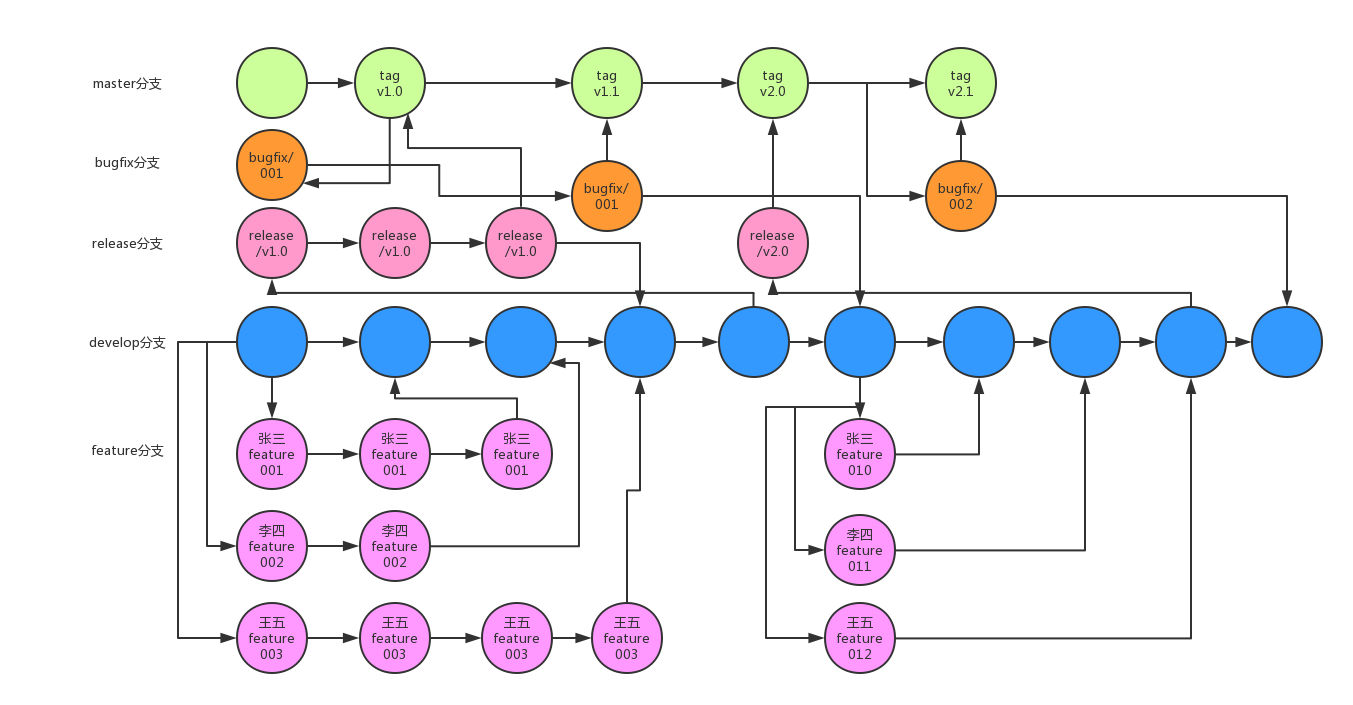
1、Git工作流

初步定的是基于GitFlow工作流来做



（1）feature分支

进入一个版本的开发之后，每个人自己拉对应的feature分支。feature分支的拉取，按照一个子系统一个feature分支；一个模块一个feature分支；一个需求一个feature分支；一个功能一个feature分支。不是固定死的，是根据当时的人力的任务分配情况，来定，怎么划分这个feature分支。

比如说我们现在，假设电商v1.0版本要做15个子系统，1+4的小team，一个架构师带4个初中级工程师。平均每个人分配到的是3个子系统。每个子系统拉一个feature分支，在那个子系统上，就专门开发需要的代码就可以了。

就是在上一讲，说工程初始化的时候，负责工程搭建的同学，就将所有系统中需要的接口全部定义出来

这样的话，每个人拉出来的featiure分支，自己仅仅写这个feature中需要的代码，但是如果你的feature中依赖了别人的子系统feature中的接口，没关系，一开始，所有的接口都定义好了。大家都面向接口开发就可以了。写单元测试的时候，对别人的接口实现，采取mock的方式来做。

每个人拉自己的feature分支，然后写自己的feature里的代码，对依赖别人的地方，面向接口编程

feature的命名规范：feature/order-system-v1.0

（2）develop分支

在上一讲，讲到代码初始化好之后，实际上代码是在master分支去初始化的。就直接从master分支拉一个develop分支出来，做为统一的代码集成的分支。

每个人的feature分支开发好之火，就每个人依次将自己的feature分支的代码，合并到develop分支，进行代码集成。所有人的代码，就在develop分支上，完成了集成，develop分支，就包含了系统这个版本需要的所有代码。

就可以基于develop分支的代码，去集成测试环境来部署，然后进行集成测试

集成测试的过程中，肯定是会不断的发现一些bug的，如何来修复bug呢？在什么分支上来修复bug？

一般来说，在集成测试环境，如果发现了这个bug的话，一般来说，对应有bug的同学，需要在自己的本地来复现这个bug

这里，推荐说直接基于develop分支的代码来复现和修复bug，因为feature分支的代码是不完整的，可能是没法复现bug的，所以是不能通过feature分支的代码来复现和修复bug的。

负责修复bug的同学，将develop分支的代码拉到自己本地，本地跑起来，基于开发环境的基础设施，应该在自己本地是可以跑起来的。复现bug，追查bug产生的原因，然后可以直接在develop分支上来修复这个bug。

push develop分支的修改到GitLab上，然后所有人就基于最新的develop分支继续进行测试

直到develop分支测试，感觉都保持稳定了，已经没有什么bug了，系统整体可以跑通，此时集成测试就结束了

（3）release分支

针对当前这个整体的版本，来从develop分支拉一个release分支出来，命名规范就是release/v1.0.0

然后就可以基于release分支进行系统测试了，QA同学介入，对于release分支部署的环境，进行功能测试，确保所有的功能都是ok的

如果此时发现有bug，同理，大家直接在release分支上去修复bug，包括在本地基于release分支代码复现bug，以及修复bug

release分支如果测试到后面，稳定了，功能都ok了，测试结束了

此时需要将release分支的代码合并到master分支上，同时将release分支的代码合并到develop分支上

（4）master分支

在最一开始，工程初始化的时候，就是基于master分支去初始化的

这边的话呢，每个版本的release分支都测试完之后，就可以将代码合并到master分支上来。此时master分支上的代码是经过了严格的测试的，单元测试、冒烟测试、集成测试、系统测试

接下来，就是要进行验收测试了

直接基于master分支的代码，部署到staging验收测试环境上去，在这个环境上，由PM来进行所有功能的验收

一般来说，99%的情况，验收测试环节，就不应该有bug了，让需求方体验一把整体的流程，在上线之前，做最后一轮check

如果这里发现了bug，那么在release分支上来复现这个bug，然后在release分支上来修改这个bug

bug修复之后，QA会在release分支和系统测试环境中，来验证说bug修复了，然后还会再做一遍回归测试

release分支分别合并到master分支和develop分支上去

验收测试通过之后，对master分支来打tag，比如货v1.0.0，打完tag之后，基于这个tag的代码，来进行线上系统的部署

（5）bugfix分支

线上发现了bug，而且判断这个bug的修复要超过1天的时间，那么需要从master分支拉一个bugfix分支下来，命名规范是bugfix/xx\_bug

在自己本地复现这个bug，基于bugfix分支上的代码，在自己本地来修复

修复好bug之后，将bugfix分支的代码，合并一份到develop分支上去，然后让QA在集成测试环境，来初步验证一下说是ok的

然后将bugfix的代码合并到master分支上去，将master分支代码在验收测试环境部署一下，让PM验证一下，bugfix是ok的

给master分支打一个tag，再次将修复好bug的代码给上线

（6）hotfix分支

线上发现了一个bug，很紧急，需要在1天之内必须修复，哪怕加班到凌晨3也得修复，整体流程跟bugfix分支一样

命名规范是hotfix/xx\_bug

（7）分支清理

在一个大的版本最终完成上线之后，需要将这个版本对应的一些分支清理掉，比如说feature分支、release分支，需要删除掉

在修复好一个bug上线之后，需要将bugfix分支、hotfix分支，删除掉

2、commit规范

在各个分支上开发的时候，git最基本的就是git commit，git push

git commit，是可以随便瞎commit的吗？每次commit的规范是什么？

（1）commit的时机：在feature分支上，一般建议，是每天提交一个commit；在release分支上，每次修复好一个bug，提交一个commit；在develop分支上，每次修复好一个bug，提交一个commit；bugfix分支，修复好bug之后，提交一个commit

（2）commit comment的规范

标题：简短的说明了，你这次commit是干了什么，一般就是几十个字，不超过一行

本次提交的代码改动列表：

1.完成UserServiceImpl的编写，完成了用户增删改查的功能实现

2.完成UserMapper的编写，完成了用户增删改查的数据库操作逻辑的实现

3.

4.