

## 合併發生衝突了，怎麼辦？

Git 有能力幫忙檢查簡單的衝突，所以並不是改到同一個檔案就一定會發生衝突，但改到同一行就沒辦法了。假設我在 `cat` 分支修改了 `index.html` 的內容如下：

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>首頁</title>
  </head>
  <body>
    <div class="container">
      <div>我是 Cat</div>
    </div>
  </body>
</html>
```

然後在 `dog` 分支剛好也修改了 `index.html`，內容如下：

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>首頁</title>
  </head>
  <body>
    <div class="container">
      <div>我是 Dog</div>
    </div>
  </body>
</html>
```

這時候進行合併，不管是一般的合併或是使用 `Rebase` 進行合併，都會出現衝突，我們先使用一般的合併：

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

Git 發現那個 `index.html` 檔案有問題了，我們先看一下目前的狀態：

```
$ git status
On branch cat
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Changes to be committed:

  new file:   dog1.html
  new file:   dog2.html

Unmerged paths:
  (use "git add <file>..." to mark resolution)

  both modified: index.html
```

說明一下：

1. 對 `cat` 分支來說，`dog1.html` 跟 `dog2.html` 是新來的檔案，但已被放置至暫存區。
2. 但是 `index.html` 這個檔案因為兩邊都修改到了，所以 Git 把它標記成「both modified」狀態。

使用 SourceTree 來看：

The screenshot shows a Git GUI interface. At the top, there's a navigation bar with 'All Branches', 'Hide Remote Branches', and 'Ancestor Order'. Below this is a table of commits. The first commit is 'Uncommitted changes' with a commit hash of '\*'. The second commit is 'dog update index page' with hash 'ed06d49'. The third commit is 'add dog 2' with hash '053fb21'. The fourth commit is 'add dog 1' with hash 'b69eb62'. The fifth commit is 'cat update index' with hash 'a44fe9f'. The sixth commit is 'add 123' with hash '9c2353c'. The seventh commit is 'add cat 2' with hash 'b174a5a'. Below the commit list, there's a section for 'Pending files, sorted by path'. It shows 'Staged files' with a checkmark and 'Unstaged files' with an unchecked checkbox. Under 'Staged files', there are three items: 'dog1.html' with a green plus icon, 'dog2.html' with a green plus icon, and 'index.html' with an orange warning icon. Under 'Unstaged files', there is one item: 'index.html' with an orange warning icon. To the right of the 'index.html' file, there's a diff view for 'Hunk 1 : Lines 6-16'. The diff shows the following changes:

Line	Old	New
6	6	</head>
7	7	<body>
8	8	<div class="container">
9		+ <<<<<< HEAD
9	10	<div>我是 Cat</div>
11		+ =====
12		+ <div>我是 Dog</div>
13		+ >>>>>> dog
10	14	</div>
11	15	</body>
12	16	</html>

可以看到那個有衝突的檔案用驚嘆號標記出來了。

## 解決問題

看看那個 `index.html` 的內容長什麼樣子：

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>首頁</title>
  </head>
  <body>
    <div class="container">
      <<<<<< HEAD
      <div>我是 Cat</div>
      =====
      <div>我是 Dog</div>
    </div>
  </body>
</html>
```

```
>>>>>> dog
    </div>
  </body>
</html>
```

Git 把有衝突的段落標記出來了，上半部是 `HEAD`，也就是目前所在的 `cat` 分支，中間是分隔線，接下是 `dog` 分支的內容。

那要解決這個問題？這問題看來是溝通不良造成的，所以遇到問題，當然是解決有問題的人...不是，是把兩邊的人請過來討論一下，到底是該用誰的 code。經過一番討論後，決定還是要採納 `cat` 分支的內容，順便把那些標記修掉，最後內容如下：

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>首頁</title>
  </head>
  <body>
    <div class="container">
      <div>我是 Cat</div>
    </div>
  </body>
</html>
```

修改完後，別忘了把這個檔案加回暫存區：

```
$ git add index.html
```

然後就可以 Commit，完成這一回合：

```
$ git commit -m "conflict fixed"
[cat a28a93c] conflict fixed
```

## 如果是使用 Rebase 的合併造成衝突？

衝突這回事，不管是一般合併或是 Rebase，會打架就是會打架，不會因為合併方式而就不會衝突。但 Rebase 的過程如果發生衝突會跟一般的合併不太一樣。例如：

```
$ git rebase dog
First, rewinding head to replay your work on top of it...
Applying: add cat 1
```

```

Applying: add cat 2
Applying: add 123
Applying: update index
Using index info to reconstruct a base tree...
M       index.html
Falling back to patching base and 3-way merge...
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
error: Failed to merge in the changes.
Patch failed at 0004 update index
The copy of the patch that failed is found in: .git/rebase-apply/patch

When you have resolved this problem, run "git rebase --continue".
If you prefer to skip this patch, run "git rebase --skip" instead.
To check out the original branch and stop rebasing, run "git rebase --abort".

```

這時候其實是卡在一半，從 SourceTree 可以看得更清楚：

Commit	File status	Graph	Description	Commit
			<b>Uncommitted changes</b>	*
			<b>HEAD</b> add 123	3a5a802
			add cat 2	e2ec267
			add cat 1	a53cf1e
			<b>dog</b> update index page	ed06d49
			add dog 2	053fb21
			add dog 1	b69eb62
			<b>cat</b> update index	a44fe9f
			add 123	9c2353c
			add cat 2	b174a5a
			add cat 1	c68537b
			<b>master</b> add database.yml in confi...	e12d8ef
			add hello	85e7e30

**HEAD** 現在並沒有指著任何一個分支，它現在有點像是在修改歷史的時候卡在某個時空縫隙裡了（其實是 3a5a802 這個 Commit）。看一下目前的狀態：

```

$ git status
rebase in progress; onto ed06d49
You are currently rebasing branch 'cat' on 'ed06d49'.
(fix conflicts and then run "git rebase --continue")
(use "git rebase --skip" to skip this patch)

```

```
(use "git rebase --abort" to check out the original branch)
```

Unmerged paths:

```
(use "git reset HEAD <file>..." to unstage)
```

```
(use "git add <file>..." to mark resolution)
```

```
both modified:   index.html
```

```
no changes added to commit (use "git add" and/or "git commit -a")
```

訊息寫著 `rebase in progress`，而且那個 `index.html` 的確也是被標記成「both modified」狀態。跟上面提到的方法一樣，把 `index.html` 有衝突的內容修正完成後，把它加回暫存區：

```
$ git add index.html
```

搞定，接著繼續完成剛剛中斷的 Rebase：

```
$ git rebase --continue
Applying: update index
```

這樣就算完成 Rebase 了。

## 那如果不是文字檔的衝突怎麼解？

上面的 `index.html` 因為是文字檔案，所以 Git 可以標記出發生衝突的點在哪些行，我們用肉眼都還能看得出來大概該怎麼解決，但如果是像圖片檔之類的二進位檔怎麼辦？例如在 `cat` 分支跟 `dog` 分支，同時都加了一張叫做 `cute_animal.jpg` 的圖片，合併的時候出現衝突的訊息：

```
$ git merge dog
warning: Cannot merge binary files: cute_animal.jpg (HEAD vs. dog)
Auto-merging cute_animal.jpg
CONFLICT (add/add): Merge conflict in cute_animal.jpg
Automatic merge failed; fix conflicts and then commit the result.
```

這下糟了，要把兩邊的人馬請過來，討論到底誰才是最可愛的動物。討論後決定貓才是這世上最可愛的動物，所以決定要用 `cat` 分支的檔案：

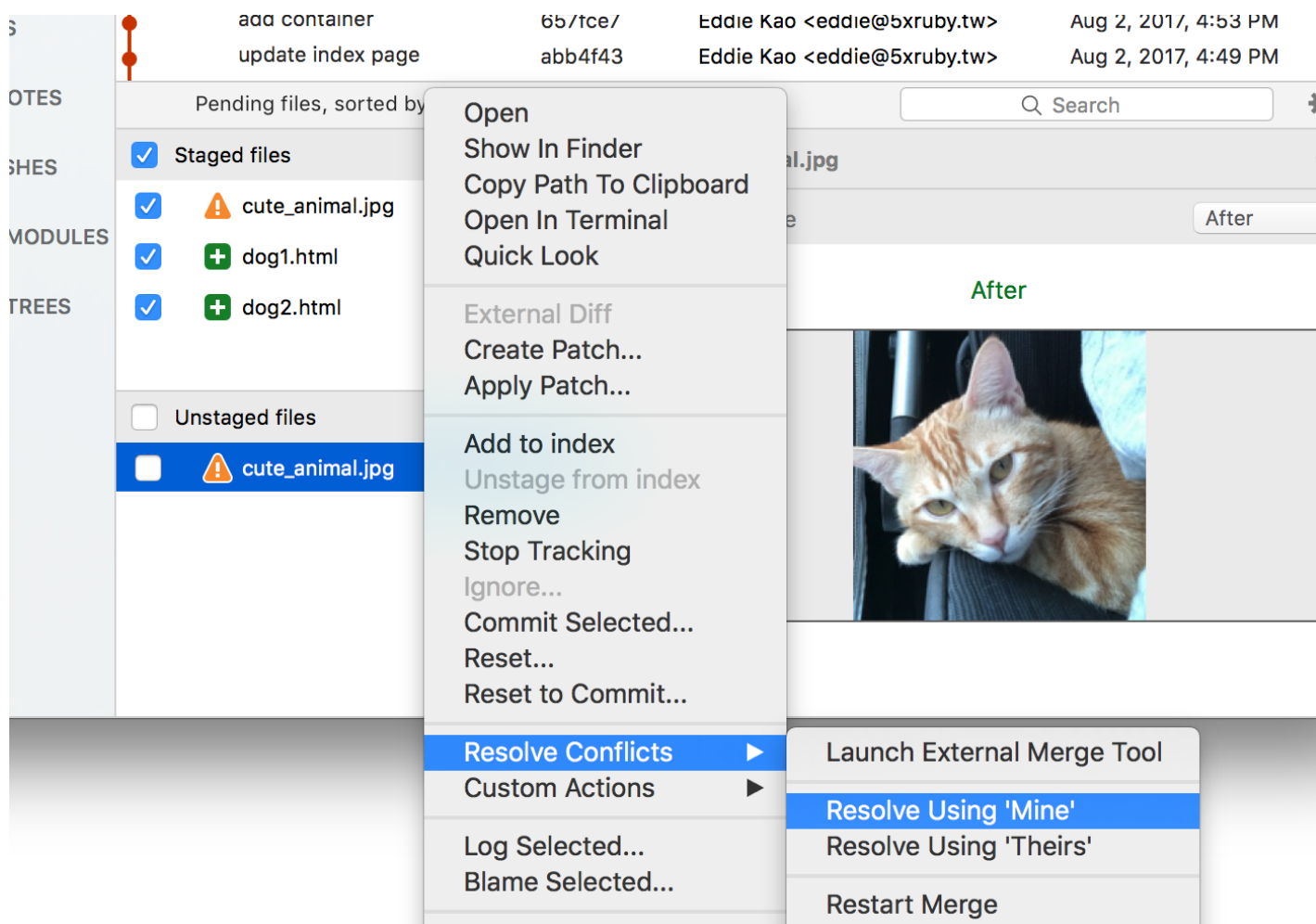
```
$ git checkout --ours cute_animal.jpg
```

如果是要用對方（`dog` 分支），則是使用 `--theirs` 參數：

```
$ git checkout --theirs cute_animal.jpg
```

決定之後，就跟前面一樣，加到暫存區，準備 Commit，然後結束這一回合。

如果使用 SourceTree，可在那個有衝突的檔案上按滑鼠右鍵，選擇「Resolve Conflicts」→「Resolve Using 'Mine'」等同於上面使用 `--ours` 參數的效果：



如果是選擇「Resolve Using 'Theirs'」則等同使用 `--theirs` 參數。

