Git 工作流程(前端) - 新

说明

目前大部分使用频率高的仓库采用新流程, 其他仓库采用旧流程, 发现项目由各种 feature- , fix- 开头的分支就是新流程, 否则发现项目有很多派生仓库就是旧流程; 如果发现新流程项目实际上未配置好自动发布, 合并后没有发布到相应的环境, 找运维和后端配置自动发布

地址

http://gitlab.leadscloud-devops.link/

仓库

以询盘云项目为例 http://gitlab.leadscloud-devops.link/fe/element

工作流程

正常迭代:

1. 迭代开始, 从 master 分支新建功能分支, 分支命名 feature-(tastName)-(yyMMdd)
2. 迭代提测, 从 gitlab 网站上创建合并请求, 发给相应测试申请合并 feature 分支到 test 分支, 测试合并后自动发布小机房
3. 迭代提测遇到冲突, 切换到本地 test-conflictResolution 分支(本地解决冲突分支), 拉取远端 test-conflictResolution 分支 + test 分支最新代码, 确保本地 test-conflictResolution 代码最新, 之后合并本地 feature 分支到本地 test-conflictResolution 分支, 解决冲突后推送到远端 test-conflictResolution , 此过程不经过测试, 随后合并 test-conflictResolution 到 test 分支, 合并申请发给测试
4. 测试通过准备上线, 首先确定 master 分支全部都是可上线的代码(无测试中的代码, 比如 bug 修复之类的), 随后拉取最新 master 分支到 feature 分支, 再检查下自己的功能有没有受影响, 无问题则上 gitlab 申请合并 feature 分支到 master 分支(目前是运维审核合并申请), 合并完成后, 代码自动发布到灰度, 产品和测试进行功能验收, 验收没问题, 前端在 tapd 项目中提交"发布评审", 审批完成后, 运维复制灰度代码到线上服务器, 上线
5. 上线完成, 没啥问题的话记得过段时间就删除掉 feature 分支, 避免分支越积越多

BUG修复:

1. 从 master 新建 BUG 修复分支, 分支命名 fix-(tastName)-(yyMMdd)
2. 同上, 合并小机房提测, 部分问题只能在灰度上验证的, 问过测试能直接合 master 上灰度测试的话就提合并申请, 尽快测完上线
3. 测试完毕, 如果已经在 master 上, 确保 master 分支无其他不可上线内容时, 要尽快让测试申请上线; 否则以测试意见为准, 马上上线 或等其他一起上线

Git 工作流程(前端) - 旧

地址

https://git-dev.leadscloud.com/

仓库

1. 线上
   不可见
   由运维(李世鹏)负责从 prod 手动发布
2. prod
   灰度环境对应的仓库
   由测试负责人(李亚丽, 杨大伟)负责从 test 提交合并申请到 prod , 运维(李世鹏)负责合并, 合并后自动发布;
   或者由前端负责人(马源志, 汤一飞)负责从 bug 修复仓库提交合并申请, 合并后自动发布 bug 修复内容;
3. test
   测试环境对应的仓库
   由前端负责人负责从 dev 的 master 分支 提交合并申请到 test ,测试负责人(李亚丽, 杨大伟)负责合并, 合并后自动发布
4. dev

开发环境对应的仓库, 包含多个分支, 详细见下方**工作流程**

相关开发人员从自己的私人仓库提交合并请求到 dev , 前端负责人负责合并, 迭代整体自测通过后, 将迭代分支合并到 dev:master 分支, dev: master 分支提交合并申请到 test 发布测试

5. 私人仓库

私人开发仓库, fork 自 dev 仓库, 需要注意及时同步 dev 的分支

迭代开发期间提交合并申请到 dev , 迭代周期中的开发周结束后, 迭代整体自测通过后, 前端负责人统一提交到测试;

注意: 尽量按功能点提交合并申请, 尽量完善提交信息, 提交合并申请时尽量指定合并负责人

6. 本地

本地开发仓库, fork 自开发人员私人仓库, 在本地进行开发

工作流程

正常迭代:

1. 迭代开始, 前端负责人从**dev:master** 上建一个迭代分支, 如:**dev:1.0** (迭代小组各自建分支, 如 **dev:1.0-1**, **dev:1.0-2**)
2. 开发人员拉取**dev:1.0** 到私人仓库和本地仓库, 如**locale:1.0** , 在此分支上进行开发, 开发人员自测完成后提交合并申请到**dev:1.0** , 前端负责人合并代码到 **dev:1.0**
3. **dev:1.0** 所在迭代整体开发完成后(正常来说是两周结束), 前端负责人合并**dev:1.0** 到 **dev:master** , 随后发布合并申请到**test** 仓库提交测试
4. 如果有迭代小组未自测通过无法提交测试, 则仅合并可提交测试的分支代码到**dev:master** , 根据迭代流程, 迭代上线时间为周三
   若因进度原因迭代小组提测时间错开, 周三可能出现如下场景:
   本周三:
   - dev:1.0-1 提测, **dev:1.0-2** 不提测, 则仅合并**dev:1.0-1** 到 **dev:master** , 再提交**test** 发布测试
   - 有多个分支可提测, 则逐个合并到**dev:master** , 如: **dev:1.0-1** 合并到 **dev:master** , **dev1.0-2** 拉取 **dev:master** 代码后解决冲突再合并到 **dev:master**
   下周三:
   - 之前 **dev** 多个分支只提测了部分分支, 如:**dev:1.0-1** 测试通过可上线,**dev:1.0-2** 提测, 则先发布**test** 到 **prod** 完成灰度验证并上线, 之后 **dev:1.0-2** 拉取 **dev:master** 合并, **dev:1.0-2** 相关开发人员再自测一遍(避免冲突或代码被覆盖), 通过后合并**dev:1.0-2** 到 **dev:master** , 再提交**test** 发布测试
   - 之前已提测的分支未测试通过不可上线, 则所有分支都无法继续向下进行, 等待 BUG 修复完成后再继续开发, 下个迭代的开发时长要减去相应延期天数
5. 提交测试随后可以开始下一个迭代, 从**dev:master** 新建分支, 如: **dev:1.1** , 新迭代功能继续在此分支上开发(同步骤1, 迭代小组各自建分支)
6. **dev:1.0** 测试期间, 如果测试反馈功能有问题需修改, 相关人员直接在**dev:master** 分支上修改, 修改后提交前端负责人提交合并申请到 **test** 测试
7. 测试通过后, 测试负责提交合并申请到灰度, 产品验证无误后上线, 迭代整个上线后, 把**dev:master** 代码同步到**dev**
8. 提交测试之前, 线上紧急 BUG 修复, 参见下方**线上BUG修复**

列入迭代的BUG和线上BUG修复:

1. 前端负责人 fork **prod** 仓库形成 **prod-fix** 仓库, 开发人员 fork **prod-fix** 到私人仓库和本地仓库, 如:**prod-fix-locale**
2. 开发人员在 **prod-fix-locale** 修复代码, 随后提交合并申请, 前端负责人合并到**prod-fix** 后, 提交合并申请合并到**prod** 仓库
3. 运维人员通过合并申请, 自动发布到灰度环境供测试验证
4. 测试人员在灰度环境验证通过后 ,运维发布**prod** 代码到线上, 前端负责人拉取**prod** 到 **dev:master ,** 随后尽可能同步到各迭代分支, 如 **dev:1.0** (如果不及时同步, 最后才合并**dev:1.0** 到 **dev:master** , 可能会造成冲突)
5. **注意**: 原则上, 上午11点前的 BUG 当天修复, 之后的 BUG 第二天修复, 特别影响使用的 BUG 当天修复;
6. **注意**: 经过和产品和测试的沟通确认, 如果当天有 BUG 修复但没改完/不上线, 不能提交代码到**prod-fix** 分支
7. **注意**: 测试强调, 哪怕是已提测的 BUG 在灰度验证未通过, 想再改 BUG 发灰度也不能晚于下午 4 点, 如果当天要上线, 应该是撤回未通过 BUG, 而不是紧急改了让测试再测, 哪怕改起来非常简单

日常工作:

1. 每天晚上结束工作前, 提交代码合并申请(到 dev:xxx 迭代分支/prod-fix 修复分支)
2. 相关负责人早上来合并代码, 第二天早上10:30(考虑到可能加班第二天晚到)前端拉取各分支代码