

Summer。桑莫。夏天

前端工程師，喜歡交換明信片、設計簡單的小物、旅遊和看電影。

這裡紀錄了我的學習和開發筆記，歡迎交流 (*'v')~♥

[關於我](#)

[文章列表](#)

[標籤列表](#)

© 2019. All rights reserved.

Git: 更新分支+解衝突

27 May 2018

[git](#) [git rebase](#) [Sourcetree](#)



git






事情是這樣的...

A、B 兩位工程師皆修改同一隻檔案 `hello_world.html` 的同一行程式碼。

- Step 1 : A 新增一個檔案 `hello_world.html`，並加入一些文字段落後提交 (Commit)、推至遠端。
- Step 2 : B 更新本地倉儲，修改 `hello_world.html` line #2 後提交、推至遠端。
- Step 3 : A 修改 `hello_world.html` line #2 後提交，在推至遠端時出現以下訊息...

```
$ git push origin master
To github.com:cythilya/git_test.git
 ! [rejected]          master -> master (fetch first)
error: failed to push some refs to 'git@github.com:cythilya/git_test.g
it'
hint: Updates were rejected because the remote contains work that you
do
hint: not have locally. This is usually caused by another repository p
ushing
hint: to the same ref. You may want to first integrate the remote chan
ges
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for deta
ils.
```

來看 Source Tree 的線圖，這表示本地端的修改尚未提交至遠端，必須先把遠端的進度更新至本地端才行，因此 A 需要做更新的動作。

Graph	Description
	 master 1 ahead 1 behind 修改文章段落
	 origin/master  origin/HEAD 修改段落文字
	 新增文章段落

- Step 4 : A 執行 `git fetch`，到遠端抓取目前本地端所沒有的資料，但不做更新。下載完成後會更新 `origin/master` 所指向的方向。若要更新本地端的

進度，可選擇 Rebase 或 Merge。

備註：Source Tree 可以設定自動更新，就不需要手動打指令 `git fetch`（Settings > Advanced > 勾選 Automatically refresh）。

方法一：Rebase

承 Step 4，做完 `git fetch` 就來做 Rebase。Rebase 意即「重新定義分支的參考基準」。

```
$ git rebase origin/master
```

表示本地端的 master 將以本地的遠端分支 `origin/master` 為參考基準，此時本地分支 `origin/master` 已更新過，進度同遠端倉儲分支 master。

出現以下訊息。

```
First, rewinding head to replay your work on top of it...
Applying: 修改文章段落
Using index info to reconstruct a base tree...
M hello_world.html
Falling back to patching base and 3-way merge...
Auto-merging hello_world.html
CONFLICT (content): Merge conflict in hello_world.html
error: Failed to merge in the changes.
Patch failed at 0001 修改文章段落
```

```
The copy of the patch that failed is found in: .git/rebase-apply/patch
```

```
Resolve all conflicts manually, mark them as resolved with  
"git add/rm <conflicted_files>", then run "git rebase --continue".  
You can instead skip this commit: run "git rebase --skip".  
To abort and get back to the state before "git rebase", run "git rebase --abort".
```

對於程式碼的修改，Git 是可以自動合併不同行的部份，但若是更改同一行的程式碼，就需要人工手動解決衝突——在這裡，由於檔案 `hello_world.html` 都被 A、B 兩人同時更改了同一行程式碼，自動合併失敗，因此需要做人工解衝突的動作。

在有衝突的檔案看到這樣的記號...

- HEAD：由於以 `origin/master` 為參考基準，因此 HEAD 指向的目前本機端程式碼位置，是剛從遠端更新的程式碼。
- `<commit_log>` 「修改文章段落」：本次修改。

```
<<<<<<< HEAD  
<p>How are you?</p>  
=====  
<p>Hello World (*´▽`)~♥</p>  
>>>>>> 修改文章段落
```

決定使用下方（即本次）的修改，刪除標記衝突的記號。

```
<p>Hello World (*'V`)~♥</p>
```

修改好有衝突的檔案後，將變更標記為已解決，然後繼續 Rebase 的操作，讓 Git 重新計算 `<sha-1>` 的值。

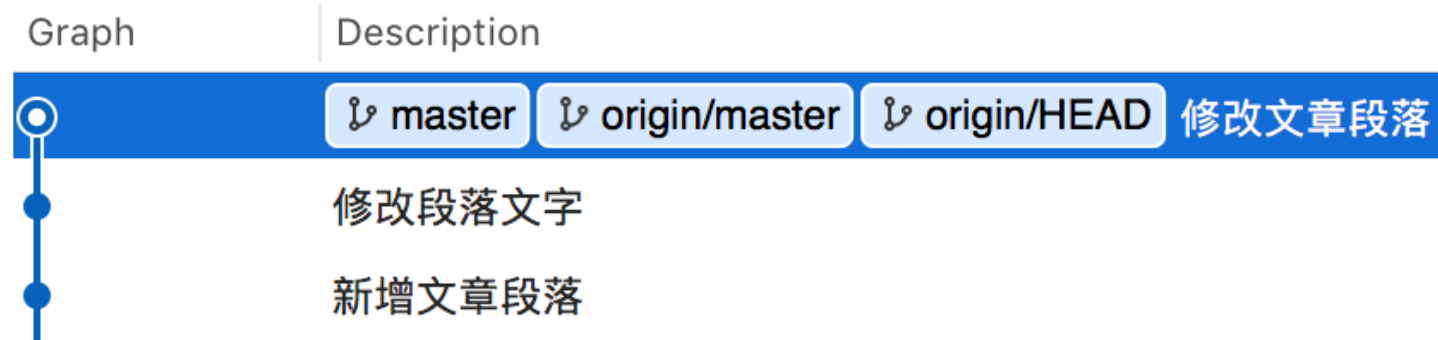
```
$ git add hello_world.html  
$ git rebase --continue
```

Applying: 修改文章段落

最後推到遠端上。

```
$ git push origin master
```

來看 Source Tree 的線圖，這表示成功推到遠端上，同步遠端分支 `origin/master` 和遠端倉儲分支 `master` 的進度。



備註：以上歷程等同於 `git pull --rebase; git push` 。意即，將遠端更新的內容抓下來後，用 Rebase 的方式合併，最後推到遠端上。[看更多-git ready » pull with rebase](#)。

方法二：Merge

承 Step 4，做完 `git fetch` 就來做 Merge。

```
$ git merge
```

```
Auto-merging hello_world.html
CONFLICT (content): Merge conflict in hello_world.html
Automatic merge failed; fix conflicts and then commit the result.
```

手動解衝突，在有衝突的檔案看到這樣的記號...

- HEAD：HEAD 指向目前本機程式碼位置，是本次修改。
- `<sha-1> XXXX.....`：是剛從遠端更新的程式碼。

```
<<<<<<< HEAD
<p>Hello World (*'V`)~♥</p>
=====
<p>How are you?</p>
>>>>>> XXXX.....
```

決定使用上方（即本次）的修改，刪除標記衝突的記號。

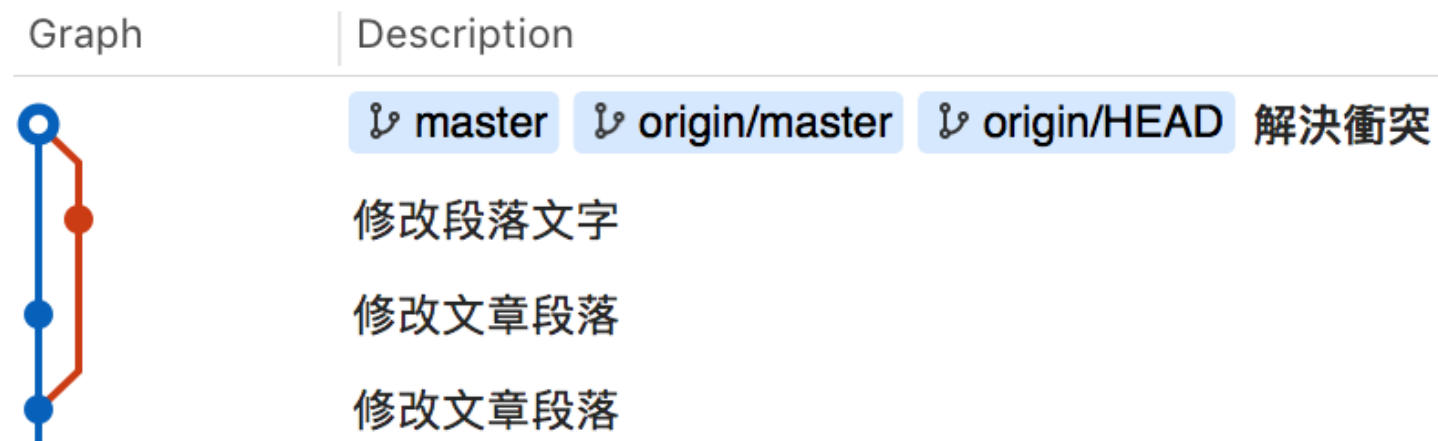
```
<p>Hello World (*`V`)~♥</p>
```

將變更加入暫存區、提交、推至遠端。

```
$ git add hello_world.html  
$ git commit -m "解決衝突"  
$ git push origin master
```

備註：`git fetch + git merge` 等同於 `git pull`。

來看 Source Tree 的線圖，這表示成功推到遠端上，同步遠端分支 `origin/master` 和遠端倉儲分支 `master` 的進度。



注意，使用 `git merge` 會多一個合併的提交記錄，因此會有一個小耳朵（紅線部份）。

總結：Rebase vs Merge

Rebase

Rebase 很像是把本次修改直接貼到新的基準點的後面。在解衝突方面，提交紀錄是一個一個 apply 後來解衝突，所以如果本次修改有很多的提交紀錄，就要做很多次 apply & 解衝突的動作。

在指令的使用上

```
$ git fetch
$ git rebase <remote>/<branch>
$ git push <remote> <branch>
```

說明

- 將資料從遠端取下來。
- 複製本次修改並接在目前 `<remote>/<branch>` 指向的位置，於是 `<branch>` 指向最新的位置。
- 同步遠端狀態，並重新設定 `<remote>/<branch>` 參考的基準點與 `<branch>` 相同。

等同於

```
$ git pull --rebase
$ git push <remote> <branch>
```

Merge

Merge 是將遠端的更新加入本次修改中，所以遠端的提交紀錄會在本次修改提交紀錄之後。並且會多一個合併的提交紀錄（小耳朵），用來記錄合併的來源與合併後的修改，如果沒有修改而只是選擇用哪一個版本，合併時的修改檔案就是空白。

在指令的使用上

```
$ git pull <remote> <branch>
$ git push <remote> <branch>
```

說明

- 將資料從遠端取下來，合併本次修改與遠端狀態，並產生一個合併的 Commit 同時指向兩個 Parent Commit。
- 同步遠端狀態，並重新設定 `<remote>/<branch>` 參考的基準點與 `<branch>` 相同。

等同於

```
$ git fetch
$ git merge <remote>/<branch>
$ git push <remote> <branch>
```

補充

- [取消 Rebase](#)
- [git fetch vs git pull](#)

推薦閱讀

- [Git - 遠端分支](#)
- [為你自己學 Git - 【狀況題】怎麼有時候推不上去...](#)
- [使用 git rebase 避免無謂的 merge](#)

標籤：[git](#) [git rebase](#) [Sourcetree](#)

[comments powered by Disqus](#)

Recent Posts

[Formik & Yup](#) 12 Mar 2019

[一到冬季就好苦惱，臉部肌膚脫皮乾癢該怎麼辦？](#) 09 Mar 2019

[Curry，Decorator 與 HOC](#) 06 Feb 2019

