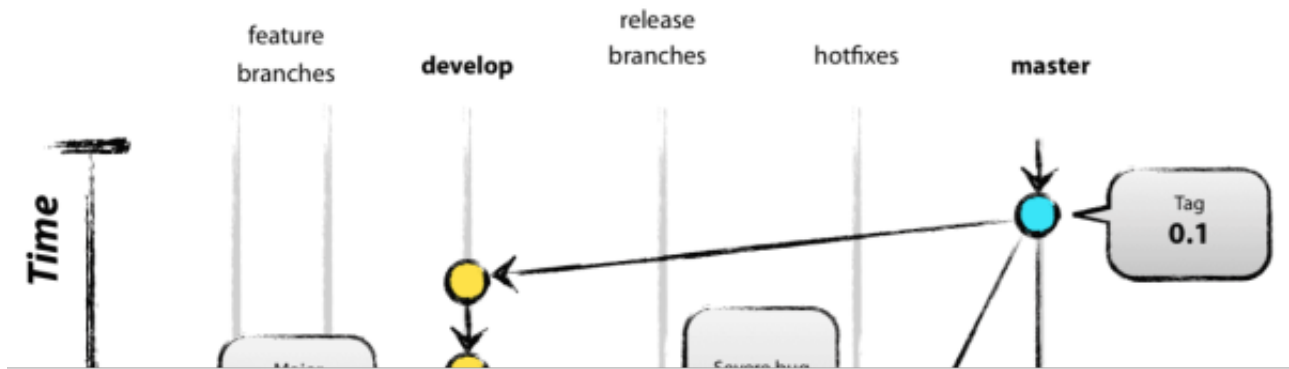


图文详解如何利用Git+Github进行团队协作开发



团队协作开发中，大部分都会用到版本控制软件，比如Git、Svn等。本文将通过一个实例，详细讲解在真实的工作环境中，一个团队应该如何利用Git+Github进行协作开发，即详解Git工作流程。并就其中比较棘手的问题作出解答，比如如何解决冲突比较合适，如何建立各种类型的分支等。

本文不会讲解Git简介、Git原理、Git基本用法等，有不了解的可以参考“Git 参考手册”。我们举例演示的是GitFlow工作流的功能，这里先放一张经典的GitFlow工作流图示：



其中涉及到的主要分支类型有：

- master分支，即主分支。任何项目都必须有个这个分支。对项目进行

tag或发布版本等操作，都必须在该分支上进行。

- develop分支，即开发分支，从master分支上检出。团队成员一般不会直接更改该分支，而是分别从该分支检出自己的feature分支，开发完成后将feature分支上的改动merge回develop分支。同时release分支由此分支检出。
- release分支，即发布分支，从develop分支上检出。该分支用作发版前的测试，可进行简单的bug修复。如果bug修复比较复杂，可merge回develop分支后由其他分支进行bug修复。此分支测试完成后，需要同时merge到master和develop分支上。
- feature分支，即功能分支，从develop分支上检出。团队成员中每个人都维护一个自己的feature分支，并进行开发工作，开发完成后将此分支merge回develop分支。此分支一般用来开发新功能或进行项目维护等。
- fix分支，即补丁分支，由develop分支检出，用作bug修复，bug修复完成需merge回develop分支，并将其删除。所以该分支属于临时性分支。
- hotfix分支，即热补丁分支。和fix分支的区别在于，该分支由master分支检出，进行线上版本的bug修复，修复完成后merge回master分支，并merge到develop分支上，merge完成后也可以将其删除，也属于临时性分支。

下边我们一步步拆分讲解各种类型分支的用法。

(1) 假设团队就一个人“xianhu”，做一个叫TestGit的项目，并将其代码托管在Github上。首先需要在Github上新建一个项目TestGit：



按照Github上的提示，在本地新建一个项目，并关联到Github上的origin/master。此时开发一个很小的demo功能，并提交到线上，并在master分支上进行打tag操作，并命名为v0.1。此时的GitFlow工作流为：

All Local "master"			Q Subject, Author, SHA	
Short SHA	Subject		Author	
9d704fe	master	origin/master v0.1	done demo in master xianhu	

(2) 如果此时master分支的代码正在线上运行，而且又需要开发新功能，则不能在master分支上直接修改。一个比较好的策略是在master分支上新建并检出develop分支，新功能的开发在develop分支上进行，此时记得将develop分支提交到远端：

```
git branch develop master    # 从master分支上新建develop分支
git checkout develop         # 检出develop分支
# 此处可进行功能开发，并add和commit到develop分支
git push origin develop      # 推送develop分支到远端的origin/develop
```

即Github上保持两个分支：master和develop。目的是为以后团队协作更新develop分支做准备。此时Github上为：



此时会出现两种情况：

- 线上版本的代码（master分支）出现了紧急bug，需要修复。这里用到了hotfix分支。

```
git checkout master      # 切换回master分支
git checkout -b hotfix master  # 新建hotfix分支，并切换到该分支
# 做一些bug修复工作
git checkout master      # 切换回master分支
git merge --no-ff hotfix    # 合并hotfix分支，此时bug已被修复（无冲突）
git tag v0.2              # 新建tag v0.2
git push origin master     # 推送master分支代码到远端
git push origin --tags     # 推送tag到远端
```

- develop分支上的功能开发完成了，需要进行测试和提交。这里用到了release分支。

```
git checkout develop     # 切换回develop分支
git checkout -b release01 develop  # 新建release分支，并切换到该分支

# 做一些测试、bug修复等工作

git checkout develop     # 切换回develop分支
git merge --no-ff release01  # 合并release01分支到develop分支（无冲突）
git push origin develop    # 推送develop分支到远端

git checkout master      # 切换回master分支
git merge --no-ff release01  # 合并release01分支到master分支（无冲突）
git tag v0.3              # 新建tag v0.3
git push origin master     # 推送master分支代码到远端
git push origin --tags     # 推送tag到远端
```

此时GitFlow工作流为：









(3) 这里可以继续develop分支，并不断push到远端。此时如果团队成员增加，多人需要开发不同的功能，这里就会用到feature分支。团队中的每个人都从Github克隆一个项目，然后新建自己的feature分支。

```
git clone xxxx.git
git checkout develop
git checkout -b feature-xx develop    # 从develop分支新建并检出feature分支
```

此时“xianhu”做如下操作，并首先第一个提交到了Github远端：

```
git checkout -b feature-hu develop    # 从develop分支新建并检出feature分支
# 这里可以进行一些功能开发，并不断的add和commit
git checkout develop                  # 切换回develop分支
git pull origin develop               # 更新远端代码，看develop分支是否有更新（无更新）
git checkout feature-hu               # 切换回feature分支
git rebase develop                    # 合并develop分支到feature分支，并解决冲突（无冲突）
git checkout develop                  # 切换回develop分支
git merge --no-ff feature-hu          # 合并feature分支到develop分支
git push origin develop               # 推送develop分支到远端
```

此时的GitFlow工作流为：

Short SHA	Subject	Author
ba7d119	 develop  Merge branch 'feature-hu' into develop, no conflict	xianhu
95c03c2	 feature-hu update feature_hu.txt in feature-hu	xianhu
794a57d	add feature_hu.txt, feature.txt in feature-hu	xianhu
ff536e7	continue develop branch	xianhu
b449d9b	Merge branch 'release01' into develop	xianhu
6ab373c	 master  v0.3 Merge branch 'release01' into master	xianhu
b51e649	fix bugs in release01	xianhu
f5558f0	update develop.txt in develop	xianhu
b2ac201	add develop.txt in develop	xianhu
35c04cf	v0.2 Merge branch 'hotfix', fix bugs in master	xianhu
ea6474f	 hotfix fix bugs in hotfix	xianhu
9d704fe	v0.1 done demo in master	xianhu

对于团队其他成员，比如zz，操作如下，并打算在“xianhu”提交后进行push操作：

```
git checkout -b feature-zz develop    # 从develop分支新建并检出feature分支
# 这里可以进行一些功能开发，并不断的add和commit
git checkout develop                # 切换回develop分支
git pull origin develop             # 更新远端代码，看develop分支是否有更新（有更新）
git checkout feature-hu             # 切换回feature分支
git rebase develop                  # 合并develop分支到feature分支，并解决冲突（有冲突）
# 这里需要进行冲突解决
git add .                           # 解决完冲突之后执行add操作
git rebase --continue                # 继续刚才的rebase操作
git checkout develop                # 切换回develop分支
git merge --no-ff feature-zz         # 合并feature分支到develop分支（无冲突）
git push origin develop              # 推送develop分支到远端
```

如果团队成员在合并feature分支到develop分支后还需要继续开发，则检出自己的feature分支继续操作，并重复上述过程即可。这里需要注意--no-ff参数，其目的是让commit的流程更加清晰，具体为什么可自行百度。

xianhu这边进行pull操作，即可看到zz进行的操作。此时的GitFlow工作流为：

Short SHA	Subject	Author
a5aead8	develop origin/develop Merge branch 'feature-zz' into develop	xianhu
830a0fb	add somefile in feature-zz	xianhu
6c2f390	Merge branch 'feature-hu' into develop	xianhu
95c03c2	feature-hu update feature_hu.txt in feature-hu	xianhu
794a57d	add feature_hu.txt, feature.txt in feature-hu	xianhu
ff536e7	continue develop branch	xianhu
b449d9b	Merge branch 'release01' into develop	xianhu
6ab373c	master origin/master v0.3 Merge branch 'release01' into master	xianhu
b51e649	fix bugs in release01	xianhu
f5558f0	update develop.txt in develop	xianhu
b2ac201	add develop.txt in develop	xianhu
35c04cf	v0.2 Merge branch 'hotfix', fix bugs in master	xianhu
ea6474f	hotfix fix bugs in hotfix	xianhu
9d704fe	v0.1 done demo in master	xianhu

(4) 如果此时需要进行测试、发版操作，则需要再次新建并检出release分支，重复（2）的步骤。完成之后的GitFlow工作流为：

Short SHA	Subject	Author
a01a702	master origin/master v1.0 Merge branch 'release02' into master	xianhu
48efc75	develop Merge branch 'release02' into develop	xianhu
5f9d6a0	release02 fix bugs in release02	xianhu
a5aead8	origin/develop Merge branch 'feature-zz' into develop	xianhu
830a0fb	add somefile in feature-zz	xianhu
6c2f390	Merge branch 'feature-hu' into develop	xianhu
95c03c2	feature-hu update feature_hu.txt in feature-hu	xianhu
794a57d	add feature_hu.txt, feature.txt in feature-hu	xianhu
ff536e7	continue develop branch	xianhu
b449d9b	Merge branch 'release01' into develop	xianhu
6ab373c	v0.3 Merge branch 'release01' into master	xianhu
b51e649	fix bugs in release01	xianhu
f5558f0	update develop.txt in develop	xianhu
b2ac201	add develop.txt in develop	xianhu
35c04cf	v0.2 Merge branch 'hotfix', fix bugs in master	xianhu
ea6474f	hotfix fix bugs in hotfix	xianhu
9d704fe	v0.1 done demo in master	xianhu

这里就完成了从版本v0.1到v1.0的迭代开发，并详细解释了如何利用Git+Github进行团队协作开发。文章中没有用到fix分支，这种分支的用法和feature分支类似，大家可以自己体会。

最后详细解释一下上张图：

- xianhu在master分支上完成v0.1版本开发，“done demo in master”
- 发现master分支上有bug，需紧急修复。新建并检出hotfix分支进行bug修复，并merge回master分支，发布版本v0.2。
- xianhu新建并检出develop分支进行迭代开发，然后在develop分支上检出release01分支，进行发版前测试和bug修复。“fix bugs in

release01”，完成后将release01分支merge回develop和master分支，并在master分支上发布版本v0.3。

- xianhu继续开发develop分支。此时团队成员增加，团队中的每个人都在develop的基础上新建并检出自己的feature分支，开发完成后merge回develop分支。这里利用到了rebase操作和冲突解决等，需要特别注意一点步骤。
- xianhu在develop分支上检出release02分支，再次进行发版前测试和bug修复，“fix bugs in release02”，完成后将release02分支merge回develop和master分支，并在master分支上发布版本v1.0。

这里只用到和解释了GitFlow工作流，团队协作开发也可以用Pull Requests或者其他方式，找一个合适自己团队和具体业务的协作方式即可。另外，Git博大精深，想要完全精通也绝非易事，只能是一边学一边用。

来自：<https://zhuanlan.zhihu.com/p/23478654>