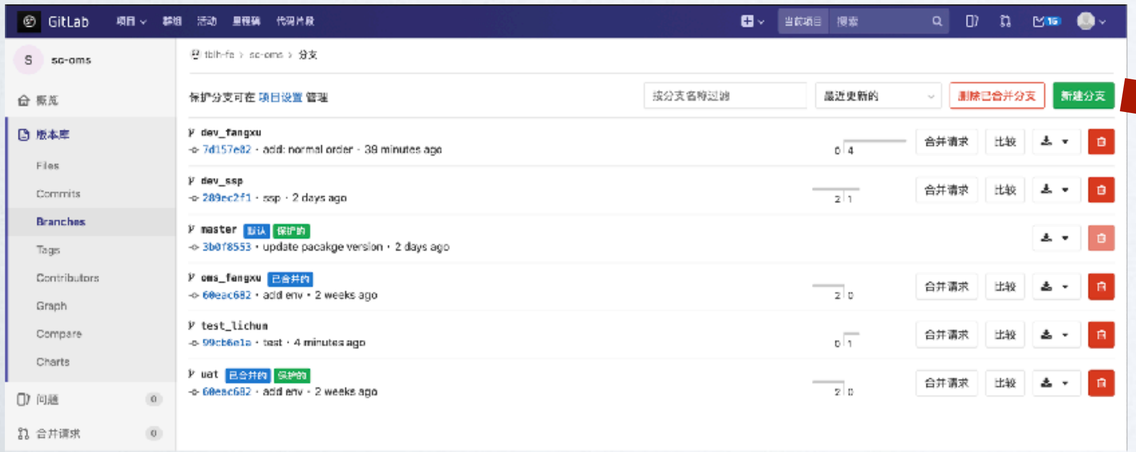
FE团队GIT使用规范（试行）

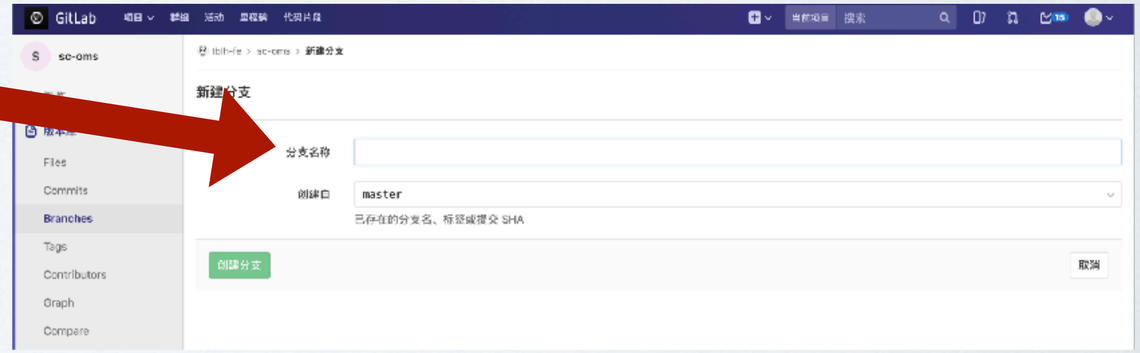
李淳 2020年5月18日

分支创建（1）

- Git是一个非常灵活的代码管理工具，但是为了避免一些难以处理的问题、减少误操作的可能性，推荐以下方式进行分支创建：

- 1、所有项目级分支都以创建远程分支为起点，在gitlab中进行远程分支的创建，多数情况下分支创建自**master**；





分支创建（2）

- 2、在本地代码仓库运行fetch命令，同步远程仓库；

```
$ git fetch
```

```
lichundeMacBook-Pro:sc-oms lichun$ git fetch
remote: Counting objects: 111, done.
remote: Compressing objects: 100% (108/108), done.
remote: Total 111 (delta 73), reused 4 (delta 0)
Receiving objects: 100% (111/111), 33.37 KiB | 1005.00 KiB/s, done.
Resolving deltas: 100% (73/73), completed with 8 local objects.
From http://gitlab.sftcwl.com/tblh-fe/sc-oms
 * [new branch]      dev_fangxu          -> origin/dev_fangxu
 * [new branch]      dev_ssp             -> origin/dev_ssp
 * [new branch]      develop_v1.0_lichun -> origin/develop_v1.0_lichun
 * [new branch]      oms_fangxu          -> origin/oms_fangxu
```

提示：业务开发时，务必创建远程分支，不要仅仅创建本地分支，开发完毕后再推送分支，不利于代码管理及协作开发。

- 3、在本地创建远程分支的**跟踪分支**，以确保分支自动切换到新建分支，且每次提交到的远程分支都符合预期；

```
$ git checkout --track origin/develop_v1.0_lichun
```

```
lichundeMacBook-Pro:sc-oms lichun$ git checkout --track origin/develop_v1.0_lichun
Branch develop_v1.0_lichun set up to track remote branch develop_v1.0_lichun from origin.
Switched to a new branch 'develop_v1.0_lichun'
```

分支管理规范

开发分支：

- 定义：正在开发的分支，也是项目开始我们需要创建的分支；
- 命名规范：develop_project-name_developer

分支类型

项目名称使用中划线分隔，例如v2.1-story2、huawei-2.0、zhongyue等

开发者姓名，例如lichun、zhangning等；
当多人开发同一项目时，根据需求类型及任务分割来确定是在同一分支协作开发，还是多分支独立开发；如若多人使用同一分支，开发者姓名可以省略；

提示：几个part之间使用下划线连接，part内使用中划线连接；

- 示例：develop_v1.0-story3_lichun

分支管理规范

多人协作开发分支：

- 定义：当多个人开发同一个项目，项目模块之间易解耦，此时进行多人多分支开发；
- 命名规范：**develop_project-name_common**

  - develop_project-name_developer1
  - develop_project-name_developer2
  - develop_project-name_developer3

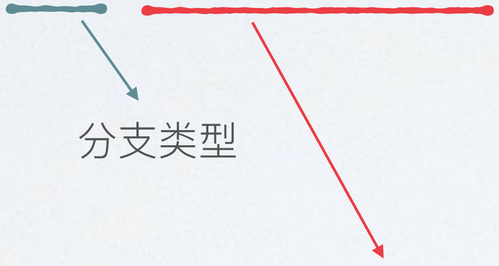
开发前由项目负责人梳理公共部分的开发范围，在开发中的公共组件、函数、框架修改都提交至此分支；

各个开发分支都以common为基准开发分支，需关注合并

分支管理规范

测试分支：

- 定义：C级以上项目开发完毕后，在提测前需要新建测试分支，将特定开发分支代码合入测试分支进行提测；而后**删除develop分支**；
- 命名规范：test_project-name



分支类型

项目名称使用中划线分隔，例如v2.1-story2、huawei-2.0、zhongyue等
与develop命名一致即可

【注意】提测至QA的分支，一定是以test_命名的，进入测试流程后，所有的修复也只在test_分支上进行，提测后务必删除develop分支，以避免更多的合并和协作问题。

分支管理规范

待上线分支**【可选步骤】**：

- 定义：C级项目测试通过，待上线UAT或正式环境前新建待上线状态分支，同步master及test_project-name分支代码；对于D级项目，自测通过后，新建待上线状态分支，同步master及develop_project-name分支代码；
- 命名规范：ready_project-name

【疑问】多数情况下test分支与ready分支代码是一致的，为什么要重新创建分支?

1、将同步master代码流程化；

2、留存明确的开发完毕分支，以备UAT环境、生产环境的双重上线；

分支管理规范

问题修复分支：

- 定义：项目上线以后，需要修复线上问题，或者进行一些优化，此时创建的分支为问题修复分支；
- 命名规范：fix_project-name_developer

分支类型

提示：如若修复的是UAT环境问题，项目代码仅上线至UAT环境未上线至生产环境，此时需要关注fix分支的留存，在允许的情况下将fix分支直接合入待上线的ready分支；

分支管理规范

master分支：

- 定义：生产环境对应master保护分支，UAT环境也需创建专门的master保护分支，以厘清UAT代码；
- 命名规范：
  - master，生产环境保护分支；
  - uat-master，代码仓库仅关联一个UAT环境；
  - uat-name-master，代码仓库关联多个UAT环境；

使用全小写字母及数字；

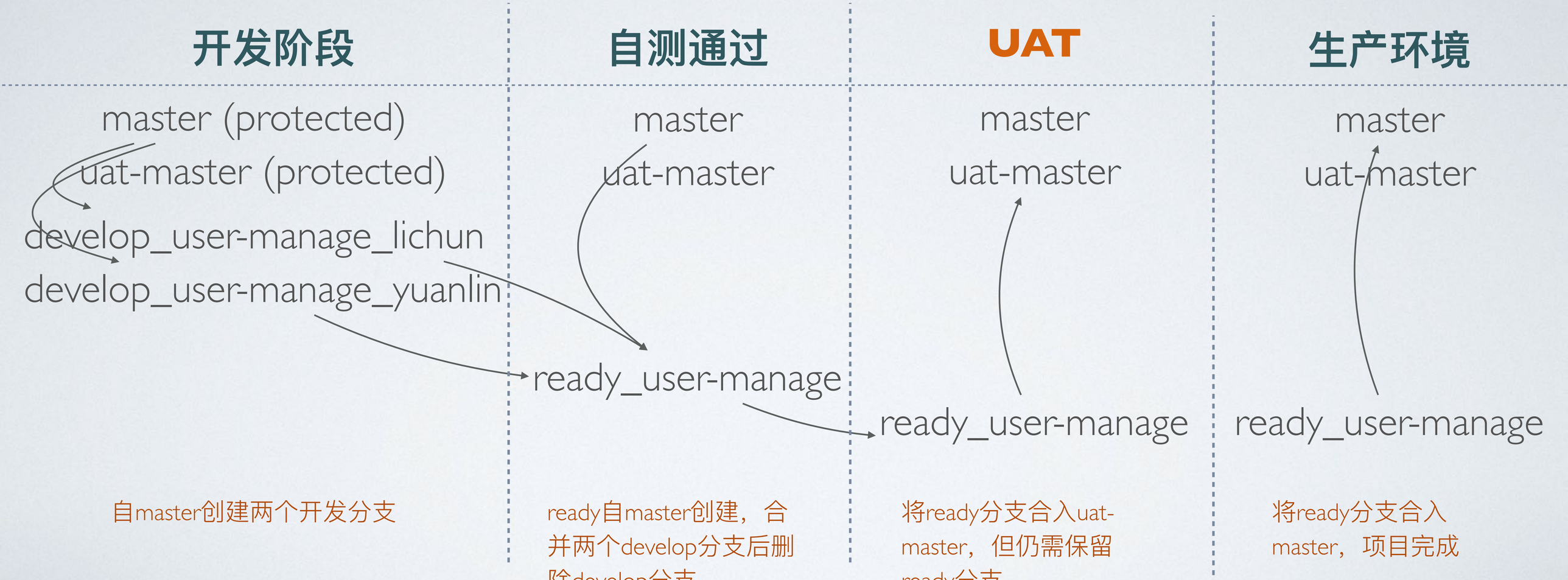
【概念释义】保护分支：此分支不能直接提交代码（所有者、主程序员除外），需发起merge request给所有者、主程序员进行代码合入，或者通过上线由Jenkins自动完成合入，也因此收紧主程序员权限；

C+级项目开发流程

提示：若无UAT环境，这两步为可选步骤

开发阶段	测试	测试通过	UAT	生产环境
master (protected)	master	master	master	master
uat-master (protected)	uat-master	uat-master	uat-master	uat-master
develop_v1.0_lichun				
develop_v1.0_yuanlin				
	test_v1.0			
develop_v1.0_common		ready_v1.0	ready_v1.0	ready_v1.0
自master创建开发分支，如有公共部分，可创建common分支	test自master创建，合并两个develop分支后删除develop分支	ready自master创建，合并test分支后，删除test分支	将ready分支合入uat-master，但仍需保留ready分支	将ready分支合入master，项目完成

D级项目开发流程


开发阶段
自测通过
UAT
生产环境
master (protected)
uat-master (protected)
develop_user-manage_lichun
develop_user-manage_yuanlin
master
uat-master
ready_user-manage
master
uat-master
ready_user-manage
master
uat-master
ready_user-manage
自master创建两个开发分支
ready自master创建，合并两个develop分支后删除develop分支
将ready分支合入uat-master，但仍需保留ready分支
将ready分支合入master，项目完成

JENKINS

- 1、使用Jenkins进行上线发布时，Jenkins会自动检查待上线分支是否落后于master，如若落后会发布失败；
- 2、Jenkins上线完毕后会自动将分支代码合入master，并删除上线分支，因此上线一定是串行的，如若需要保留上线分支，请上线前进行备份；
- 3、Jenkins支持编译时向build.sh脚本传递参数，可在此进行一些区分打包的操作；
- 4、...

分支管理的一些原则

- 1、代码在云端；
- 2、舍得建分支，但是分支关系一定要简单，明确上下游；
- 3、不要彼此交叉合，尽量创建新分支来合并多个分支；
- 4、要舍得删，在项目的不同阶段保留明确的active分支即可；
- 5、要记得留，特殊分支的备份和保留，要防止被Jenkins上线后删除；