

Linux大棚

体会Linux，体会一种生活方式。

欢迎你的到来：

我们希望这里留下你们的脚印：最诚心学习Linux的小鸟们，最热爱Linux的朋友们，最精通Linux的高手们。 我们的希望从2008年9月9日开始。

roccrocket

- 092010-12

GIT分支管理是一门艺术

看日记学git



原创文章属于《Linux大棚》博客，博客地址为<http://roclinux.cn>。

文章作者为 roccrocket。

为了防止某些网站的恶性转载，特在每篇文章前加入此信息，还望读者体谅。

===

[正文开始]

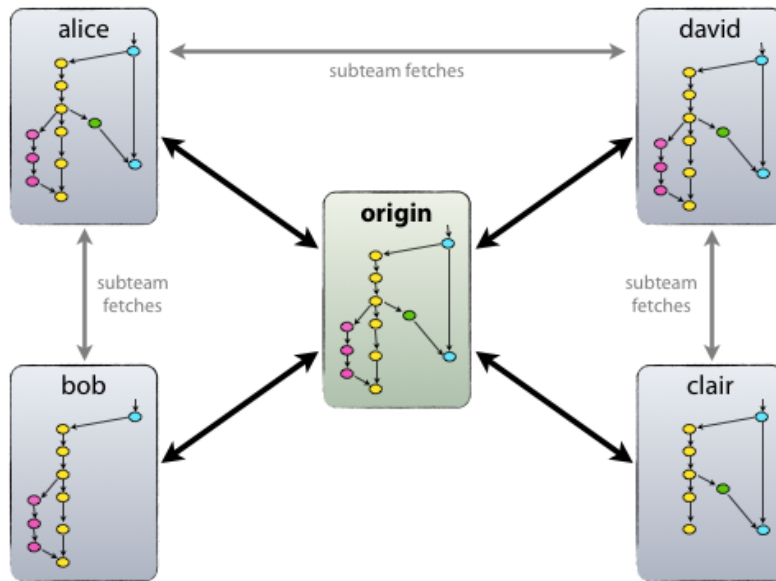
原文链接： <http://www.nvie.com/posts/a-successful-git-branching-model/>

原文作者： Vincent Driessen

本文经Linux大棚博主总结精简而成。

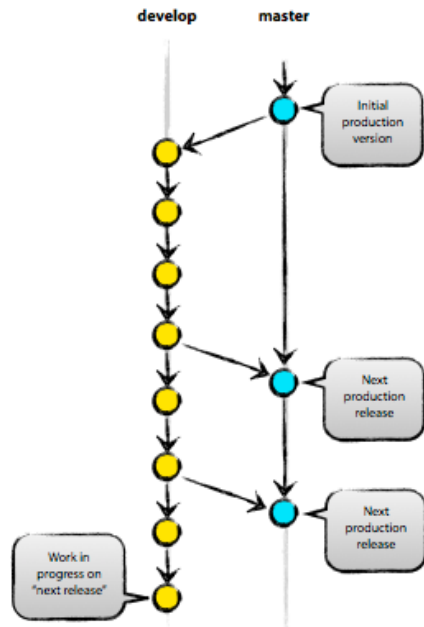
1

GIT，在技术层面上，绝对是一个无中心的分布式版本控制系统，但在管理层面上，我建议你保持一个中心版本库。



2

我建议，一个中心版本库(我们叫它origin)至少包括两个分支，即“主分支(master)”和“开发分支(develop)”



3

要确保：团队成员从主分支(master)获得的都是处于可发布状态的代码，而从开发分支(develop)应该总能够获得最新开发进展的代码。

4

在一个团队开发协作中，我建议，要有“辅助分支”的概念。

5

“辅助分支”，大体包括如下几类：“管理功能开发”的分支、“帮助构建可发布代码”的分支、“可以便捷的修复发布版本关键BUG”的分支，等等

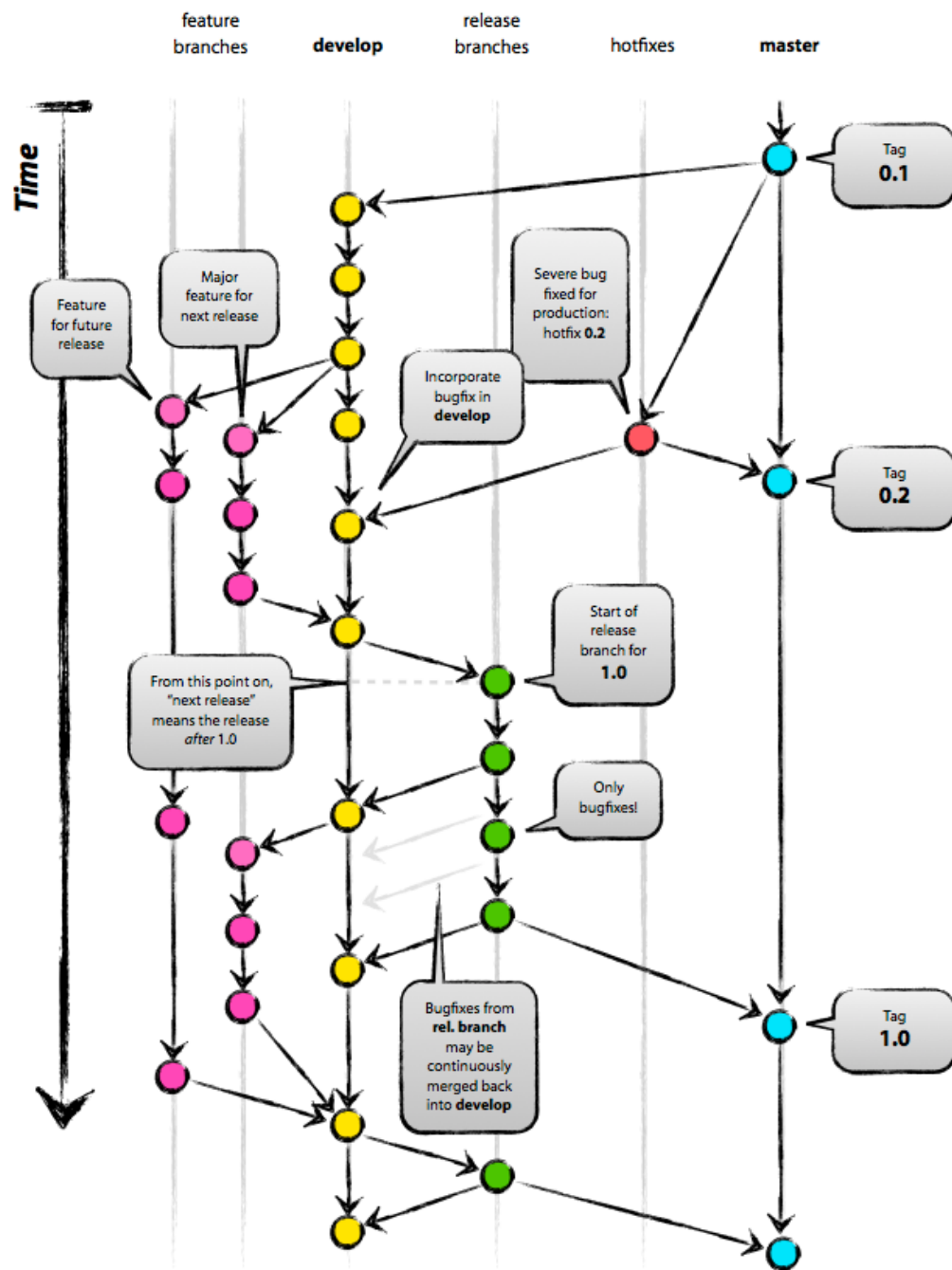
6

“辅助分支”的最大特点就是“生命周期十分有限”，完成使命后即可被清除。

7

我建议至少还应设置三类“辅助分支”，我们称之为“Feature branches”，“Release branches”，“Hotfix branches”。

至此，我们形成了如下这张最重要的组织组，包含了两个粗体字分支（**master/develop**）和三个细体字分支（**feature/release/hotfixes**）。



8

“Feature branches”，起源于develop分支，最终也会归于develop分支。

9

“Feature branches”常用于开发一个独立的新功能，且其最终的结局必然只有两个，其一是合并入“develop”分支，其二是被抛弃。最典型的“Feature branches”一定存在于团队开发者那里，而不应该是“中心版本库”中。

10

“Feature branches”起源于“develop”分支，实现方法是：

```
git checkout -b myfeature develop
```

11

“Feature branches”最终也归于“develop”分支，实现方式是：

```
git checkout develop
```

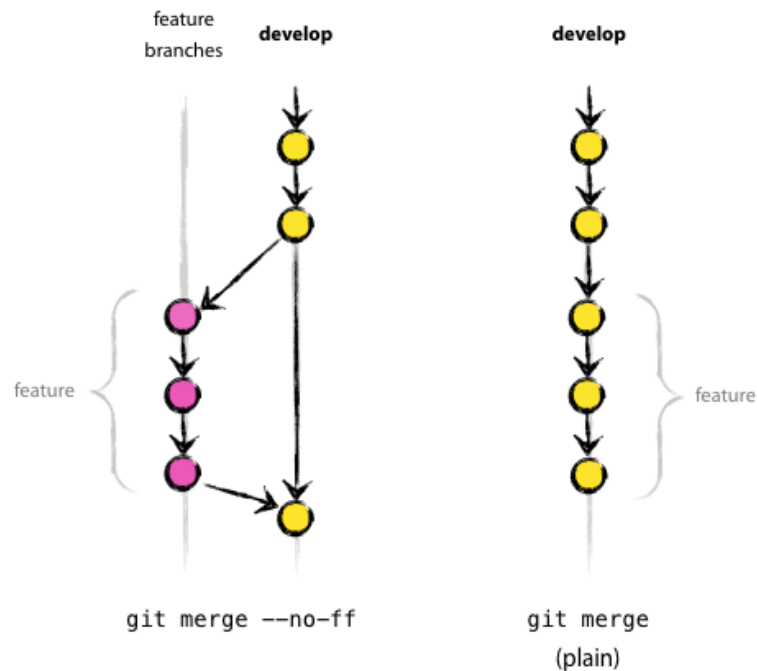
```
git merge --no-ff myfeature
```

(--no-ff, 即not fast forward, 其作用是：要求git merge即使在fast forward条件下也要产生一个新的merge commit)

(此处，要求采用--no-ff的方式进行分支合并，其目的在于，希望保持原有“Feature branches”整个提交链的完整性)

```
git branch -d myfeature
```

```
git push origin develop
```



12

“Release branch”，起源于develop分支，最终归于“develop”或“master”分支。这类分支建议命名为“release-*”

13

“Release branch”通常负责“短期的发布前准备工作”、“小bug的修复工作”、“版本号等元信息的准备工作”。与此同时，“develop”分支又可以承接下一个新功能的开发工作了。

14

“Release branch”产生新提交的最好时机是“develop”分支已经基本到达预期的状态，至少希望新功能已经完全从“Feature branches”合并到“develop”分支了。

15

创建“Release branches”，方法是：

```
git checkout -b release-1.2 develop
```

`./bump-version.sh 1.2`（这个脚本用于将代码所有涉及版本信息的地方都统一修改到1.2，另外，需要用户根据自己的项目去编写适合的**bump-version.sh**）

```
git commit -a -m "Bumped version number to 1.2"
```

16

在一段短时间内，在“Release branches”上，我们可以继续修复bug。在此阶段，严禁新功能的并入，新功能应该被合并到“develop”分支的。

17

经过若干bug修复后，“Release branches”上的代码已经达到可发布状态，此时，需要完成三个动作：第一是将“Release branches”合并到“master”分支，第二是一定要为master上的这个新提交打TAG（记录里程碑），第三是要将“Release branches”合并回“develop”分支。

```
git checkout master
```

```
git merge --no-ff release-1.2
```

```
git tag -a 1.2 （使用-u/-s/-a参数会创建tag对象，而非软tag）
```

```
git checkout develop
```

```
git merge --no-ff release-1.2
```

```
git branch -d release-1.2
```

18

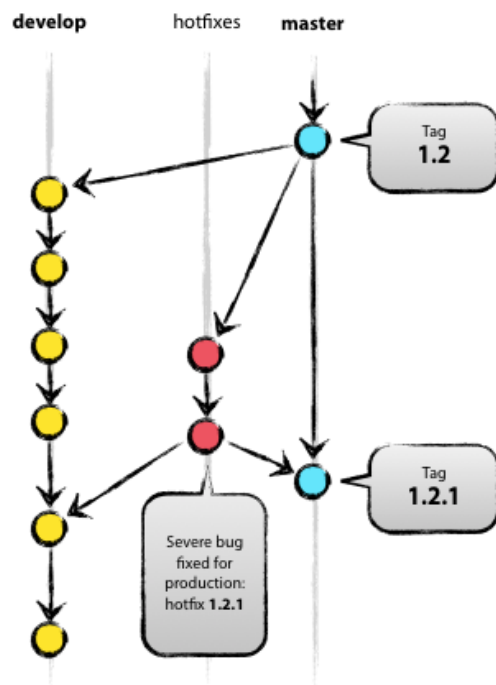
“Hotfix branches”源于“master”，归于“develop”或“master”，通常命名为“hotfix-*”

19

“Hotfix branches”类似于“Release branch”，但产生此分支总是非预期的关键BUG。

20

建议设立“Hotfix branches”的原因是：希望避免“develop分支”新功能的开发必须为BUG修复让路的情况。



21

建立“Hotfix branches”，方法是：

```
git checkout -b hotfix-1.2.1 master
```

```
./bump-version.sh 1.2.1
```

```
git commit -a -m "Bumpt version to 1.2.1" （然后可以开始问题修复工作）
```

```
git commit -m "Fixed severe production problem" （在问题修复后，进行第二次提交）
```

22

BUG修复后，需要将“Hotfix branches”合并回“master”分支，同时也需要合并回“develop”分支，方法是：

```
git checkout master
```

```
git merge --no-ff hotfix-1.2.1
```

```
git tag -a 1.2.1
```

```
git checkout develop
```

```
git merge --no-ff hotfix-1.2.1
```

```
git branch -d hotfix-1.2.1
```

23



还记得文章开始时的那张大图么，我建议你把这幅大图从这里下载下来，打印出来，贴在你写字台的墙壁上，好处不言而喻。

over~

Posted by rocrocket @ 12:18 am

Tags: git, 分支, 模型, 管理

27 Comments to *GIT分支管理是一门艺术*

-  *scorpioo* says:
December 9, 2010 at 10:19 am
真的是非常不错，非常实用的模式 ;D
-  *Kevin* says:
December 9, 2010 at 1:08 pm
roclinux的公司现在用的什么版本管理软件呢，是git吗？

回复

回复



rocrocket Reply:

December 9th, 2010 at 1:39 pm

公司在用svn，我自己使用git。

回复



Kevin Reply:

December 9th, 2010 at 9:34 pm

@rocrocket,

也是，GIT没有权限控制，公司会比较少用。

公司现在使用的版本控制方式很不成熟，也引发了我对版本控制的一些思考，所以比较关注这方面的知识。

rocrocket有空可以分享些使用SVN的体会。


回复




山里来的鱼 Reply:


July 8th, 2014 at 8:49 am


@Kevin, gitolite等服务器软件是有权限管理的:)


-  *Wolfkin* says:
December 14, 2010 at 5:28 pm
好东西！正需要呢。
在你的日记里学到很多东西。
谢谢！
回复


 *rocrocket* Reply:
December 15th, 2010 at 6:03 pm
谢谢支持:)
回复


-  *Meteor Liu* says:
December 15, 2010 at 9:15 am
good, thanks
回复

-  *Iceway* says:
March 5, 2011 at 11:18 pm
文章非常好, 现在工作上就在用git, 不过感觉公司对branch的管理规范很差, 所以只好自己维持多个branch, 确定更新后在按照公司的要求修改,push.
回复

-  *名字大全* says:
April 12, 2011 at 2:34 pm
传说中的牛人
回复

-  *ddatsh* says:
May 31, 2011 at 5:02 pm
博主有空研究下 TOPGIT和GIT FLOW 怎么有机的能结合使用吧
回复

-  *neo* says:
June 2, 2011 at 12:56 pm
把git系列都看完了, 博主写的很不错, 清晰明了, 感谢
回复

-  *James* says:
October 20, 2011 at 9:36 am
请问下博主，用的是什么画图工具阿？上面的图很不错。


回复




rocrocket Reply:
October 28th, 2011 at 12:59 pm

图是转载的:)


回复

-  *wisfern* says:
November 16, 2011 at 11:27 pm
最近才在动手学习git，看这一篇文章，相当地清晰。


回复

-  *hongruiqi* says:
March 9, 2012 at 11:24 am
如果发布了新的版本之后，需要对旧的版本打hotfix，有什么解决方法没？

回复

-  *wing* says:
April 1, 2012 at 9:02 pm
很朴实但内容很丰富的站点,特此表示感谢一下. 😊

回复

-  *hefabao* says:
September 7, 2012 at 10:00 am
非常好的文章,谢谢楼主

回复







rocrocket Reply:
September 7th, 2012 at 9:46 pm

多谢支持！继续努力！

回复

- *Git 分支管理是一门艺术 - 博客 - 伯乐在线* says:
September 22, 2012 at 1:33 am
[...] 11:01 来源：Linux大棚 分类：程序员 都等你发言 😊 分享到： (function(){ var [...]

-  小拿 says:
November 16, 2012 at 5:22 pm
对于Git分支管理分析的很好，值得借鉴！
回复
- *Git学习笔记 (4) - 分支 | 架构 (architecture.riao.com)* says:
December 27, 2012 at 12:27 pm
[...] 这里再次推荐童鞋先看看这篇文章，以对Git的工作组成和分支概念有直观的了解 《Git分支管理是一门艺术》 [...]
- *Git学习笔记 (2) - Git基本操作 | 架构 (architecture.riao.com)* says:
December 27, 2012 at 12:28 pm
[...] 关于分支的妙用，这里推荐一个篇文章 《Git分支管理是一门艺术》 [...]
- *Git学习笔记 (2) - Git基本操作 | 移动开发 (mobile.riao.com)* says:
December 27, 2012 at 12:59 pm
[...] 关于分支的妙用，这里推荐一个篇文章 《Git分支管理是一门艺术》 [...]
-  zebulon says:
September 27, 2013 at 4:55 pm
谢谢，学习了~
回复
-  smileEvday says:
May 25, 2014 at 5:17 pm
多写楼主翻译，看着很清晰
回复
-  array020 says:
July 27, 2014 at 2:13 pm
希望楼主能结合github来讲讲git~~毕竟不少人是因为github才转移到git上来的吧~
回复

Leave a Reply