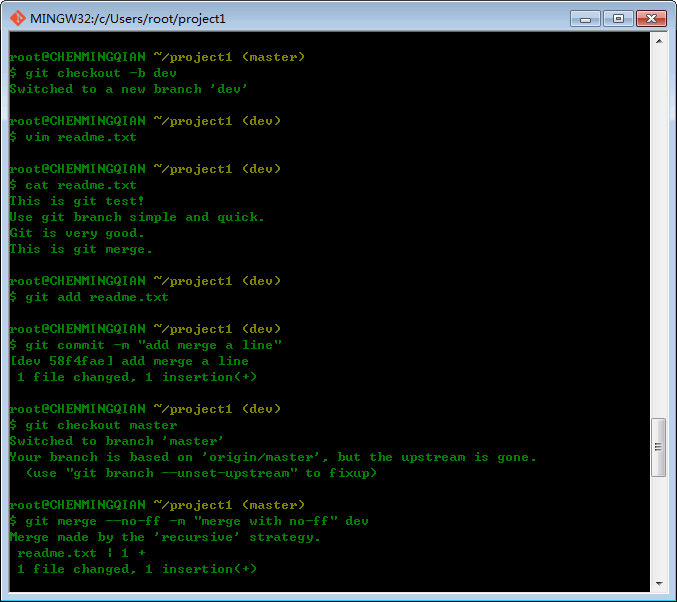
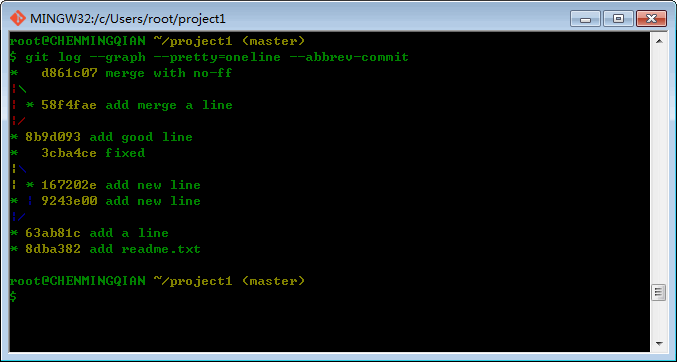
7.合并分支（普通合并）

分支合并分为快速合并与普通合并两种模式，普通合并，合并后的历史有分支记录，能看出来曾经做过合并，而快速合并就看不出来曾经做过合并。下面我们来演示一下普通合并，

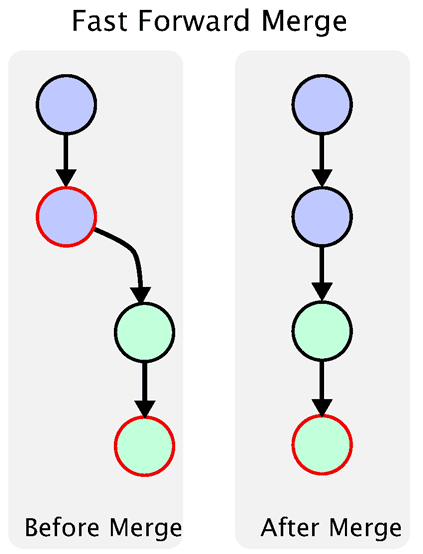


大家可以看到我们这次合并用的普通模式合并，--no-ff参数表示禁用快速合并。下面我们用git log命令查看一下合并历史：

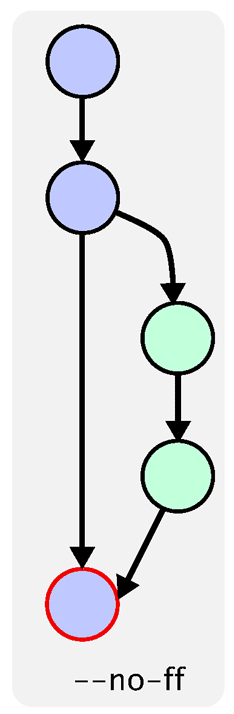


注，合并分支时，加上*--no-ff*参数就可以用普通模式合并，合并后的历史有分支记录，能看出来曾经做过合并，而fast forward合并就看不出来曾经做过合并。如下图：

1).快速合并



2).普通合并



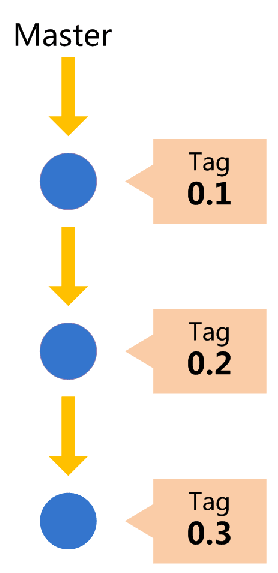
8.分支管理策略

下面我们来说一下一般企业中开发一个项目的分支策略：

* 主分支 master
* 开发分支 develop
* 功能分支 feature
* 预发布分支&#160; release
* bug 分支 fixbug
* 其它分支 other

1).主分支 master

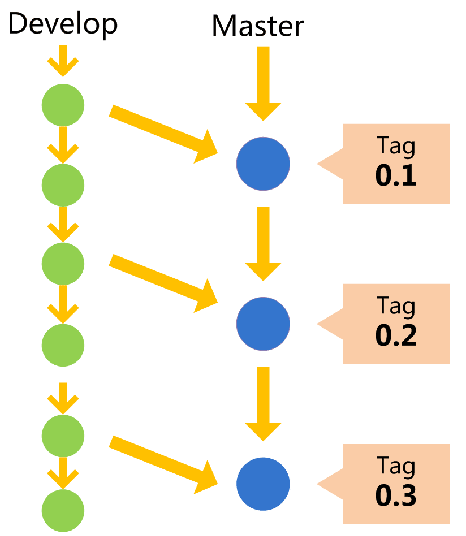
代码库应该有一个、且仅有一个主分支。所有提供给用户使用的正式版本，都在这个主分支上发布。



Git主分支的名字，默认叫做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

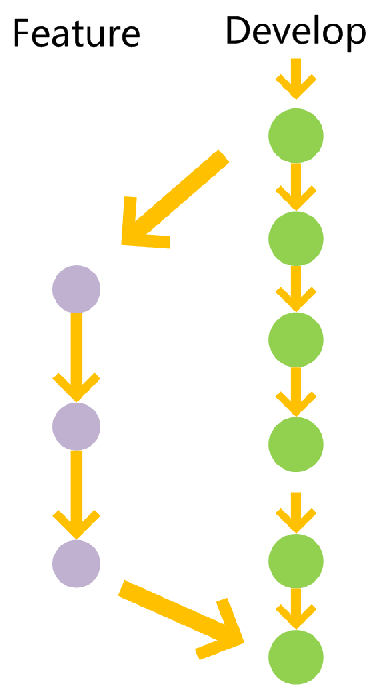
主分支只用来分布重大版本，日常开发应该在另一条分支上完成。我们把开发用的分支，叫做Develop。



这个分支可以用来生成代码的最新代码版本。如果想正式对外发布，就在Master分支上，对Develop分支进行"合并"（merge）。

3).功能分支 feature

功能分支，它是为了开发某种特定功能，从Develop分支上面分出来的。开发完成后，要再并入Develop。



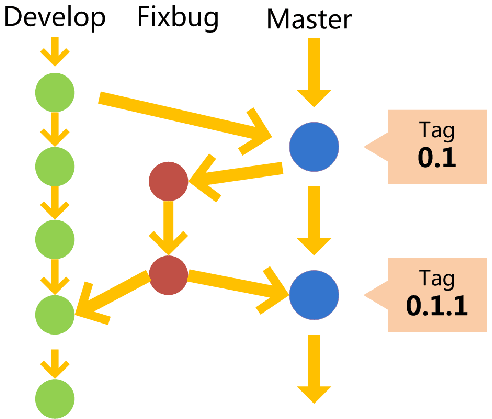
功能分支的名字，可以采用feature-\*的形式命名。

4).预发布分支&#160; release

预发布分支，它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。

5).bug 分支 fixbug

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。



6).其它分支 other

还有就是其它分支了，大家可以根据需要创建即可……

[linux](http://www.linuxidc.com/)