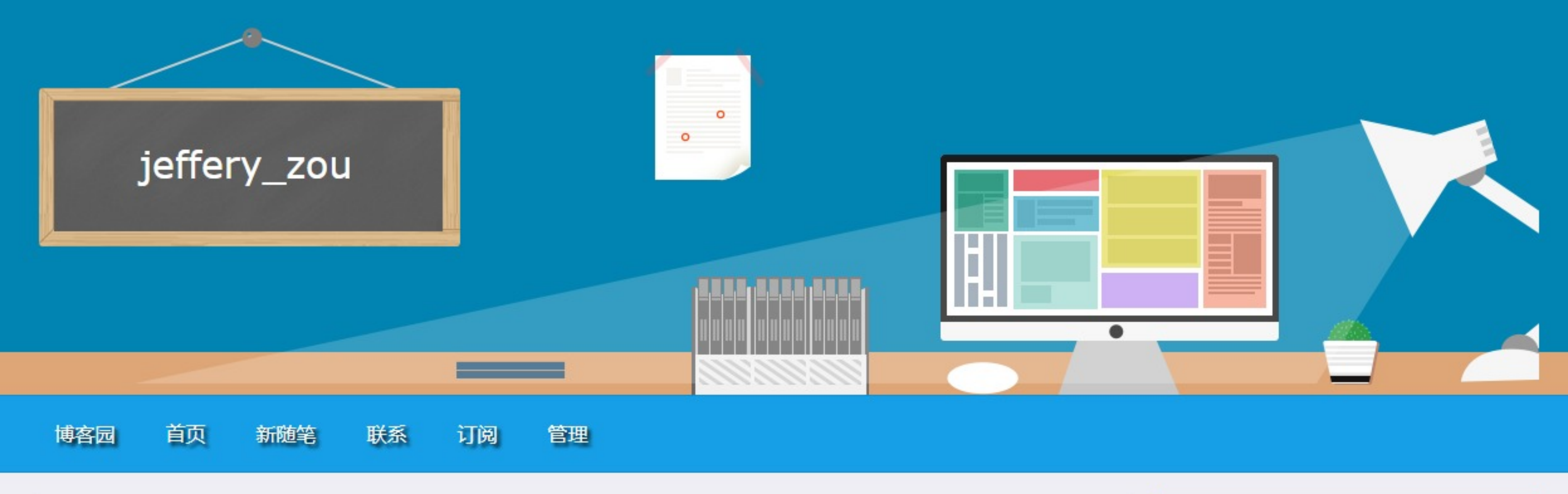
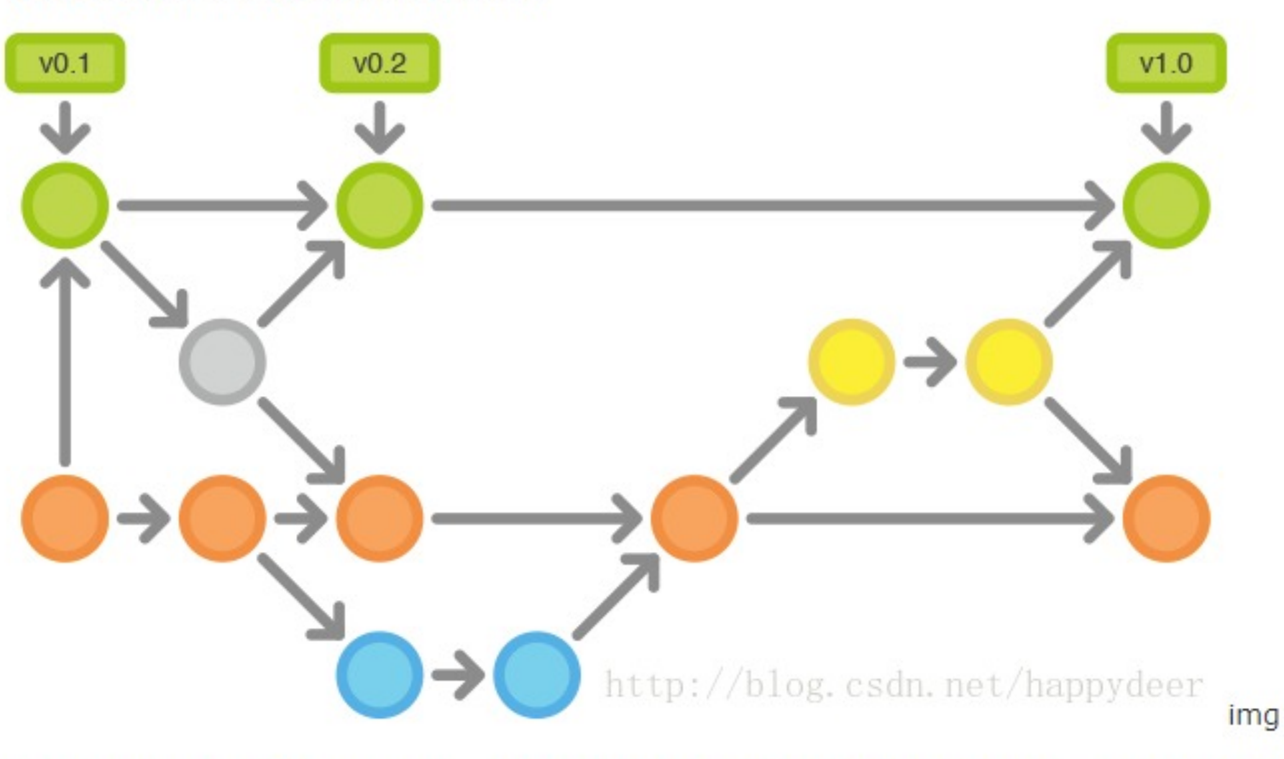


Gitflow工作流程

在工作场合实施Git的时候，有很多种工作流程可供选择，此时反而会让你手足无措。本文罗列了企业团队最常用的一些git工作流程，包括Centralized Workflow、Feature Branch Workflow、Gitflow Workflow、Forking Workflow。愿以此文抛砖引玉。

在你开始阅读之前，请记住：这些流程应被视作为指导方针，而非"铁律"。我们只是想告诉你可能的做法。因此，如果有必要的话，你可以组合使用不同的流程。

（本文主要介绍Gitflow Workflow......）



img

Vincent Driessen曾经写过一篇博文，题为"A successful Git branching model"（一个成功的Git分支模型）。Gitflow工作流程就是从这篇文章里来的。

Gitflow工作流程围绕项目发布定义了严格的分支模型。尽管它比Feature Branch Workflow更复杂一些，但它也为管理更大规模的项目提供了坚实的框架。

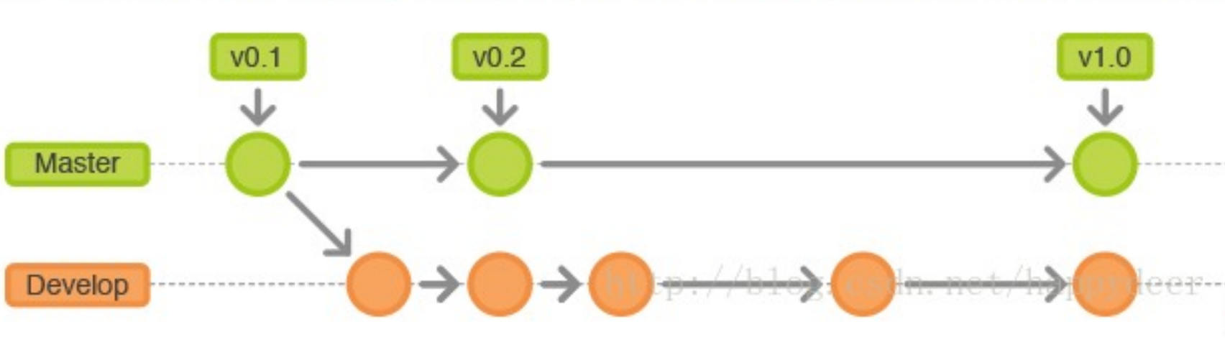
与Feature Branch Workflow比起来，Gitflow流程并没有增加任何新的概念或命令。其特色在于，它为不同的分支分配了非常明确的角色，并且定义了使用场景和用法。除了用于功能开发的分支，它还使用独立的分支进行发布前的准备、记录以及后期维护。当然，你还是能充分利用Feature Branch Workflow的好处：拉拽请求（Pull Request）、隔离的试验以及更高效率的合作。

它是怎么工作的?

Gitflow流程仍然使用一个中央代码仓库，它是所有开发者的信息交流中心。跟其他的工作流程一样，开发者在本地完成开发，然后再将分支代码推送到中央仓库。唯一不同的是项目中分支的结构。

用于记录历史的分支

Gitflow使用两个分支来记录项目开发的历史，而不是使用单一的master分支。在Gitflow流程中，master只是用于保存官方的发布历史，而develop分支才是用于集成各种功能开发的分支。使用版本号为master上的所有提交打标签（tag）也很方便。

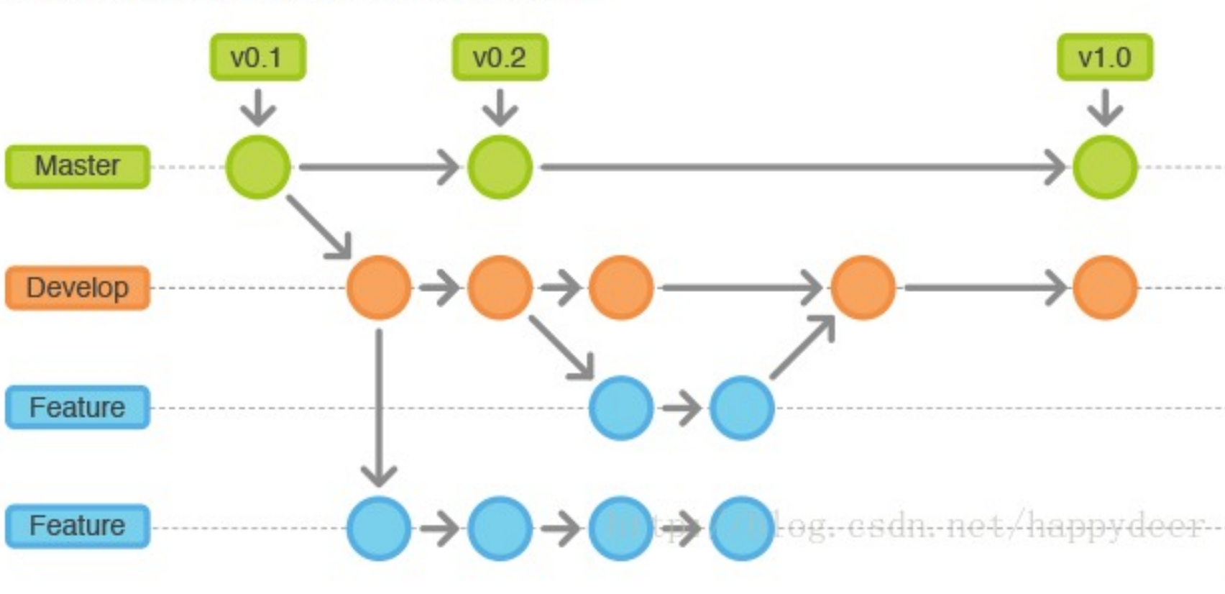


img

事实上，Gitflow流程就是围绕这两个特点鲜明的分支展开的。

用于功能开发的分支

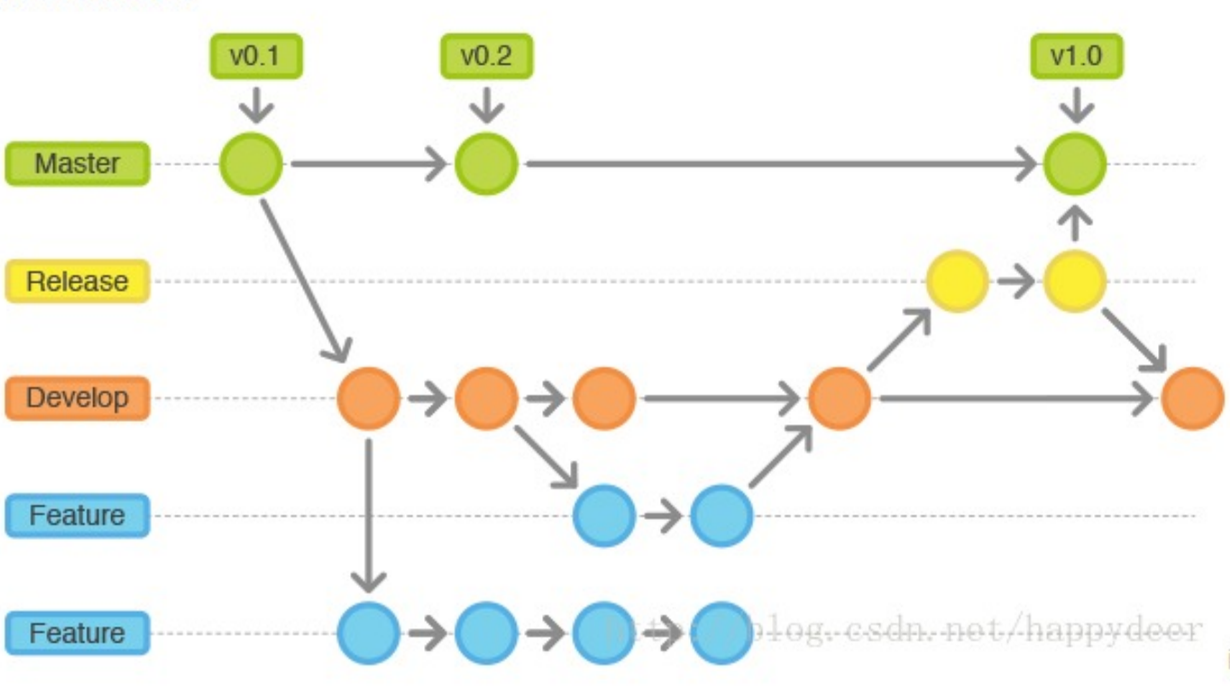
每一个新功能的开发都应该各自使用独立的分支。为了备份或便于团队之间的合作，这种分支也可以被推送到中央仓库。但是，在创建新的功能开发分支时，父分支应该选择develop（而不是master）。当功能开发完成时，改动的代码应该被合并（merge）到develop分支。功能开发永远不应该直接牵扯到master。



img

注意：组合使用功能开发分支和develop分支的这种设计，其实完全就是Feature Branch Workflow的理念。然而，Gitflow流程并不止于此。且看下文分解。

用于发布的分支

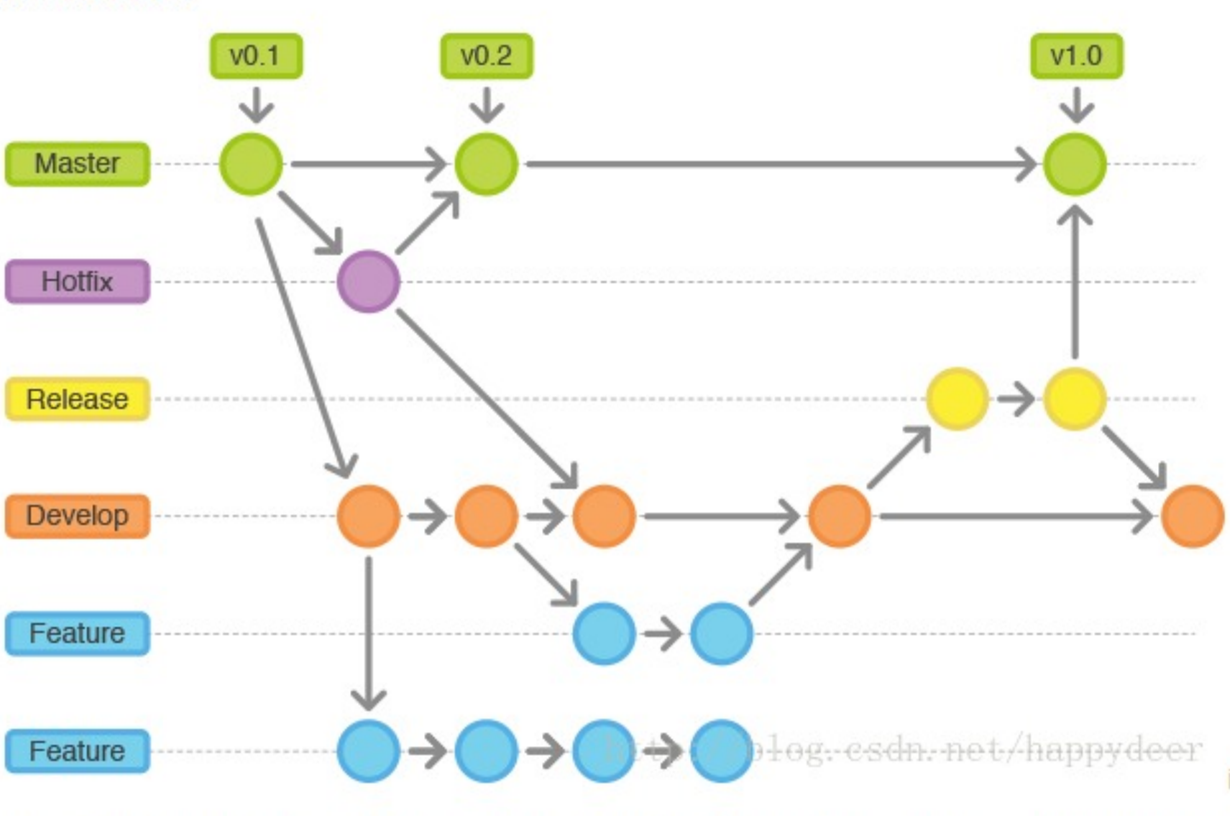


img

一旦develop分支积聚了足够多的新功能（或者预定的发布日期临近了），你可以基于develop分支建立一个用于产品发布的分支。这个分支的创建意味着一个发布周期的开始，也意味着本次发布不会再增加新的功能——在这个分支上只能修复bug，做一些文档工作或者跟发布相关的任务。在一切准备就绪的时候，这个分支会被合并入master，并且用版本号打上标签。另外，发布分支上的改动还应该合并入develop分支——在发布周期内，develop分支仍然在被使用（一些开发者会把其他功能集成到develop分支）。

使用专门的一个分支来为发布做准备的好处是，在一个团队忙于当前的发布的同时，另一个团队可以继续为接下来的一次发布开发新功能。这也有助于清晰表明开发的状态，比如说，团队在汇报状态时可以轻松使用这样的措辞，"这星期我们要为发布4.0版本做准备。"从代码仓库的结构上也能直接反映出来。常用的一些措辞还有：基于develop新建分支，合并入master；命名规则为：release-*或 release/*

用于维护的分支



img

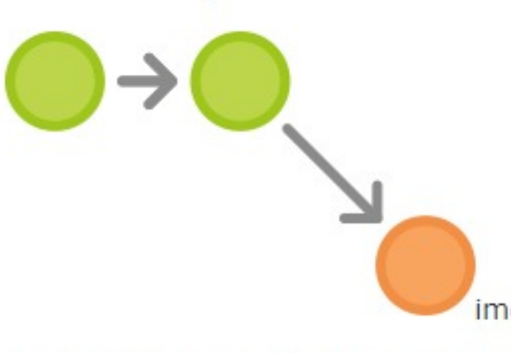
发布后的维护工作或者紧急问题的快速修复也需要使用一个独立的分支。这是唯一一种可以直接基于master创建的分支。一旦问题被修复了，所做的改动应该被合并入master和develop分支（或者用于当前发布的分支）。在这之后，master上还要使用更新的版本号打好标签。

这种为解决紧急问题专设的绿色通道，让团队不必打乱当前的工作流程，也不必等待下一次的产品发布周期。你可以把用于维护的分支看成是依附于master的一种特别的发布分支。

***举例说明 ***

下面的例子将演示Gitflow流程如何被用来管理一次产品发布。假设你已经创建好了一个中央仓库。

1. 创建develop分支



img

第一步是给默认的master配备一个develop分支。一种简单的做法是：让一个开发者在本地建立一个空的develop分支，然后把它推送到服务器。

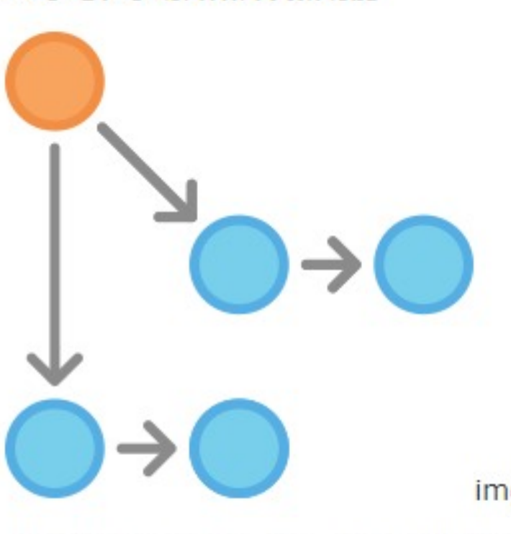
```
git branch develop
git push -u origin develop
```

develop分支将包含项目的所有历史，而master会是一个缩减版本。现在，其他开发者应该克隆（clone）中央仓库，并且为develop创建一个追踪分支。

```
git clone ssh://user@host/path/to/repo.git
git checkout -b develop origin/develop
```

到现在，所有人都把包含有完整历史的分支（develop）在本地配置好了。

2. 小马和小明开始开发新功能



img

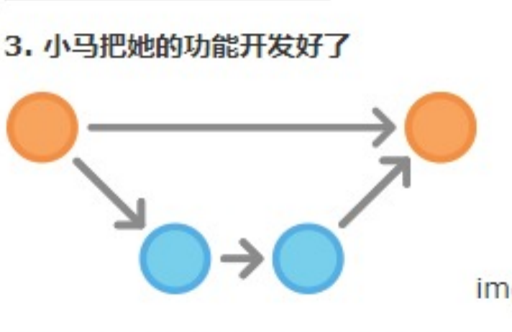
我们的故事从小马和小明要分别开发新功能开始。他们俩各自建立了自己的分支。**注意**，他们在创建分支时，父分支不能选择master，而要选择develop。

```
git checkout -b some-feature develop
```

他们俩都在自己的功能开发分支上开展工作。通常就是这种Git三部曲：edit，stage，commit：

```
git status
git add <some-file>
git commit
```

3. 小马把她的功能开发好了



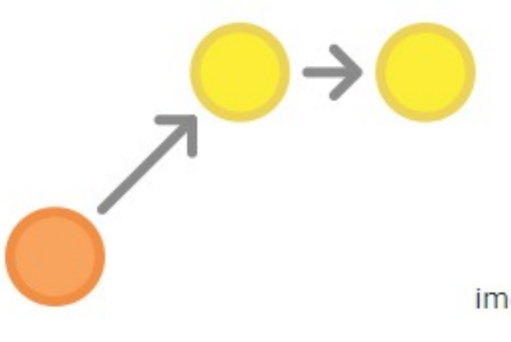
img

在提交过几次代码之后，小马觉得她的功能做完了。如果她所在的团队使用"拉拽请求"，此刻便是一个合适的时机——她可以提出一个将她所完成的功能合并入develop分支的请求。要不然，她可以自行将她的代码合并入本地的develop分支，然后再推送到中央仓库，像这样：

```
git pull origin develop
git checkout develop
git merge some-feature
git push
git branch -d some-feature
```

第一条命令确保了本地的develop分支拥有最新的代码——这一步必须在将功能代码合并之前做！**注意**，新开发的功能代码永远不能直接合并入master。必要时，还需要解决在代码合并过程中的冲突。

4. 小马开始准备一次发布



img

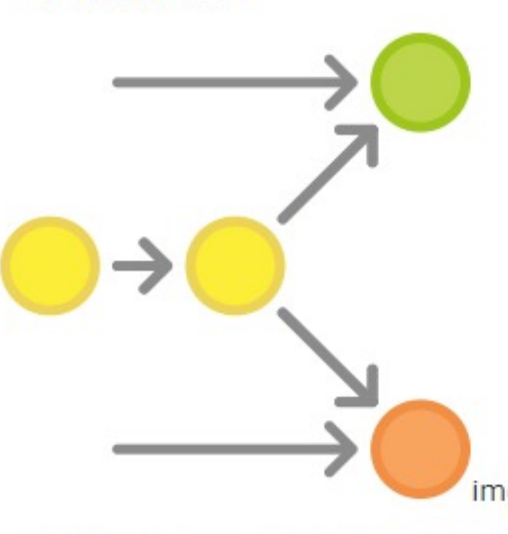
尽管小明还在忙着开发他的功能，小马却可以开始准备这个项目的第一次正式发布了。类似于功能开发，她使用了一个新的分支来做产品发布的准备工作。在这一步，发布的版本号也最初确定下来。

```
git checkout -b release-0.1 develop
```

这个分支专门用于发布前的准备，包括一些清理工作、全面的测试、文档的更新以及任何其他的准备工作。它与用于功能开发的分支相似，不同之处在于它是专为产品发布服务的。

一旦小马创建了这个分支并把它推向中央仓库，这次产品发布包含的功能也就固定下来了。任何还处于开发状态的功能只能等待下一个发布周期。

5. 小马完成了发布



img

一切准备就绪之后，小马就要把发布分支合并入master和develop分支，然后再将发布分支删除。**注意**，往develop分支的合并是很重要的，因为开发人员可能在发布分支上修复了一些关键的问题，而这些修复对于正在开发中的新功能是有益的。再次提醒一下，如果小马所在的团队强调代码评审（Code Review），此时非常适合提出这样的请求。

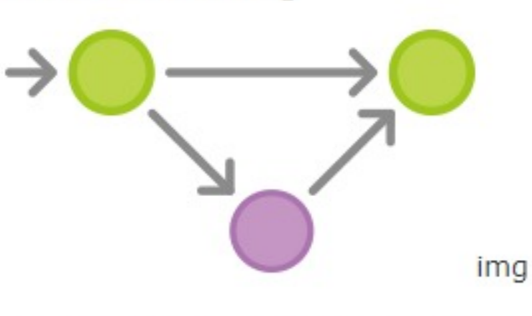
```
git checkout master
git merge release-0.1
git push
git checkout develop
git merge release-0.1
git push
git branch -d release-0.1
```

发布分支扮演的角色是功能开发（develop）与官方发布（master）之间的一个缓冲。无论什么时候你把一些东西合并入master，你都应该随即打上合适的标签。

```
git tag -a 0.1 -m"Initial public release" master
git push --tags
```

Git支持钩子（hook）的功能，也就是说，在代码仓库里某些特定的事件发生的时候，可以执行一些预定义的脚本。因此，一种可行的做法是：在服务器端配置一个钩子，当你把master推送到中央仓库或者推送标签时，Git服务器能为产品发布进行一次自动的构建。

6. 用户发现了一个bug



img

当一次发布完成之后，小马便回去与小明一起开发其他功能了。突然，某个用户提出抱怨说当前发布的产品里有一个bug。为了解决这个问题，小马（或者小明）基于master创建了一个用于维护的分支。她在这个分支上修复了那个bug，然后把改动的代码直接合并入master。

```
git checkout -b issue-#001 master
\# Fix the bug
git checkout master
git merge issue-#001
git push
```

跟用于发布的分支一样，在维护分支上的改动也需要合并入develop分支，这一点是很重要的！因此，小马务必不能忘了这一步。随后，她就可以将维护分支删除。

```
git checkout develop
git merge issue-#001
git push
git branch -d issue-#001
```

原文链接：https://www.atlassian.com/git/workflows#!workflow-gitflow

本文作者：happydeer
原文链接：http://blog.csdn.net/happydeer/article/details/17618935
版权归作者所有，转载请注明出处

标签： git

好文要顶 关注我 收藏该文

Jeffery-Zou
关注 - 15
粉丝 - 3
+加关注

1 推荐　0 反对

« 上一篇： ACC、LDW、BSD究竟是怎么实现的?
» 下一篇： UML图

posted @ 2019-01-17 00:05 Jeffery-Zou 阅读(28656) 评论(5) 编辑 收藏

评论列表

#1楼 [楼主] 2019-01-17 00:06 Jeffery-Zou

https://nvie.com/posts/a-successful-git-branching-model/

A successful Git branching model

支持(1) 反对(0)

#2楼 [楼主] 2019-01-17 00:07 Jeffery-Zou

https://www.git-tower.com/learn/git/ebook/cn/command-line/advanced-topics/git-flow

git-flow 的工作流程

支持(0) 反对(0)

#3楼 [楼主] 2019-01-17 00:09 Jeffery-Zou

http://www.ruanyifeng.com/blog/2015/12/git-workflow.html
阮一峰

支持(0) 反对(0)

#4楼 [楼主] 2019-01-17 00:10 Jeffery-Zou

https://danielkummer.github.io/git-flow-cheatsheet/index.zh_CN.html
git-flow 备忘清单

支持(0) 反对(0)

#5楼 2020-03-25 00:53 Alice_Mye

赞

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册， 访问 网站首页。

公告

昵称： Jeffery-Zou
园龄： 8年8个月
粉丝： 3
关注： 15
+加关注

< 2020年8月 >

日	一	二	三	四	五	六
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

搜索

找找看
谷歌搜索

我的标签

- git(3)
- C/C++(3)
- android(3)
- 开源软件架构(3)
- Automotive Safety(2)
- ADAS(2)
- AUTOSAR(1)
- 汽车电子(1)
- CV(1)
- EEPROM(1)
- 更多

随笔档案 (34)

- 2019年4月(2)
- 2019年2月(1)
- 2019年1月(15)
- 2018年12月(2)
- 2014年4月(1)
- 2013年6月(1)
- 2013年4月(2)
- 2012年4月(2)
- 2012年3月(4)
- 2011年12月(3)
- 2011年11月(1)

最新评论

1. Re:Gitflow工作流程
赞
--Alice_Mye

2. Re:Gitflow工作流程
git-flow 备忘清单
--Jeffery-Zou

3. Re:Gitflow工作流程
阮一峰
--Jeffery-Zou

4. Re:Gitflow工作流程
git-flow 的工作流程
--Jeffery-Zou

5. Re:Gitflow工作流程
A successful Git branching model
--Jeffery-Zou

阅读排行榜

1. Gitflow工作流程(28655)
2. FFMpeg测试例子x学习ffplayer.c(2811)
3. OpenCV的Mat IplImage CvMat(2657)
4. C语言异或指针双向链表(2260)
5. 谷歌访问助手(2213)

评论排行榜

1. FFMpeg测试例子x学习ffplayer.c(10)
2. Gitflow工作流程(5)
3. 谷歌访问助手(1)

推荐排行榜

1. 开通msdn博客园(1)
2. FFMpeg测试例子x学习ffplayer.c(1)
3. Gitflow工作流程(1)