GitLab迁移实施方案

一、前言

当前，研发中心GitLab服务面临如下问题。

- 有两个服务平台，项目和人员维护成本增加，数据统计不便，资源也无法共享，协同效率受影响。
- 人员及组织架构与实际不符，权限不明确，导致成员角色定位模糊，操作时易混乱。
- 项目及分支缺乏管理，无法进行全局管理，阻碍后续扩展工作开展。
- Git代码提交不规范，无法关联提交、需求和人员，导致代码追溯、责任明确及项目复盘等工作困难重重。

为解决上述存在问题，制定本方案，用于指导完成GitLab服务合并，人员组织管理工作，为后续工作开展奠定基础。
除特殊声明外，本文中的操作项默认在云上仓库中。

二、目标

1. 完成两个GitLab服务平台的整合，将所有项目迁移至云上GitLab服务。
2. 依据研发中心组织架构信息，完善GitLab服务中的用户群组关系。
3. 制定项目及分支管理规范标准，实现项目及分支的规范化管理。
4. 借助规范的代码提交机制，实现代码追溯、责任明确以及项目复盘等工作开展。

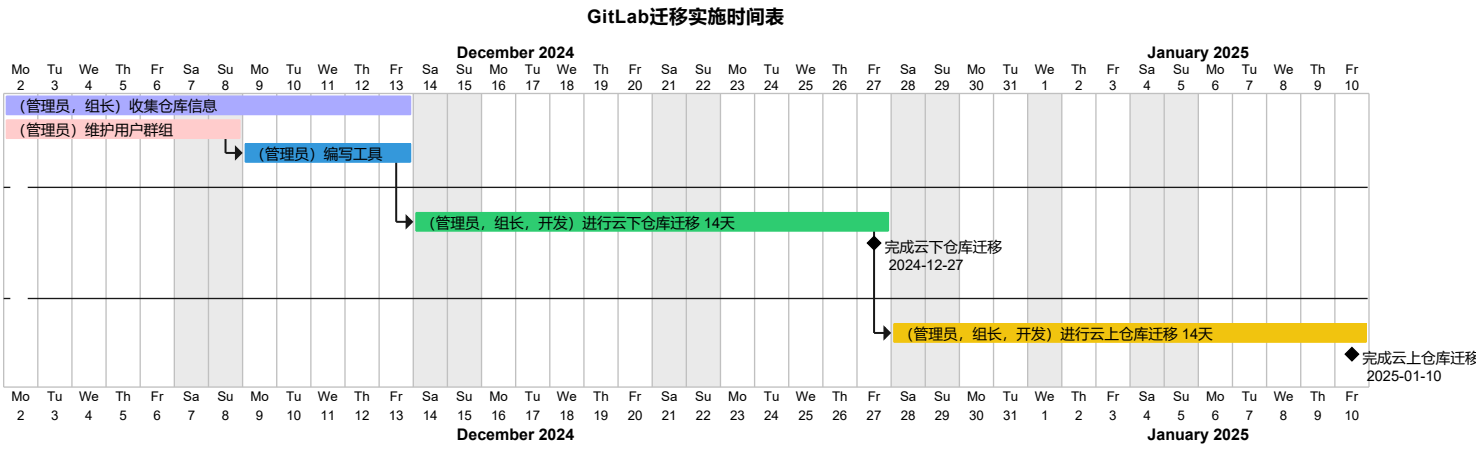
三、职责分工

实施方案的参与人员包括GitLab管理员、研发组长及所有开发人员。

- 管理员负责制定方案，采集仓库信息，维护用户群组，编写迁移工具，编排迁移计划，调度围殴仓库迁移，处理紧急问题等工作。
- 研发组长负责协助采集仓库信息，协调开发人员配合迁移等工作。
- 开发人员需要配合完成仓库迁移阶段工作。

四、实施阶段

时间表



阶段一、收集项目仓库信息

- 该阶段参与人员为研发小组组长及管理员，各研发小组组长负责填写信息内容，管理员负责设计组织整理等工作。

- 收集目的在于界定项目仓库与研发小组之间的归属关系，用于支撑云下仓库迁移，云上仓库管理，方便后续工作开展实施。
- 收集范围涵盖云上，云下所有部署在生产环境中的及被公共使用的应属于研发小组的项目仓库。
- 收集内容包括，归属研发小组，仓库地址，开发主语言，描述。
  - 研发小组：研发中心组织架构中的研发小组，包括基建运维中心，数据管理中心，交易中心，技术中心，质量管理组，渠道管理组，客车业务组，运营中心组，基础业务组，货车业务组，智能洗车，静态交通组，新业务一组，车服能源，ETC自助机组，企业平台组，开放平台组，个人平台组。
  - 仓库地址：代码仓库地址，.git结尾。
  - 主语言：项目主要开发语言，可为空。
  - 描述：项目简单描述。
- 收集表示例：

研发小组	仓库地址	主语言	描述
智能洗车	http://10.10.7.52/wash/wash-server.git	Java	洗车业务核心业务系统

阶段二、维护用户及群组

管理员负责维护GitLab用户及群组，该阶段与收集项目仓库信息同步进行。该阶段工作内容包括：

1. 根据研发中心组织架构中的人员信息创建所有开发人员用户信息。
2. 禁用已离职人员，已废弃服务用户信息。
3. 根据研发中心组织架构中的研发小组信息创建群组。
4. 为群组分配成员，一个用户可以是多个群组的成员。
5. 权限配置，研发组长为群组Owner，研发人员身份（Developer，Maintainer）由研发组长分配。
   - Owner 群组管理员，可以修改群组信息，管理子群组，管理群组成员，管理群组项目。
   - Maintainer 项目管理员，可以管理项目，管理项目成员，添加标签，管理保护分支，进行代码合并。
   - Developer 项目开发人员，可以拉取代码，提交代码，发起合并请求。

阶段三、编写迁移工具

管理员负责编写迁移脚本工具，对接GitLab API，实现迁移管理能力。工作内容如下：

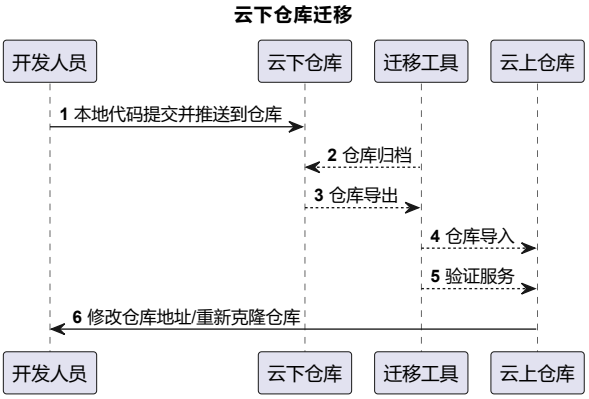
1. 因云上云下GitLab版本不一致，需分别开发云下仓库归档，取消归档，仓库导出功能。云上仓库导出，仓库导入，仓库归档，取消归档，迁移群组功能。
2. 根据收集的项目仓库信息，确定目标新仓库的归属信息，包括群组，命名空间，项目描述。
3. 编排迁移执行流程，制定分批实施计划。

阶段四、云下仓库迁移

参与人员

云下仓库迁移分批进行，该阶段参与人员包括批次内涉及到的开发人员及开发组长。
管理员负责调度及使用迁移工具实施迁移工作。
开发组长负责协调组内开发人员配合迁移。
开发人员需要完成代码提交，更换地址工作。

迁移步骤



1. 开发人员将本地代码提交并推送到云下仓库。
2. 迁移工具将云下仓库进行归档操作，归档后无法再对代码内容进行修改。
3. 迁移工具将云下仓库导出，并下载到本地。
4. 迁移工具将仓库导入到云上仓库，并配置项目归属群组，项目描述等。
5. 迁移工具进行服务验证，包括克隆，提交，推送，确保迁移成功。
6. 开发人员可选择修改本地代码对应的仓库地址或者重新克隆仓库，完成迁移。

- 完成迁移后，项目成员自动继承群组成员，组长可进行管理。
- 完成迁移后，云下仓库处于归档状态，无法推送代码，但可以查看代码。
- 云下个人项目可由个人自行迁移。

风险应对

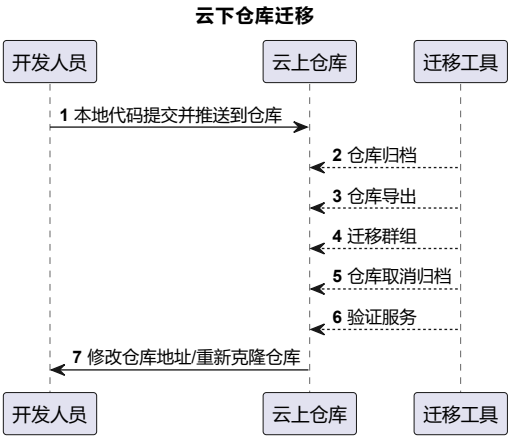
- 风险一，会对开发工作造成影响。每个项目单独进行迁移操作，借助工具自动化操作，将对开发工作影响降到最低。
- 风险二，未提交代码丢失。积极沟通确认，确保所有的本地代码已提交到代码仓库，防止有代码遗漏。
- 风险三，迁移失败。保留旧仓库，一旦迁移失败，将云下仓库取消归档，恢复正常服务。

阶段五、云上仓库迁移

参与人员

云上仓库迁移分批进行，该阶段参与人员包括批次内涉及到的开发人员及开发组长。
管理员负责调度及使用迁移工具实施迁移工作。
开发组长负责协调组内开发人员配合迁移。
开发人员需要完成代码提交，更换地址工作。

迁移步骤



1. 开发人员将本地代码提交并推送到仓库。
2. 迁移工具将仓库进行归档操作，归档后无法再对代码内容进行修改。
3. 迁移工具将仓库导出，并下载到本地，做仓库备份留存。
4. 迁移工具将将仓库迁移到目标群组。
5. 迁移工具将将仓库取消归档，恢复正常使用。
6. 迁移工具进行服务验证，包括克隆，提交，推送，确保迁移成功。
7. 开发人员可选择修改本地代码对应的仓库地址或者重新克隆仓库，完成迁移。

风险应对

- 风险一，会对开发工作造成影响。每个项目单独进行迁移操作，借助工具自动化操作，将对开发工作影响降到最低。
- 风险二，未提交代码丢失。积极沟通确认，确保所有的本地代码已提交到代码仓库，防止有代码遗漏。
- 风险三，迁移失败。发生迁移失败，仓库丢失时，可利用仓库备份重新导入项目，恢复正常服务。

五、提交规范

制定提交规范

1. 提交用户应为真实姓名。

```
# 设置用户名
git config  --global user.name '猪小明'
git config  --global user.name
```

2. 提交信息应该涵盖提交类型，描述，需求信息

```
<type>: <desc>

[detail]

no: <no>
subject: <subject>
```

- type:
  - feat: 功能新增，新需求，新功能开发
  - fix: 问题修复， 修复问题，Bug
  - docs: 文档更新，文档更新，注释修改等非功能源码文件
  - style: 格式调整，不影响代码逻辑的格式调整，如代码缩进、空格使用、换行符统一等。
  - refactor: 代码重构，重构代码，但不改变其行为。例如，提取公共函数、重命名变量或函数、优化算法等。
  - test: 测试相关，单元测试代码，测试挡板修改等。
  - chore: 杂项任务，其他不修改源代码或文档的维护性任务，如构建系统调整、依赖库更新、项目配置文件修改等。
- desc: 内容简述
- detail: 可选，内容详细描述
- no: 需求，问题编号（钉钉，飞书，质量管理平台）
- subject: 可选，需求，问题标题

eg.

```
feat: 用户登录功能

实现用户登录，用户同步，权限分配

no: 52003714
subject: 优惠券营销需求
```

校验提交规范

利用GitLab提供的服务器钩子机制，实现代码提交规范校验，编写校验脚本，在每次推送入库前校验提交用户名和提交信息是否符合规范。