灵雀云全栈云原生平台

产品手册

北京凌云雀科技有限公司

目 录

1 产品背景

传统企业在今天都面临着新兴业务模式的剧烈冲击，同质化的竞争手段已无法让企业在愈演愈烈的竞争中脱颖而出，包括金融、能源、制造、汽车以及政府机构在内的传统企业，纷纷致力于数字化转型。通过数字化转型，企业能够快速感知用户的需求并做出调整，加速产品迭代更新，不断地提升用户体验和满意度，从而获得或提高市场差异化竞争优势。

在数字化换转型浪潮的推动下，业务的敏捷、弹性、个性化和智能化需求凸显，应用的交付模式也发生了深刻的变化，要求技术架构具备轻量化、松耦合、灵活敏捷的特点。软件能力成为了帮助企业实现软件定义业务、自主 IT 研发和业务的持续创新，让企业具备从竞争对手中脱颖而出并保持优势的 IT 核心竞争力。

一直以来，在传统的企业软件开发模式中，开发、测试和运维人员分别隶属于不同的部门。各部门虽然共享组织目标，但是彼此之间缺乏高效的协作方式，导致在日常 IT 运营和维护的过程中矛盾不断。

在传统的交付周期中，软件开发人员在编写代码后，将软件交给测试团队进行测试，再将最终版本交给运维团队部署。运维团队需要解决代码部署过程中出现的问题，或将代码交还给开发团队来解决遇到的问题，复杂的流程导致软件从开发到交付效率低下且周期较长，因此原有的开发运维流程，已不适应新时代的科技发展。许多初创企业急需一种快速创新、频繁迭代的能力来适应科技变革，以顺应行业发展趋势，通过减少成本控制预算，提高产品质量获得市场认可，从而提高企业的核心竞争力。

伴随着 Docker 容器、Kubernetes、微服务、DevOps 等热门技术的兴起和逐渐成熟，利用云原生（Cloud Native）解决方案为企业数字化转型，已成为主流趋势。云原生解决方案通过使用容器、Kubernetes、微服务等这些新潮且先进的技术，能够大幅加快软件的开发迭代速度，提升应用架构敏捷度，提高 IT 资源的弹性和可用性，帮助企业客户加速实现数字化转型。通过容器技术搭建的云原生 PaaS（Platform-as-a-Service）平台，可以为企业提供业务的核心底层支撑，同时能够建设、运行、管理业务应用或系统，使企业能够节省底层基础设施和业务运行系统搭建、运维的成本，将更多的人员和成本投入到业务相关的研发上。

本公司长期聚焦在云原生领域，在技术、产品和客户方面都有深厚的积累。为了满足更加广泛的企业云原生需求，本公司借助多年来服务不同领域头部客户的技术实践经验，推出了新一代的聚焦于使用场景的全栈云原生开放平台（以下简称平台或本平台）。

2 产品简介

平台拥抱云原生（Cloud Native）技术，以可以灵活组合订阅的子产品（Container Platform、DevOps、Service Mesh）的形式，整合了 Docker 容器、Kubernetes 和 Istio 原生架构、DevOps、Operator 等相关新技术和新理念，可实现业务应用从开发、测试，到部署、运维、治理的全生命周期平台化管理，能有效帮助企业实现数字化转型，提升企业的 IT 交付能力和竞争力。

作为企业级的云原生容器平台，可服务于不同规模的企业，支持管理各种复杂度的基础设施环境（单台机器或多个异构的数据中心）、组织结构清晰的部门建制和人员团队。

平台能够满足企业级应用逐步向容器化、微服务化过渡的广泛需求，支持企业建立一个覆盖内外部各环节和组织结构的私有云平台。提升企业 IT 资源利用率，加快应用迭代速度，降低应用交付成本，实现业务应用的智能运维，从而助力企业获得持续创新的核心能力。

2.1 基础平台

基础平台是企业级全栈云原生开放平台的基座，能够为子产品（Container Platform、DevOps、Service Mesh）提供核心的底层支撑。通过可视化界面一键部署后，即可通过基础平台部署、管理子产品。

基础平台主要面向平台管理员、运维人员和审计人员，从这三类人员的使用角度出发将平台管理、项目管理、智能运维相关的核心能力集中在了平台中心视图，为使用者提供了独有的操作视角，便捷地实现了访问权限控制，保障了平台安全。

- ✧ 统一管理平台上的集群和 Kubernetes 资源；
- ✧ 统一管理平台上的用户，并能灵活地控制用户角色的分配和回收；
- ✧ 通过项目、命名空间实现资源、权限隔离，为企业提供细粒度的租户管理能力；
- ✧ 可通过可视化面板查看详细的平台审计、日志、事件、监控和运营统计数据，当问题发生时，可为运维人员提供快速排查、解决问题的关键信息；

- ✧ 支持对平台执行自动化巡检和功能健康状态检查，可通过可视化界面实时了解平台资源、功能的运行状况，规避平台运行风险；
- ✧ 结合告警、通知可实现平台的智能运维，提升运维效率，节约运维成本。

2.2 Container Platform

结合基础平台的能力，Container Platform（容器平台）能有效助力企业数字化转型，智能化 IT 基础架构，通过**平台化的基础设施、高效的容器服务、自动化运维管理、服务化的 IT 治理**，赋能中小型企业的 IT 部门，使其同时扮演业务支撑者和业务驱动者的双重角色。

- ✧ **平台化的基础设施管理**：全面集成 Kubernetes 容器编排引擎，在 Kubernetes 业务集群与平台对接后，执行统一管理，并提供健全的容器网络与容器存储解决方案，形成平台化的基础设施；
- ✧ **高效的容器服务**：使用容器管理应用，实现快速部署、节省资源占用的同时，还具备灵活的应用编排和交付能力，保证在多场景下交付应用。支持将容器运行在 Docker 和 Containerd 两类容器进行时中；
- ✧ **自动化运维管理**：提供以应用为中心的智能运维体验，屏蔽基础运维架构，使用户更专注于核心业务；
- ✧ **服务化的 IT 治理**：支持中小型企业的多租户管理场景，实现细粒度权限控制和自助 IT 治理。统一管理和监控不同基础设施环境上的资源，通过安全审计机制，保障系统安全性。
- ✧ **开箱即用的数据库和消息队列服务**：通过 Operator 简化数据库和消息队列服务的构建、部署和运行过程，帮助用户更便捷、自如地管理和使用数据库及消息队列。

图 2-1 Container Platform 产品结构图

2.3 DevOps

平台基于 DevOps 理念搭建，重视软件开发人员（Dev）和 IT 运维技术人员（Ops）之间沟通合作的文化和惯例，为企业提供包含流水线管理、代码仓库纳管、制品管理等在内的开箱即用一站式服务，能够帮助企业专注于业务目标，提升企业内研发、测试、运维之间业务的连续性、安全性、敏捷性，实现自动化部署和运维，缩短开发周期，提高部署频率，轻松应对瞬息万变的市场需求。

- ✧ 平台提供开放式的 DevOps 工具链选型，在平台中深度集成敏捷项目管理、代码管理、CI、CD、制品管理等类别中的主流工具并灵活兼容客户已有的工具选型，灵活实践 DevOps。

- ✧ DevOps 强调产品管理、自动化软件交付和灵活应对基础设施的变更，旨在缩短产品开发周期，提高部署频率，紧密结合业务目标支撑应用的全生命周期，为企业建立起能够快速、频繁、稳定构建且能实现可靠的测试、发布的文化与环境。
- ✧ 平台以 Jenkins 作为实现 CI/CD 的标准工具，助力实现更敏捷、可靠的应用发布，加快产品迭代速度，使企业更专注于核心业务目标。
- ✧ 平台提供方便、快捷、易落地的解决方案，为企业提供基于 DevOps 的开发体验，帮助研发团队缩短交付周期、提高研发质量。

图 2-2 DevOps 产品结构图

2.4 Service Mesh

Service Mesh（微服务治理平台）同时支持 Istio 和 Spring Cloud 两种主流的微服务框架，为云原生提供了一整套稳定、可靠、开放的微服务解决方案。平台的一键式基础设施部署升级能力，可视化服务治理能力，高效的应用性能管理能力，以及高可用、高性能的微服务网关管理服务，能够帮助企业提升服务治理效率，全面降低维护微服务框架的成本。同时，可减少开发人员对框架的依赖，让企业聚焦于业务开发，不断提升核心竞争力。

- ✧ 构建分布式微服务系统

通过分布式微服务架构取代传统的单体应用架构，满足快速扩容、弹性伸缩、敏捷迭代、快速交付等新型业务的诉求。

✧ 自动化灰度发布

基于动态流量策略，实现应用及服务的自动化灰度发布，满足金丝雀、蓝绿等多种灰度发布场景。

✧ 可靠的服务治理

平台提供丰富的可视化治理策略，动态配置服务路由、服务熔断、服务限流、负载均衡规则，覆盖大部分微服务治理场景。

✧ 服务全局可视化

平台基于流量监控数据，全局展示服务间的调用关系和调用性能。通过从全局聚焦到局部，可为服务调优、服务排障、下钻分析等场景提供多维度监控分析能力。

✧ 数据平面无侵入扩展

通过 WebAssembly（WASM）对数据平面的流量治理能力进行自定义扩展。平台通过 OCI 镜像以插件化的形式管理 WASM 制品和配置，并基于 ECDS （Extension Config Discovery Service）API 进行下发遥测，为安装过程提供监督管控能力。

✧ “多地多中心”业务容灾

基于“多主”架构的服务网格，支持纳管多个集群，多个集群之间完全独立。任一集群故障，不会影响到网格下其他集群的工作。

服务可部署在网格纳管的任意集群中。当某个集群下的服务因故障触发熔断后，服务在该集群下的部分入口流量将转移至其他集群，由其他集群上的服务负载，以保障服务整体的负载性能稳定。并可根据集群的实际地域分布，自定义容灾负载优先级。

✧ 灵活的多网关（网关多端口）管理

支持为服务网格配置多个入口或出口网关，每个网关允许通过多个端口访问。其中，入口网关支持通过不同的路由场景将外部流量路由到不同的后端资源；出口网关可实现网格内服务对外部服务的访问管理及流量监控。可满足业务区域之间网络隔离、业务独占网关、端口隔离等丰富的使用场景。不再需要的网关，可轻松无损下线。

除此之外，可基于网关的黑白名单和 JWT 对发起请求的客户端进行身份认证与鉴权，并根据请求路径、请求 Header、URI 重写等多样化的方式配置路由规则；支持实例级别的网关监控，配合多种内置告警策略，可帮助用户及时发现、解决问题。监控数据涵盖了 CPU、内存、QPS、连接数、传入流量、传出流量等指标。

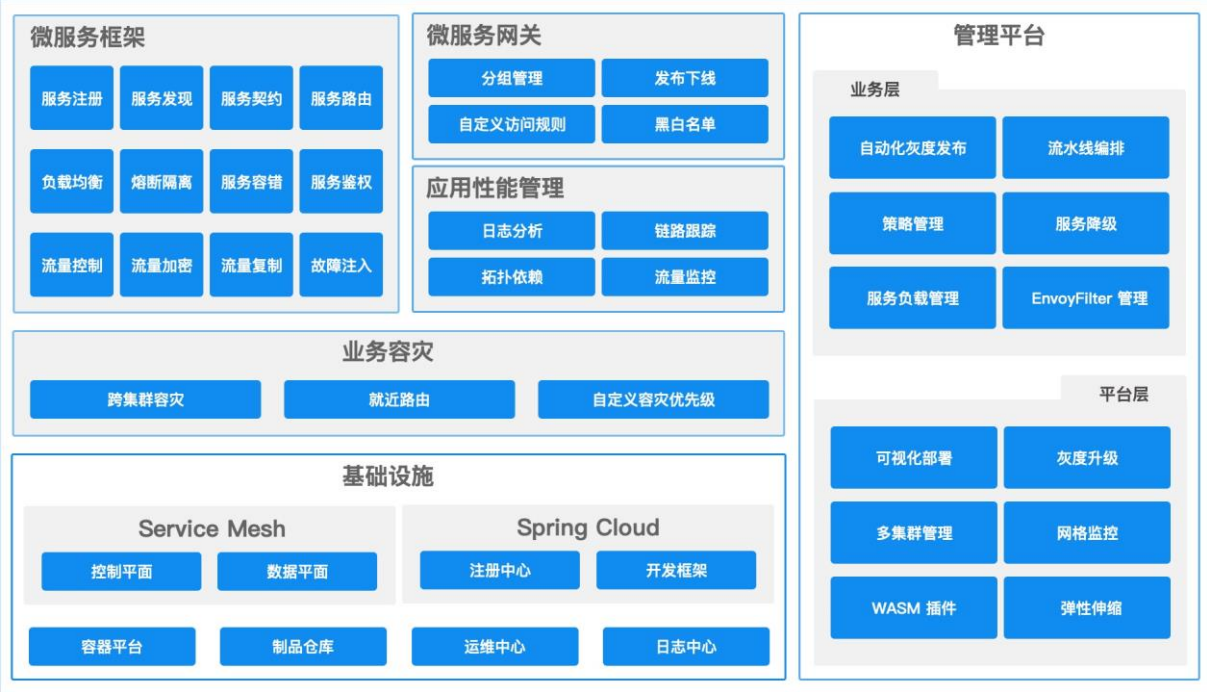

图 2-3 Service Mesh 产品结构图

2.5 中间件 PaaS 服务

中间件 PaaS 服务基于容器与 Kubernetes 技术，借助 Operator 为企业提供开箱即用的 MySQL、Redis、Kafka、RabbitMQ、MongoDB、PostgreSQL 等中间件服务，帮助企业的中间件上云，有效保障企业的业务连续性。

中间件 PaaS 服务遵循灵活的容器编排和高可用设计，提供高可用、高性能、高保障的企业级中间件服务，使企业更专注于核心业务目标。

- ✧ 简化了部署和开发过程，帮助开发人员更自如地使用中间件。
- ✧ 提供丰富的可视化运维能力，助力运维人员管理大规模中间件，快速定位问题并排障。

3 产品特点

3.1 多集群管理

- ✧ 具备超大规模的集群管理能力，可同时管理 5000 个主机节点，并可在 5000 个主机节点上同时运行 15 万个 Pod。
- ✧ 支持将其它第三方平台部署的不同云端的原生 Kubernetes 集群接入平台，统一管理不同基础设施环境中的集群和主机，保障了故障迁移，使企业能快速有效并低成本地跨区域、跨平台运行集群。
- ✧ 支持纳管接入平台的标准 Kubernetes 集群以及 OpenShift 集群下的资源，对接集群和平台的账号权限后，即可通过平台管理接入平台的集群下的资源，并对集群进行运维管理。
- ✧ global 集群支持同时纳管 X86 和 ARM 架构的业务集群，提高了对国产化支持程度，降低客户资源使用成本。
- ✧ 基于 Kubernetes Operator（状态管理器）实现了集群全生命周期地自动化管理，包括：部署、升级、组件管理、备份、恢复和异地灾备。
- ✧ 支持将两个或两个以上集群联邦化后统一管理，可通过跨集群部署和管理应用，实现“两地三中心”容灾备份解决方案。
- ✧ 支持对集群进行可视化备份及恢复，可以让您从意外灾难中快速恢复，也能帮您轻松地跨集群迁移业务数据，并保证数据的可靠性。

- ✧ 支持统一管理、统一调度企业使用的私有云、公有云，既保持私有云的可控性、安全性和保密性，也可发挥公共云的简单性、灵活性和成本效益管控。能够以更快的速度、更高的可扩展性和敏捷度，跨私有云和公有云环境为企业应用程序提供自动化的基础架构服务，同时，降低了成本和风险。

3.2 基于 RBAC 的用户权限体系

- ✧ 在支持搭建自由用户体系的基础之上，对接 LDAP（Lightweight Directory Access Protoco，轻量级目录访问协议）或 OAuth 2.0 等常见认证协议，纳入企业已有的用户体系，可通过 RBAC（Role-Based Access Control）细粒度设置用户权限；
- ✧ 基于 Kubernetes RBAC 的权限体系，贯通了平台上所有 Kubernetes 集群下的资源。仅需在平台中心配置一次，即可自动同步至平台上的所有集群；
- ✧ 基于企业实际使用场景，系统抽象出了六个内置的系统角色，可满足大部分企业的权限配置需求。

3.3 企业级多租户管理

支持基于多租户的跨集群项目管理，项目可共享平台的基础设施资源。可根据项目需求，为项目分配平台上的一个或多个集群，并通过配额限定集群下该项目可用的资源大小。项目管理员可在被分配的集群上创建命名空间，为命名空间分配配额，并添加命名空间成员。命名空间成员登录平台后，可在自己具有相应权限的命名空间下管理应用。

平台的企业级多租户管理能力，极大地简化了为角色相同成员分配相同资源权限的重复工作，利于项目内外部的资源隔离，也加强了数据的保密性。

3.4 自动化运维管理

- ✧ 平台提供了监控、日志、事件、审计数据的可视化面板，平台上集群、主机节点以及 Kubernetes 资源的状态变化有源可溯，当系统产生故障时，可有效缩短故障排查时间，减少故障排查处理难度；
- ✧ 除监控之外，还支持日志、事件的指标化，对数据进行统一分析后可根据分析结果制定告警规则。通过通知的形式将告警信息实时地发送给运维人员，可有效地避免一些系统运营过程中资源不足导致的故障（例如：CPU、内存不足），降低系统运维成本；
- ✧ 告警策略模版、通知策略可灵活地对接企业已有的通知系统（邮箱、短信、webhook、钉钉、企业微信）。
- ✧ 支持在线对平台进行资源风险巡检和资源用量巡检，可视化展示巡检结果，并可下载巡检报告。

3.5 无缝对接 Kubernetes

深度集成 Kubernetes 容器编排引擎，提供标准的 Kubernetes 容器编排服务，平台更好地发挥了 Kubernetes 的产品特性。

- ✧ 支持一键部署业务集群和网络，支持 Kube-OVN、Calico、Flannel 等网络模式；
- ✧ 自定义应用基于标准 App CRD 定义，可将应用下所有资源作为一个整体进行管理。支持容器化应用的自动化部署、扩展，自动保留应用全局的历史版本，可按需创建应用快照，实现应用的全局回滚以及应用拓扑的可视化；同时，支持将导出的应用（应用模板）上传至平台的本地仓库，并通过分配仓库的使用权限，在平台上特定的多个命名空间下快速部署相同的应用；

- ✧ 支持 OAM（Open Application Model）应用管理。OAM 应用能使您在更高层级管理应用，尽量屏蔽容器和 kubernetes 层的细节，使您能更关注于应用程序本身的运维。
- ✧ 可按需集成存储、监控、日志等解决方案；
- ✧ 容器化管理计算组件，通过平台可以像管理产品一样管理应用的全生命周期，进一步提高资源利用率。

把 Kubernetes 原生架构当做开发框架，基于 Kubernetes 扩展机制开发平台功能。在平台性能、稳定性和可扩展性得到保证的同时，支持标准的 Kubernetes API，兼容 Kubernetes 生态工具和系统，支持集成 Kubernetes 原生插件。

3.6 成熟的微服务治理能力

- ✧ 服务注册中心支持高可用的多集群、多网格架构；
- ✧ 业务应用改造，可无侵入、零成本接入；
- ✧ 提供成熟、灵活的应用部署方案，支持灰度发布、联邦容灾等高阶能力；
- ✧ 调用链跟踪支持与日志联动，提供良好的交互体验；
- ✧ 基于流量实现服务治理，不影响业务；
- ✧ 灵活的多网关（多端口）部署、管理、无损下线功能，配合网关监控、告警能力，可为平台提供稳定的网关服务。
- ✧ 微服务网关提供 API 分组、发布/禁用、网关路由等可视化管理功能；
- ✧ 日志服务提供从采集、呈现到分析、检索的一站式服务能力，能够帮助开发者快速定位目标日志，提升排查故障、解决问题的效率；

- ✧ 多维度的资源监控（服务流量监控、API 流量监控和 JVM 监控）配合告警功能提供“监控-告警”一体化智能运维体验，帮助开发者快速定位、解决问题。

3.7 可开箱即用、灵活选择的 DevOps 工具链

平台支持快速部署 DevOps 工具链，如 Jenkins，GitLab、Harbor，SonarQube 等，满足了开箱即用的需求，为企业提供基础的 DevOps 工具，同时支持集成企业已经在用的工具链，丰富工具选型，降低学习成本和迁移成本，强大的集成功能助力企业快速落地生产环境，实现开箱即用的 DevOps 体验。

3.8 持续构建，持续交付

平台提供基于 Jenkins 的软件开发实践方案，支持图形化创建、模板创建等多种流水线创建方式，通过可视化界面的引导，快速创建特定交付流程的流水线，使用流水线触发器自动或手动触发交付流程，为企业提供高可用的持续交付服务。

平台同时扩展了基于 Tekton 的软件开发实践方案，支持通过 YAML 与图形化表单结合的方式自定义持续构建、持续发布流程，使用人工审批、质量门将发布流程透明化，确保整个应用交付流程如期进行，借助其强大的云原生能力，完美支持资源下沉、资源横向扩展，为企业提供高可用的持续交付能力。

3.9 基于 Operator 的可视化数据服务

平台提供开箱即用的数据库和消息队列服务，简化构建、部署和运行过程，帮助用户更易管理，更自如地使用数据库及消息队列。

- ✧ 支持快速部署 Operator。开发者在使用操作器能力的同时，无需了解复杂的 Kubernetes API 配置。
- ✧ 基于“定制资源定义”（CRD，Custom Resource Definitions），统一管理所有实例资源，例如 Kafka 集群、Kafka Topic 及 Kafka User。
- ✧ 自动化部署和运行 Kubernetes 相关资源，进一步提高资源利用率。

- ✧ 提供界面化的实例创建、更新及管理操作，包括界面表单和界面 YAML。在表单中展示已筛选并翻译的关键字段，且提供相应注解，帮助开发者快速完成基本配置。同时也开放界面 YAML 功能，可用于更多字段的进阶配置。

3.10 快速部署、可视化运维、高可靠的中间件服务

平台基于 Operator 能力，支持图形化一键部署中间件实例，支持不同版本、不同架构的中间件实例，提供标准的中间件服务。还为关键特性提供了开箱即用的操作界面，例如 MySQL 参数配置、Kafka 的 Topic 管理，简化开发过程。

此外，平台还提供了丰富的可视化运维能力，包括对实施的监控图表展示、日志可视化展示、状态展示及生命周期管理。并支持中间件的故障自愈。当中间件节点发生故障时，平台将自动拉起故障节点，实现自动运维。

本公司基于为不同领域头部客户的服务经验，将持续优化提升中间件的性能，助力业务获得高性能体验的中间件。同时提供中间件的高 SLA 保障，服务可用性可达 99.95%。

3.11 标准化交付

本公司借助于多年来服务不同领域头部客户的经验，打造标准化的产品旨在服务于更加广泛的领域和更多的企业客户，助力更多的企业客户数字化转型落地，获得持续创新的核心能力。标准化交付的平台还具有部署、实施、运维、升级标准化以及可以和第三方容器平台进行融合的特点，结合专业的技术团队和实施团队，能够为目标客户和合作客户提供更多的便利性和可能性，顺势推动 PaaS 容器平台的普及和落地。

同时，标准化交付的平台可单独使用为企业提供容器服务，也可以结合其他功能模块（例如：DevOps、微服务治理）提供更为丰富的 PaaS 生态体验，或接入第三方容器平台帮助容器化产品落地。

4 核心功能介绍

4.1 平台管理

- 基础设施管理

平台以统一多集群管理为核心，可对接稳定快速的物理机服务器、资源利用率高的虚拟机、不同云环境（公有云或托管云）下的云主机创建 Kubernetes 集群。

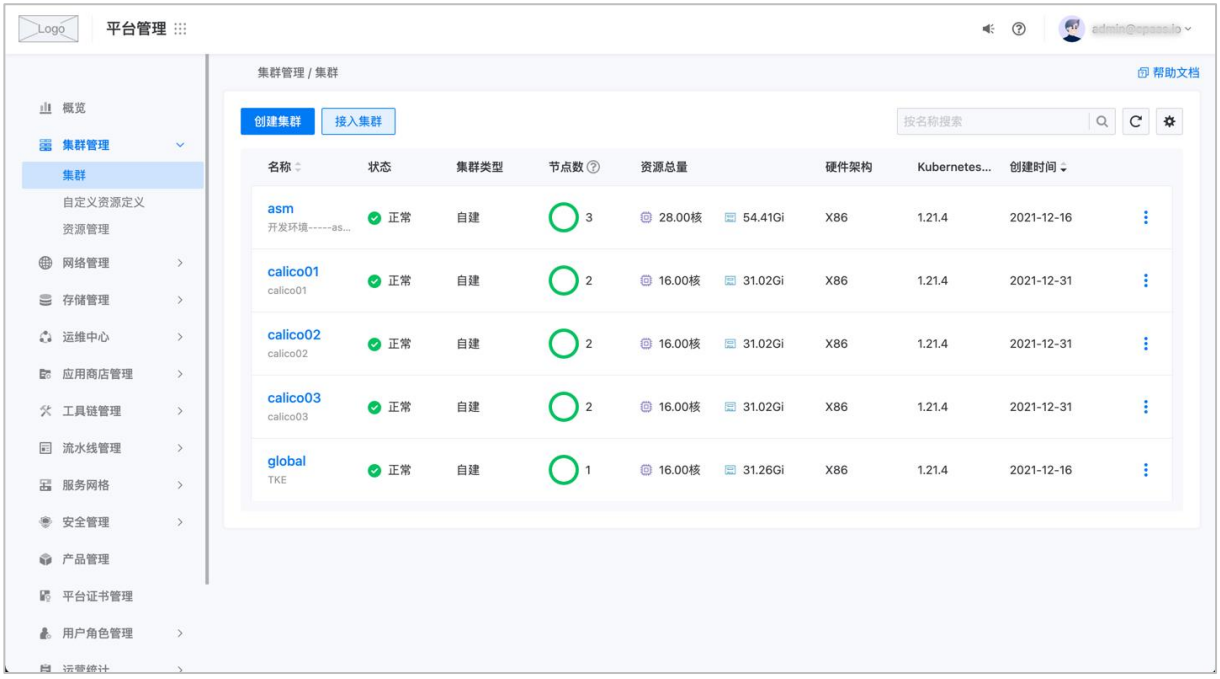

图 4-1 多集群管理

✧ 集群部署

以统一多集群管理为核心，支持在物理机、虚拟机、不同云环境（公有云或私有云）下的云主机上一键部署 Kubernetes 集群，方便从 IaaS 资源池添加新主机到 Kubernetes 集群里，并快速完成节点初始化。

针对使用场景可选择部署不同类型 Kubernetes 集群，例如：开发/测试环境、部署单个控制节点的 POC 环境、部署多个控制节点的高可用生产环境。

✧ 多集群管理

-支持运行多集群，即跨集群容器资源池统一管理运行在多个云端上 Kubernetes 集群，保证集群的高可用，解决多云灾备问题。具备超大规模的集群管理能力，可同时管理 5000 个主机节点，并可在 5000 个主机节点上同时运行 15 万个 Pod。

-支持接入外部标准 Kubernetes 集群和 OpenShift 集群和 Amazon EKS 集群，并将集群下的命名空间纳管至平台进行统一管理。

-集群对接 Kubernetes 发行版后，用户可通过发行版统一管理部署在平台上的集群。

-满足一云多芯能力，支持纳管 X86、ARM 架构的业务集群，全面支持信创需求。

-为确保业务集群地址高可用，支持使用“自建 VIP”作为业务集群地址。

✧ 集群监控

支持全局集群监控仪表盘，运维人员可便捷地通过监控和统计数据了解平台上每个集群的状况，查看集群下主机节点的运行状态和数据。

✧ 节点维护

支持变更集群中节点的调度状态，在需要进行节点维护时，将节点设置为不可调度，并可将节点上运行的 Pod 迁移到其他节点上。

自建集群支持添加或者删除控制节点，在控制节点故障时，添加新的控制节点替换故障的控制节点。

支持节点的标签管理，创建应用时通过配置节点选择器可将应用部署在指定的节点上运行。

✧ 资源配额

支持将集群的资源分配给多个项目，在项目下创建命名空间时，可选择已分配资源的集群，并为命名空间设置资源配额。

支持更新集群的超售比，帮助管理员将集群下命名空间中用户设置的容器 CPU、内存的限制值（limit）和请求值（request），限制在合理范围之内，提高计算资源利用率。

✧ 联邦集群

将两个或两个以上集群联邦化后统一管理，可实现联邦应用跨集群部署和管理。

✧ 备份恢复

-支持 ETCD 可视化备份，针对不同的备份恢复场景支持在本地备份或远端对象存储备份。

-支持业务应用的可视化备份与恢复，可用于业务应用灾难恢复及业务应用跨集群迁移。

- 快速部署并集成

使用快速部署并集成功能可在首次使用工具时，无须在后台环境中进行繁琐的手动部署过程，只需简单填写相关参数，即可将工具一键部署并集成至平台，降低工具的环境准备和部署的操作复杂度，使用单点登录统一用户权限，方便用户快速实践 DevOps，提高开发效率。

- 集成 DevOps 工具链

平台具有完备的工具链集成体系，并致力于丰富开发者生态，从生产端到应用端，助力开发流程敏捷化，通过搭建可扩展、开放、丰富的工具链集成体系，帮助企业更快更好地进行应用开发与服务创新。平台支持集成多种类型工具链，如：持续集成、代码管理、制品仓库、代码质量分析、测试管理和项目管理工具等。您可通过以下方式将工具链集成至平台中：

集成已有工具链：若您正在使用的工具链在已支持集成的工具列表内，可直接集成至平台。

插件集成工具链：若平台现有工具链列表中没有您正在使用的工具链，可通过插件集成的方式添加至平台工具链列表中并集成，例如：Jira、禅道。

自定义工具链插件：若以上两种方式并未满足您的集成需求，可根据联系技术人员获取平台工具链插件开发规范，自定义工具链插件，真正实现工具链的灵活扩展。

平台现已支持集成的工具类型如下：

✧ 持续集成

平台支持集成 Jenkins。通过编排流水线，做多语言的持续集成、自动化的代码与镜像安全管理、镜像同步等等。

✧ 代码管理

平台支持集成多种代码管理工具，例如：GitLab、Gitea 等。在整个研发流程中，让用户专注于业务本身，不再关注代码仓库地址、凭据等信息，通过简单的选择即可快速、便捷地使用代码仓库。

✧ 制品管理

平台集成了多种制品管理工具来管理制品，实现镜像的存储、管理等，例如：Harbor Registry、Docker Registry。同时平台集成了制品仓库管理工具 Nexus，简化了内部仓库的维护和外部仓库的访问。

✧ 代码质量分析

平台支持集成代码质量分析工具，例如：质量管理平台 SonarQube，可帮助用户对代码质量做自动化分析和管理，添加代码扫描质量门，提前预测应用风险。

✧ 测试管理

平台支持集成测试管理工具，例如：TestLink，通过绑定测试管理服务，使用自动化的流水线实现测试用例的自动管理，提高测试人员的工作效率。

- 分配平台项目

平台管理员可将已集成的工具链资源，统一手动或自动将工具链内资源分配至平台项目，使项目人员更加专注于业务开发，无需关心资源分配细节，方便平台统一管理和维护工具链。

- 用户管理

平台即支持创建并管理本地用户，也支持管理已对接并同步至平台的第三方用户。

平台支持 Dex 服务，通过修改 Dex 相关配置，即可使用 Dex 已实现的连接器（Connectors）认证的账号登录本平台。例如：LDAP、GitHub、SAML 2.0、GitLab、OpenID Connect（OIDC）、LinkedIn、Microsoft、AuthProxy、Bitbucket Cloud。使用平台的 IDP（Identity Provider）配置功能，手动添加 LDAP 和 OIDC 后，即可通过同步 LDAP（Lightweight Directory Access Protocol，

轻量级目录访问协议）导入企业已有的用户体系，或直接使用 OIDC（OpenId Connect）协议认可的第三方账号登录平台。

为了进一步保障平台及用户账号安全，平台增加了用户安全策略，包括密码安全、用户禁用、用户锁定、密码通知、访问控制策略。并且在审计日志中，详细记录了用户在平台上的全部操作。

- 角色管理

平台支持基于角色的权限访问控制（RBAC，Role-Based Access Control）。基于企业的使用场景，系统默认内置了五大权限角色：平台管理员、平台审计人员、项目管理员、命名空间管理员、命名空间成员。

同时，为了满足更为复杂的权限控制，平台支持企业根据自己的实际使用场景自定义角色。

通过给不同的用户绑定不同的角色，将权限分配给用户。简化了权限管理，优化了权限隔离。

- 运营统计

平台提供运营统计功能，方便平台管理员实时查询平台上各资源的使用情况并导出统计报表。结合全面的统计数据分析平台资源的分配、利用情况，合理地为项目、命名空间分配资源，提升平台上资源的利用率和运营效率。

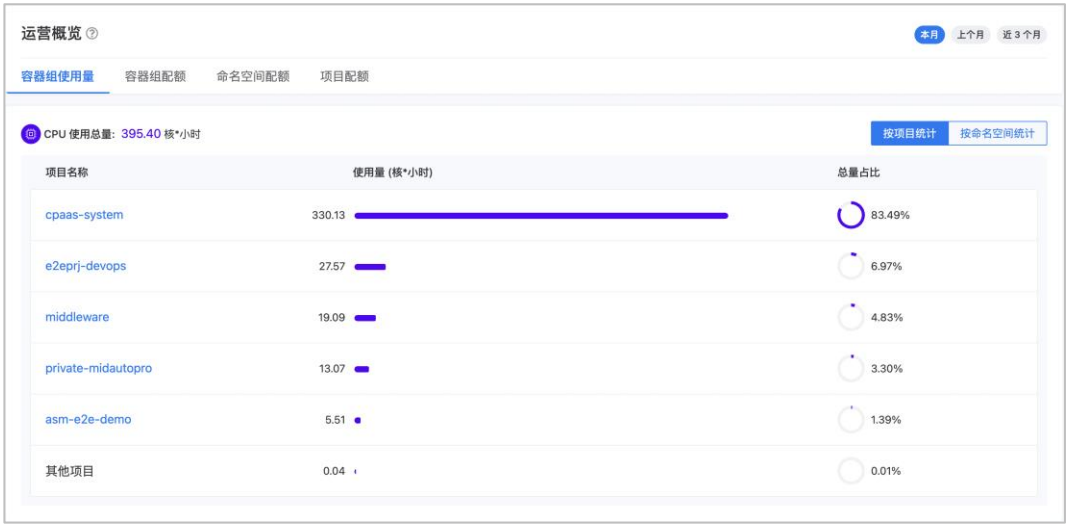

图 4-2 运营概览

4.2 项目管理

平台的项目之间可以灵活地划分出独立且相互隔离的资源空间，每个项目都拥有独立的项目环境，能够代表企业中不同的子公司、部门或项目组。通过项目管理，能够轻松实现项目组之间的资源隔离、租户内的配额管理以及项目下的人员管理。如图 4-3 项目列表所示。

一个项目支持绑定多个集群，并管理计算资源配额，提高了资源利用率。

一个项目下可以创建多个命名空间，作为互相隔离的工作区间，进一步实现了更小粒度地资源隔离和人员管理。保障了安全的同时，满足了不同组织架构的企业需求。一个项目的多个命名空间可以分布在不同的集群，一个命名空间只能在一个集群中。

同时，支持为项目绑定多个企业私有的域名，以便通过域名访问项目下应用。

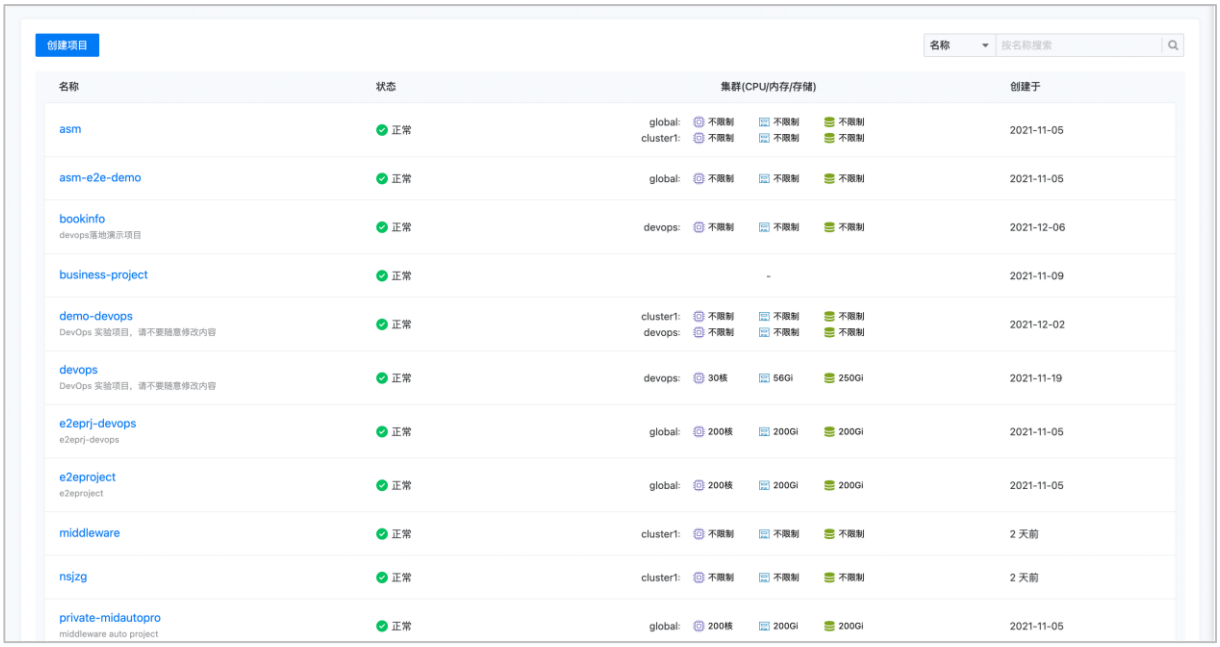

图 4-3 项目列表

图 4-4 项目详情

4.3 容器服务

深度集成 Kubernetes 容器编排引擎。支持类型多样的容器调度设置，例如：通过设置主机亲和性可将容器调度到指定的主机上运行；支持 Flannel、Calico、Kube-OVN 等多种主流的容器网络模式；支持对接不同类型的存储资源。

- 资源管理

无缝对接 Kubernetes，可按需自定义的 Kubernetes CRD（Custom Resource Definition，自定义资源类型）资源，建立贴合业务需求的应用模型以及应用创建流程。Kubernetes 原生资源或通过平台的资源管理功能按需自定义的 Kubernetes CRD（Custom Resource Definition，自定义资源类型）资源，统一由 Kubernetes 进行管理。在减少资源数据量过大对平台性能和稳定性影响的同时，降低了平台维护的成本。

同时，支持查看和更新集群或命名空间下的 Kubernetes 资源，提供 Kubernetes 集群的深度使用体验。

- 容器调度

全面深度支持 Kubernetes 容器编排引擎的管理功能特性，容器化管理工作负载，并根据配置自动化执行容器调度。

提供标准的 Kubernetes 容器编排服务，可实现容器应用的自动化部署、扩展和管理。

支持多种部署策略，可根据服务的特性选择不同的部署策略。

支持为容器配置健康检查，只有当容器内进程处于就绪状态时才能对外提供服务。

- 容器网络

提供强大的容器网络功能，支持 Kubernetes 的 Flannel 覆盖网络（VXLAN 和 Host-GW）以及 Calico、Kube-OVN 网络；提供网络双栈支持。

提供支持典型的四层或七层负载均衡，以及可实现多种复杂规则灰度发布的负载均衡器（Load Balancer）方案。通过配置负载均衡，可把应用组件中的容器实例，挂载到负载均衡器上，把负载均衡器的服务地址作为统一访问入口。

负载均衡本身支持高可用部署。方便应用运维人员配置服务端口，且可保证多个应用共享同一负载均衡器时，端口不会冲突。

支持统一管理客户企业私有的域名资源，可将域名分配给指定的集群下的一个或全部项目使用。

- 容器存储

管理员可通过 Kubernetes 的存储类使用不同类型的存储资源：外部存储、内置分布式存储或本地存储。普通用户通过为应用关联持久卷声明（PVC）请求持久卷（PV），或挂载主机本地存储的方式，可实现容器持久存储。

✧ 外部存储

平台支持对接的外部存储资源类型包括：CephFS、CephRBD、NFS（Network File System）、Huawei OceanStor Pacific 海量存储。

✧ 分布式存储

分布式存储为平台提供的集群内超融合存储方案。基于开源的 ROOK + Ceph 存储方案，内置分布式存储实现了可自动管理、自动扩容、以及自动修复的分布式存储能力，可满足中小规模的块存储、文件存储和对象存储需求。

-基于 Ceph 提供统一存储，包括文件存储，块存储、对象存储。

-支持存算一体/存算分离两种部署模式。

-提供图形化方式配置和自动部署存储集群和存储管理服务。

-分布式和多副本机制保障数据安全可靠。

-简单可靠的自动化管理，快速扩容存储资源。

-提供弹性和高性能存储服务。

-支持可视化的容量、性能和组件级监控，便于日常运维和管理。

-提供内置告警策略，满足大部分存储运维场景的需求。

-提供持久卷快照备份和恢复新卷功能。

-支持 COSI 对象存储接入方式，更便捷地使用 Ceph 对象存储桶资源。

-支持图形化方式跨集群对接内置分布式存储和接入第三方 Ceph 存储系统。

-提供混合磁盘类型配置方案，提升存储性能。

-提供物理网络隔离的部署方案，保障网络传输性能。

-支持根据硬件配置和性能要求自定义 Ceph 配置参数。

-与腾讯 CSP 采用同源 Ceph 版本，提供一致的售后保障。

✧ 本地存储

为了满足数据库、机器学习、区块链等业务场景下使用高性能存储的需求，平台也提供了本地存储方案。本地存储是一种软件定义的服务器本地存储方案，提供简单、易维护和高性能的本地存储服务能力。基于社区的 TopoLVM 方案，通过系统的 LVM 方式来实现本地存储的持久卷编排管理。

-基于 TopoLVM 提供云原生架构的本地存储方案。

-提供与本地磁盘一致的零损耗高性能存储。

-支持图形化向导快捷部署本地存储服务。

-支持动态创建 PVC，持久卷动态扩容。

-支持按节点进行设备管理，对本地存储进行扩容和故障管理。

-支持可视化的存储集群状态、容量和性能监控图表。

-配合 Kubernetes 实现自动定点部署配置。

-提供持久卷快照备份和恢复新卷功能。

-支持存储节点扩增和存储设备扩容管理。

-开箱即用资源告警策略。

-自主研发并开源，提供全面的售后保障和技术支持

- 容器配置

通过 Kubernetes ConfigMap、Secret 统一管理容器配置，并支持在更新配置字典/保密字典后，通过自定义命令，热更新业务容器中的相关配置，不重启容器即可实现配置的即时生效。

- 容器日志

容器内进程一般会将日志以标准输出方式展示，或写入到日志文件内。容器平台支持将这两种类型的日志收集起来，在前端 UI 展示和查询。

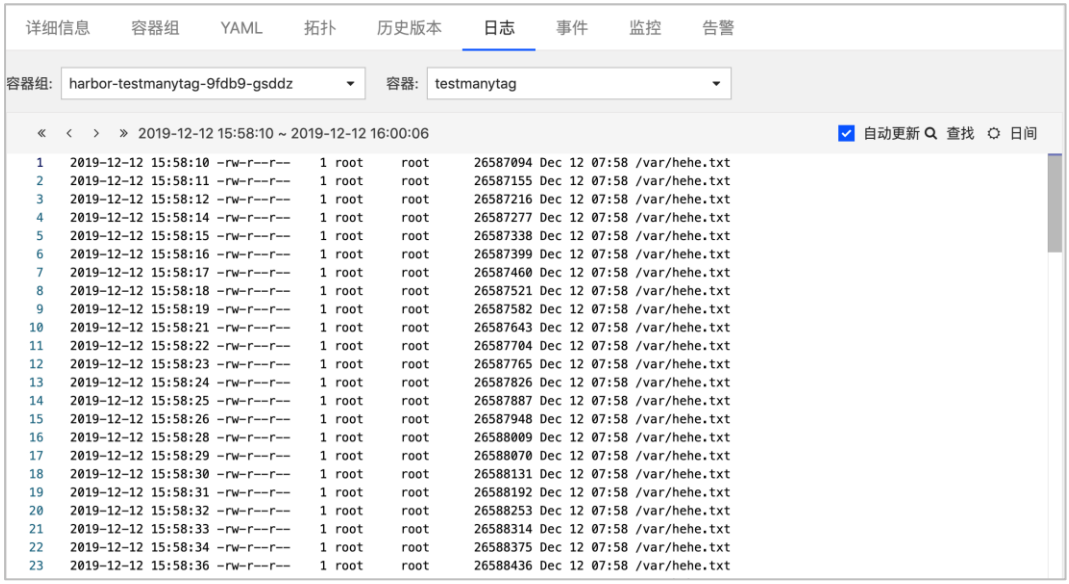

图 4-5 容器日志

- 容器监控

平台提供应用、计算组件、容器组监控能力，收集基础的监控数据，例如：CPU 使用率、内存使用率和每秒网络接受/转发字节数。

图 4-6 容器监控

- 应用编排&应用治理

✧ 原生应用

原生应用按创建方式，又分为两种类型。通过 UI 编辑模式或 YAML 编排文件创建的，称为自定义应用；通过 Helm Chart 创建的，称为模板应用。原生应用支持用户在研发、运维、测试或生产环境中运行不同类型的业务。

业务的连续性需要长期稳定运行的环境，应用治理是一项持续的工作。平台的自定义应用（Application）作为 Kubernetes 的 CRD 资源，由一个或多个关联的工作负载构成，支持根据不同的部署和使用需求，选择对应的部署模式：Deployment、DaemonSet 和 StatefulSet，并关联网络、存储、监控、健康检查以及其它配置。

自动保留应用全局的历史版本，可按需创建应用快照，实现应用的全局回滚以及应用拓扑的可视化。同时，支持将导出的应用（应用模板）上传至平台的本地仓库，并通过分配仓库的使用权限，在平台上特定的多个命名空间下快速部署相同的应用。

支持查看应用下所有资源的可视化拓扑图。

结合实时监控、Kubernetes 原生资源 HPA（Horizontal Pod Autoscaler，自动扩缩容）以及 CronHPA 组件，支持在部署类型为 Deployment 的组件中，根据 CPU 的利用率设置指标自动扩缩容规则或定时自动扩缩容规则。

✧ OAM 应用

原生应用作为一组 Kubernetes 资源集合的方式存在，对原生应用的操作实质上是对一个一个 Kubernetes 资源的操作。而 OAM 应用采用了一种更高层的抽象，让应用维护人员可以在不深入 Kubernetes 资源层的情况下，以更简单和更符合应用架构的方式管理应用。

OAM 应用由应用组件组成，应用组件可以包含多个运维特征。应用组件和运维特征都被精心设计为贴近应用的概念。如：WEB 服务组件、有状态服务组件、自动伸缩运维特征、自定义监控运维特征等等。平台提供预制的组件和运维特征，同时也开放扩展能力，让用户可以根据自身应用的特点自定义组件和运维特征，从而让应用运维更加标准化、简单化。

✧ 其他应用

平台中还支持组件应用和联邦应用。组件应用基于 Operator，帮助用户深入管理有状态应用。联邦应用基于联邦集群，为应用灾备提供解决方案。

- 云原生虚拟化

  云原生虚拟化是一种以云原生容器组方式运行和编排虚拟机的技术。平台采用开源的 Kubevirt 组件，通过 Kubernetes add-on 方式，以容器镜像为模板创建虚拟机并提供对虚拟机的完整生命周期管理能力。

✧ 支持图形化方式快速创建和管理 Linux 或 Windows 虚拟机。

- ✧ 支持 X86/ARM 架构芯片
- ✧ 支持虚拟机镜像管理，提供公共镜像导入和自定义镜像方案。
- ✧ 支持动态分配虚拟机 IP 地址和固定地址功能。
- ✧ 提供虚拟机资源可视化监控和告警配置。
- ✧ 支持通过 VNC 控制台对虚拟机进行维护管理。
- ✧ 支持对虚拟机进行启动、停止、重启、删除等全生命周期管理。
- ✧ 支持通过虚拟机快照进行数据备份和恢复。
- ✧ 支持设置密码/密钥方式登录虚拟机，提供重置管理功能。
- ✧ 可根据业务情况升降虚拟机配置、重装操作系统。
- ✧ 云原生架构统一编排虚拟机，与容器化应用相同的云原生网络和存储资源。
- ✧ 支持将虚拟机应用的服务端口通过云原生网络方式（Ingress/NodePort/LB）发布到外部网络。

- 应用商店

针对 Kubernetes 编排下微服务管理问题，平台集成了 Helm 和 Operator 开源项目并进行了扩展（例如：提供了图形化界面），帮助简化 Kubernetes 应用的部署和管理。

应用商店提供了预置模板一键部署复杂容器化应用及常用中间件服务，并支持使用自定义模板，包括从远端模板仓库、代码仓库导入的模板。

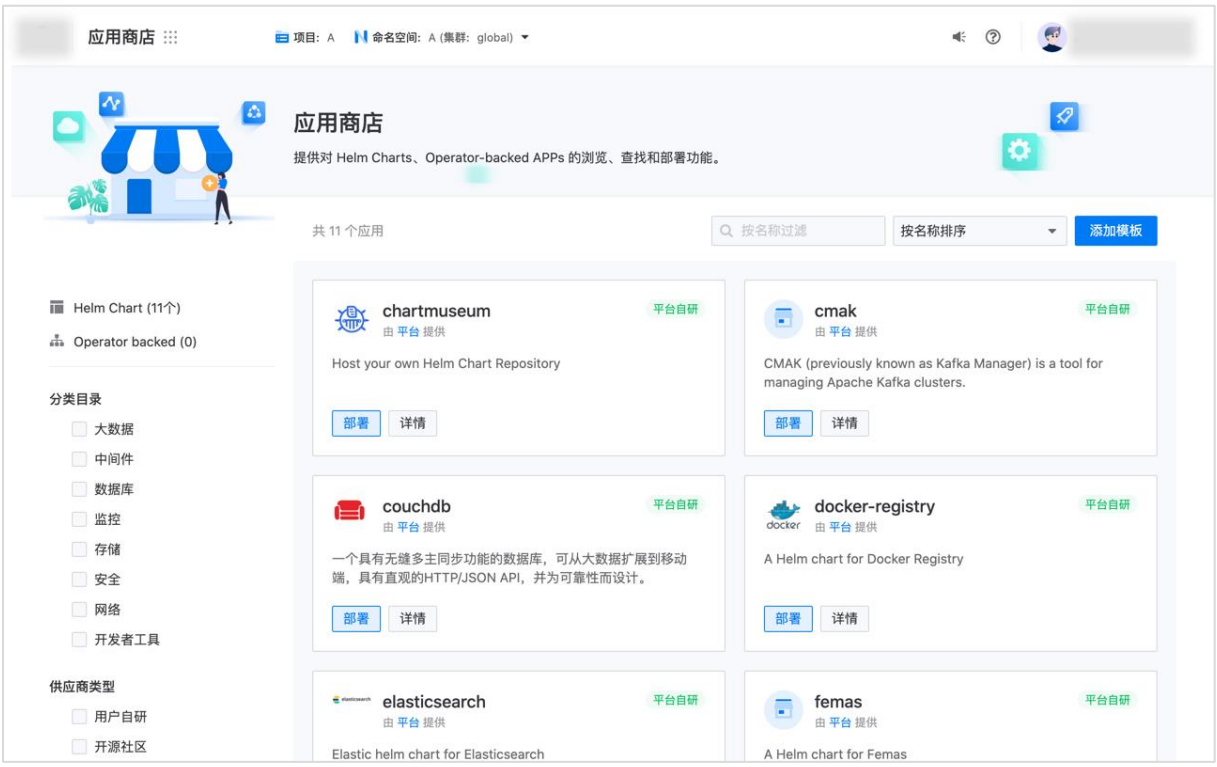

图 4-7 应用目录

平台管理中的 Chart、Git、SVN 类型的模板仓库，是存储用户私有化定制的 Helm Charts 的远端代码仓库的本地映射。而 Local 类型的仓库则可用于管理用户从本地上传的应用模板。支持用户将远端代码仓库中的应用模板（例如：企业定制开发的 MySQL、Kafka 等中间件应用模板）同步至平台或将本地 Chart 上传至平台的本地仓库，并通过分配项目配置仓库中模板的使用权限，控制企业的不同部门或团队访问专属的模板仓库。命名空间开发人员可基于权限范围内可见的应用模板快速在不同的命名空间中部署模板应用。

通过应用商店，平台管理员可便捷地将企业内部的公共服务下发给多个项目开发人员使用。方便企业内部服务的统一管理、维护和分配，一旦服务变更可快速地同步变更并重新部署应用。

- OperatorHub

通过 OperatorHub，展示用作部署和管理 Kubernetes 原生应用程序的 Operator。，Operator 可以理解为解决某些问题或实现某功能的一个或多个应用的合集，具有以下功能特点。

✧ **开箱即用**：集成常用的 Operator，包括平台自研 Operator、第三方 Operator、开源社区 Operator 和用户自研 Operator。平台部署完成后，即可使用。

- ✧ **种类多样**：集成多种类型的 Operator，包括应用架构、数据服务、开发流程等类型，满足开发者常用场景。
- ✧ **易部署**：支持 Operator 的一键部署，简单易用。
- ✧ **易配置**：为 Operator 实例的创建，提供了友好的 UI 配置界面。
- ✧ **技术支持**：对于平台自研的 Operator 组件提供升级和技术支持。

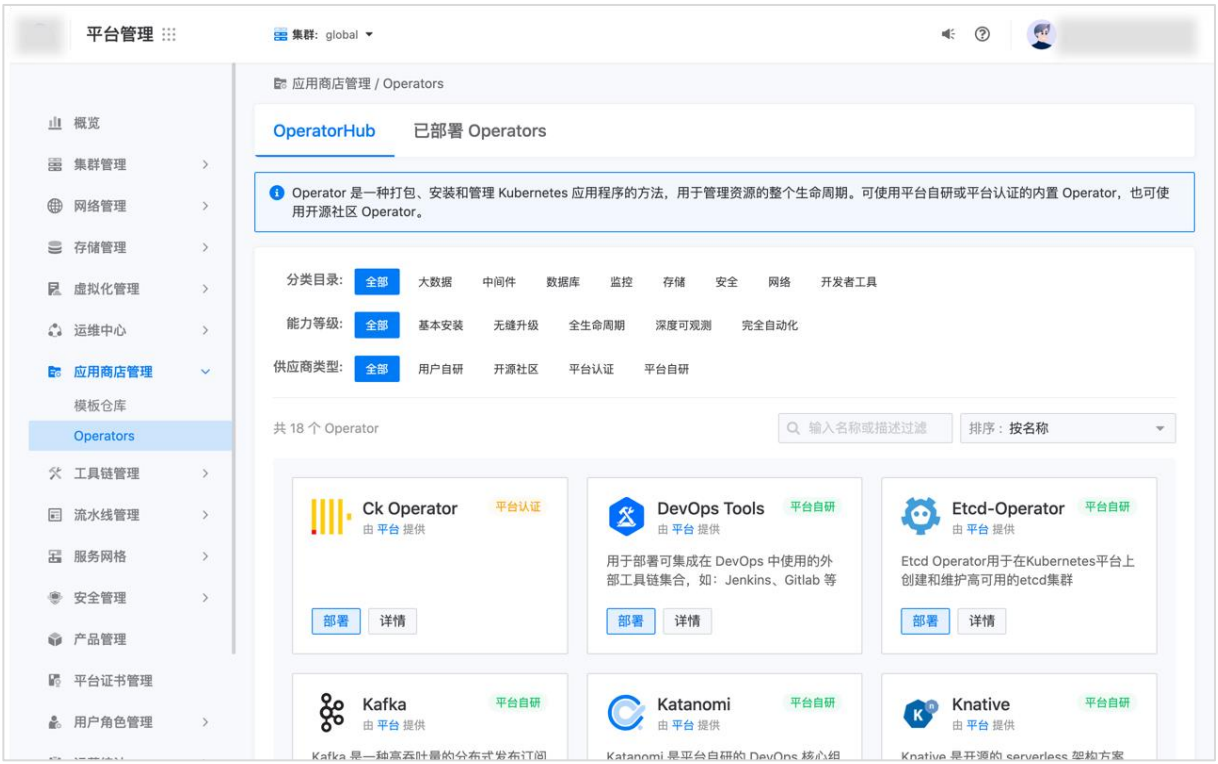

图 4-11 OperatorHub

- vGPU 支持

通过 Kubernetes 的 GPUManager、GPUScheduler 模块，支持对接、调度节点上的 GPU 设备资源。

在使用者创建计算组件时，支持设置所需的 vGPU 核心数和显存配额；在计算组件运行过程中，支持基于 vGPU 利用率、显存利用率指标，设置自动扩缩容。

- 开发测试

通过镜像启动容器应用，能够快速地部署开发、测试环境。同时，使用相同的镜像能保持开发、测试、运维环境一致，减少错误。

4.4 智能运维

基于平台自身的特性，同时结合 Prometheus / VictoriaMetics 监控和 Grafana 可视化的优势，支持对平台管理的集群、节点、应用、Pod、容器等进行实时监控。支持快捷设置集群、节点、计算组件层面的监控指标告警、日志告警（仅计算组件）、事件告警（仅计算组件），也可以根据实际需求自定义监控指标算法，增加需要的告警指标及规则。

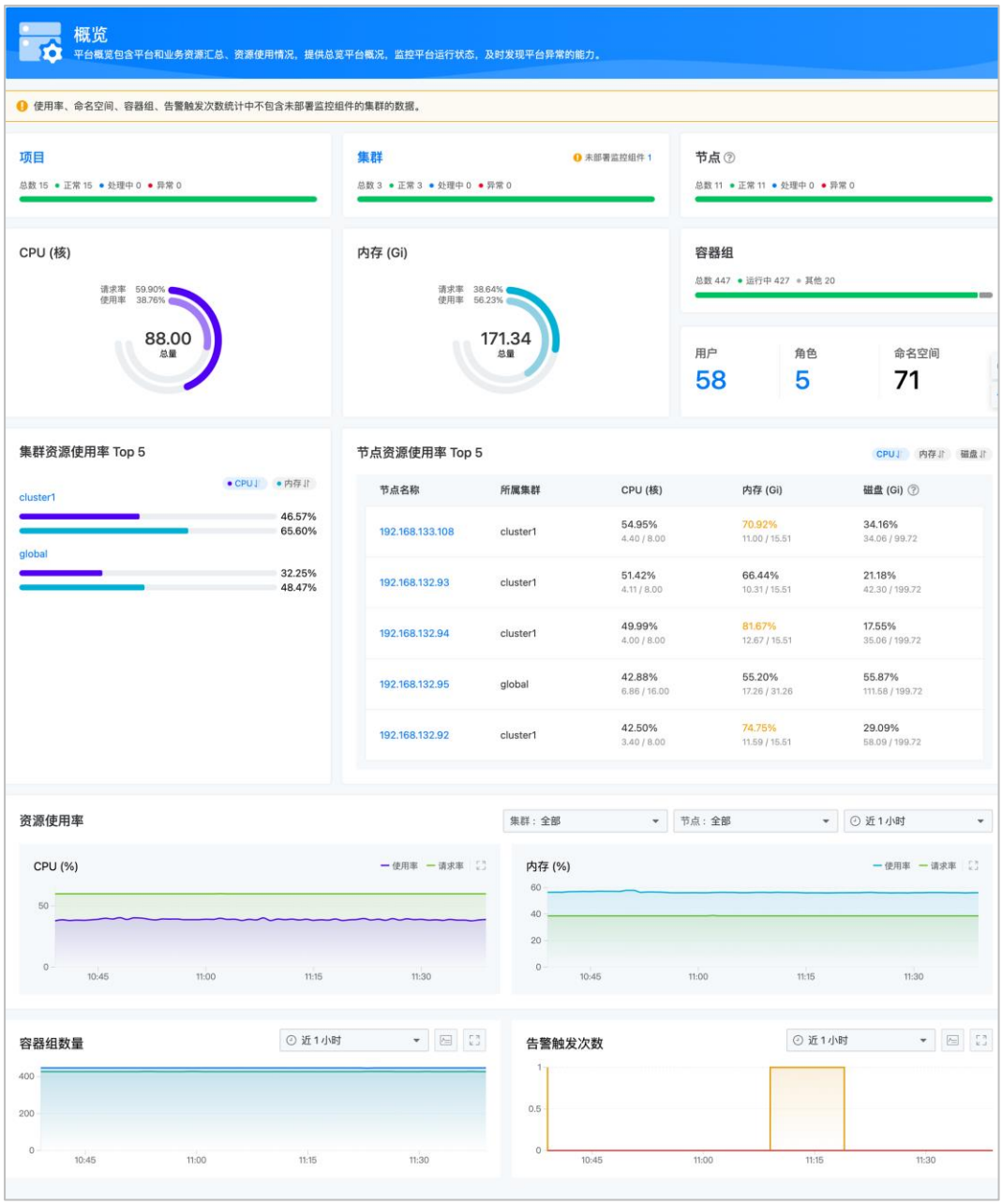

图 4-8 监控概览

图 4-9 实时告警

可通过配置通知策略及时将告警信息发送给运维人员，及时感知系统异常避免系统故障，减少系统运维成本，保障系统的稳定性。

全面集成 Kubernetes 的审计和事件，可以查看平台上所有的用户操作信息（审计）和 Kubernetes 事件信息。支持查看平台上所有资源的日志信息，包括容器内标准输出的日志和容器内指定文件内记录的日志，能够帮助用户快速地排查和解决问题。

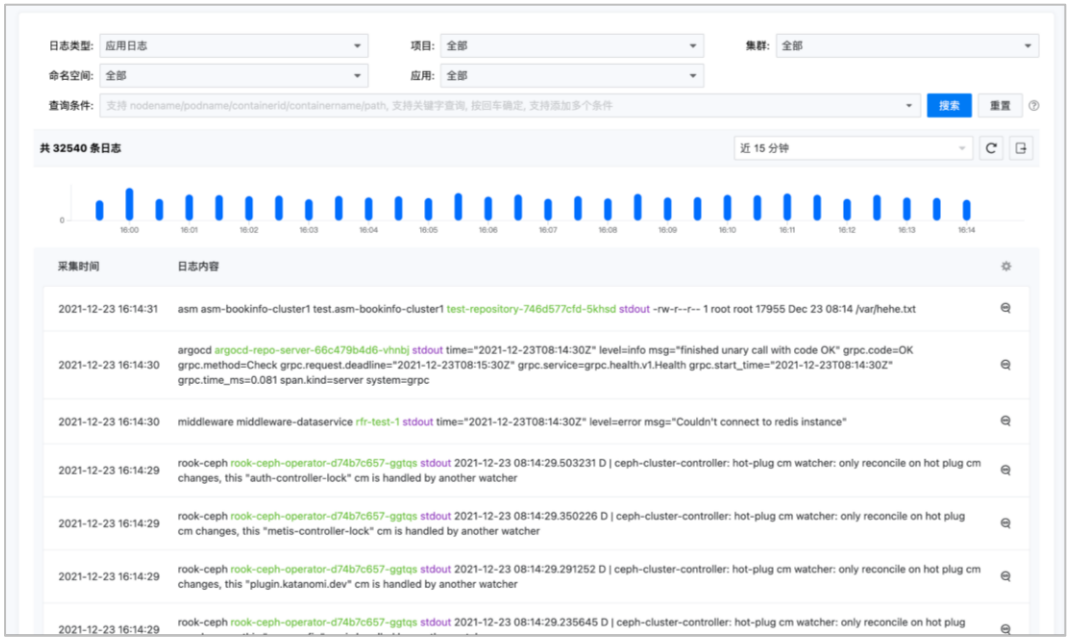

图 4-10 日志

为帮助企业客户降低人工巡检的成本，平台的巡检功能基于为企业客户执行人工巡检的经验设计。

- ✧ 支持在线执行巡检任务，包括资源风险巡检和资源用量巡检，实时获取巡检进度；
- ✧ 巡检结束后，可视化展示巡检结果，包括资源风险、资源用量信息；
- ✧ 支持下载 PDF 或 Excel 格式的巡检报告；
- ✧ 为保障客户数据安全，仅允许具有相关访问权限的用户使用巡检功能。

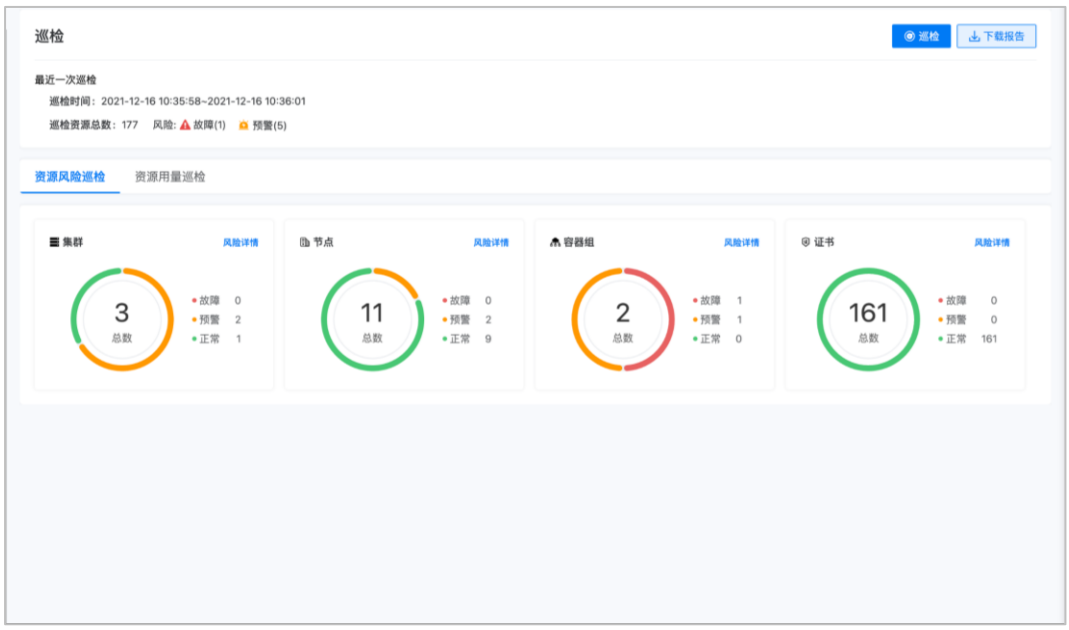

图 4-11 巡检

4.5 持续交付

- 流水线

流水线是一个自定义的 CI/CD 流水线模式，定义了包含构建、测试和发布的完整构建过程。自动化持续交付流水线涉及到代码管理、构建、代码质量分析、部署、发布、通知等环节。平台流水线功能基于 Jenkins 进行构建和部署，支持多种流水线创建方式：图形化创建流水线，模版创建流水线，多分支流水线，脚本创建。

- 流水线模板

平台内置了基于常用开发场景的官方流水线模板，例如：Java、Golang、Python 构建和部署、同步镜像、通知、自动化测试等。尽管不熟悉 CI/CD 工具，仍可直接使用平台提供的多种流水线模板创建流水线，满足更多不同企业的工作场景和特有的业务需求。

- 触发器

触发器是 DevOps 自动化流程中最关键的一环，它涵盖了开发迭代过程中的大部分自动化场景，支持的触发器如：代码仓库触发器、制品触发器、定时扫描触发器和定时触发器。

- 持续构建

目前，DevOps 流水线是完全基于 Jenkins 实现，随着云原生技术的发展，现有 Jenkins 架构已无法满足各种复杂业务场景。为此，平台基于云原生开源框架 Tekton 为 DevOps 平台重新定义了持续构建流程，以满足在不同项目、不同命名空间下的构建需求。

平台践行 GitOps 的理念，将 YAML 定义的流水线编排文件通过 Git 进行统一的版本管理，开发人员可以使用熟悉的工具、熟悉的语言定义自己的流水线内容，做到 Pipeline as Code，降低学习成本和流水线的迁移复用成本。

- 持续发布

平台支持基于 Tekton 的持续发布流水线管理，支持通过 YAML 与图形化表单结合的方式定义应用在不同环境间的更新部署和验证流程，同时结合人工审批、质量门和透明的过程信息，确保整个应用在不同环境中持续在可控的状态下顺畅地自动化部署。

4.6 微服务治理

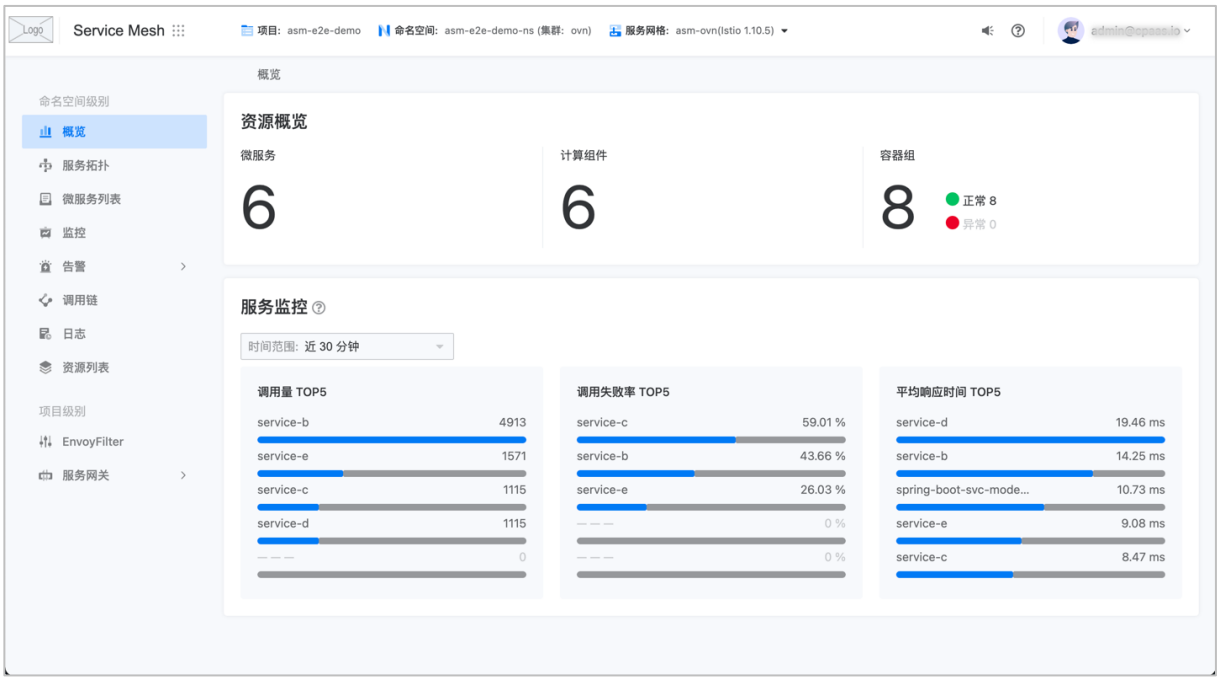

图 4-12 微服务治理平台概览

- 服务全局可视化

支持可视化查看服务之间的调用关系、流量信息、调用链信息。

平台提供统一的微服务管理视角，用户通过可视化表单对服务进行负载均衡、连接池管理、服务熔断、服务安全等服务治理策略的配置。

满足企业用户随时掌握微服务治理全局的需求，有效降低 Istio 实践的门槛。

✧ 服务拓扑

服务拓扑图展示命名空间中微服务的调用关系，可从全局聚焦到局部，直至定位查看某服务的详细信息，如服务状态、请求情况、流量信息等。

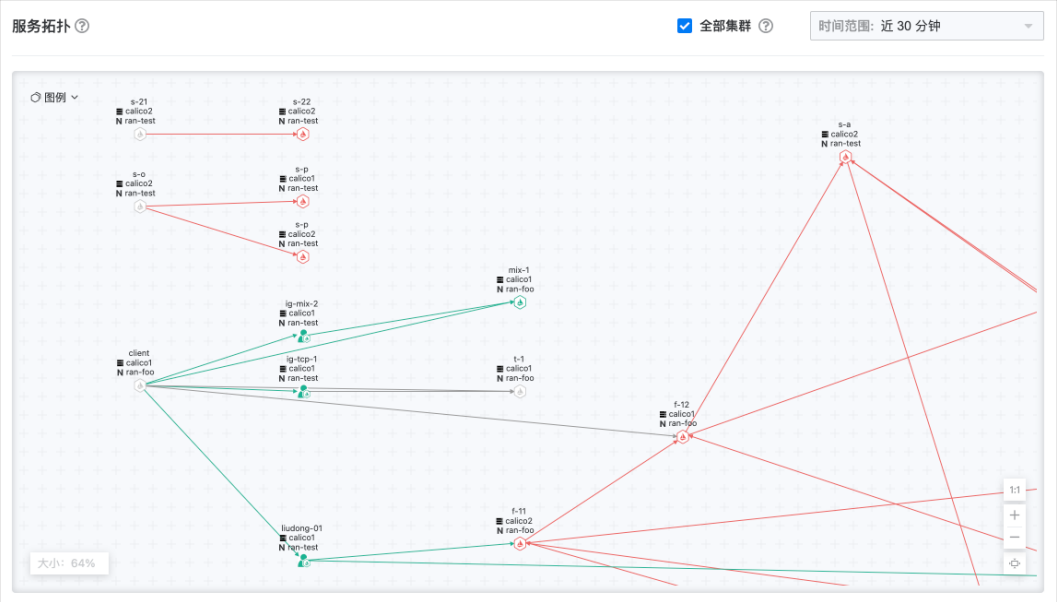

图 4-13 全局拓扑

✧ 调用链追踪

开发人员和运维人员可通过调用链功能对服务间的调用进行下钻分析。

-支持气泡图展示，以便用户快速定位调用性能较低的链路；

-支持全链路跟踪；

-支持通过 TraceID 快速定位调用支持；

-支持指定 TraceID 的链路信息查询。

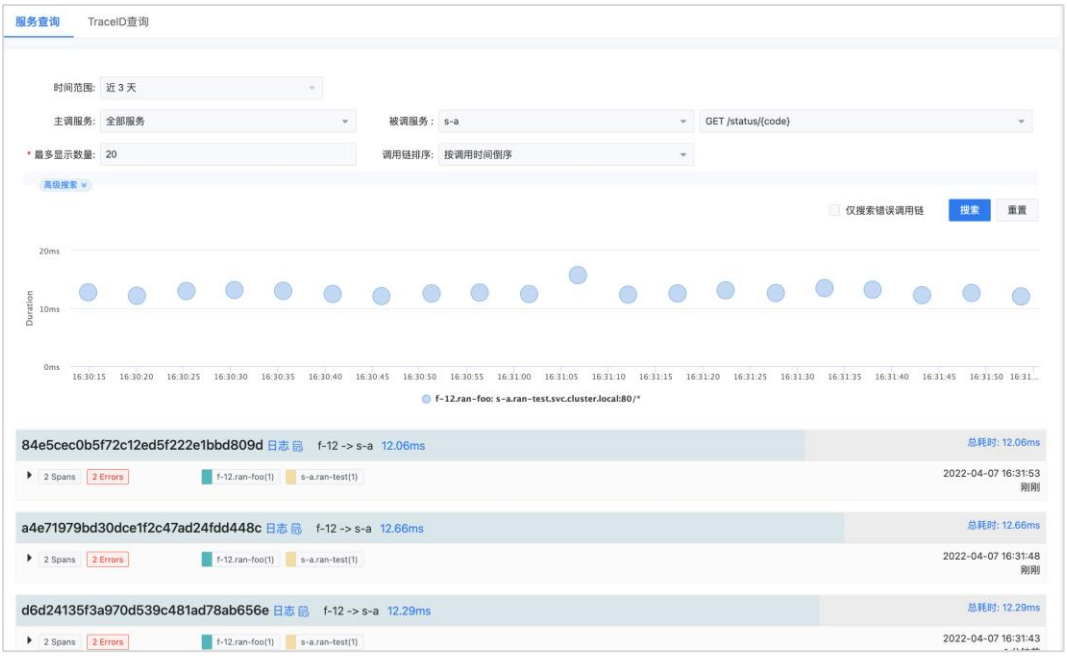

图 4-14 调用链

✧ 微服务列表

支持列表展示不同治理框架的微服务。

✧ 服务路由

所有访问微服务的流量均可以进行服务路由管理。

服务路由支持多组条件规则对流量进行权重分配，条件规则支持 uri、scheme、method、headers、port、sourceLabels 与多种匹配方式。

✧ 微服务网关

支持为服务网格配置多个入口或出口网关，每个网关允许通过多个端口访问。其中，入口网关支持通过不同的路由场景将外部流量路由到不同的后端资源（API 分组、微服务）；出口网关可实现网格内服务对外部服务的访问管理及流量监控。可满足业务区域之间网络隔离、业务独占网关、端口隔离等丰富的使用场景。不再需要的网关，可轻松无损下线。

除此之外，可基于网关的黑白名单和 JWT 对发起请求的客户端进行身份认证与鉴权，并根据请求路径、请求 Header、URI 重写等多样化的方式配置路由规则；支持实例级别的网关监控，配合多种内置告警策略，可帮助用户及时发现、解决问题。监控数据涵盖了 CPU、内存、QPS、连接数、传入流量、传出流量等指标。

✧ 原生资源列表

方便平台高阶用户查看 Istio 原生资源（资源 YAML 信息），可按资源类型分类查看。

- 服务发布管理

基于动态流量策略，实现服务在多版本间的自动化灰度发布。

✧ 金丝雀发布

创建灰度发布时，对服务的流量进行简单的发布规则配置。发布时，将全程按照发布规则自动化执行发布、流量观测、流量异常回滚。

图 4-15 金丝雀发布记录

图 4-16 发布记录版本对比

- 服务连接可靠性治理

多个微服务之间通过 HTTP/HTTP2、gRPC 等方式远程完成数据交互。在调用链路中，当某个微服务响应时间过长或者服务不可用时，对前序微服务的调用就会占用越来越多的资源，进而引起系统崩溃的“雪崩效应”。

为了避免这种状况，需要对微服务之间的调用制定保护机制和故障模拟，从而保证服务连接的可靠性，提升微服务系统整体的稳定性。

✧ 流量策略

-**负载均衡**：调用微服务时，根据指定的负载均衡策略自动在服务实例之间进行流量分配。

-**熔断策略**：对服务实例进行异常检测，无感化处理异常服务，自动隔离异常实例，减少对下游服务的影响。

-**连接池设置**：通过连接池设置，可防止流量太大时导致系统崩溃。连接池中的参数设置作用于客户端的单个实例上。

✧ 路由规则策略

微服务治理平台提供了在运行时动态配置的故障恢复和故障注入功能，支持错误注入、延迟返回、超时重试、请求重写、流量复制。通过使用这些功能可以辅助服务可靠地运行，防止局部故障级联到其它服务。

- 服务网格全生命周期管理

在多集群环境中，服务网格的管理变得尤为复杂。服务网格的全生命周期中不仅涉及到服务网格的部署、更新、删除，更重要的是对服务网格的监控和健康状态的查看，使运维人员能够及时发现异常，排查故障，从而保证服务网格对业务的支撑是连贯、稳定的。

✧ “多主”集群服务网格管理

“多主”集群架构的服务网格管理功能将部署微服务治理平台组件的流程可视化。管理员只需配置主要参数，可以方便快捷的执行部署服务网格、修改服务网格参数等操作。

管理员为每一个需要微服务治理的 Kubernetes 集群，部署包括 Istio 在内的所有微服务治理平台组件，并接入必要的外部组件，使集群具备容器化微服务治理的能力。

✧ 服务网格监控

服务网格的监控功能内置多套监控面板，为管理员提供多维度、多粒度的的控制平面监控和数据平面监控数据查询、可视化功能。

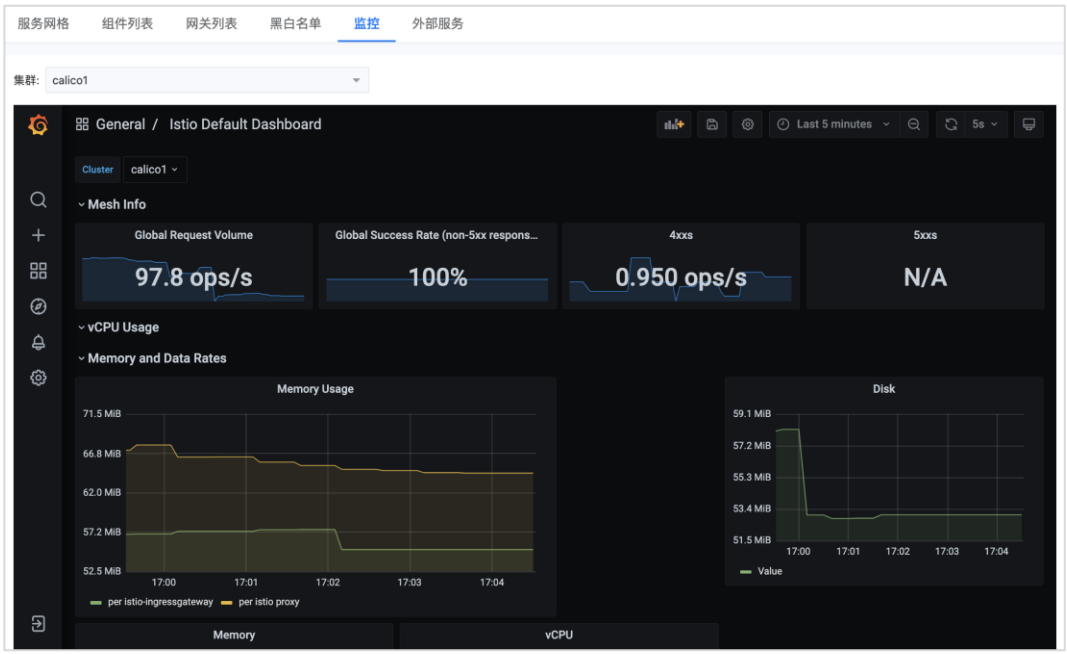

图 4-17 服务网格监控

- 组件管理

  管理员可根据实际业务规模自定义网格组件的资源配置和扩缩容模式（手动、弹性）。

- 服务错误排查

  当用户的服务发生故障时，通过对异常流量的追踪，可定位到故障引起的位置，从而缩小排查范围，分析具体原因。基于此，平台搭建了完整的服务问题排查路径，帮助客户基于异常流量迅速定位可疑位置，保障问题的快速解决。

  ✧ 流量监控和 JVM 监控

  提供客户端、服务端服务的多维度（服务、API）的流量监控数据面板，监控颗粒度可以细化到 API 级别。

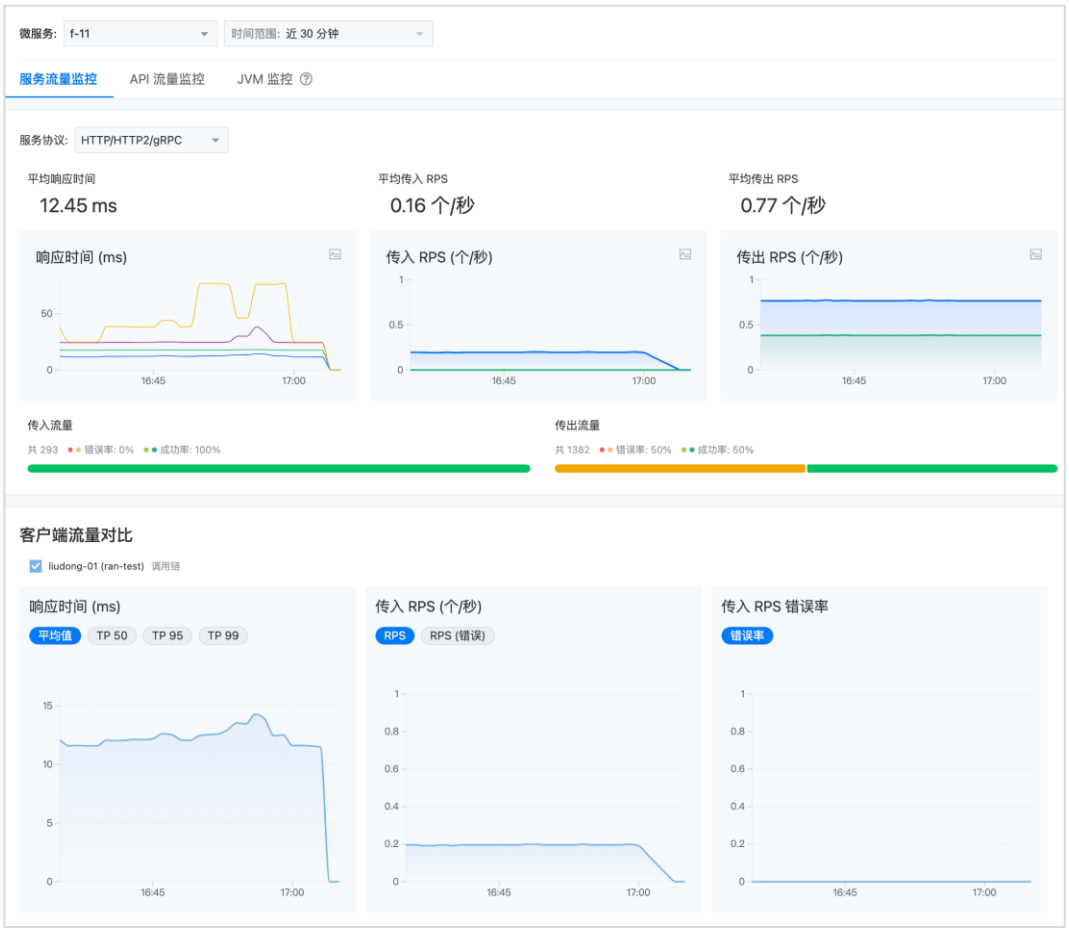

图 4-18 服务流量监控

JVM 监控支持查看单/双实例的监控数据，支持 CPU/内存、线程、类等多种指标的采集与展示，能够帮助开发者快速定位具体问题的指标项和时间点。

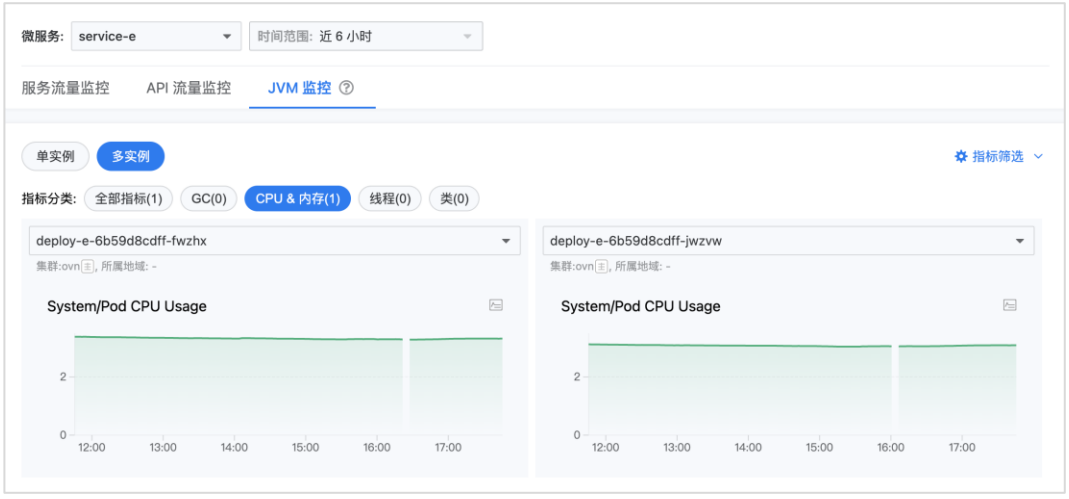

图 4-19JVM 监控

✧ 告警

提供基于常用监控指标的告警功能，支持指标采集、策略管理、告警触发和告警处理等功能，配合流量监控和 JVM 监控，旨在提供完整的“监控-告警”智能运维体验。

提供告警策略配置模版，支持配置单条告警策略或通过模版快速创建告警策略。

支持通过“实时告警”面板总览正在告警的资源信息，可按照告警级别筛选数据，支持查看告警详情，帮助用户掌握问题影响范围并快速定位问题根因。

图 4-20 实时告警面板

✧ 快速排障

平台对服务拓扑、流量监控、调用链、服务日志在产品功能上提供了完整的排查路径，基于异常流量帮助用户快速定位服务故障。

- 服务安全性治理

微服务场景下，服务之间的调用由本地调用改为网络协议接口调用，随之也带来了安全方面的风险，平台提供服务间流量加密功能。

通过服务的流量策略可以设置安全规则，为流量进行 mTLS 加密处理。

- 微服务容灾

多地多数据中心场景下，当某个数据中心的微服务实例发生故障被熔断隔离时，为保证服务端稳定提供服务，平台将自动将客户端的流量分发给其他数据中心的计算组件。

✧ 就近路由

所有数据中心的服务调用将默认使用就近路由。当调用链路节点较长时，可以有效保障容灾流量优先被同集群的服务端处理。

✧ 自定义容灾优先级

可通过配置地域负载，根据集群所在地域自定义容灾流量在多集群之间的容灾负载优先级。

4.7 中间件 PaaS 服务

平台提供开箱即用的中间件 PaaS 服务，包括数据库和消息队列服务。简化构建、部署和运行过程的同时，还能帮助用户更易管理，更自如地使用数据库及消息队列。

- 快速部署

图形化一键部署实例，有效简化实例的创建、管理和释放过程。各中间件支持部署的版本如下。

- ✧ MySQL：5.7、8.0
- ✧ Redis：4.0、5.0、6.0
- ✧ Kafka：2.4.0、2.4.1、2.5.0
- ✧ RabbitMQ：3.8.11、3.8.12、3.8.16
- ✧ MongoDB：4.2
- ✧ PostgreSQL：9.6、10、11、12

图 4-21 创建 MySQL 实例

- 开箱即用

  MySQL

  ✧ 参数配置：支持对运行中的 MySQL 实例进行可视化参数配置，提供作用域、可选值等辅助用户进行参数定义；支持从参数模板或文件导入参数；支持参数导出。

  ✧ 用户管理：支持创建、删除 MySQL 用户，并对 MySQL 中的用户分配相对应的数据库权限。

  ✧ 拓扑：支持查看 MySQL 拓扑，查看节点状态，支持节点切换。

  ✧ 访问方式：支持便捷获取 MySQL 的访问地址，支持配置集群外访问。

  ✧ 查询分析：支持对 MySQL SQL 语句的查询分析与统计，方便数据库使用者与运维人员查看影响数据库性能的 SQL 语句。

Redis

参数配置：支持对运行的 Redis 进行可视化参数配置。

Kafka

- ✧ Topic 管理：支持对 Kafka 的 Topic 管理，创建 Topic 并设置分区数与消息副本数、大小、保留时长。
- ✧ 用户管理：支持对 Kafka 的用户管理，包括创建、编辑及删除 Kafka 用户。
- ✧ 参数配置：支持对运行的 Kafka 进行可视化参数配置。
- ✧ 消费者列表：支持查看消费者列表。
- ✧ 拓扑：支持查看 Kafka 拓扑，查看节点状态。
- ✧ 访问方式：支持便捷获取 Kafka 的访问地址，支持配置集群外访问。

RabbitMQ

- ✧ 控制台集成：集成 RabbitMQ 控制台插件，支持通过控制台管理 RabbitMQ 实例。
- ✧ 参数配置：支持对 RabbitMQ 进行可视化参数配置。

- 备份与恢复
  - ✧ MySQL：支持全量和增量（仅 MGR 架构）的手动和自动备份，一旦出现数据库异常，可恢复已备份数据。
  - ✧ Redis：支持全量的手动和自动备份，一旦出现数据库异常，可恢复已备份数据。
  - ✧ Kafka：支持使用 KafkaMirrorMaker 跨集群生成 Kafka 镜像数据，如遇到系统故障，可使用镜像数据进行恢复。
  - ✧ RabbitMQ：支持在 RabbitMQ 集群中创建镜像队列，若队列的 leader 副本所在节点发生故障，其他节点上的镜像队列副本会被提升为新的 leader。

- 监控与告警

支持通过 Grafana 可视化面板，对实例进行资源、性能等监控，并支持配置自定义告警。开发人员可直接监控当前实例（MySQL/Redis/Kafka/ RabbitMQ/MongoDB /PostgreSQL），运维人员可在平台管理的运维中心集中监控所有实例。

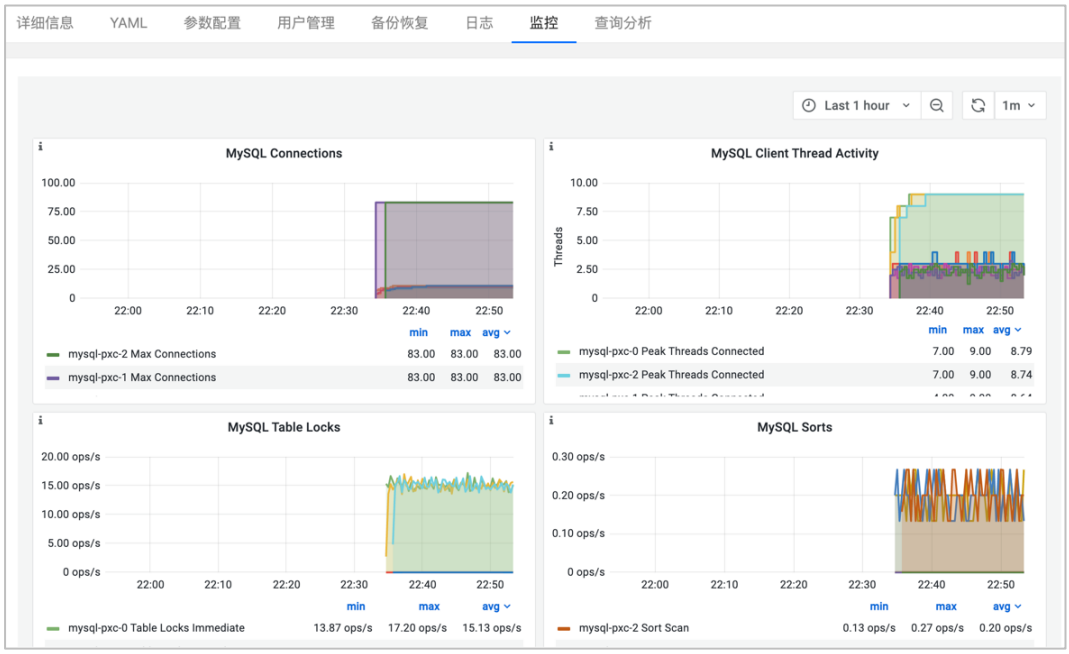

图 4-22 MySQL 实例监控

- 日志管理

从容器层面提供实例运行过程中的日志（例如错误日志），帮助快速定位问题，处理故障和异常。

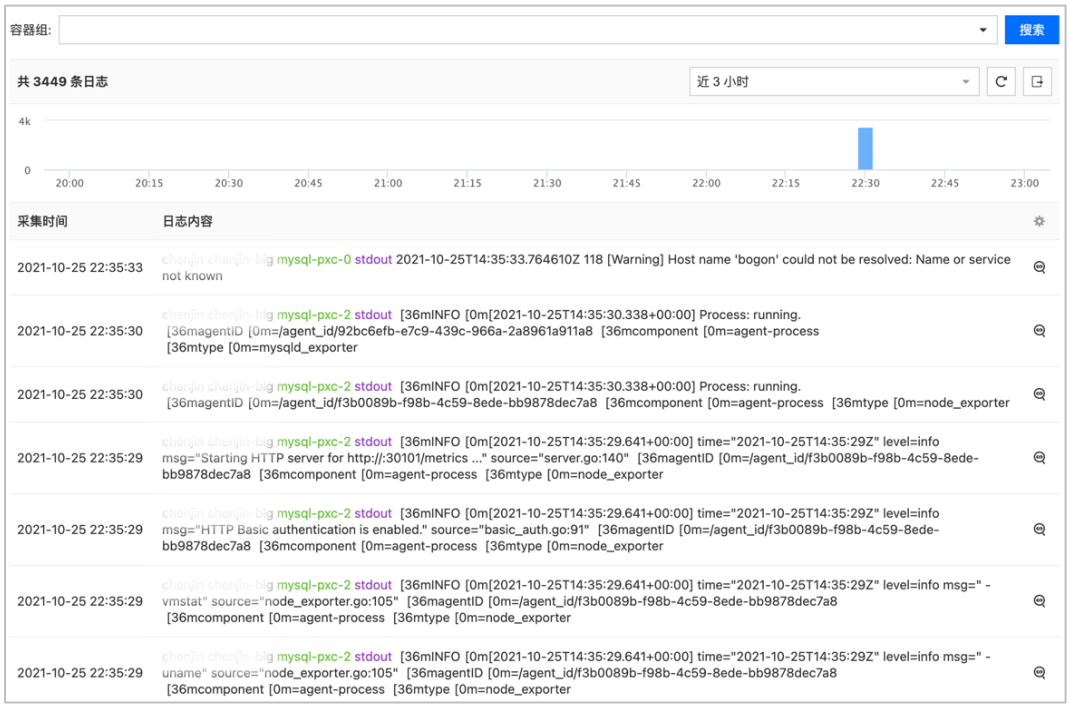

图 4-23 日志管理

5 客户价值

- 快速搭建 PaaS 容器平台

为各领域更多的企业客户快速搭建 PaaS 容器平台将现有的基础设施一键升级成新一代的容器云平台，管理容器的全生命周期。为容器应用创造一个灵活轻便、高效稳定、高可用的资源调度管理平台。帮助企业客户轻松实现数字化转型，获得持续创新的核心能力。

- 持续扩展、集成 Kubernetes 生态工具的能力

在 Kubernetes 越来越流行、成熟的趋势下，必将建立起更全面、更具扩展性的生态圈。平台深度对接 Kubernetes，以 Kubernetes 原生架构作为开发框架，可兼容或集成越来越多的 Kubernetes 生态工具、系统、插件和解决方案，将源源不断的为使用平台的企业提供最前沿、最具竞争力的技术，从而保证企业的核心竞争力。

- 容器化应用的全生命周期管理

实现从测试环境到生产环境的平滑过渡，帮助企业节省环境部署、迁移和运维成本。

- 自动化的容器调度

自动创建 Kubernetes 资源对象。建立贴合业务需求的应用模型以及应用创建流程，将 Kubernetes 资源的创建过程与业务流程完美地结合。

- 专业化的智能运维体验

提供专业的智能运维体系，多维度、全方位的收集日志、监控指标、事件、审计，帮助企业客户提升运维质量，降低运维成本。

- 高效的 IT 治理

基于角色设计权限管理模型，实现多租户间的资源隔离，租户内的配额管理，建立完备的平台租户和权限管理体系，真正帮助企业客户实现高效的 IT 治理。

- 生产级中间件 PaaS 服务

致力于解决中间件部署、管理、运维中的痛点，提供稳定可靠的生产级中间件 PaaS 服务，保障业务连续性，让企业专注于业务，助力企业客户以更低的成本实现中间件上云的解决方案。

- 企业级的微服务治理

平台以无侵入的方式为企业应用及服务提供全方位的治理能力，功能全面覆盖服务全局可视化、服务发布管理、服务连接可靠性治理、服务网格全生命周期管理、服务错误排查、服务安全性治理六大微服务治理场景。

- 助力实现完整的 DevOps 实践

DevOps 在自动化方面提供全面的控制和协调，涵盖基础设施的整个层次结构，深度集成代码仓库、制品仓库、持续集成等类别中的主流工具，将 DevOps 最佳实践自动化并作为服务进行输出，为企业快速实践 DevOps 提供技术支撑，并通过企业级 Kubernetes 容器化应用的全生命周期管理，大幅提高了交付质量和交付速度。

- 将安全融入 DevOps

DevSecOps，在应用的整个生命周期内确保安全性。实现安全防护自动化，以保护整体环境和数据；同时实现持续集成/持续交付流程，并确保容器中应用的安全性。实现用户身份和访问控制功能的集中化。支持集成适用于容器的安全性扫描程序。

- 持续提升 IT 运营能力

以质量和安全为基础支撑保障，覆盖从需求到部署上线的软件全生命周期管理。将线下 IT 生产过程转变为线上高度自动化、可视化的 IT 生产流水线，提升产品研发效率，快速响应业务需求，持续提升 IT 运营能力。