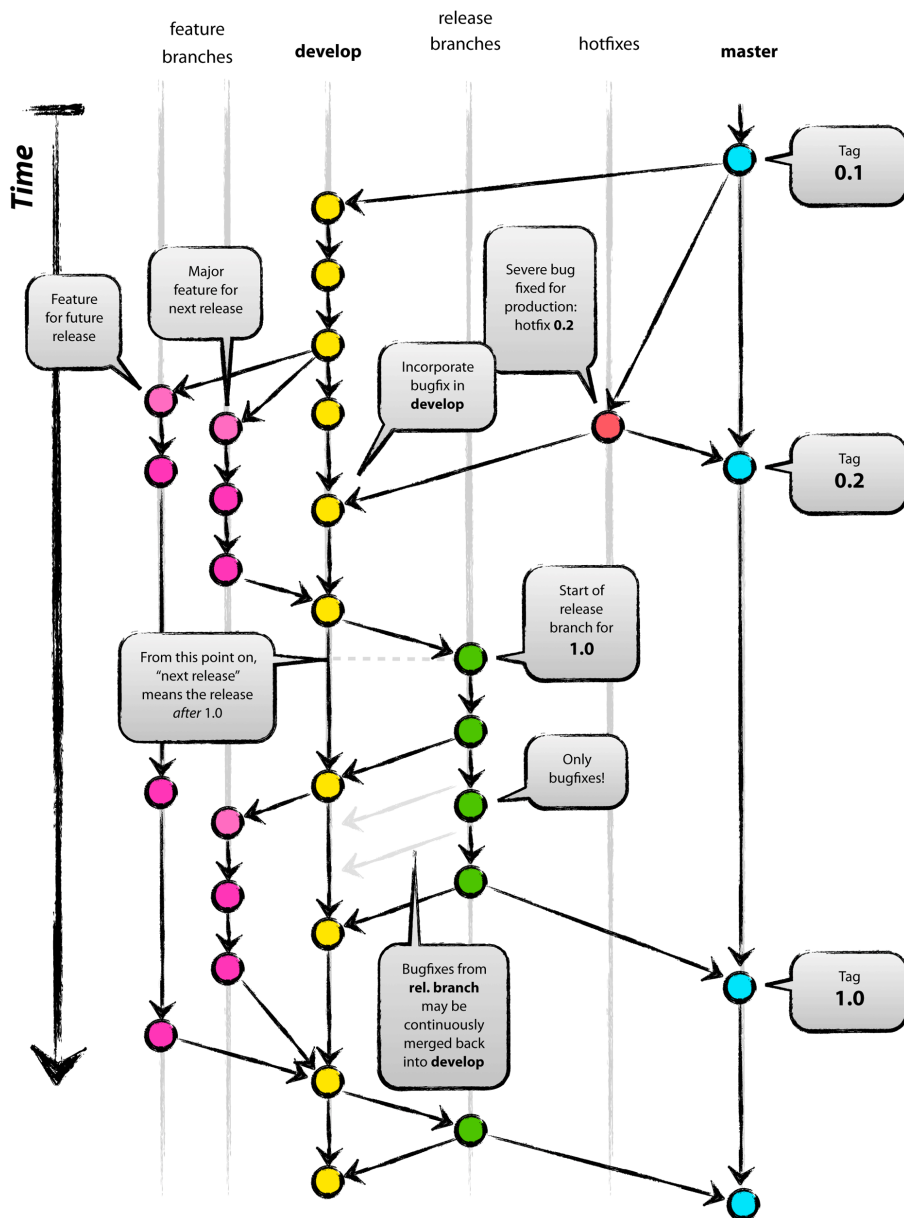


Git-Flow

Git Flow的流程图



上面的图你理解不了？没关系，这张图应该左转90度，大家应该就很用以理解了。

Git Flow常用的分支

- Master 分支

这个分支是最近发布到生产环境的代码，最近发布的Release，这个分支只能从其他分支合并，不能在这个分支直接修改

- Develop 分支

这个分支是我们的主开发分支，包含所有要发布到下一个Release的代码，这个主要合并与其他分支，比如Feature分支

- Feature 分支

这个分支主要是用来开发一个新的功能，一旦开发完成，我们合并回Develop分支进入下一个Release

- Release分支

当你需要一个发布一个新Release的时候，我们基于Develop分支创建一个Release分支，完成Release后，我们合并到Master和Develop分支

- Hotfix分支

当我们在Production发现新的Bug时候，我们需要创建一个Hotfix, 完成Hotfix后，我们合并回Master和Develop分支，所以Hotfix的改动会进入下一个Release

Git Flow如何工作

初始分支

所有在Master分支上的Commit应该Tag



Feature 分支

分支名 feature/*

Feature分支做完后，必须合并回Develop分支, 合并完分支后一般会删点这个Feature分支，但是我们也可以保留

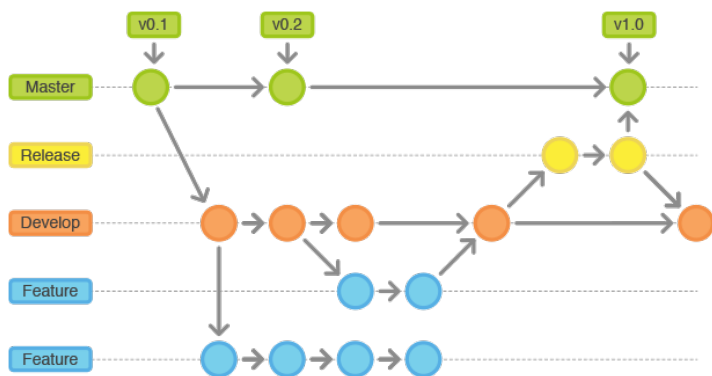


Release分支

分支名 release/*

Release分支基于Develop分支创建，打完Release分支之后，我们可以在这个Release分支上测试，修改Bug等。同时，其它开发人员可以基于开发新的Feature (记住：一旦打了Release分支之后不要从Develop分支上合并新的改动到Release分支)

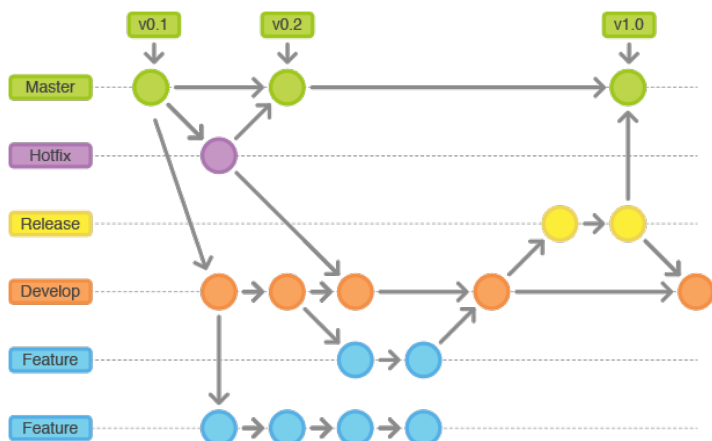
发布Release分支时，合并Release到Master和Develop， 同时在Master分支上打个Tag记住Release版本号，然后可以删除Release分支了。



维护分支 Hotfix

分支名 hotfix/*

hotfix分支基于Master分支创建，开发完后需要合并回Master和Develop分支，同时在Master上打一个tag



Git Flow代码示例

a. 创建develop分支

```
1 git branch develop
2 git push -u origin develop
3
```

b. 开始新Feature开发

```
1 git checkout -b some-feature develop
2 # Optionally, push branch to origin:
3 git push -u origin some-feature
4
5 # 做一些改动
6
7 git status
8 git add some-file
9 git commit
10
```

c. 完成Feature

```
1 git pull origin develop
2 git checkout develop
3 git merge --no-ff some-feature
```

```
4 git push origin develop
5
6 git branch -d some-feature
7
8 # If you pushed branch to origin:
9 git push origin --delete some-feature
10
```

d. 开始Release

```
1 git checkout -b release-0.1.0 develop
2
3 # Optional: Bump version number, commit
4 # Prepare release, commit
5
```

e. 完成Release

```
1 git checkout master
2 git merge --no-ff release-0.1.0
3 git push
4
5 git checkout develop
6 git merge --no-ff release-0.1.0
7 git push
8
9 git branch -d release-0.1.0
10
11 # If you pushed branch to origin:
12 git push origin --delete release-0.1.0
13
14
15 git tag -a v0.1.0 master
16 git push --tags
17
```

f. 开始Hotfix

```
1 git checkout -b hotfix-0.1.1 master
2
```

g. 完成Hotfix

```
1 git checkout master
2 git merge --no-ff hotfix-0.1.1
3 git push
4
5
6 git checkout develop
7 git merge --no-ff hotfix-0.1.1
8 git push
9
10 git branch -d hotfix-0.1.1
11
12 git tag -a v0.1.1 master
13 git push --tags
14
```

