汕学派第一次博客之GIT学习

2020/12/11 林嘉声

1. 基础概念及一般并行开发流程：

1.1 Git flow中基础名词概念：

(1) 仓库（Repository）：本质上是一个文件目录，目录所有文件被git管理，无论任何操作如增删改等都会被记录。仓库又分为源仓库和开发者仓库。
1）源仓库（origin）：存放于GitHub，主要作用是：1.汇总开发者代码2.存放稳定可发布代码。
2）开发者仓库： 开发者clone源仓库到本地的日常开发仓库，与源仓库关联，随时可通过push和pull操作与源仓库进行关联。
(2) 分支（Branch）：主要作用将开发者工作从开发主线中分离开来，在开发中会频繁使用git提供的创建、合并、删除分支等操作。
1) 主分支：存在于整个项目的开始、开发、迭代、终止过程。
Master：主分支，经过测试的完全稳定代码，任何时刻都是可供用户直接使用的，更新tag时代表有新版本发布。
Develop：开发分支，最初从master中分离出来，开发者用于存放基本稳定代码，保持着当前最新的开发成果。
2) 辅助分支：开发过程中临时存在且会被删除的分支。
Feature：功能性分支，用于开发者开发功能。开发完成后要merge到develop分支。
Release：预发布分支，可从develop分支中派生，用于发布前调整测试。发布完成后打tag合并入master和develop。
Hotfix：修复bug分支，可从master中派生，紧急bug修复上线版本的问题。修复后打tag并合入master和develop。
(3) 转移测试（conversion testing）：测试已有系统的数据能否转换到替代系统。

1.2 一般的git并行开发流程：

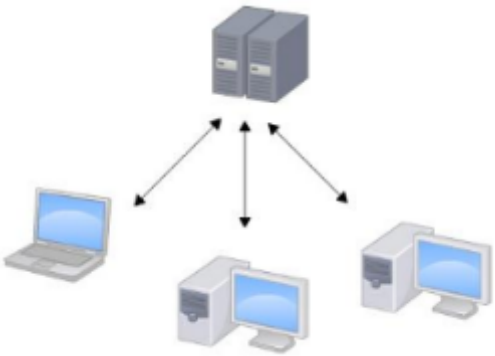
(1) 创建源仓库，初始化master和develop，添加团队开发者。
(2) 开发者clone源仓库，并创建自己的分支 develop-name。
(3) 在自己本地的develop-name开发，从develop切出新分支，根据是功能还是bug，命名feature-*或者fixbug-*。
(4) 完成开发后提交commit，推送到远端。
(5) 发起merge请求，在GitHub上pull request，在code reviewer进行review并且在其同意后才能把自己的分支merge到develop分支。（为防止冲突，在发起合并请求前，开发者应该将develop分支合并到自己分支，提前查看合并的冲突，在解决冲突后再pull request）。
(6) Reviewer进行code review，可在GitHub上进行在线review也可在本地创建测试分支进行review。
(7) Reviewer经过测试没有问题，同意可以把代码合并到源仓库的develop分支中。否则close该请求，通知开发者进行调整。

(8) 转测时，从当前develop分支，直接合并develop到pre-release分支，重新构建测试环境完成转测。
(9) 转测完成后，从pre-release分支合并到master分支，基于master分支构建生产环境完成上线。同时对master分支打tag，tag名字为版本号_上线时间。

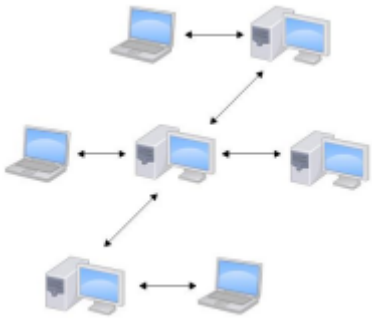
2. Git分布式版本控制工作原理：

依我的理解，分布式（DVCS）管理的特点就是去中心化，相比于SVN等所有成员共享一个仓库服务器的集中式管理，分布式控制管理下每个客户端都是一个服务器，都有一个版本库，可以在无网络的条件下进行对代码的操作，可以在同一个项目中，分别和不同工作小组的人相互协作。 你可以根据需要设定不同的协作流程，比如层次模型式的工作流，而这在以前的集中式系统中是无法实现的。

SVN集中式版本控制系统：



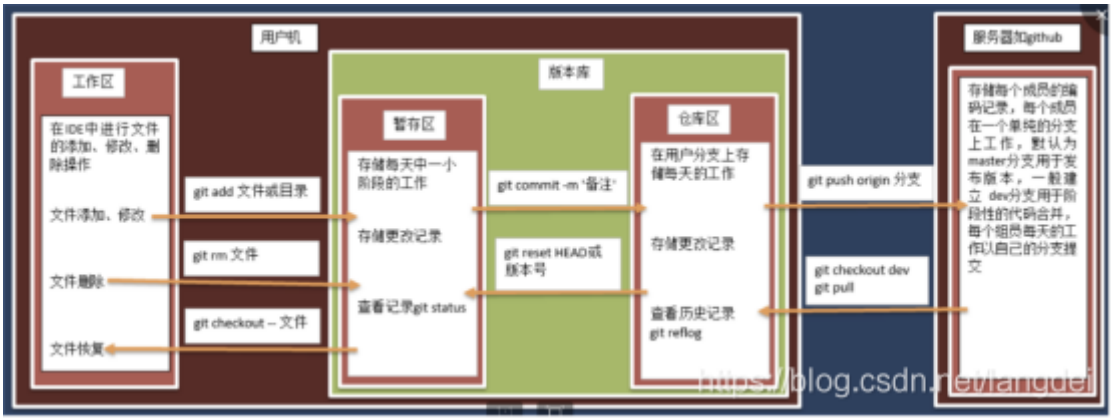
DVCS分布式版本控制系统：



（小厂常用SVN 大厂常用GIT 我选GIT！）

Git分为三四个区：

（1） 工作区（workspace）： 对于文件的添加，修改，删除。
（2） 暂存区（stage）： 将工作区的操作完成小阶段的存储，是版本库的一部分。
（3） 本地仓库（repository）： 包含所有版本信息。
（4） 远程仓库（remote）： 代码托管服务器，如GitHub。

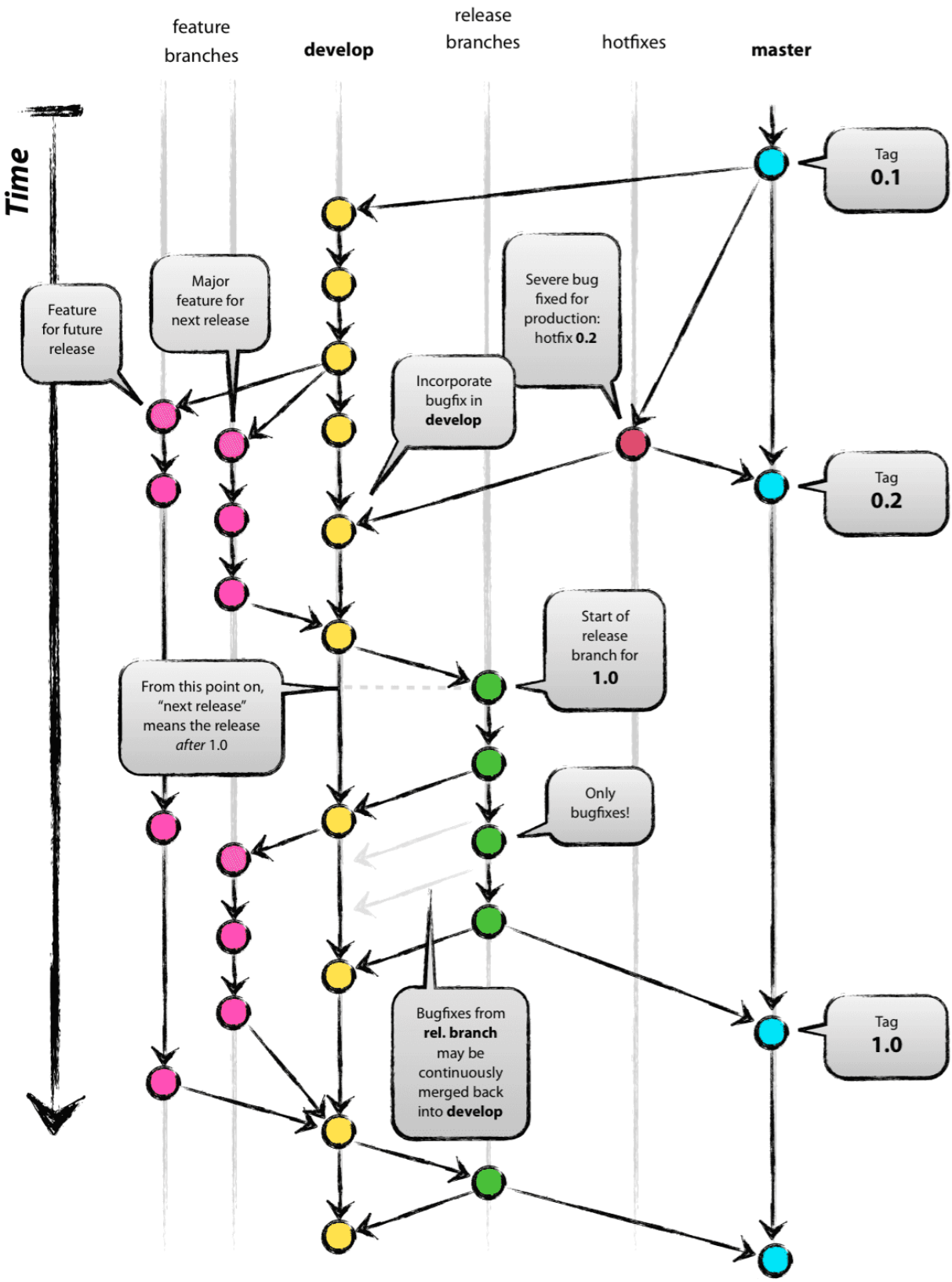


3. Git flow并行开发分支模型：

Git flow：Git Flow是构建在Git之上的一个组织软件开发活动的模型，是在Git之上构建的一项软件开发最佳实践。Git Flow是一套使用Git进行源代码管理时的一套行为规范和简化部分Git操作的工具。
Git Flow重点解决的是由于源代码在开发过程中的各种冲突导致开发活动混乱的问题。
Git Flow开发模型让开发代码仓库保持整洁，让小组各个成员之间的开发相互隔离，能够有效避免处于开发状态中的代码相互影响而导致的效率低下和混乱。

Git flow模型定义了主分支和辅助分支：

主分支：master, develop
辅助分支：feature，release，hotfix
关于分支我理解的概念已经在题一中解释，在此不做赘述，重点在下面这张经典的图。


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0