**Git 少用 Pull 多用 Fetch 和 Merge 【已翻译100%】**

英文原文：[git: fetch and merge, don’t pull](http://longair.net/blog/2009/04/16/git-fetch-and-merge/)

标签： [Git](https://www.oschina.net/translate/tag/git)

[彭博](https://my.oschina.net/dtec) 推荐于 4年前 (共 15 段, 翻译完成于 06-28) [评论 5](https://www.oschina.net/translate/git-fetch-and-merge#comments)

分享

收藏

52

参与翻译 (5人) : [Andy](https://my.oschina.net/hand), [赵亮-碧海情天](https://my.oschina.net/theforever), [\_Raymond](https://my.oschina.net/laravel), [super0555](https://my.oschina.net/super0555), [几点人](https://my.oschina.net/jidianren)[仅中文](https://www.oschina.net/translate/git-fetch-and-merge) | [中英文对照](https://www.oschina.net/translate/git-fetch-and-merge?cmp) | [仅英文](https://www.oschina.net/translate/git-fetch-and-merge?lang=eng) | [打印此文章](https://www.oschina.net/translate/git-fetch-and-merge?print)

本文有点长而且有点乱，但就像*Mark Twain*[*Blaise Pascal*](http://en.wikiquote.org/wiki/Blaise_Pascal)*的*笑话里说的那样：我没有时间让它更短些。在Git的邮件列表里有很多[关于本文的讨论](http://thread.gmane.org/gmane.comp.version-control.git/116903/)，我会尽量把其中相关的观点列在下面。

我最常说的关于git使用的一个经验就是：

不要用git pull，用git fetch和git merge代替它。

git pull的问题是它把过程的细节都隐藏了起来，以至于你不用去了解git中各种类型分支的区别和使用方法。当然，多数时候这是没问题的，但一旦代码有问题，你很难找到出错的地方。看起来git pull的用法会使你吃惊，简单看一下git的使用文档应该就能说服你。

将下载（fetch）和合并（merge）放到一个命令里的另外一个弊端是，你的本地工作目录在未经确认的情况下就会被远程分支更新。当然，除非你关闭所有的安全选项，否则git pull在你本地工作目录还不至于造成不可挽回的损失，但很多时候我们宁愿做的慢一些，也不愿意返工重来。

[](https://my.oschina.net/hand)

[Andy](https://my.oschina.net/hand" \o "Andy)

翻译于 4年前

4人顶

顶 翻译得不错哦！

**分支(Branches)**

在说*git pull*之前，我们需要先澄清分支的概念（branches）。很多人像写代码似的用一行话来描述分支是什么，例如:

* 准确而言，分支的概念不是一条线，而类似于开发中的有向无环图
* 分支类似于一个重量级的大对象集合。

我认为你应该这样来理解分支的概念：它是用来标记特定的代码提交，每一个分支通过SHA1sum值来标识，所以对分支进行的操作是轻量级的--你改变的仅仅是SHA1sum值。

[](https://my.oschina.net/hand)

[Andy](https://my.oschina.net/hand" \o "Andy)

翻译于 4年前

3人顶

顶 翻译得不错哦！

这个定义或许会有意想不到的影响。比如，假设你有两个分支，“stable” 和 “new-idea”, 它们的顶端在版本 E 和 F:

A-----C----E ("stable")

\

B-----D-----F ("new-idea")

所以提交(commits) A, C和 E 属于“stable”，而 A, B, D 和 F 属于 “new-idea”。如果之后你用下面的命令 将“new-idea” merge 到 “stable” ：

git checkout stable # Change to work on the branch "stable"

git merge new-idea # Merge in "new-idea"

…那么你会得到这个：

A-----C----E----G ("stable")

\ /

B-----D-----F ("new-idea")

要是你继续在“new idea” 和“stable”分支提交, 会得到：

A-----C----E----G---H ("stable")

\ /

B-----D-----F----I ("new-idea")

因此现在A, B, C, D, E, F, G 和 H 属于 “stable”，而A, B, D, F 和 I 属于 “new-idea”。

当然了，分支确实有些特殊的属性——其中最重要的是，如果你在一个分支进行作业并创建了一个新的提交(commits)，该分支的顶端将前进到那个提交(commits)。这正是你所希望的。当用*git merge*进行合并(merge)的时候，你只是指定了要合并到当前分支的那个并入分支，以及当前分支的当前进展。

[](https://my.oschina.net/super0555)

[super0555](https://my.oschina.net/super0555" \o "super0555)

翻译于 4年前

3人顶

顶 翻译得不错哦！

另一个表明使用分支会有很大帮助的观点的常见情形是：假设你直接工作在一个项目的主要分支（称为“主版本”），当你意识到你所做的可能是一个坏主意时已经晚了，这时你肯定宁愿自己是工作在一个主题分支上。如果提交图看起来像这样：

last version from another repository

|

v

M---N-----O----P---Q ("master")

那么你把你的工作用下面的一组命令分开做（如图显示的是执行它们之后所更改的状态）：

git branch dubious-experiment

M---N-----O----P---Q ("master" and "dubious-experiment")

git checkout master

# Be careful with this next command: make sure "git status" is

# clean, you're definitely on "master" and the

# "dubious-experiment" branch has the commits you were working

# on first...

git reset --hard <SHA1sum of commit N>

("master")

M---N-------------O----P---Q ("dubious-experiment")

git pull # Or something that updates "master" from

# somewhere else...

M--N----R---S ("master")

\

O---P---Q ("dubious-experiment")

这是个看起来我最终做了很多的事情。

[](https://my.oschina.net/theforever)

[赵亮-碧海情天](https://my.oschina.net/theforever" \o "赵亮-碧海情天)

翻译于 4年前

4人顶

顶 翻译得不错哦！

**分支类型**

分支这个术语不太容易理解,而且在git的开发过程中发生了很多变化。但简单来说git的分支只有两种：

a）“本地分支(local branches)” ，当你输入“git branch”时显示的。例如下面这个小例子：

$ git branch

debian

server

\* master

b)“远程跟踪分支(Remote-tracking branches)” ，当你输入“git branch -r”是显示的，如:

$ git branch -r

cognac/master

fruitfly/server

origin/albert

origin/ant

origin/contrib

origin/cross-compile

从上面的输出可以看到，跟踪分支的名称前有一个“远程的”标记名称（如 :origin, cognac, fruitfly）后面跟一个“／”，然后远程仓库里分支的真正名称。（“远程名称”是一个代码仓库别名，和本地目录或URL是一个含义，你可以通过"git remote"命令自由定义额外的“远程名称”。但“git clone”命令默认使用的是“origin”这个名称。）

[](https://my.oschina.net/hand)

[Andy](https://my.oschina.net/hand" \o "Andy)

翻译于 4年前

4人顶

顶 翻译得不错哦！

如果你对分支在本地是如何存储感兴趣的话，看看下面文件： 

* .git/refs/head/[本地分支]
* .git/refs/remotes/[正在跟踪的分支]

两种类型的分支在某些方面十分相似-它们都只是在本地存储一个表示提交的SHA1校验和。（我强调“本地”，因为许多人看到"origin/master" 就认为这个分支在某种意义上说是不完整的，没有访问远端服务器的权限- 其实不是这种情况。）   
不管如何相似，它们还是有一个特别重大的区别： 

* 更改远端跟踪分支的安全方法是使用git fetch或者是作为git-push副产品，你不能直接对远端跟踪分支这么操作。相反，你总得切换到本地分支，然后创建可移动到分支顶端的新提交 。

因此，你对远端跟踪分支最多能做的是下面事情中的一件： 

* 使用git fetch 更新远端跟踪分支
* 合并远端跟踪分支到当前分支
* 根据远端跟踪分支创建本地分支

[](https://my.oschina.net/jidianren)

[几点人](https://my.oschina.net/jidianren" \o "几点人)

翻译于 4年前

2人顶

顶 翻译得不错哦！

**基于远程跟踪分支创建本地分支**

如果你想基于远程跟踪分支创建本地分支（在本地分支上工作）,你可以使用如下命令：*git branch –track*或*git checkout –track -b*，两个命令都可以让你切换到新创建的本地分支。例如你用*git branch -r命令*看到一个远程跟踪分支的名称为“origin/refactored”是你所需要的，你可以使用下面的命令：

git checkout --track -b refactored origin/refactored

在上面的命令里，“refactored”是这个新分支的名称，“origin/refactored”则是现存远程跟踪分支的名称。（在git最新的版本里，例子中‘-track’选项已经不需要了，如果最后一个参数是远程跟踪分支，这个参数会被默认加上。）

[](https://my.oschina.net/hand)

[Andy](https://my.oschina.net/hand" \o "Andy)

翻译于 4年前

2人顶

顶 翻译得不错哦！

“–track”选项会设置一些变量，来保持本地分支和远程跟踪分支的相关性。他们对下面的情况很有用：

* git pull命令下载新的远程跟踪分支之后，可以知道合并到哪个本地分支里
* 使用git checkout检查本地分支时，可以输出一些有用的信息：

Your branch and the tracked remote branch 'origin/master'

have diverged, and respectively have 3 and 384 different

commit(s) each.

或者：

Your branch is behind the tracked remote branch

'origin/master' by 3 commits, and can be fast-forwarded.

允许使用的配置变量是：“branch.<local-branch-name>.merge”和“branch.<local-branch-name>.remote”，但通常情况下你不用考虑他们的设置。

[](https://my.oschina.net/hand)

[Andy](https://my.oschina.net/hand" \o "Andy)

翻译于 4年前

2人顶

顶 翻译得不错哦！

当从远程代码仓库创建一个本地分支之后，你会注意到，“*git branch -r*”能列出很多远程跟踪分支，但你的电脑上只有一个本地分支，你需要给上面的命令设置一个参数，来指定本地分支和远程分支的对应。

有一些术语上的说法容易混淆需要注意一下：“track”在当作参数"-track"使用时，意思指通过本地分支对应一个远程跟踪分支。在远程跟踪分支中则指远程代码仓库中的跟踪分支。有点绕口。。。

下面我们来看一个例子，如何从远程分支中更新本地代码，以及如何把本地分支推送到一个新的远程仓库中。

[](https://my.oschina.net/hand)

[Andy](https://my.oschina.net/hand" \o "Andy)

翻译于 4年前

3人顶

顶 翻译得不错哦！

**从远端仓库进行更新**

如果我想从远端的源仓库更新到本地的代码仓库，可以输入“*git fetch origin*”的命令，该命令的输入类似如下格式：

remote: Counting objects: 382, done.

remote: Compressing objects: 100% (203/203), done.

remote: Total 278 (delta 177), reused 103 (delta 59)

Receiving objects: 100% (278/278), 4.89 MiB | 539 KiB/s, done.

Resolving deltas: 100% (177/177), completed with 40 local objects.

From ssh://longair@pacific.mpi-cbg.de/srv/git/fiji

3036acc..9eb5e40 debian-release-20081030 -> origin/debian-release-20081030

\* [new branch] debian-release-20081112 -> origin/debian-release-20081112

\* [new branch] debian-release-20081112.1 -> origin/debian-release-20081112.1

3d619e7..6260626 master -> origin/master

最重要的是这两行：

3036acc..9eb5e40 debian-release-20081030 -> origin/debian-release-20081030

\* [new branch] debian-release-20081112 -> origin/debian-release-20081112

第一行表明远端的origin/debian-release-20081030分支的提交(commit)ID已经从3036acc更新为9eb5e40。箭头前的部分是远端分支的名称。第二行是我们采取的动作，创建远程跟踪分支（如果远程仓库有新的tags，git fetch也会一并下载到本地）。