非常经典的GitFlow工作流

这个工作流，我其实以前也用的挺多的，版本稳定迭代的中小型项目，中大型项目，版本稳定迭代

1、GitFlow工作流是什么

（1）项目刚开始建立，就两个分支，master分支和develop分支，指向的commit是一样的，代码是一样的

（2）开始做第一个版本，v1.0，涉及到3个功能，3个码农，每个人开发一个功能，每个人从develop分支拉一个feature分支出来

（3）张三，李四，王五，分别在自己的feature分支上，吭哧，吭哧，吭哧，埋头开发，每个人自己写自己的代码，写单元测试，本地冒烟测试，这个过程中，肯定是每天都会提交一部分代码，可能就是涉及到多个commit。gitlab，为了保证程序员本地的代码不能丢失，每天都完成一部分代码，然后就要将那个分支的代码推送到gitlab去保存。

（4）一段时间之后，李四、张三、王五分别写完了代码，做完了自己本地该做的测试，就开始要做集成测试了。但是比较传统的方式，就是每个人先吭哧吭哧自己玩儿，写完代码之后，可能都好多天过去了，然后开始集成，互联网公司，不叫集成，联调测试

（5）每个人都将自己的代码合并到develop分支上去，以develop分支作为基础进行集成测试，此时就可以三人关小黑屋，传统开发，没有用敏捷，持续集成，没有自动化测试。公司里弄一个会议室，三个人关在小黑屋，开始执行各种各样的集成测试用例，3个人的代码集成在一起，看看整个系统的流程能不能跑通

（6）此时开始进行QA测试，QA妹子开始介入，90%都是女孩儿，搞一个这个v1.0版本对应的release分支，从develop分支拉出来，一边测试一边修复bug，QA测试结束，此时可以进行预发布测试

（7）QA测试都结束了，功能，性能，压力，可用性，稳定性，易用性，进入预发布环节，要将release分支的代码合并到master分支

（8）此时用master分支的代码在预发布环境，模拟线上的环境，进行测试和验证，最后的验证。互联网公司，可以从线上拷贝部分线上流量，观察系统是否正常运作。同时QA和PM，一起进行最后的验收和验证

（9）直接用master分支的代码进行v1.0版本的上线，预发布测试完之后，马上就要上线了。先干一件事情，就是给此时此刻的master分支打一个tag，标签，v1.0.0。作为v1.0.0可以上线生产使用的代码的一个快照版本。以后如果要回退代码的话，很方便，直接使用v1.0.0 tag对应的代码即可。

（10）上线之后发现有bug，此时需要从master分支拉一个bugfix分支下来，快速进行bug复现，修复，合并到master和develop两个分支。

（11）每次master代码上线之前，都必须对master打一个标签，v1.0，v1.1。tag，标签，是什么？其实简单来说，就是指向master指向的那个commit的一个指针而已，tag指针是不能移动。说明这个commit就是一个可以稳定的可以上线的某个版本的代码。

（12）feature分支用完了之后，是要删除掉的；release分支测试完了之后，也是要删除掉的；bugfix分支用完了之后，也是要删除掉的；唯一长期保存的只有master和develop两个分支，一个用于持续集成，一个用于稳定上线

2、GitFlow工作流+GitLab的实战

（1）首先要先准备好master和develop两个稳定分支，可以通过命令行去做，为了让大家学习git命令行的操作，其实真正在公司里，我们创建分支，都不是研发人员在本地自己创建的，都是在gitlab上创建分支的

在本地创建develop分支，然后推送到远程仓库去

git checkout -b develop，基于master创建了一个develop分支，切换到了develop分支

git push -u origin develop，将本地的develop分支推送到远程仓库，同时在远程仓库建立一个develop分支，将本地develop分支和远程develop分支关联起来

git branch -vv，看一下本地分支和远程分支的对应关系

（2）模拟张三和李四两个人来做功能的开发，先做第一个v1.0.0版本，每个人拉一个feature分支，从develop拉的feature分支，分别是feature/001和feature/002，基于gitlab去创建

（3）张三在本地将远程的feature/001分支拉取下来，会得到一个origin/feature/001，然后他需要在本地建立一个本地分支，叫做feature/001，将两者关联起来

先执行：git fetch origin，可以得到两个最新创建的分支

再执行：git checkout -b feature/001 origin/feature/001，建立本地分支feature/001，跟踪远程的origin/feature/001

李四如法炮制，在本地搞一个feature/002，同时需要搞一个develop分支

（4）张三和李四在本地对feature001和feature002都开发完了，假设做完了单元测试，本地的简单的功能测试，冒烟测试，然后开始进行集成测试

（5）张三和李四需要将feature/001和feature/002都merge到develop分支上去，注意，此时不再是在本地执行了，需要将feature/001和feature/002的代码都push到gitlab，然后在gitlab上提交merge request，发起code review，让级别更高的同事，或者是其他同事看一下你修改的代码有没有问题，进行代码走读，代码审查，code review

（6）然后在develop分支进行集成测试

（7）基于develop分支拉一个release/v1.0.0分支出来，基于这个分支开始测试，然后各种修复bug

（8）然后将release/v1.0.0合并到master分支

（9）对master分支打一个标签，然后master代码就可以上线了

（10）上线以后，可能会发现张三负责的代码有bug

3、GitFlow工作流适用的场景是什么

适合的就是版本稳定迭代的项目，一个版本，接一个版本，接一个版本

面向企业内部的系统或者项目，OA系统，CRM系统，ERP系统，或者面向外部的系统，比如说电商系统，但是系统已经成熟稳定了，版本稳定迭代

4、GtiFlow工作流的不足之处在于什么

我们之前做很多项目，刚开始都是遵照GitFlow去走的，但是如果是做一些互联网公司里面，面向外部的重要的业务系统，业务发展的过程中，快速迭代，同时5个版本，8个版本，10个版本，多达几十号人频繁的拉出多个版本并行开发

develop分支会处于一个极其不稳定的状态

而且不同版本的测试和开发进度不一样的

比如v1.0已经集成测试完了，要进入QA测试了，但是此时develop分支还混合了v1.1版本的多个代码，你如果此时直接基于develop分支拉一个release/1.0分支，会把develop分支里面的v1.1的代码也带进去

853769620