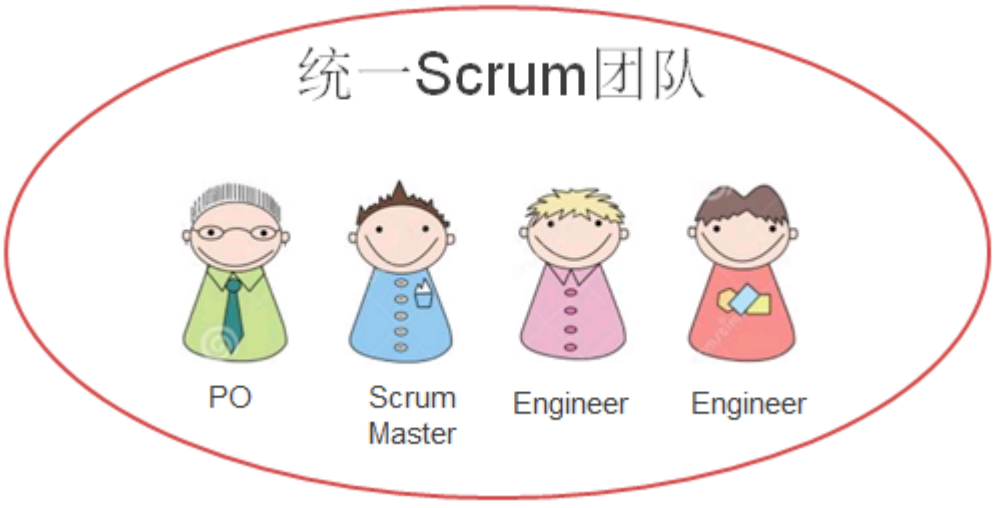
PaaS Scrum团队管理实践流程梳理

一、团队组成与团队目标

经过部门内多番讨论，个人理解目前的团队组成已基本达成统一：**对业务统一负责的全功能团队**。



具体理解上，个人认为有两个重点：

- 对业务统一负责：基层的Scrum团队是按照DDD思想，根据领域划分的。对领域内业务，团队是直接责任主体，统一对该领域业务的成败负责。目标明确。
- 全功能团队：团队内包含前后端开发、PO（产品经理角色）、Scrum Master（兼职），并通过自动化流水线和测试工具实现对交付质量及流程的管控，因此团队是业务从设计到交付的全功能团队。责任清晰。

需统一的两个观念：

- 前后端在业务团队上属于同一个业务团队，共同承担业务领域的成败责任。成，一起接受荣誉；败，一起承担责任。
- 业务团队对交付内容的质量负第一责任。意味着，团队不要等待转测后由专门测试团队来对我们的交付质量把关。而应通过工具和基础测试用例，在交付前就对交付质量做到心中有数。（性能压测、边界测试、复杂场景测试除外）

二、团队外部流程梳理

Scrum团队作为公司研发的基层团队，除自身的内部运作外，还需要与部门内、跨部门、市场、客户等多个利益相关方存在交互。

这部分团队外的协作流程，公司已经长期运行了以IPD为核心的产品研发管理流程，作为Scrum团队的外部环境和流程，我们需要有一个统一大致的认识。

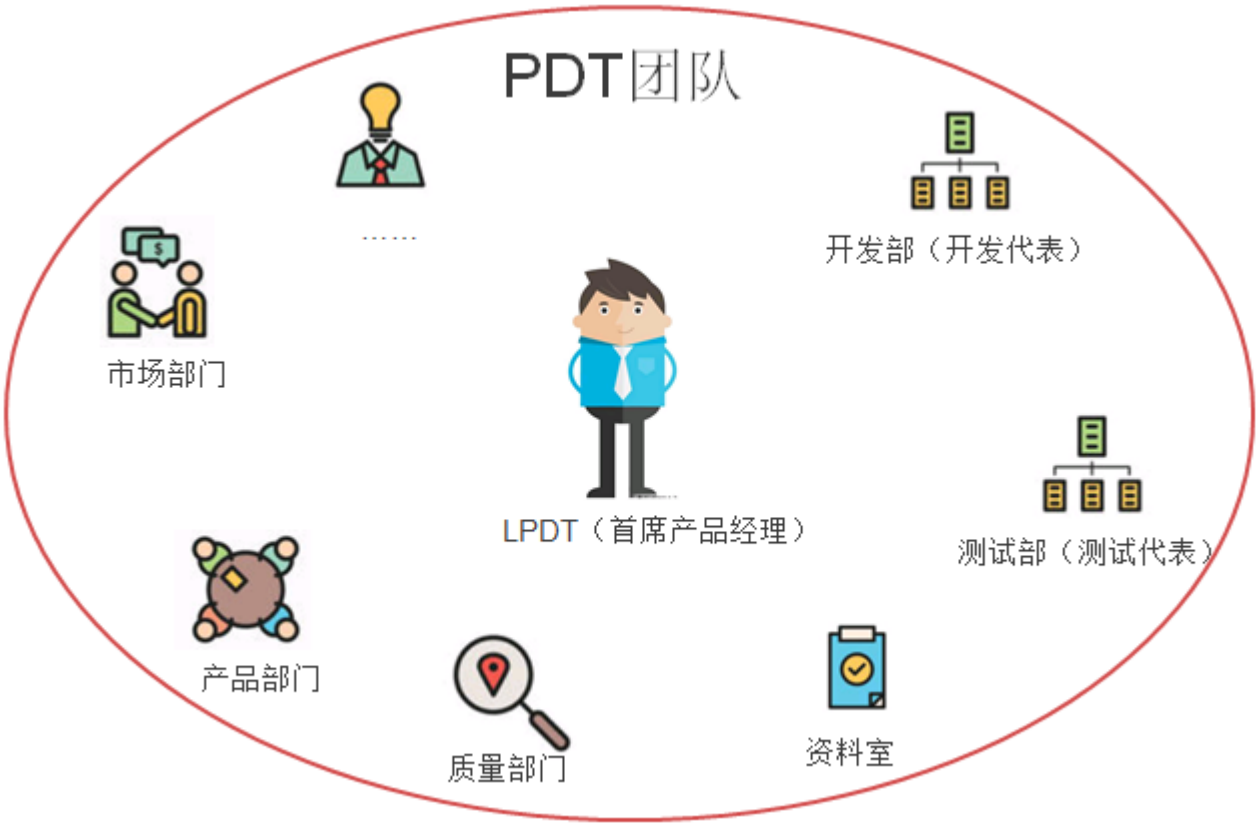
2.1 PDT团队

在IPD流程中，定义了跨部门的PDT团队：Product Development Team。

团队包括LPDT（Leader of PDT，可以理解为是各首席产品经理）、开发部、测试部、质量部、产品部、资料室、技术服务部门、市场部门等。

Scrum团队属于开发部，与Scrum团队交互较密切的团队包括：测试部、质量部、产品部、资料室等。

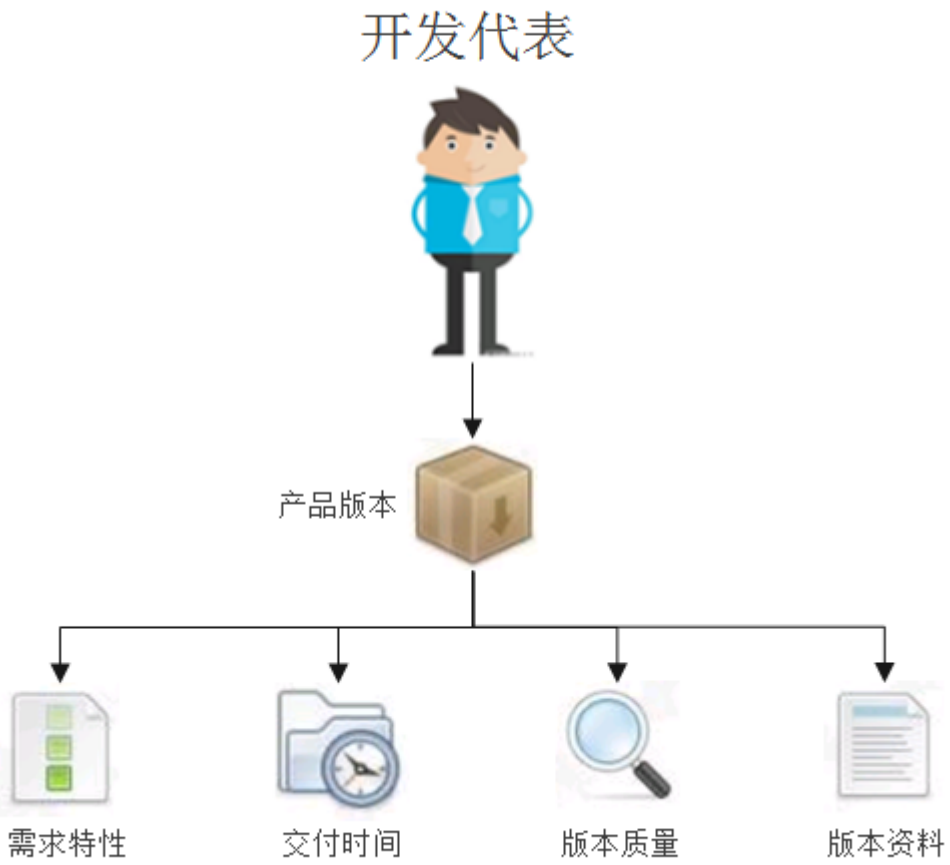
开发部在PDT团队中的负责人，也被成为“开发代表”。例如，CloudOS开发代表可以理解为是王睿总。他协调多个Scrum团队的交付工作目标，以达到PDT团队对开发交付的要求。



2.2 主要运作流程

在PDT团队中，开发代表围绕着PDT团队一起制定的大版本计划安排工作，所以在开发代表的视角看，开发团队的工作主要是各版本的按期交付，因此与版本经理工作重合度高。

围绕版本开展的工作，按关注维度主要可以分为四个要素：需求特性、时间、质量、资料。下面逐一简单介绍。



2.2.1 需求特性

PDT正常情况下，每年都会有一个大版本计划。其中最重要的一点就是，我们今年做什么。也即大版本的需求特性。

但由于软件产品和原IPD管理的硬件产品有所不同，所以在一年中，版本的需求还会根据客户和产品经理的反馈进行一定范围内增加。这部分版本计划，去年王睿总带着我们按每月一次进行规划。

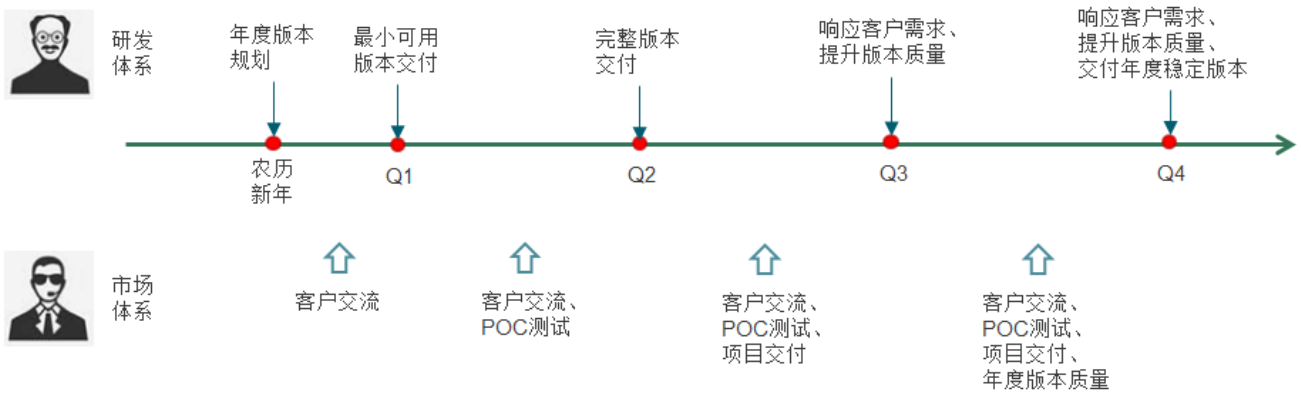
需求特性是开发团队工作的来源和价值交付的起点。实际执行中的来源主要包括：

- 年度产品规划：根据行业趋势、竞品分析、战略规划等，形成的提升产品和公司业务竞争力的关键特性。例如大的像紫光混合云，小的像应用诊断等。
- 客户项目反馈：在客户交流、竞标资料、项目测试、项目交付过程中，从市场一线反馈来的客户需求，经过分析后，形成需求特性。（这部分之前实践过每月一次版本规划）

2.2.2 交付时间

IPD流程中，对一个产品开发过程有严格的里程碑时间点的定义（例如：TR1、2、3、4、5等），并对每一个时间点的交付内容也做了明确定义。

对CloudOS而言，这个产品开发过程对应的产品，即是年度产品版本。因此，开发团队在一年中的活动，一定程度上也跟这个时间点形成共振。



大致时间点，我个人理解如下：

1、12月后到过年前，确定第二年的产品规划，也即年度版本的需求特性清单。

2、第一季度完成最小可用版本开发。

3、第二季度，完成规划的年度版本的主体功能。同时市场使用最小可用版本进行客户交流和POC测试。

4、第三季度、第四季度，持续进行客户交流与客户项目交付。同时不断完善版本功能，改善版本质量，最终形成年度稳定版本并归档。

此外，在客户项目交付过程中，会有较多的小版本交付时间点，这部分相对灵活规划处理。

2.2.3 版本质量

对于PDT项目，有专门的测试代表，对版本质量负责。除此外，对于年度版本，在归档前公司还有专门的测试中心对年度稳定版本进行验收鉴定。

与开发团队关系密切的主要是测试代表对应的测试团队。测试团队对版本的转测验收也有自己的流程规范，需开发配合提供的主要包括以下两点：

1. 特性转测验收：在版本验收前，需对较复杂的新特性进行专项，以保证特性合入版本后的版本稳定性。特性验收前，开发团队需提供测试用例自检情况表、UI性能测试报表、特性说明等相关资料。
2. 版本转测验收：版本转测验收，需开发团队提供测试用例自检情况表、版本规格表、版本说明书等相关资料。

由于每个版本需达到相应阶段的质量标准方可完成交付，因此一个版本从初次转测，到最终验收通过，可能需要经过多轮测试。

测试对版本指标的指标，可以归结为一个指标：DI值。计算公式参考：

DI:Defect Index(缺陷率)

定义：DI值是衡量软件质量的高低的指标之一。

公式：DI= 致命级别的问题个数*10+严重级别的问题个数*3+一般级别的问题个数*1+提示级别的问题个数*0.1

2.2.4 版本资料

与开发团队相关的版本资料，可以分为三类：

- 联机帮助文档：产品的联机帮助文档，由资料室主要编写，开发团队需提供产品功能的澄清和演示环境，以便资料室的同事进行产品截图。
- 版本转测文档：这部分主要由开发团队提供。在2.2.3版本质量章节中已述。
- 版本过点资料：根据IPD流程，在过TR5等关键里程碑时间节点时，除了版本外，还需要提供必要的文档资料。这部分资料，目前由开发团队提供。资料包括：产品说明书、产品技术白皮书、关键特性说明书、产品规格书等。

2.3 交付件

通过前面梳理可知，开发团队在PDT团队中的职责，就是按时保质地进行版本交付。

其中交付件主要为两类：

- 版本包：版本部署所使用的二进制文件。

- 版本资料：在2.2.4版本资料章节已述。

三、团队内部实践流程

3.1 团队运作模式

云计算特别是PaaS业务目前还处于快速变化的摸索阶段，未来的云长什么样，未来的PaaS是什么，行业都还没有统一标准。

在此背景下，意味着PaaS团队必须以敏捷的方式进行业务开发，才能快速响应市场和行业的变化。

但不同的企业有不同的管理特色和组织形态，教条化地照搬敏捷，容易水土不服，达不到预期效果。

因此，在实际的内部管理实践中，PaaS团队通过实践结合敏捷及DevOps思想，形成了一套与公司主体流程相互协作配合的有自主特色的敏捷管理模式。

下面主要介绍此模式的三个典型原则，后面章节会详细介绍各个环节的实践细节。

3.1.1 统一团队

敏捷的Scrum团队是一个独立的业务研发全功能团队。它的含义包括：

- 通过DDD思想划分业务，根据领域业务划分Scrum团队。
- Scrum团队包括了从设计到交付的全流程能力（成员），因此团队以业务成败为结果导向，形成统一的业务团队，对业务成败负责。业务成败主要两个衡量维度：①销售收入（虽然不完全由研发决定，但销售收入高是最直观的业务成功）②行业竞争力（在销售收入不那么理想的时候，再看产品的竞争力，如果产品竞争力强而销售收入不理想，可反向给销售施压）。
- 在统一团队统一目标的前提下，可通过OKR对团队进行管理，而团队内可有一定自主性，在敏捷原则下通过自组织自管理方式高效运作。

3.1.2 明确输入输出

统一的团队可以理解为是一个细分领域的制造工厂，工厂制造的原料是什么，产品是什么，需要明确且最好能量化。

这样即可通过OKR实现对团队工作的度量评估。

- 输入：

  团队的输入只有一个，就是原始需求。原始需求的来源，主要有两个：市场及客户反馈、行业竞品及行业趋势。团队通过收集原始需求，进行需求分析分解形成一个个待开发的产品需求和优先级（Product backlog）。以此作为团队工作的基础原料。

- 输出：

  团队的输出形式相对较多，但最主要的输出有两个：一是可工作的软件版本，二是配套的文档资料。团队的业务价值通过这两个主要输出得以实现。也即团队的价值交付。

3.1.3 持续交付

采用持续交付主要有两方面的诉求：

- 持续交付以接收持续反馈，加速产品演进，使产品更具竞争力。主要应用在互联网产品或其他公网线上产品，也包含公有云。
- 完整产品的实现需要较长周期，因此需要通过持续交付进行分阶段目标的过程管理。主要应用在toB项目或传统企业内部项目。

不管是哪种诉求，通过持续交付都可以降低交付风险，短时间即可看到工作价值输出。

另外，持续交付过程需要一系列自动化工具和方法论的支撑，这也是我们DevOps工具链所提供的能力。

3.2 工程工具

工欲善其事，必先利其器。在团队运作模式确定后，我们首先关注的是如何搭建与团队运作模式相匹配的基础设施。

这部分基础设施，主要包括：代码库等配置管理工具、构建集成流水线、开发联调和测试预发生产等运行环境。下面我们分别说明。

3.2.1 配置管理

配置管理主要包括代码库、依赖库、镜像仓库、制品库的管理。

目前我们使用在私有环境搭建gitlab作为开发代码库。这样可以我们的DevOps产品实现无缝对接。

依赖库目前使用Nexus，稳定性没问题，也可直接与我们的DevOps产品对接。

镜像仓库可用harbor或PaaS内置的镜像仓库。使用内置镜像仓库不需要额外配置，直接可用。使用第三方Harbor，需要进行配置。

制品库，目前业务应用可直接放在PaaS内置的镜像仓库或应用仓库。如果是大版本包，目前我们建议放在外部的ftp服务器上。

DevOps产品上有专门的页面进行这些系统的配置对接：

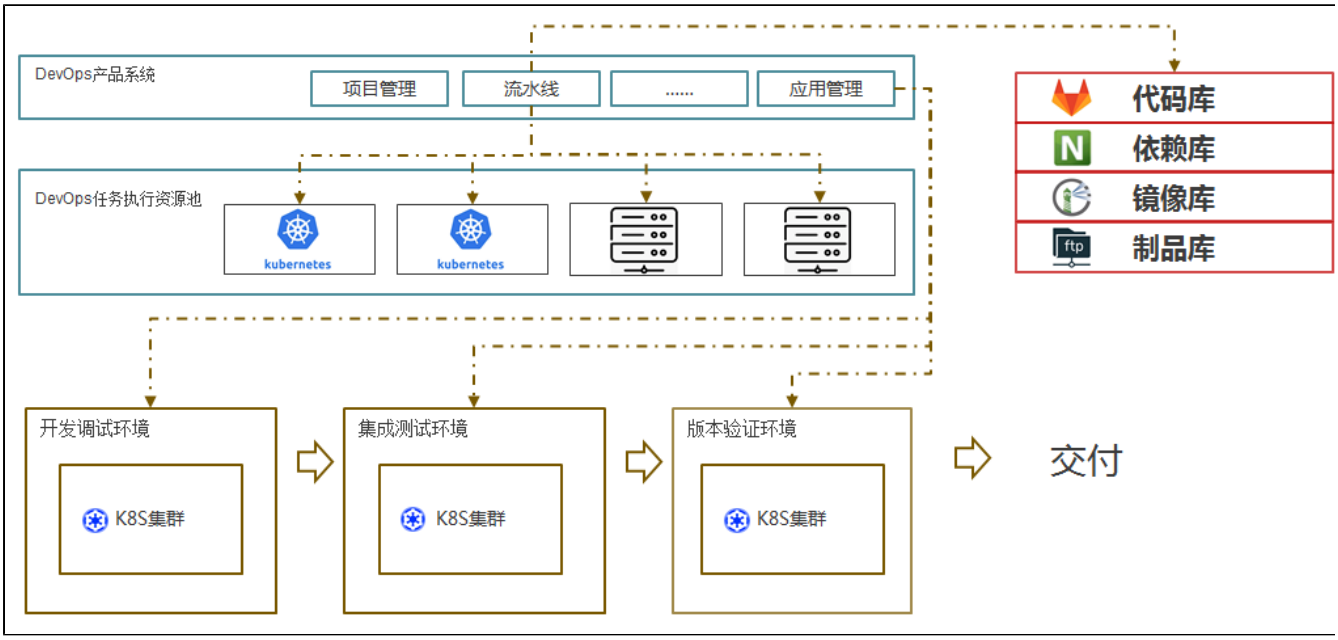
3.2.2 环境规划

业务团队要实现高效高质的敏捷持续交付，一定是需要分级分用途的环境规划。根据公有云和私有云的交付形式不同，环境规划也略有不同。

私有云的环境规划包括：

- 开发联调环境：常规环境。最好每个团队独有，用于日常开发过程中前后端联调和快速验证。
- 每日集成环境：常规环境。一个系统的多个Scrum业务团队可共用一个每日集成环境。通过自动化工具每日定时将各组件主线版本自动构建并部署到每日集成环境中，进行自动化测试用例的执行。以保证系统主线的稳定和质量。
- 版本验证环境：临时环境。针对要发布的版本包，单独准备环境进行自验证。包括自动化测试验证和基础的测试用例自验证。

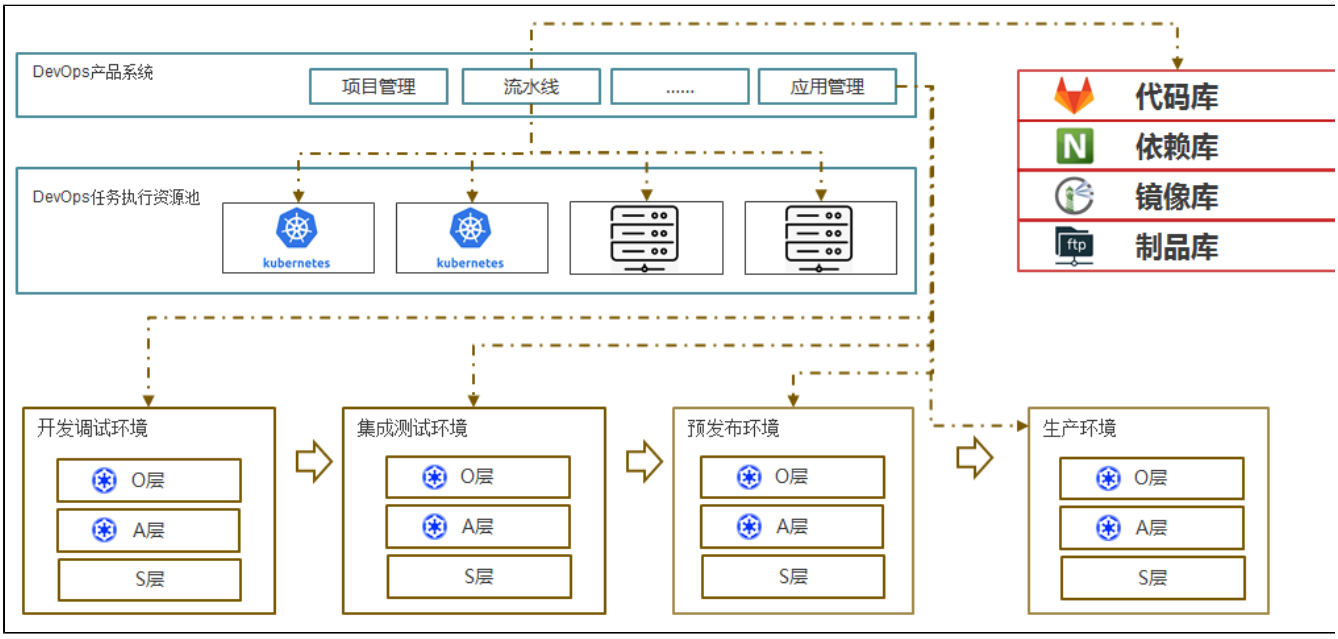
环境与交付流程的结合如下图所示：



公有云的环境规划包括：

- 开发联调环境：常规环境。最好每个团队独有，用于日常开发过程中前后端联调和快速验证。
- 每日集成环境：常规环境。一个系统的多个Scrum业务团队可共用一个每日集成环境。通过自动化工具每日定时将各组件主线版本自动构建并部署到每日集成环境中，进行自动化测试用例的执行。以保证系统主线的稳定和质量。
- 预发布环境：常规环境。针对增量的交付功能，在预发环境中进行专项测试。预发布环境通过与生产环境共享数据，可通过数据同步实现。
- 生产环境：公网线上生产环境。

环境与交付流程的结合如下图所示：



在我们的DevOps产品上，根据不同的场景，环境可以通过可用域、KaaS/CCE集群、子集群或应用命名空间进行划分。

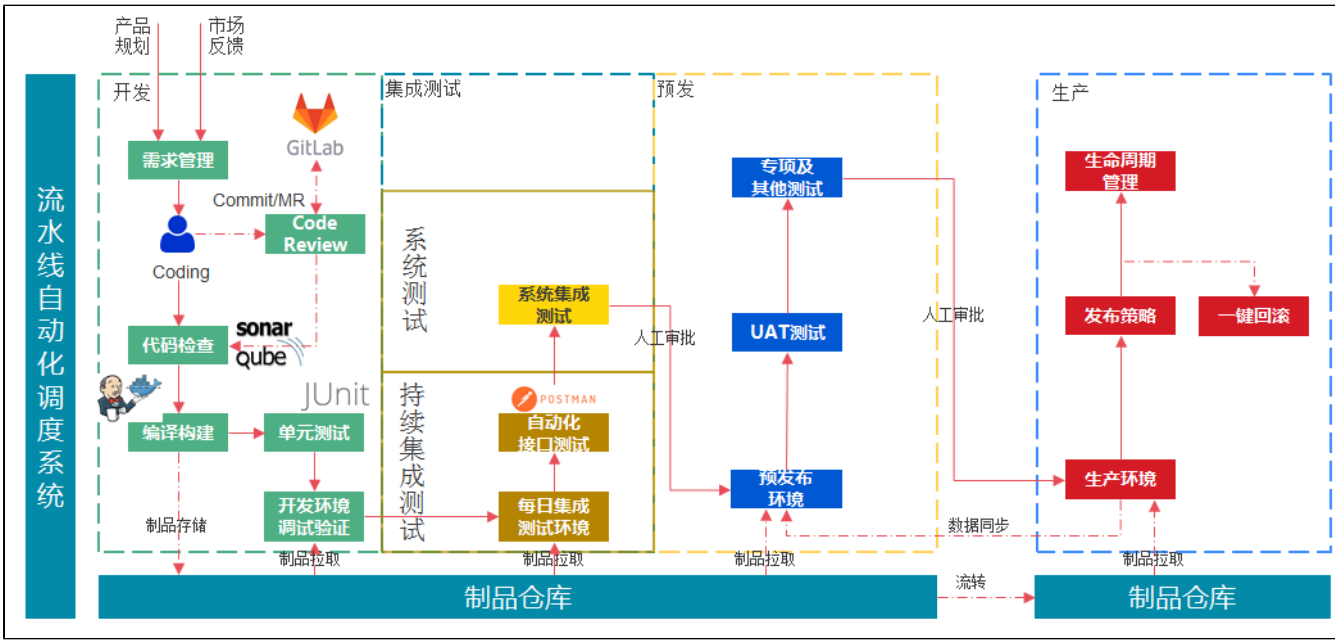
3.2.3 构建集成流水线

随着系统朝分布式微服务架构演进，加之多套环境的划分，靠人工验证和部署应用越来越低效易出错。

在这个背景下，集中式的编译构建和流水线等自动化服务，成为日常团队开发、调试、交付、发布的必需品。它一方面使DevOps流程更高效且可实际地落地，一方面保障了交付件的来源和流程可信。

特别在敏态业务的交付场景下，交付越通畅效率越高，越能使产品获得更好的市场竞争力和客户满意度。

因此，我司花重金在新川机房搭建了一套DevOps的构建集成执行资源，包括约20台物理服务器的大集群作为资源池。结合上一节中规划的多套环境，流水线可实现从代码到交付的全流程标准化和自动化。



在流水线的使用中，每个团队可参考创建如下一些流水线：

- 开发联调流水线：个人流水线，可根据开发者开发情况制定feature会bugfix分支。开发提交代码后自动触发编译构建，并直接部署到开发联调环境。以便开发者能快速联调验证。
- 每日集成流水线：团队流水线，负责每日将团队各组件的主线分支（或版本分支）自动编译构建并部署到每日集成环境，并执行代码检查、单元测试、自动化测试等自动化质量验证，以保证主线分支或版本的集成稳定性。
- 版本发布流水线：团队流水线，负责私有云版本的编译构建（Helm包）。
- 持续交付流水线：团队流水线，负责公有云场景下，通过流水线跨多环境实现自动化的持续交付过程。

3.3 需求管理与用户故事Product Backlog

介绍完工程工具的使用，我们开始进入团队内部的流程管理部分。首先，介绍团队的输入项——需求的管理。

需求是团队外部大流程对团队的主要输入，也是团队工作的起点。我们提倡，无需求不开发，所有的代码提交都应该与需求相对应。（缺陷解决的代码需与缺陷相对应）

3.3.1 需求收集

前面介绍到，Scrum团队的需求来源主要有两个，一是市场及客户反馈，二是行业竞品及行业趋势：

- 市场及客户反馈类的需求，基本从开发代表（例如王睿总）、市场代表（产品SE，例如龙飞老师）或需求经理（例如母佳璐）处反馈给团队的PO。
- 行业竞品及行业趋势，则由团队PO与PDT团队中负责产品定义的领导和专家（（例如阮总、计老师等））共同讨论确定，并由团队PO带回团队实施。

需要注意两点：

- 需求的收集不是一次性的，而是日常性的。才能推动产品不断提升竞争力、不断演进。
- 这里的需求往往是一句话的原始需求，在收集后还需要对原始需求进行分析分解，最终的实现方案可能与原始需求并不完全等同。

原始需求由需求经理记录在newdjdam产品类型项目中，同时也可精简为类似如下的表格，以便PDT团队跟踪管理：

需求描述	详细说明	需求来源	反馈人	预期交付时间	当前状态及进度	备注

3.3.2 需求分析

PO拿到的需求收集的原始需求往往是一句话需求，只有基础的描述和预期交付时间。并且客户在反馈需求时，往往把自己想象中的解决方案当做需求。

因此，PO拿到原始需求的第一件工作不是设计实现，而是需求分析。

① 需求分析第一步，PO应该弄明白，什么样的客户（用户），在什么场景下，遇到什么问题，需要什么办法（特性）解决。

特别是当原始需求是客户自己给出的解决方案时，更要找出场景和问题，从而可能找出更好的解决方案，而不是直接按照客户的说法开始开发。

这时，不是一个人闷着头想。而应该多沟通多调研：

- 对于市场客户反馈需求，应主动联系市场接口人，有机会应争取直接与客户沟通。需要明确，客户的角色、场景和遇到的问题。
- 对于竞品分析和行业趋势总结出的需求，应该多进行竞品细节分析和行业专家交流，以便深刻领会需求的内在价值和逻辑，而不是照抄功能。

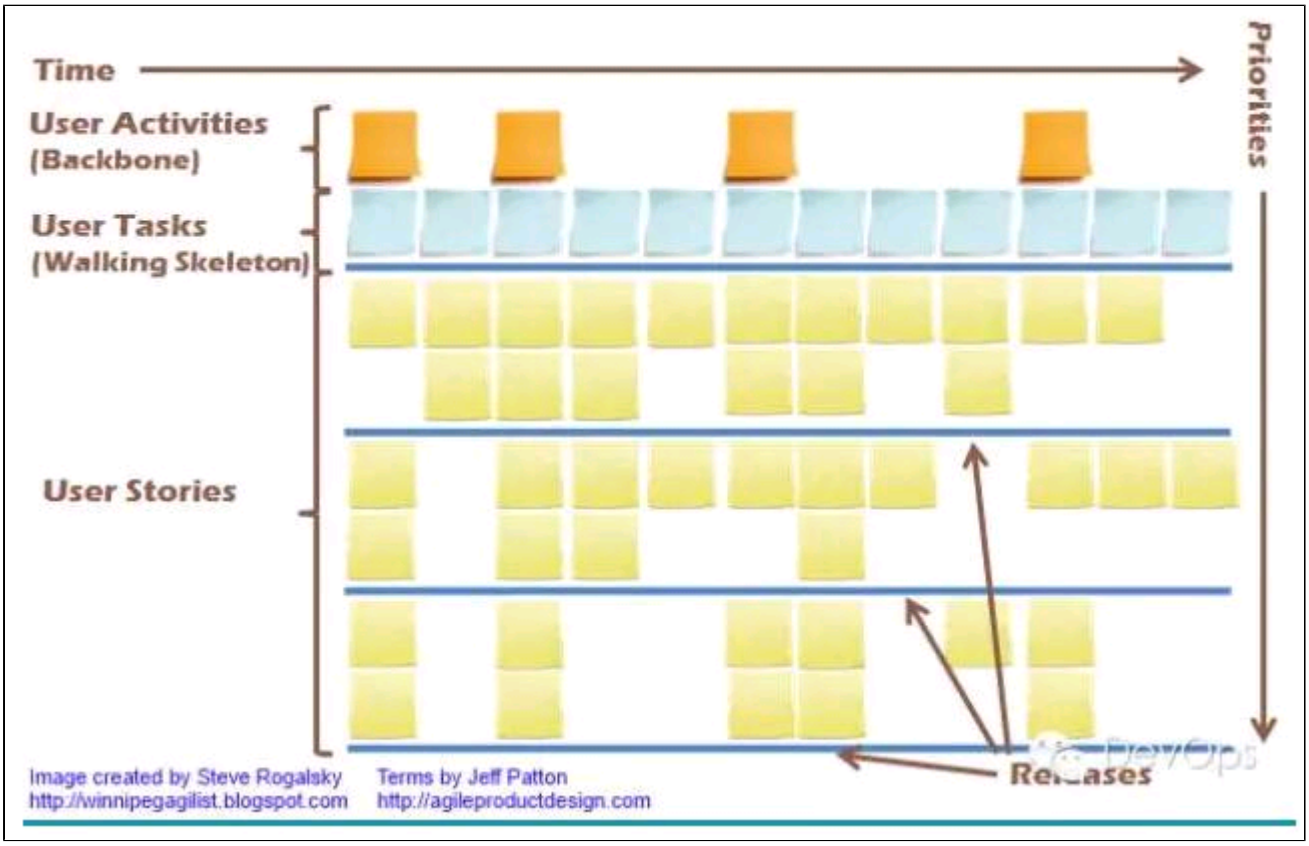
② 需求分析第二步，PO在明白了需求的场景、角色、问题和解决方案后，应整理为用户故事。

标准的用户故事描述为：*“作为<用户角色>，我希望<功能特性>，以便我能<实现价值>”*。

在实际描述中，我们不用僵化地使用此描述，但仍需要在用户故事的描述中，讲清楚用户角色、功能特性和实现的客户价值。

③ 需求分析第三步，PO将原始需求整理为用户故事后，通常会发现规模较大，这时就需要对用户故事进行拆解。

用户故事的拆解，一定要以用户能理解的维度进行拆解，不能按技术维度。比较便捷常用的方法，可以参看用户故事地图，根据用户旅程，先找出主要用户活动（User Activities），再分解为用户任务（User Tasks），最终分解为用户故事（User Stories）。



④ 需求分析第四步，初步评估每个用户故事的工作量规模（故事点）、价值规模（价值点）以及紧急程度，以便客观定量地确定各用户故事的优先级。

故事点的评估，行业内有很多不同的说法，为了便于实操，我们就统一为工作量。我司习惯以人天计算工作量，我们可以以一个故事点=一人天进行换算。

价值点的评估，来源于IBM的敏捷实践，用于衡量一个用户故事对于客户和产品价值的高低。价值点越高，则表示价值越高。我们可以1~100点进行量化。

紧急程度，是toB项目所独有的。由于不同项目不同客户有不同的交付截止时间，根据交付截止时间的远近程度，可形成紧急程度的度量。可以1~10点量化，点数越高表示越紧急。

3.3.3 product backlog与优先级

对收集的需求进行分析后，会形成一个汇总的用户故事列表（Product Backlog）。这张表由PO日常维护，会经常进行增删调整，是一个动态的需求列表。

Product Backlog中重要的一点是需要对所有用户故事进行统一的优先级排序，以便作为迭代计划开展的依据。

我们通过对用户故事工作量规模、价值规模和紧急程度的量化，优先级可以自动得出推荐值，PO再根据实际情况进行调整。

优先级的计算，可参考如下公式：

```
优先级=价值点*紧急程度/故事点
```

优先级的值越大，则优先级越高。

product backlog的基础表格可参考如下：

优先级	需求描述	状态	目标用户	用户价值	故事点	价值点	紧急度	计划交付时间	验收条件

其中验收条件，可在迭代计划时，与团队一起确定。

3.4 需求澄清与迭代计划

3.4.1 敏捷迭代与里程碑对齐

敏捷迭代一般是持续进行的，每个迭代选取backlog中最高优先级进行下一个迭代的计划。但在IPD流程中，项目的关键里程碑时间点是确定的。

因此，在有了Product backlog之后，我们需要初步根据里程碑时间点和backlog规模，确定整个里程碑版本的工作范围和迭代计划。

具体操作参考：

① 根据团队每个迭代能完成的故事点数，划分迭代。

② 根据里程碑时间点，确认上述迭代划分是否可完成里程碑版本的交付。

③ 如两者不能对齐，则调整Product backlog，去掉部分非核心功能和低优先级需求，或调低验收条件，以使迭代规划能满足里程碑交付。

3.4.2 需求设计与澄清

由团队内具备业务设计能力的成员，根据用户故事，实现业务设计。设计内容可能包括：

- 实现业务所需的技术架构和技术选型
- 业务原型

在需求设计完成后，需要整个团队，再加上PDT团队相关成员参加需求的澄清会（评审会）。PDT团队参与的成员，主要包括测试代表指派测试人员、资料室相关人员、用户体验UED成员等。

需注意两点：

- 需求设计和澄清是团队工作不可或缺的一部分，因此也应该排入迭代计划中（作为任务）。
- 需求设计任务也需要按需求优先级安排，并可排一定的提前量，以便后续开发计划能按正常时间开展。也即，用户故事拆分的设计任务，可以提前开始进行。

3.4.3 迭代计划活动一——确定迭代目标和交付范围（sprint backlog）

每个迭代计划之初，都应该确定这个迭代的主要交付目标。这样在迭代过程中，遇到变动，细节可以调整，整体目标尽量保持不变。

交付范围的确定，主要依据Product backlog，从最高优先级往下选取，使迭代交付的故事点基本与团队每个迭代的工作量规模相等（或一定比例系数）。

3.4.4 迭代计划活动二——确认验收条件和分解任务

在确定迭代目标和交付范围（sprint backlog）后，需要团队成员一起对本迭代要交付的用户故事进行任务分解。

注意用户故事的拆解，是以用户可理解的维度进行拆解，不管如何拆都是有用户价值的。而任务分解，则是将用户故事按照团队成员的任务进行分解，目的是便于团队成员执行。

在任务分解时，PO应和大家共同确定验收条件。这样才能将任务分解得更准确。

3.4.5 迭代计划活动三——确定责任人

在完成任务分解和验收条件确定后，就可以进行迭代计划的最后一步：确定责任人。

责任人的确定上，敏捷主张自主选择，实际操作中往往是自然形成。但有一点需要注意，就是团队各个成员应任务均衡，且每个人的工作量不应超过迭代时长（或超过一定比例内）。

补充说明：迭代计划会的结果就是形成团队认可、目标和任务清晰、责任人明确的迭代计划。以上活动均是围绕这个目标的一些实践活动。

实践活动的细节不用过于拘泥，只要能达成目标就好。如果在活动中有执行不顺畅或者后续执行发现计划存偏差等问题，可在迭代回顾时团队一起讨论改进。

另需注意：迭代计划会，除了计划本身，还是一个仪式感。是团队所有成员一起制定迭代目标的仪式，团队成员对迭代最终的交付做出承诺，同时PO也应该对需求的价值和计划的稳定性做承诺，在一定程度上顶住外部压力，以便团队能摒除干扰全力冲刺。

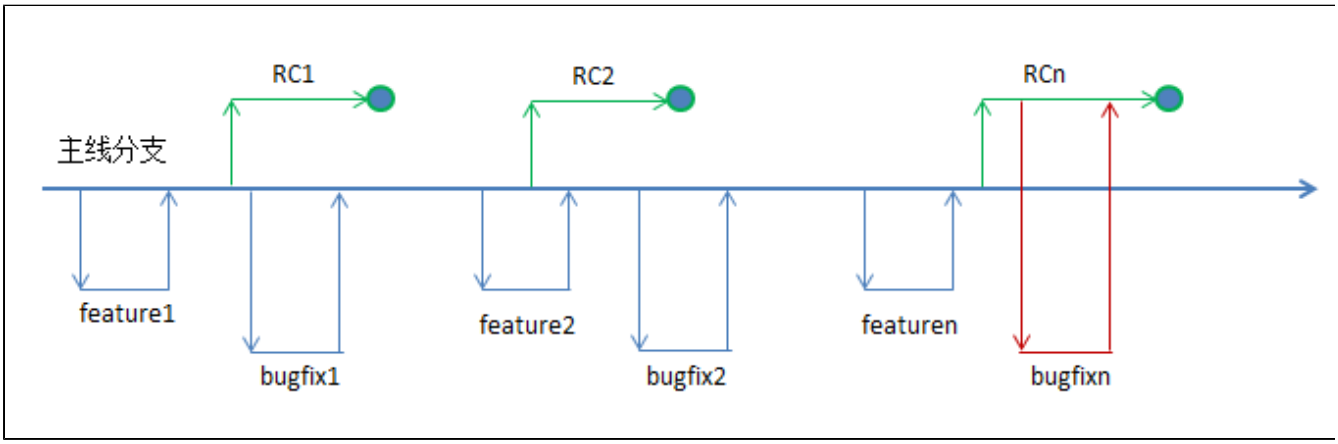
3.5 迭代过程管理

在迭代计划会完成后，迭代就正式开始了。在迭代的过程中，也有一些实践细节可以帮助团队更好地进行迭代冲刺，以促进迭代交付的顺利达成。下面主要介绍这部分。

3.5.1 代码分支模型

分支规划与PDT的版本管理密切相关，经过行业通用实践和我们自己长时间的实践经验，最后形成了目前的分支模式，以配合PDT版本发布和迭代的持续交付开发。

具体如下图所示：



主要包括主线分支、版本分支、feature分支和bugfix分支。

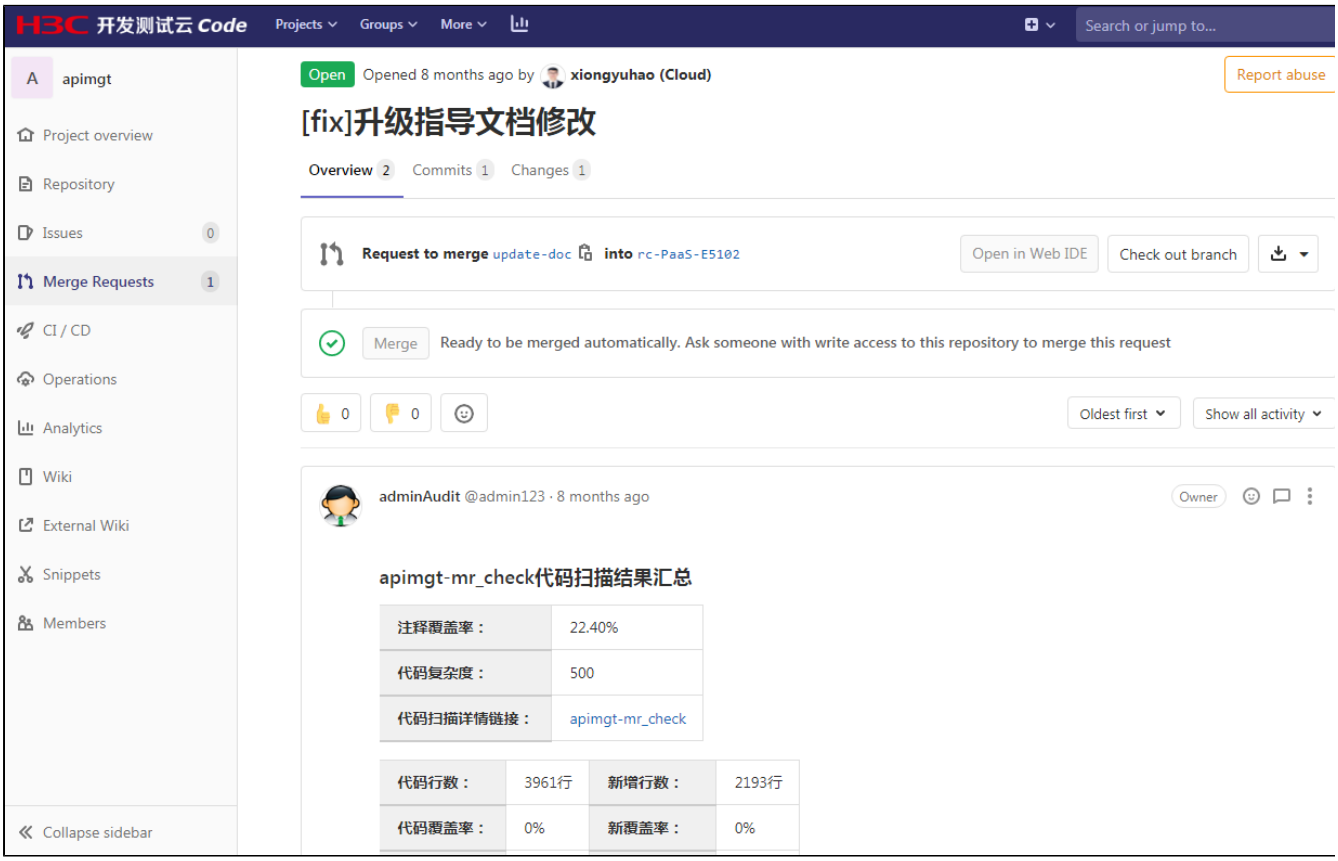
- 主线分支：长期分支，主线版本持续演进。需保证主线分支持续稳定，因此每日集成应该从此分支进行集成验证。
- 版本分支（RC）：版本分支为临时分支。当需要出版本时，从主线分支拉出。如版本分支在PDT团队的测试中发现问题或网上反馈问题需要解决，应该同时合入该版本分支和主线分支。（在主线不够稳定的情况下，也可以再发版过程中冻结主线特性合入，专门解决缺陷。但这不利于持续交付的执行，是特殊的操作方式）。
- feature分支：临时分支，用于新特性开发的分支。实际执行过程中，我们应将功能相对独立的一个或一组用户故事作为一个特性。在迭代交付时，完成feature的交付。交付时feature分支代码通过发起merge request的方式合入主线。代码合入后，该临时分支即可删除。
- bugfix分支：临时分支，用于缺陷修复。正常情况下，bugfix分支也同feature分支类似，在交付时通过merge request方式合入主线，然后删除。但一些版本问题的缺陷修复，在bugfix分支合入主线的同时，还需同步到版本分支。

因此可知，当迭代开始时，团队开发成员应先拉出特性分支，然后开始开发。

3.5.2 Code Review

代码评审是保障代码质量的第一道关卡。目前我们的做法是，在feature分支或bugfix分支向主线发起merge request时，进行Code Review。

在工具使用上，我们基于gitlab工具，发起merge request后，需要团队内两人以上对代码评价OK后，方可合入主线。



3.5.3 每日站会与风险管控

每日站会是敏捷迭代过程中，一个最典型的活动。也是行业内大部分实行敏捷的团队都采用的敏捷实践活动。站会如何开展就不再赘述，主要强调两个思想：

- 每日站会不是汇报会，而是团队全员参与的信息交流会。团队每个成员都应该关注，了解团队本迭代的整体情况，一起出谋划策和补位。
- 站会中对于PO和Scrum Master而言，则要重点关注是否存在阻塞问题和风险点。及时识别这些问题并在自己能力范围内或调动更高层面协调解决，将极大地提高团队的战斗力和凝聚力。

3.5.4 需求变动

提到敏捷，管理层最喜欢的一句话就是“拥抱变化”。敏捷相较瀑布模型而言，确实对需求的变化更为友好。但敏捷的“拥抱变化”绝不是需求的随意变动的理由。

对于敏捷团队而言，需求的变动不可避免，但如何管控需求的变动则是需要注意的。

- PO应充分理解产品方向，在产品大方向上保持定力，避免产品需求随意变动。（随意变动很重要的原因是产品方向不清晰）。
- 针对未预见的市场变化或客户反馈，PO的第一原则是先分析。尽量放入下一个迭代，而保证当前迭代的稳定。（冲刺中的团队被打断，非常影响效率）
- 针对突发紧急情况，必须第一时间响应交付的情况。则原则上应相应移出本迭代同等工作量的任务，或推迟本迭代。（应尽量减少这种突发情况的出现）

3.6 需求完成与任务完成

前面我们明确了迭代中的需求即拆分的用户故事，它是以用户业务价值划分的。而任务则是按执行者的工作内容划分的。

因此在迭代过程中，如何认定一个任务为完成，和一个需求被认定为完成。有不同的标准。

简单来说，任务的完成只要能保证后续任务可以继续执行即可。不需要确定严格的验收条件。

而需求在迭代计划时，就应该明确定义需求的验收条件。因此需求的完成也应该以验收条件完全满足为准。不能只是开发完成。

3.7 迭代演示与迭代回顾

迭代演示和迭代回顾，经常是被忽略的两个敏捷实践活动。但由于没有这两个活动，会导致产品和项目的严重问题。因此建议需要实践起来。（或者定期开展而不是每个迭代开展，或与下个迭代的计划会合并）

3.7.1 迭代回顾

由于内部的运作模式团队自己有较大的自主性。而所有敏捷知识都是原则性的。

因此为了团队运作更为流畅，更符合团队的实际情况，“敏捷流程”不应该是固化的。而应该通过迭代回顾会，找出运行不流畅的瓶颈点，坚持做得好的，改善做得不好的。持续地进行优化改进。

因此迭代回顾会，一般会要求大家发言说出自己认为本迭代团队做得好的是什么，不好的是什么。（也可以通过不记名纸条的方式）并一起讨论下个迭代怎么才能做得更好。

建议迭代回顾会可以与下一个迭代的迭代计划会合并。

3.7.2 迭代演示

迭代演示，顾名思义，就是将本迭代交付的内容拿出来给PO和周边相关人（例如产品经理、资料、测试团队。或可考虑包括市场相关接口人等）进行产品功能的演示。

对于团队而言，这又是敏捷迭代中的另一个仪式感。只有当团队成员知道交付完成立马要给大家演示时，才会更踏实认真地去最求完整交付。

另一方面，对于周边相关人甚至是PO而言，也是了解产品实现进展的第一渠道（方便大家做到心中有数）。同时如果实现与预期不符，也能第一时间发现并纠正。

3.8 PDT版本发布与缺陷解决

与传统敏捷不同的是，我们的Scrum团队流程还需对接PDT团队的IPD流程。主要结合点在团队的输入和输出上。输入是需求，前面已经详细说明了。

剩下就是输出。

Scrum对于PDT团队的输出，最终落在版本的交付上。而私有云的版本交付过程涉及到Scrum团队外的重流程。

主要工作包括资料准备、版本的系统集成自检、团队团队反馈的缺陷修复。

建议的处理方式是，将资料准备、版本自检作为版本任务直接放入迭代计划中，一并执行。缺陷作为累积缺陷（或技术债），根据严重程度和优先级也直接放入迭代计划中。

考虑到这一因素，我们在迭代计划会时，需要先审视两个问题：

- 下个迭代是否有版本发布计划。如果有，则需要先排入版本任务。并适当减少需求的交付规模。
- 累积缺陷是否影响最近的版本发布或导致质量不可控的风险。如果是，则应根据实际情况排入需高优先级解决的缺陷，并适当减少需求的交付规模。