# 分支管理策略

通常，合并分支时，如果可能，Git会用Fast forward模式，但这种模式下，删除分支后，会丢掉分支信息。

如果要强制禁用Fast forward模式，Git就会在merge时生成一个新的commit，这样，从分支历史上就可以看出分支信息。

首先，创建并切换新分支dev

$ git checkout -b dev

Switched to a new branch 'dev'

然后修改example.docx文件，加一句：dev branch，并提交一个新的commit;

$ git add example.docx

$ git commit -m "add merge"

[dev 8abb7ba] add merge

1 file changed, 0 insertions(+), 0 deletions(-)

我们切换回master:

$ git checkout master

Switched to branch 'master'

Your branch is up-to-date with 'origin/master'.

准备合并dev分支，注意使用--no-ff参数，表示禁用Fast forward:

$ git merge --no-ff -m "merge with no-ff" dev

Merge made by the 'recursive' strategy.

example.docx | Bin 0 -> 11588 bytes

1 file changed, 0 insertions(+), 0 deletions(-)

然后使用带了一堆参数的git log看看分支历史：

$ git log --graph --pretty=oneline --abbrev-commit

\* 2ef27de (HEAD -> master) merge with no-ff

|\

| \* e2f7846 (example) no-Fast forward

|/

\* bc70212 dev

\* 923e701 master

图3

删除example分支

$ git log --graph --pretty=oneline --abbrev-commit

\* 2ef27de (HEAD -> master) merge with no-ff

|\

| \* e2f7846 no-Fast forward

|/

\* bc70212 dev

\* 923e701 master

对比下使用Fast forward

merge后，删除dev前

$ git log --graph --pretty=oneline --abbrev-commit

\* bc70212 (HEAD -> master, dev) dev

\* 923e701 master

图1

merge后，删除dev后

$ git log --graph --pretty=oneline --abbrev-commit

\* bc70212 (HEAD -> master) dev

\* 923e701 master

图2

在实际开发中，我们应该按照几个基本原则进行分支管理：

首先，master分支应该是非常稳定的，也就是仅用来发布新版本，平时不能在上面干活；

那在哪干活呢？干活都在dev分支上，也就是说，dev分支是不稳定的，到某个时候，比如1.0版本发布时，再把dev分支合并到master上，在master分支发布1.0版本；

你和你的小伙伴们每个人都在dev分支上干活，每个人都有自己的分支，时不时地往dev分支上合并就可以了。

图4