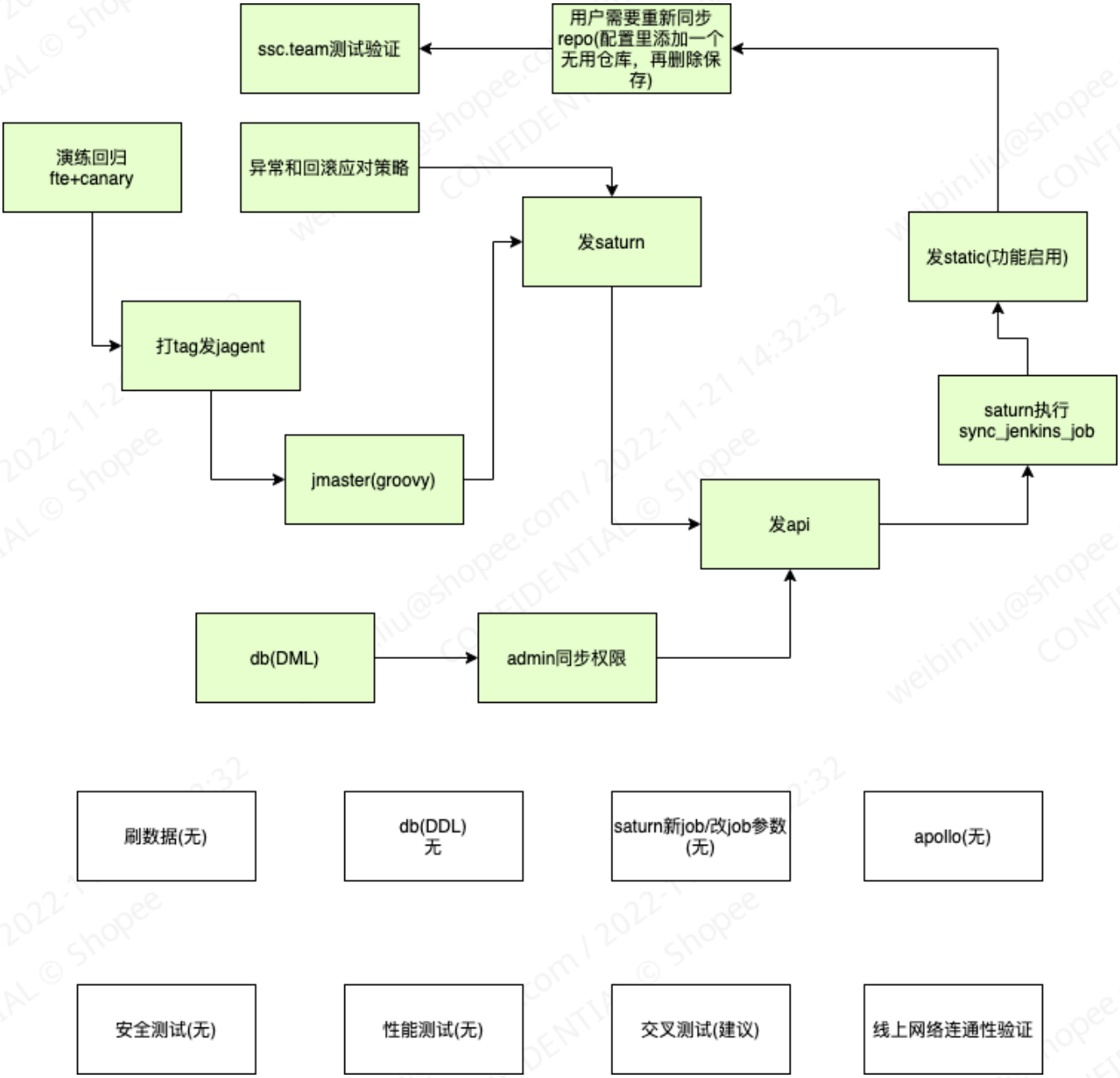
上线反思和规范

dms系统上线涉及的细节较多，感觉不够注意的话，很容易出现测试通过，但上线后存在问题的局面，需要从多次发生的问题里吸取教训

基本思路

系统涉及的边缘较多，以上是目前所有上线操作所涉及的checklist，并处于动态完善中（不含依赖系统，比如athena，k8s，zeus，等其他三方系统需要仔细联调，并且确认网络连通性）



如上操作，绘制上线线路图。图上是fte功能上线的示例，将需要涉及的点放到一起，并且使用箭头来描述依赖顺序，这样梳理以后，方便知道哪些任务可以并行，哪些要提前做。如果两个先后步骤之间，服务存在部分不可用，则需要用特别的线形加文字描述，提醒在这里需要连续操作。

开始时，所有步骤都是白色，每执行完一个步骤，将步骤变色(长时间的步骤，可以在执行中时有自己的颜色)，方便记忆，避免遗漏操作。 思考：processon和drawio可以共享文档？是否适合于多人配合的上线？

一般的顺序经常是jagent->jmaster->saturn->api->static，前面可能有DDL，后面可能有数据同步脚本和线上验证。如果一个新功能有条件使用开关，则在live环境验证完成之前不开放功能。

原则上，需要依赖外部团队的操作（如审批，和三方沟通）尽量提前操作，如果这样的操作会影响旧版本代码的业务，且不能有效把控服务受影响的时间，则考虑通过先上开关配置等方式解决。假设上线前演练需要一定时间，且上线前需要执行DDL等操作，且DDL操作执行后需立即上线者，由于DDL的执行时间点不定，要么和DBA沟通好；要么先执行完演练，再提DDL单(更推荐)。

每一步的要点

演练回归应该简短描述回归的范围(哪些功能)，包括新增/修改的功能的演练和旧功能的兼容。并且演练时应该尽量将系统恢复到上线前水平，并且逐一按图上的上线步骤操作(最好是staging环境，test环境有时候存在回退不彻底，导致遗漏的教训)

db应该用google文档去记录，每次执行了任何DB操作，都应该在第一时间记录文档，避免遗漏DB语句的问题；同样的apollo也是类似，建议apollo如果只是新增，则尽早在live同步新增(apollo有对比脚本可以对比不同环境的差异，db有data manager可以对比库表结构，只有DML要特别小心谨慎)

加接口需要注意执行DML和权限同步的问题

加按钮和菜单需注意权限记录

加saturn任务需注意在live环境也需要做saturn配置。可以提前配置但禁用，在上线后打开，避免遗忘

需要有回退方案，如果上线操作较大，应演练回退方案

需要访问三方服务时，如果要从live的容器内访问，需要做网络连通性验证。（FTE就是因为忘记在live验证连通性，结果test能访问的env列表在live无法访问，只得架桥）

刷数据脚本要点

刷数据的脚本应该做成幂等的，并且如果刷数据是不能直接回退的操作，则应该在刷数据前dump全部数据再刷，并且事先演练好dump脚本和回滚数据脚本(在添加单元测试时使用了这个手段，结果在刷数据时，因为live和test的原始json数据存在不同，导致test可测的脚本不能兼容live数据，引发数据全部被破坏，幸亏有回退脚本，立即做回退操作后逐步排查修复)

测试建议

使用ssc team(普通权限)和自己(superuser)分别测试，如果添加了接口，需要使用ssc team验证权限是否有问题。涉及多项目的使用id为2的项目验证

最好交叉测试，即一部分功能由另一个人验证

进行中和需要做的事情

已做，需维护：

mock同步和完善(其实在上线上很有用)

可做：

搭建staging环境用于回归