git使用流程和规范

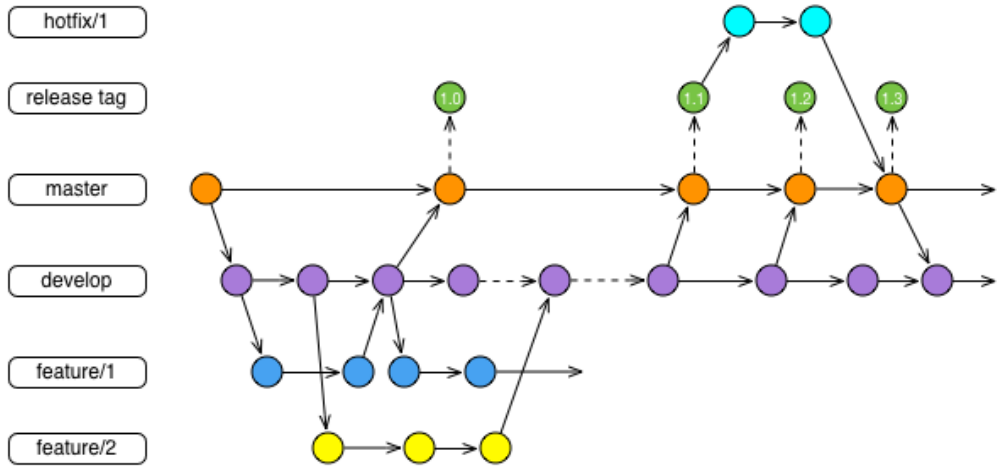
修改日期	修改者	修改描述
2018年1月9日		初始版本

目标

1. 最大限度提升团队的长期工作效率。
2. 灵活应对产品需求变化和并行开发。
3. 方便测试部署自动化打包。

设计

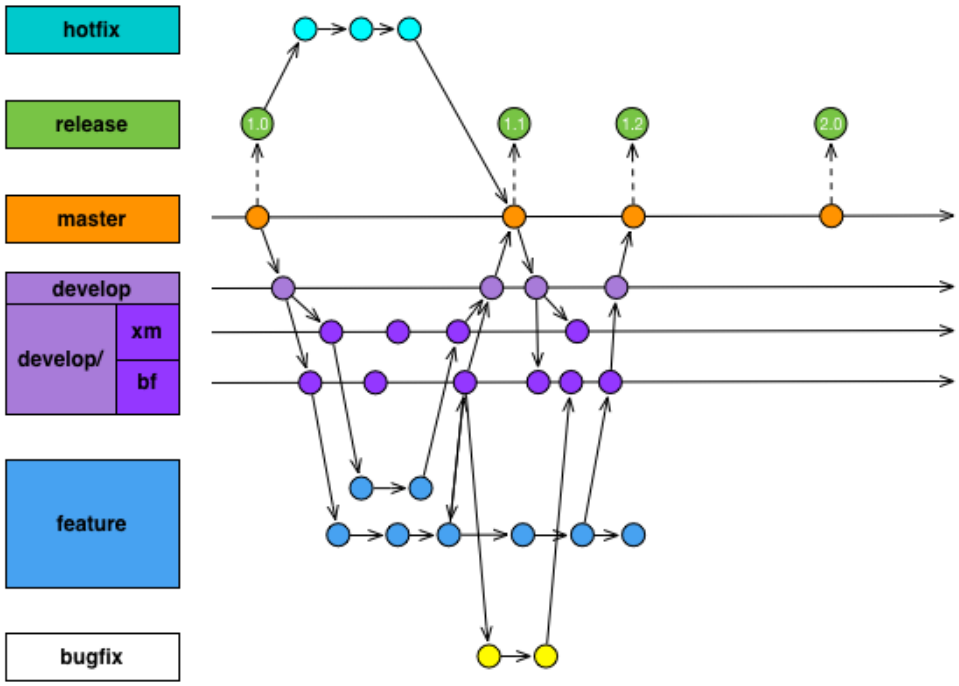
一、git-flow



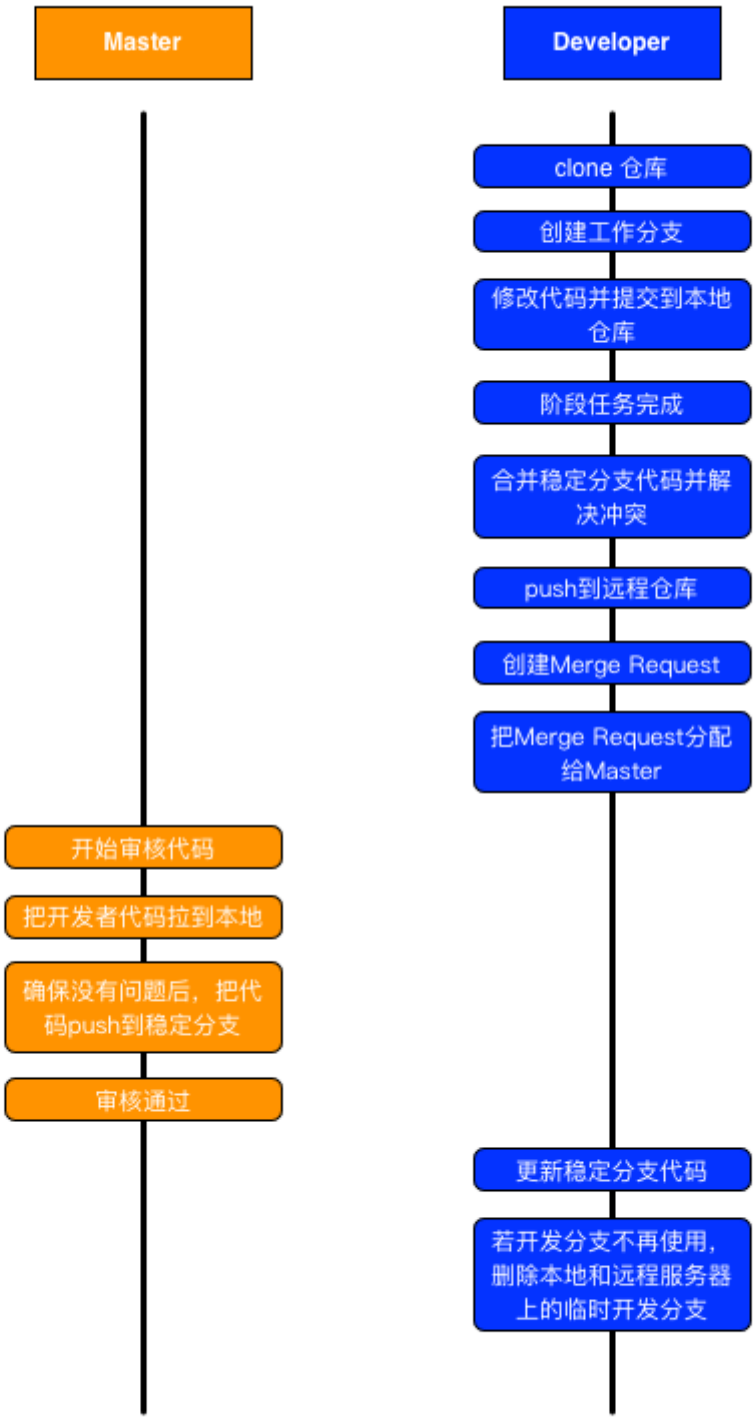
说明：

> - master: 主分支，确保从master获得的都是处于可以发布状态的代码。
> - develop: 开发分支，确保从develop总能获得最新开发进展的代码，测试组能从此分支随时获取可以打测试包的代码。
> - feature/xxx: 功能开发分支，以独立的功能模块为单位建立新分支，确保开发的功能模块的独立性和完整性。从develop检出，并最终合并到develop分支去。特点是生命周期短，一个功能模块完成即可删除该分支。
> - release/xxx: 由项目负责人(master)根据发版版本号打的tag，确保根据版本号可以追溯所有已发版本的代码。
> - hotfix/xxx: 热修复分支，线上需要紧急修复的bug，从release tag分支检出。修复bug完成后，合并分支到master，再从master打紧急修复版本的tag，并把代码合并到develop。hotfix的生命周期短，bug修复完成后即可删除该分支。

二、git-model



三、work-flow



说明：

1. Developer检出仓库代码。示例：

```
$ git clone git@gitlab.szy.net:szy-ios/ztjy.git
```

2. Developer根据git-model模型，从对应的开发分支上检出，创建工作分支。示例：

```
$ git checkout develop/xm
$ git checkout -b feature/user-login
```

3. Developer修改代码后，提交到本地仓库。示例：

```
$ git add files
$ git commit -m "登录模块-接口联调完成"
```

4. Developer阶段性任务完成后，合并将要发起 Merge Request的目标分支代码，并push到远程仓库。示例：

```
$ git pull
$ git merge develop/xm feature/user-login
$ git push origin feature/user-login
```

5. Developer在gitlab网站上创建Merge Request，并把Merge Request分配给项目负责人。

6. 项目负责人审核Developer提交的代码，并把代码拉到本地，确保不会影响团队工作后（比如编译不过或者进入App后就有必先的崩溃等），然后把代码push到稳定分支。审核通过。

7. 若开发分支功能已完成，即分支不再使用（也可以在每个项目结束时统一整理），删除远程服务器和本地上的开发分支。示例：

```
$ git push origin :feature/user-login
$ git branch -d feature/user-login
```

规范

1. 分支目录名有develop, feature, bugfix, release 及 hotfix。确保根据目录名，立马就能知道这个分支的类型。
2. 分支名，用英文描述清楚改分支的模块和功能，英文单词之间用短横线"-"连接。除非业内通用的英文简称，否则不要自己创造英文简称。例如`feature/shop-address-manager`可以用来表示开发商城地址管理的分支。
3. 代码提交(commit)时，用中文或者英文写上完整扼要的提交信息，要让其他研发同学和测试同学能看懂修改了什么模块的什么功能。**禁止**使用诸如`"bug fix"`这种毫无信息量的提交信息。
4. 如果提交的修改是修复了某个bug，可以把bug平台对应的id写进提交信息，方便测试同学跟踪。例如`git commit -m "[bug25002]修复家园联系册新建练习册时，时间控件不能选择时间的问题"`。