

Git (/zh-tw/tutorial/git/)

安裝Git (/zh-tw/tutorial/git/install-git/)

初始化本地倉庫 (/zh-tw/tutorial/git/initialize-the-repository/)

Git工作流程 (/zh-tw/tutorial/git/git-workflow/)

比較版本差異 (/zh-tw/tutorial/git/git-diff-to-compare-differences/)

Git檔案操作 (/zh-tw/tutorial/git/git-file-operation/)

撤銷修改 (/zh-tw/tutorial/git/unstage-files/)

版本分支 (/zh-tw/tutorial/git/git-branches/)




Git合併分支 (/zh-tw/tutorial/git/merge-branches/)

Git Rebase (/zh-tw/tutorial/git/git-rebase/)

教程 (/zh-tw/tutorial/) / Git (/zh-tw/tutorial/git/) / Git合併分支 (/zh-tw/tutorial/git/merge-branches/)

Git合併分支

Created: August-15, 2018

 (<http://www.facebook.com/sharer.php?src=bm&u=https%3a%2f%2fwww.delftstack.com%2fzh-tw%2ftutorial%2fgit%2fmerge-branches%2f&t=Git%e5%90%88%e4%bd%b5%e5%88%86%e6%94%af>)  (<https://twitter.com/intent/tweet/?text=Git - Git%e5%90%88%e4%bd%b5%e5%88%86%e6%94%af&url=https%3a%2f%2fwww.delftstack.com%2fzh-tw%2ftutorial%2fgit%2fmerge-branches%2f&hashtags=web,development>)  (<https://www.linkedin.com/shareArticle?mini=true&url=https%3a%2f%2fwww.delftstack.com%2fzh-tw%2ftutorial%2fgit%2fmerge-branches%2f&title=Git - Git%e5%90%88%e4%bd%b5%e5%88%86%e6%94%af&source=https%3a%2f%2fwww.delftstack.com%2fzh-tw%2ftutorial%2fgit%2fmerge-branches%2f&summary=Short%20summary>)  (<https://plus.google.com/share?url=https%3a%2f%2fwww.delftstack.com%2fzh-tw%2ftutorial%2fgit%2fmerge-branches%2f>)

我們將在本節教程中學習如何合併分支以及如何解決可能的衝突。

在上一教程中，我們學到了建立了一個新的分支來實現一些新功能，並且不會弄亂 master 分支。

在我們建立這些新分支並對新功能感到滿意之後，我們需要將其合併回 master 分支。這樣，當我們釋出下一個版本程式碼時，新功能會被包含其中。

Fast-forward 快進合併

假設我們對 new_test_branch 分支的結果感到滿意，我們會將其合併回 master 分支。

01. 切換成 master 分支

```
$ git checkout master
```

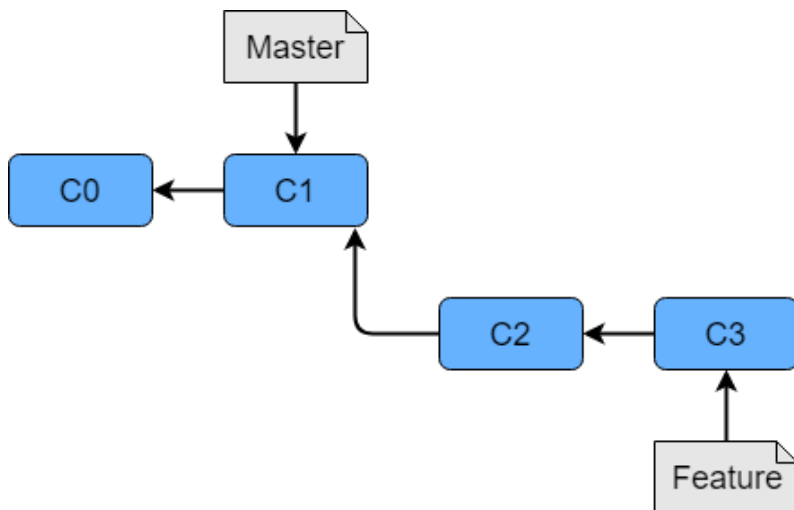
02. 將新分支合併到 master 分支

```
$ git merge new_test_branch
Updating 5fef94e..e7a7e81
Fast-forward
 test3.txt | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
```

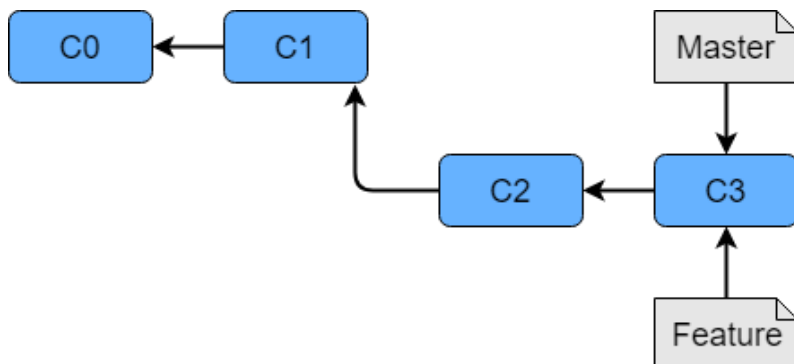
我們從分支 new_test_branch 中得到了新功能並將其合併到此 master 分支中。

在上面訊息中顯示的 Fast-forward 意味著 git 合併的時候進行的是快進合併，也就是直接把 master 指向 test_branch 的最新提交，所以稱之為快進合併。

這是未合併之前的情形，



在快進合併以後，master 指標直接移動到 test_branch 的最新提交上。



Recursive 遞迴合併

在我們介紹這種不同的 git 合併策略之前，我們需要建立兩個分支 test_branch_A 和 test_branch_B 並且對它們進行不同的更改。然後我們將合併 test_branch_A 回到 master 分支。

現在我們來把 test_branch_B 分支合併到 master 。

```
$ git merge test_branch_B
Merge made by the 'recursive' strategy.
 test1_rename.txt | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
```

這種合併策略和 fast-forward 快進合併之間的區別在於合併是由遞迴策略進行的，因為此分支程式碼中沒有整合 test_branch_A 分支中的新特性。換句話說，master 分支在 test_branch_B 建立後被修改了。

現在 master 分支將兩個新特性分支都合並進來了，所以是否不同開發人員同時處理不同特性分支，這對 git 來說並不重要，只要最後能將它們合併回 master 就好。

解決衝突

我們在上面的兩個測試分支中修改的內容沒有任何衝突，但是如果分枝間存在衝突時，我們將某分支合併到 master 時要怎麼辦呢？

讓我們建立兩個分支 test_branch_C 和 test_branch_D，然後附加文字 The letter should be C 到分支 test_branch_C 中的檔案 test3.txt 中，然後附加文字 The letter should be D 到分支 test_branch_D 的檔案 test3.txt 中。

然後我們將 test_branch_C 合併到 master 分支中。

現在，我們將合併 test_branch_D 到 master 中，

```
$ git merge test_branch_D
Auto-merging test3.txt
CONFLICT (content): Merge conflict in test3.txt
Automatic merge failed; fix conflicts and then commit the result.
```

你可以在此處看到它無法成功合併，因為它檢測到檔案 test3.txt 中的衝突。我們需要通過手動編輯衝突檔案來解決衝突。

該檔案 test3.txt 顯示衝突為，

```
<<<<<< HEAD
The letter should be C
=====
The letter should be D
>>>>>> test_branch_D
```

HEAD 和 ===== 之間的文字是 master 分支的內容，它來自已經合並進來的 test_branch_C 分支，這部分文字不同於待合併分支 test_branch_D 中的對應文字。

你需要選擇將要最終合併到 master 分支中的文字，並刪除所有符號，比如 <<<<< 和 >>>>> 。

比如說，這部分文字應該是 The letter should be D。我們將衝突區域更新為 The letter should be D。
解決衝突後，我們可以通過使用 `git commit` 提交解決好的衝突。

Top Self-Learning Python Books

[◀ 版本分支 \(/ZH-TW/TUTORIAL/GIT/GIT-BRANCHES/\)](/ZH-TW/TUTORIAL/GIT/GIT-BRANCHES/)

[GIT REBASE ▶](#)



Copyright © 2018. All right reserved

[關於本站 \(/zh-tw/about-us/\)](/zh-tw/about-us/)

[隱私政策 \(/zh-tw/privacy-policy/\)](/zh-tw/privacy-policy/)

[聯絡方式 \(/zh-tw/contact/\)](/zh-tw/contact/)