## **Git的优点**

Git的优点很多，但是这里只列出我认为非常突出的几点。

1. 由于是分布式，所有本地库包含了远程库的所有内容。
2. 优秀的分支模型，打分支以及合并分支，机器方便。
3. 快速，在这个时间就是金钱的时代，Git由于代码都在本地，打分支和合并分支机器快速，使用个SVN的能深刻体会到这种优势。

感兴趣的，可以去看一下Git本身的设计，内在的架构体现了很多的优势，不愧是出资天才程序员Linus (Linux之父) 之手

## **版本管理的挑战**

虽然有这么优秀的版本管理工具，但是我们面对版本管理的时候，依然有非常大得挑战，我们都知道大家工作在同一个仓库上，那么彼此的代码协作必然带来很多问题和挑战，如下：

1. 如何开始一个Feature的开发，而不影响别的Feature？
2. 由于很容易创建新分支，分支多了如何管理，时间久了，如何知道每个分支是干什么的？
3. 哪些分支已经合并回了主干？
4. 如何进行Release的管理？开始一个Release的时候如何冻结Feature, 如何在Prepare Release的时候，开发人员可以继续开发新的功能？
5. 线上代码出Bug了，如何快速修复？而且修复的代码要包含到开发人员的分支以及下一个Release?

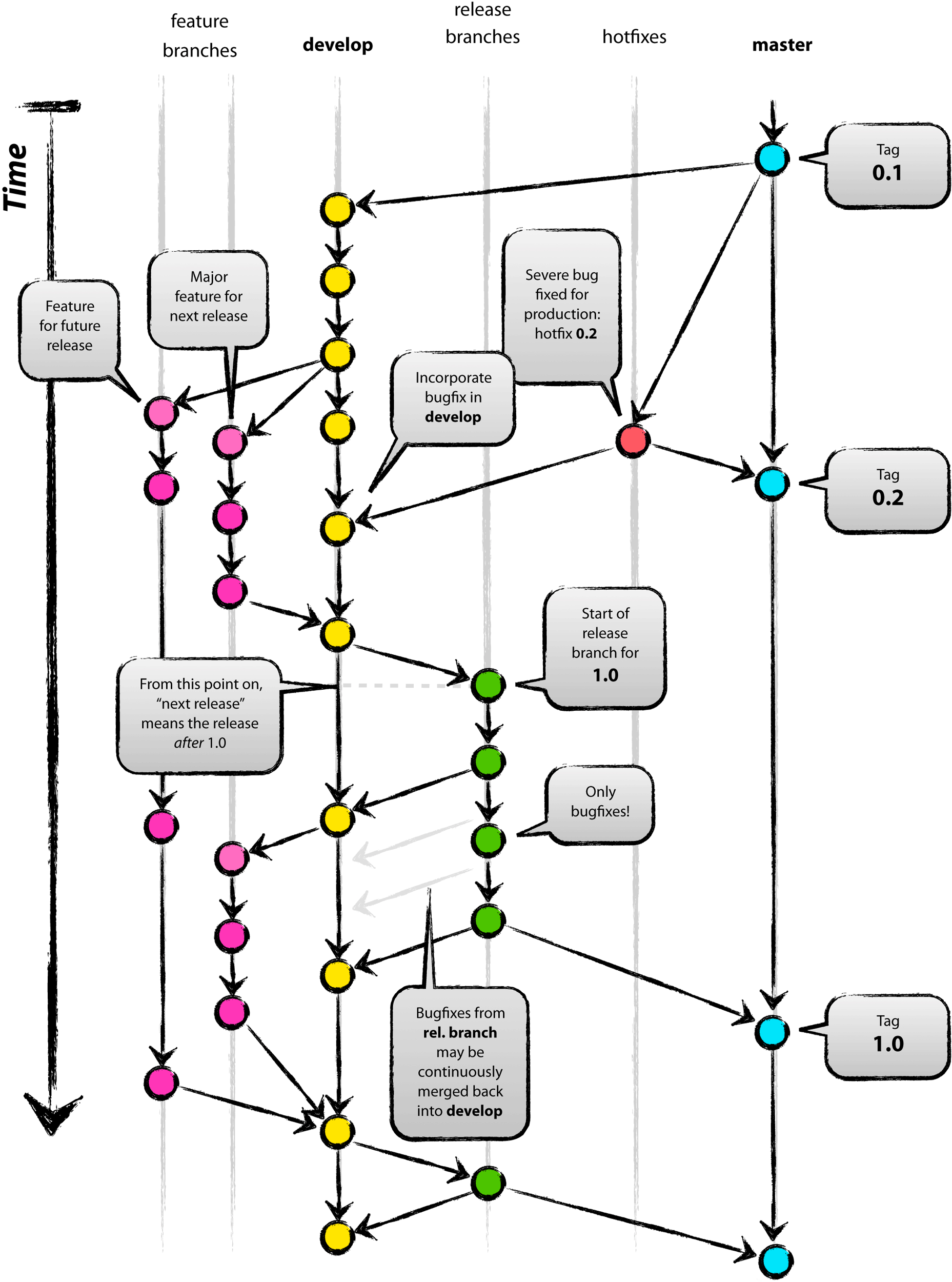
大部分开发人员现在使用Git就只是用三个甚至两个分支，一个是Master, 一个是Develop, 还有一个是基于Develop打得各种分支。这个在小项目规模的时候还勉强可以支撑，因为很多人做项目就只有一个Release, 但是人员一多，而且项目周期一长就会出现各种问题。

## **Git Flow**

就像代码需要代码规范一样，代码管理同样需要一个清晰的流程和规范

Vincent Driessen 同学为了解决这个问题提出了 [A Successful Git Branching Model](http://nvie.com/posts/a-successful-git-branching-model/)

下面是Git Flow的流程图



上面的图你理解不了？ 没关系，这不是你的错，我觉得这张图本身有点问题，这张图应该左转90度，大家应该就很用以理解了。

### **Git Flow常用的分支**

* Production 分支

也就是我们经常使用的Master分支，这个分支最近发布到生产环境的代码，最近发布的Release， 这个分支只能从其他分支合并，不能在这个分支直接修改

* Develop 分支

这个分支是我们是我们的主开发分支，包含所有要发布到下一个Release的代码，这个主要合并与其他分支，比如Feature分支

* Feature 分支

这个分支主要是用来开发一个新的功能，一旦开发完成，我们合并回Develop分支进入下一个Release

* Release分支

当你需要一个发布一个新Release的时候，我们基于Develop分支创建一个Release分支，完成Release后，我们合并到Master和Develop分支

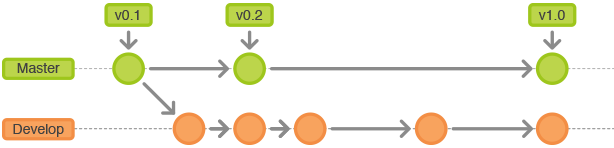
* Hotfix分支

当我们在Production发现新的Bug时候，我们需要创建一个Hotfix, 完成Hotfix后，我们合并回Master和Develop分支，所以Hotfix的改动会进入下一个Release

### **Git Flow如何工作**

#### **初始分支**

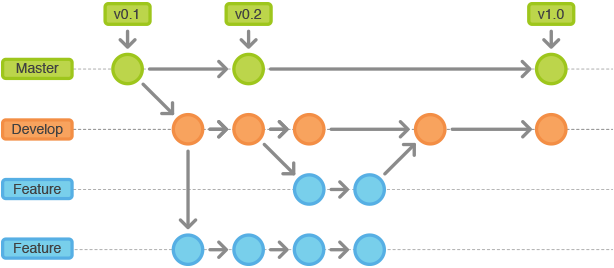
所有在Master分支上的Commit应该Tag



#### **Feature 分支**

分支名 feature/\*

Feature分支做完后，必须合并回Develop分支, 合并完分支后一般会删点这个Feature分支，但是我们也可以保留

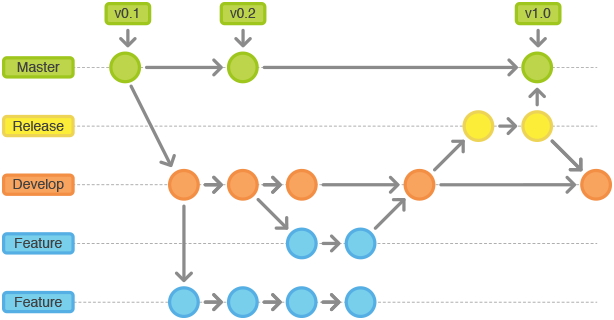


#### **Release分支**

分支名 release/\*

Release分支基于Develop分支创建，打完Release分之后，我们可以在这个Release分支上测试，修改Bug等。同时，其它开发人员可以基于开发新的Feature (记住：一旦打了Release分支之后不要从Develop分支上合并新的改动到Release分支)

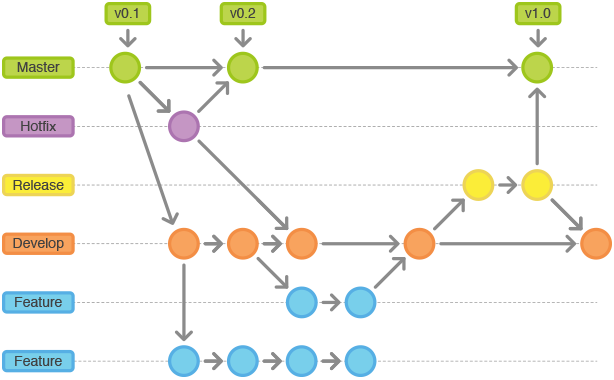
发布Release分支时，合并Release到Master和Develop， 同时在Master分支上打个Tag记住Release版本号，然后可以删除Release分支了。



#### **维护分支 Hotfix**

分支名 hotfix/\*

hotfix分支基于Master分支创建，开发完后需要合并回Master和Develop分支，同时在Master上打一个tag



### **Git Flow代码示例**

a. 创建develop分支

git branch develop

git push -u origin develop

b. 开始新Feature开发

git checkout -b some-feature develop# Optionally, push branch to origin:

git push -u origin some-feature

# 做一些改动

git status

git add some-file

git commit

c. 完成Feature

git pull origin develop

git checkout develop

git merge --no-ff some-feature

git push origin develop

git branch -d some-feature

# If you pushed branch to origin:

git push origin --delete some-feature

d. 开始Relase

git checkout -b release-0.1.0 develop

# Optional: Bump version number, commit

# Prepare release, commit

e. 完成Release

git checkout master

git merge --no-ff release-0.1.0

git push

git checkout develop

git merge --no-ff release-0.1.0

git push

git branch -d release-0.1.0

# If you pushed branch to origin:

git push origin --delete release-0.1.0

git tag -a v0.1.0 master

git push --tags

f. 开始Hotfix

git checkout -b hotfix-0.1.1 master

g. 完成Hotfix

git checkout mastergit merge --no-ff hotfix-0.1.1git push

git checkout developgit merge --no-ff hotfix-0.1.1git push

git branch -d hotfix-0.1.1

git tag -a v0.1.1 mastergit push --tags

### **Git flow工具**

实际上，当你理解了上面的流程后，你完全不用使用工具，但是实际上我们大部分人很多命令就是记不住呀，流程就是记不住呀，肿么办呢？

总有聪明的人创造好的工具给大家用, 那就是Git flow script.

#### **安装**

* OS X

brew install git-flow

* Linux

apt-get install git-flow

* Windows

wget -q -O - --no-check-certificate <https://github.com/nvie/gitflow/raw/develop/contrib/gitflow-installer.sh> | bash

#### **使用**

**初始化:** git flow init

**开始新Feature:** git flow feature start MYFEATURE

**Publish一个Feature(也就是push到远程):** git flow feature publish MYFEATURE

**获取Publish的Feature:** git flow feature pull origin MYFEATURE

**完成一个Feature:** git flow feature finish MYFEATURE

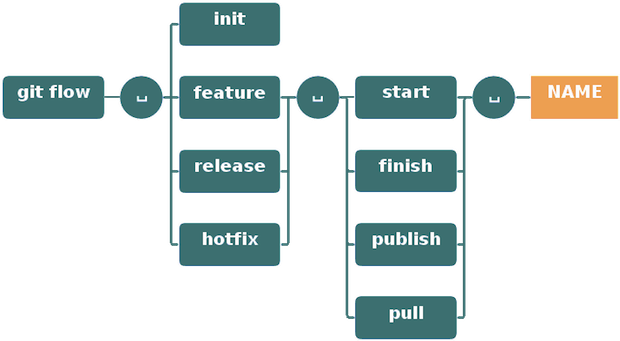
**开始一个Release:** git flow release start RELEASE [BASE]

* **Publish一个Release:** git flow release publish RELEASE

**发布Release:** git flow release finish RELEASE  
别忘了git push --tags

**开始一个Hotfix:** git flow hotfix start VERSION [BASENAME]

**发布一个Hotfix:** git flow hotfix finish VERSION



### **Git Flow GUI**

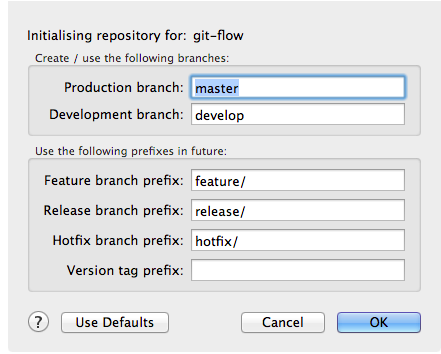
上面讲了这么多，我知道还有人记不住，那么又有人做出了GUI 工具，你只需要点击下一步就行，工具帮你干这些事！！！

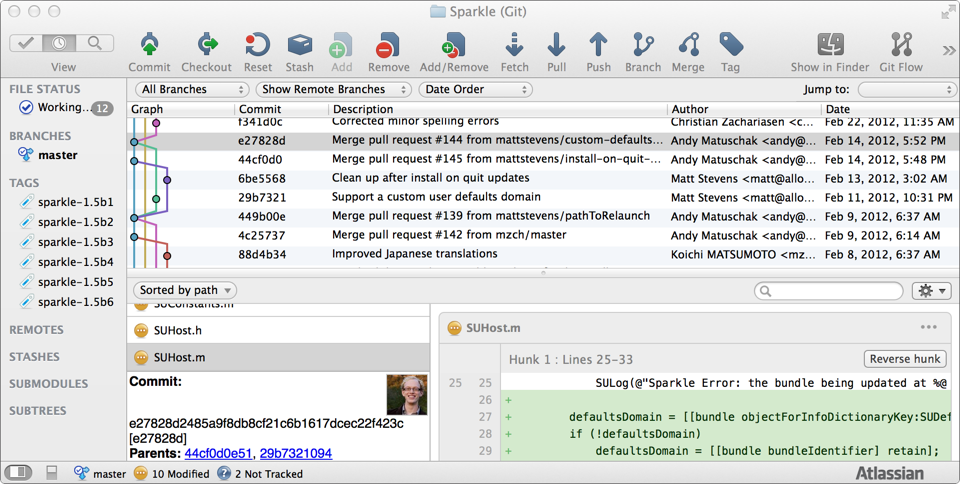
#### **SourceTree**

当你用Git-flow初始化后，基本上你只需要点击git flow菜单选择start feature, release或者hotfix, 做完后再次选择git flow菜单，点击Done Action. 我勒个去，我实在想不到还有比这更简单的了。

目前SourceTree支持Mac, Windows, Linux.

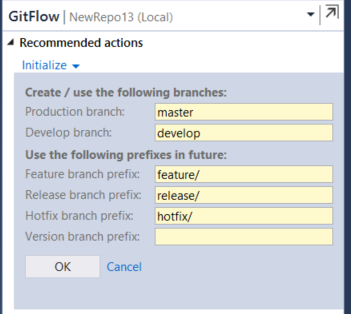
这么好的工具请问多少钱呢？ **免费!!!!**

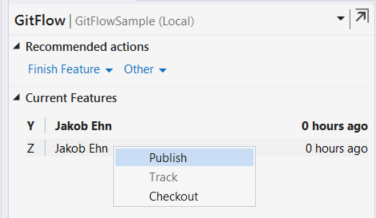


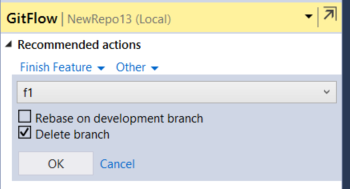


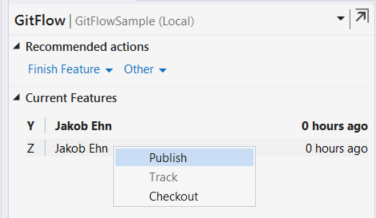
#### **Git flow for visual studio**

广大VS的福音  
[GitFlow for Visual Studio](https://visualstudiogallery.msdn.microsoft.com/27f6d087-9b6f-46b0-b236-d72907b54683)









## **最后**

我经常收到邮件问我，他想使用Git, 但是公司还在坚持使用SVN等，问我最么办？ 我的办法是：

第一： 把我这篇文章给他看

第二： 立即找我，加入我们公司，我的邮箱是wangdeshui@outlook.com QQ: 353275476

作者: 王德水  
出处：http://www.cnblogs.com/cnblogsfans  
版权：本文版权归作者所有,转载需经作者同意。