一、git分支常用操作

Git 相对于 SVN 最强大的优势就在于「分支」，Git 的分支操作简单方便，而实际项目开发中团队合作最依赖的莫过于分支了，关于分支前面的课时也提到过，本课时会详细讲述什么是分支、分支的具体操作以及实际项目开发中是怎样依赖分支进行团队合作的。

**1. 什么是分支？**

对于分支这个概念，可以这样理解，几个人一起去旅行，走到一个三岔口，每条路可能有不同的风景，大家约定 3 天之后在某地汇聚，然后各自出发了。这三条分叉路就可以理解为各自的分支，而大家汇聚的时候就相当于将各自的分支进行了合并。  

**2. 分支的常用操作**

通常默认会有一个主分支叫 master ，下面首先来介绍一下关于分支的一些基本操作：

（1）新建一个叫 develop 的分支

git branch develop 

注意，新建分支的命令是基于当前所在分支进行的，即以上是基于 mater 分支新建了一个叫做 develop 的分支，此时 develop 分支与 master 分支的内容完全一样。例如，有 A、B、C三个分支，各分支内容不同，如果当前是在 B 分支，执行新建分支命令，则新建的分支内容与 B 分支相同，同理如果当前在 C 分支，那就是基于 C 分支基础上新建的分支。

（2）切换到 develop 分支

git checkout develop

如果把以上两步合并，即新建并自动切换到 develop 分支：

git checkout -b develop 

（3）把 develop 分支推送到远程仓库

git push origin develop

如果你远程的分支想取名叫 develop2 ，执行以下代码：

git push origin develop:develop2

注意：实际开发管理，建议不要这样做，这样会导致很混乱，难管理，建议本地分支与远程分支名称要保持一致。

（4）查看本地分支列表

git branch 

（5）查看远程分支列表

git branch -r

（6）删除本地分支

git branch -d develop

git branch -D develop (强制删除)

（7）删除远程分支

git push origin :develop

如果远程分支有 develop ，而本地没有，想要将远程的 develop 分支迁到本地：

git checkout develop origin/develop

同样的把远程分支迁到本地顺便切换到该分支：

git checkout -b develop origin/develop

二、基本的团队协作流程

一般来说，如果是一个人开发，可能只需要 master、develop 两个分支就可以了，平时开发在 develop 分支进行，开发完成之后，发布之前合并到 master 分支，这个流程不会有大问题。

如果是 3、5 个人，那就不一样了，有人觉得也不会有什么大问题，直接新建 A、B、C 三个人的分支，每人各自开发各自的分支，然后开发完成之后再逐步合并到 master 分支，然而现实情况是，当某人正在某个分支开发某个功能时，突然发现线上有一个很严重的 bug ，不得不停下手头的工作优先处理 bug ，并且很多时候多人协作下如果没有一个规范，很容易产生问题，所以多人协作下的分支管理规范很重要，就像代码规范一样重要，接下来就向大家推荐一种分支管理流程 Git Flow。

三、GitFlow详细操作

准确的说 Git Flow 是一种比较成熟的分支管理流程，我们先看一张图能清晰的描述他整个的工作流程：



第一次看上面的图是不是有些晕，接下来用简单的语言给大家解释一下。

一般开发来说，大部分情况下都会拥有两个分支 master 和 develop，他们的职责分别是：

* **master：永远处在即将发布(production-ready)状态**
* **develop：最新的开发状态**

确切的说 master、develop 分支大部分情况下都会保持一致，只有在上线前的测试阶段 develop 比 master 的代码要多，一旦测试没问题，准备发布时，会将 develop 合并到 master 上。

但是产品发布之后又会进行下一版本的功能开发，开发中间可能又会遇到需要紧急修复的 bug ，一个功能开发完成之后突然需求变动了等情况，所以 Git Flow 除了以上 master 和 develop 两个主要分支以外，还提出了以下三个辅助分支：

* **feature:  开发新功能的分支, 基于 develop, 完成后 merge 回 develop**
* **release: 准备要发布版本的分支, 用来修复 bug，基于 develop，完成后 merge 回 develop 和 master**
* **hotfix: 修复 master 上的问题, 等不及 release 版本就必须马上上线. 基于 master, 完成后 merge 回 master 和 develop**

举个例子，假设已经有 master 和 develop 两个分支了，这个时候准备做一个功能 A，第一步要做的就是基于 develop 分支新建个分支：

git branch feature/A 

其实就是一个规范，规定了所有开发的功能分支都以 feature 为前缀。

但是这个时候发现线上有一个紧急的 bug 需要修复，可以立刻切换到 master 分支，然后再此基础上新建一个分支：

git branch hotfix/B

代表新建一个紧急修复分支，修复完成之后直接合并到 develop 和 master ，然后发布。

然后再切换回 feature/A 分支继续开发，如果开发完了，那么合并回 develop 分支，然后在 develop 分支属于测试环境，跟后端对接并且测试，感觉可以发布到正式环境了，这个时候再新建一个 release 分支：

git branch release/1.0

此时，所有的 api、数据等都是正式环境，然后在这个分支上进行最后的测试，发现 bug 直接进行修改，直到测试，达到发布的标准，就可以把该分支合并到 develop 和 master 然后进行发布。

以上就是 Git Flow 的概念与大体流程，看起来有些复杂，但是对于人数比较多的团队协作现实开发中确实会遇到这种复杂的情况， Git Flow 是目前很流行的一套分支管理流程，但是有人会觉得每次都要进行各种合并操作有些麻烦， Git Flow 为此专门推出了一个工具，并且是开源的：GitHub 开源地址：[[https://github.com/nvie/gitflow]](https://github.com/nvie/gitflow%5d)

简单来说，这个工具会帮大家节省很多步骤，例如，当前处于 master 分支，如果想要开发一个新的功能，第一步切换到 develop 分支，第二步新建一个以 feature 开头的分支名，有了 Git Flow 直接如下操作就可以完成了： 

git flow feature start A

这个分支完成之后，需要合并到 develop 分支，进行如下操作：

git flow feature finish A

如果是 hotfix 或者 release 分支，会自动合并到 develop、master 两个分支。    

到目前为止，大家已经了解了这个工具的具体作用，具体安装与用法就不多介绍了，感兴趣的同学可以观看这篇博客：[http://stormzhang.com/git/2014/01/29/git-flow/](http://stormzhang.com/git/2014/01/29/git-flow/%5d)

**总结**

以上就是分享给大家的关于分支的所有知识，一个人你也许感受不到什么，但是实际工作中大都以团队协作为主，而分支是团队协作必备技能，Git Flow 是一个很流行的分支管理流程，也是公司团队内部经常使用的一套流程，希望大家能够掌握。