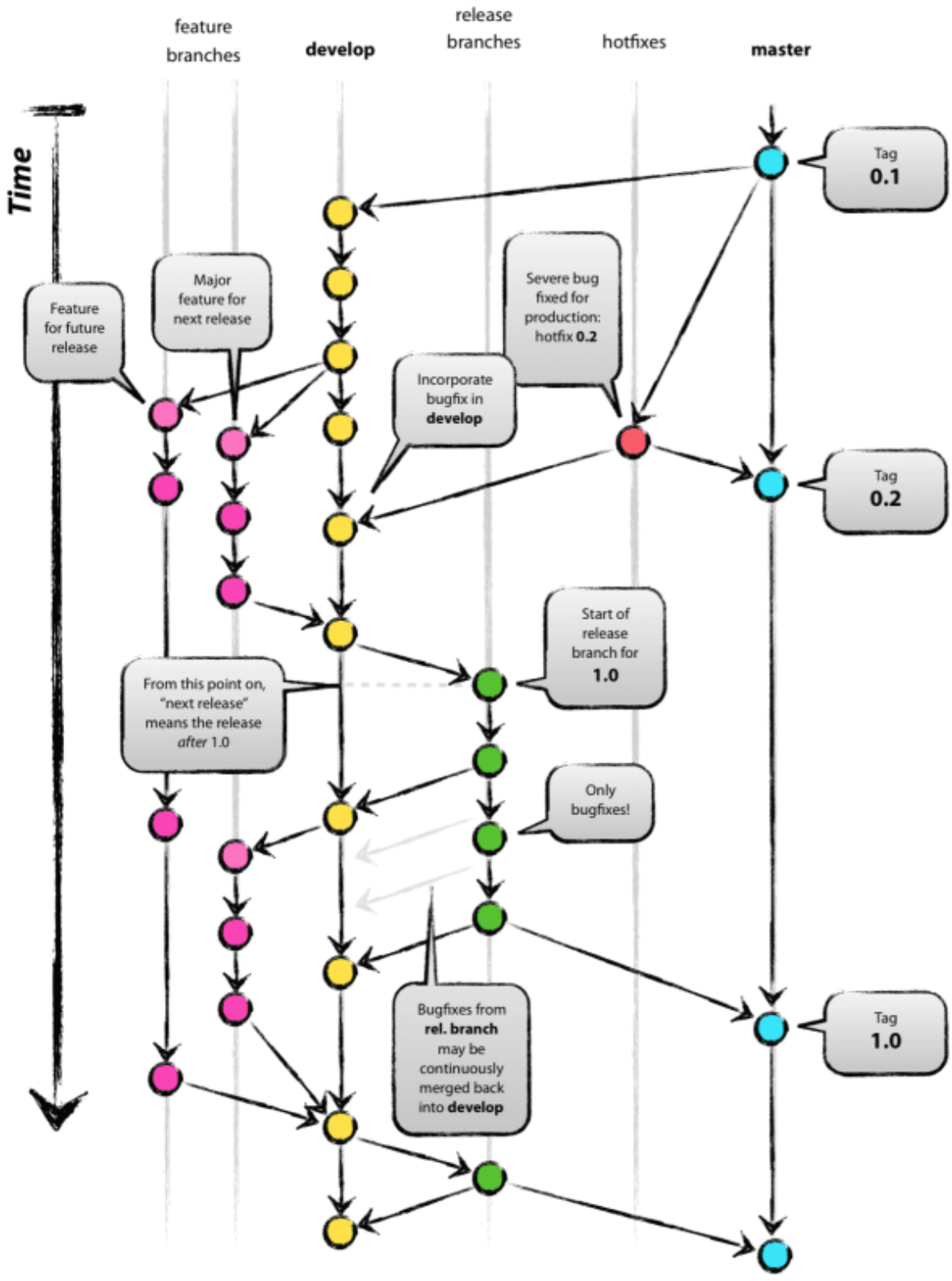
如何正确使用Git

参考URL：https://www.cnblogs.com/yhaing/p/8473746.html



其中涉及到的主要分支类型有：

- master分支，即主分支。任何项目都必须有个这个分支。对项目进行tag或发布版本等操作，都必须在该分支上进行。
- develop分支，即开发分支，从master分支上检出。团队成员一般不会直接更改该分支，而是分别从该分支检出自己的feature分支，开发完成后将feature分支上的改动merge回develop分支。同时release分支由此分支检出。

- release分支，即发布分支，从develop分支上检出。该分支用作发版前的测试，可进行简单的bug修复。如果bug修复比较复杂，可merge回develop分支后由其他分支进行bug修复。此分支测试完成后，需要同时merge到master和develop分支上。
- feature分支，即功能分支，从develop分支上检出。团队成员中每个人都维护一个自己的feature分支，并进行开发工作，开发完成后将此分支merge回develop分支。此分支一般用来开发新功能或进行项目维护等。
- fix分支，即补丁分支，由develop分支检出，用作bug修复，bug修复完成需merge回develop分支，并将其删除。所以该分支属于临时性分支。
- hotfix分支，即热补丁分支。和fix分支的区别在于，该分支由master分支检出，进行线上版本的bug修复，修复完成后merge回master分支，并merge到develop分支上，merge完成后也可以将其删除，也属于临时性分支。

主要的管理命令及用法：

1. 克隆git仓库

   作为普通开发人员只需要克隆develop分支，然后在develop分支上生成自己的分支即可

```
git clone XXX.git #克隆master分支上的内容
git clone -b xxx(分支名) xxx.git #克隆指定分支上的内容
```

2. 线上版本的代码（master分支）出现了紧急bug，需要修复。这里用到了hotfix分支。

```
git checkout master    # 切换回master分支
git checkout -b hotfix master    # 新建hotfix分支，并切换到该分支
# 做一些bug修复工作
git checkout master    # 切换回master分支
git merge --no-ff hotfix    # 合并hotfix分支，此时bug已被修复（无冲突）
git tag v0.2    # 新建tag v0.2
git push origin master    # 推送master分支代码到远端
git push origin --tags    # 推送tag到远端
```

3. develop分支上的功能开发完成了，需要进行测试和提交。这里用到了release分支。

```
git checkout develop    # 切换回develop分支
git checkout -b release01 develop    # 新建release分支，并切换到该分支

# 做一些测试、bug修复等工作

git checkout develop    # 切换回develop分支
git merge --no-ff release01  # 合并release01分支到develop分支（无冲突）
git push origin develop    # 推送develop分支到远端

git checkout master    # 切换回master分支
git merge --no-ff release01   # 合并release01分支到master分支（无冲突）
git tag v0.3    # 新建tag v0.3
git push origin master    # 推送master分支代码到远端
git push origin --tags    # 推送tag到远端
```

4. 合并分支出现冲突的情况

```
git checkout -b feature-zz develop    # 从develop分支新建并检出feature分支
# 这里可以进行一些功能开发，并不断的add和commit
git checkout develop    # 切换回develop分支
git pull origin develop    # 更新远端代码，看develop分支是否有更新（有更新）
git checkout feature-hu    # 切换回feature分支
git rebase develop    # 合并develop分支到feature分支，并解决冲突（有冲突）
# 这里需要进行冲突解决
git add .    # 解决完冲突之后执行add操作
git rebase --continue    # 继续刚才的rebase操作
git checkout develop    # 切换回develop分支
git merge --no-ff feature-zz    # 合并feature分支到develop分支（无冲突）
git push origin develop   # 推送develop分支到远端
```

5. 撤销已有的提交

```
#先切换到指定的分支
git log #查找需要撤销的提交的id号
        #commit aa909cff2239536df14820fe086d96305b24e9f1
git reset -soft id(commit的id号)
```