CloudEasy CAP

产品手册

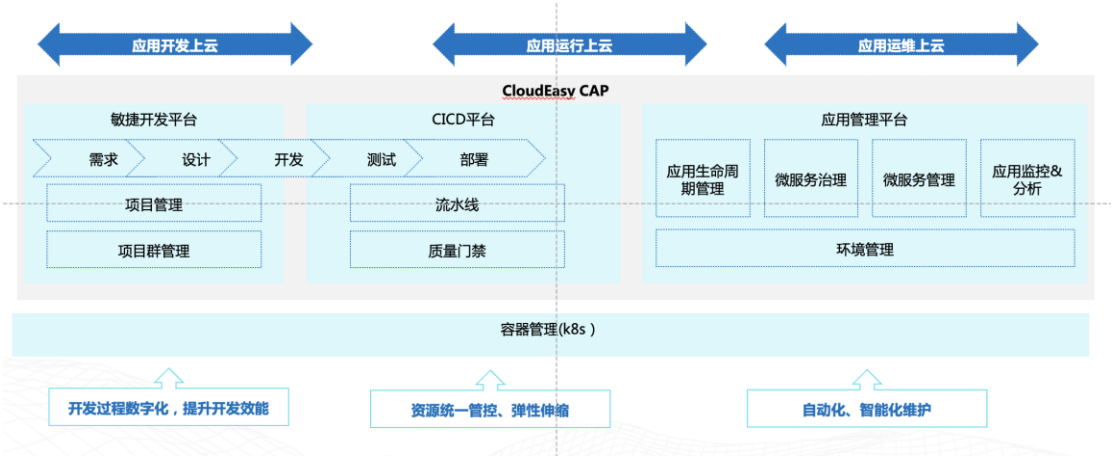
产品版本：V100R001C00　　文档版本：V1.0

目 录

1 总体技术方案说明

1.1 整体架构

CloudEasy CAP 是一款面向云原生时代的一站式 DevOps 平台，，涵盖软件研发从构想、开发、测试到交付的全过程。通过统一工具链，落地云原生新技术和研发新模式，实现文化、流程、工具的协同改进和提升，助力企业创新和数字化转型，快速实现软件研发敏捷和组织敏捷，实现快速、持续、高质量交付，打造高绩效组织，提升组织效能。整体架构下图所示。

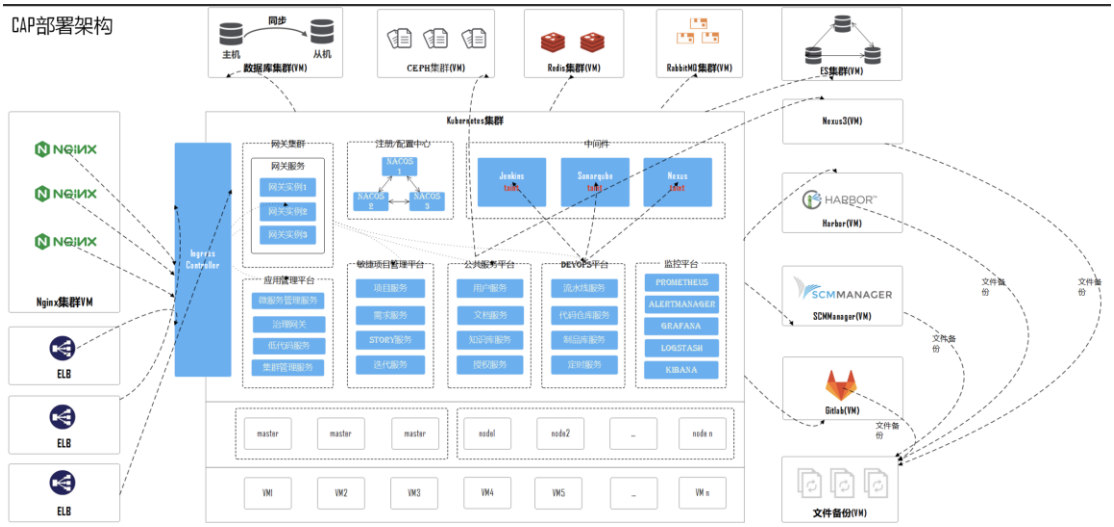


CloudEasy CAP 主要由以下三个子平台构成：

1. 应用开发平台：提供基于 safe 规模化敏捷的项目群管理、项目管理、PI 计划、精益迭代看板、知识库等功能，助力规模化敏捷转型。
2. CICD 平台：提供应用从需求、构建、部署、测试、管理到上线发布的一体化 DevOps 作业平台，帮助用户实现持续交付，提升研发效能
3. 应用管理平台：为新老应用提供统一的运行管理、监控、治理的平台，最终提升业务上线后的稳定性、可靠性和 SLA

1.2 部署架构

CloudEasy CAP 支持部署在公有云、私有云和混合云，适应云时代的各种部署场景。部署架构如下图所示。



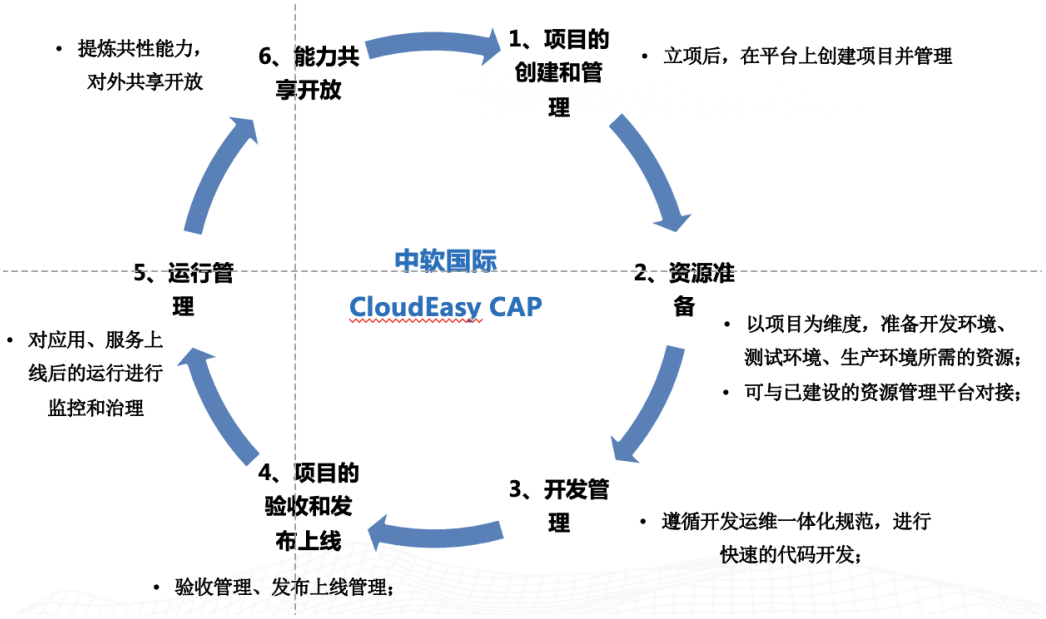
系统采用前后端分离的方式部署，具体方式如下：

1、前端部署方案：前端的 VUE 开发的界面，经过 WebPackage 构建后通过 CDN 的方式部署，通过前端版本监控组件检测前端代码版本，达到使用客户无感知更新发布。

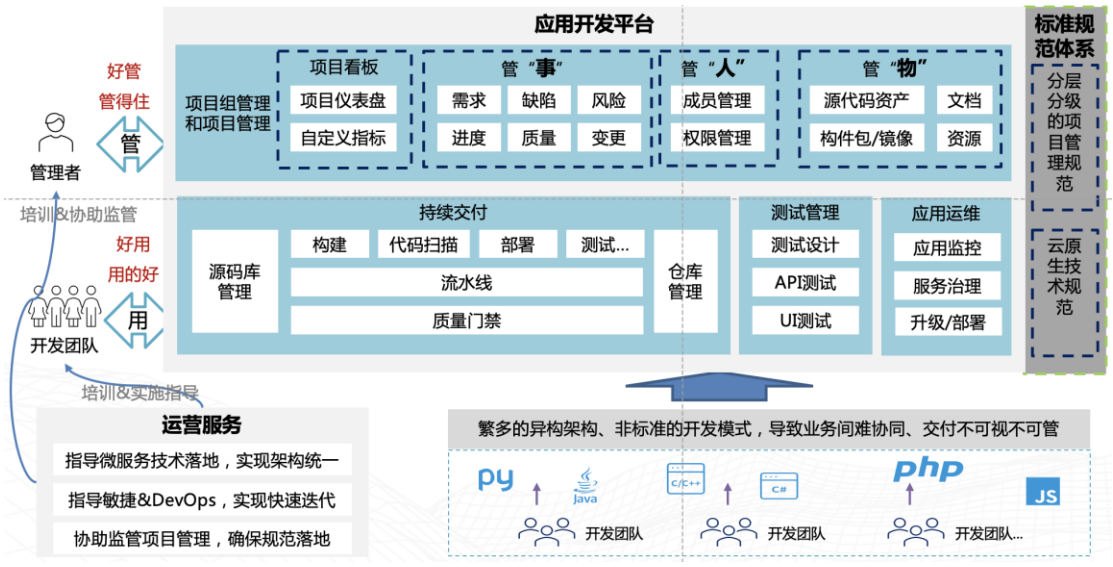
2、后端部署方案：后台服务通过 Docker 镜像的方式推送镜像到镜像服务器。

1.3 典型应用场景

为了更好的服务政企机构数字化转型，提供更全栈的数字化转型路径，中软国际基于云原生技术提供完整闭环的能力模型：



CloudEasy CAP 有效支撑了软件项目开发管理流程，涵盖了从项目启动、需求分析、系统设计到开发、运维全过程。整体业务场景说明如下。



CloudEasy CAP 主要实现了开发运维一体化流程。开发维一体化是一个研发组织中，人员使用频率最高，技术标准多，管理难度最大的环节。通过 CAP 将数量繁多，标准不一的工具集成整合到一起，使得研发流程，人员技能、操作规范及质量标准达到高度统一。从而有效的规范软件开发的流程，提高软件开发的质量，把控软件开发的时间周期和软件代码的管控能力，加强了技术架构的实施可行性。

同时结合政企开发自身特点，以及业界项目管理标准，建立系统的、结构合理、定义清晰的标准规范体系，与 CAP 工具平台相铺相成，提升软件开发项目的整体效率。

最后，通过运营服务进行培训和实操演练，有助于工具、流程和规范在项目中快速平稳落地。

2 敏捷开发平台

CloudEasy CAP 敏捷开发平台结合 SAFe、Scrum、用户故事地图、KANBAN 等理念，结合中软国际多年软件研发的优秀实践，帮助软件开发者聚焦价值，响应变化，“小步快跑”，快速获取反馈，加强团队协作，助力精细化管理，实现持续地、快速地、高质量地交付价值。

2.1 平台概述

敏捷开发平台为研发团队提供敏捷项目管理与协作，包括项目群与项目管理、需求管理、缺陷跟踪、计划管理、进度管理、风险管理、文档管理、Wiki 协作、统计报表等功能。敏捷开发平台结合 SAFe、DevOps、Scrum、用户故事地图、KANBAN 等先进管理理念：

- SAFe

敏捷开发平台通过项目群管理版本火车的愿景、路线图、团队、CoP 等，通过项目管理每个 PI 的具体研发过程，并提供项目群层级和项目层级的两层度量，从而实现 SAFe 规模化敏捷。

- DevOps

  CAP 秉承 DevOps 的流动、反馈和持续学习与实验的三原则，通过用户故事地图、多维度迭代看板、自动化测试、流水线、微服务等实践，实现过程可视化、自动化、可度量，实现产品增量快速、高质量、持续地流向客户。

- Scrum

  Scrum 是敏捷开发的主流方法，通过迭代冲刺的方式，持续交付，从用户需求到用户反馈实现软件开发闭环。在 Scrum 过程中，需遵循 3355 原则即：

  涉及三个核心角色：PO、SM 和开发团队；

  输出三个工件：产品待办列表、迭代待办列表和产品增量；

  执行五个事件：迭代计划会议、迭代冲刺、每日站会、迭代评审会议、迭代回顾会议。

  在整个过程中，所有参与人应遵循“勇气、开放、专注、承诺、尊重”的五大价值观。

- 用户故事地图

  用户故事地图是采用讲故事的方式，将产品使用过程中的用户体验图形化和可视化的一种方法，能在需求拆分过程中保持产品和用户体验的全景图。全景图可以帮助团队和用户有效地沟通，帮助团队避免开发非必要的特性，也为一致的用户体验提供了基准。

- KANBAN

  KANBAN 采用拉动式系统，通过可视化工作流、显示化流转规则、限制在制品数量、度量和管理流动、协同改进等特性，达到准时生产（JIT）的目的，可以聚焦客户价值，快速发现并解决问题。

CAP 敏捷开发平台具有如下特点：

- 支持项目群和项目分层管理

  项目群负责统筹与规划，项目负责跟踪与实施，保障多团队的目标和运作机制一致，实现产品统一管理，资源统一调配，团队间分工协作，左右拉通，同时又相互独立的目的。

- 支持全过程、多维度可视化。

  通过项目群概览，查看产品总体情况；通过待办事项列表、故事地图、迭代看板，查看需求在不同的信息、进展、风险、阻碍等情况；通过项目群和项目不同层面的统计报表，查看敏捷交付、质量内建、人力成本、团队能效的统计报表。

- 提供丰富的沟通协作工具，提高协作效率。

  工作项的信息、附件、评论，可以传递和沟通关于工作项本身的价值、目的、实现方式、进展、风险等信息。项目群和项目的知识库可以记录和传承经验文档、

技术文档、需求相关文档、会议纪要等。当事件发生时，系统可以自动发送相关信息通知到相关的人员，提升沟通效率。

- 支持灵活的自定义功能，满足不同项目的不同管理需求。

  包括自定义工作项模板和工作流，自定义统计报表，自定义工作项显示字段，自定义角色和权限，自定义通知策略。

- 多维度管理工作项

  敏捷开发平台采用 Epic-Fearue-Story 需求模型，结合开发任务（Task）和产品缺陷（Bug），形成工作项模型：Epic-Fearue-Story-Task/Bug，便于最小化、标准化管理需求和缺陷。

  管理工作项时，支持采用树形结构、用户故事地图、看板维度进行管理，也支持项目和项目群分层管理。

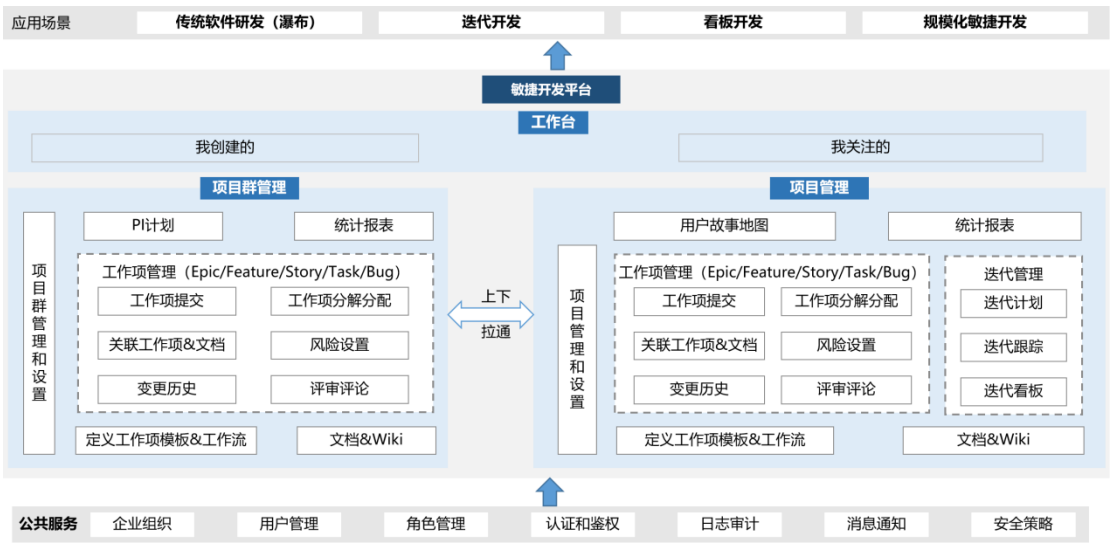
2.2 功能架构

敏捷开发平台在组织架构、用户、权限、安全等公共服务的基础上，实现三大部分的功能：

- 项目群管理：主要实现统筹与规划。
- 项目管理：主要实现跟踪与实施。
- 工作台：便于快速找到需要的内容。

应用于多种研发场景：

- 传统软件研发，即瀑布式研发
- 迭代开发
- 看板开发
- 规模化敏捷开发

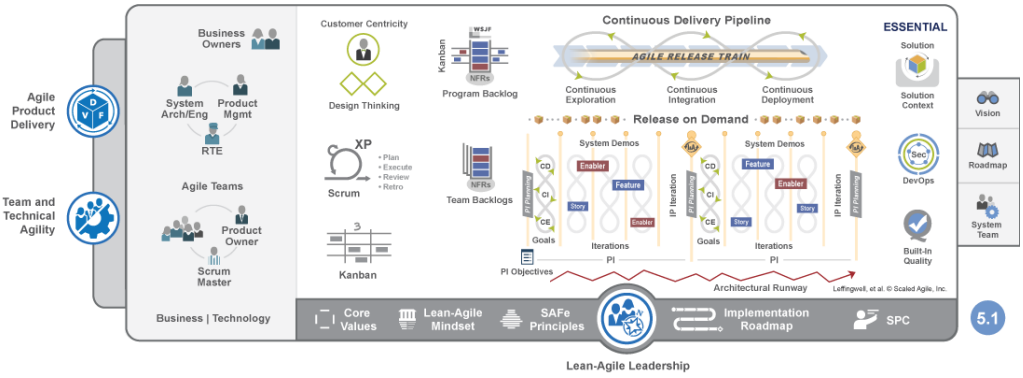


2.3 功能描述

2.3.1 项目群

一般情况下，敏捷讲的是一个敏捷团队（一般 3~9 人）或两三个敏捷团队的敏捷。为了满足更大的企业及其开发大型系统的需要，同时发挥敏捷的优势，并且利用基于系统思考和精益产品开发的相关知识，人们开发出了大规模敏捷框架，用于管理大规模场景下的多团队管理。SAFe 就是其中常用的一种框架。

SAFe 是可扩展和可配置的框架，SAFe 5 中基本型框架如下图所示。关于其它类型的 SAFe 框架和 SAFe 的详细描述请参见 SAFe 官网。



在 SAFe 中，一个投资策略由一列敏捷版本火车（Agile Release Train，ART）承载。一个企业级投资策略可以由多个敏捷版本火车组成。SAFe 采用 PI（Program Increments）对一列 ART 的提交和发布时间进行总体规划。每个 PI 为一个发布版本。每个 PI 的增量由多个敏捷团队经过多个迭代开发完成。PI 级团队由 System and Solution Architest/Engineering、Product and Solution Management、Release Train Engineer 组成。迭代级团队由敏捷团队组成，包括全功能开发团队、Product Owner 和 Scrum Master。

项目群管理秉承 SAFe（Scaled Agile Framework）规模化敏捷理念设计，帮助大规模的开发团队实施敏捷开发。

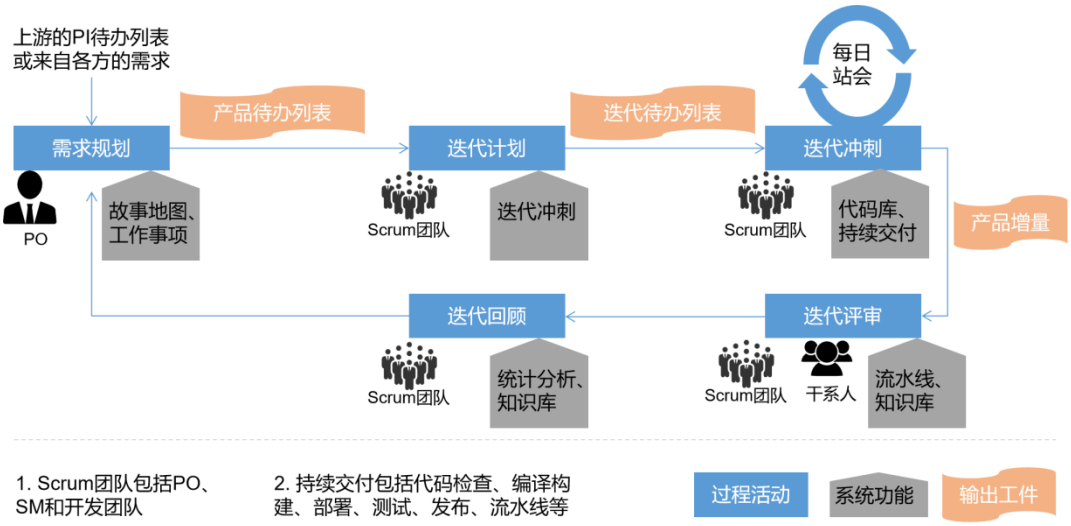
在系统中，项目群用于统一管理产品所有工作项，通过 PI 路线和子项目协调管理多个团队，同时提供项目群级的统计报表、知识库管理。项目也可以统一管理产品所有工作项（同时可以从项目群中同步工作项）、根据需求进行迭代开发、持续交付、运营&运维。一个项目群可以包含多个子项目。

2.3.2 项目

项目为一个敏捷开发团队提供敏捷项目管理和协作服务。

CAP 的项目管理秉承严谨的 Scrum 敏捷框架和实践，为软件开发团队提供敏捷项目管理与协作，支持项目自身管理、需求规划和管理、缺陷管理、迭代管理、自定义工作项模板和字段、自定义工作流、多维度看板、报表统计分析、项目文档管理等特性，支持从需求到交付端到端的可视化、透明化、高效率地管理项目。

Scrum 是敏捷开发的主流方法，通过迭代冲刺的方式，持续交付，从用户需求到用户反馈实现每一个闭环的软件开发过程。在 Scrum 过程中，涉及三个核心角色：PO、SM 和开发团队；输出三个工件：产品待办列表、迭代待办列表和产品增量；执行五个事件：迭代计划会议、迭代冲刺、每日站会、迭代评审会议、迭代回顾会议。在整个过程中，所有参与人应遵循“勇气、开放、专注、承诺、尊重”的五大价值观。



2.3.3 工作台

工作台支持查看我创建的、我关注的项目\项目群，支持采用卡片形式或列表形式显示。

3 CICD 平台

在 DevOps 实践中，持续集成/持续交付，也就是 CI/CD，是非常重要的。部署流水线确保代码和基础设施始终处于可部署状态，可根据需求将代码持续地、安全地部署到环境中，使产品更快、更高质、更安全地交付到客户手中，更快占领市场，获取收益。同时，能更快更全面地获取反馈，持续改进产品，持续交付符合市场需求的特性，形成良好的产品交付正反馈，提升研发效能，使产品研发事半功倍。

3.1 CICD 平台概述

CloudEasy CAP CICD 平台覆盖代码管理、代码检查、编译构建、部署、测试、发布等

完整的、端到端的持续交付流程。通过灵活自定义的持续交付流水线，实现分钟级的产品发布，达到统一发布出口标准，并快速上线的目的。

CAP CICD 平台在组织架构、用户、权限、安全等公共服务的基础上实现持续交付，包括代码管理、代码检查、构建、部署、制品管理、测试、流水线，并可以对各类环境进行统一管理。适用于提升研发效能、DevOps 转型、持续集成/持续发布、自动化测试的场景。

其中，代码管理支持 Git 和 SVN 代码库，支持一键导入外部仓库，支持分支和标签管理，支持查看提交历史。

代码检查采用 Sonar 静态检查，支持查看规则，并定义规则集，支持设置质量门禁，支持代码提交触发检查任务，支持按多种维度对问题进行统计，并支持查看问题详情。

构建支持 Java、javascript、PHP、c、c++、HTML、css 等主流编码语言，后续也将不断增加编码语言。支持 Maven、gradle、NPM、Conan、docker、tomcat、springboot 等主流框架。

部署支持部署到物理机、虚拟机、容器、K8s 集群，支持多云部署。

制品管理支持依赖组件管理，支持自动归档镜像包、构建包，支持制品包的下载发布与拉取。

测试支持功能测试、接口测试、UI 测试三种测试类型，支持用例设计、任务管理，支持测试执行并输出测试报告和测试问题详情。

流水线支持从代码管理到发布部署的所有交付环节，包括代码源、代码检查、构建、制品上传、部署、测试、人工审核等过程，支持部署到开发、测试、类生产、生产等多种环境，实现云上云下一键部署。同时，流水线支持质量门禁控制，助力产品的高质量交付。

CICD 平台为了提升环境资源的利用率，也支持统一的环境管理，包括 K8s 集群、容器、Jenkins、Nexus 仓库和主机资源。

CICD 平台具有如下特点：

- 安全高效的代码托管

  支持主流的 Git 和 SVN 代码库，提升跨地域跨团队协同开发效率。基于 Git 的代码仓库支持分支管理、保护分支、分支合并等功能。
- 全面的代码检查

  代码检查覆盖主流编程语言、主流编码标准，支持质量门禁设置，支持跨函数的深度检查，准确定位代码缺陷，提供问题修复建议。
- 一站式测试解决方案

测试服务覆盖功能测试、接口测试、UI 测试，多维度评估产品质量，帮助用户高效管理测试活动，保障产品高质量交付。支持一站式开展用例设计、测试执行、生成报告，提高测试效率。

支持高效自动化测试，快速编排测试用例，集成流水线，支持微服务测试、分层自动化测试等多种测试场景。

支持需求-测试用例-缺陷双向追溯，测试有的放矢，多角色高效协同，多维度产品质量看板，全方位评估产品质量，保障产品高效验收。

- 端到端自动化流水线

  集成代码拉取、代码检查、编译构建（包括前端、后端、镜像构建）、部署、制品上传、测试、Jenkins 任务、人工审核、人工执行脚本、延迟执行等流水线任务。

  支持可视化、可定制的流水线编排，适应不同场景。

  集成质量门禁，保障产品质量。

  支持定时执行，实现流水线的自动化、持续可交付。

- 集中式制品管理

  制品库提供 Maven、Gradle 仓库，Docker 镜像依赖仓库和制品文件仓库，提供软件包上传/下载功能，实现软件包版本管理，提升发布质量和效率，实现产品的持续发布。

3.2 功能架构

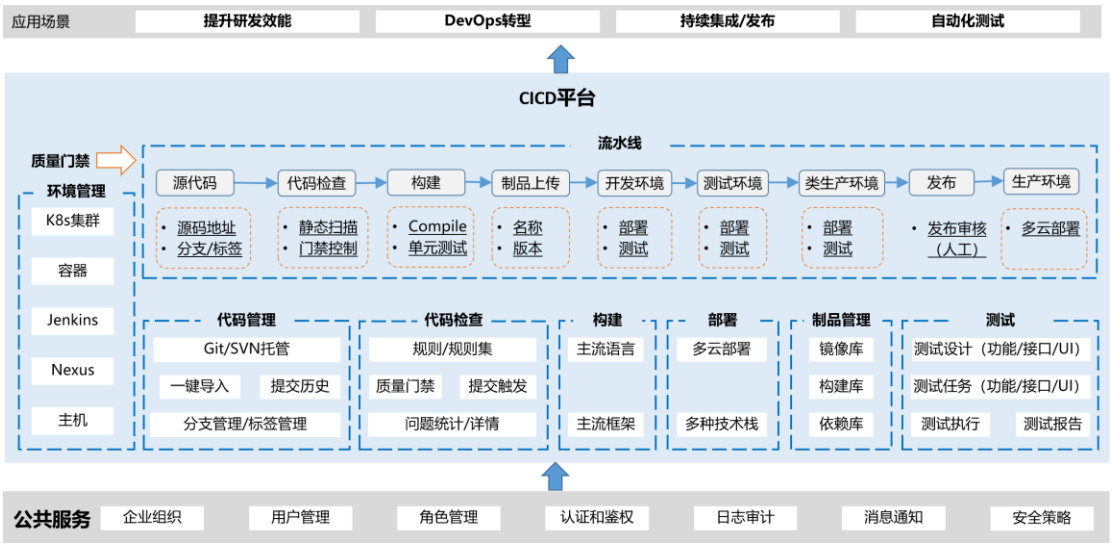
CICD 平台在组织架构、用户、权限、安全等公共服务的基础上，实现如下功能：

- 代码管理：完成 Git/SVN 代码托管。
- 代码检查：实现代码静态检查，保障代码质量。
- 构建：支持主流语言主流框架的编译构建。
- 部署：支持多种技术栈在多云环境下的部署。
- 制品管理：包含依赖库、构建库和镜像库。
- 测试：支持功能、接口、UI 测试设计，支持自动化测试，支持按任务执行测试，并输出测试报告。
- 流水线：覆盖检查、构建、部署、测试等完整流程，可视化按需编排流水线，支持手工、定时执行，步骤数据分离，根据实际需要灵活配置，适应不同团队和场景需要，实现分钟级应用上线。
- 环境管理：统一管理项目中的资源环境。

应用于多种目标场景：

- 提升研发效能
- DevOps 转型

- 持续集成/持续发布
- 自动化测试



3.3 功能描述

3.3.1 代码管理

系统可以基于 Git 和 SVN，提供在线代码托管服务。

- 对于基于 Git 的代码仓库，提供安全、可靠、高效的分布式代码托管服务。包括代码克隆/下载/提交/推送/比较/合并/分支等功能，并支持在线查看代码、在线新建代码文件、查看代码提交历史、基于标签提交代码、版本管理等功能。
- 对于基于 SVN 的代码仓库，具备代码托管和成员/权限管理等功能。系统的代码管理致力于解决研发团队在跨地域协同、多分支并发、代码版本管理等方面的问题。

同时，代码库还具备单独的成员和权限管理，保障数据安全。

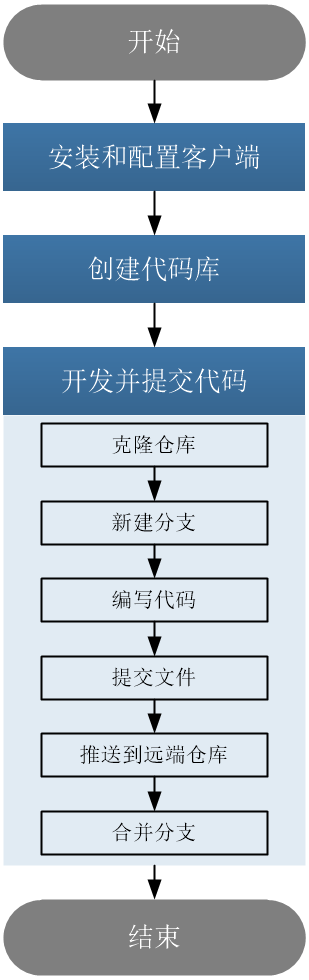
代码管理功能及特点：

- 适配 Git 和 SVN，覆盖常用的版本及代码管理工具。
- Git 代码库一键导入，高效完成代码迁移。
- 分支管理，实现多分支并行开发，达到开发隔离，发布统一的目的，管理效率高。
- 在线代码预览，随时随地阅读代码，不受地域限制。
- 分支保护，可防止分支被其他人提交或误删。
- 成员及权限管理，保护核心资产安全。
- Git 标签管理，帮助团队进行版本管理，以便日后精确检索历史版本。

- 代码提交记录查询，可视化代码修改过程，问题可追溯。

通过系统的代码库，从创建仓库到完成最终代码上库的基本流程，帮助开发人员使用代码库统一管理代码，并协同开发。

代码管理的基本流程如下图所示。



3.3.2 代码检查

代码检查是基于云端实现代码质量管理的服务。软件开发者可在编码完成后执行多语言的代码静态检查，获取全面的质量报告，并提供缺陷的分组查看与改进建议，有效管控代码质量，帮助产品成功。

CICD 平台的代码检查采用 Sonar 代码检查工具。代码检查规则沿用 Sonar 检查规则，代码检查服务根据经验，针对每种编码语言预置了规则集，您也可以根据实际情况自定义规则集。

代码检查具备如下功能特性：

- 支持主流编码语言检查： Java、JavaScript、PHP、C、C++、HTML、CSS。

- 融合开发流程，提供多分支检查。
- 预置上万条检查规则，针对编码语言预置规则集，并可根据实际需求自定义规则集。
- 问题精确定位，支持在线查看并修复代码问题。
- 针对问题，提供问题影响、修改建议和修改示例等，帮助您修改问题。
- 问题责任归属：采用问题责任人制，加速质量问题闭环。
- 多种度量指标，全面保障代码质量：代码问题、代码圈复杂度（内置风险度量体系）、代码重复率、代码注释率等。
- 支持设置质量门禁，确保流入下一环节的代码质量。
- 支持设置忽略文件，缩小检查的范围，提升检查效率，减少误报率。

3.3.3 测试管理

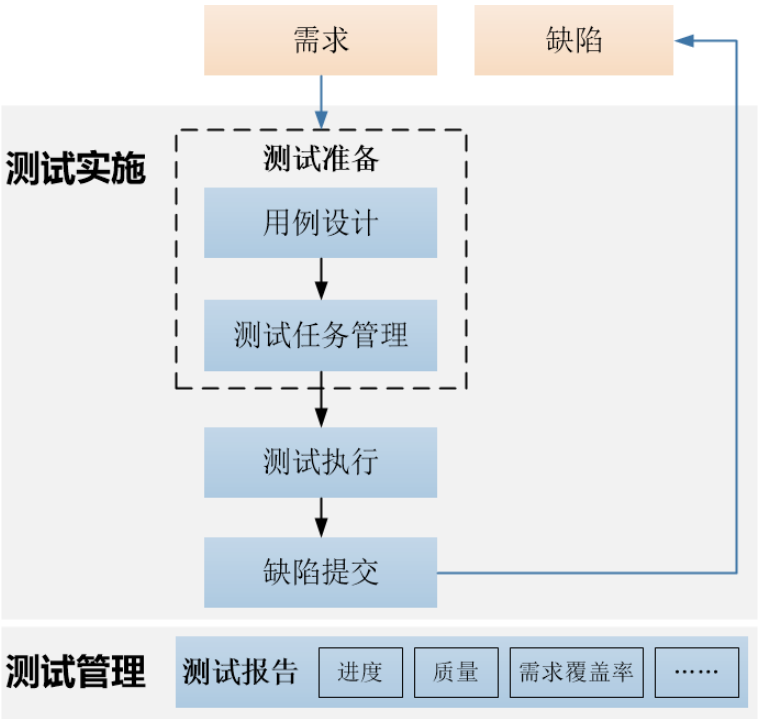
为了保障产品交付质量，所有的开发和修改都应该经过测试，且测试通过后才能流入下一个环节。

系统支持功能测试、接口测试和 UI 测试，其中接口测试和 UI 测试支持自动化测试。

系统的测试服务融入 DevOps 敏捷测试理念，覆盖功能测试、接口测试、UI 测试，支持与流水线集成，实现自动化测试，提高测试管理效率，保障产品高质量交付。

测试服务使用指引

测试服务支持的功能以及在测试过程中的应用如下图所示。



测试类型

- 功能测试

  功能测试是对产品的各功能进行验证，根据功能测试用例，逐项测试，检查产品是否达到用户要求的功能。

- 接口测试

  接口测试是测试系统组件间接口的一种测试，主要用于验证外部系统与系统之间的相互依赖关系，以及系统内部各模块之间的数据的交换，传递和控制管理过程是否符合预期。

  接口测试具备如下功能：

  可视化用例和脚本编辑、编排界面，免代码编写，技术门槛低，适合接口开发者、接口消费者、测试人员、业务人员等不同角色使用。

  一键导入 Swagger 接口定义文件，自动生成脚本模板。可基于脚本模板组装编排测试用例。

  支持丰富的参数化应用，包括全局参数、局部参数、响应提取参数，提升用例设计效率，降低测试复杂度。

  集成流水线，实现持续自动化测试。

- UI 测试

  UI 测试的对象是用户界面（User Interface），主要用于测试用户界面的功能模块的布局是否合理、美观、易懂，操作是否便捷，用户界面的功能是否在各种场景下能按照预期响应等。

  UI 测试具备如下功能：

  可视化用例和脚本编辑、编排界面，免代码编写，技术门槛低。

  测试数据与测试步骤分离，复用测试逻辑，降低脚本冗余度，增强测试用例的可维护性和投入产出比。

功能及特点

- 测试分层

  融入测试金字塔分层管理理念，支持功能测试、接口测试、UI 测试分别管理。

- 向导式用例设计

  功能测试、接口测试、UI 测试采用统一的用例设计模板，包括用例分级、前置条件、操作步骤、预期结果、关联需求等，引导完成测试用例设计。

- 任务式测试执行

  支持将多个测试用例组装成一个测试任务，用于完成特定的测试任务。您可以使用测试任务完成多轮测试或回归测试。

- 可视化脚本编写及编排

  接口测试和 UI 测试均支持测试脚本。脚本编写和脚本步骤的编排采用 UI 界面，技术门槛低，适用于各种技术背景人员。

接口测试脚本还支持 Swagger 文档导入，实现快速编排测试步骤。

- 完善的测试报告

  测试报告覆盖需求覆盖率、用例完成率、用例通过率、缺陷数量和用例数量，且支持从总体测试维度和每类测试维度分别统计。

- 需求-用例-缺陷双向追溯

  用例可关联需求和缺陷，确保所有需求都被正确测试，杜绝漏测、误测；确保所有缺陷可追溯来源。

- 集成流水线

  接口测试任务和 UI 测试任务可集成到流水线中，实现持续自动化测试。

3.3.4 流水线管理

流水线提供可视化、可定制的自动交付流水线，帮助项目提升持续交付效率，缩短产品交付周期。

流水线提供灵活易用的构建自动化、集成自动化、验证自动化、部署自动化功能，完成从开发到上线过程的持续集成、持续部署、 持续交付。同时，通过持续向团队提供及时反馈，让交付过程高效顺畅。

流水线支持流程自定义编排，通过代码检查、构建、部署、文件上传、测试、管控等组件化能力，把从开发到交付的各项工作串联起来，从而让企业轻松地实现持续交付 。

功能特性

本系统的流水线具备如下功能特性。

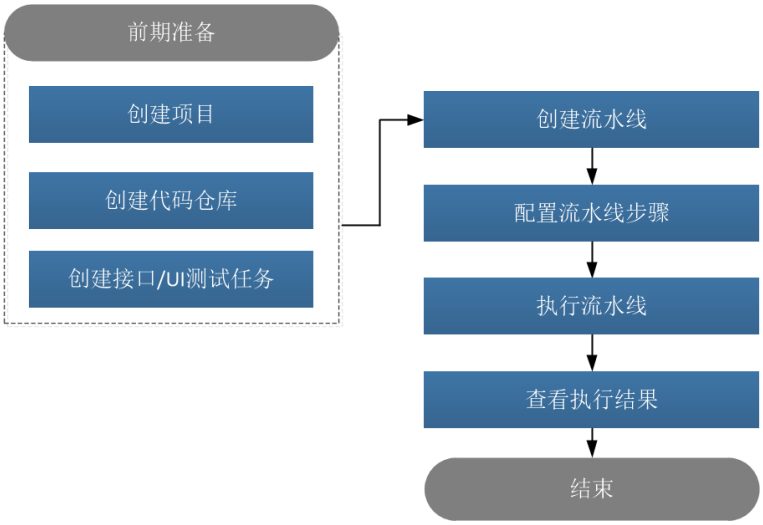
功能特性	说明
一键创建流水线	系统根据编程语言预置不同目的的流水线模板，并可自定义模板。创建流水线时，可选择模板，一键创建流水线。
自定义流水线模板	系统根据编码语言和目的预置流水线模板，也支持将当前流水线保存为模板，或者全新创建模板，便于后续使用。
流水线自定义编排	可根据项目需要，自定义流水线的阶段、任务和步骤。
流水线任务支持串并行执行	流水线任务支持串行编排，也支持并行编排，提升流水线执行效率。
支持多种步骤类型	支持代码检查、构建、部署、文件上传、测试、人工执行脚本、人工审核、延迟执行等多种任务类型。

功能特性	说明
流水线参数化执行	流水线支持自定义参数，在流水线步骤中引用，实现步骤和数据分离。也支持在执行时由用户指定参数值，任务使用指定值替换相应参数执行。
任务延迟执行	当后续任务依赖的多个任务不能同时执行完成时，可以使用延迟执行步骤，确保后续任务能成功执行。
流水线定时执行	流水线可以根据您定义的时间，自动执行。
支持执行历史记录	系统记录流水线的历史执行情况，供用户查看。
成员和权限独立管理	各流水线的成员和权限独立管理，充分保障流水线的访问安全。
消息通知	您可以定义流水线的事件发生后，通过邮件通知流水线创建者和执行者。

流水线基本操作流程

通过创建一条简单的流水线，介绍配置流水线的前置准备工作及基本操作流程，帮助您快速建立对流水线的整体印象。

在简单流水线的基础上，您可以根据实际情况设置流水线参数、设置执行计划、设置 webhook 等。当流水线执行结果有问题时，通过执行结果提示，处理问题。



3.3.5 制品管理

系统的制品库包含 maven 仓库、docker 镜像仓库和文件仓库。其中：

- maven 仓库

  用于存放 maven 和 gradle 编译构建需要的依赖包。

- docker 镜像仓库

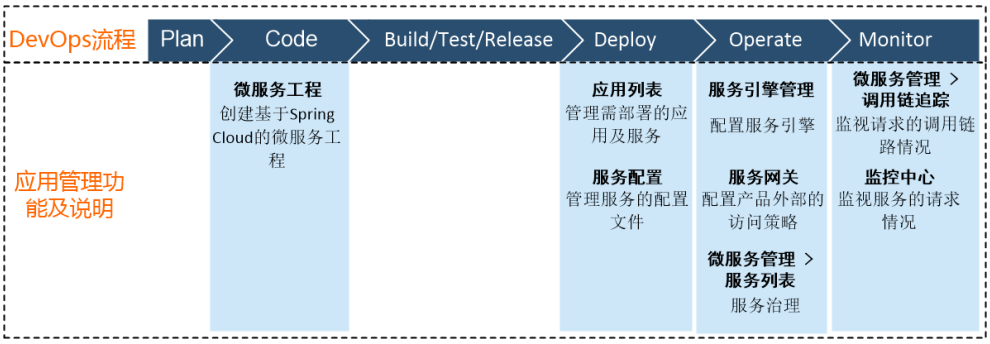
  用于 docker 镜像构建，将代码打包为 docker 镜像。打包好的镜像软件包存放在 docker 镜像仓库中。

- 文件仓库

  用于主机部署和回退。您可以将代码文件上传到文件仓库中，然后在主机部署和回退时，选择对应的版本部署到主机。

4 应用管理平台

应用管理平台覆盖 DevOps 流程编码、部署、运维、监控阶段。应用管理对于以上几个阶段的支撑如下图所示。应用管理在运维阶段主要实现服务治理功能。



4.1 平台概述

应用管理平台实现一站式微服务管理和运维能力，降低复杂系统的运维成本，提升服务性能和可扩展性。

应用管理平台提供完整易用的微服务注册、服务治理、服务监控和调用链等功能，具有如下特点：

- 应用托管，高效智能

  托管应用从创建、变更到释放的全生命周期，支持 Spring Cloud 和 Service Mesh 服务在相同注册中心互相发现和统一管理，实现跨框架的服务链路打通、治理策略下发和服务配置管理，降低技术复杂度。

- 策略丰富，按需治理

  可按需配置路由、熔断和限流等服务治理策略，最大化利用微服务的优势，保障系统的高可用性。

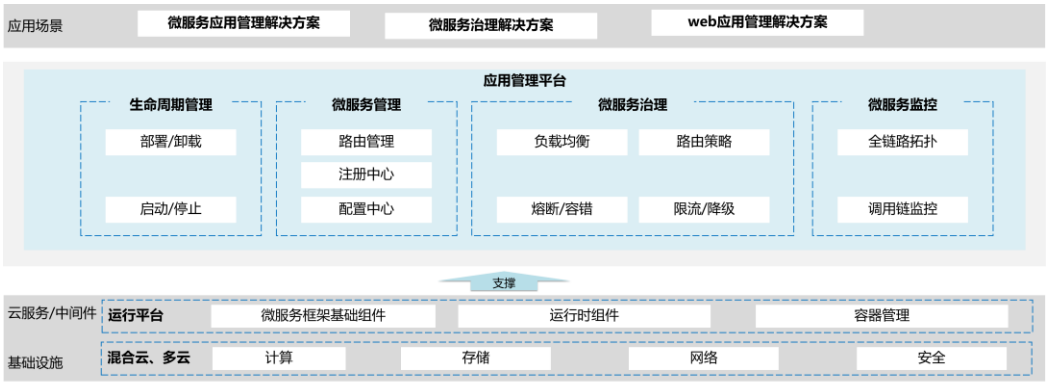
- 服务监控，可视分析

  提供调用链追踪全景展示，快速问题定位和性能瓶颈；提供全方位请求统计报表，发现趋势，防患未然。

4.2 功能架构

应用管理平台在基础设置、云服务、中间件的基础上，实现以下功能：

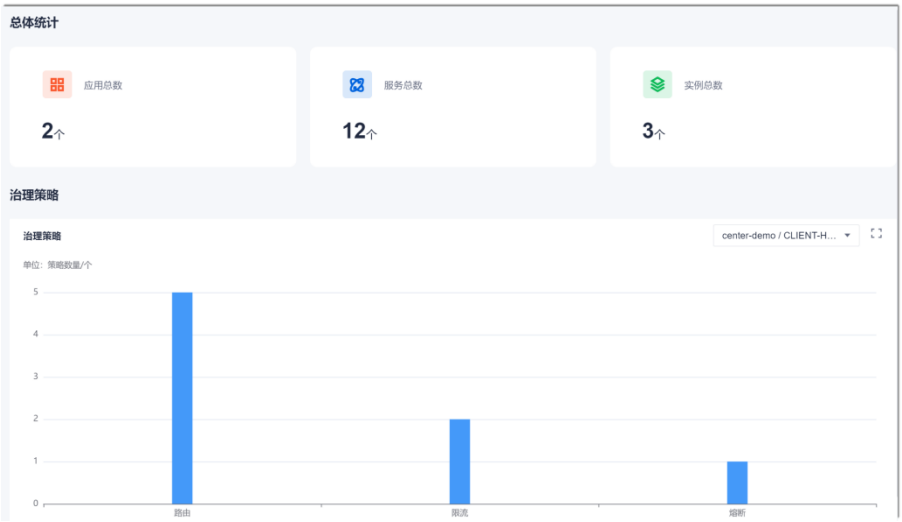
- 应用生命周期管理
- 微服务管理
- 微服务治理
- 微服务监控



4.3 功能描述

4.3.1 查看应用总览

应用管理总览提供微服务应用的相关统计信息，方便您了解应用的总体情况。



4.3.2 管理微服务工程

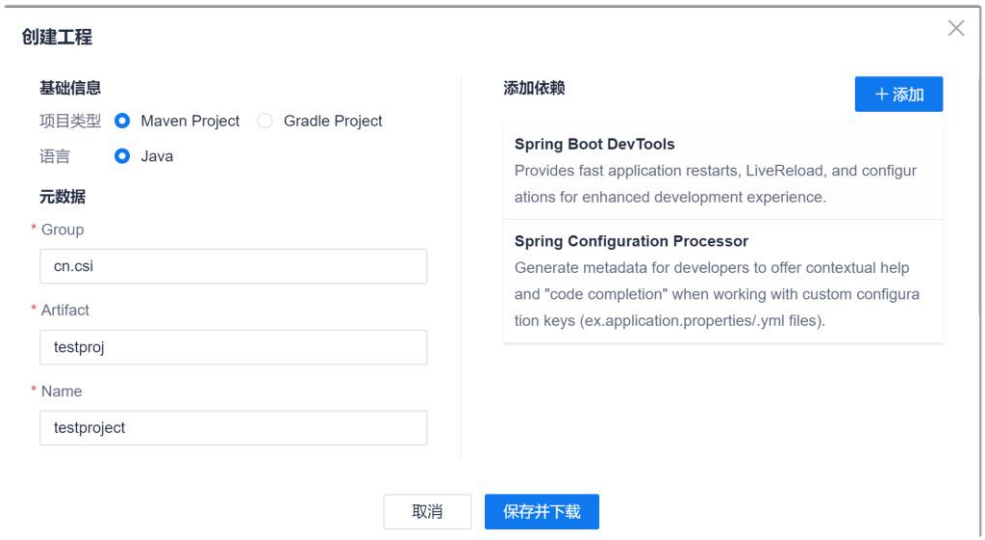
在开始编码前，需要创建微服务工程。如果采用 Spring Cloud 框架，则您可以在“应用管理 > 微服务工程”中创建工程，然后下载工程包到本地进行编码，减少手工创建工程的操作，提升准备工作的效率，提升代码结构的标准性。在创建工程时，可以添加常用依赖包。

系统当前支持的编译构建类型包括 Maven 和 Gradle，支持的编程语言为 Java 语言。

创建微服务工程

您可以创建基于 Spring Cloud 框架的微服务工程，并且添加依赖包。

在创建微服务工程时，可以配置项目类型、语言、元数据、打包格式、JAVA 版本、依赖包等。



下载工程包

微服务工程创建成功后，您可以多次下载工程包。

修改/删除/查看工程

工程创建成功后，您可以修改除“Group”、“Artifact”和“Name”外的信息。您也可以删除工程，查看工程的详细信息。

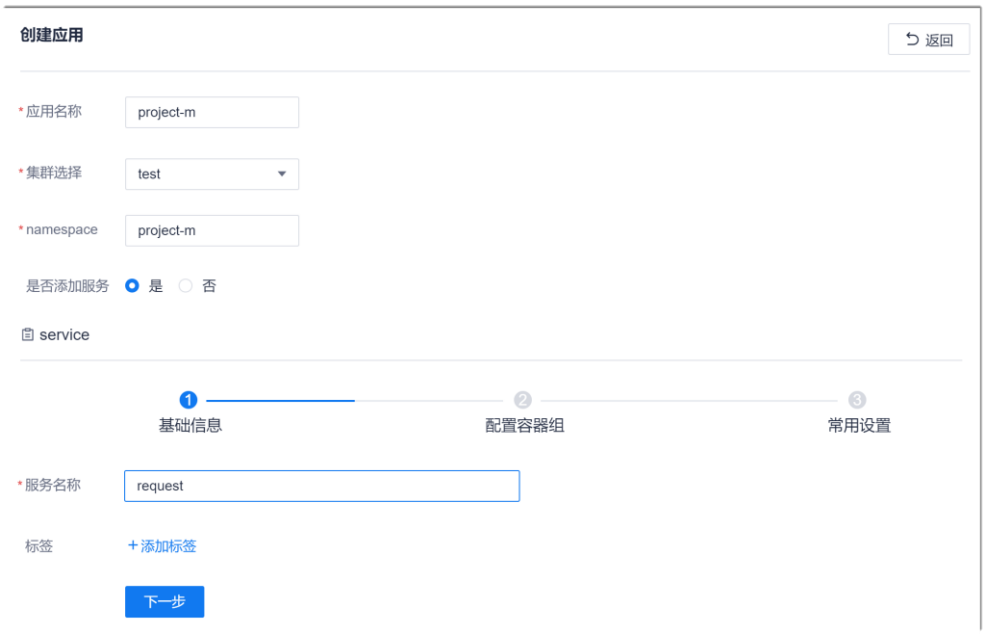
4.3.3 管理应用和服务部署

在 K8s 中，一个应用代表了一组功能的集合。完整的支撑应用的运行一般需要多个服务。一个服务可能由多个容器实现。

创建应用

应用中包含服务。在创建应用时，可以配置应用和服务的如下信息：

- 应用部署的集群、namespace
- 服务的镜像、容器信息、端口、实例数、升级策略等。



在应用中添加服务

在应用创建成功后，您可以持续添加服务到应用中。

部署服务

在创建应用和服务后，您可以直接在应用管理平台中部署服务。

启停服务

您可以根据实际需求启动或停止应用中的服务。

修改应用中的服务信息

可以修改服务的相关信息。

删除应用中的服务

可以在应用中删除服务。

查看服务详情

可以查看服务的详细信息。

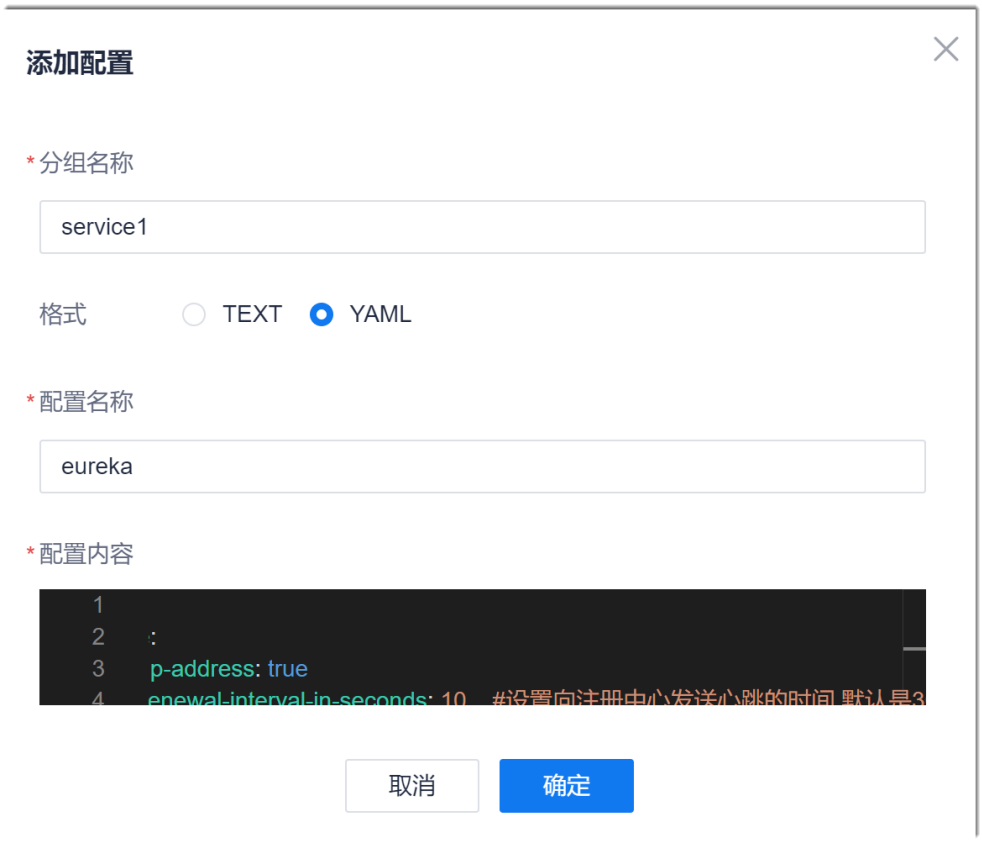
拷贝服务的访问地址

需要使用服务的地址时，可以一键复制。

4.3.4 管理服务配置文件

您可以将服务的通用配置写成配置文件，由服务调用，减少重复配置，提升配置效率。

您可以添加、修改、删除、查看配置文件。

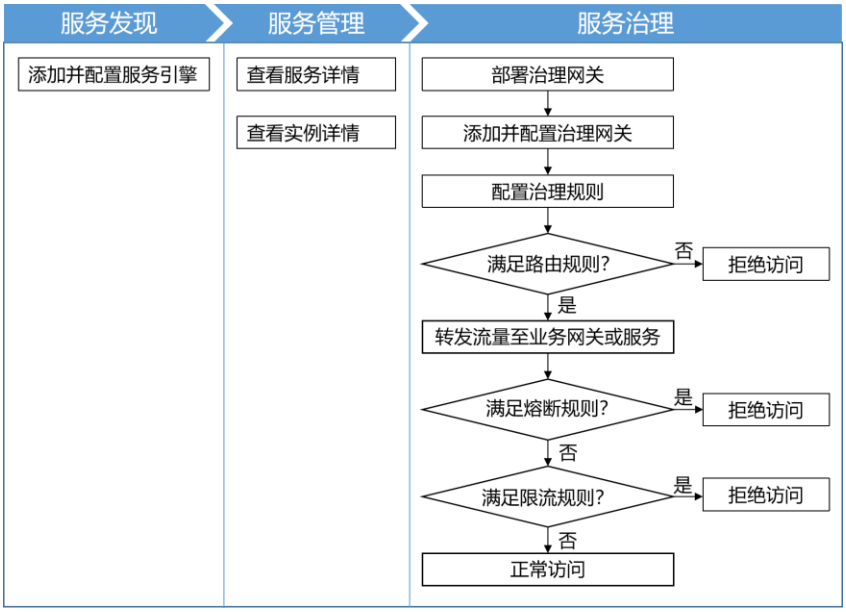


4.3.5 服务治理

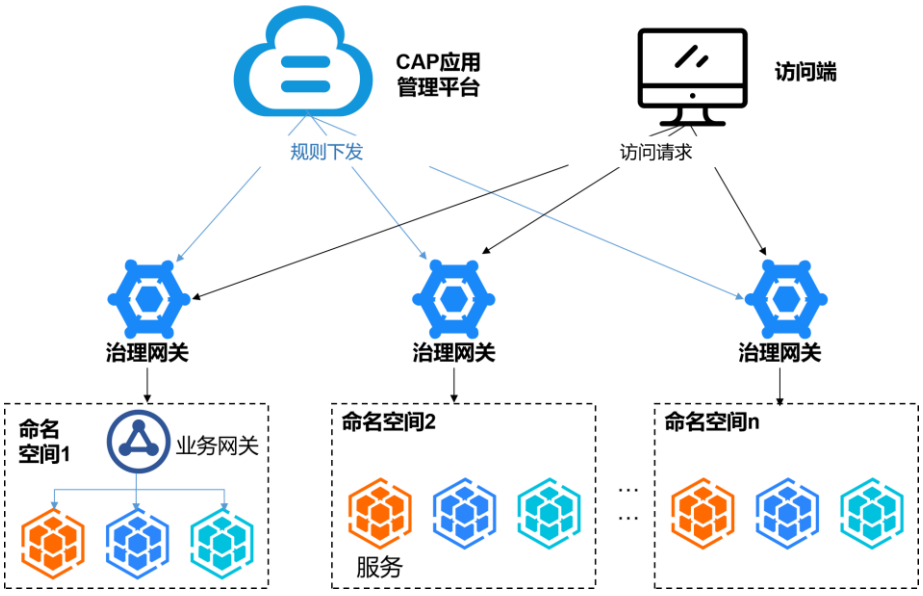
微服务治理首先需要发现服务。通过配置服务引擎的注册中心、配置中心、Kubernetes 集群、命名空间、认证信息等，可以发现服务并获取服务列表。 服务被发现以后，您可以查看服务中的实例详情，并根据需要进行服务治理，包括设置路由规则、限流规则和熔断规则。

在进行服务治理前，需要部署并配置治理网关，用于治理规则的下发和请求流量的转发。治理网关与服务引擎中的命名空间一对一绑定，便于数据隔离，保障数据安全。

微服务治理的业务流程如下图所示。



微服务治理的逻辑结构图如下图所示。



其中：

CAP 应用管理平台控制多个环境、多个节点的治理网关。在应用管理平台配置的治理策略下发到治理网关，再由治理网关进行规则判断，执行相应操作。

治理网关收到访问请求后，基于治理策略进行判断，将允许访问的请求下发到业务网关或直接下发到服务。

CAP 当前的微服务治理支持基于 K8s 集群的微服务架构，支持如下服务引擎框架：

系统当前支持 Eureka、Nacos、Istio 服务引擎类型：

- Spring Cloud Eureka

  Spring Cloud Euraka 是 Spring Cloud 集合中一个组件，它是对 Euraka 的集成，用于服务注册和发现。Eureka 是 Netflix 中的一个开源框架。

- Nacos

  Nacos 是构建以“服务”为中心的服务基础设施，致力于微服务的发现、管理和信息配置，能帮助用户快速实现动态服务发现、服务配置、服务元数据及流量管理，从而更敏捷、更容易的构建、 交付和管理微服务平台。

  Nacos 支持几乎所有主流类型(诸如: Kubernetes Service. gRPC & Dubbo RPC Service.SpringCloud RESTful Service)的服务发现、配置和管理。

- Istio

  Istio 是基于 Serice Mesh 的一种服务管理平台，适用于微服务架构。服务之间的通信通过代理来进行。Istio 提供服务的连接（例如服务发现、负载均衡、流量控制等）、安全加固、控制（例如访问规则控制）和观察（例如流量）等服务。从而为微服务架构下的故障排查、应用容错性、应用升级发布、系统安全等难题提供了可能的解决方案。

4.3.6 监控服务

您可以监控服务被请求后的调用链路，也可以监控服务在某段时间内，被请求的总数量、平均响应时间、实例数、吞吐量、包大小、出错率等。