# Git 原理详解及实用指南 - 扔物线 - 掘金小册

## 进阶 4：Feature Branching：最流行的工作流

在前面的《上手 2》这一节里，我介绍了一种最基本的团队工作模型。在这种模型里，所有人都工作在 `master` 上，写完了的 `commit` 可以通过 `push` 来发送到中央仓库，并且可以使用 `pull` 来获取到别人的最新 `commit` s。

这种工作模型解决了团队合作最基本的问题：多人并行开发和版本管理。事实上，这也是早期的 VCS——中央式 VCS 的工作模型。

但这种工作模型也有它的限制：使用这种工作模型时，每个人的代码在被大家看到的时候，就是它进

入正式的生产库的时候。所有人的工作都会被直接 `push` 到 `master`，这导致每个人的代码在正式启用前无法被别人看到（严格来讲是有办法的，别

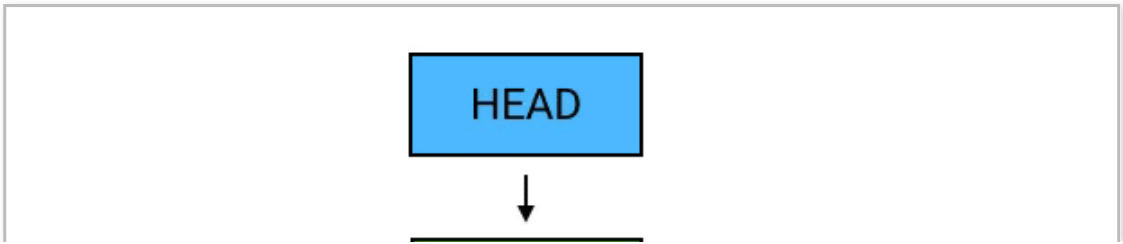
人可以直接从你的电脑上 `pull` ，Git 的「分布式」不是说说的。但——这种做法超级不方便），这样就让代码在正式启用前的讨论和 review（审阅）非常不方便。现在的商业团队，开发项目多是采用「边开发边发布、边开发边更新、边开发边修复」的持续开发策略，所以代码分享的不便会极大地影响团队的开发效率。

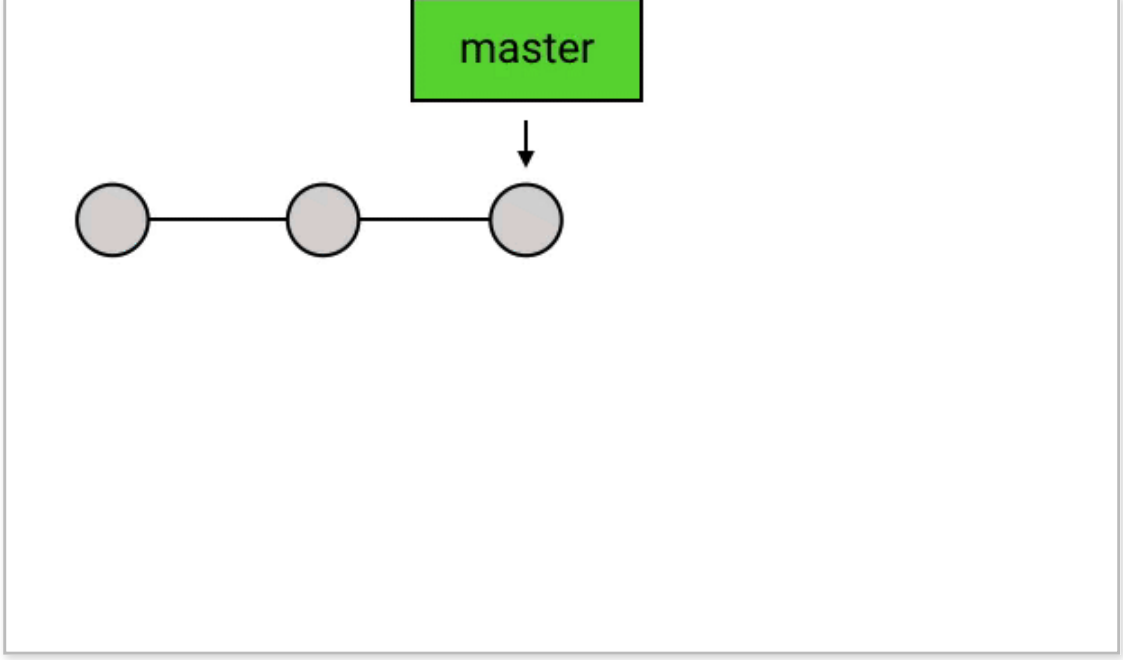
这一节，我将介绍的是目前最流行（不论是中国还是世界）的团队开发的工作流：Feature Branching。

## 简介

这种工作流的核心内容可以总结为两点：

- 任何新的功能（feature）或 bug 修复全都新建一个 `branch` 来写；
- `branch` 写完后，合并到 `master` ，然后删掉这个 `branch` 。






Feature Branching

这就是这种工作流最基本的模型。

从上面的动图来看，这种工作流似乎没什么特别之处。但实质上，Feature Branching 这种工作流，为团队开发时两个关键的问题——代码分享和一人多任务——提供了解决方案。

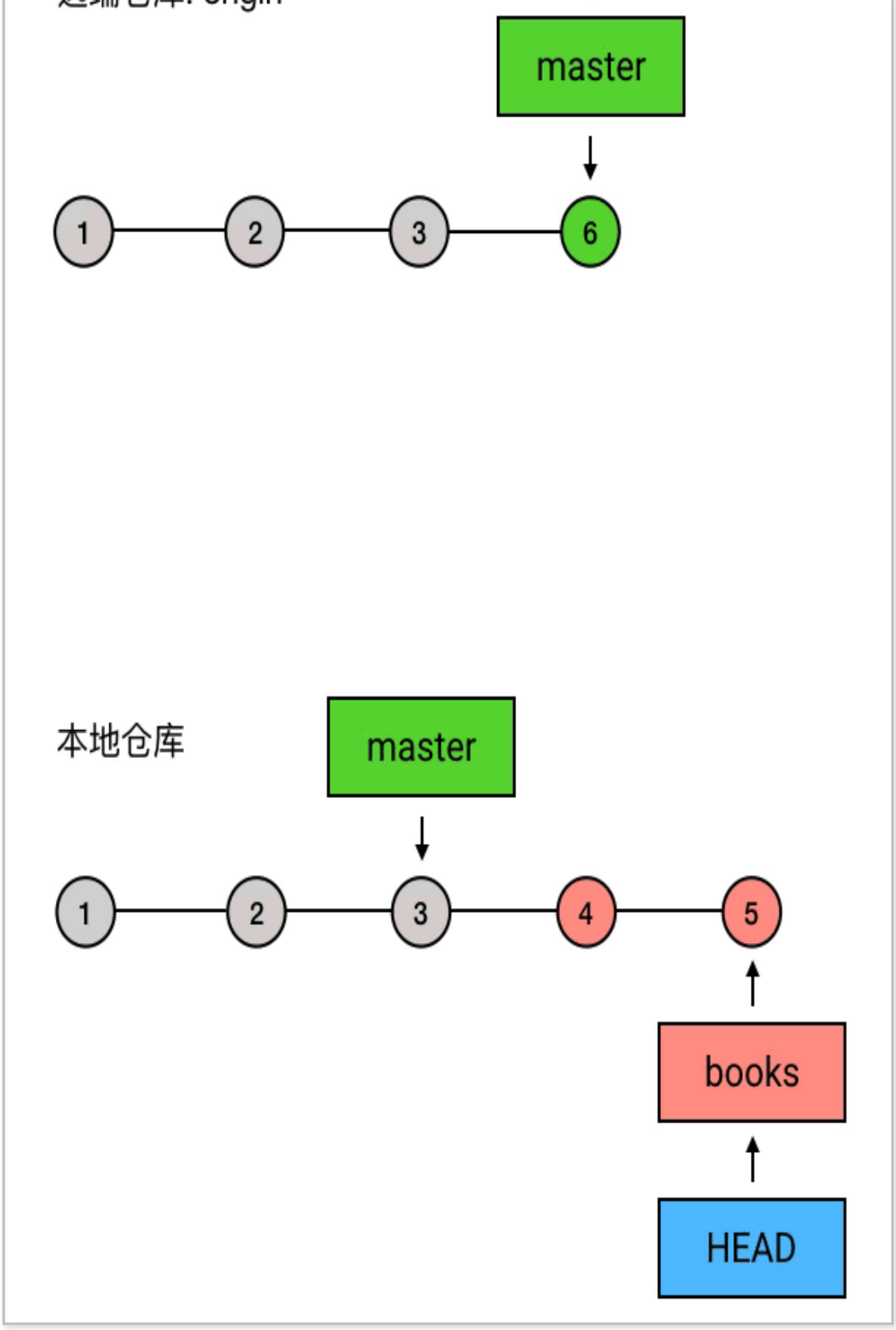
## 1. 代码分享

假设你在一个叫做「掘金」的团队工作，现在你要开发一个叫做「掘金小册」的功能（呵呵），于是你创建了一个新的 `branch` 叫做 `books`，然后开始在 `books` 上进行开发工作。

```
git checkout -b books
```

在十几个 `commit` s 过后，「掘金小册」的基本功能开发完毕，你就把代码 `push` 到中央仓库（例如 GitHub）去，然后告诉同事：「嘿，小册的基本功能写完了，分支名是 `books` ，谁有空的话帮我 review 一下吧。」

```
git push origin books
```




把 books push 到中央仓库

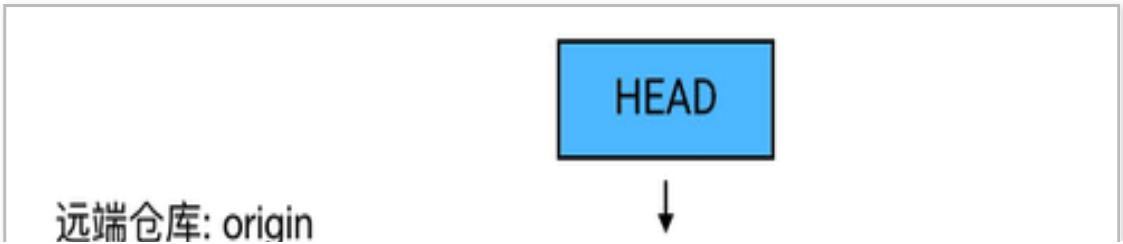
然后你的同事明明正好有空，他就从中央仓库拉下来了你的代码开始读：

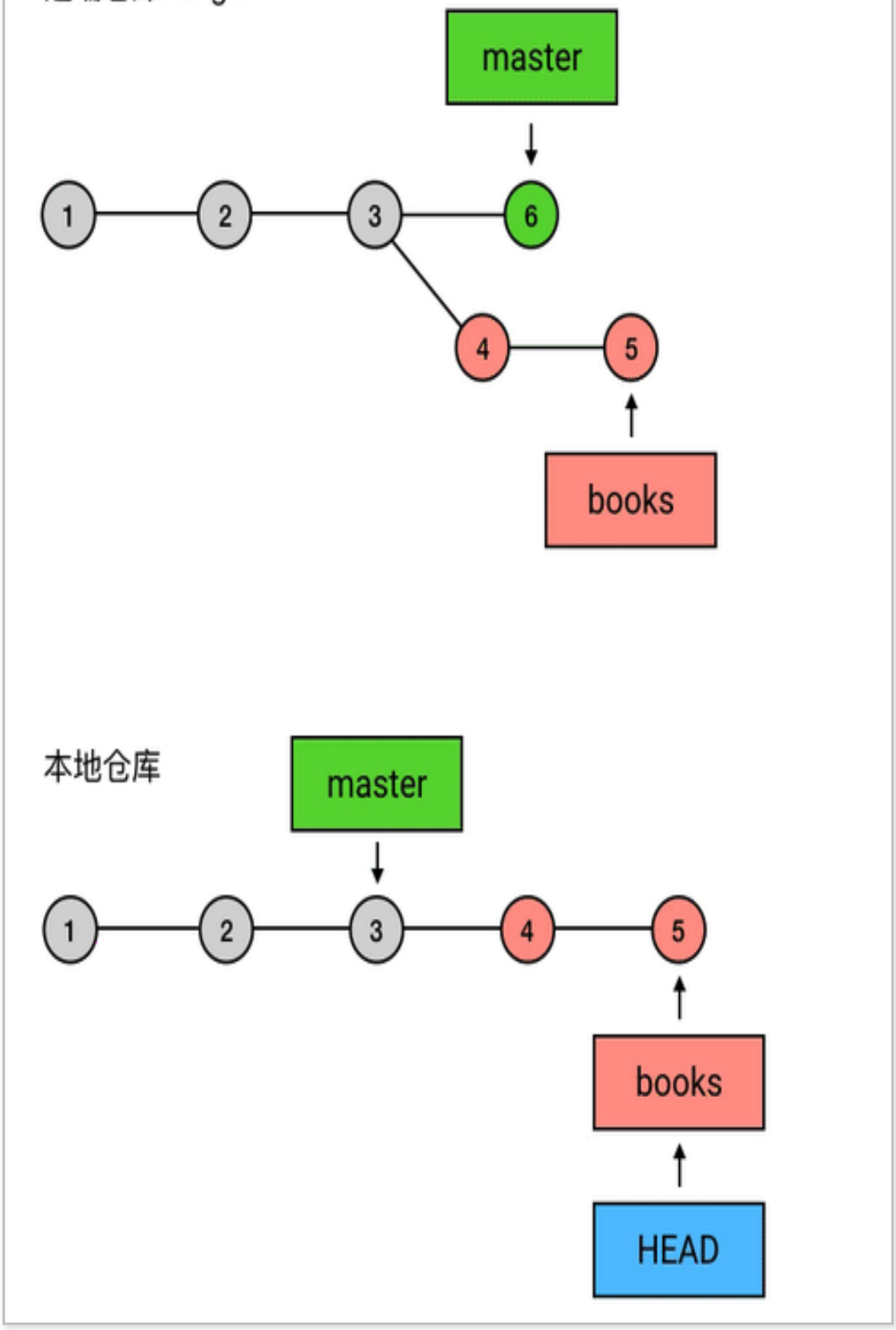
```
# 明明的电脑：
git pull
git chekcout books
```

读完以后，明明对你说说，嗯我看完了，我觉得不错，可以合并到 `master` ！

于是你就把 `books` 合并到了 `master` 上去：

```
git checkout master
git pull # merge 之前 pull 一下，让 master 更新到和
git merge books
```

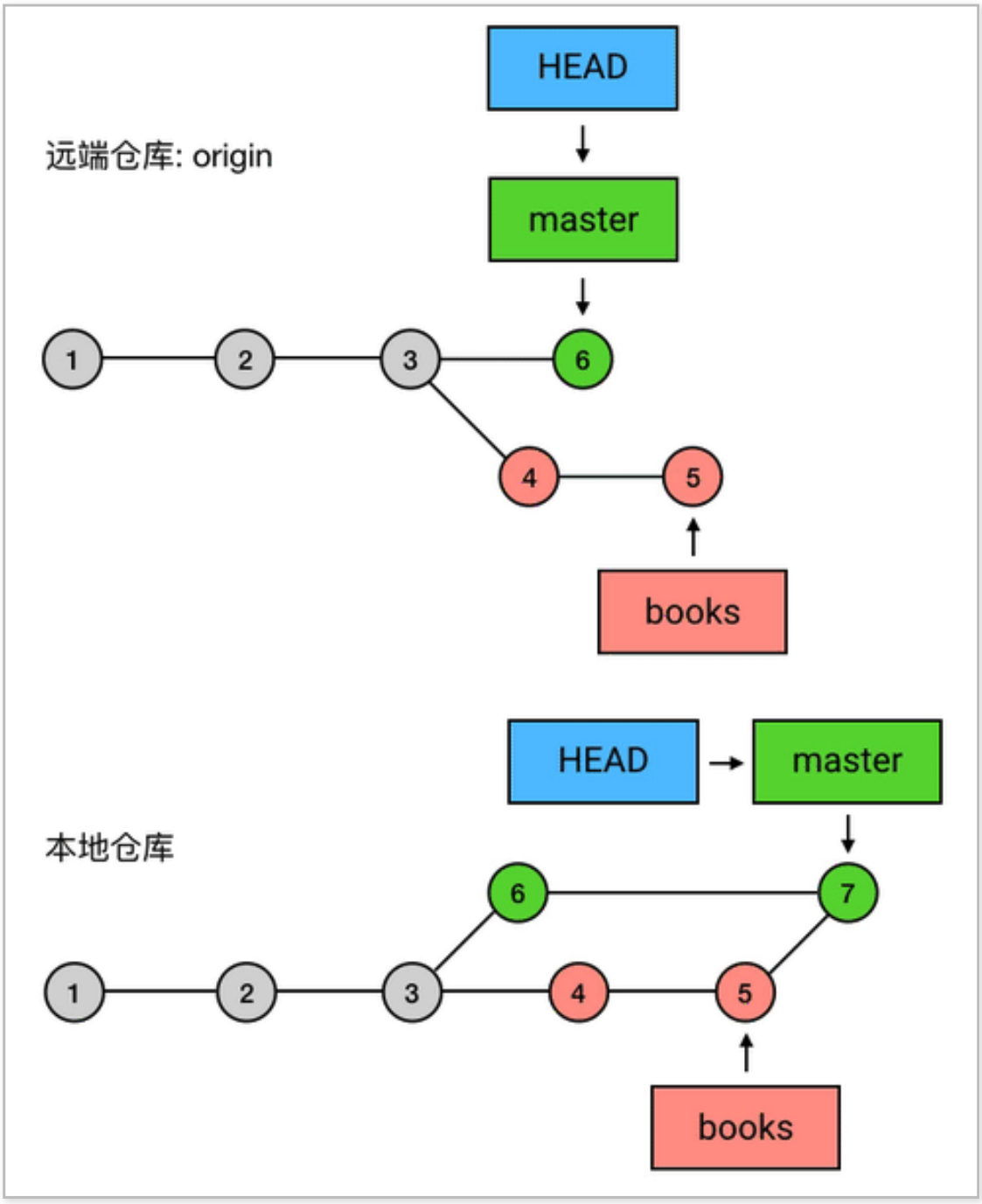





把 books 合并到 master

紧接着，你把合并后的结果 `push` 到了中央仓库，并删掉了 `books` 这个 `branch` ：

```
git push
git branch -d books
git push origin -d books # 用 -d 参数把远程仓库的
```




把 master push 上去，并删除本地和远程的 books

## 如果同事有意见

上面讲的是明明对你的代码没有意见，而假如他在你的代码里看到了问题，例如他跑来对你说：「嘿，你的代码缩进为什么用的是 TAB？快改成空格，不然砍死你哦。」

这时，你就可以把你的缩进改成空格，然后做一个新的提交，再 `push` 上去，然后通知他：「我改完啦！」

明明 `pull` 下来你的新提交看了看：「嗯，这下可以合并了。」

于是你依照上面的那一套操作，把代码合并进 `master` ，并 `push` 了上去，然后删掉了 `books` 。

瞧，代码在同事竖大拇指之前都不会正式发布到 `master` ，挺方便的吧？

## Pull Request

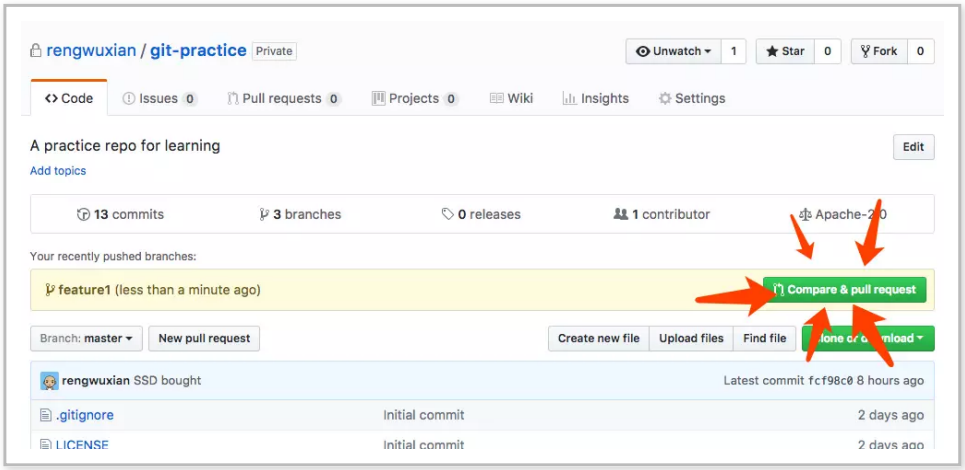
事实上，上面讲的这个流程，还可以利用 Pull Request 来进一步简化。

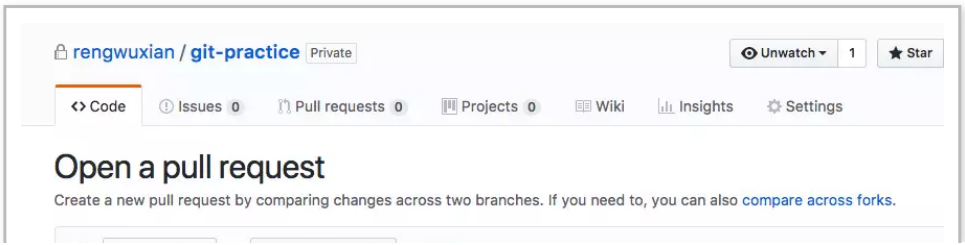
Pull Request 并不是 Git 的内容，而是一些 Git 仓库服务提供方（例如 GitHub）所提供的一种便捷功能，它可以让团队的成员方便地讨论一个

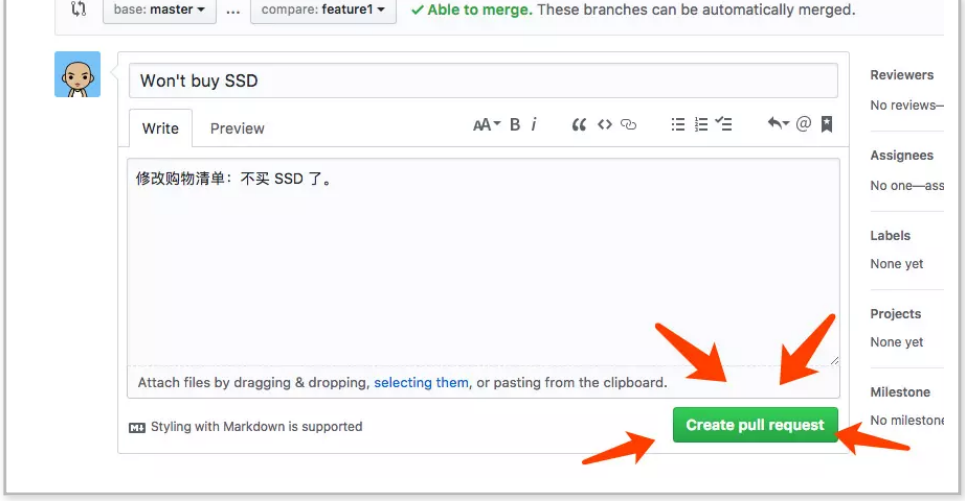
`branch` ，并在讨论结束后一键合并这个 `branch` 到 `master` 。

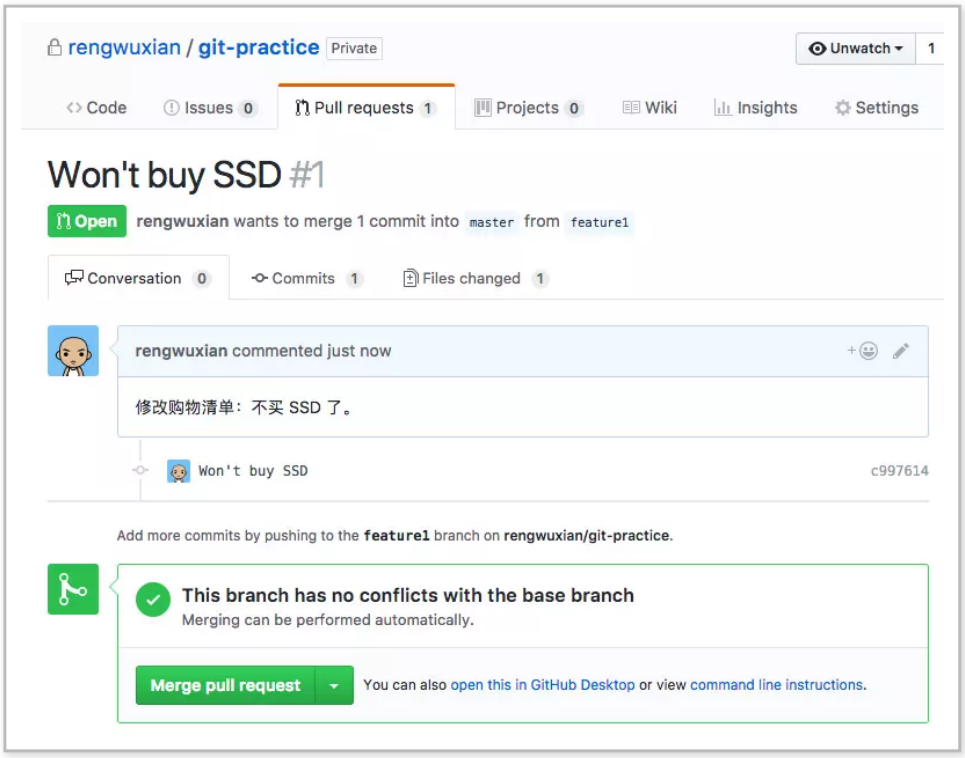
同样是把写好的 `branch` 给同事看，使用 Pull Request 的话你可以这样做：

- 把 `branch` `push` 到中央仓库；
- 在中央仓库处创建一个 Pull Request。以 GitHub 为例：



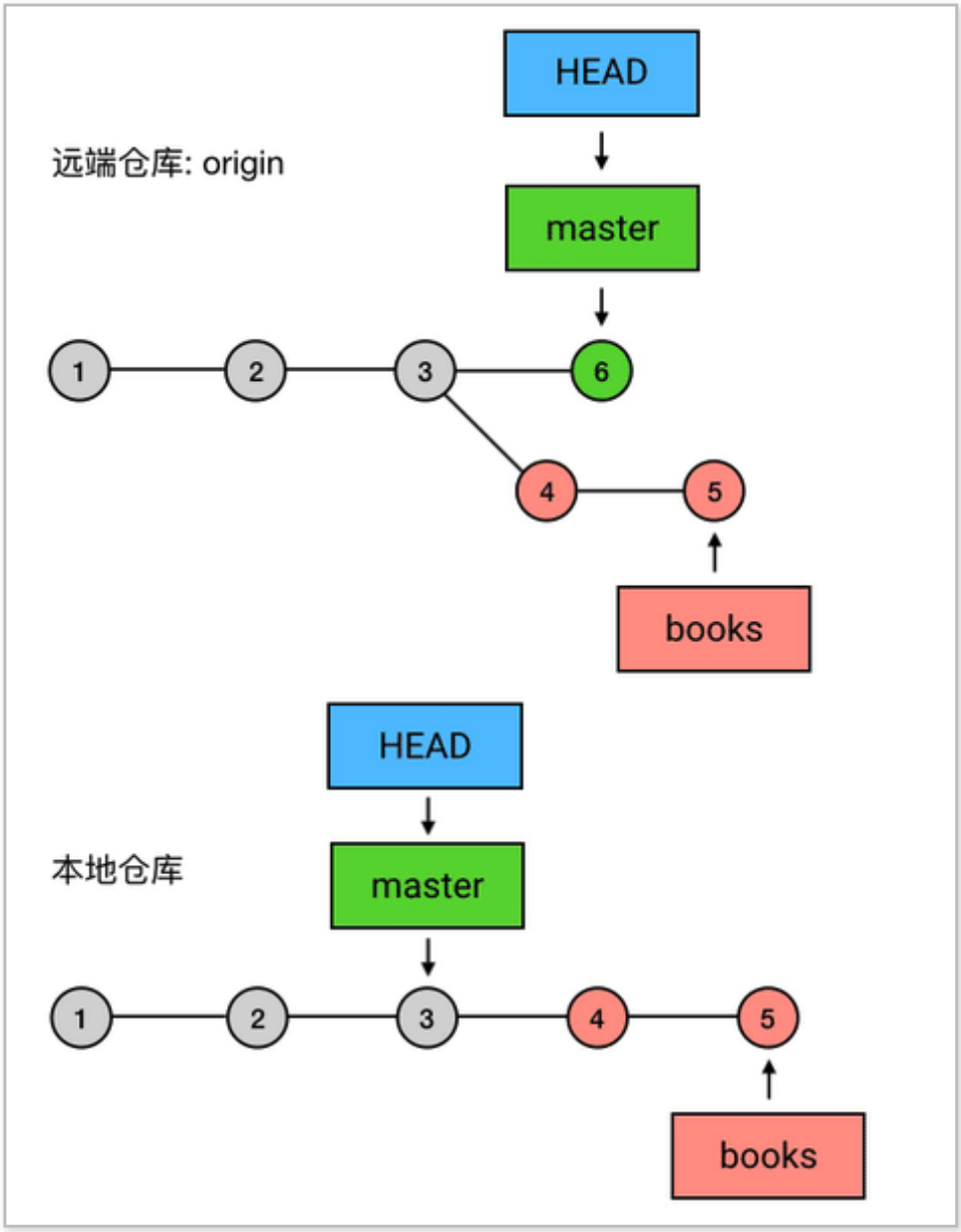






然后你的同事就可以在 GitHub 上看到你创建的 Pull Request 了。他们可以在 GitHub 的这个页面查看你的 `commit s`，也可以给你评论表示赞同或提意见，你接下来也可以根据他们的意见把新的 `commit s` `push` 上来，这也页面会随着你新的 `push` 而展示出最新的 `commits` 。

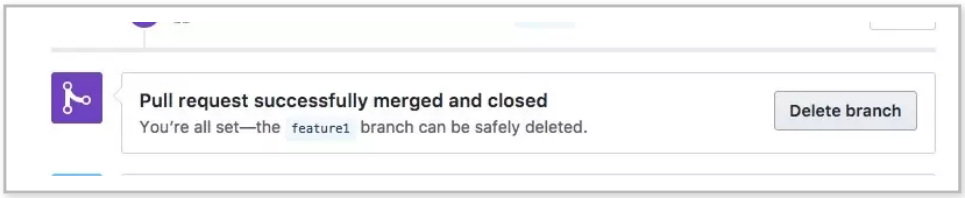
在讨论结束以后，你们一致认为这个 `branch` 可以合并了，你只需要点一下页面中那个绿色的 "Merge pull request" 按钮，GitHub 就会自动地在中央仓库帮你把 `branch` 合并到 `master` 了：




"Merge pull request" 按钮自动在中央仓库 merge

然后你只要在本地 `pull` 一下，把最新的内容拉到你的电脑上，这件事情就算完成了。

另外，GitHub 还设计了一个贴心的 "Delete branch" 按钮，方便你在合并之后一键删除 `branch` 。




## 2. 一人多任务

除了代码分享的便捷，基于 Feature Branch 的工作流对于一人多任务的工作需求也提供了很好的支持。

安安心心做事不被打扰，做完一件再做下一件自然是很美好的事，但现实往往不能这样。对于程序员来说，一种很常见的情况是，你正在认真写着代码，忽然同事过来跟你说：「内个..... 你这个功能先放一放吧，我们最新讨论出要做另一个更重要的功能，你来做一下吧。」

其实，虽然这种情况确实有点烦，但如果你是在独立的 `branch` 上做事，切换任务是很简单的。你只要稍微把目前未提交的代码简单收尾一下，然后做一个带有「未完成」标记的提交（例如，在提交信息里标上「TODO」），然后回到 `master` 去

创建一个新的 `branch` 就好了。

```
git checkout master
git checkout -b new_feature
```

> 上面这两行代码有更简单的操作方式，不过为了小册内容的简洁性，我就不引入更多的内容了，有兴趣的话可以自己搜索一下。

如果有一天需要回来继续做这个 `branch` ，你只要用 `checkout` 切回来，就可以继续了。

## 小结

这一节介绍了 Feature Branching 这种工作流。它的概念很简单：

- 每个新功能都新建一个 `branch` 来写；
- 写完以后，把代码分享给同事看；写的过程中，也可以分享给同事讨论。另外，借助 GitHub 等服务提供方的 Pull Request 功能，可以让代码分享变得更加方便；
- 分支确定可以合并后，把分支合并到 `master`

，并删除分支。

这种工作流由于功能强大，而且概念和使用方式都很简单，所以很受欢迎。再加上 GitHub 等平台提供了 Pull Request 的支持，目前这种工作流是商业项目开发中最为流行的工作流。

## 留言

评论将在后台进行审核，审核通过后对所有人可见

- **mazhenzhong86**

  git push origin --delete branchname 我用 - d 是不能删除远端分支的，用 --delete 可以

  ▲0 评论 2 天前

- **M78Code**

  git checkout master， git checkout -b new_feature 上面这两行代码有更简单的操作方

式，这两行更简单的操作方式是什么呀

▲0　　收起评论　2 月前

- IAM 四十二
  Android 开发工程师

  git checkout -b new_feature origin/master  基于远程 master 分支切出一个新分支。

  2 月前

评论审核通过后显示　　评论

- **订阅公众号**

  大佬，有一个问题，为什么，别人提交分支了，然后我本地进行更新，然后我使用使用 git branch 看不到别人的分支，虽然我知道可以直接 checkout 切换到别人的分支，但是想知道为什么看不到

▲0　　收起评论　2 月前

- MDove

Android 开发工程师 @ 人人车

git branch -a 看到所有分支

2 月前

评论审核通过后显示

评论

- **肥猫不吃鱼**

一直就是这么做的，就是不知道是这个概念。

▲0 评论 2 月前

- **碧海银剑**

feature branching 学习了，点个赞！

▲0 评论 3 月前

- **WhartonJason** andriod @ 分米金科

11 点钟打卡, 再次感谢扔物线表哥, Android 路上有你真是太好了

0 评论 3 月前

- **大白酱**

pull request 的讲解中有个错别字 “这也页面会随着你新的 push 而展示出最新的 commits”，第二个“也” 字应该是“样”？

0 评论 3 月前

- **jackywu**

显然这样效率很低，还要删除分支，review 有专门的工具 gerrit

0 收起评论 3 月前

  - Michael 翔
    CID @ HW
    删除分支是为了让分支不野蛮增长，gerrit 用了，也是同样要去删除没用的分支
    1 月前

评论审核通过后显示

评论

- **DR.HYDRA**

  有个问题，如果所有的开发都放到一个新的 branch 做，那当你在新的 feature 还没有开发完，没办法 merge 回去，但是却在里面写了一个通用的小模块，想要给同事用应该怎么办

  ▲1 收起评论 6 月前

  - 这是一个昵称 Z Android 开发工程师 @ 北京千橡网景科技发展有限公司

    http://blog.csdn.net/ybdesire/article/details/42145597 可以参考一下这个，做法就是如果

    新的 feature 是以小模块为单位，写好一个模块就 commit 一

次，那么就可以实现只合并该分支的某一部分 commits

6 月前

- Hemione
  Android

  你可以将这个通用小模块提交到当前分支，这样在其他分支可以 cherry-pick 本次 commit

  3 月前

- Michael 翔
  CID @ HW

  赞，总算知道 cherry-pick 的用处了

  1 月前

评论审核通过后显示

评论

**ERROR** Android

“如果同事有意见 " 那里最后一句有点疑问，tab 没改空格之前不就已经提交到 master 了吗，怎么是在有问题之前都不会正式提交到 master 呢

▲0 收起评论 7 月前

- ivotai

  你是 push 分支让同事 review 的，他提出意见，你修改分支，他说 OK，你再合并到 master。

  6 月前

- Landroid

  Android 应用开发工程师

  凯哥说的是两种情况，不要把第一种情况的结果当成第二种情况的开始。

  6 月前

评论

- **爱撸铁的攻城狮** Android @ 北京

  神特么好理解，一看就懂，大赞。。。

  ▲0 评论 7 月前

- **ClimberJing** 后台开发 @ 携程

  本地分支删除后想删除远程分支命令平时用的是 git push origin :branch，请问和 git push origin -d books # 有何区别？# 是代表什么呢？必须有吗？谢谢

  ▲0 收起评论 8 月前

  - 菜菜鸟 android 菜鸟

    #代表注释，（写在文章里给看的人注释，用的时候就写 git push origin -d books）

    5 月前

  - BeeShark iOS

    两个命令都是删除远

程分支

3 月前

评论审核通过后显示

评论

- **鹏鹏鹏先森__** 移动开发

  凯哥，没用过 git 表示这节看不懂啊，不知道这节内容实际项目怎么使用的

  ▲0 评论 8 月前

- **Rkhcy**

  图文并茂

  ▲0 评论 9 月前

- **Amor** Android 开发工程师

  git push origin -d new-branch 这个命令会把本地的代码 push 到 origin 的 new-branch 分支上，同时会删除掉 origin 上的 new-branch，那我在 origin 上找不到 new-branch 的 ref，怎么

合并去呢.... 这个命令应用场景是什么

▲0 收起评论 9 月前

- IAM 四十二
  Android 开发工程师

  这个场景是，你的 new-branch 已经被正常合并到 master 分支了，而且这个分支确定不再需要了，然后进行删除；并不是 push 完成之后立即执行这个命令

  9 月前

评论审核通过后显示 评论

- **hailongzhao**

为什么我用 - d 删除远程分支不行 error: unknown switch `d' 使用 --delete 可以。git 版本问题?

▲0 收起评论 9 月前

- Songlcy
  Android React Native @ weego
  可以试试 git push origin :branch
  9 月前

评论审核通过后显示 评论

- **小葱** 前端
  现在公司用的就是这样的工作模式, 虽然我们稍微做了一下修改, 中间加了不一样的发布流程, 看完之后可以理一理思路

▲0 评论 9 月前

- **纸飞猫 hjt**
  感谢

▲0 评论 9 月前

- **RTFSC**

  666

  ▲0 评论 9 月前

- **祺在掘金**

  感谢

  ▲0 评论 9 月前

全文完