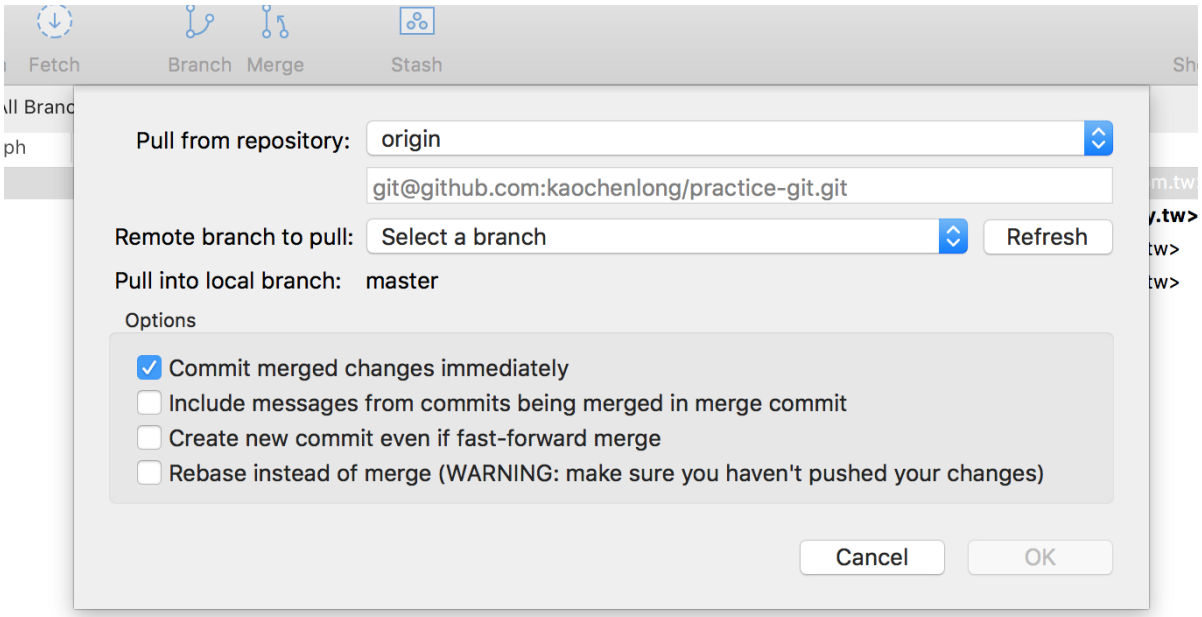
Pull 指令

如果你能理解上面 Fetch 指令在做什麼，那 Pull 指令對你來說就沒什麼問題了，因為：

> git pull = git fetch + git merge

就這樣而已，Pull 指令其實就是去上線抓東西下來（Fetch），並且更新本機的進度（Merge）而已。

如果使用 SourceTree，在上方的工具箱裡就直接有一顆 Pull 跟 Fetch 按鈕，按下 Pull 按鈕後：



pull

在「Remote branch to pull」下拉選單可選擇想要拉的遠端分支。

另外，在下方有一些選項，其中第一個「Commit merged changes immediately」這個選項，如果你知道 Pull 指令其實還有外帶 Merge 效果，現在你應該知道這個選項代表的意思了。

而第三個選項「Create new commit even if fast-forward merge」意思就是「請不要幫我使用快轉模式（Fast Forward）方式合併」，也就是使用 `--no-ff` 參數的意思。

最後一個選項「Rebase instead of merge」嗎？這個可看個人喜好而決定要不要勾，稍後會再說明。

按下 OK 鈕便可完成 Pull 指令。

Pull + Rebase

我們現在知道 Pull 其實等於 Fetch 加上 Merge，而在「另一種合併方式（使用 rebase)」章節也曾介紹過使用 Rebase 方式來合併，在執行 `git pull` 指令的時候，也可以再加上 `--rebase` 參數，它在 Fetch 完成之後，便會使用 Rebase 方式進行合併：

```
$ git pull --rebase
```

這有什麼好處？在多人共同開發的時候，大家都各自在自己的分支進行 Commit，所以拉回來用一般的方式合併的時候常會產生為了合併而產生的額外 Commit（詳情請參閱「合併分支」章節）。為了合併而產生的這個 Commit 本身並沒有什麼問題，但如果你不想要這個額外的 Commit，可考慮使用 Rebase 方式來進行合併。

而在 SourceTree 的 Pull 對話框中，勾選「Rebase instead of merge」選項就會有一樣的效果。