版本控制指引

概念

产品版本

ALM里项目/子项目所定义的版本，两节：x.y。如智计划1.2，那么这个产品版本号为1.2。

工程版本

代码构建产物的版本。工程版本号通常为四节,x.y.z.build。 x.y 继承产品版本号；z在敏捷项目团队中是冲刺编号，在瀑布团队中为流水号。该节用于区分不同的研发、上线周期；build为构建流水号，通常由构建工具自动生成。

测试代码同样也有工程版本号。目前测试代码没有发布流程，也没有通过工具发布，所以z.build由测试团队人员自行决定。原则上每一次正式交付，变更z。

发布标签

每一次发布到生产，在git代码库中，给对应的commit标记一个标签，值为部署物的工程版本，即x.zy.z.build。

目标

统一分支管理策略以及定义。版本化一切，最终提高项目的团队合作效率、加速新功能开发和发布管理。

原则

- 采用GitHub Flow策略（推荐）或基于主干开发策略。
- 要求项目有完善的自动化测试、持续集成和部署等相关的基础设施。
- 版本化一切。
- 团队有代码审查的相应流程。
- 具备自动部署的条件

规范

代码仓库

- 必须：代码应放在集团统一的代码仓库，https://git.longhu.net。
- 必须：工程的可见性不能设置为public。
- 必须：只允许给项目相关开发人员配置权限，并应该遵循最小授权原则。
- 必须：代码项目创建在团队空间内，不允许放在个人空间。
- 建议：团队空间命名格式为：group/[subgroup]。group为产品线简称，如果产品线有多个子产品，再加上subgroup。subgroup为子产品简称。如 gaia/gfs，group=gaia，subgroup=gfs。
- 建议：代码项目命名格式为：项目代号-模块[-子模块][-孙模块]。如：gaia-gfs-demo，halo-wechat。这样，GIT中完整的空间-项目名为：gaia/gfs/gaia-gfs-demo。

分支命名【GitHub Flow】

- 必须：master分支被保护，不允许直接提交至master。
- 必须：创建分支使用有意义的名称头，功能开发用 feature/{JIRA编号}-*，bug修复用 bug/{JIRA编号}-*，热修复用 hotfix/{JIRA编号}-*。如 feature/XM1907901-1390-enable_audit_log_for_inventory, bug/XM1907901-1403-xxxxxxx。
- 建议：创建分支开始后，就创建一个MR，用于描述思路，并记录讨论过程。
- 建议：未完成的MR以WIP:开头，如： “WIP:用户1分钟未有动作，自动锁屏”。
- 必须：至少有一个成员同意合并，才允许合并分支。
- 建议：合并分支时，使用 --squash，或选中gitlab界面上的"[X]Squash commits when merge request is accepted"。

版本

- 必须：每一次准备发布生产的交付物，打上发布标签，并记录Release Notes。
- 建议：数据库有版本号。不同版本的数据库变更脚本，支持幂等操作，以便自动化完成升级/降级。
- 建议：测试用例、测试数据也有版本号，要么和代码工程版本一致，要么自定义，和代码工程版本有对应关系。
- 建议：推荐使用Flyway，EntityFramework Migrations来管理DB版本。

推荐实践

版本号使用场景

1. 敏捷项目冲刺开始，定义本冲刺版本的前三节。
2. 提交、解决、关闭bug时，正确填入了DevOps中对应的工程版本号。
3. DevOps构建时，根据前三节，自动补充最后一节的编译流水号。
4. 发布至生产环境时，DevOps根据工程版本，自动生成tag标签。
5. 如果只通过DevOps部署，则没有冲刺版本。DevOps根据产品版本，自动加上制品的上传批次号，如1.2.13，以追踪各环境和不同版本部署物的对应关系。

假设场景

项目名称	开发启动时间	移交测试时间	回归时间	上线时间
XX项目	9月1日	9月15日	9月25日	9月26日

开发流程

序号	时间	事项	描述	命令	说明
1	9月1日	从master新建分支	从master创建分支。分支名称：feature/11-Add_redis_support	git checkout master git pull git checkout -b feature/11-add_redis_support git push --set-upstream origin feature/11-add_redis_support	所有参与人员都提交到此分支
2	9月2日	设计、编写代码与测试	提交、提交、提交	git add --allgit commit -a -m "move display name to redis" git push	
3	9月3日	开启MR，以开启讨论和Review	从gitlab的站点中创建一个MR。		如果并未做完，MR以"WIP:"开头
4	9月15日	讨论、修改、测试	讨论方案、修正review的改进项。	git add .	
5	9月16日		循环修复bug并不断push到远程分支。	git commit -am "move on and on"	
6	.....		每日merge master代码到分支。	git merge master	
7	9月25日	自动化、人工验收全部通过，MR通过	master代码可以发布时，打上版本号标签，1.1.0	git tag add "1.1.0" git push --tag	选中 "Squash commits when merge request is accepted.“ 选中 "Delete source branch when merge request is accepted"
8	9月26日	部署	生产环境部署master。		通告其他分支开发人员尽快merge master代码