Git 新手指南

— SmartGit 的实战操作

Git 的优势

- 分布式，本地仓库包含了远程库的所有内容
- 快速，所有的操作都是本地执行
- 优秀的分支模型
- 无尽的后悔药

Git 文件的三种状态

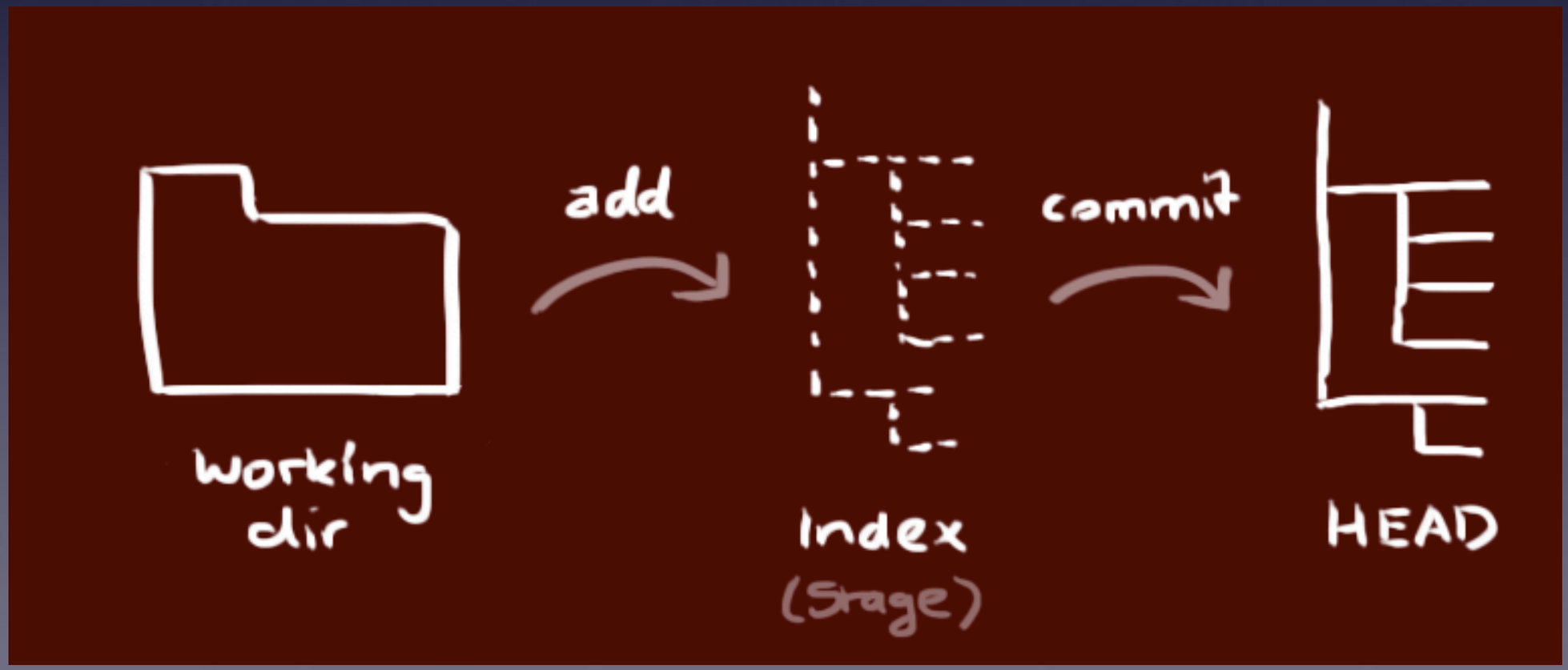
- 已提交(committed)
- 已修改(modified)
- 已暂存(staged)

Git 的三个工作区

- 工作目录： 持有实际文件
- 暂存区（stage）： 缓存区域，临时保存改动
- Git 仓库（HEAD）： 指向最后一次提交的结果

Git 基本的工作流

- 在工作目录中修改文件
- 暂存文件，将文件的快照放入暂存区域
- 提交更新，找到暂存区域的文件，将快照永久性储存到 Git 仓库目录



版本管理的挑战

- 如何开始一个 Feature 的开发，而不影响别的 Feature
- 分支多了要如何管理，时间久了，如何知道每个分支是干什么的
- 哪些分支已经合并回了主干
- 如何进行 Release 的管理？如何在Prepare Release的时候，开发人员可以继续开发新的功能?
- 线上的代码出现 Bug 了，如何快速修复？而且修复的代码要包含到Develop分支以及下一个Release?

Git-Flow

A Successful Git Branching Model

Git Flow 流程图

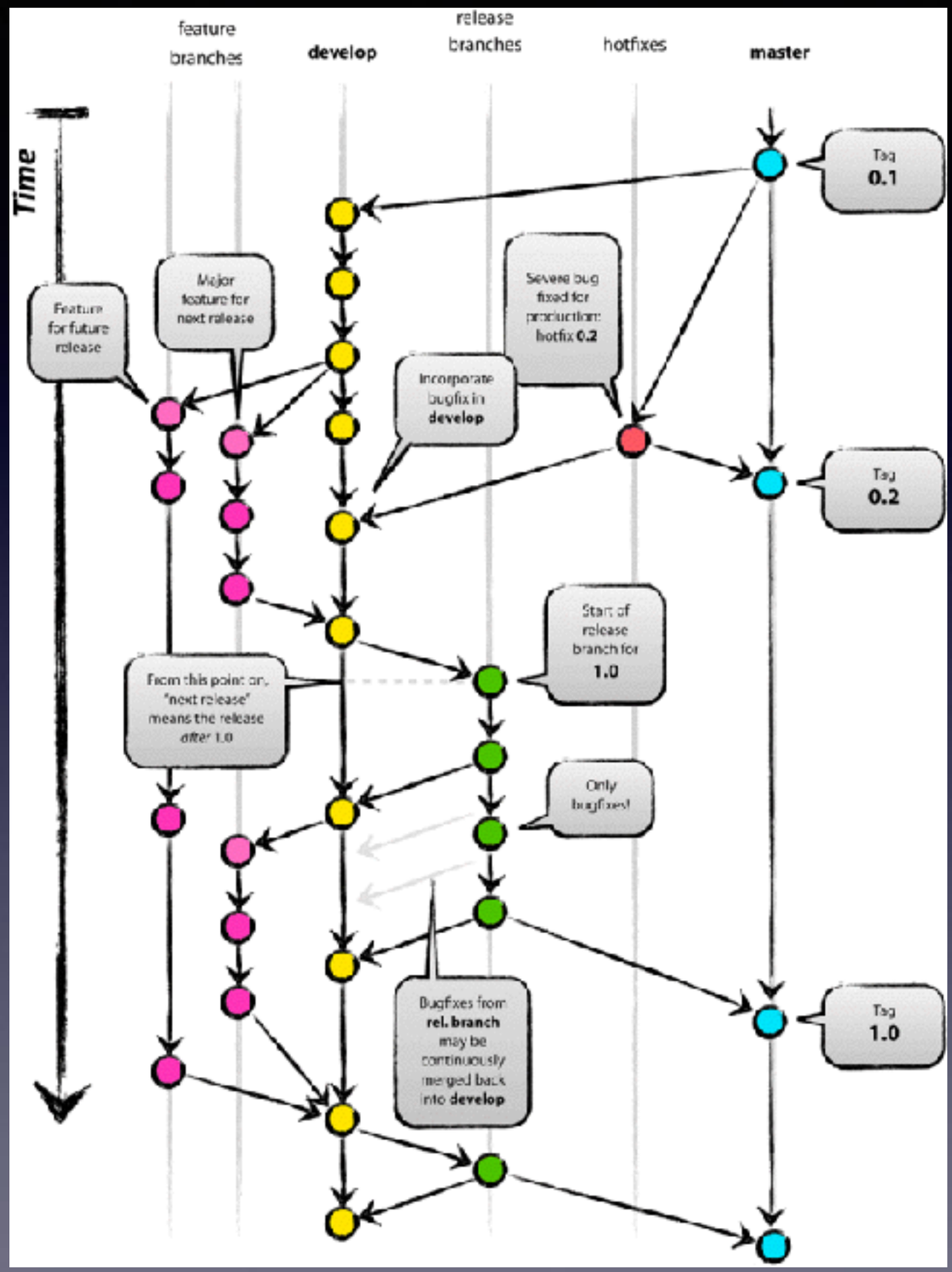
Master
只用于版本发布

Develop
主开发分支

Feature
用来发一个新功能

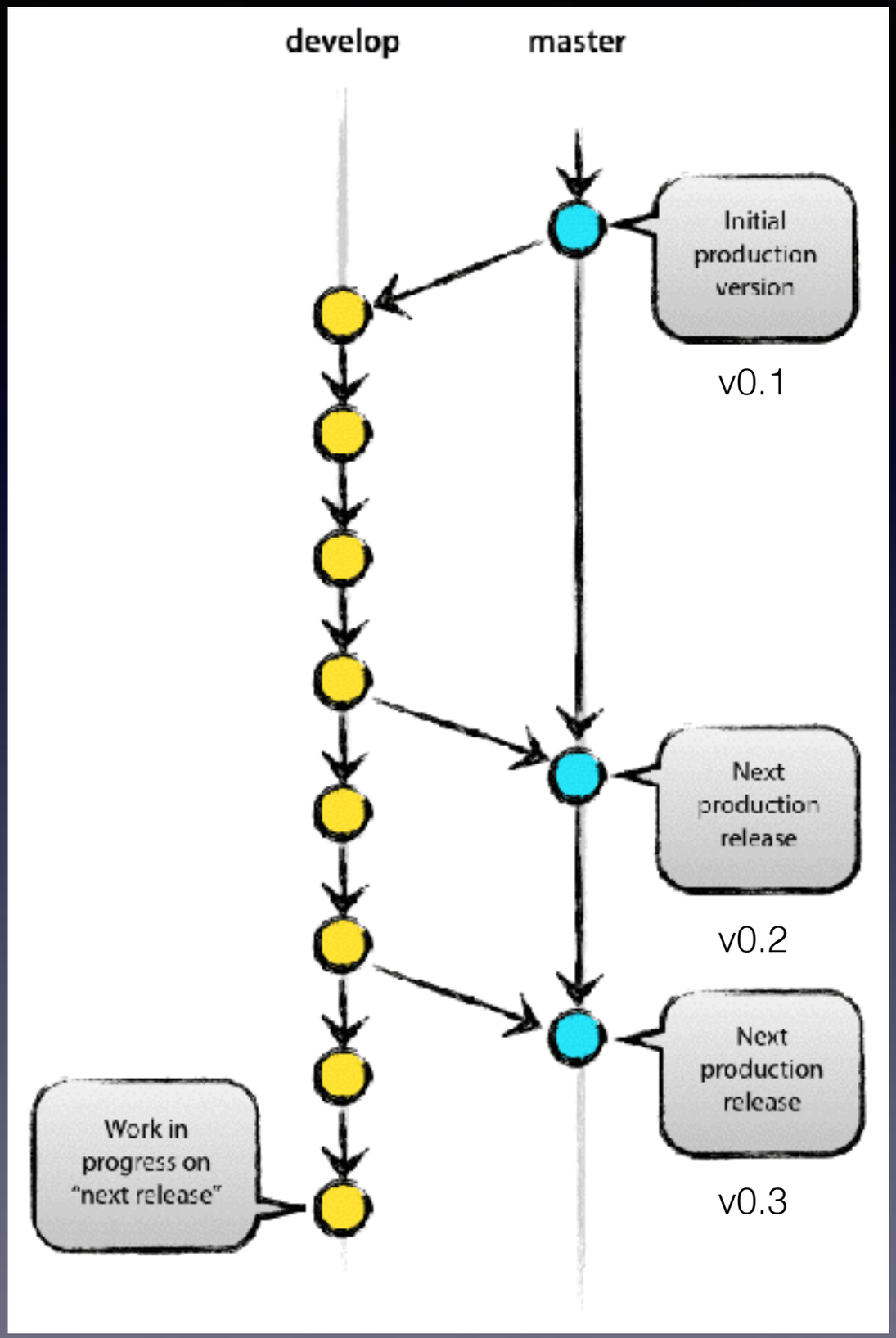
Release
用来发布 Release

Hotfix
用来修复线上 Bug 的



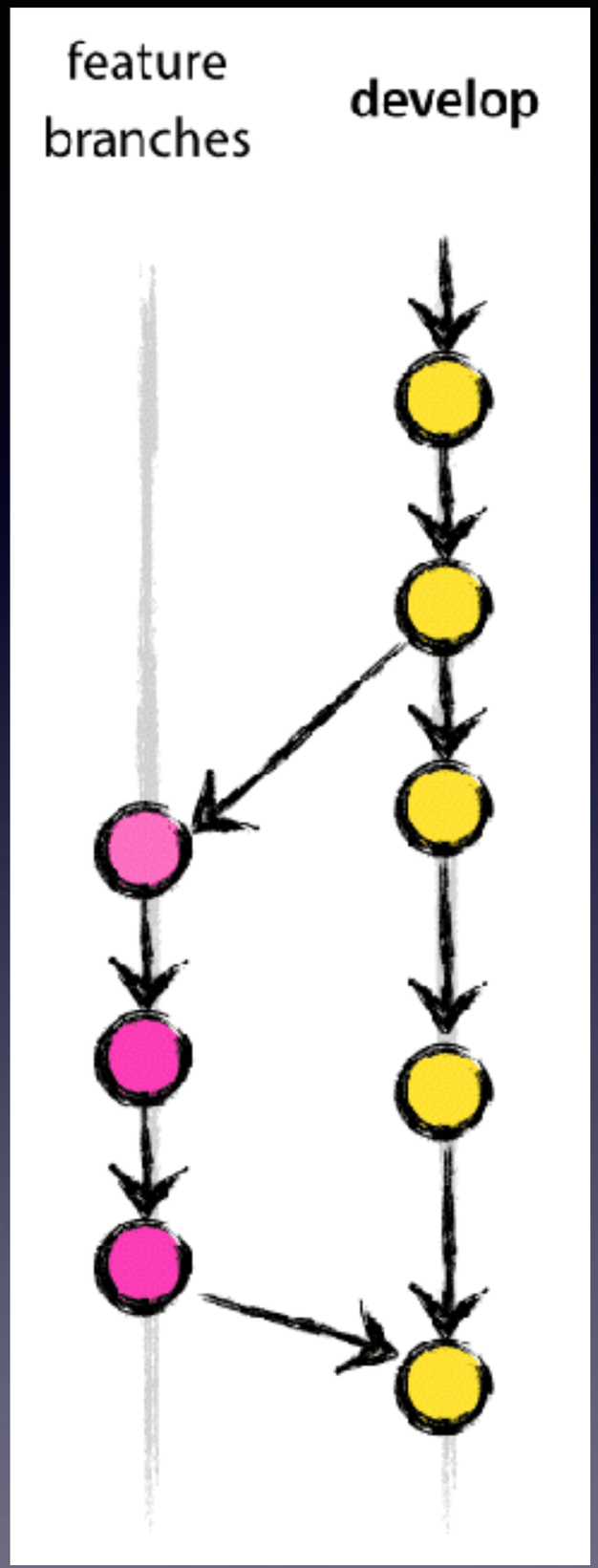
初始化主要分支

所有在 Master 分支上的
Commit 应该要有 Tag



Feature 分支

Feature 分支做完后，必须合并回 Develop 分支，合并完分支一般会删除这个 Feature 分支



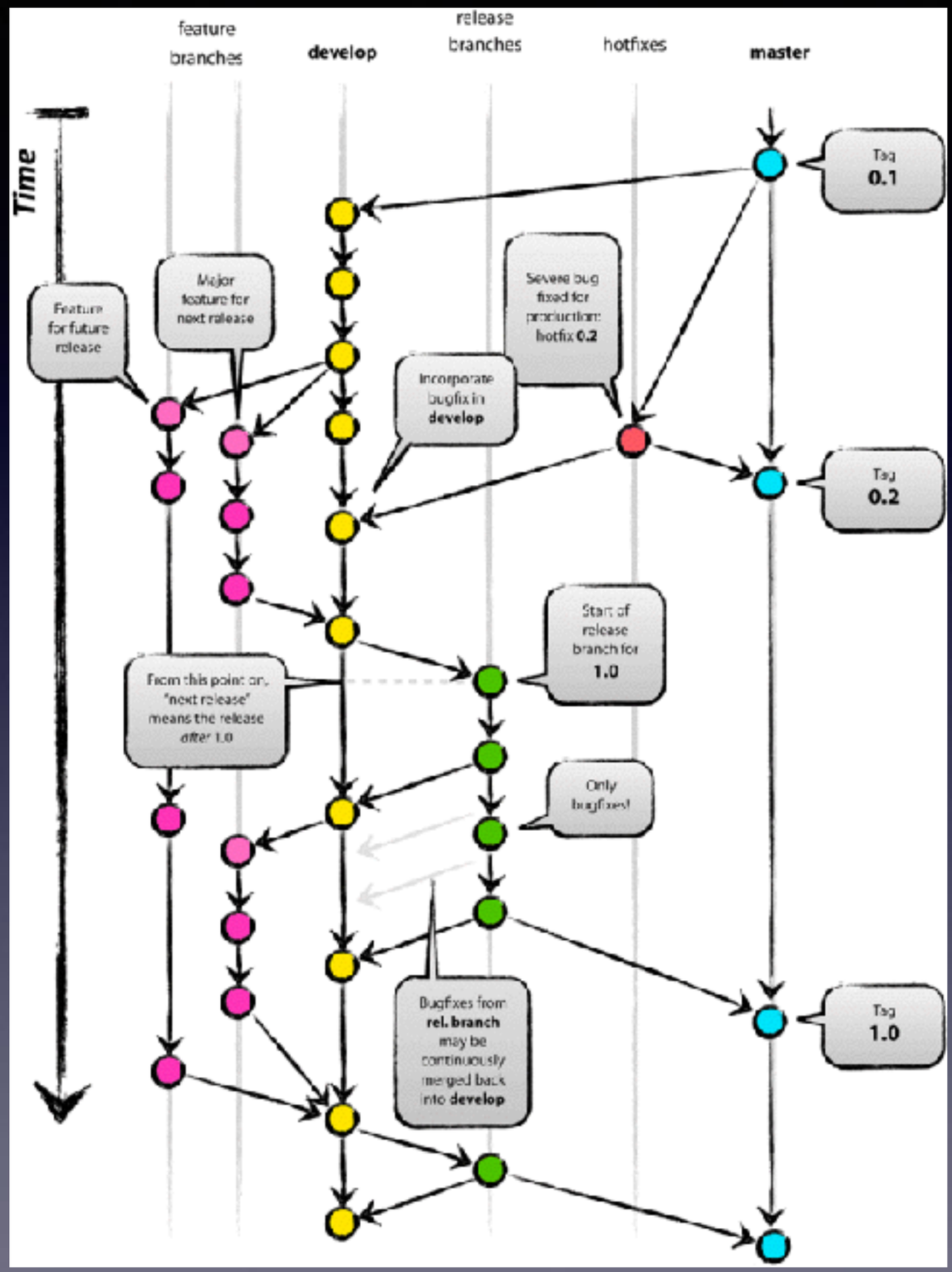
Release 分支

Release 分支基于 Develop 分支创建

可以在 Release 分支上测试，修改 Bugs 等

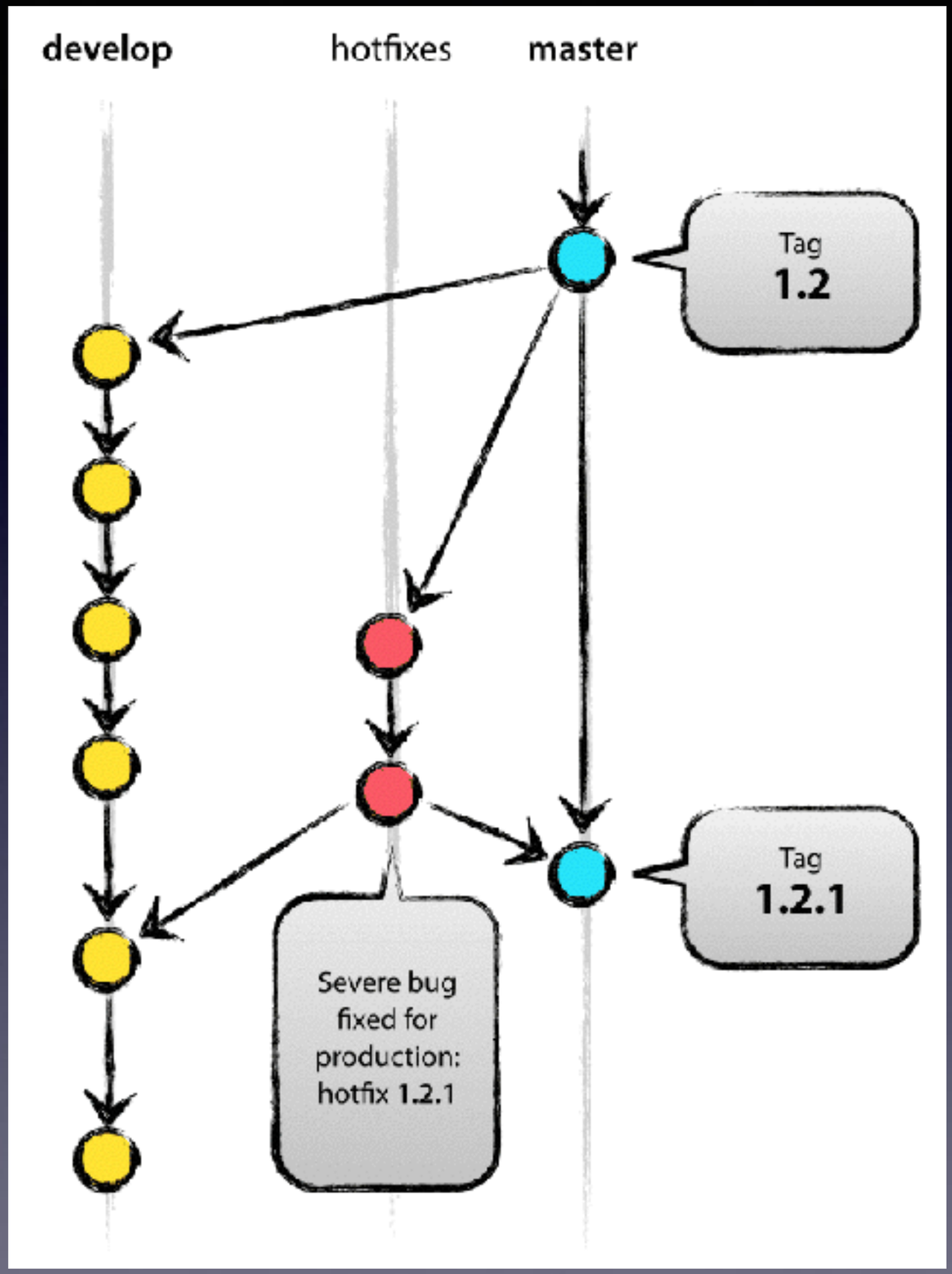
发布 Release 分支，就是把 Release 分支合并到 Master 和 Develop，同时需要给 Master 打上 Tag

完成后，需要删除 Release



Hotfix 维护分支

Hotfix 分支基于 Master 分支创建的，开发完成后需要合并回 Master 和 Develop 分支，同时 Master 要打上一个 tag



SmartGit 实战