MdEditor

Git分支管理及命名规范

时间 2020-10-11 14:15:06 叽歪小Q 语言: CN / TW / HK

主题: 掘金 稀土 Vue.js 微信小程序 Kotlin RxJava React Native Wireshark 敏捷开发 Bootstrap OKHttp 正则表达式 WebGL Webpack Docker MVVM

Git分支管理及命名规范

一、分支分类

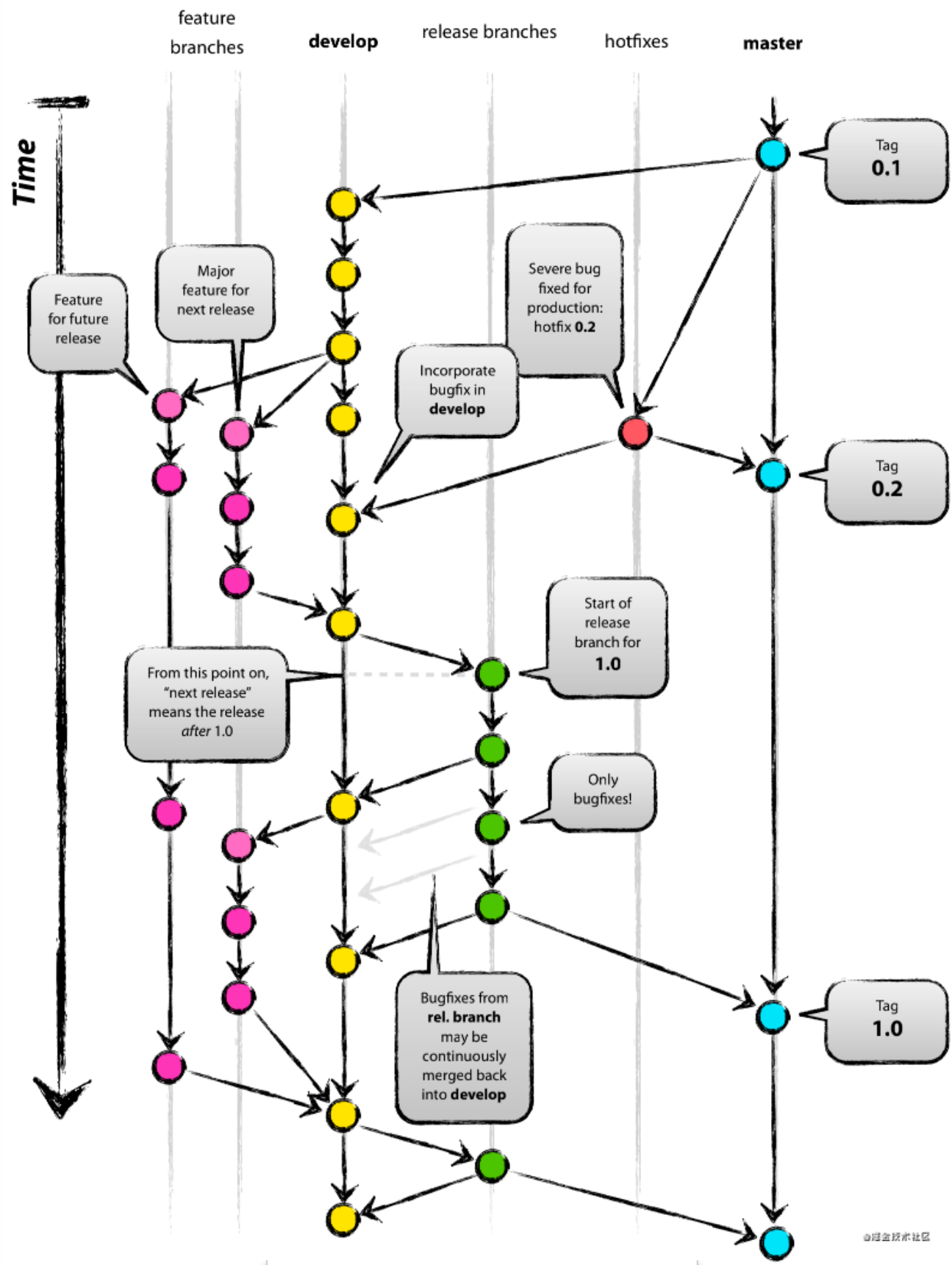
- **Git主分支（保留分支）：master、develop**

  主要分支：Master和Develop。前者用于正式发布，后者用于日常开发。

- **Git辅助分支（临时分支）：feature、release、fixbug**

  除了常设分支以外，还有一些临时性分支，用于应对一些特定目的的版本开发。临时性分支主要有三种：

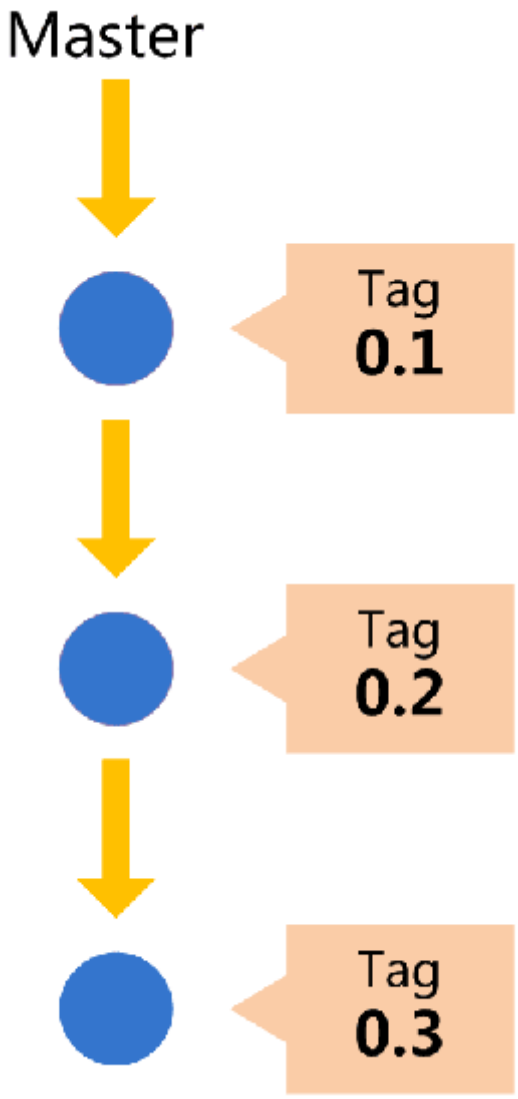
  > - 功能（feature）分支
  > - 预发布（release）分支
  > - 修补bug（fixbug）分支
  >
  > 这三种分支都属于临时性需要，使用完以后，应该删除，使得代码库的常设分支始终只有Master和Develop。



二、分支简介

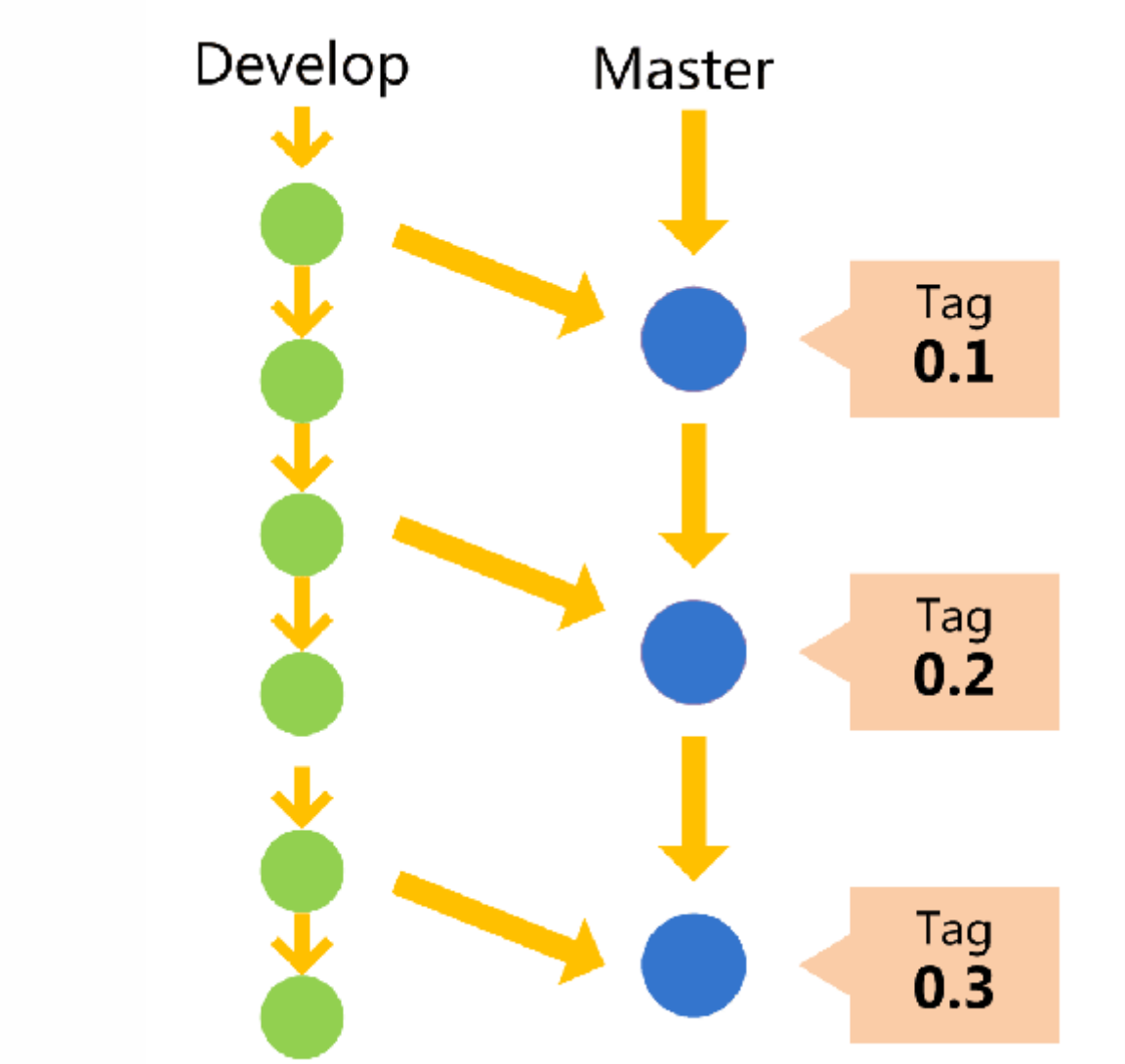
1、主分支 Master

代码库有且仅有一个主分支，所有提供给用户使用的正式版本，都在这个主分支上发布。Git默认主分支Master，它是自动建立的，版本库初始化以后，默认就是在主分支在进行开发。



2、开发分支 Develop

主分支只用来分布重大版本，日常开发应该在另一条分支上完成。我们把开发用的分支，叫做Develop。这个分支可以用来生成代码的最新隔夜版本（nightly）。如果想正式对外发布，就在Master分支上，对Develop分支进行"合并"（merge）。



- **创建分支**

Git创建Develop分支的命令：

```
git checkout -b develop master
复制代码
```

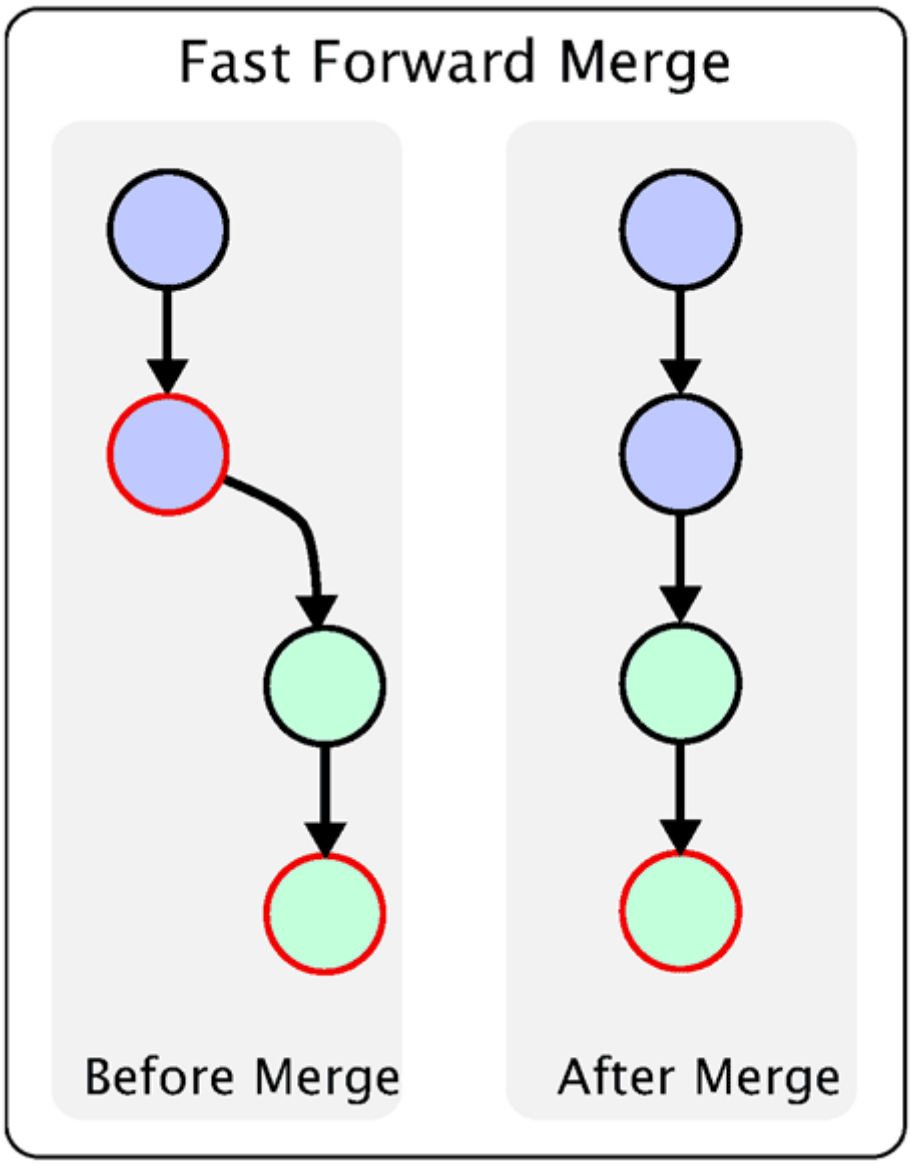
- **发布分支**

将Develop分支发布到Master分支的命令：

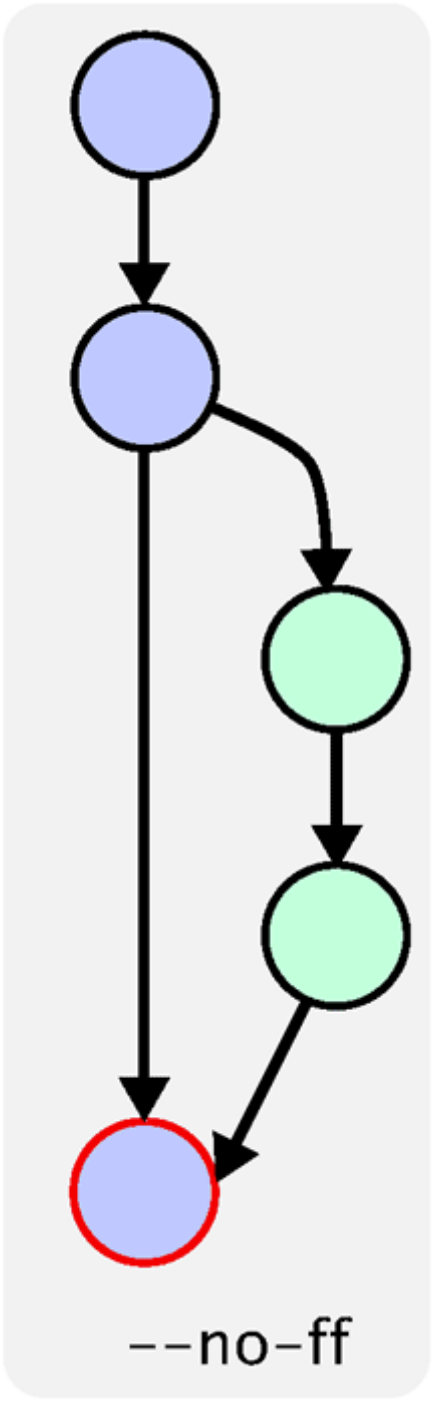
```
// 切换到 Master 分支
git checkout master

// 对 Develop 分支进行合并
git merge --no-ff develop
复制代码
```

> --no-ff参数是什么意思？默认情况下，Git执行"快进式合并"（fast-farward merge），会直接将Master分支指向Develop分支。

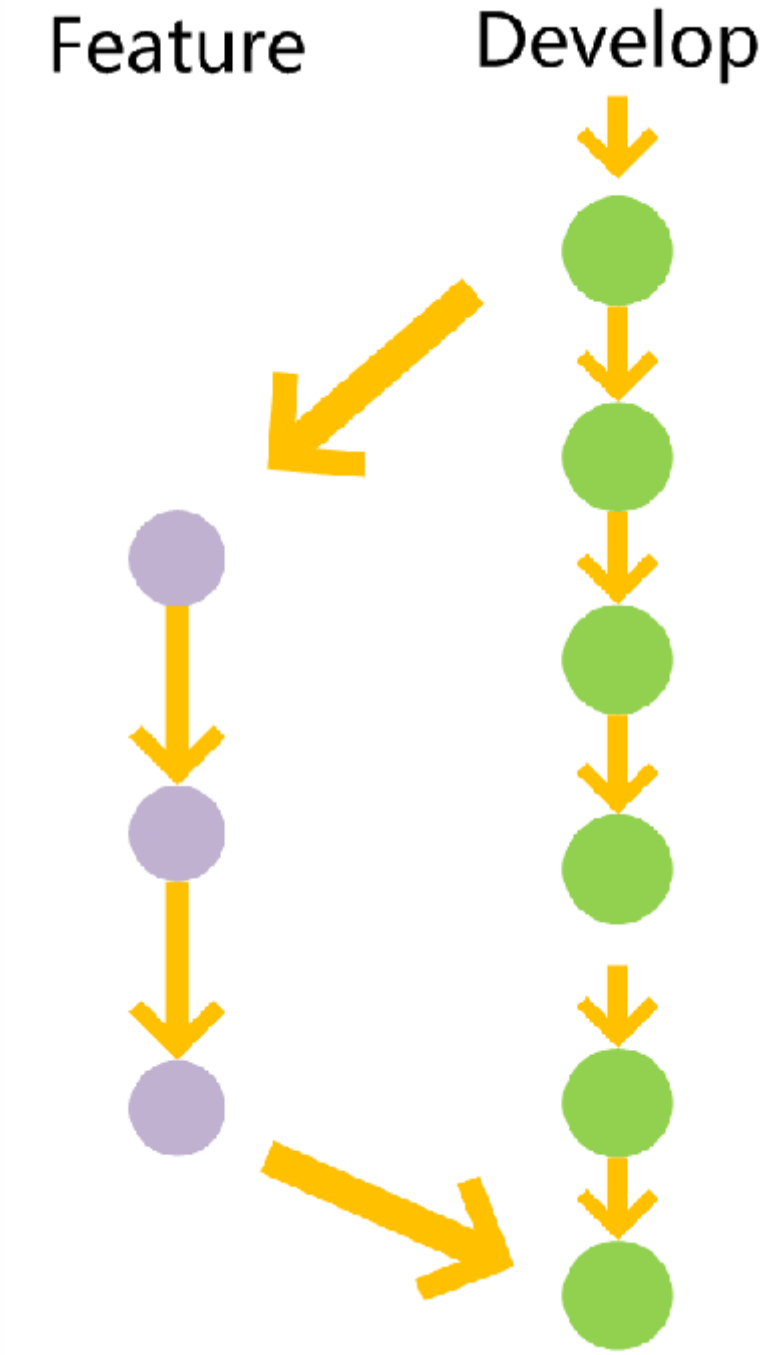


使用--no-ff参数后，会执行正常合并，在Master分支上生成一个新节点。为了保证版本演进的清晰，建议采用这种做法。



3、功能分支 feature

功能分支，是为了开发某种特定功能，从Develop分支上面分出来，**开发完成后，再并入Develop。** 功能分支的名字，可以采用feature-*的形式命名。



- **创建分支**

  创建一个功能分支的命令：

```
git checkout -b feature-xxx develop
复制代码
```

- **合并分支**

  开发完成后，将功能分支合并到develop分支的命令：

```
git checkout develop
```

```
git merge --no-ff feature-xxx
复制代码
```

- **删除分支**

删除feature分支的命令：

```
git branch -d feature-xxx
复制代码
```

4、预发布分支 release

预发布分支，是指发布正式版本之前（即合并到Master分支之前），我们可能需要有一个预发布的版本进行测试。

预发布分支是从Develop分支上面分出来的，**预发布结束以后，必须合并进Develop和Master分支**。它的命名，可以采用release-*的形式。

- **创建分支**

创建一个预发布分支的命令：

```
git checkout -b release-2.5 develop
复制代码
```

- **合并分支**

确认没有问题后，合并到master分支的命令：

```
git checkout master

git merge --no-ff release-2.5

# 对合并生成的新节点，做一个标签
git tag -a 2.5
复制代码
```

再合并到develop分支的命令：

```
git checkout develop

git merge --no-ff release-2.5
复制代码
```

- **删除分支**

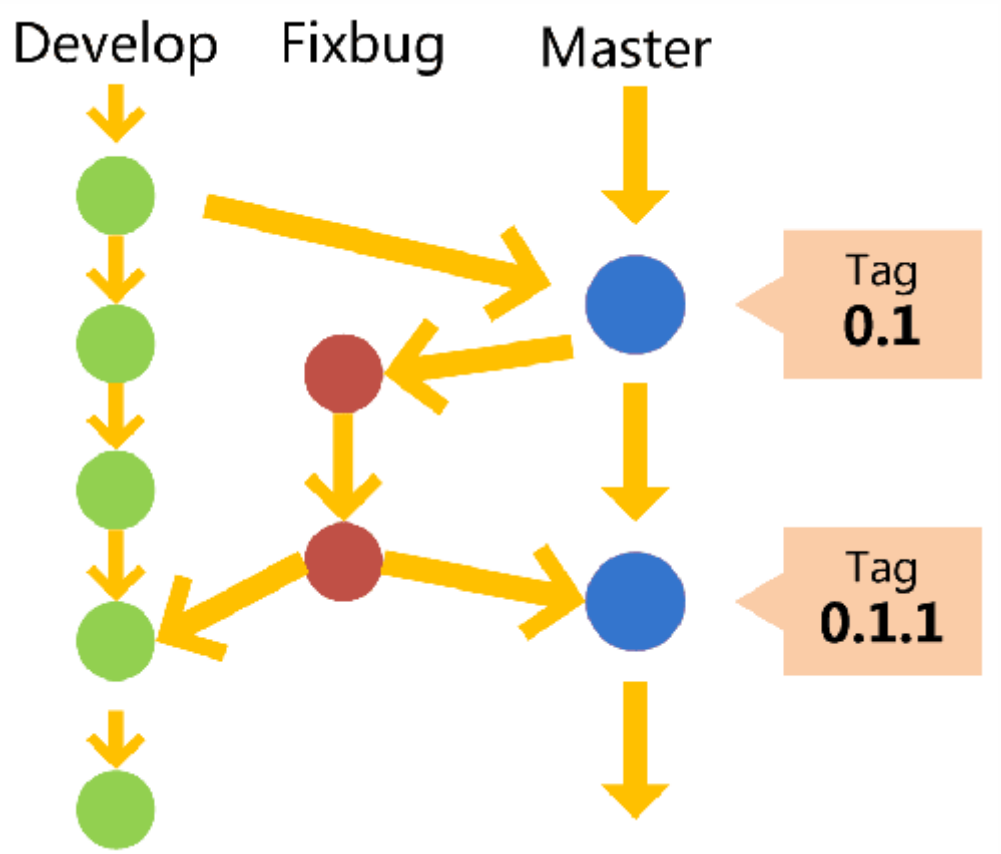
最后，删除预发布分支的命令：

```
git branch -d release-2.5
复制代码
```

5、修补bug分支 fixbug

修补bug分支，软件正式发布以后，难免会出现bug，这时就需要创建一个分支，进行bug修补。

修补bug分支是从Master分支上面分出来的。**修补结束以后，再合并进Master和Develop分支**。它的命名，可以采用fixbug-*的形式。



- **创建分支**

创建一个修补bug分支的命令：

```
git checkout -b fixbug-0.1 master
复制代码
```

- **合并分支**

修补结束后，合并到master分支的命令：

```
git checkout master

git merge --no-ff fixbug-0.1

git tag -a 0.1
复制代码
```

再合并到develop分支的命令：

```
git checkout develop

git merge --no-ff fixbug-0.1
复制代码
```

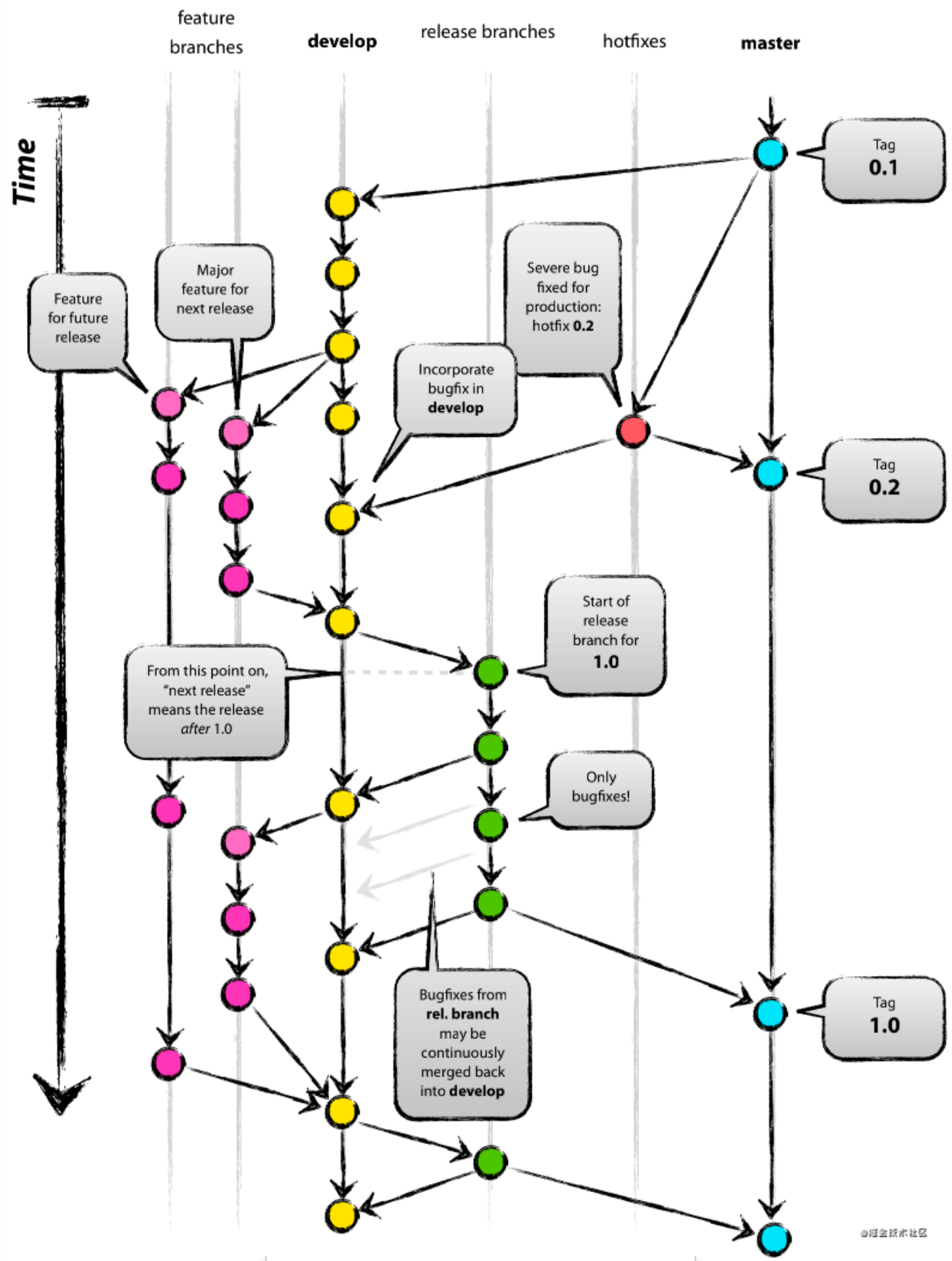
- **删除分支**

最后，删除修补bug分支的命令：

```
git branch -d fixbug-0.1
复制代码
```

三、总结归纳

最后回顾开头的大图，掌握两大主分支、三个辅助分支的生命周期，起于哪个分支始于哪个分支。



参考资料：

Git分支管理策略

Git 分支管理是一门艺术

分享到: