Apusic 金蝶天燕

生成时间：2026-05-19

用户手册

金蝶Apusic中间件云平台v8.0.4

版权声明

本文档所涉及的软件著作权、版权等知识产权已依法进行了注册，由金蝶天燕云计算股份有限公司合法拥有。受《中华人民共和国著作权法》《计算机软件保护条例》《知识产权保护条例》和相关国际版权条约、法律、法规以及其它知识产权法律和条约的保护。未经授权许可，不得非法使用。

免责声明

本文档包含的版权信息由金蝶天燕云计算股份有限公司合法拥有，受法律的保护，金蝶天燕云计算股份有限公司对本文档可能涉及到的非金蝶天燕云计算股份有限公司的信息不承担任何责任。在法律允许的范围内，您可以查阅并仅能够在《中华人民共和国著作权法》规定的合法范围内复制和打印本文档。任何单位和个人未经金蝶天燕云计算股份有限公司书面授权许可，不得使用、修改、再发布本文档的任何部分和内容，否则将被视为侵权，金蝶天燕云计算股份有限公司有依法追究其责任的权利。

本文档如有更新，不另行通知。对本文档中的问题您可向金蝶天燕云计算股份有限公司告知或查询。未经本公司明确授予的任何权利均予保留。

商标声明

Apusic是深圳市金蝶天燕云计算股份有限公司向中华人民共和国国家商标局申请注册的注册商标，注册商标专用权由金蝶天燕合法拥有，受法律保护。未经金蝶天燕的书面许可，任何单位及个人不得以任何方式或理由对该商标的任何部分进行使用、复制、修改、传播、抄录或与其它产品捆绑使用销售。凡侵犯金蝶天燕商标权的，金蝶天燕将依法追究其法律责任。本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

目录

1 修订说明

本文根据实际情况进行更新，最新版本包含历史修改记录。

日期	手册版本	适用产品	更新说明
2024年12月	V8.0.3	ACP v8.0.3	调整格式，细化手册内容
2025年12月	V8.0.4	ACP v8.0.4	内容调整
2026年4月	V8.0.4	ACP v8.0.4-20260427 +	简化安装步骤

2 简介

金蝶Apusic中间件云平台（Apusic Cloud Platform，简称“ACP”）是一款面向云上中间件服务供应、管理、运维与运营全场景的中间件PaaS平台。 ACP能够基于云基础设施提供全栈信创中间件服务，并通过开放的中间件服务软件定义模型帮助云服务提供方构建中间件服务生命周期管理能力， 支持中间件PaaS平台的服务供应、管理、运维与运营，从而支撑专有云、混合云等模式下的PaaS核心能力建设，帮助政府和企业加速构筑业务应用的数字化底座，让应用上云更简单、更便捷。

3 功能清单

ACP v8.0.4 版本产品功能清单如下表所示：

一级功能	二级功能	功能名称	功能描述
资源中心	主机管理	主机管理	实现基础设施资源统一接入，支持物理机、虚拟机、云主机的接入、监控、告警与管理；
		远程终端	支持通过远程终端进行资源管理；
		代理管理	支持代理管理，包括代理版本查看、代理升级；
		区域管理	支持区域、可用区的管理；
		云账号管理	支持私有云公有云的云账号管理及数据同步；
		标签管理	支持多标签管理；
中间件管理	服务管理	服务管理	负责中间件服务全生命周期管理，包括服务的创建、启动、停止、重启、删除操作；
		服务监控	支持服务状态实时监控、性能指标采集，以及服务异常自动告警。
		服务访问	支持服务访问；
		服务回收	支持服务回收；
	实例管理	实例管理	支持实例的全生命周期管理，包括启动、停止、重启；
		实例日志	支持实例日志查看、下载；

		实例监控	跟踪实例运行状态，监控实例 CPU / 内存 / 磁盘资源占用
		实例配置	支持配置版本级管理；
		实例升级	支持补丁升级、制品包升级；
		实例运维	支持实例端口监听、事件管理、环境变量管理、数据卷管理、健康检查；
	软件管理	软件管理	管理中间件软件版本库，支持软件、版本、部署模式的多层级管控；
		部署编排	支持部署流程编排；
		配置库管理	支持软件配置库管理；
		软件分类	支持软件分类管理；
	制品管理	制品管理	支持中间件制品包的上传、存储、版本控制与全链路溯源；
		制品项目	支持按项目分类管理制品；
		制品制作	提供可视化制品包制作流程，降低打包操作门槛；
		配置母版管理	内置制品包配置母版管理功能，沉淀标准化配置模板，支撑部署编排快速复用；
		补丁管理	覆盖补丁包全生命周期管理，支持补丁管理，提供补丁规范，提供包括补丁上传、下载、查看删除等功能。
	扫描发现	特征库管理	支持中间件特征库的管理；
		扫描发现	基于特征库实现自动扫描发现，精准识别实例类型、版本号、部署位置及实时运行状态；

		实例纳管	支持对扫描发现的未纳管实例一键纳入平台管理体系，实现后续的集中化监控、配置下发与全生命周期运维；
	变更管理	变更管理	管控中间件配置变更、版本升级的操作记录；支持变更失败的原因查看；
		变更回滚	记录变更详情，实现追溯，支持变更失败一键回滚；
监控告警	指标监控	指标监控	支持对中间件的指标监控，通过监控任务采集指标数据，支持可视化界面配置监控任务，包括采集端点，采集路径，采集周期配置等。
		监控模板配置	内置采集指标项和告警模板，支持自定义指标项，配置告警模板。
		指标图表面板	支持指标可视化展示，可配置多套图表面板，自定义图表筛选条件，支持图表数据动态刷新。
	日志监控	日志配置	支持二进制应用日志，Docker日志， Kubernetes容器节点日志等多种服务日志；系统内部内置采集器安装，支持数据源配置对应的解析规则；
		日志解析	支持使用正则解析，JSON解析， 数值型字段转换，时间戳识别等解析器解析日志；支持对MySQL, Redis, AAS, Zookeeper, Keepalived, Lvs, Nginx, Json等格式的日志进行解析；
		日志分析	支持日志信息查找，模糊查找，根据字段匹配查找，支持Apache Lucense查询语法；支持日志根据解析规则解析，可以添加多字段进行可视化展示；支持日志事件计数，时间分段， 数值分段，字段值分类等功能；
	告警	告警策略	支持告警触发器配置管理，基于指标阈值设定告警触发条件；支持告警级别设置管理，包括普通，警告，严重，灾难等级别或自定义级别管理，提供各级别的告警历史事件列表查看及统计；
		告警模板	内置告警模板，支持根据指标阈值自定义配置告警模板，支持基于模板快速创建告警任务。
		告警事件	支持告警事件认领，关闭等操作，告警事件认领后自动生成工单，支持告警处理过程记录；支持告警一键关闭，自动恢复；
		告警通知	支持按照告警级别，告警状态设置通知策略；支持通知时间，通知方式，通知人自定义；支持多级通知，轮询通知；支持阿里云SMS、腾讯云SMS、金蝶云SMS、华为云SMS等短信通知方式；
平台运营	租户管理	租户管理	支持租户管理，包括租户账号创建，删除等。

		项目管理	支持租户下多级项目的精细化层级管理与资源隔离；
		配额管理	支持按租户 / 项目维度配置主机资源配额、权限授权配额，并支持配额动态调整与超限预警；
		角色权限管理	支持租户 / 项目下成员的增删维护及角色自定义配置，实现成员与角色的权限精准映射与管控。
	授权管理	文件授权	支持文件授权与集中授权双模式管控；文件授权支持授权文件的上传、下发、下载、删除全流程操作，支持下发记录追溯，且支持授权的回收与二次下发；
		集中授权	集中授权采用授权中心模式，支持通过授权特征码导入授权信息，可实时查看授权关联的在线实例，以及授权的使用状态。
	审计日志	-	支持记录平台用户操作日志，包括操作人，操作项，操作时间等。
平台运维	中间件巡检	-	支持巡检规则，巡检规则组以及巡检任务管理；自动化巡检中间件运行状态、性能指标、配置合规性；生成巡检报告并标记异常项。
	主机巡检	-	支持巡检规则，巡检规则组以及巡检任务管理；巡检中间件所在主机的 CPU / 内存 / 磁盘 ； 检测主机系统参数、进程异常，生成主机健康报告，为主机故障预警提供依据。
	批量操作	-	支持文件的批量上传与下发操作，支持批量操作任务的进度跟踪，以及上传文件的明细信息查询与操作记录追溯。
平台设置	用户管理	-	管理平台用户账号全生命周期（创建 / 编辑 / 禁用）；维护用户信息、密码重置，关联用户所属租户 / 角色，支持用户信息批量导出
	平台角色	-	采用 RBAC 权限模型，支持角色创建、功能权限分配，通过给用户分配角色对用户进行权限限制
	数据字典	-	维护中间件相关标准化数据；支持数据字典的新增 / 修改 / 查询，保障平台数据一致性与规范性
	系统菜单	-	配置平台功能菜单结构；维护菜单名称 / 路径 / 图标，支持菜单启用 / 禁用管理
	系统设置	-	支持基础设置、登录设置、回收策略、漏洞库等配置
	系统组件	-	管理平台依赖的监控 / 存储 / 调度组件；支持组件状态监控管理，保障平台自身稳定运行

4 平台架构

4.1 部署架构

组件说明:

1. maas-alarm
    - 功能:
        - 数据监控：maas-alarm 会从各种监控工具（如 Prometheus、Zabbix、Nagios 等）获取系统、应用、网络等的实时数据，并分析这些数据以确定是否有任何异常。
        - 阈值检测：根据预设的阈值，当某个指标（如 CPU 使用率、内存占用、磁盘空间、服务响应时间等）超过或低于某个阈值时，会触发告警。
        - 复杂规则：支持复杂的告警规则，例如多个条件组合、连续时间段内的问题监控等。
2. maas-aump
    - 功能:
        - 自动化安装：通过脚本或配置管理工具（如 Ansible、Chef、Puppet）实现中间件的自动化安装，减少人为操作，确保一致性。
        - 依赖关系管理：管理中间件所依赖的其他组件和服务，确保安装过程中所需的资源和软件包都已正确安装和配置。
        - 环境配置：根据部署环境（如开发、测试、生产等）配置中间件的参数，如端口号、内存限制、日志存储路径等。
        - 容器化部署：使用容器化技术（如 Docker）来部署中间件，确保在不同环境下的一致性和易于扩展。
3. maas-cloudlog
    - 功能:
        - 多种数据源支持：CloudLog 支持从多种来源收集日志数据，包括服务器、容器、微服务、数据库、消息队列等。它支持常见的日志格式，如 JSON、文本日志、二进制日志等。
        - 日志采集器：CloudLog 提供轻量级的日志采集器，可以部署在各个应用实例或服务容器中，负责将日志数据采集并发送到中心日志系统。
        - 容器与 Kubernetes 集成：CloudLog 通过与 Kubernetes 的集成，能够实时捕获容器中的日志信息，帮助监控和诊断容器化环境中的应用。
        - 分布式日志收集：支持分布式系统中的日志收集，能够跨多个数据中心和云平台进行日志统一管理。
4. maas-auth

- 功能:
    1. 用户身份认证
        - 用户登录与验证：通过用户名、密码、验证码等方式验证用户身份，确保用户是其声称的身份。
        - 单点登录（SSO）：支持跨多个应用系统实现统一认证，用户只需一次登录即可访问所有授权的应用系统，简化用户体验。
        - 角色管理：通过设置不同的角色（如管理员、普通用户、超级管理员等），为不同用户分配不同的访问权限。
        - 权限管理：基于角色、资源、操作等制定精细化的权限控制，确保每个用户只能访问他们被授权的资源。
        - 基于角色的访问控制（RBAC）：通过角色来管理权限，用户被分配角色后，自动获得角色对应的权限。
        - 访问审计：记录每个用户的登录、操作历史，便于进行安全审计和追踪。
        - 用户注册与账户管理：提供用户注册、账号创建、修改用户信息、重置密码等功能，帮助用户管理自己的账户。

5. maas-gateway

- 功能:

    1. 请求路由

    - 路由转发：根据请求的URL路径、请求头、参数等信息将请求转发到不同的微服务或后端系统。网关能够根据配置的路由规则，将不同类型的请求引导到不同的服务实例。
    - 动态路由：支持动态调整路由规则，例如根据负载情况、服务健康状态等自动选择最佳的服务实例进行路由。
    - 路径重写与转发：网关可以根据配置对请求路径进行重写或修改，例如将 /api/v1/* 转发为 /service-v1/*。

        1. 负载均衡

    - 流量分配：根据设定的负载均衡策略（如轮询、最少连接、加权等），在多个后端实例之间分配客户端请求的流量。
    - 自动伸缩：结合容器编排平台（如 Kubernetes）和云服务的能力，网关可以根据后端服务的负载和健康状态进行自动扩展和缩减。
    - 高可用性：通过对多个实例进行负载均衡，确保系统的高可用性，避免单点故障。

6. maas-manager

- 功能:

    1. 业务流程管理（BPM）

    - 流程设计与配置：提供可视化的流程设计工具，使得业务流程可以通过配置来设计与定制，包括审批流程、任务流转、通知机制等。 流程监控与优化：实时监控业务流程的执行状态，分析瓶颈，优化流程，确保业务流畅运行。
    - 自动化与协同：自动化处理一些重复性任务，减少人工干预，提高业务处理效率。同时，支持团队协作，多个角色间的任务分配和协同工作。

        1. 任务与工作流管理

    - 任务分配与跟踪：根据角色、权限和业务规则将任务分配给相应人员，并跟踪任务的完成情况。
    - 工作流审批：管理和处理需要审批的业务流程，如请假申请、合同审批、资金审批等，支持多层级、多角色的审批流程。
    - 任务提醒与通知：系统能够根据任务进度和到期时间自动发出提醒，减少遗漏和延迟。

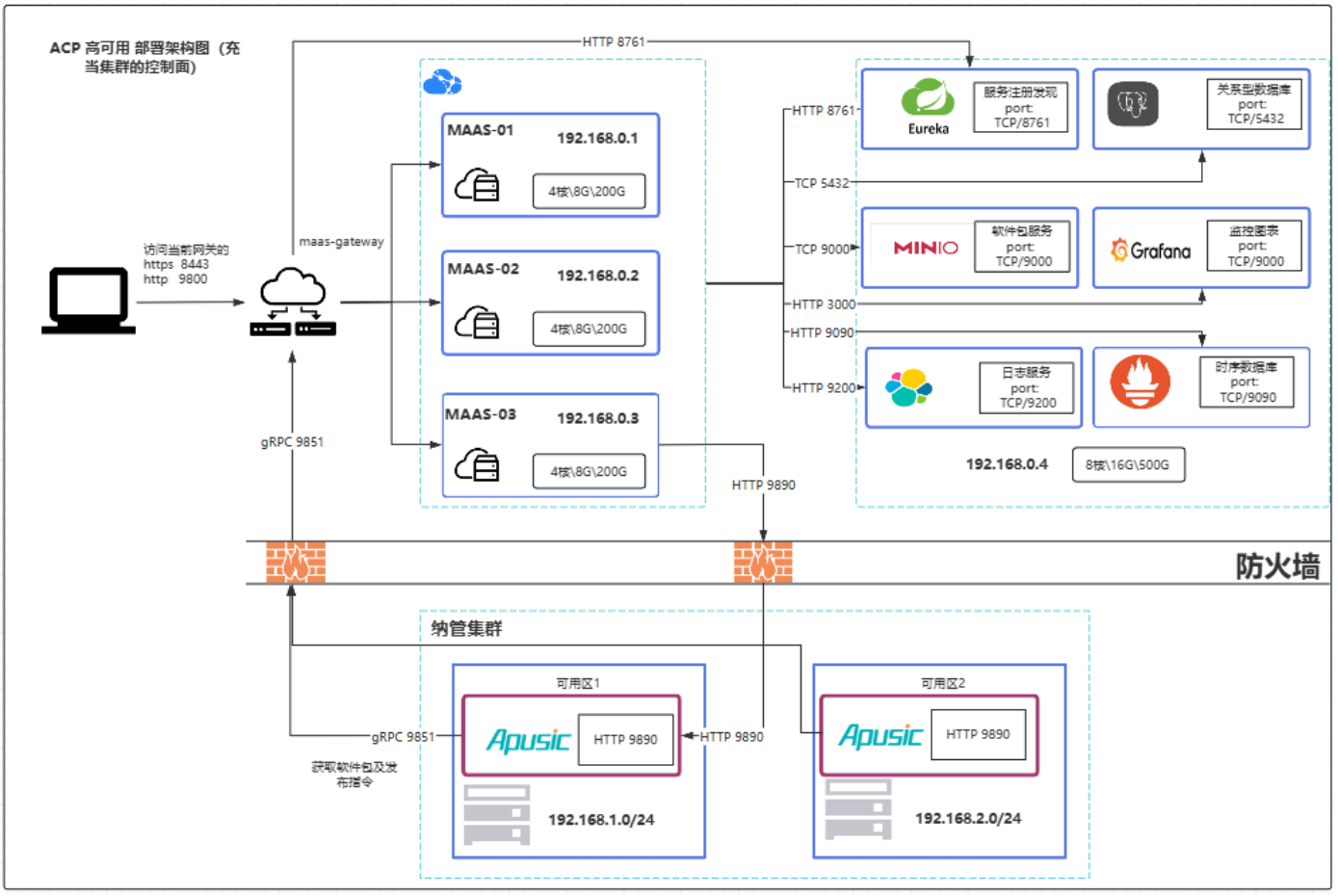
        1. 数据与报表管理

    - 业务数据收集与存储：收集并存储业务过程中的相关数据，例如销售数据、客户信息、订单详情等。
    - 自定义报表生成：根据用户需求，提供自定义的业务报表，支持数据可视化展示，帮助决策者快速了解业务状况。
    - 数据分析与决策支持：分析历史业务数据，帮助公司识别趋势、预测未来发展，并为管理层决策提供依据。

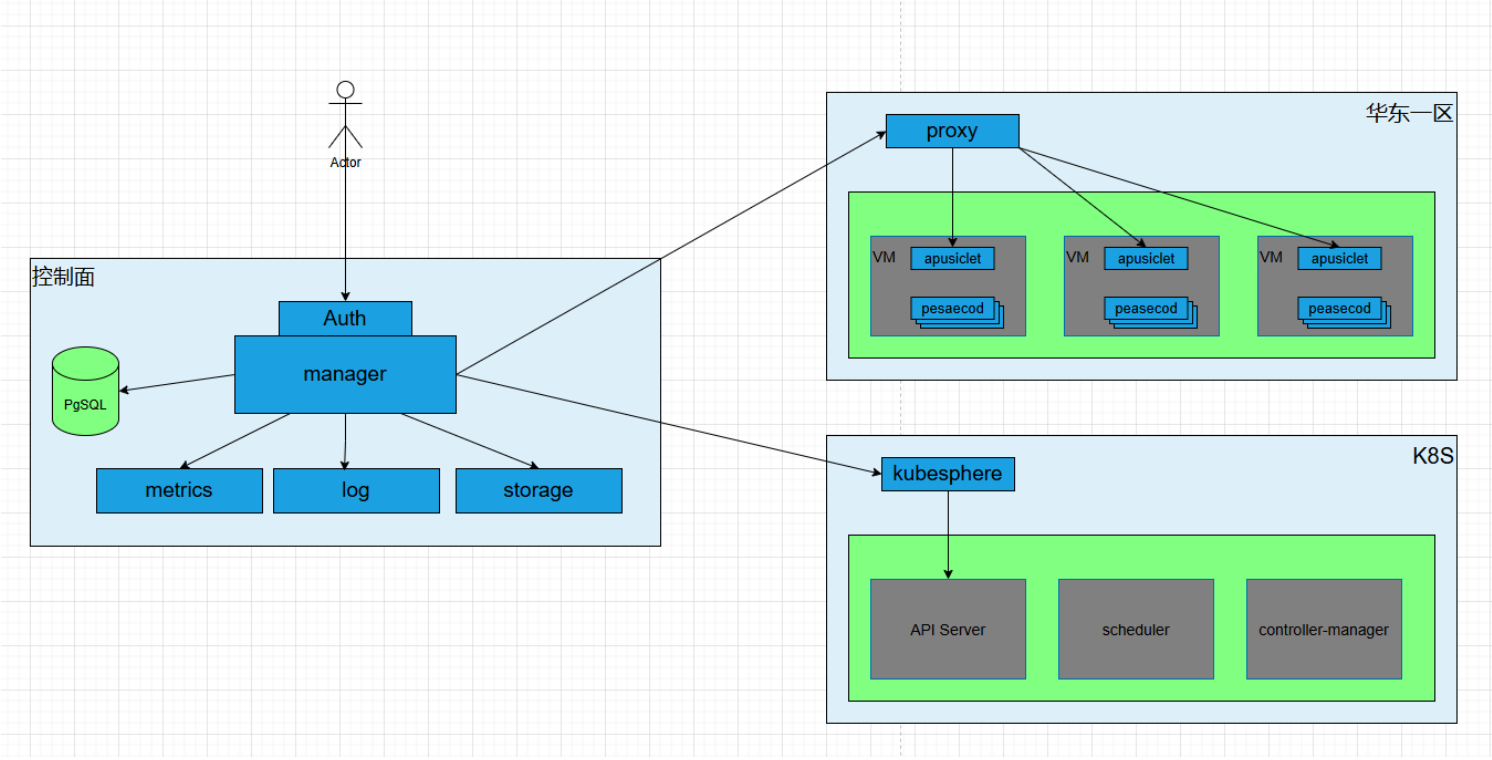
7. maas-resource

- 功能:
    - 云计算平台：选择合适的云计算平台（如 阿里云、腾讯云等）作为云资源提供方，确保与现有架构的兼容性和适配。
    - 虚拟机与容器管理：在云端创建虚拟机实例或容器化的服务来承载应用，支持弹性扩展和负载均衡。
    - 云存储与数据库：对接云端存储服务（如 AWS S3、阿里云 OSS）和数据库服务（如 AWS RDS、Google Cloud SQL），使数据可以轻松迁移、存储并高效访问。
    - 容器服务与编排：使用云平台的容器服务（如 Kubernetes、Docker Swarm等）进行资源管理，确保云资源的自动化部署和扩展。

高可用部署架构图:



4.2 技术架构


Actor
控制面
Auth
manager
PgSQL
metrics
log
storage
proxy
华东一区
VM
apusiclet
pesaecod
VM
apusiclet
peasecod
VM
apusiclet
peasecod
K8S
kubesphere
API Server
scheduler
controller-manager

5 产品安装

ACP支持单机、 **高可用模式** 两种部署方式。单机模式仅限于用于测试、验证，生产环境推荐使用 **高可用模式** 。

高可用部署方案,需要开放的防火墙策略:

序号	源IP地址(发起请求的IP)	发起请求的系统名称	目的IP地址(提供服务的IP)	目的端口	服务协议	提供服务的系统名称	用途
1	192.168.1.0/24	apusiclet	192.168.0.1-3	9800	HTTP	maas-gateway -> maas-aump	获取软件包下载地址
2	192.168.1.0/24	apusiclet	192.168.0.1-3	9851	GRPC	maas-aump	中间件安装及安装事件处理
3	192.168.0.1-3	maas	192.168.0.1-3	8761	HTTP	maas-register	服务注册与发现
4	192.168.0.4-6	maas	192.168.0.1-3	5432	TCP	postgres	关系型数据库
5	192.168.0.4-6	maas	192.168.0.1-3	9000	HTTP	minio	软件包存放目录
7	192.168.0.4-6	maas	192.168.0.1-3	9200	TCP	elastic	搜索引擎
8	192.168.0.4-6	maas	192.168.0.1-3	9090	HTTP	prometheus	指标监控 时序数据库

> *注: `192.168.0.1-3` 分别为 192.168.0.1，192.168.0.2， 192.168.0.4 三个IP，主要的目的是为了部署 maas 高可用集群和 maas所需的中间件。*

5.1 支持的环境

平台类型	系统类型

芯片类型	华为鲲鹏、海思、飞腾、兆芯等X86/ARM架构芯片
操作系统	银河麒麟系列、统信UOS、中标麒麟等
其他Linux系列	RedHat系列、CentOS系列、Ubuntu系列等

5.2 推荐硬件配置

平台提供单机、集群两种安装模式，对应的配置要求，如下表：

部署模式	操作系统	硬件规格 (CPU/内存/硬盘)	服务器台数
单机	Linux	8核/16G/500G	1
集群	Linux	8核/16G/1T	4

5.3 部署前准备

5.3.1 环境要求

软件及操作系统环境要求，如下表：

组件	要求
操作系统	Linux Red Hat 5.2或以上； 国产操作系统如银河麒麟系列、中标麒麟系列、普华、中科红旗、深度等
CPU	Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz或以上； HUAWEI,Kunpeng 920； phytium FT1500a等
浏览器	FireFox 70及以上、Chrome 60及以上、IE 11及以上
JDK	acp平台默认自带JDK(17.0.10+11-LTS-240),若虚拟机中默认设置了 JAVA_HOME 以系统环境变量优先,`unset JAVA_HOME`可以跳过系统JDK

5.3.2 软件要求

数据库

ACP的主要业务数据存储在关系型数据库中。在这种情况下，ACP运行环境需要安装数据库服务，来实时记录授权产品的使用情况，并将数据同步到数据库中。

当前支持MySQL、PostgresQL、瀚高、达梦、金仓等多种类型的关系数据库。

ACP默认使用PostgresQL，PostgreSQL安装可参考【附录-安装PostgreSQL】。

> *提示：生产环境推荐使用商业数据库。*

系统软件

安装之前需要安装以下软件

```
yum install -y wget netstat
```

5.3.3 关于License

1. 使用官方授权的有效期内license文件。
2. 使用金蝶天燕官方授权中心。
3. 使用KBC统一授权中心。

5.3.4 获取安装包

从金蝶天燕官方网站下载金蝶 Apusic 中间件云平台软件安装包，或从金蝶Apusic中间件云平台软件产品光盘中获得相应的安装包文件。

5.3.5 安装介质说明

ACP V8.0.4完整的产品包括如下安装程序文件，不同CPU架构平台请使用对应的产品安装包。若产品介质名称中不包含x86_64、arm64平台架构的字样，则适合跨平台部署，无需区分。

组件名称	文件名	说明
中间件云平台	ACP-MAAS-8.0.4-20250210-Linux-amd64.tar.gz	Web管控台，包括云门户、管控中心、监控中心、日志中心等
授权中心	ACLS-1.1.zip	实现Apusic许可授权中心（Apusic Cloud License Server,简称"ACLS"），便于对中间件产品的授权进行统一管理

5.3.6 依赖详情

组件名称	默认端口	版本	备注
Postgresql	5432	15.6	关系型数据库
alertmanager	9093	0.26.0	监控告警
elasticsearch	9200|9300	7.17.24	日志存储
logstash	5044	8.16.1	日志收集

minio	9000	20210422	对象存储
prometheus	9090	2.37.9	监控服务
grafana	3000	9.5.0	监控图表

5.3.7 服务端口说明

中间件云平台

端口	对应组件	作用	访问范围	服务名称
8443	中间件云平台	反向代理 HTTPS	外部访问	maas-gateway
9800	中间件云平台	反向代理HTTP	外部访问	maas-gateway
9810	中间件云平台	统一认证服务	内部访问	maas-auth
9820	中间件云平台	云门户	内部访问	maas-manager
9830	中间件云平台	监控中心	内部访问	maas-monitor
9880	中间件云平台	告警中心	内部访问	maas-alarm
9840	中间件云平台	资源中心-主机管理	内部访问	maas-resource
9870	中间件云平台	工单系统	内部访问	maas-workorder
9860	中间件云平台	日志中心	内部访问	maas-cloudlog
9850	中间件云平台	中间件管理	内部访问	maas-aump
9851	中间件云平台	中间件管理	外部访问	maas-aump
9801	中间件云平台	前端服务 管控台	内部访问	maas-fe
9890	中间件云平台	agent	内部访问	apusiclet

- 指标监控

端口	对应组件	作用	访问范围	服务名称
9090	指标监控	指标后台管理	内部访问	prometheus
9093	指标监控	告警后台管理	内部访问	alertmanager
9094	指标监控	告警集群间通讯	内部访问	alertmanager

- 日志服务

端口	对应组件	作用	访问范围	服务名称
9200	日志服务	日志引擎API接口	内部访问	elasticsearch
9300	日志服务	日志集群间通讯	内部访问	elasticsearch
5044	日志服务	日志解析	内部访问	logstash

- 存储服务

端口	对应组件	作用	访问范围	服务名称
9000	存储服务	后台管理页w	外部访问	minio
9001	存储服务	api端口	外部访问	minio

- 注册中心

端口	对应组件	作用	访问范围	服务名称
8761	注册中心	服务注册发现	内部访问	maas-register

5.4 部署

5.4.0.1 环境检查

- JDK：避免JDK冲突，建议卸载服务器上已经安装的JDK
- 安装wget、netstat

  保存文件并运行以下命令使配置生效：

```
yum install -y wget netstat fontconfig
```

- 内核参数

  编辑 /etc/sysctl.conf 文件

```
vim /etc/sysctl.conf
```

在文件末尾添加或者修改以下内容：

```
vm.max_map_count=262144
fs.file-max = 65536
```

保存文件并运行以下命令使配置生效：

```
sysctl -p
```

编辑用户资源限制配置文件 /etc/security/limits.conf ， 添加或修改以下内容：

```
  * hard nofile 65536
  * soft nofile 65536
```

> *提示：上述修改需要重新登录才生效*

5.4.0.2 安装前准备

1. 创建非 Root 用户（如 apusic），并拥有其账户权限

```
sudo groupadd apusic
sudo useradd -m -g apusic apusic
# 切换至apusic 用户
su apusic
```

2. 将安装包传输到用户目录（如/home/apusic）

3. 解压安装包

```
tar -xzvf ACP-MAAS-8.0.4-20250210-Linux-amd64.tar.gz
```

5.4.0.3 安装平台组件

ACP平台依赖数据库，监控、存储等组件，支持一键安装平台的所有组件

> *注意：*
>
> - *ACP提供的PostgreSQL数据库仅供测试使用，我们建议生产环境使用商业数据库。*

> - *目前，我们支持的PostgreSQL仅适用于 amd64 架构 Ubuntu、Kylin、Centos、OpenEuler、Redhat，aarch64 架构支持 Kylin、OpenEuler、Redhat、Ubuntu 操作系统，其它自行安装PostgreSQL*

1. 安装平台组件

```
cd /home/apusic/maas-8.0.4
# 安装所有组件
./bin/maas.sh middleware install
```

```
[apusic@linux-4-234 maas-8.0.4]$ ./bin/maas.sh middleware install
 09:51:36.68 INFO  ==> Start installing all middleware components...
 09:51:36.69 INFO  ==> Installing postgresql...
 09:51:36.69 INFO  ==> install postgresql
centos
The files belonging to this database system will be owned by user "apusic".
This user must also own the server process.

The database cluster will be initialized with locale "en_US.UTF-8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are disabled.

fixing permissions on existing directory /home/apusic/maas-8.0.4/service/postgresql/data ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... Asia/Shanghai
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok

initdb: warning: enabling "trust" authentication for local connections
initdb: hint: You can change this by editing pg_hba.conf or using the option -A, or --auth-local and --auth-host, the next time you run initdb.

Success. You can now start the database server using:

    /home/apusic/maas-8.0.4/service/postgresql/bin/pg_ctl -D /home/apusic/maas-8.0.4/service/postgresql/data -l logfile start

 09:51:38.26 INFO  ==> Allow all incoming connections from any IP
 09:51:38.26 INFO  ==> install postgresql success!
 09:51:38.27 INFO  ==> postgresql installed successfully.
 09:51:38.27 INFO  ==> Installing minio...
 09:51:38.27 INFO  ==> install minio
 09:51:38.74 INFO  ==> install minio success!
 09:51:38.74 INFO  ==> minio installed successfully.
 09:51:38.75 INFO  ==> Installing elasticsearch...
 09:51:38.75 INFO  ==> install elasticsearch
 09:51:43.87 INFO  ==> install elasticsearch success!
 09:51:43.87 INFO  ==> elasticsearch installed successfully.
 09:51:43.87 INFO  ==> Installing logstash...
 09:51:43.87 INFO  ==> install logstash
 09:51:50.55 INFO  ==> install logstash success!
 09:51:50.55 INFO  ==> logstash installed successfully.
 09:51:50.55 INFO  ==> Installing prometheus...
 09:51:50.55 INFO  ==> install prometheus
 09:51:52.33 INFO  ==> install prometheus success!
 09:51:52.34 INFO  ==> prometheus installed successfully.
 09:51:52.34 INFO  ==> Installing grafana...
 09:51:52.34 INFO  ==> install grafana
 09:51:54.91 INFO  ==> install grafana success!
 09:51:54.91 INFO  ==> grafana installed successfully.
 09:51:54.91 INFO  ==> Installing alertmanager...
 09:51:54.91 INFO  ==> install alertmanager
 09:51:55.49 INFO  ==> install alertmanager success!
 09:51:55.49 INFO  ==> alertmanager installed successfully.
 09:51:55.49 INFO  ==> Installing license...
 09:51:55.49 INFO  ==> install license
 09:51:56.66 INFO  ==> install license server success!
 09:51:56.66 INFO  ==> license installed successfully.
 09:51:56.66 INFO  ==> All middleware components installation completed.
[apusic@linux-4-234 maas-8.0.4]$ ^C
```

> *提示：当出现"All middleware components installation completed."，则表示安装成功*

2. 启动平台组件

```
# 启动所有组件
./bin/maas.sh middleware start
```

```
[apusic@linux-4-234 maas-8.0.4]$ ./bin/maas.sh middleware start
 09:58:06.16 INFO  ==> Start launching all middleware components...
 09:58:06.16 INFO  ==> Starting postgresql...
waiting for server to start.... done
server started
 09:58:06.27 INFO  ==> start postgresql success!
 09:58:06.28 INFO  ==> postgresql started successfully.
 09:58:06.28 INFO  ==> Starting minio...
 09:58:06.28 INFO  ==> start minio success!
 09:58:06.28 INFO  ==> minio started successfully.
 09:58:06.29 INFO  ==> Starting elasticsearch...
nohup: appending output to 'nohup.out'
 09:58:06.29 INFO  ==> start elasticsearch success!
 09:58:06.29 INFO  ==> elasticsearch started successfully.
 09:58:06.29 INFO  ==> Starting logstash...
 09:58:06.30 INFO  ==> start logstash success!
 09:58:06.30 INFO  ==> logstash started successfully.
 09:58:06.30 INFO  ==> Starting prometheus...
 09:58:06.30 INFO  ==> start prometheus success!
 09:58:06.30 INFO  ==> prometheus started successfully.
 09:58:06.31 INFO  ==> Starting grafana...
 09:58:06.31 INFO  ==> start grafana success!
 09:58:06.31 INFO  ==> grafana started successfully.
 09:58:06.32 INFO  ==> Starting alertmanager...
 09:58:06.32 INFO  ==> start alertmanager success!
 09:58:06.32 INFO  ==> alertmanager started successfully.
 09:58:06.32 INFO  ==> Starting license...
 09:58:06.33 INFO  ==> start license server success!
 09:58:06.33 INFO  ==> license started successfully.
 09:58:06.33 INFO  ==> All middleware components launched successfully.
```

> *提示：当出现"All middleware components launched successfully."，则表示启动成功。*

3. 查看组件状态

```
# 启动所有组件
./bin/maas.sh middleware satus
```

```
[apusic@linux-4-234 maas-8.0.4]$ ./bin/maas.sh middleware status
 11:11:38.16 INFO  ==> Checking status of all middleware components...
 11:11:38.16 INFO  ==> Checking postgresql status...
pg_ctl: server is running (PID: 23267)
/home/apusic/maas-8.0.4/service/postgresql/bin/postgres "-D" "/home/apusic/maas-8.0.4/service/postgresql/data"
 11:11:38.16 INFO  ==> Checking minio status...
MinIO is running successfully!
 11:11:38.18 INFO  ==> Checking elasticsearch status...
 11:11:38.20 INFO  ==> ☑ Elasticsearch is healthy (GREEN)
 11:11:38.21 INFO  ==> Checking logstash status...
Logstash is healthy (GREEN)
 11:11:38.27 INFO  ==> Checking prometheus status...
 11:11:38.29 INFO  ==> Prometheus is healthy
 11:11:38.29 INFO  ==> Checking grafana status...
GRAFANA is healthy (GREEN)
 11:11:38.30 INFO  ==> Checking alertmanager status...
 11:11:38.32 INFO  ==> Alertmanager  is running successfully!
 11:11:38.32 INFO  ==> Checking license status...
 11:11:38.34 INFO  ==> license-server  is running successfully!

+------------+------------------+---------+----------------------+
| Middleware | Health           | Status  | Description          |
+------------+------------------+---------+----------------------+
| postgresql | healthy          | running | Service is running   |
| minio      | healthy          | running | Service is running   |
| elasticsearch | healthy       | running | Service is running   |
| logstash   | healthy          | running | Service is running   |
| prometheus | healthy          | running | Service is running   |
| grafana    | healthy          | running | Service is running   |
| alertmanager | healthy        | running | Service is running   |
| license    | healthy          | running | Service is running   |
+------------+------------------+---------+----------------------+
```

> *提示：当组件状态不是running时，可以在logs目录下查看相关日志*

4.创建数据库及用户

```
# 初始化postgresql
./bin/maas.sh middleware init postgresql
```

```
[root@linux-4-116 maas-8.0.3]# bin/maas.sh middleware postgresql init
Initializing database connection configuration...
Database: test
Username: admin
Password:
CREATE ROLE
ALTER ROLE
CREATE DATABASE
```

> *提示：记住这里填入的数据库名、用户名、密码，后续安装步骤会使用*

5.4.0.4 安装管控台

1. 修改配置文件
    - 修改网关地址

```
cd /home/apusic/maas-8.0.4
vim conf/application-prod.properties

# 如下localhost修改为真实的IP。
apusic.gateway.addr=http://localhost:9800
```

- 修改数据库连接配置（以PgSQL为例）：

```
vim  conf/application-postgresql.properties
# 数据库名称、用户、密码请与安装数据库时输入的参数保持一致。
apusic.datasource.host=localhost
apusic.datasource.port=5432
apusic.datasource.username=root
apusic.datasource.password=password11
apusic.datasource.database=maas-release_v813
```

> *提示：数据库密码如需使用密文方式，需要将apusic.datasource.encrypt.enabled设置为true。同时通过如下命令./bin/maas.sh generate {password}获取密码密文，并进行配置。*

2. 执行启动脚本

```
./bin/maas.sh start all -d
```

出现如下结果，所有模块均为success，表示安装成功：

```
[apusic@linux-4-234 maas-8.0.4]$ ./bin/maas.sh start all -d

 11:18:57.14 后台启动 ...
27394
 11:18:57.18 INFO  ==> start console success ...
 11:18:57.18 start maas application maas-register-8.0.4.jar
 11:18:57.19 启动配置文件为: /home/apusic/maas-8.0.4/conf/application-prod.properties
 11:18:57.19 INFO  ==> start maas-register-8.0.4.jar success ...
 11:18:57.19 start maas application maas-resource-8.0.4.jar
 11:18:57.20 启动配置文件为: /home/apusic/maas-8.0.4/conf/application-prod.properties
 11:18:57.20 INFO  ==> start maas-resource-8.0.4.jar success ...
 11:18:57.21 start maas application maas-manager-8.0.4.jar
 11:18:57.21 启动配置文件为: /home/apusic/maas-8.0.4/conf/application-prod.properties
 11:18:57.22 INFO  ==> start maas-manager-8.0.4.jar success ...
 11:18:57.22 start maas application maas-gateway-8.0.4.jar
 11:18:57.22 启动配置文件为: /home/apusic/maas-8.0.4/conf/application-prod.properties
 11:18:57.23 INFO  ==> start maas-gateway-8.0.4.jar success ...
 11:18:57.23 start maas application maas-aump-8.0.4.jar
 11:18:57.24 启动配置文件为: /home/apusic/maas-8.0.4/conf/application-prod.properties
 11:18:57.24 INFO  ==> start maas-aump-8.0.4.jar success ...
 11:18:57.25 start maas application maas-cloudlog-8.0.4.jar
 11:18:57.25 启动配置文件为: /home/apusic/maas-8.0.4/conf/application-prod.properties
 11:18:57.26 INFO  ==> start maas-cloudlog-8.0.4.jar success ...
```

3. 查看管控状态

```
./bin/maas.sh status
```

若所有组件均UP，则说明安装成功。

```
[apusic@linux-4-234 maas-8.0.4]$ ./bin/maas.sh status
注册中心与服务发现服务(maas-register):运行中, PID: 27409
资源服务(maas-resource):运行中, PID: 27425
管理服务(maas-manager):运行中, PID: 27443
网关服务(maas-gateway):运行中, PID: 27462
中间件服务(maas-aump):运行中, PID: 27490
日志服务(maas-cloudlog):运行中, PID: 27522
```

4. 访问

浏览器访问http://IP:9800，默认用户名密码:acpadm/admin

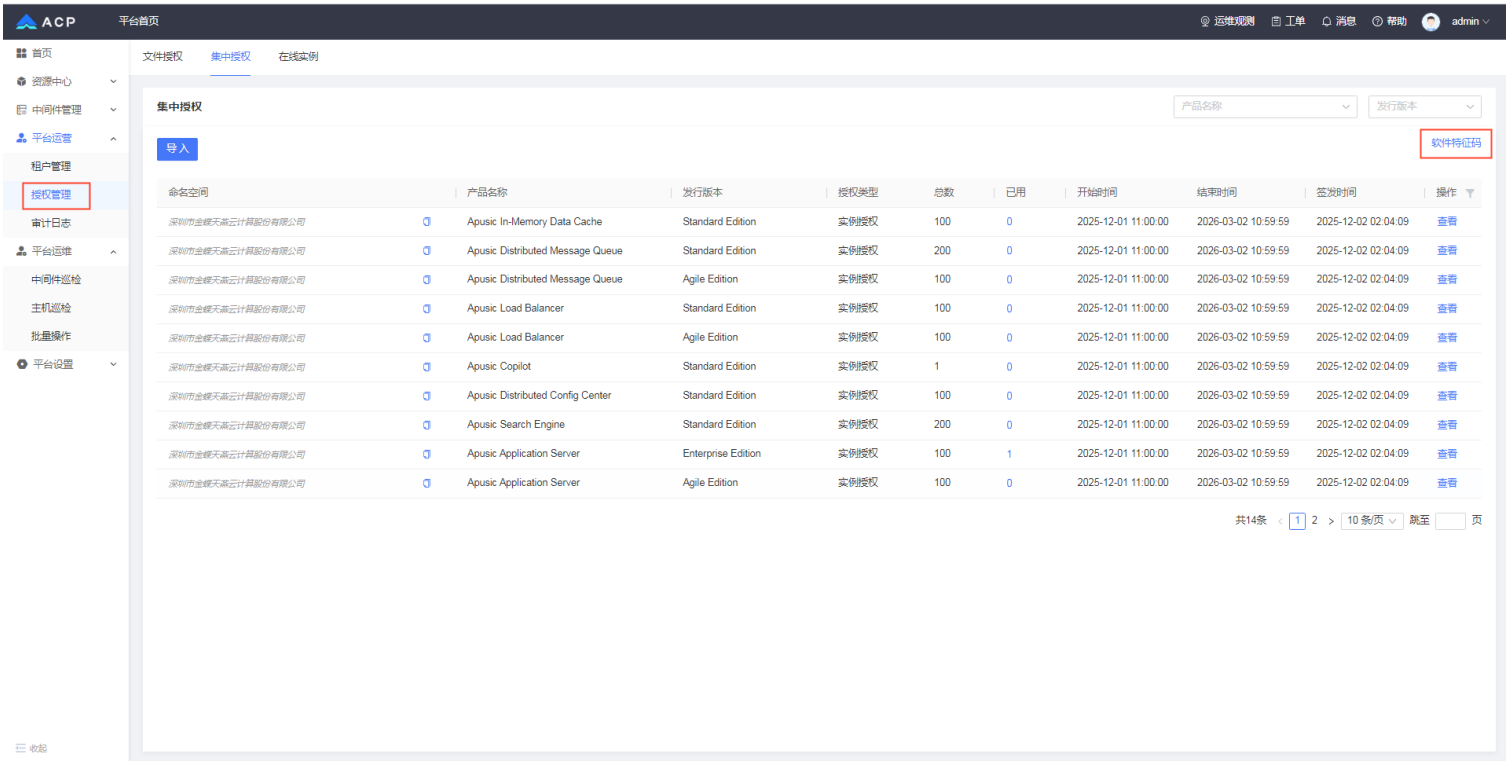
5.4.0.5 授权激活

获取授权

首次访问 ACP 平台需完成授权操作，具体步骤如下：

1.首次登ACP 后，页面弹出“授权校验失败”提醒，点击【跳转授权】按钮，进入平台授权管理页面。

2.在【平台运营】-【授权管理】-【集中授权】页面中，点击【软件特征码】选项，获取当前平台的特征码。

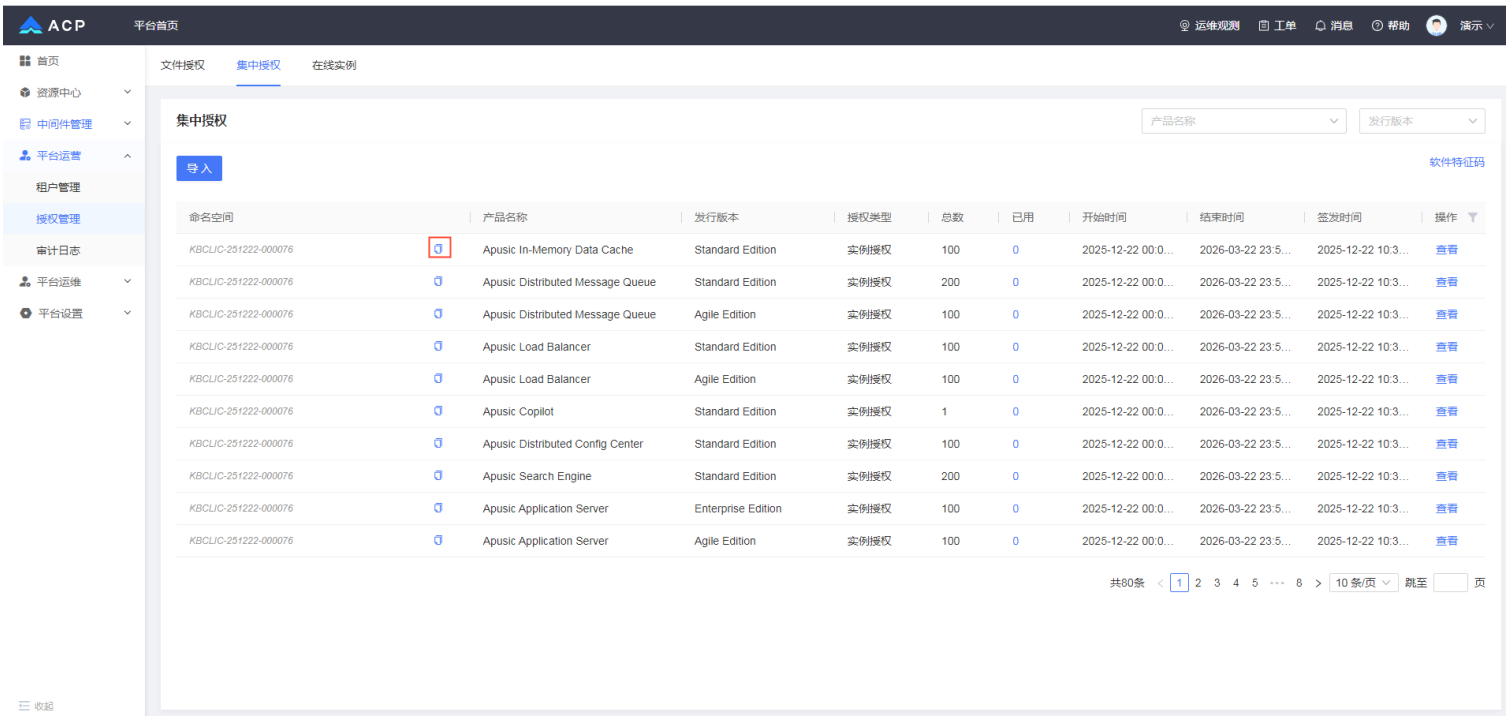


3.联系技术支持人员，1) 提供上述特征码完成平台激活，并获取对应授权。

导入授权

获取授权文件（.lic文件）后，需要完成如下操作：

1.在【平台运营】-【授权管理】-【集中授权】页面中，点击【导入】按钮，导入授权文件后，点击“命令空间”复制按钮获取命名空间



3.修改ACP安装目录下的授权配置acls.properties

```
cd home/apusic/maas-8.0.4
vim acls.properties
...
# 修改下述配置为上一步骤授权列表中的命名空间
apusic_acls_ns=KBCLIC-250922-000072
# 修改localhost为具体ip
apusic_acls_authUrls=172.24.4.217:6869
```

> *提示：本步骤是配置ACP自身授权*

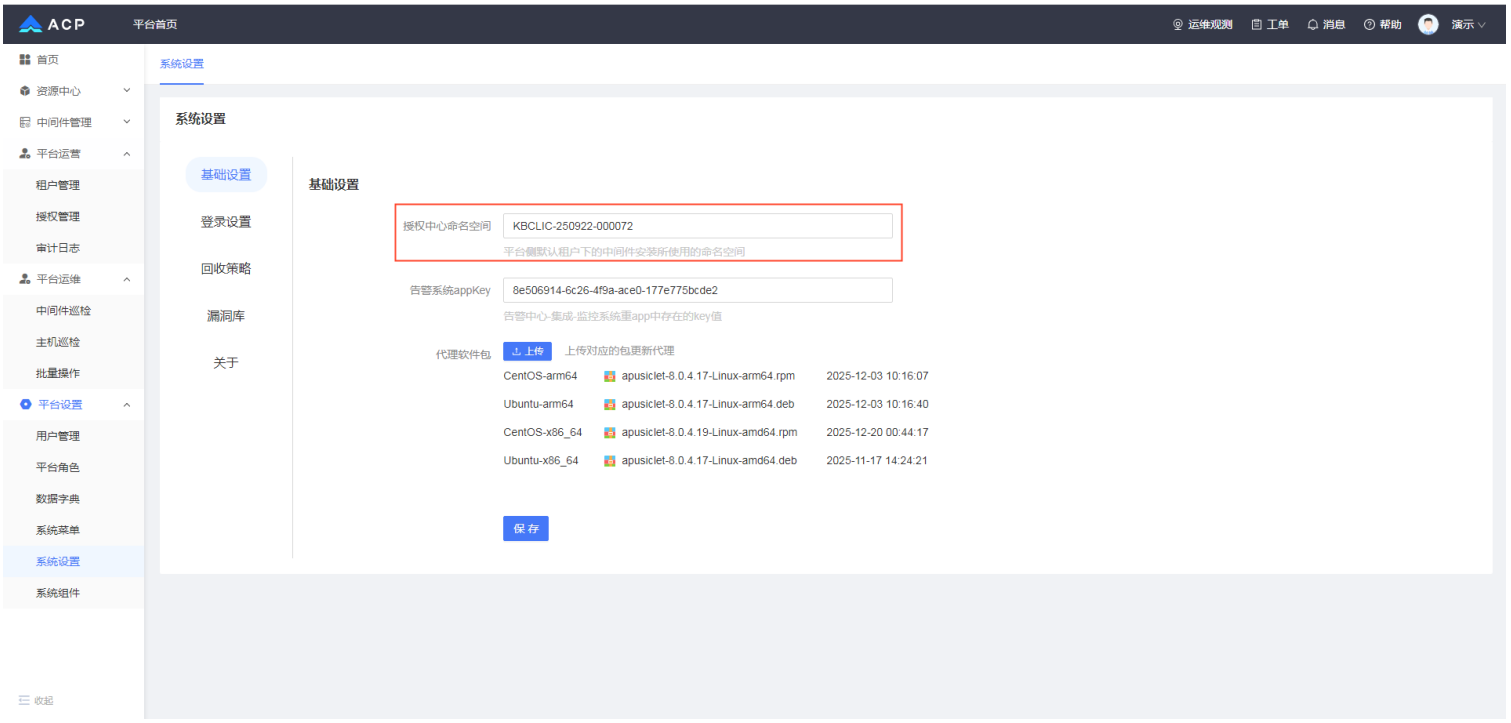
3.重新启动ACP

```
cd home/apusic/maas-8.0.4
./bin/maas.sh stop all
./bin/maas.sh start -d
```

最后，浏览器访问http://IP:9800，默认用户名密码:acpadm/admin

修改授权中心命名空间

在【平台设置】-【基础设置】页面，修改授权中心命名空间。

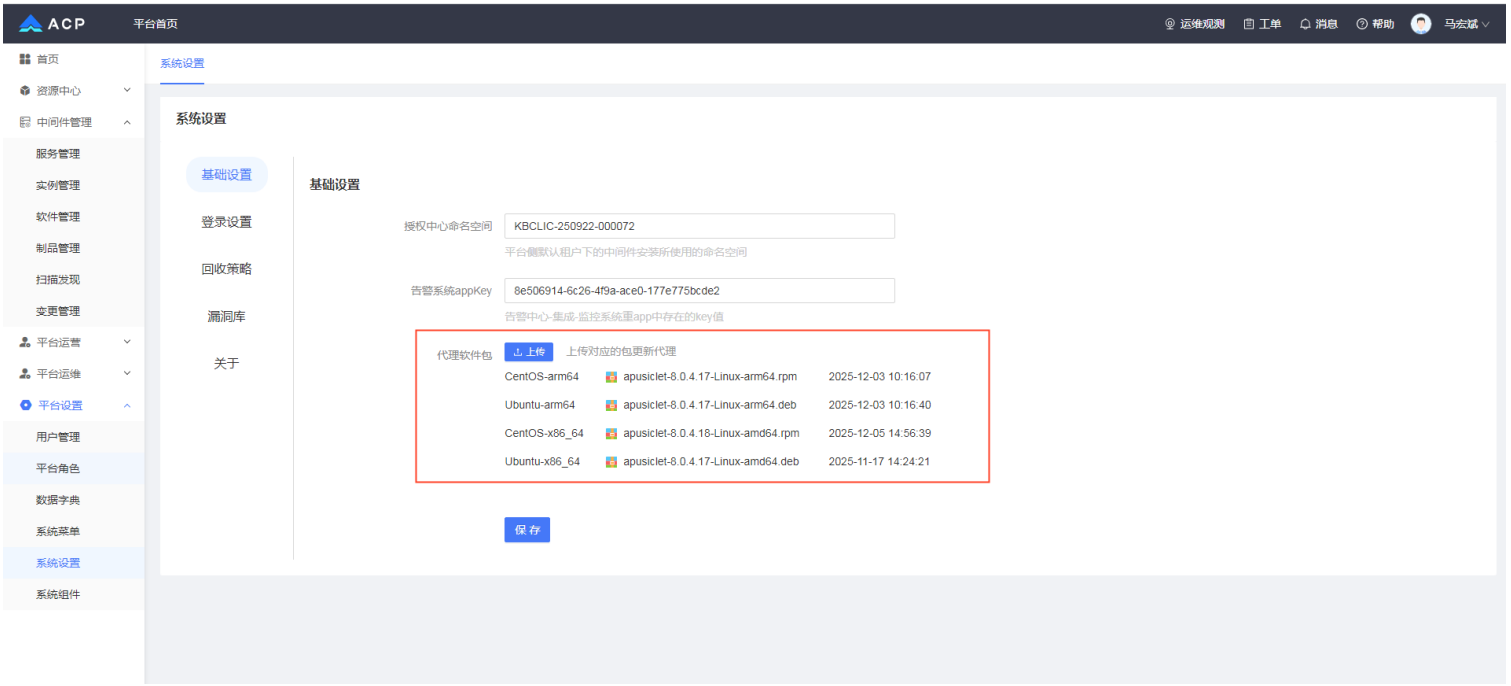


> *提示：通过ACP安装的Apusic中间件默认采用授权中心模式，本步骤是配置中间件使用的授权中心地址。*

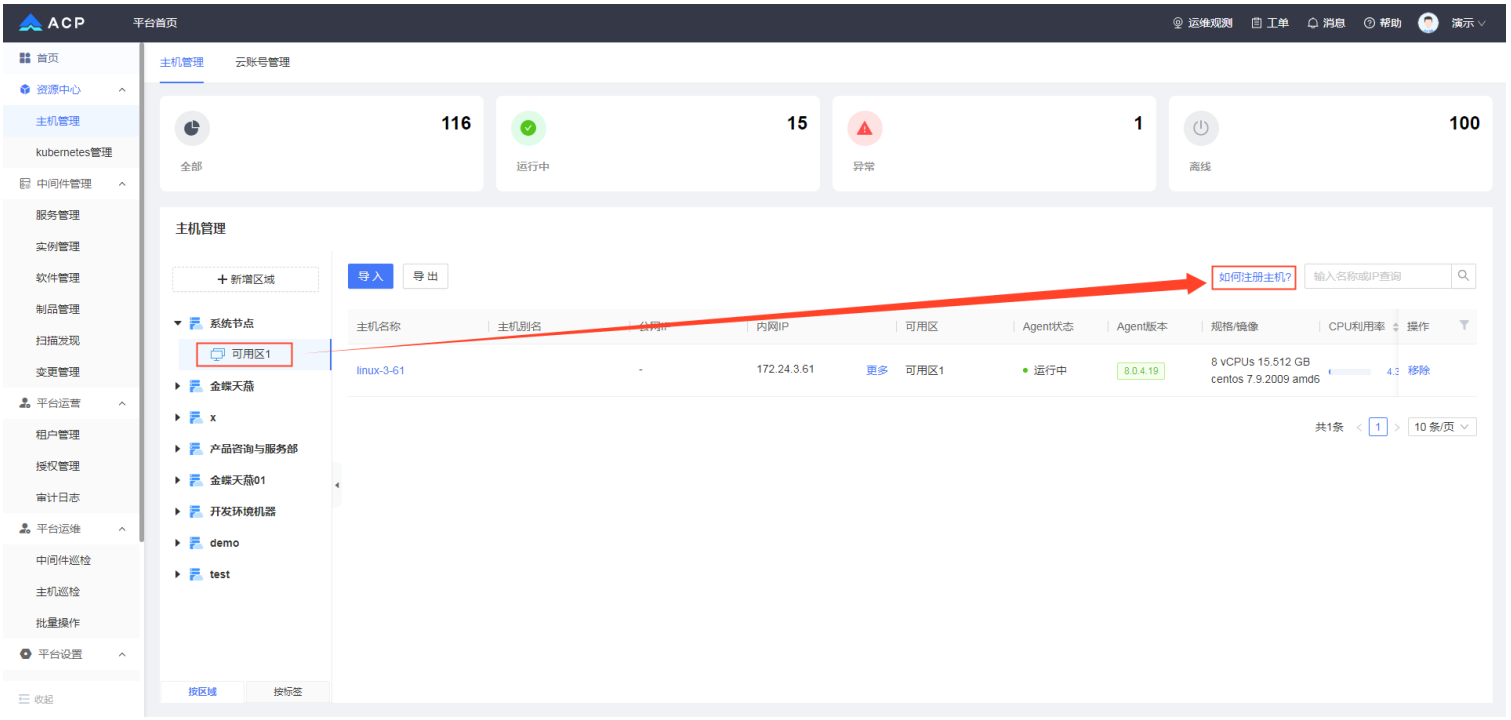
5.4.0.6 注册系统节点

1.【平台设置】-【系统设置】-【基础设置】中导入最新的代理软件包

> *提示：上传完成后记得点击保存*



2.在【资源中心】-【主机管理】页面，选择系统节点，点击打开页面上的"如何注册主机"，按指导在部署ACP的服务器上完成注册操作



5.5 高可用部署

安装高可用的方式 是以单机的方式在三台机器上面安装好maas服务之后将maas组成 **高可用模式** 。

> *提示：Eureka 服务为 maas服务中的maas-register 服务，默认端口为: 8761 默认在安装maas服务的时候,默认已经安装了,不需要另外安装eureka集群。*

修改所有节点maas的配置文件。

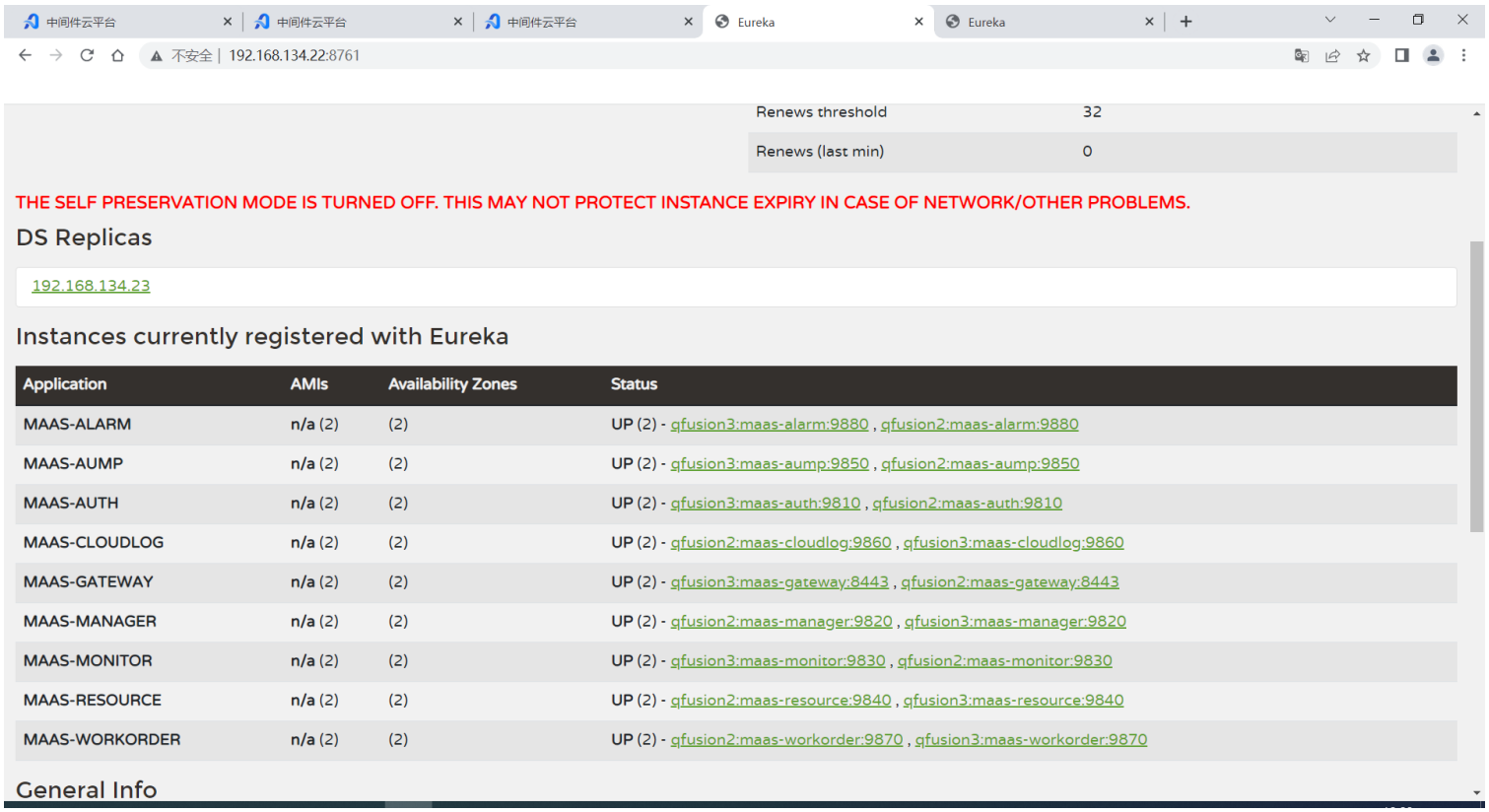
```
vim  conf/application-prod.properties
```

```
# 当前节点的主机名或 IP，集群模式的时候需要开启
# eureka.instance.hostname=localhost


### --- Eureka --- ###
# 这个地方是使用注册中心的地址，当部署一个节点的时候这个地方不需要修改，如果需要部署 高可用集群的时候
# 我们这个地方 需要配置其它俩个节点的eureka节点地址 默认在其他的节点使用的8761
```

```
的端口
# 部署高可用这个地方需要配置 其他节点的服务的监听地址
http://172.24.3.61:38761/eureka,http://172.24.3.61:48761/eureka
# 例如: server1 http://172.24.3.61:8761/eureka
# server2 http://172.24.3.62:8761/eureka
# server3 http://172.24.3.63:8761/eureka
# server1 配置文件如下:
apusic.registry.server.address=http://172.24.3.62:8761/eureka,http://17
# server2 配置文件如下:
apusic.registry.server.address=http://172.24.3.61:8761/eureka,http://17
# server3 配置文件如下:
apusic.registry.server.address=http://172.24.3.61:8761/eureka,http://17
# 如果当前eureka 配置了用户名密码的话 按照上面的配置添加用户密码 例如:
#
apusic.registry.server.address=http://${apusic.registry.user.name}:${ap
#
http://${apusic.registry.user.name}:${apusic.registry.user.password}@17
# 以此类推
# eureka 使用的用户名密码
apusic.registry.user.name=apusic
apusic.registry.user.password=apusic
apusic.registry.server.address=http://${apusic.registry.user.name}:${ap
# 默认不需要修改
apusic.registry.server.port=8761
# 当前节点的主机名或 IP, 集群模式的时候需要开启
# eureka.instance.hostname=localhost

# 当前节点的主机名或 IP, 集群模式的时候需要开启
# eureka.instance.hostname=localhost
### --- Eureka 客户端地址 --- ###
# 如果是单机部署的话 只用连接本地的 eureka地址,这个地方 只需要填写
http://localhost:8761/eureka 地址就好了
# 如果部署的是一个高可用的 eureka 集群, 这个地址我们只需要改成所有的eureka的地
址。
# 例如: server1 http://172.24.3.61:8761/eureka
# server2 http://172.24.3.62:8761/eureka
```

```
# server3 http://172.24.3.63:8761/eureka
# 所以这个配置如下: eureka.client.service-
url.defaultZone=http://172.24.3.61:8761/eureka,http://172.24.3.62:8761/
eureka.client.service-
url.defaultZone=http://${apusic.registry.user.name}:${apusic.registry.u
eureka.instance.metadata-map.zone=local
# 心跳间隔，10秒发送一次
eureka.instance.lease-renewal-interval-in-seconds=10
# 心跳过期时间，90秒内未收到心跳则认为服务不可用
eureka.instance.lease-expiration-duration-in-seconds=90
# 启用注册表拉取 客户端会定期向 Eureka Server 拉取服务注册表，确保获取最新的服务信息。如果在拉取时连接失败，客户端会进行重试。
eureka.client.fetch-registry=true
# 注册表拉取超时10秒->设置客户端获取注册表时的超时时间。如果超过这个时间还未能获取到注册表，客户端会失败并重新尝试。
eureka.client.registry-fetch-timeout-seconds=10
eureka.client.register-with-eureka=true
```

修改完成后，启动所有节点的 MAAS 服务，并等待各节点上的 MAAS 服务完全启动和运行正常。接着，依次访问每个节点的 8761 端口。当显示的服务状态数量与当前集群的节点数量一致时，说明集群已成功组建，且各节点已互为高可用集群。



> *提示：安装完成后需要激活平台，才能正常使用*

6 主要功能

6.1 计算资源准备

平台提供两种主机接入方式，满足不同场景需求：

6.1.1 自动注册

适用于快速接入单台主机的场景。

前提条件

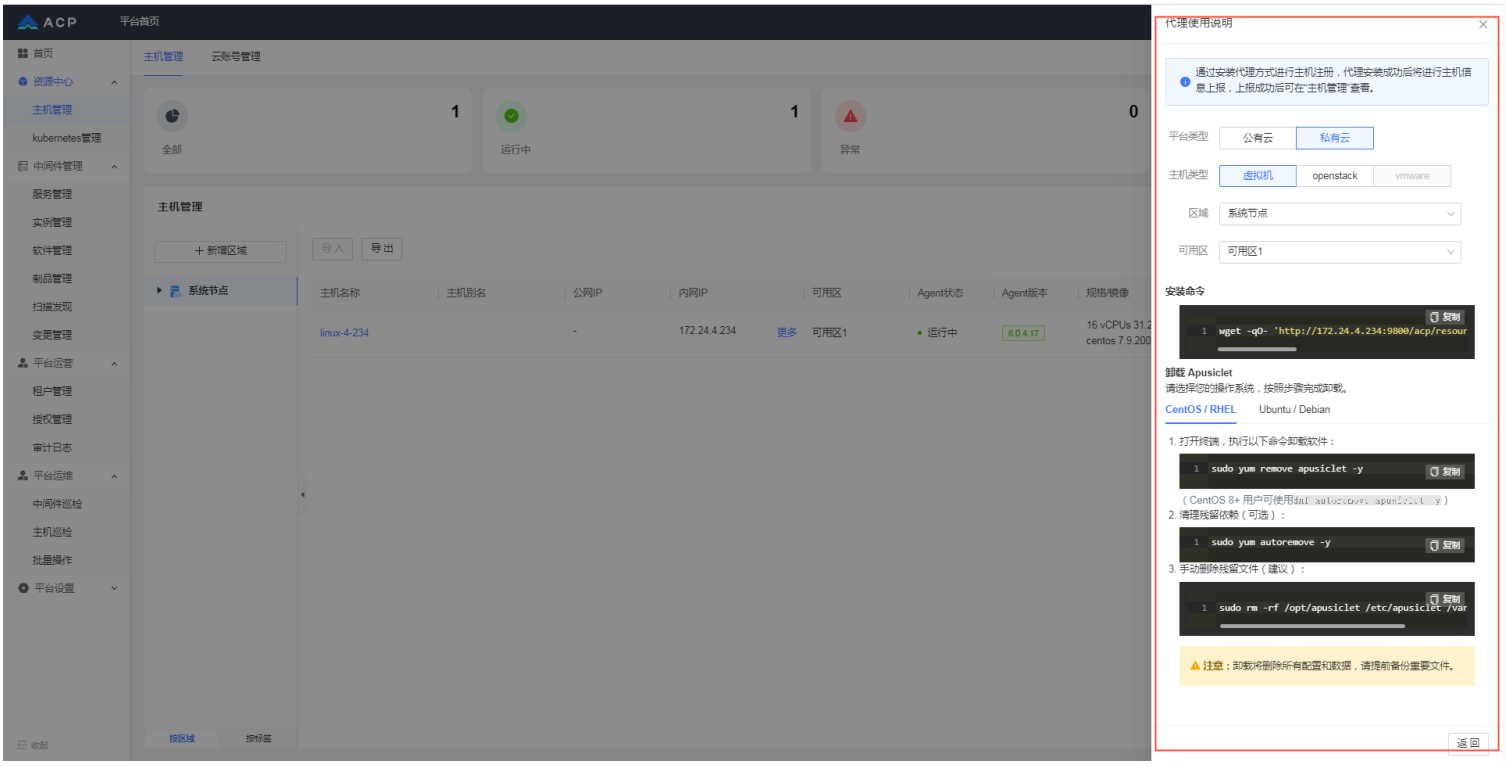
- 待接入主机与平台网络互通

操作步骤

1.进入【资源中心】-【主机管理】，点击【新增区域】：

- 选择"平台类型"（公有云/私有云）、"主机类型"（虚拟机/OpenStack等）
- 填写"区域名称"（如"华北一区"）、"可用区"（如"可用区1"，同一区域可添加1-10个）
- 点击【确定】完成区域配置

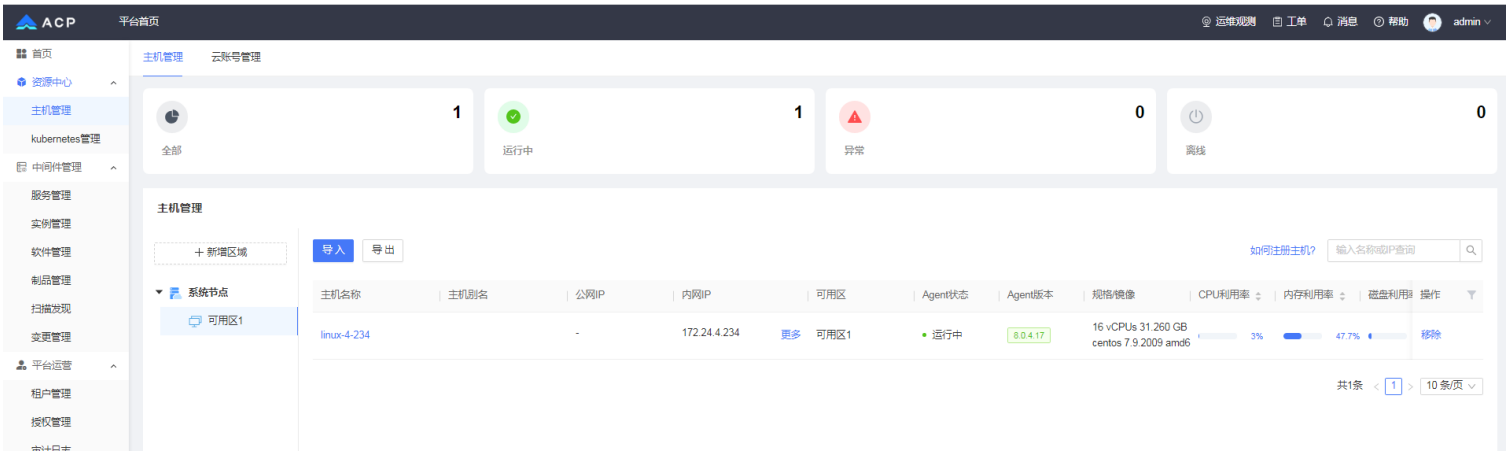
2.选中目标区域及可用区，点击【如何注册主机】，复制系统生成的注册命令（含 apusiclet 代理安装）



3.以Root身份登录待接入主机，执行注册命令（示例）：

```
wget http://{平台IP}:9800/apusiclet-8.0.3-Linux-x86_64.rpm  \
&& rpm -ivh apusiclet-8.0.3-Linux-x86_64.rpm
```

4.执行成功后，返回平台【主机管理】，刷新后可见主机（状态为"运行中"），点击主机名称可查看CPU、内存等详细信息。



> *提示*
>
> - *需要将部署ACP的主机注册到系统节点区域*
> - *其他业务主机建议创建新的区域、可用区*

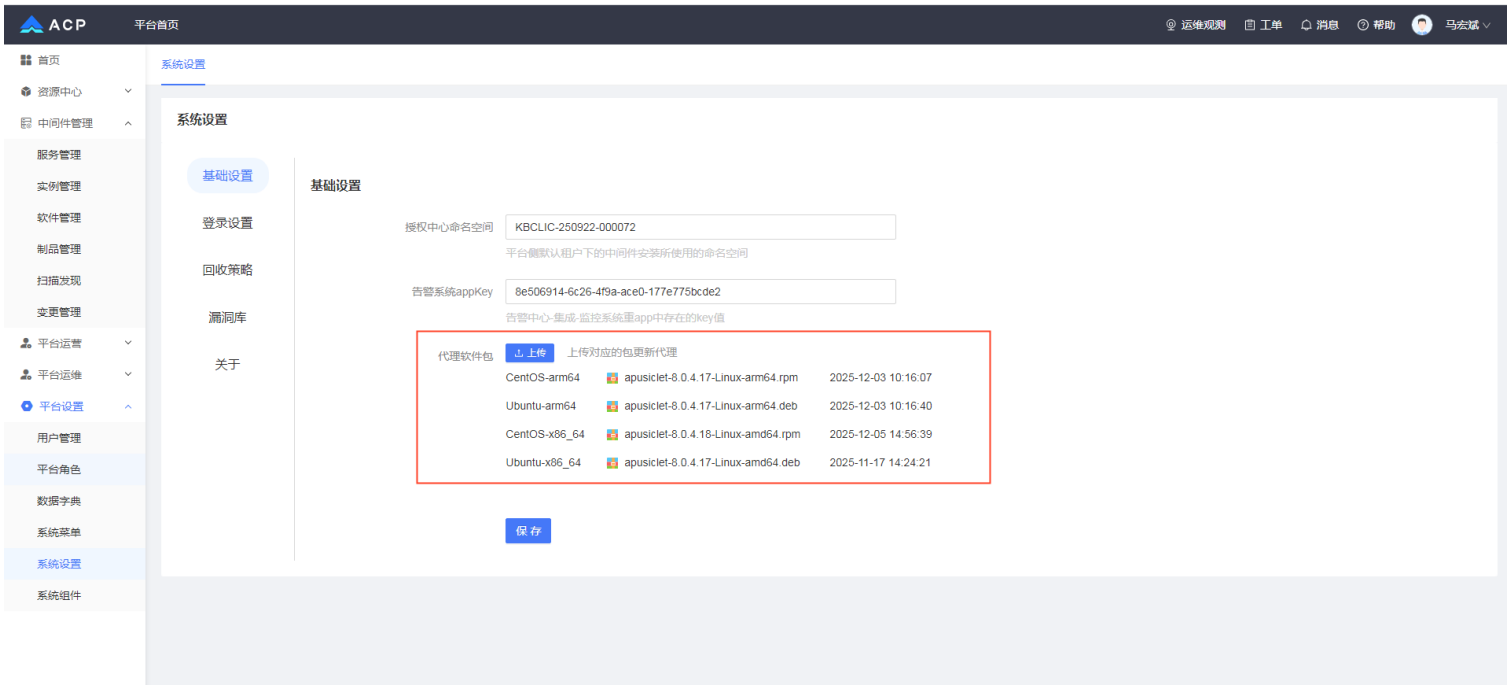
6.1.2 手动注册（批量接入）

适用于批量接入多台主机的场景。

前提条件

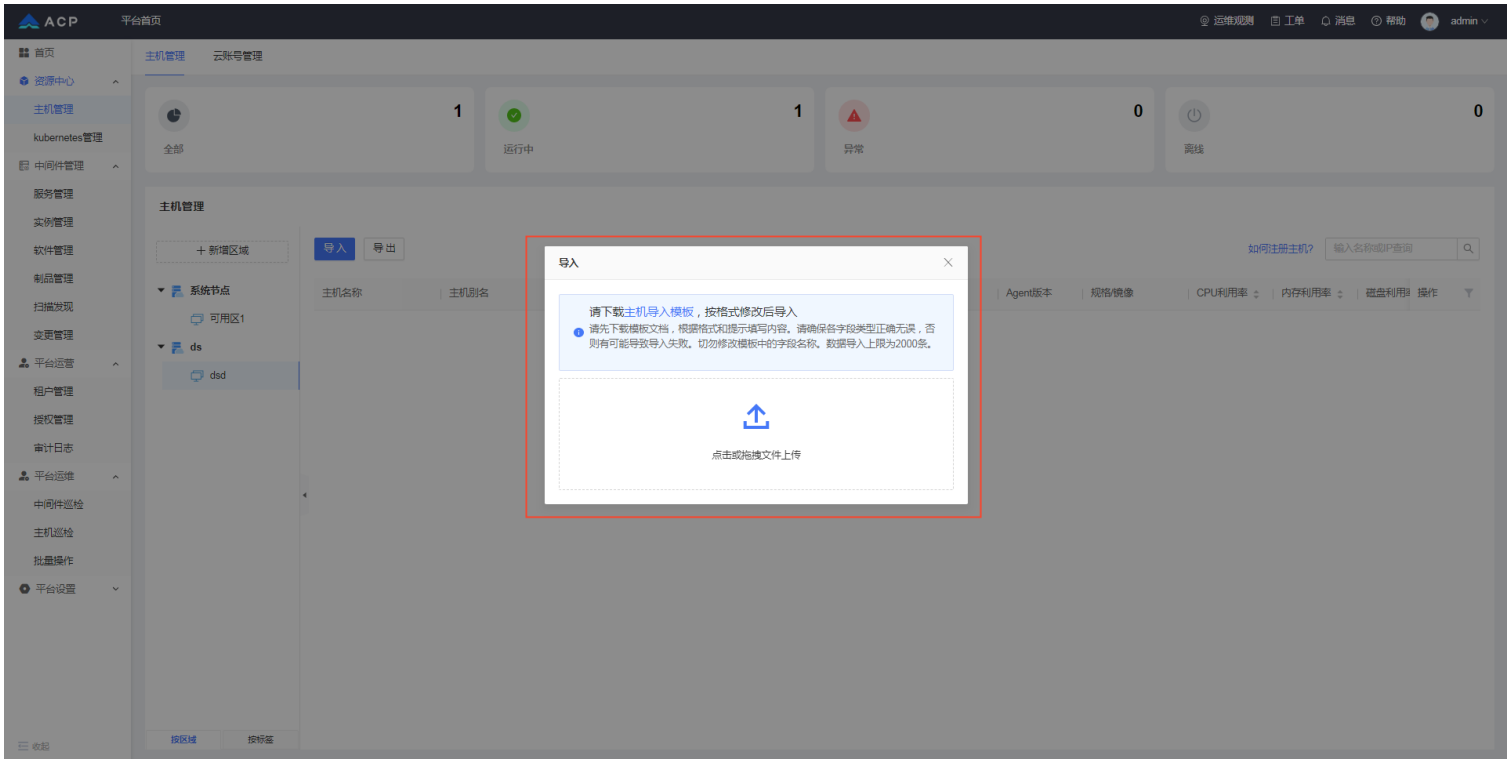
- 待接入主机与平台网络互通
- 【平台设置】-【系统设置】-【基础设置】中导入最新的代理软件包

> *提示：上传完成后记得点击保存*



操作步骤

1. 进入【主机管理】，选中区域及可用区，点击【导入】
2. 点击【主机导入模板】，下载 `主机导入模板.xls`，按要求填写（单文件最多支持 2000 条记录）
3. 上传填写好的 Excel 文件，点击【确定】，系统将自动校验并导入
4. 查看导入结果：成功记录将显示在主机列表中，失败记录可通过【下载文档】查看具体原因（如 IP 不可达）



提示

> - *需要将部署ACP的主机注册到系统节点区域*
> - *其他业务主机建议创建新的区域、可用区*

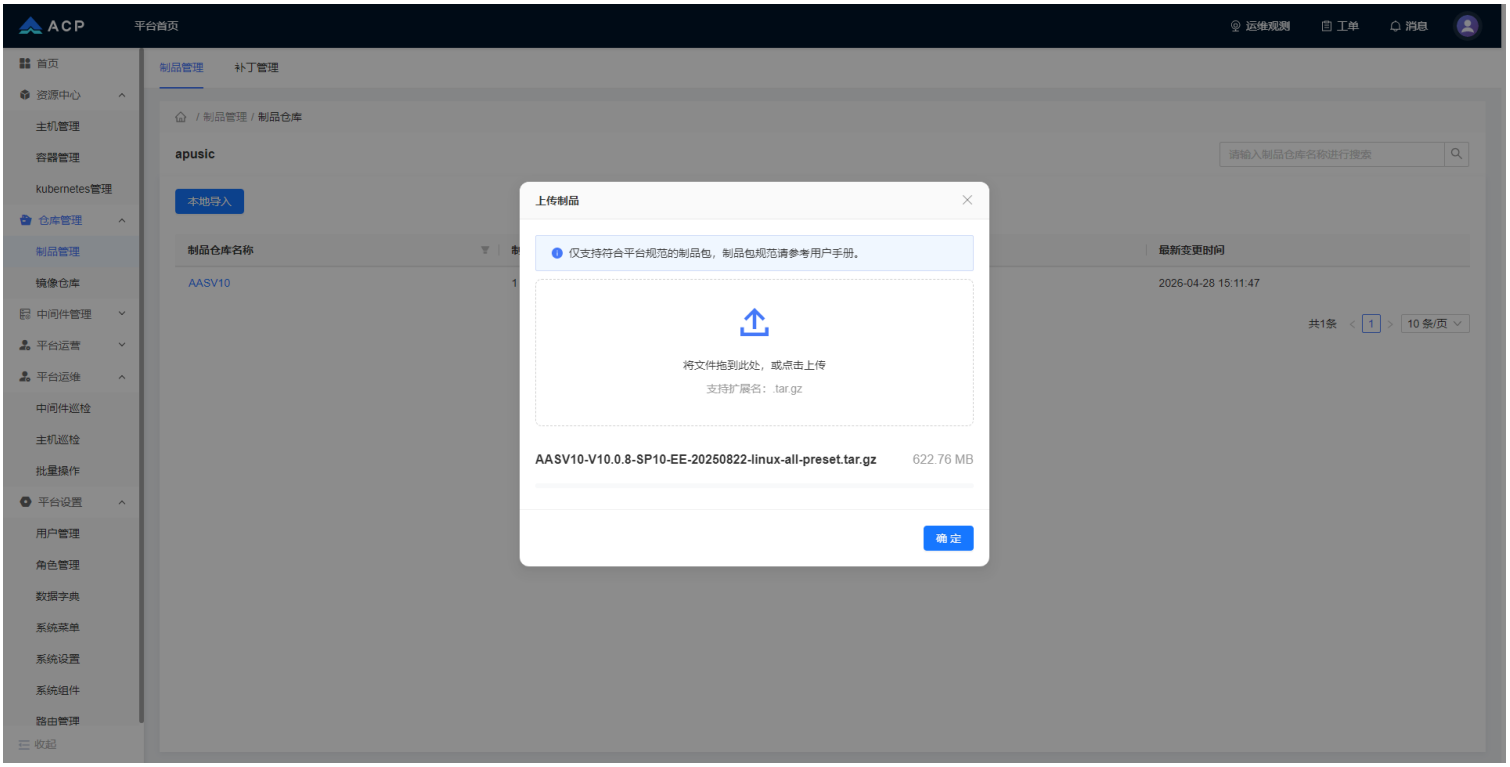
6.2 导入软件制品包

前提条件

- 已获取研发提供的各中间件制品包

操作步骤

1. 进入【仓库管理】-【制品管理】，点击`apusic`进入该制品项目，点击【本地导入】
2. 上传中间件制品包.tar.gz格式，点击【确定】，系统将自动校验并上传至文件服务器



6.3 部署中间件

ACP平台支持多品类、多版本、多部署模式的中间件安装，满足测试和生产环境的不同需求。以下以 AMDC（金蝶Apusic分布式缓存）为例演示安装流程。

前提条件

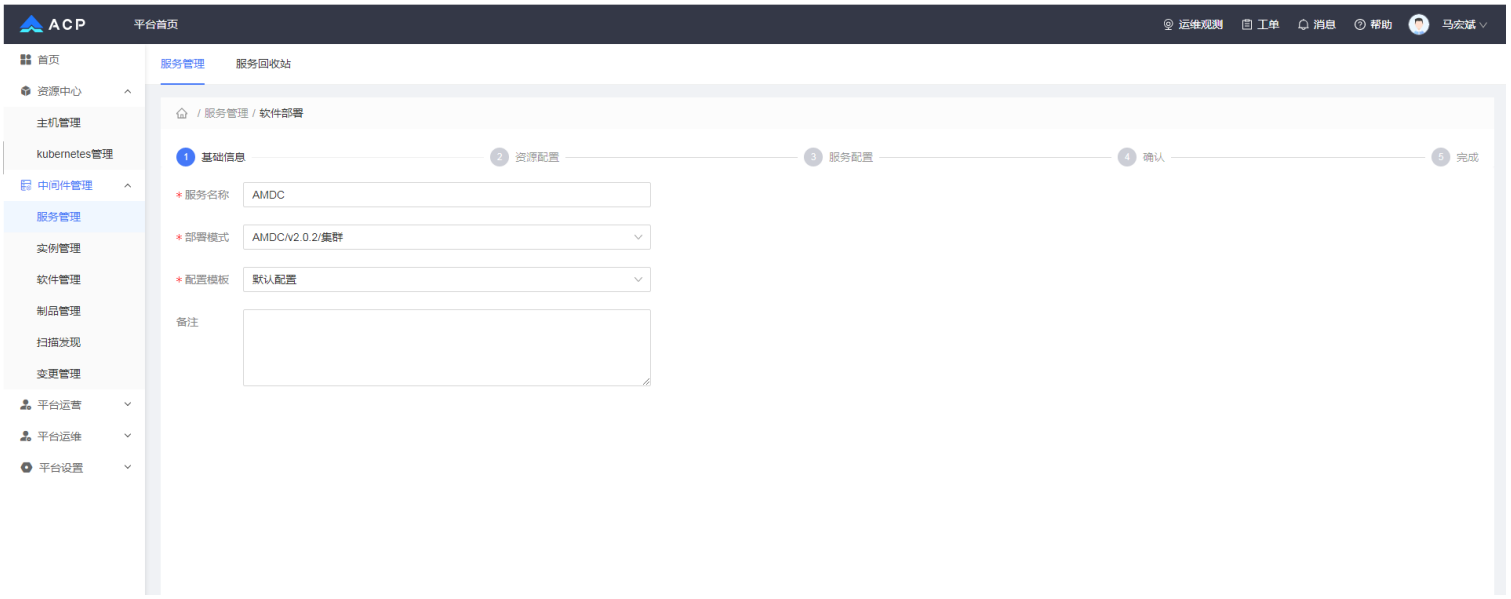
- 软件管理中已内置或者导入相关中间件
- 制品管理中已导入相关制品包

6.3.1 基础配置

1. 进入【中间件管理】-【服务管理】-【部署】

2. 填写服务名称，部署模式，配置模板，点击【下一步】。

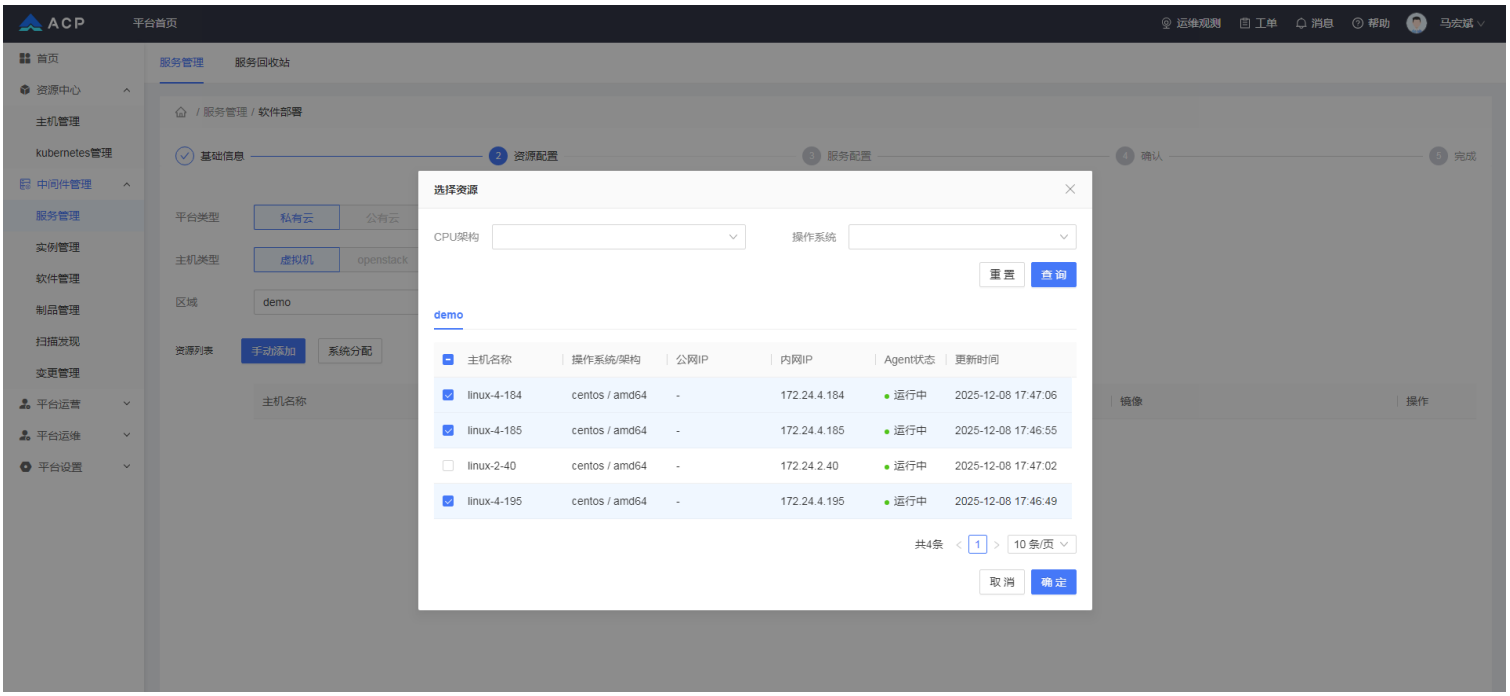
> *提示：本次部署选择AMDC集群模式*



6.3.2 资源配置

1. 选择合适的主机，点击【下一步】

> *提示：本次部署选择3台主机进行部署*

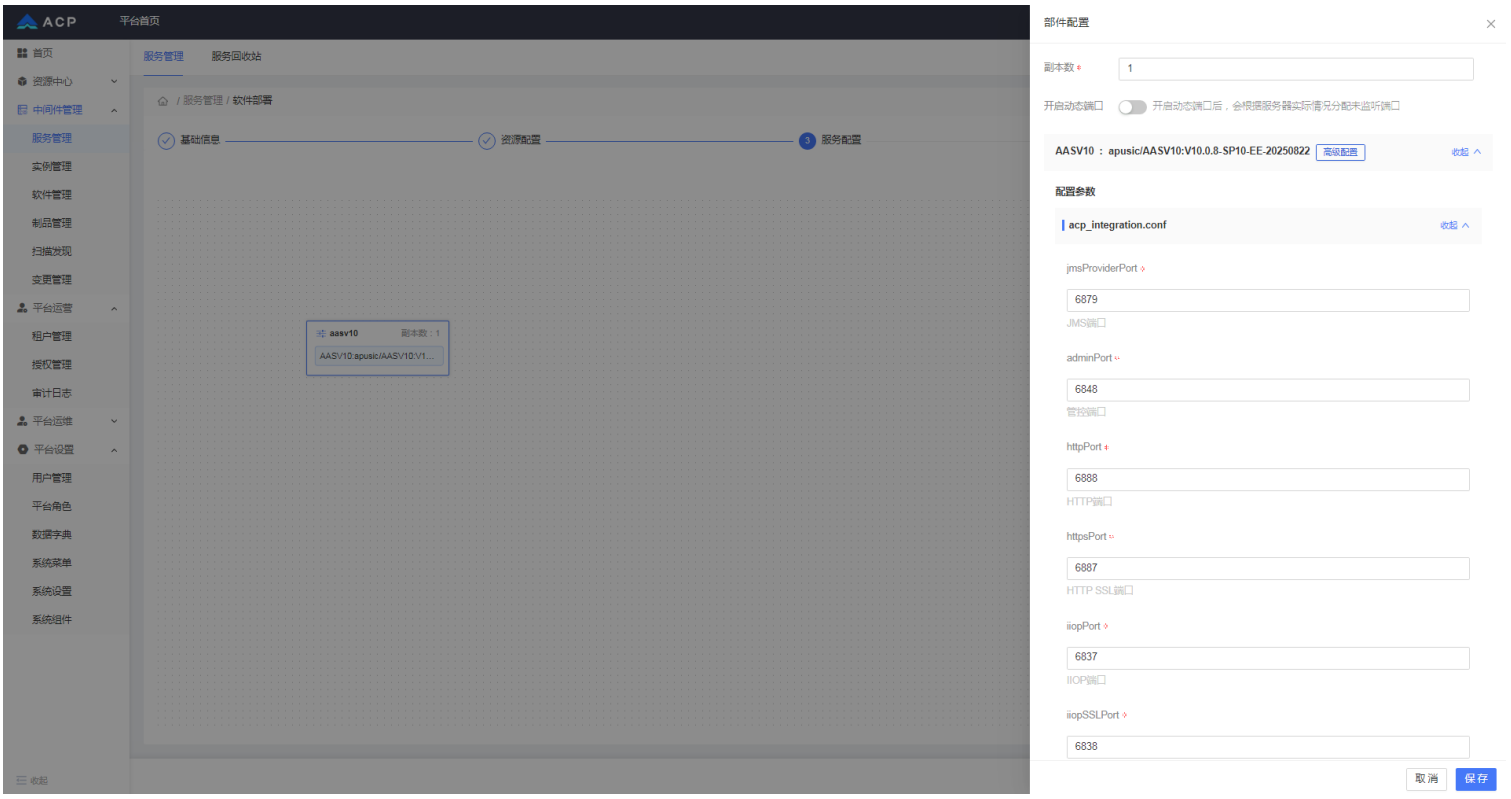


6.3.3 服务配置

点击部件，打开部件配置抽屉，在抽屉页面，可以进行如下配置：

- 副本数：即本次部署的中间件实例数量。
- 开启动态端口：如若开启，会根据设置的端口范围以及目标主机上的空闲端口进行动态分配
- 组件高级配置：组件的制品包、启停脚本，数据路径等配置。
- 组件配置参数设置：基于事先定义好的配置模板（即配置库），根据实际需要对配置进行调整。

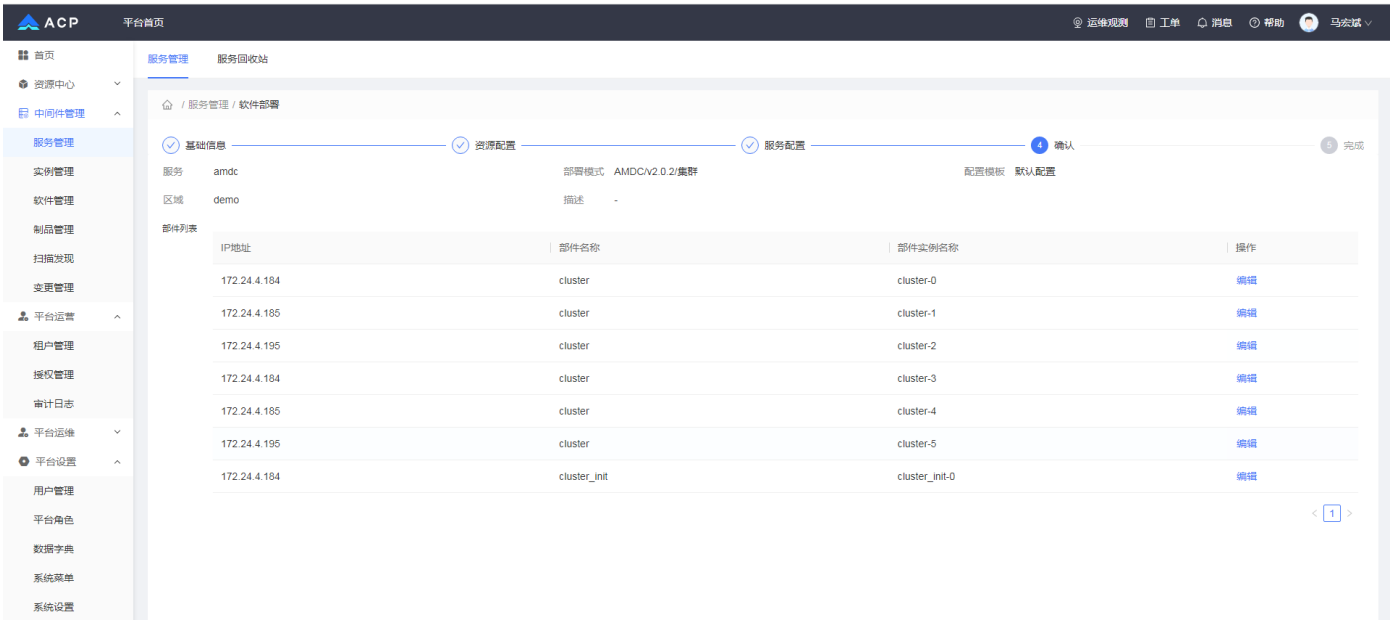
> *提示：*
>
> - *ACP平台会根据副本数，将实例在主机上进行均匀分布。如副本数设置为4，主机数为2，则每台主机会部署2个实例*
> - *动态端口仅对配置母版中通过@Port声明的端口生效*
> - *组件高级配置一般仅需要根据实际场景选择制品包，其他项不作修改。*
> - *组件配置参数中需要关注IP、端口、密码等配置*



6.3.4 确认安装

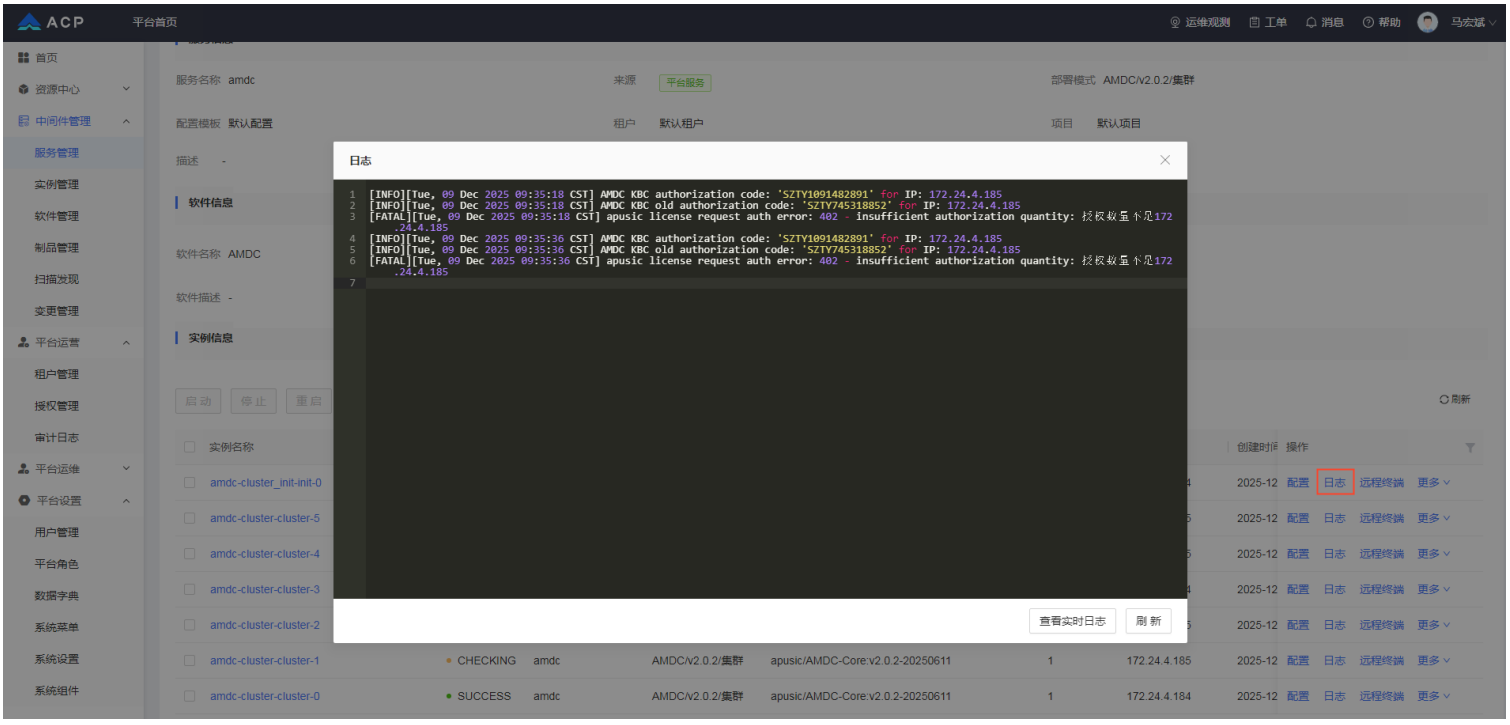
1. 确认所有配置信息无误后，点击【安装】

2. 返回列表页，查看安装状态和进度



6.3.5 错误排查

如遇安装失败，可在【中间件管理】-【服务管理】页中点击服务名称进入服务详情页，然后点击实例列表中右侧【日志】查看详细失败原因，根据日志信息进行问题定位和解决



> *提示：如上图，部署失败的原因是授权不足，可通过增加授权数解决。*

6.4 纳管中间件

平台支持对客户环境中已独立部署的第三方中间件（如 Tomcat、Kafka）进行纳管，实现全生命周期统一管控。

前提条件

- 软件管理中已内置或者导入相关中间件
- 制品管理中已导入相关制品包

6.4.1 创建扫描特征

扫描特征是平台识别第三方中间件的关键依据，需要准确定义中间件的核心标识信息。

前提条件

- 该软件已经在【中间件管理】-【软件管理】中定义。

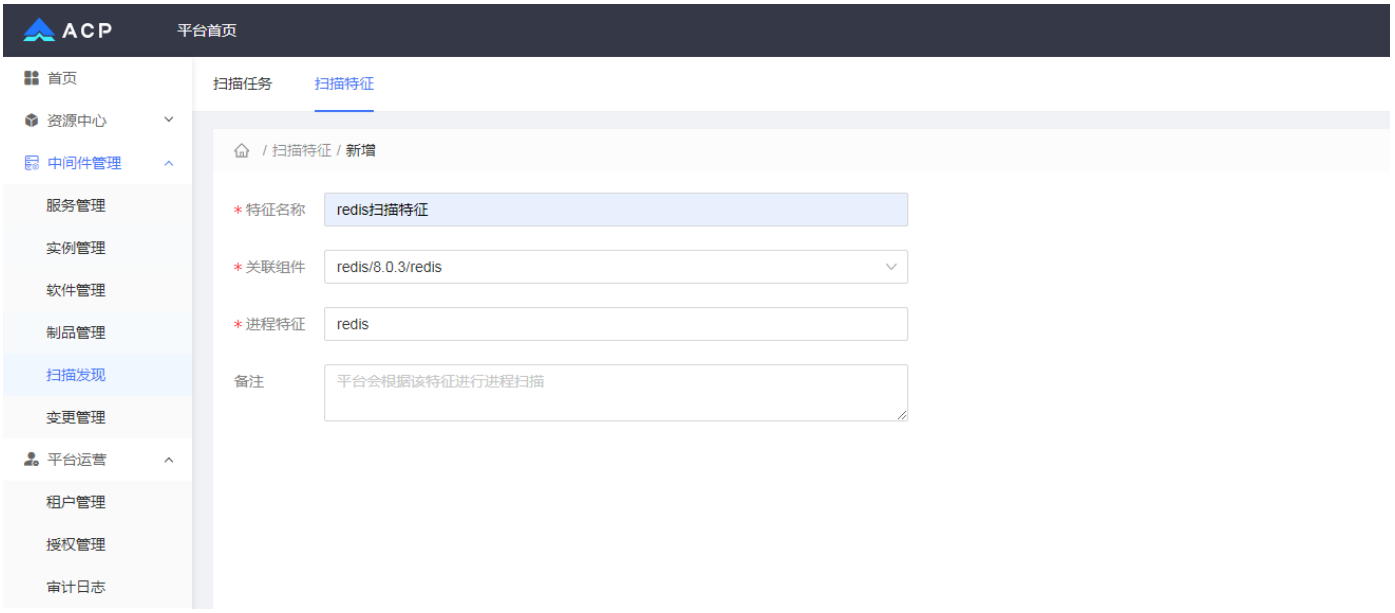
操作步骤：

1. 进入【中间件管理】-【扫描发现】-【扫描特征】页面，点击【创建】按钮
2. 在弹出的对话框中，填写特征基础信息：

- **特征名称**：自定义名称（例如："redis扫描特征"）
- **关联组件**：关联【软件管理】中定义的软件、版本、组件
- **进程特征**：输入用于识别该中间件的进程关键字（例如："redis"）
- **备注**：可选项，填写相关补充说明信息

> *提示：进程特征参考ps命令，如查询Reids进程命令为：ps -ef | grep redis，则进程特征为redis*

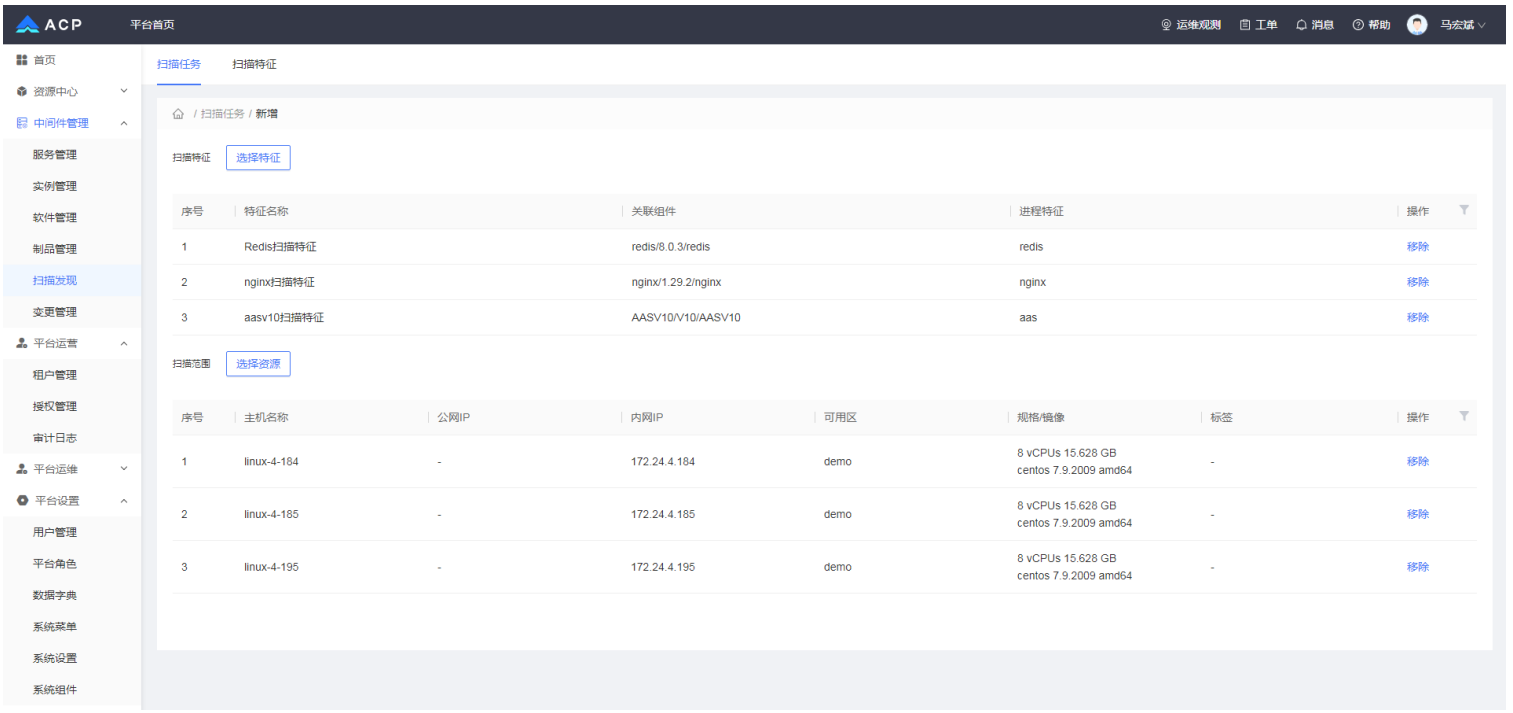
3. 确认信息无误后，点击【确定】按钮完成创建



6.4.2 创建扫描任务

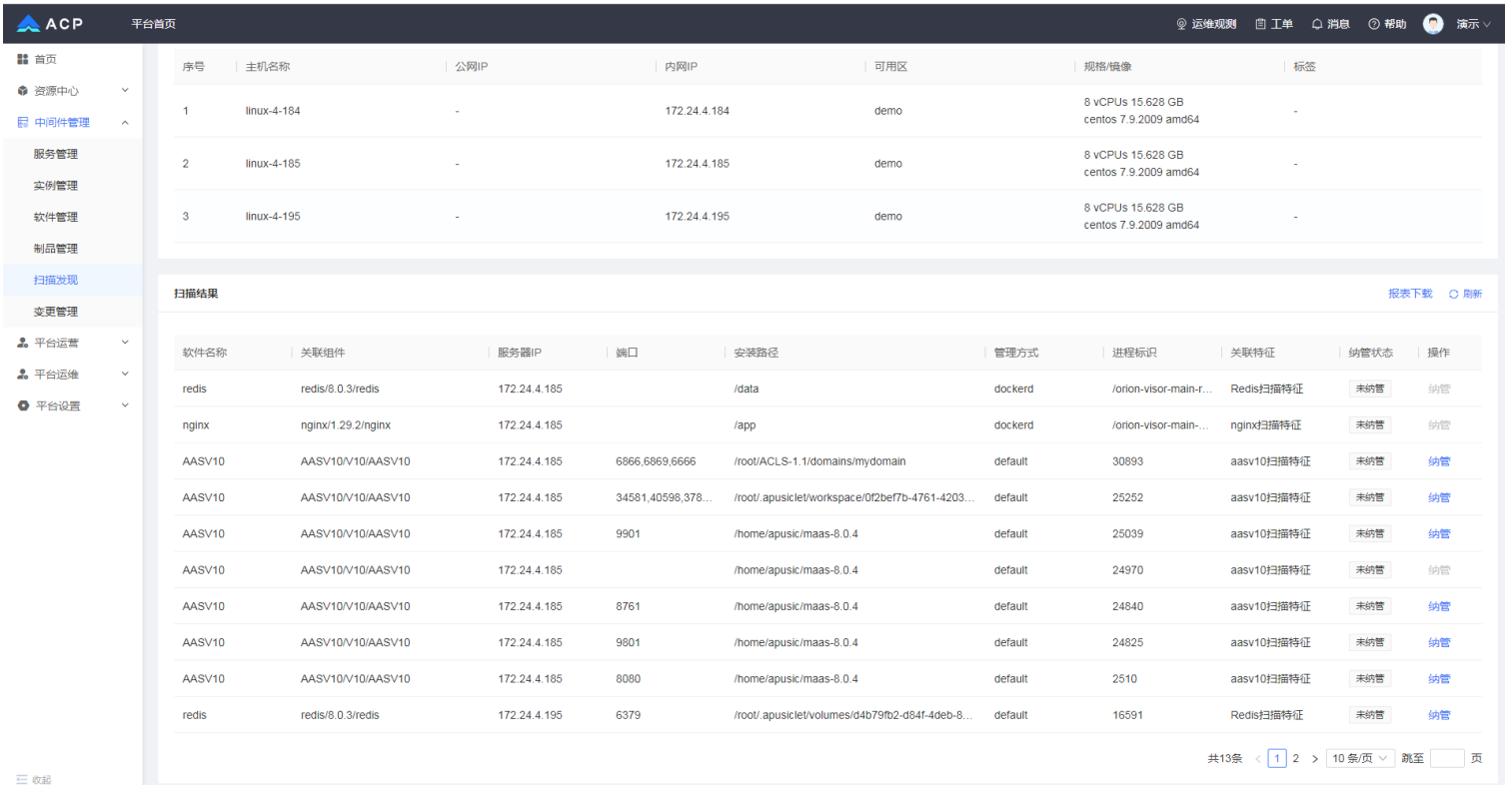
基于已创建的扫描特征，在指定主机中执行扫描，发现待纳管的中间件实例。

1. 进入【中间件管理】-【扫描发现】-【扫描任务】，点击【创建】
2. 配置任务参数：
   - 任务名称：自定义（如 "扫描生产环境 redis"）
   - 选择扫描特征：勾选 "redis扫描特征"（可多选）
   - 选择目标主机：从已接入主机中勾选（支持按区域 / 名称筛选）
3. 点击【确定】，任务创建完成，可在【扫描任务】列表中查看进度状态



6.4.3 纳管实例

扫描完成，点击【扫描结果】，可查看扫描发现到的中间件实例。在扫描结果列表中，查看实例信息（包括主机 IP、软件名称、管理方式、进程标识、端口、关联特征、标签等）



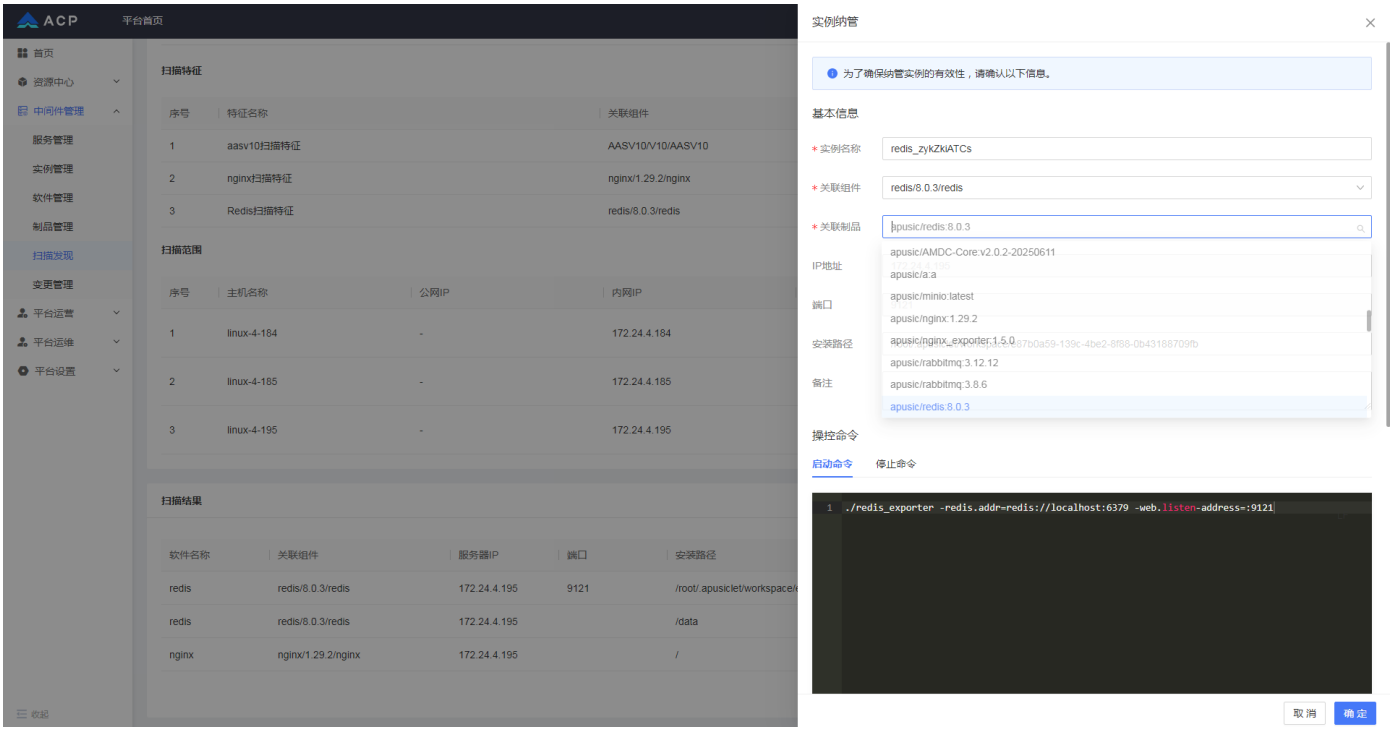
点击实例列表右侧【纳管】按钮，进行纳管。

1. 在弹出窗口中，配置实例纳管信息：

    - 基本信息：实例基本信息，如实例名称、关联组件，关联制品、IP地址、端口、安装路径、备注
    - 操控命令：启动命令、停止命令
    - 高级信息：安装用户信息、信息收集路径、健康检查

> *提示*
>
> - *平台会根据进程信息自动填充启动或者停止命令，请根据实际情况修改*
> - *请正确填写数据、日志、配置路径、健康检查等信息*

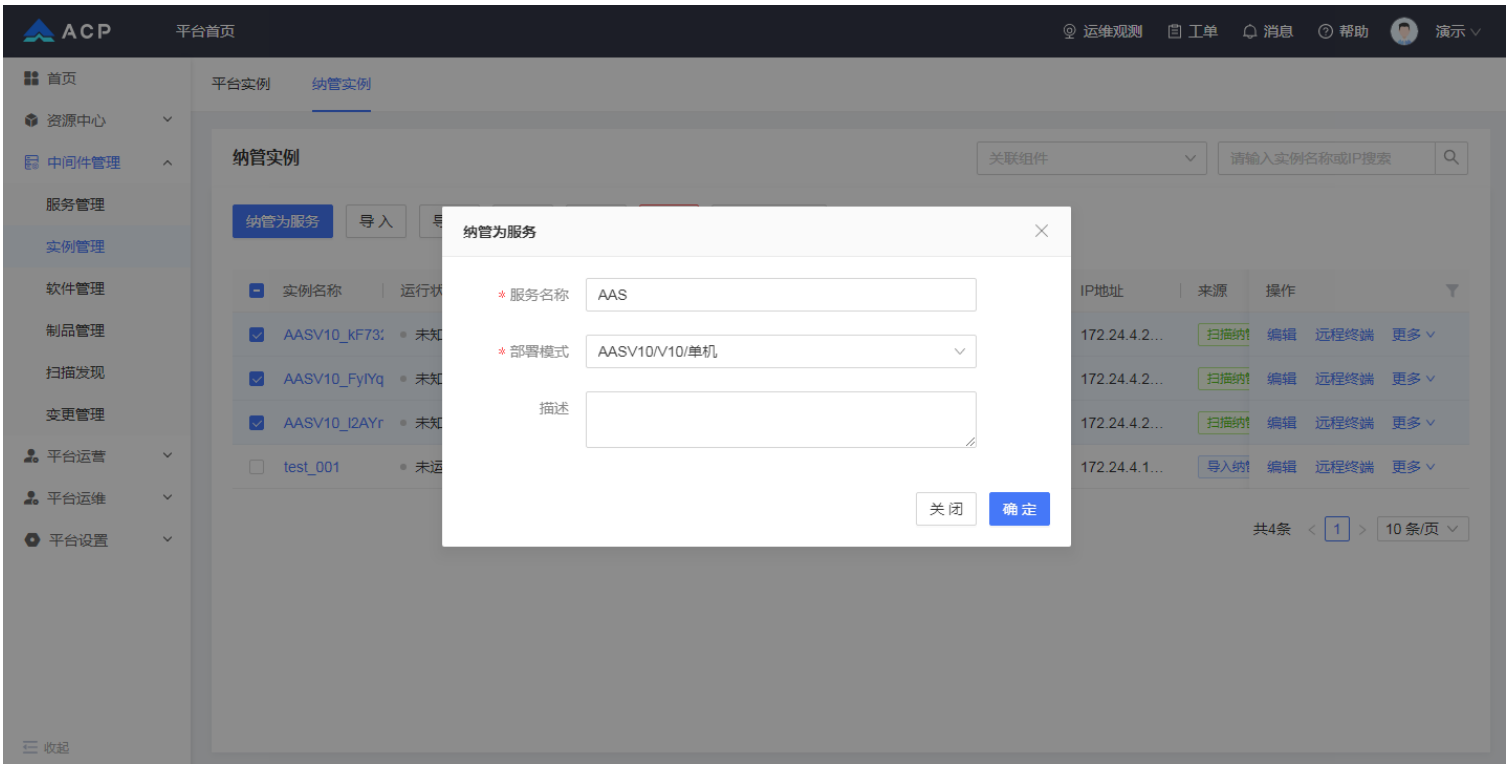
2. 点击【确定】，实例进入纳管流程，完成后可在【实例管理】-【纳管实例】列表中查看



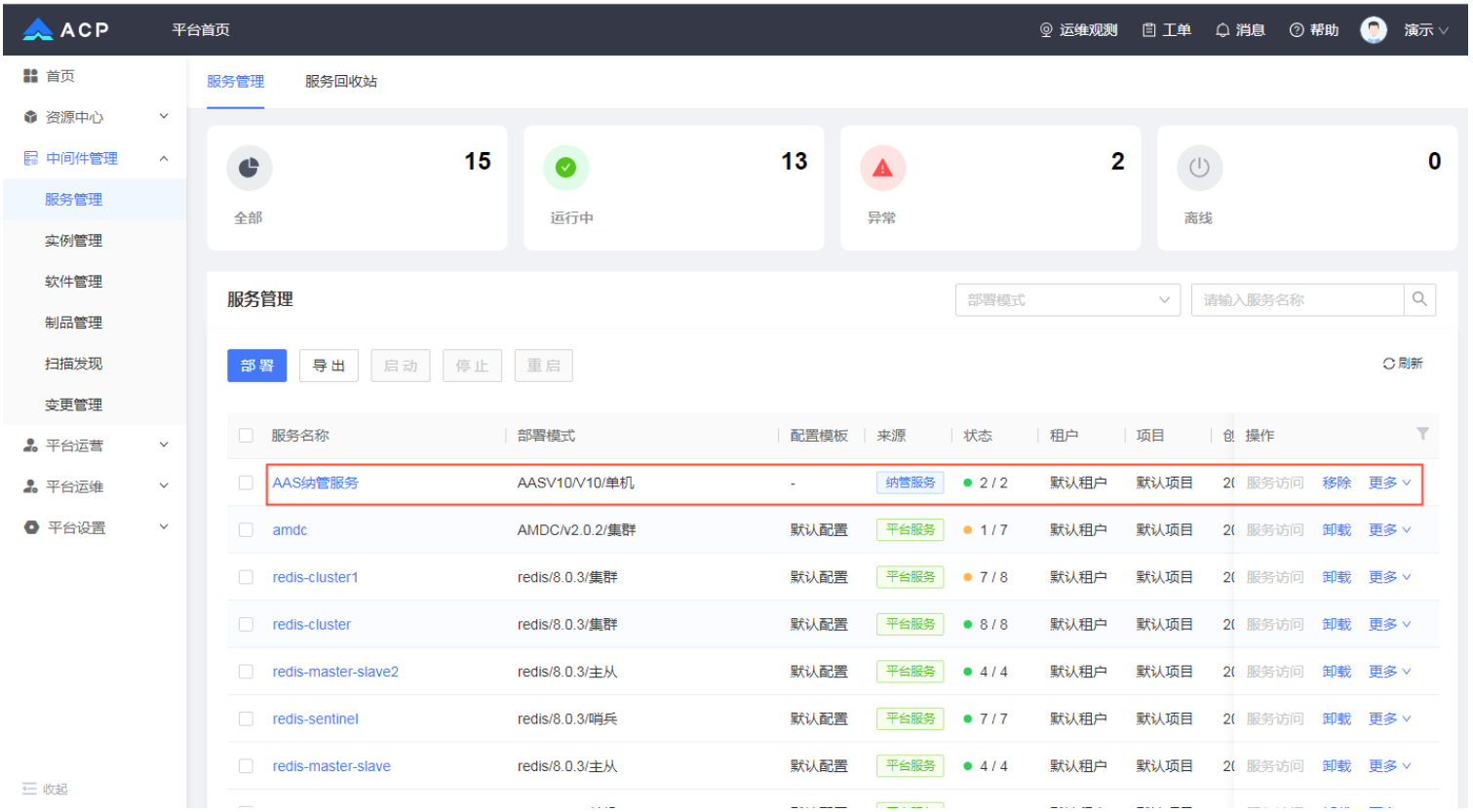
6.4.4 纳管服务

中间件服务通常由一个或多个中间件实例共同提供，如AMDC集群由6个AMDC实例组成。通过ACP平台部署的中间件默认会以服务形式展现，纳管的中间件实例则需要手动关联为服务。为此，ACP提供了纳管为服务功能。

1.在【中间件管理】-【实例管理】-【纳管实例】纳管实例页面，选择相关的中间件实例，点击【纳管为服务】，填写表单，即可生成服务。



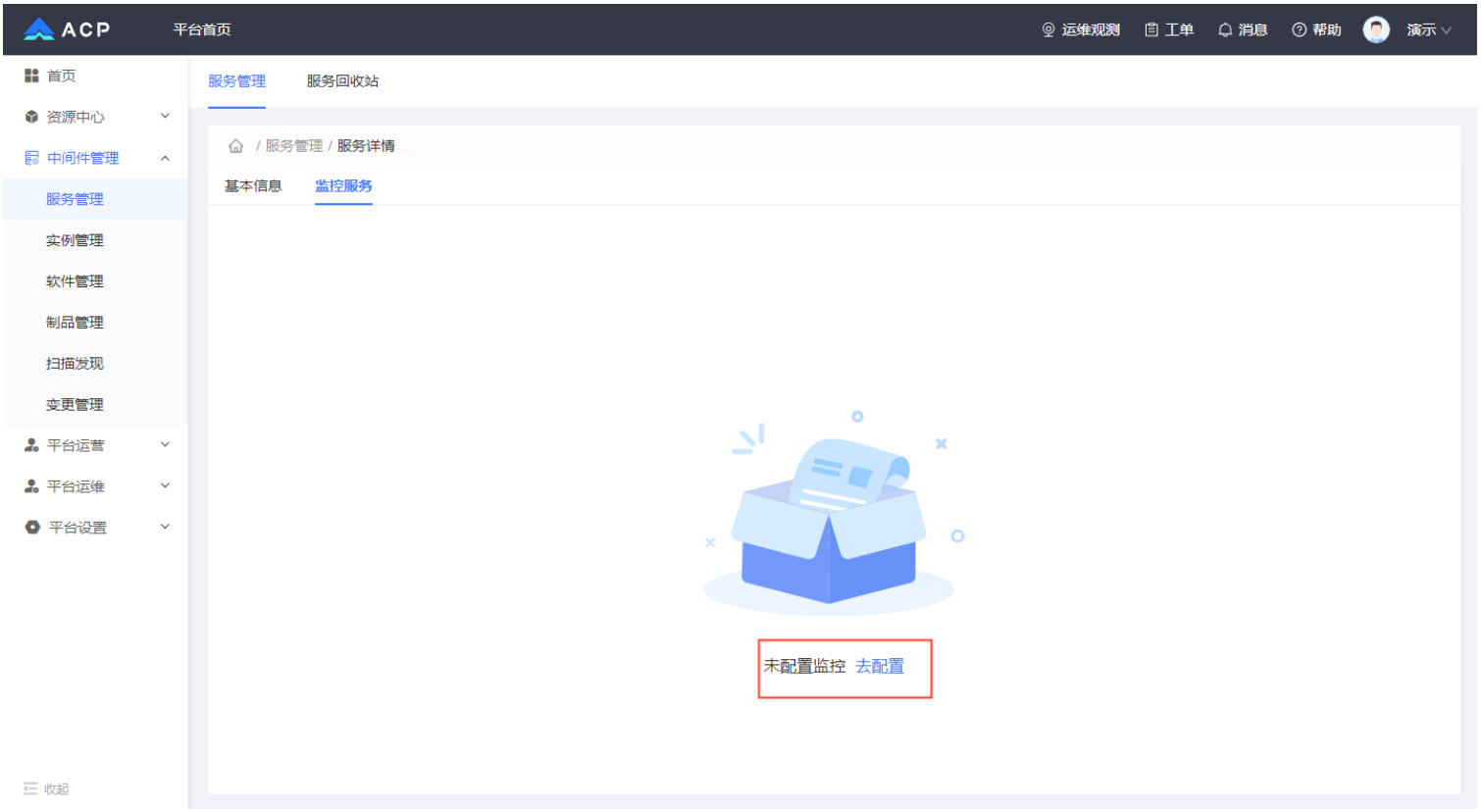
2.然后在【中间件管理】-【服务管理】页面，可以看到刚才生成的服务。



> *提示：针对已生成的纳管服务，添加纳管实例，可在服务详情页点击【新增实例】按钮添加*

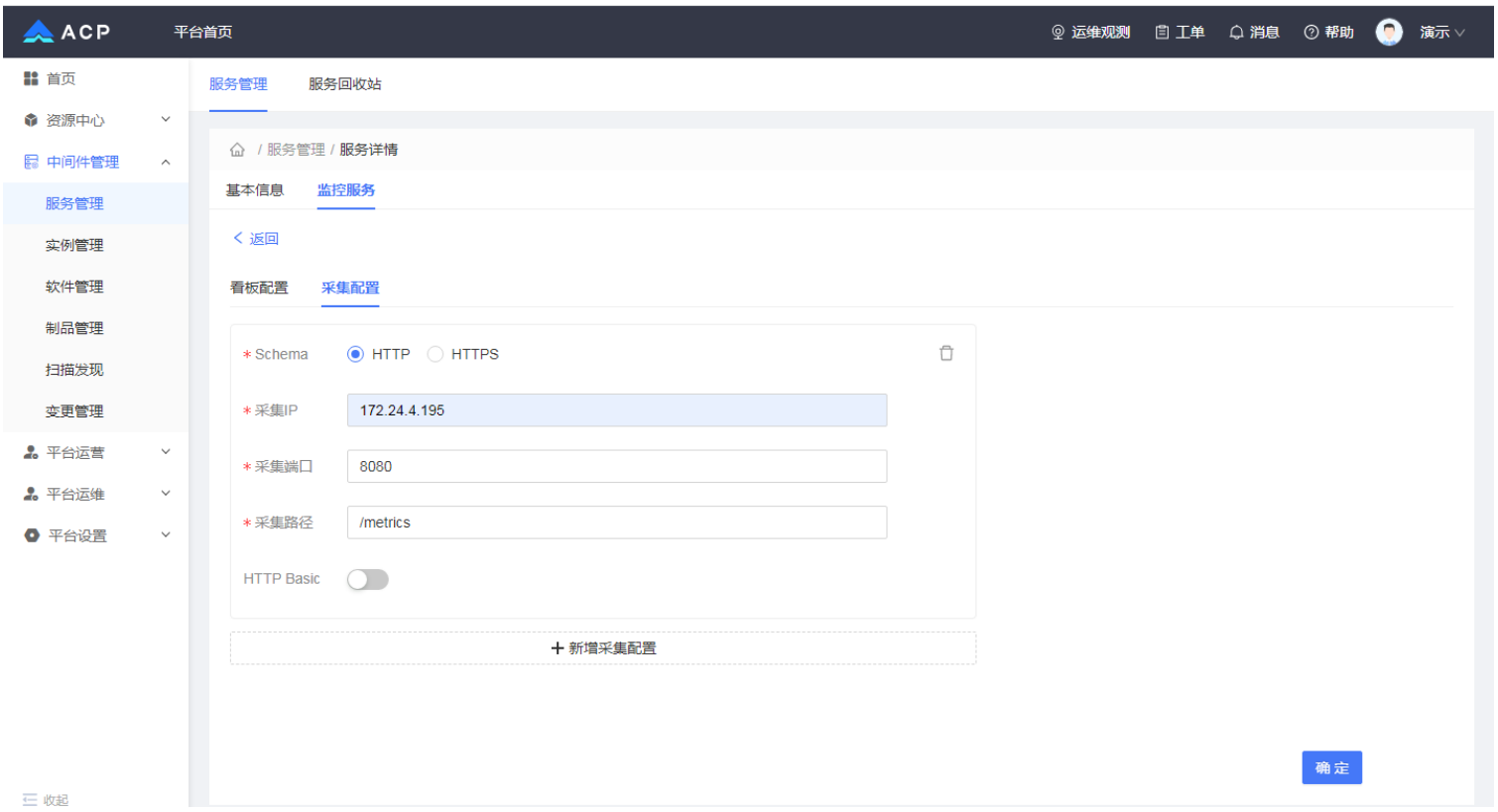
6.4.5 服务监控

纳管服务默认不会进行监控，如若需要监控，可以在服务详情中点击【监控服务】-【去配置】，进入监控配置。



采集配置

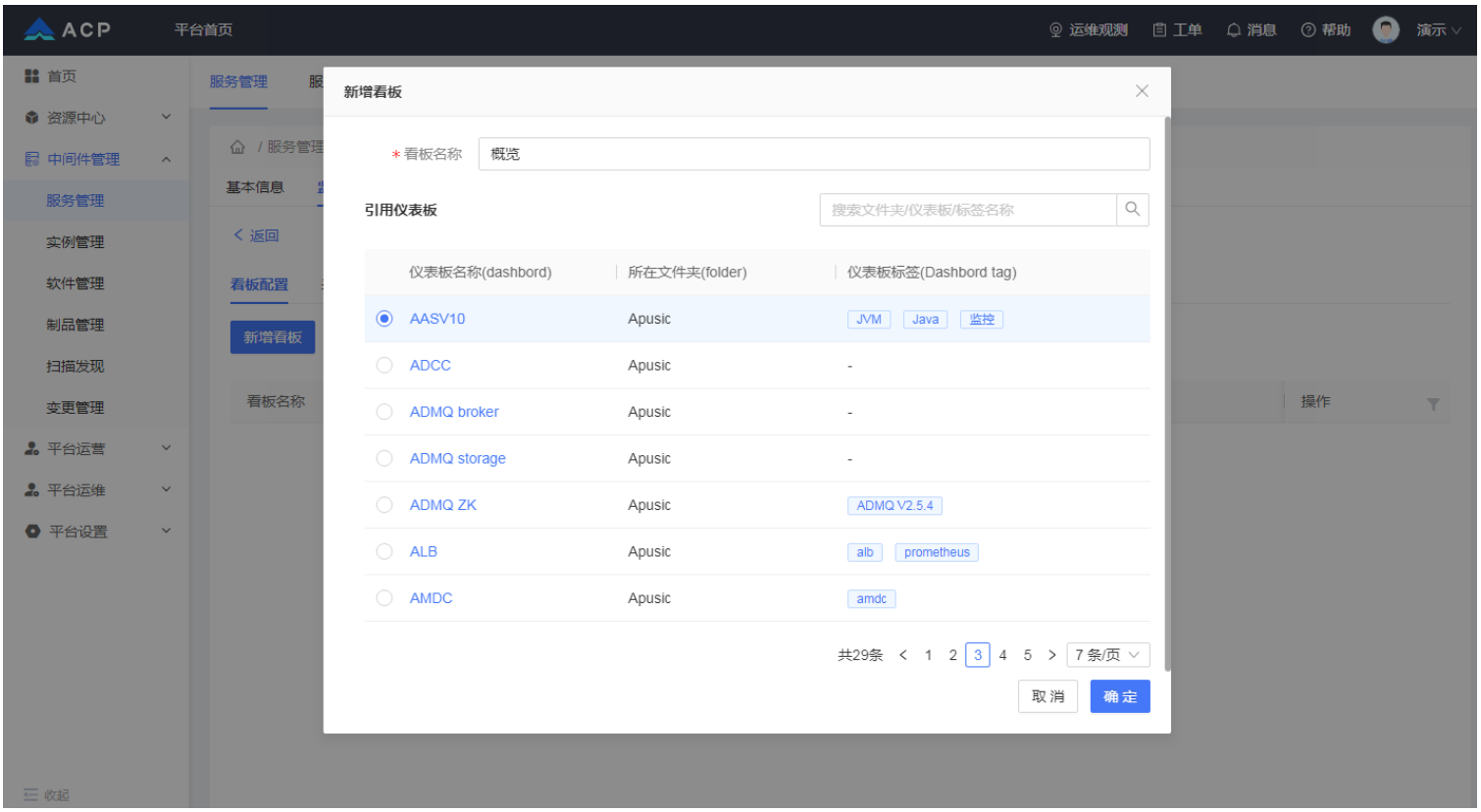
在监控配置页面点击【采集配置】-【新增采集配置】添加prometheus采集器



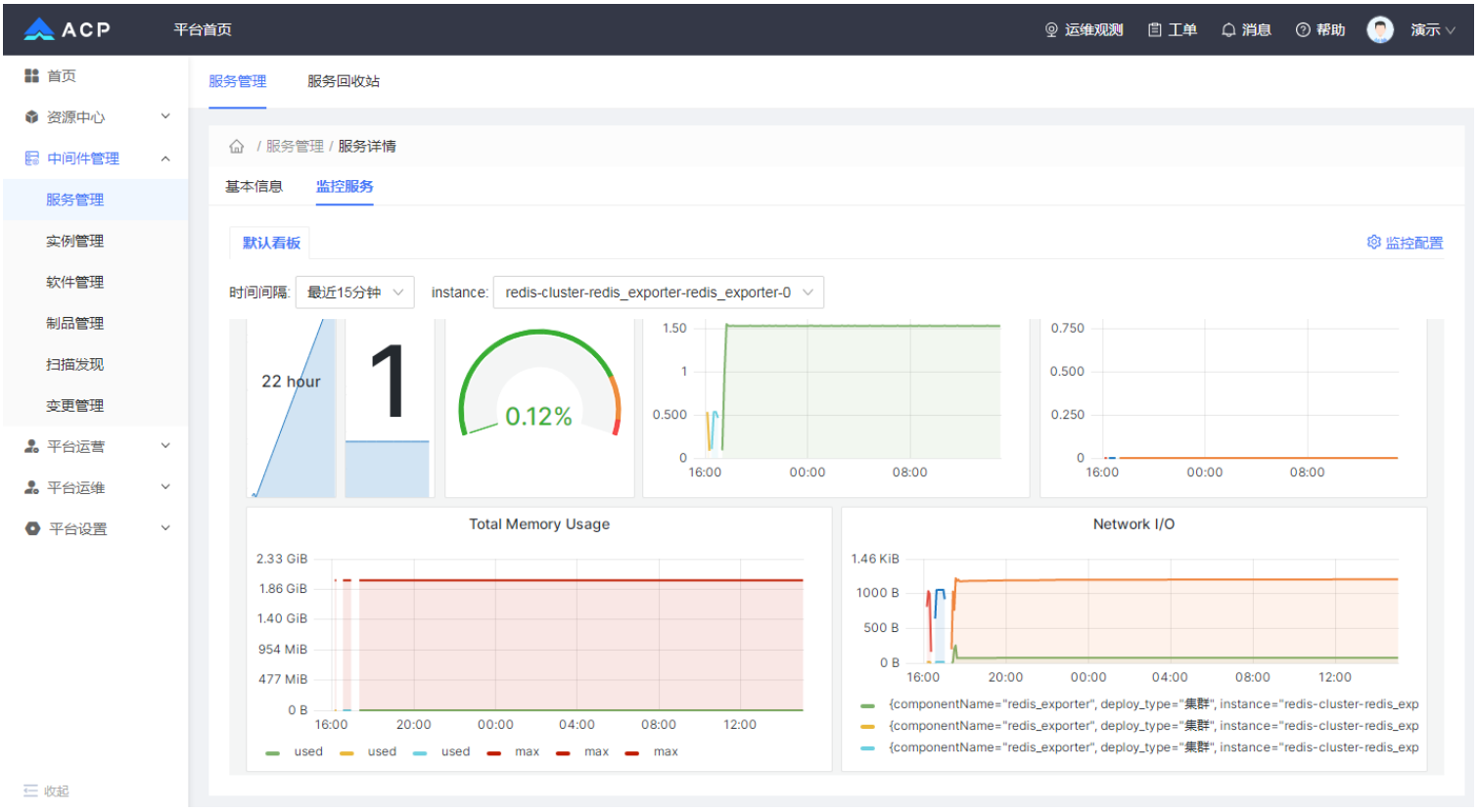
> *提示：建议先验证采集地址是否可用*

看板配置

在监控配置页面点击【采集配置】-【看板配置】-【新增看板】，可以在平台内置的看板中选择对应中间件的仪表板。



点击【确定】，然后返回【监控服务】，看到看板成功展示，则表示配置成功。



6.5 管理中间件

6.5.1 服务管理

6.5.1.1 服务启停

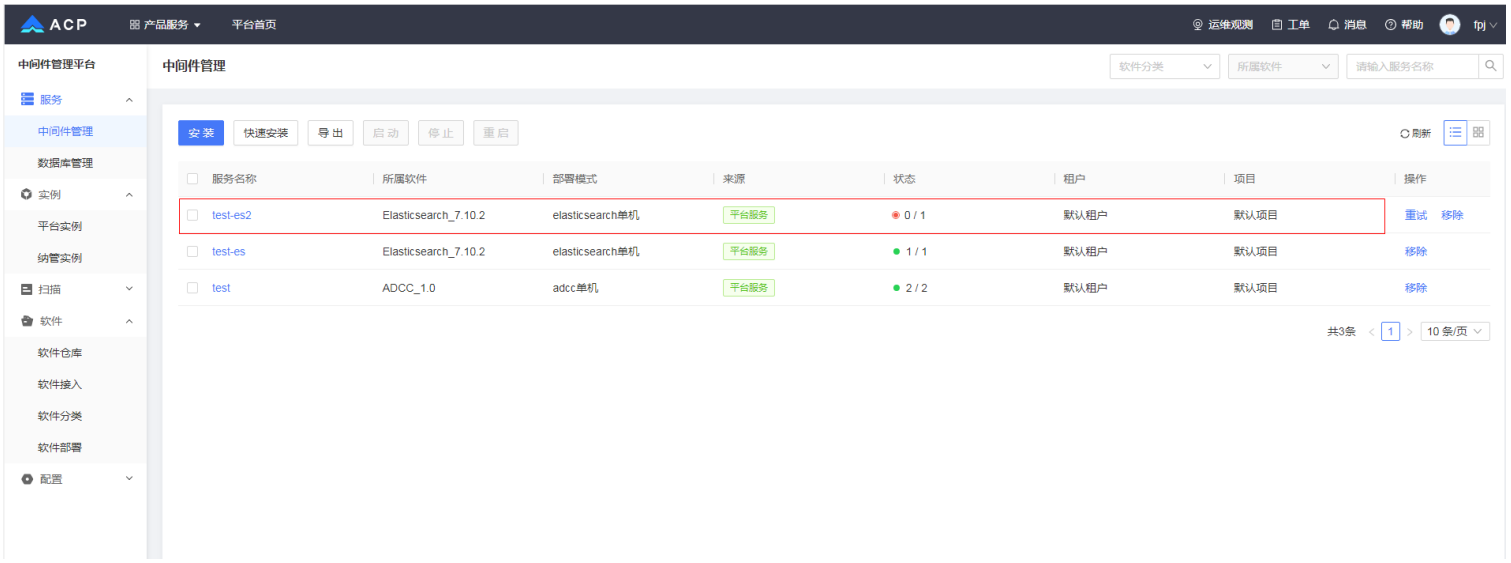
进入【中间件管理】-【服务管理】，选中目标服务，点击【启动】/【停止】/【重启】，服务状态将实时更新。

服务列表"状态"栏展示服务实例的状态，“0/1”表示正常实例/全部实例。

服务状态颜色标识如下：

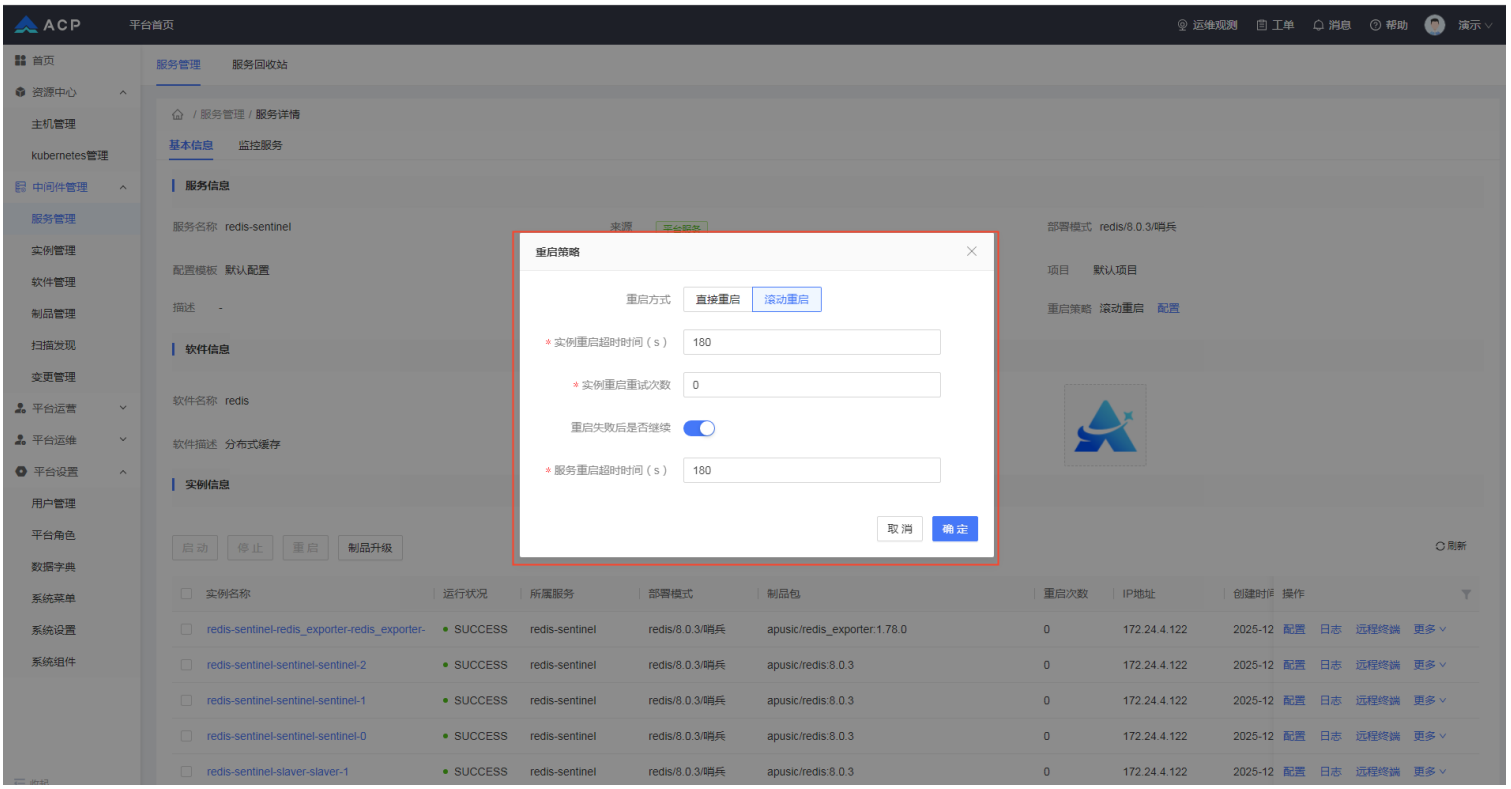
- 红色：表示全部实例异常或者已停止
- 橙色：表示部分实例异常或者已停止
- 绿色：表示全部实例正常运行

如下图服务总共1个实例，0个正常。



重启策略

ACP平台默认采用滚动重启的方式。你可以在【服务详情】-【服务信息】-【重启策略】中设置重启策略。



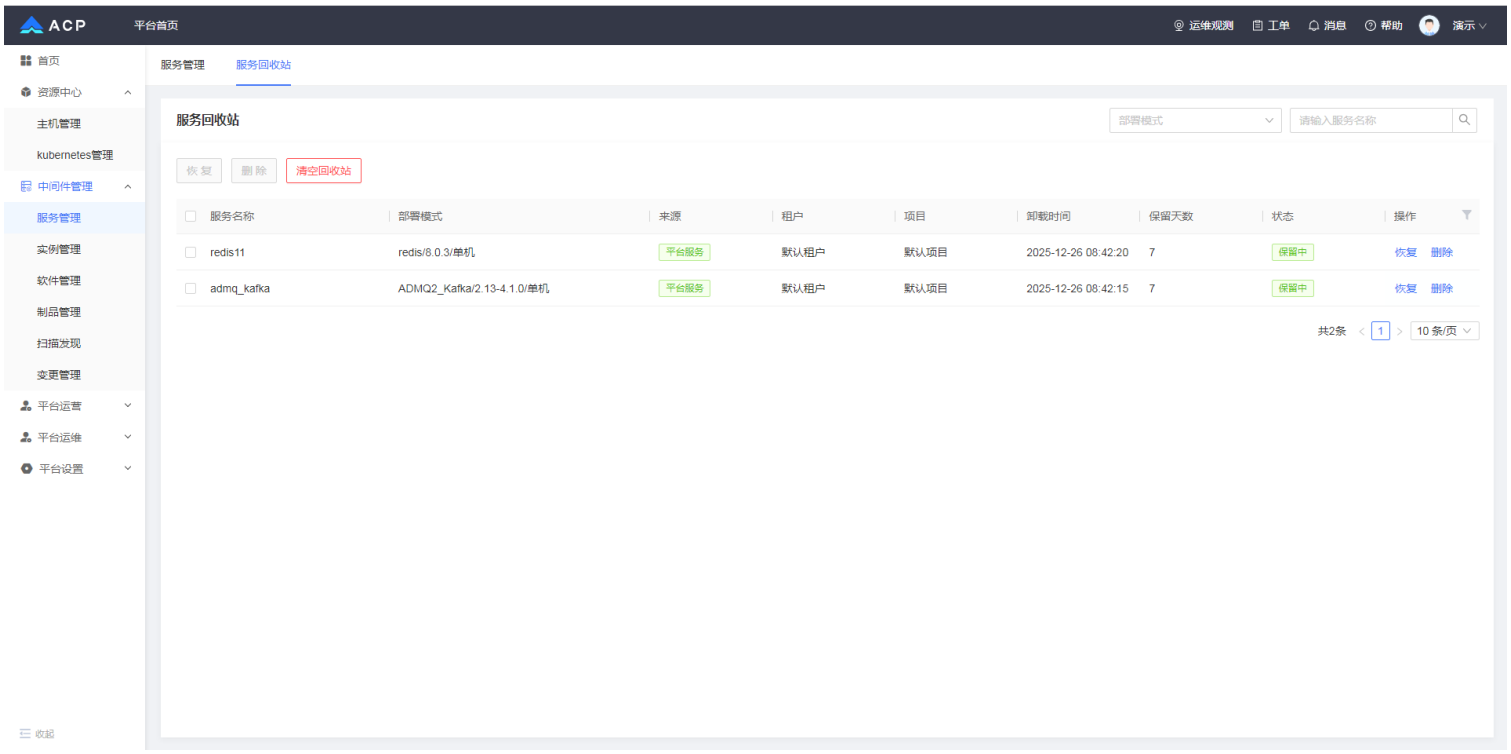
6.5.1.2 服务卸载

点击服务列表右侧【卸载】按钮，卸载服务。

- 卸载后，服务不会立马删除，而是会放入服务回收站。
- 卸载后，目标主机上的中间件实例会被停止。

6.5.1.3 服务回收站

服务回收站可对被卸载的服务进行恢复、删除等操作。



服务恢复

点击【中间件管理】-【服务管理】-【服务回收站】，进入服务回收站。点击服务列表右侧【恢复】按钮，即可恢复服务。服务恢复后，可以在服务列表查看服务状态。

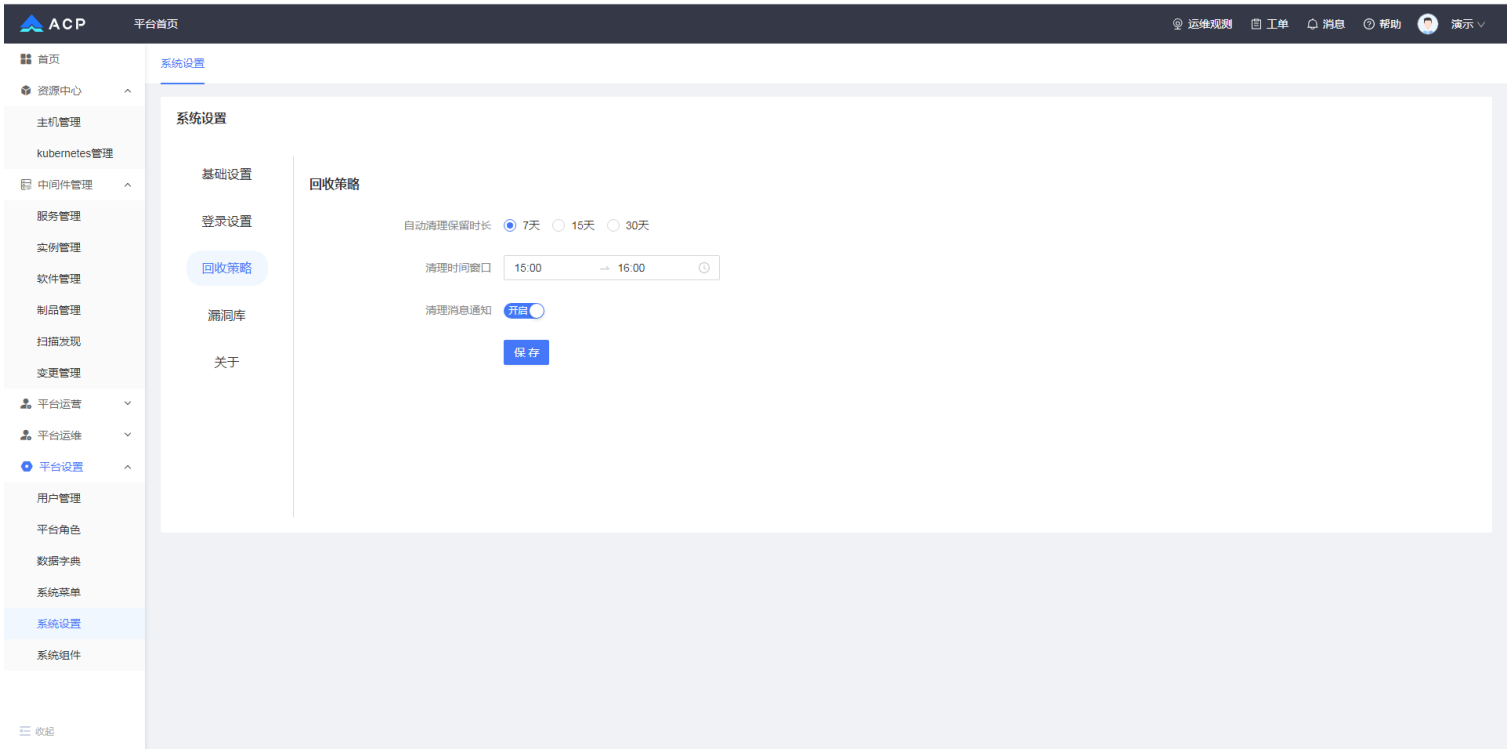
服务删除

点击服务列表右侧【删除】按钮，即可删除服务。

- 删除后，服务会彻底删除。
- 删除后，目标主机上的中间件实例会被卸载并删除。

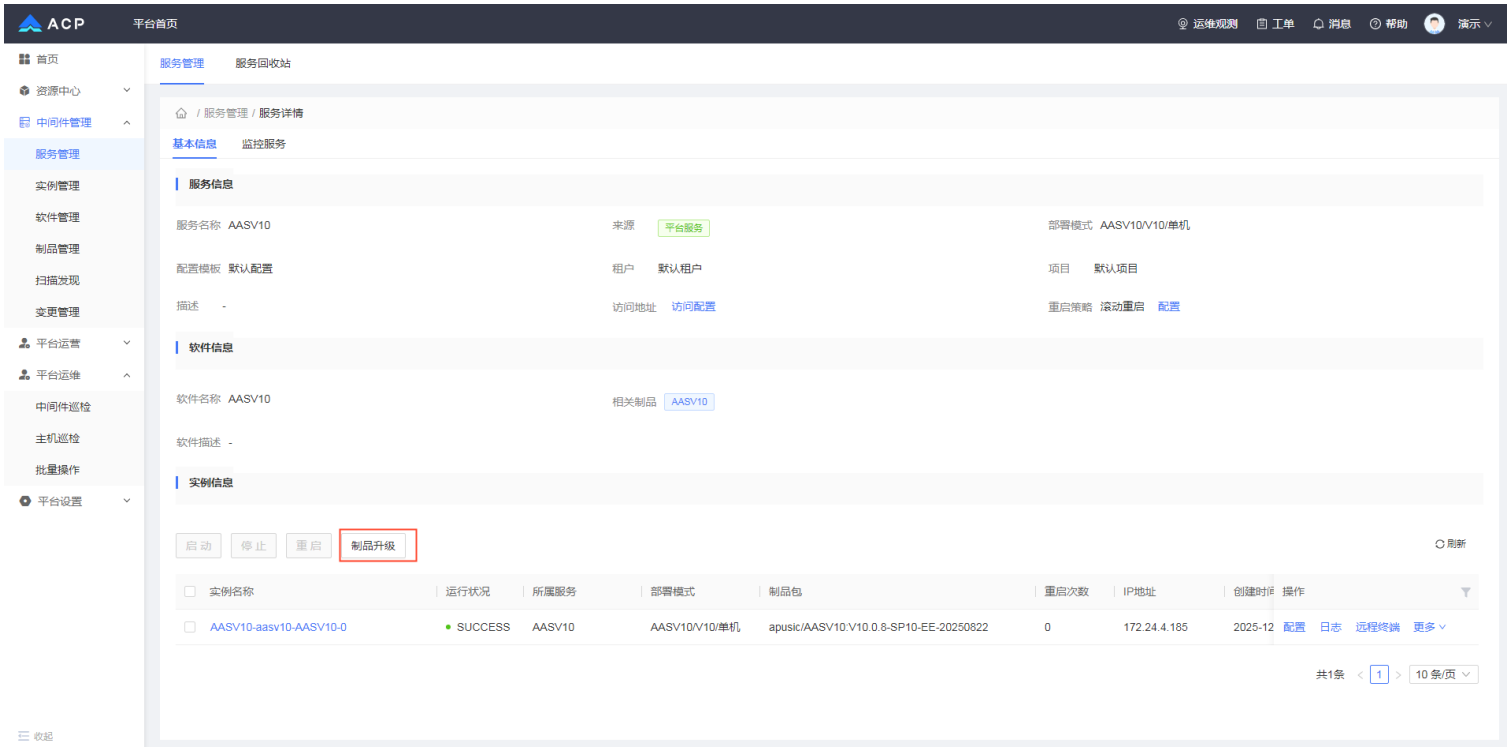
自动回收

平台会根据回收策略定期删除回收站中的服务。你可进入【平台设置】-【系统设置】-【回收策略】配置回收策略。



6.5.1.4 版本升级/回滚

进入【中间件管理】-【服务管理】-【服务详情】，点击实例列表上部【制品升级】，即可进入升级表单。



升级表单如下：

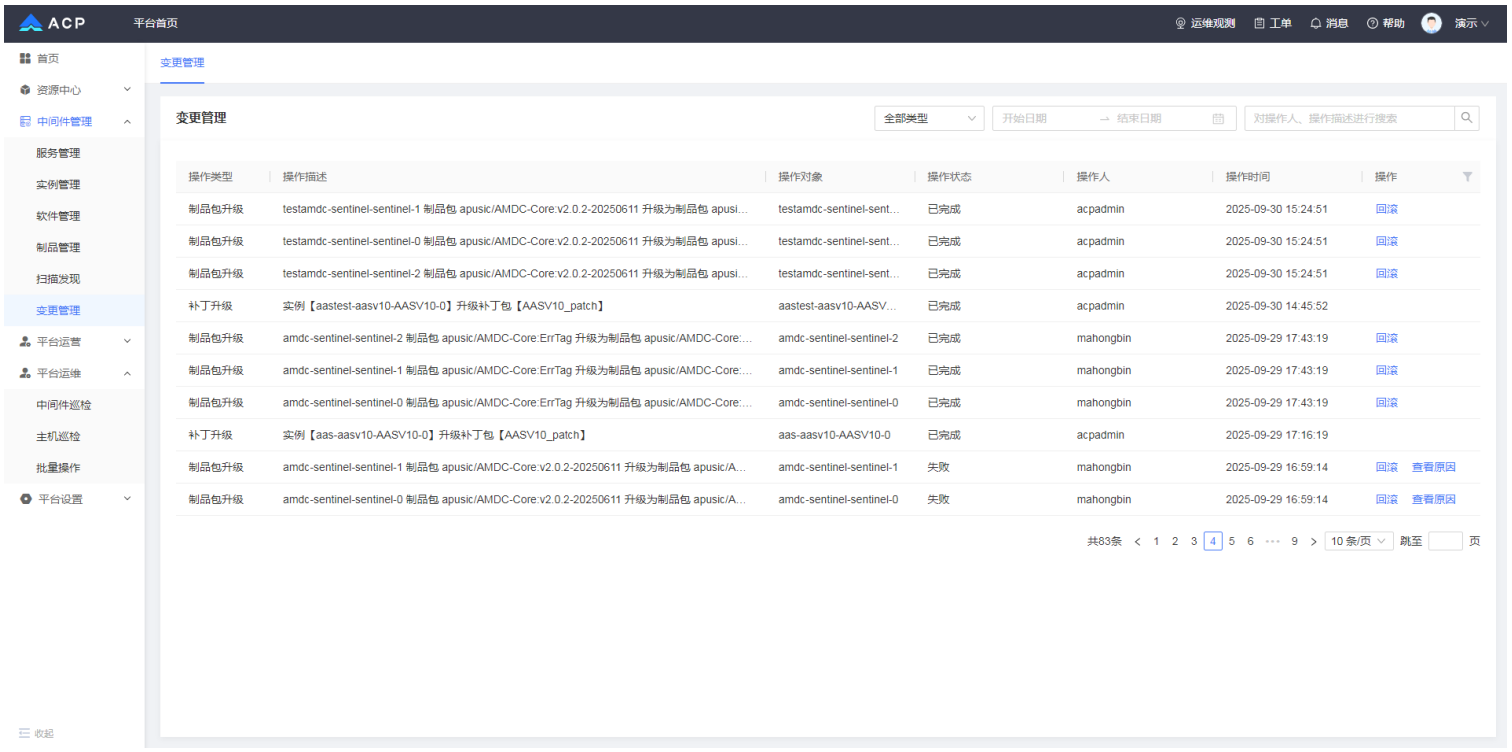
- 部件：即本次升级的该软件下的部件。

- 制品包：需要升级到的制品包
- 升级策略：支持滚动神级与重建升级
- 最大不可用实例数：升级期间不可用实例的最大数量

> *提示：*
>
> - *滚动升级：默认策略，逐步替换旧实例。通过以下参数控制升级过程：*
>
> *最大不可用实例数：指定升级期间不可用实例的最大数量。例如，3节点zookeeper集群，配置2可确保升级过程中始终有2的可用实例，从而保证集群可用性。*
>
> - *重建升级：先停止所有旧实例，再一次性启动新实例。此策略会导致短暂服务中断。*

点击【确定】即可完成升级。

制品包升级后可在变更管理中查看相关记录

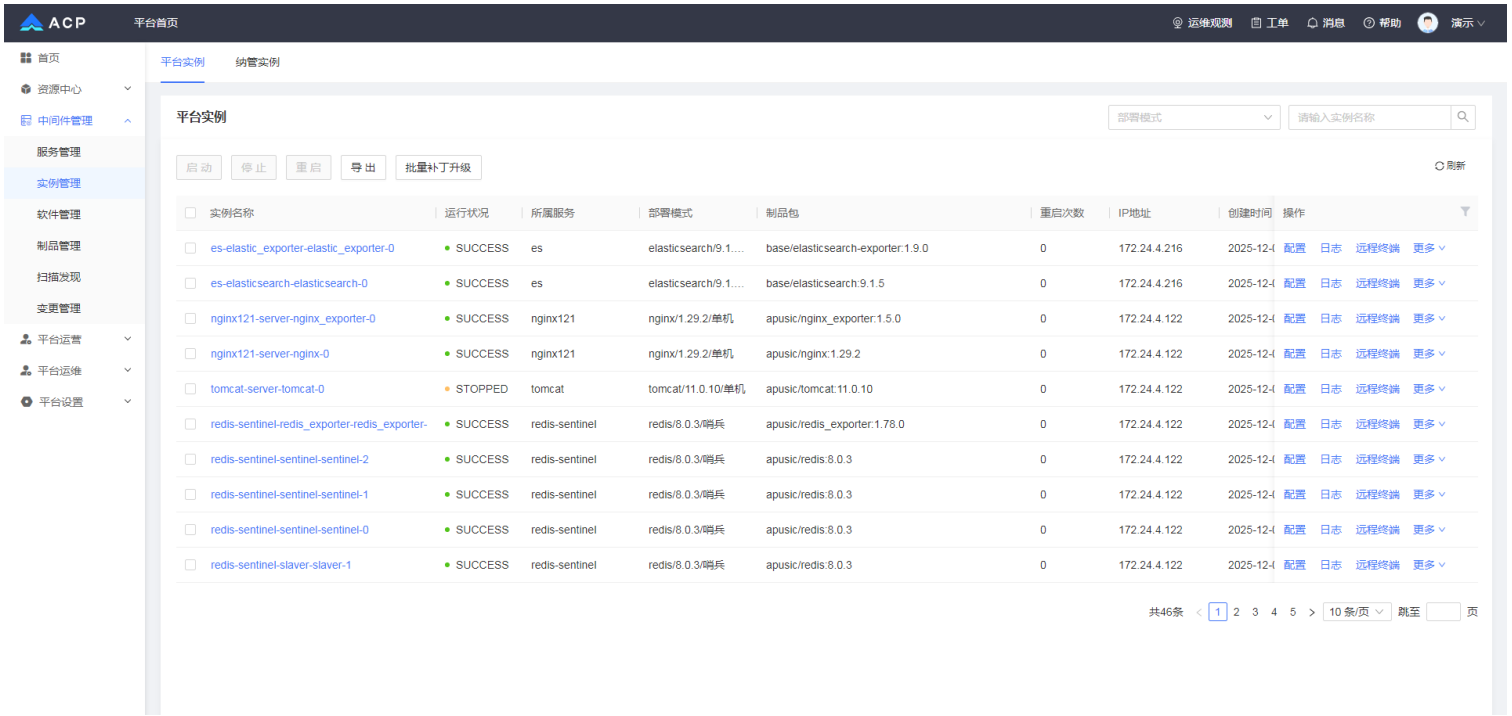


点击【回滚】按钮，即回滚到上个版本。

6.5.2 实例管理

6.5.2.1 实例启停

进入【中间件管理】-【实例管理】，选中目标中间件（如 Redis），点击【停止】/【启动】，状态将实时更新



6.5.2.2 批量启停

支持同时对多个实例进行启停操作，提高运维效率：

1. 进入【中间件管理】-【实例列表】
2. 勾选需要操作的多个中间件实例
3. 点击列表上方的【批量启动】或【批量停止】按钮
4. 系统将依次执行操作，可在列表中查看各实例状态变化

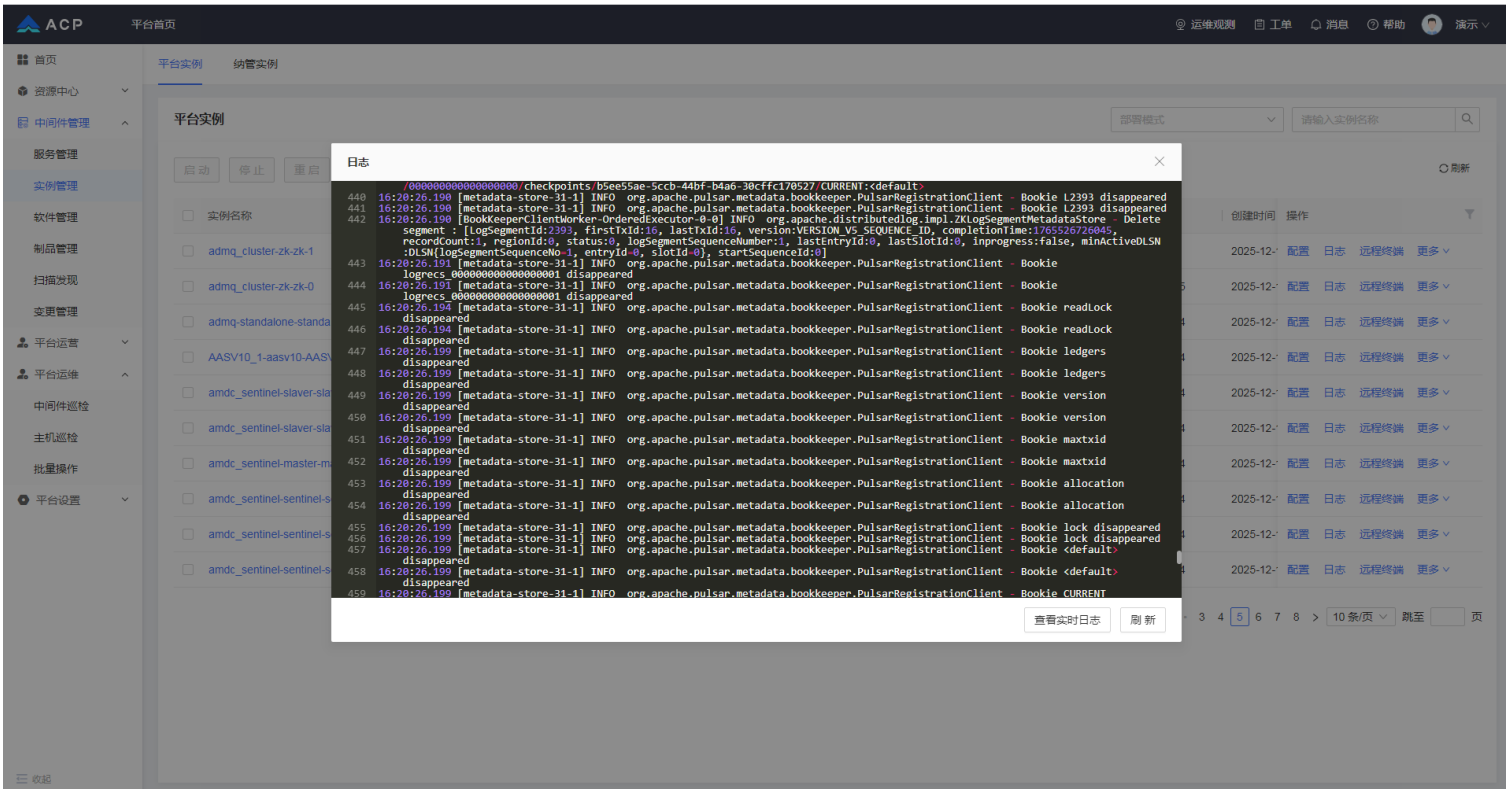
6.5.2.3 日志查看

在ACP平台钟，日志查看分以下为3个场景：

- 标准输出日志：通常为启动脚本输出的日志
- 文件日志：重定向到文件的日志
- 日志中心：通过日志中心统一采集、查询

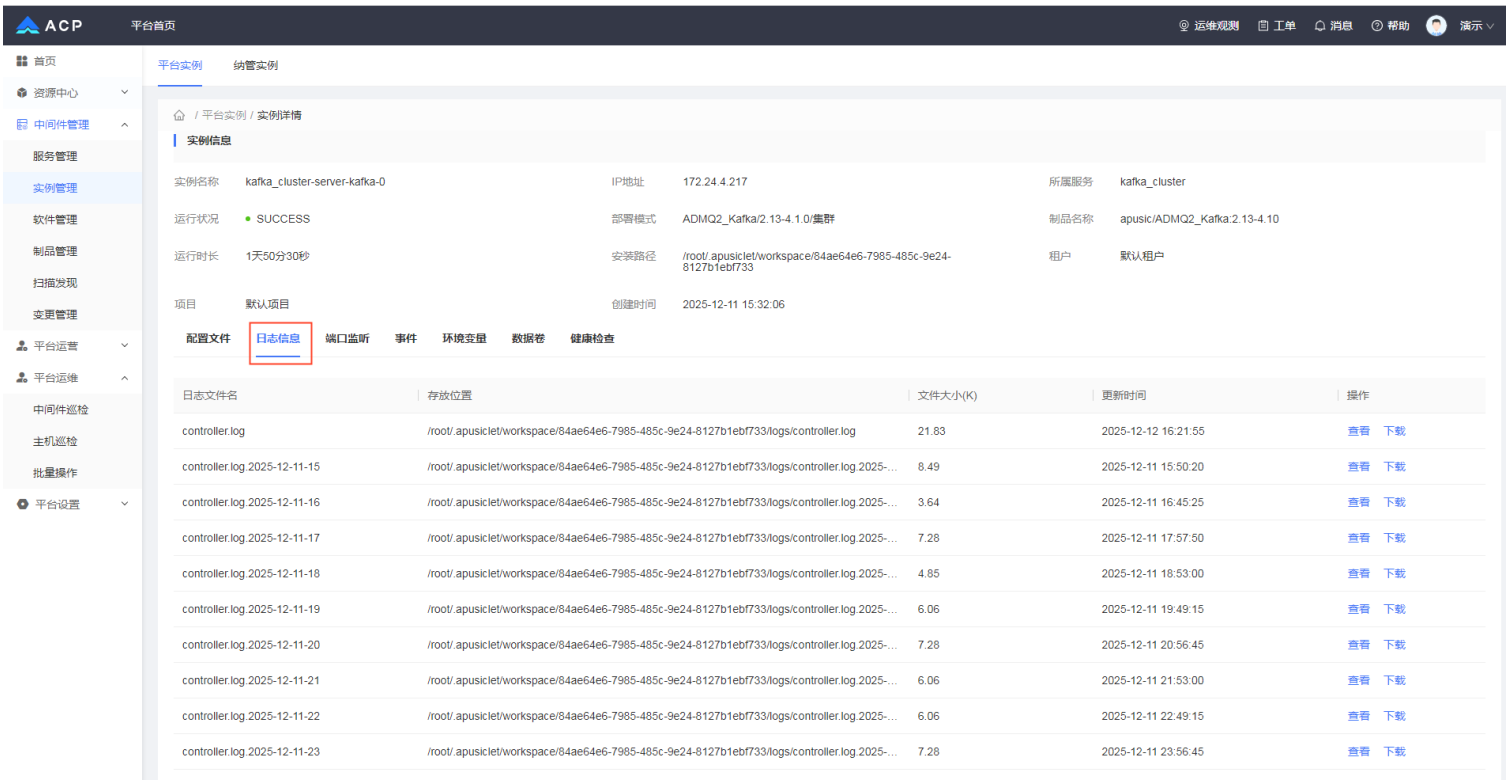
标准输出日志

进入【中间件管理】-【实例管理】-【平台实例】，点击实例列表右侧【日志】按钮，即可查看标出输出日志。



文件日志

进入实例详情页，点击【日志信息】，即可查看日志文件。



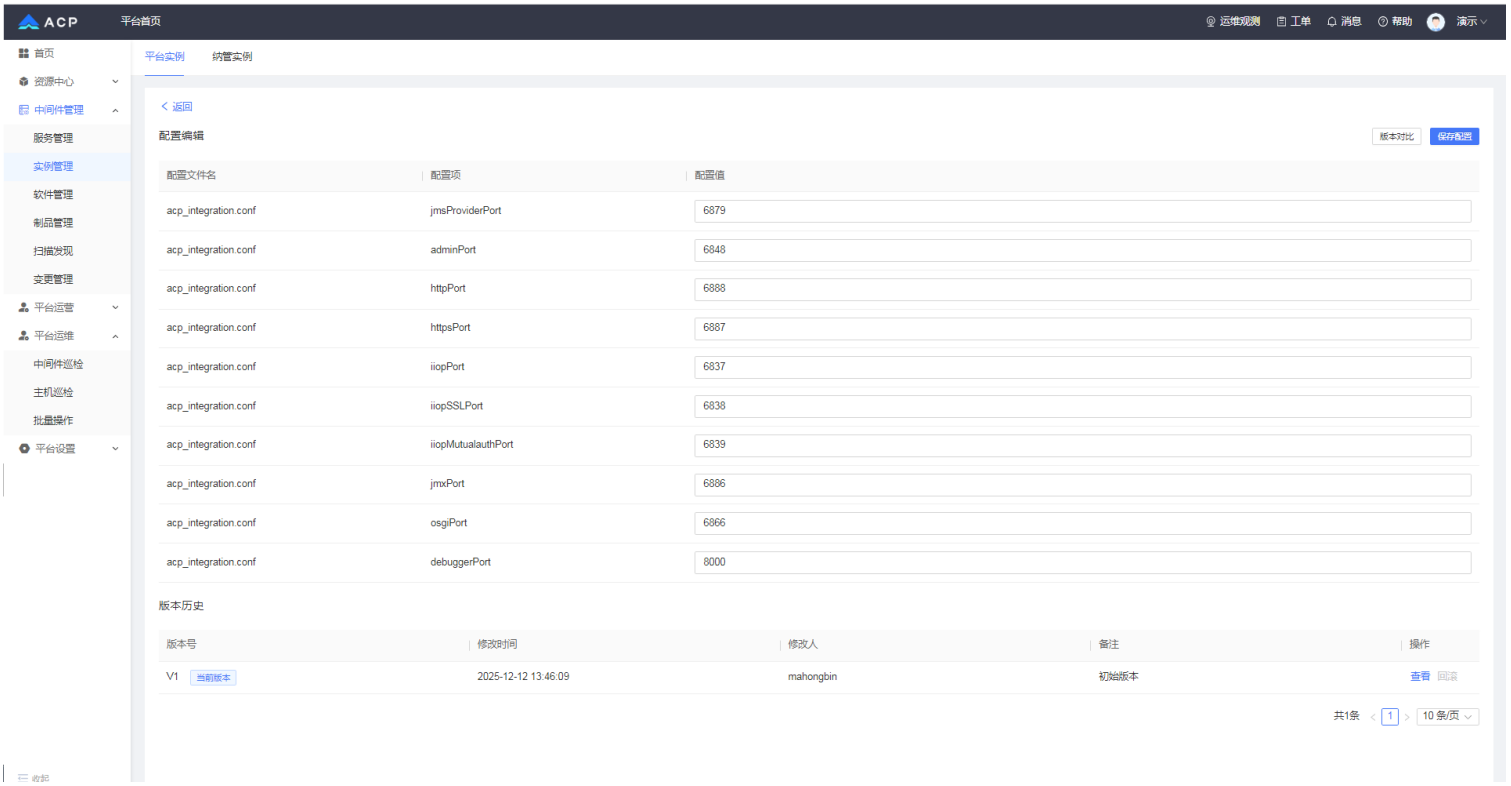
日志中心

请参考下文《日志中心》章节。

6.5.2.4 实例配置

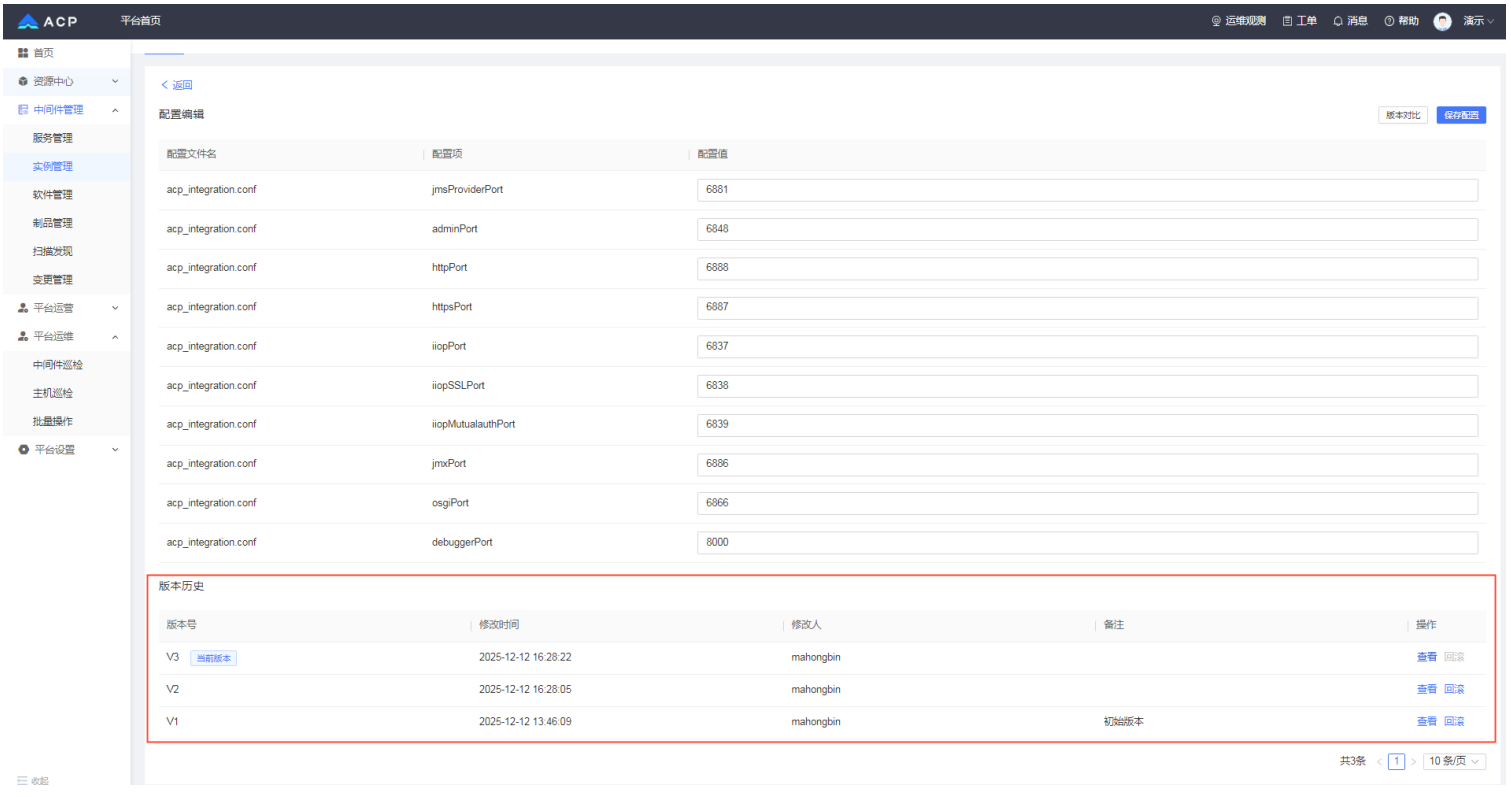
进入【实例管理】-【平台实例】，点击实例列表右侧【配置】按钮，即可进入配置页。

修改配置表单中的配置项，点击【保存配置】，即修改完成



配置版本

ACP支持配置的版本控制，在配置表单页面，可以查看当前实例的配置版本历史。

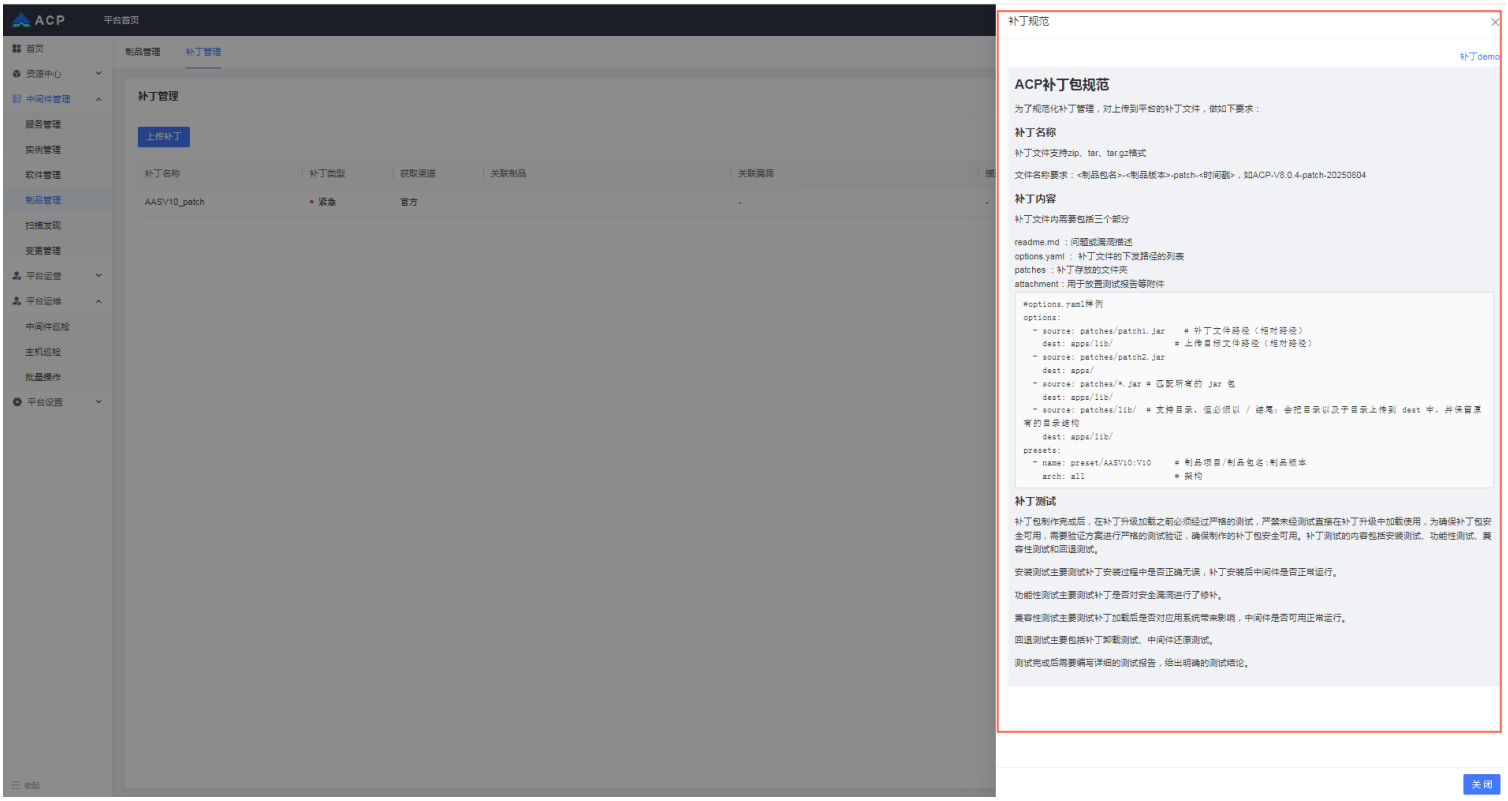


6.5.2.5 补丁升级

补丁是常见的运维操作，ACP提供批量补丁升级功能。

补丁管理

进入【中间件管理】-【制品管理】-【补丁管理】补丁管理页，点击右上角【补丁规范】，即可查看。



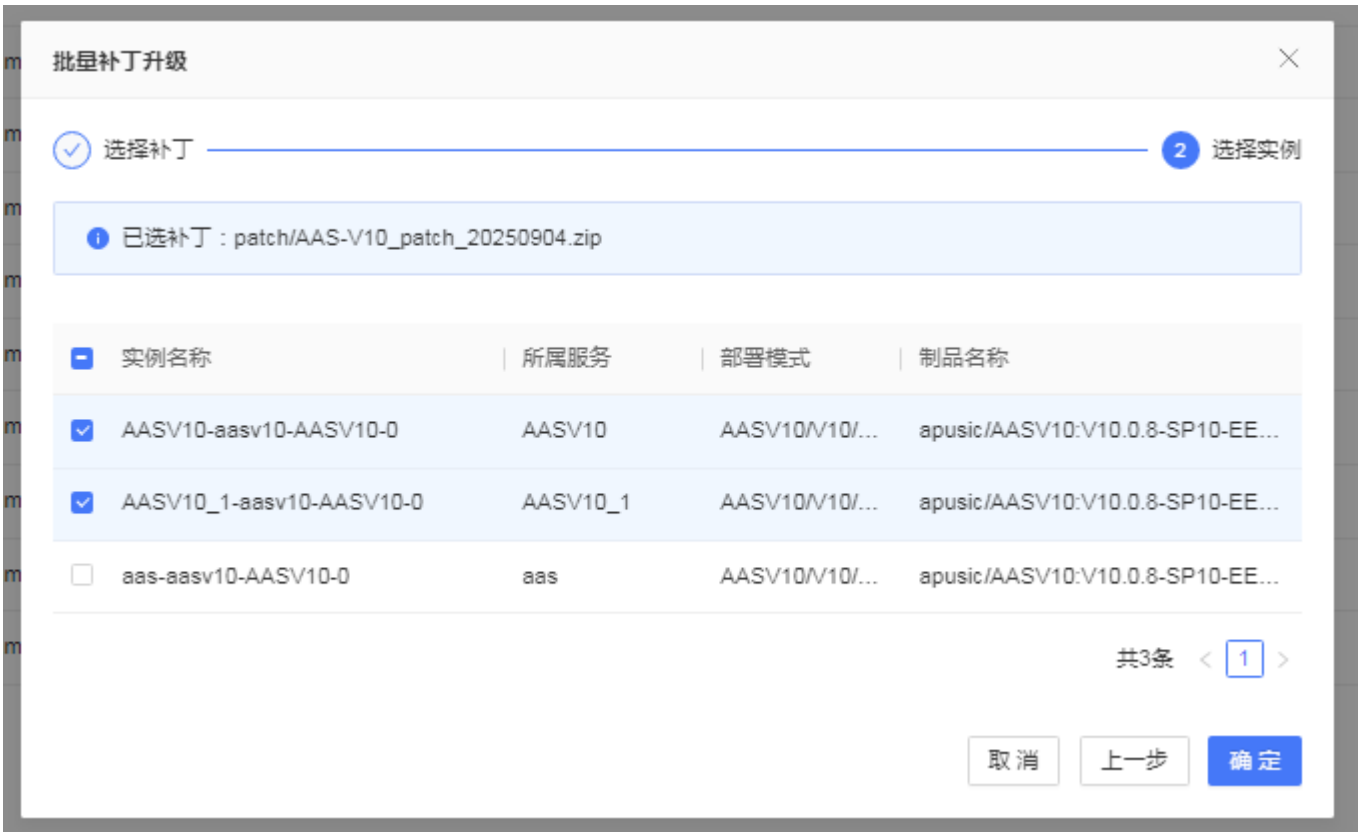
需要根据ACP平台规范制作补丁，并上传。

> *提示：option.yaml中的制品版本、架构需要与平台制品包保持一致*

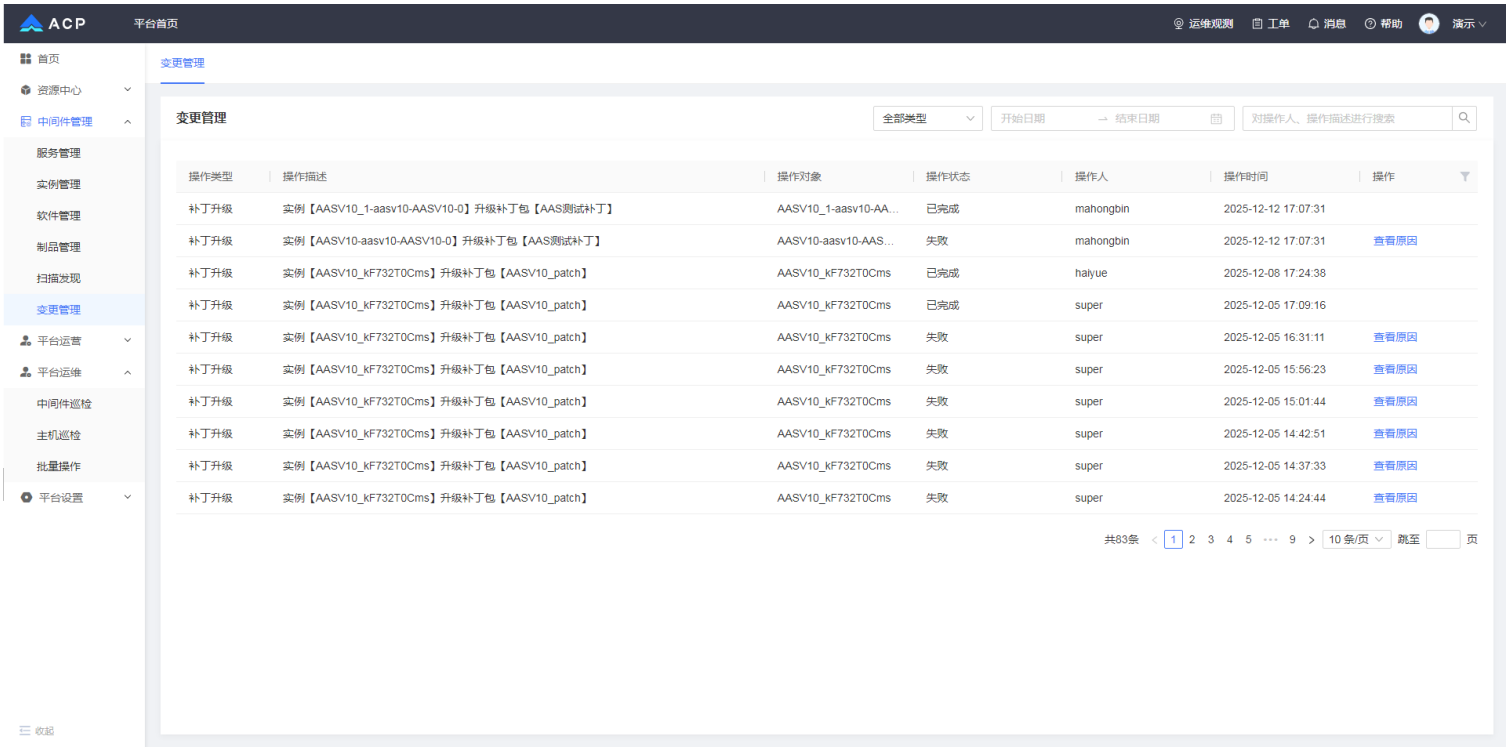
批量补丁

进入【中间件管理】-【实例管理】-【平台实例】|【纳管实例】，点击【批量补丁升级】按钮，进入补丁升级流程。

选择补丁及关联的中间件实例，点击【确定】，即可完成。



进入【中间件管理】-【变更管理】页面，即可查看补丁操作的结果。如若失败，可查看具体失败原因。



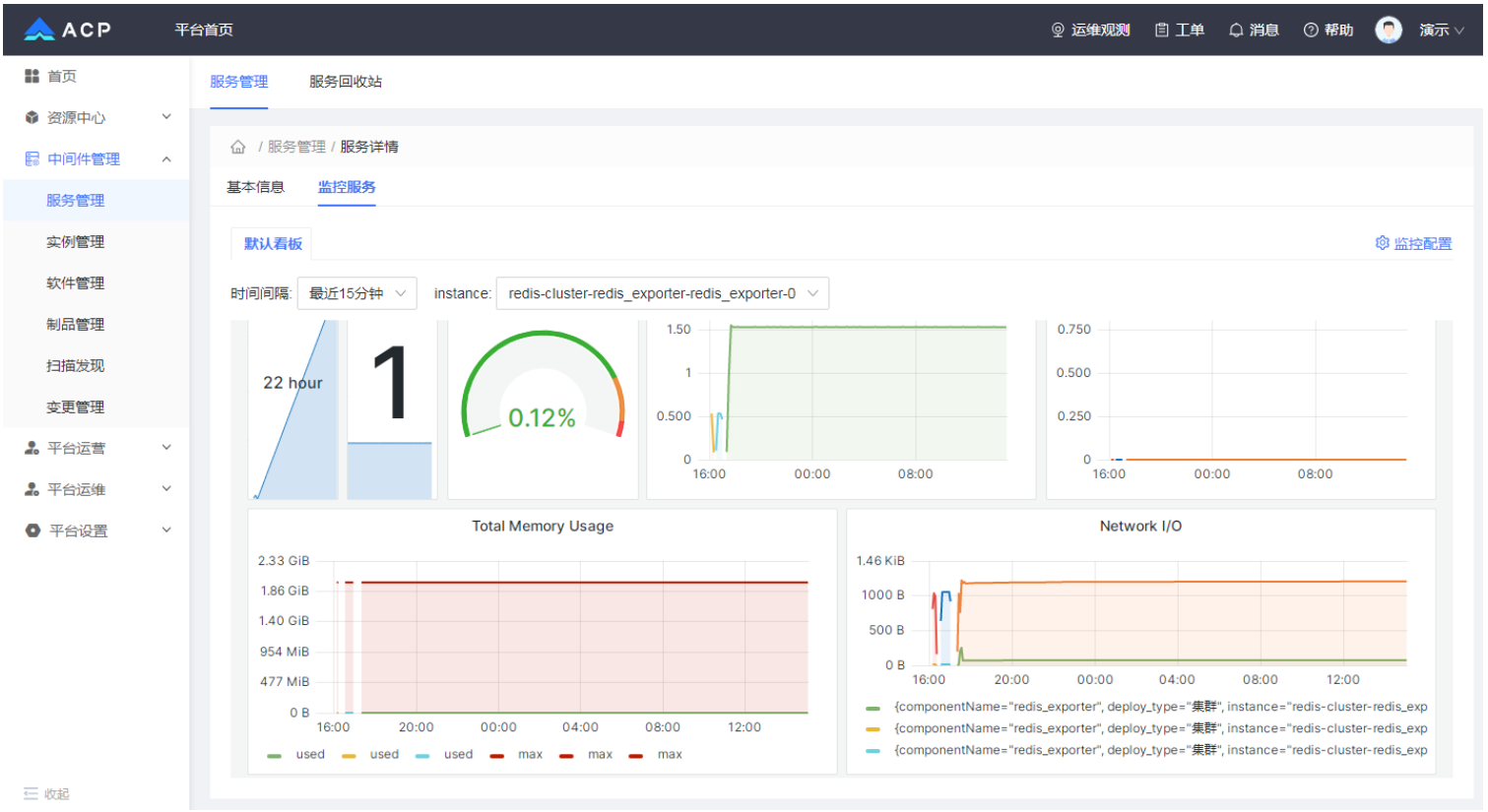
6.6 运维中间件

6.6.1 指标监控

基于prometheus技术栈实现的系统监控，用于可量化关键性能指标（如CPU使用率、请求延迟、错误率等）的采集、展示、告警。

6.6.1.1 监控看板

进入【中间件管理】-【服务管理】-【服务详情】，点击监控服务，可查看该服务的监控看板（支持多个看板）：



右侧可进入监控配置，配置分为看板配置及采集配置：

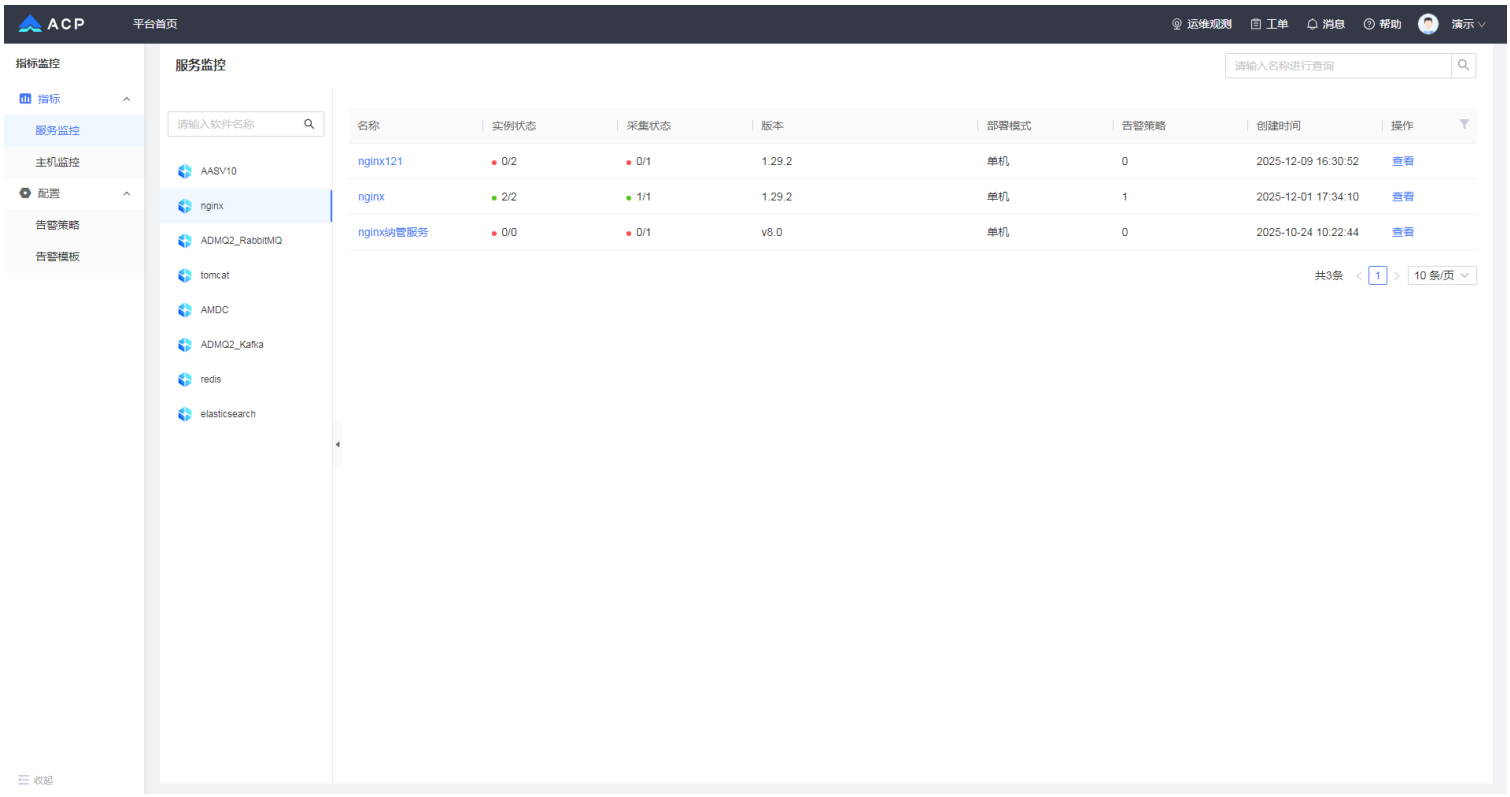
- 看板配置：点击新增看板选择相关的仪表盘即可成功新增自定义看板
- 采集配置：填写采集配置相关表单，系统将实时获取采集状态

> *提示：具体操作可参考<纳管中间件>-<服务监控>章节*

6.6.1.2 监控中心

监控中心用于平台指标监控的集中管理，包含了主机、中间件监控，告警策略设置等。

点击头部菜单【运维观测】-【指标监控】，即可进入指标监控中心，在这里你可以看到整个平台的中间件、主机的监控情况，包括实例监控列表，实例状态、采集状态，告警配置、告警事件等。



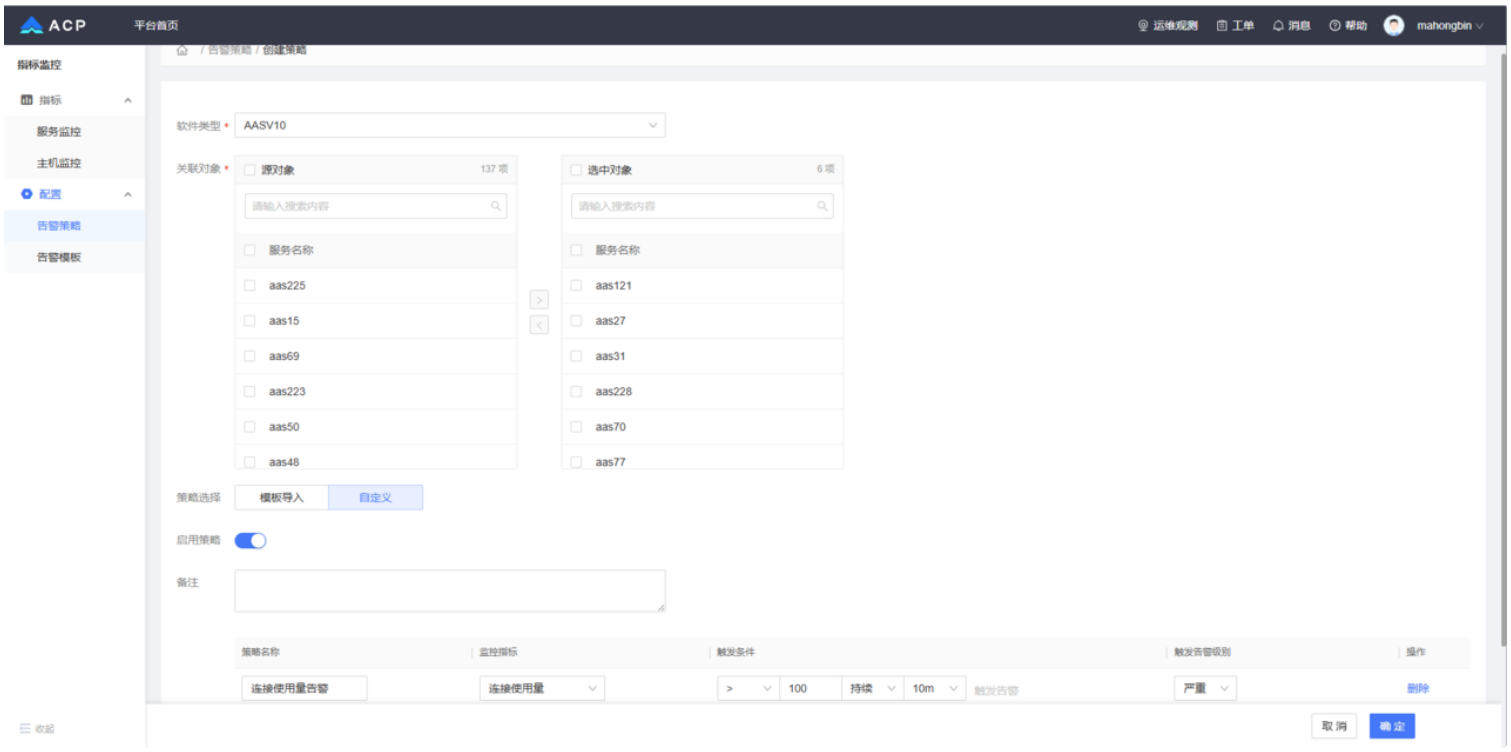
6.6.1.3 指标告警

告警策略（亦称告警规则）是监控系统的核心配置，用于定义当指定指标在预设时间窗口内达到或超出阈值条件时，自动触发告警通知的逻辑规则。其核心要素包括：监控对象（指标）、评估周期（时间粒度）、阈值条件（静态/动态阈值）等，通过精准配置可实现异常状态的及时发现与响应。

点击【配置】-【告警策略】-【添加】，即可进入告警策略创建页面，页面主要信息包含：

- 软件类型：本次告警设置的软件类型，数据来源于软件管理
- 关联对象：需要设置告警策略的中间件服务，支持批量设置
- 策略选择：支持通过告警模板、自定义两种设置方式
- 策略项：告警配置的核心，每个策略项对应一条告警规则。包含了监控指标、触发（阈值）条件、告警级别的设置。

> *提示：*
>
> - *平台会内置一些告警模板，供实际使用参考*
> - *监控指标从监控看板中自动提取得到*

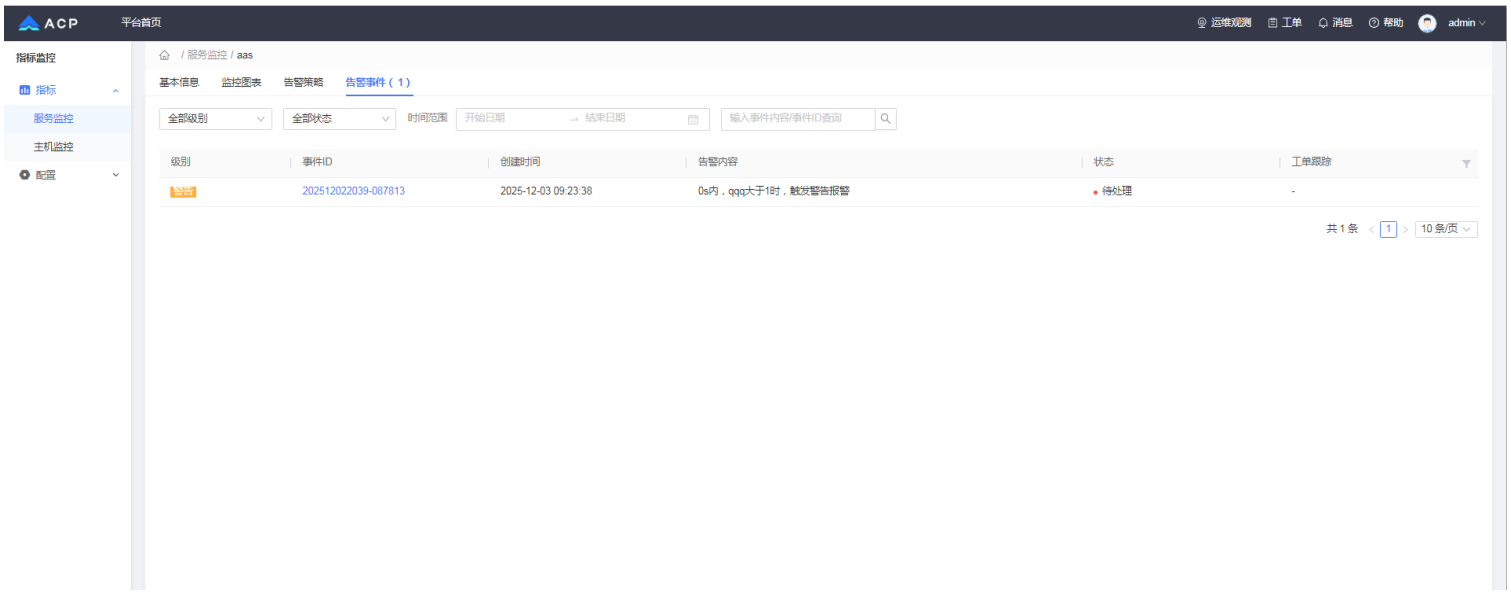


假设有一天告警规则：当连接使用量连续10分钟以上超过100个时，可判断系统过载，需要立马介入处理，则可进行如下设置：

告警事件

当告警规则触发时，会产生告警事件。

进入【指标】-【服务监控】，在监控服务详情页，可以查看当前服务的告警事件。



> *提示：告警事件会统一推送至告警中心。事件的处理、通知配置均需要在告警中心完成。*

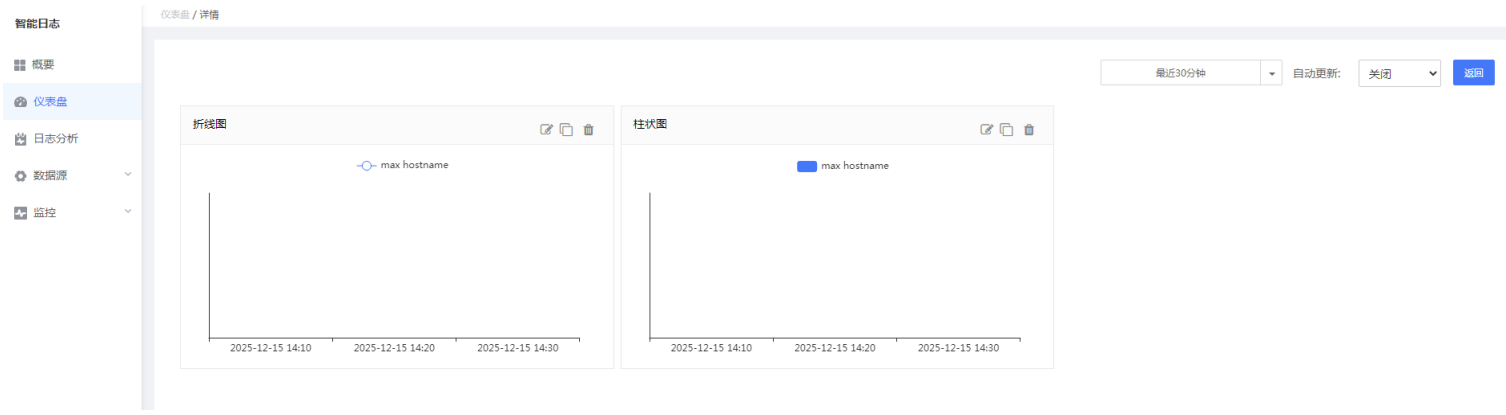
6.6.2 日志中心

日志中心用于平台部署中间件服务日志的集中管理，包含了日志数据统计，查询，日志告警配置等。

点击头部菜单【运维观测】-【日志中心】，即可进入日志中心，在这里你可以看到整个平台部署的中间件的日志数据，包括日志概览，日志查询，日志告警配置等。

6.6.2.1 仪表盘

仪表盘通过可视化配置的方式，满足用户不同场景的需求自定义日志分析。支持图形面 板、数字面板以及表格面板。其中图形面板包括区域图、柱状图、折线图以及饼图。



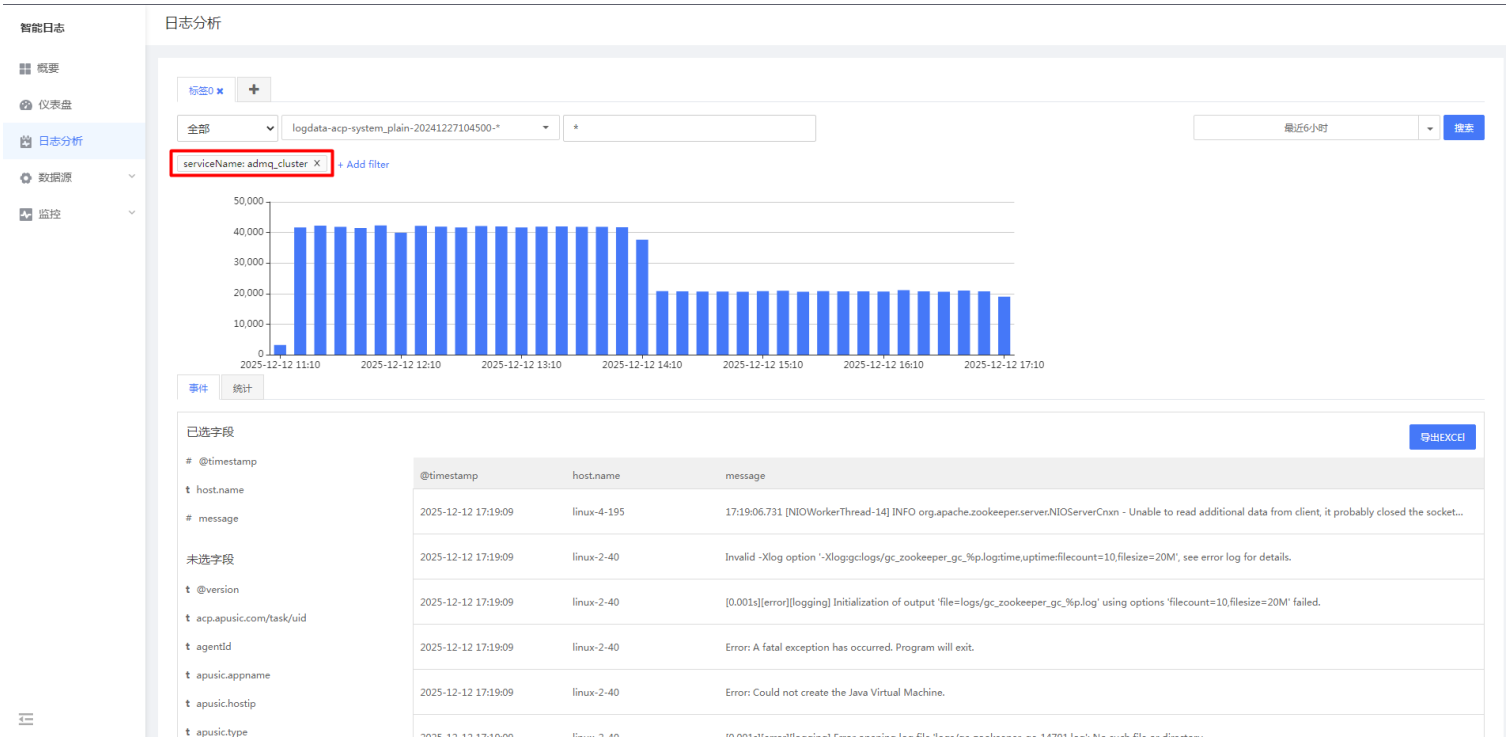
6.6.2.2 日志分析

日志分析（即日志查询）提供了功能强大、简单易用的方式来检索日志，可以快速过滤并找到相关的结果

点击【日志分析】，即可进入日志查询页面测试，支持根据属性查询，以及lucene语法查询。

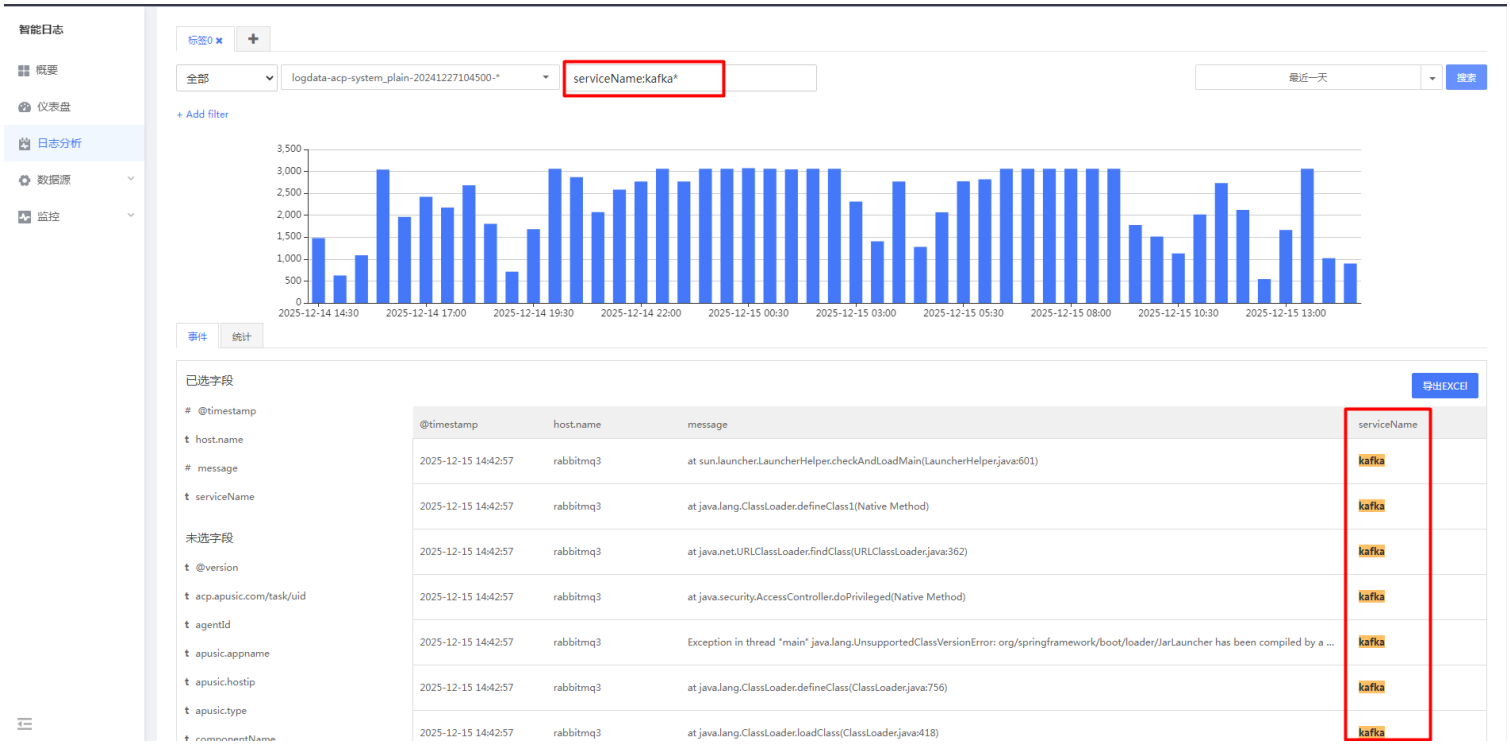
下面给出两个场景。

- 使用属性查询：选择属性字段及其值



- 使用lucene语法：搜索框输入lucene语法进行查询

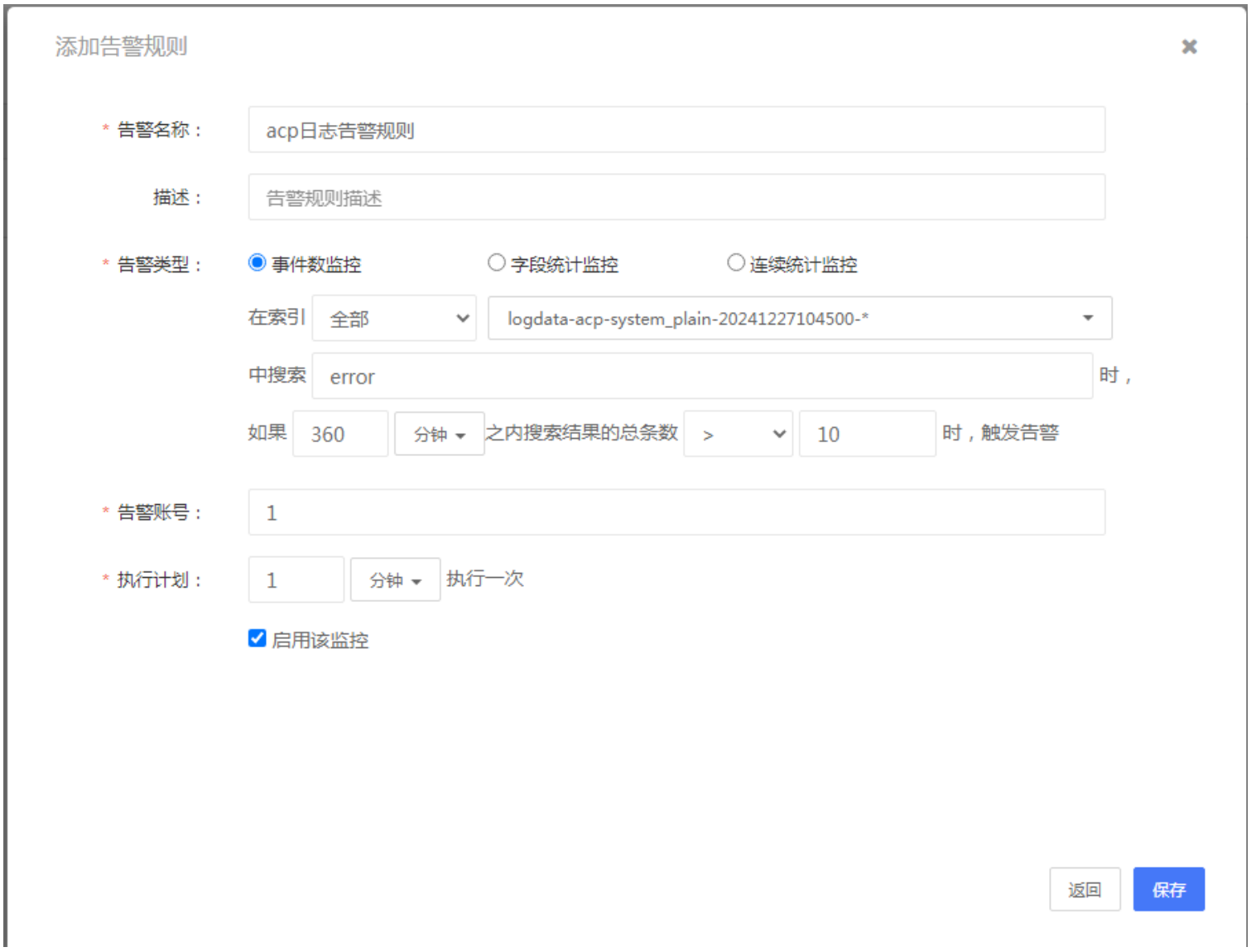
下面是查询服务名serviceName字段的值以kafka开始的日志数据。



6.6.2.3 日志告警

日志中心支持告警，通过检索一段时间内，日志中出现某字段次数触发告警，例如360分钟内出现error的次数大于10，触发告警通知。

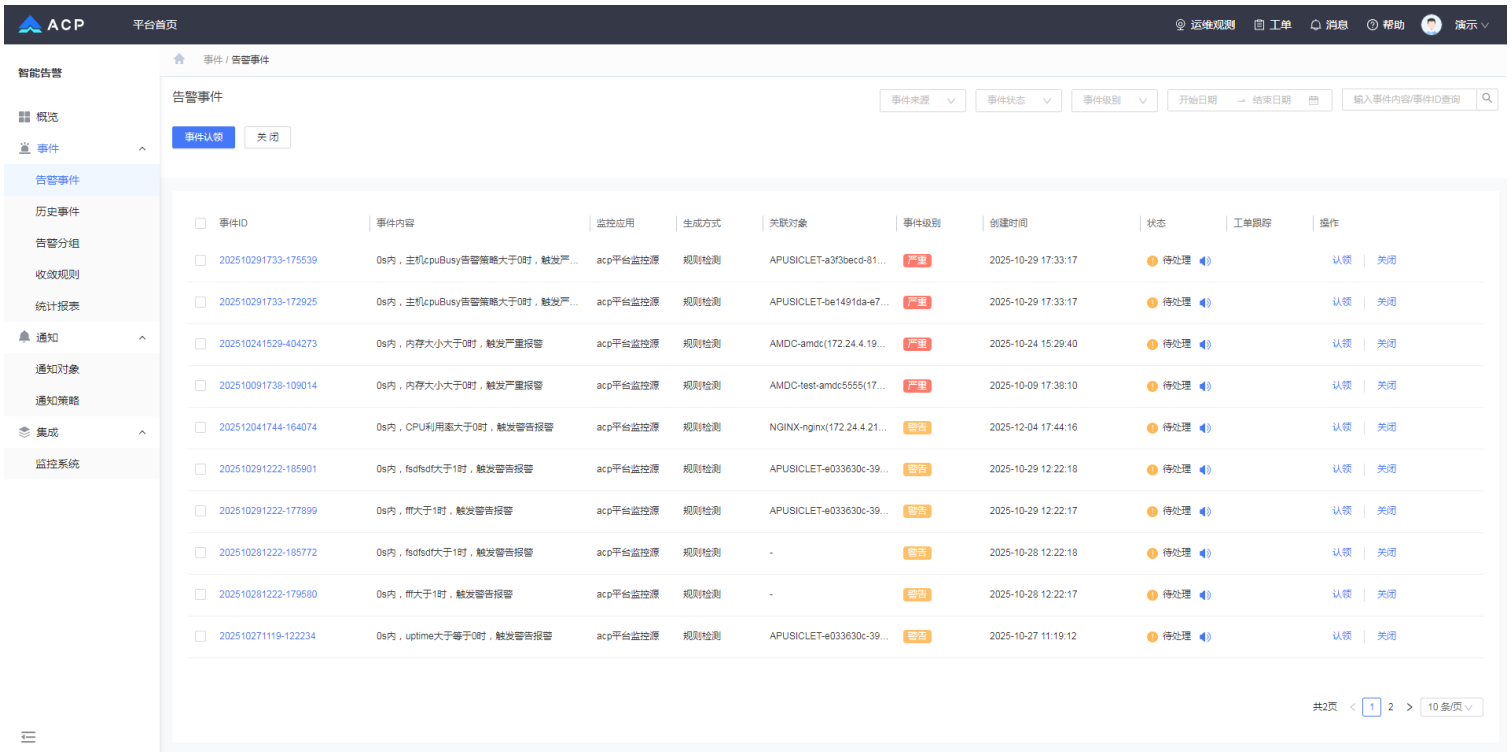
下面是日志告警规则配置。



6.6.3 告警中心

告警中心用于ACP平台各类告警事件的统一处理与通知，包含指标监控、日志等告警事件。

点击头部菜单【运维观测】-【告警中心】，然后进入【事件】-【告警事件】，则可以查看到平台现有的告警事件。



6.6.3.1 告警收敛

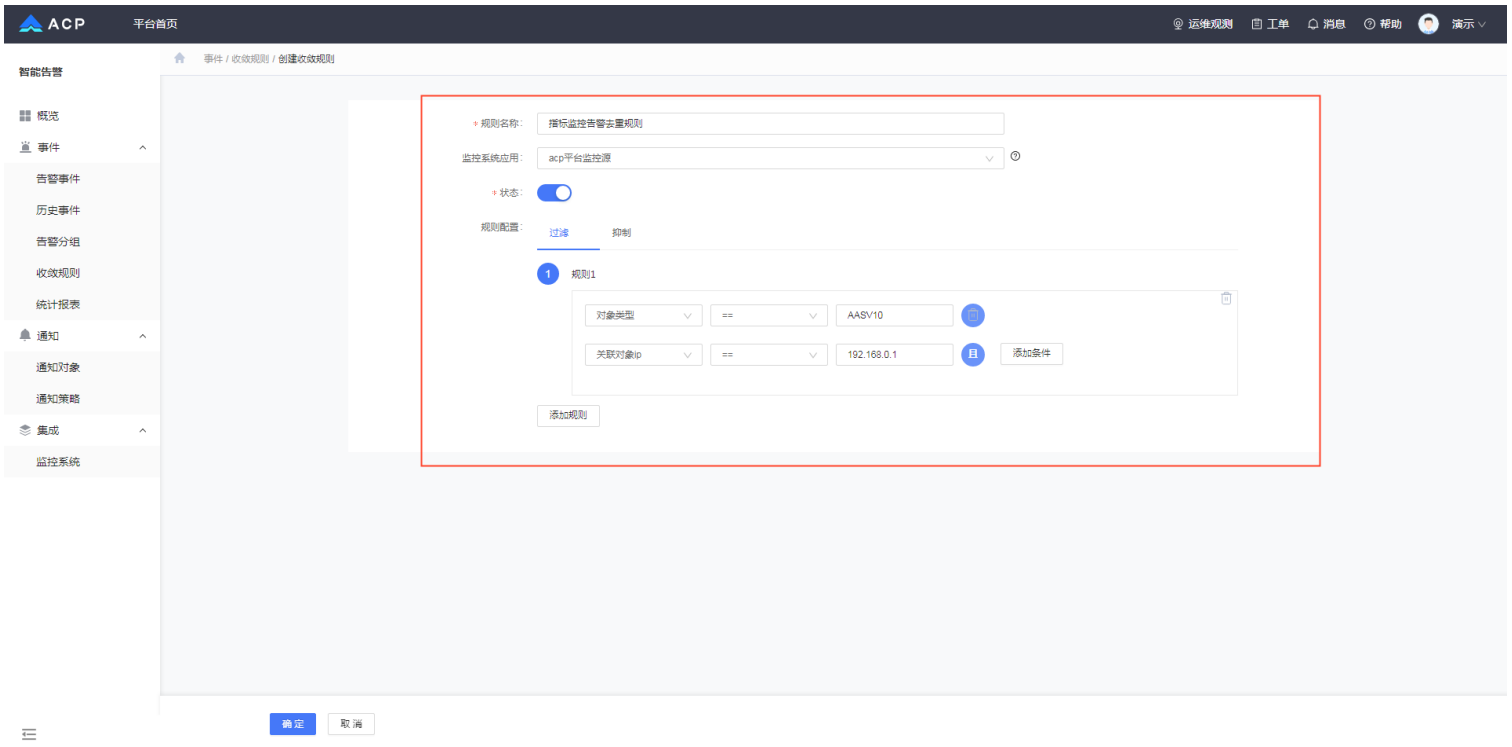
告警收敛是指通过智能化算法和策略对系统产生的海量告警信息进行过滤、聚合、关联分析与优先级排序的过程。其核心目标是解决传统监控中"告警风暴"问题，将分散、重复、低价值的告警转化为精准、可行动的故障通知，从而降低运维人员的认知负担，缩短故障定位与响应时间。

ACP支持的告警抑制机制包括：

- **告警过滤**：基于预定义规则剔除无效告警（如测试环境告警、已知良性告警）
- **降噪抑制**：对高频重复告警实施抑制策略，避免告警疲劳

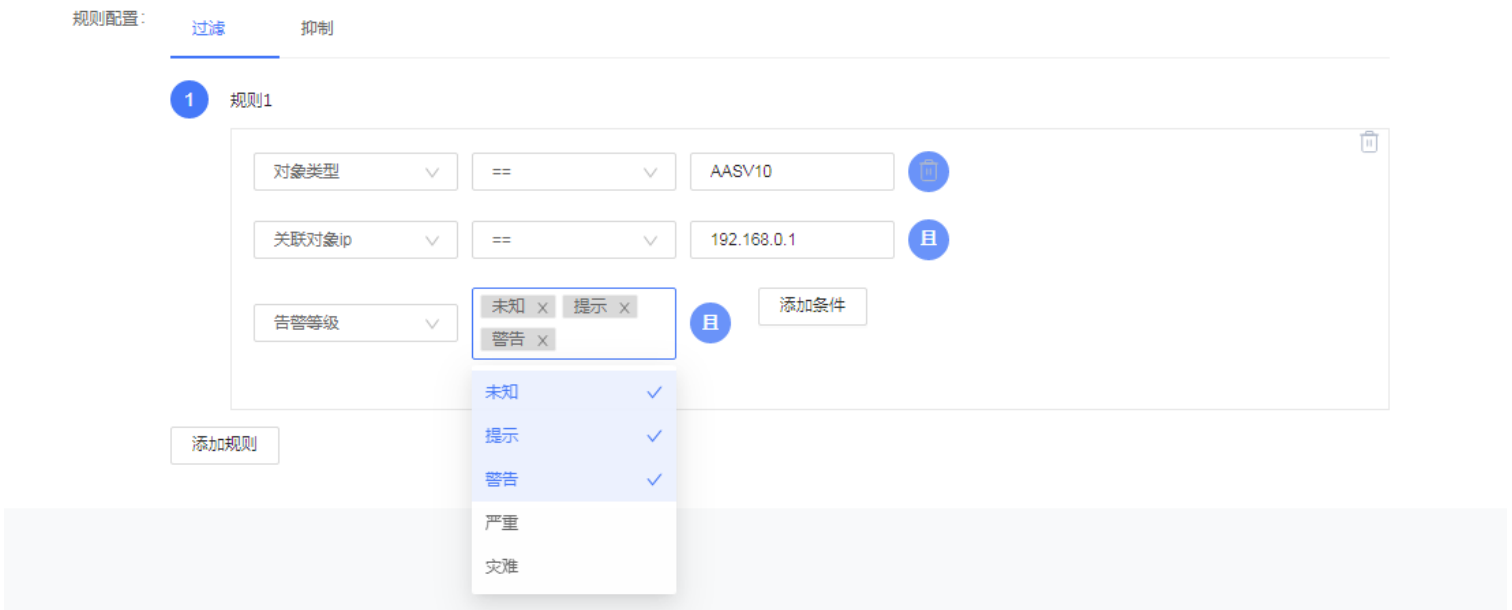
点击【事件】-【收敛规则】-【新增收敛规则】，进入创建页面，页面包含：

- 规则名称：收敛规则名称，尽量保证名称与实际收敛目的相关
- 监控系统应用：该规则作用的监控系统，目前可选择
    - acp平台监控源：即指标监控产生的告警
    - acp智能日志：即日志中心产生的告警
- 状态：启用状态，关闭则不启用
- 规则配置：支持过滤、抑制两种规则添加



这里，列举两个场景，供参考：

- 过滤：忽略部署在192.168.0.1这台主机上的AASV10所产生的严重级别以下的告警，则你可以进行如下配置



- 抑制：当某个中间件产生了高级别告警时，忽略该中间件后续产生的低级别

6.6.3.2 告警通知

告警通知是配置告警模板，以及通知方式，最终实现告警信息的发送。

- **通知模板**：用于格式化告警信息并发送给相关人员的预定义文本结构。
- **通知方式**：用于将告警信息发送给相关人员或系统的具体途径或渠道，如通过邮件、短信、云之家等。

通知模板

系统中内置了邮件，云之家的通知模板。用户可以创建通知模板，选择合适的告警信息字段，以及格式。

下面是默认通知模板的模板内容。

通知方式

通知方式支持邮件，协作（云之家），webhook, 短信等通知方式。

6.6.4 巡检

中间件巡检是指对企业IT架构中各类中间件（如应用服务器、消息队列、缓存系统、数据库连接池等）进行定期、系统性的状态检查、性能监控、日志分析及故障排查的运维管理活动。其核心目标是通过主动探测潜在风险、识别性能瓶颈、验证配置合规性，确保中间件在支撑业务应用过程中的稳定性、可靠性和高效性，从而降低系统故障发生率，保障业务连续性。

巡检内容通常包括：

- 运行状态检查**：服务进程存活、资源占用（CPU/内存/磁盘I/O）、连接数等基础指标
- 性能监控**：响应延迟、吞吐量、并发处理能力等关键性能指标
- 日志审计**：错误日志、异常堆栈、安全告警等信息的自动化采集与分析
- 配置合规性验证**：检查配置参数是否符合最佳实践及企业安全规范
- 高可用性验证**：集群状态、主备切换机制、数据同步情况等冗余能力检测

ACP的巡检基于shell脚本实现。如果你有现成的巡检脚本，那么只需要做简单的改造就可以在ACP平台上使用。

6.6.4.1 巡检规则

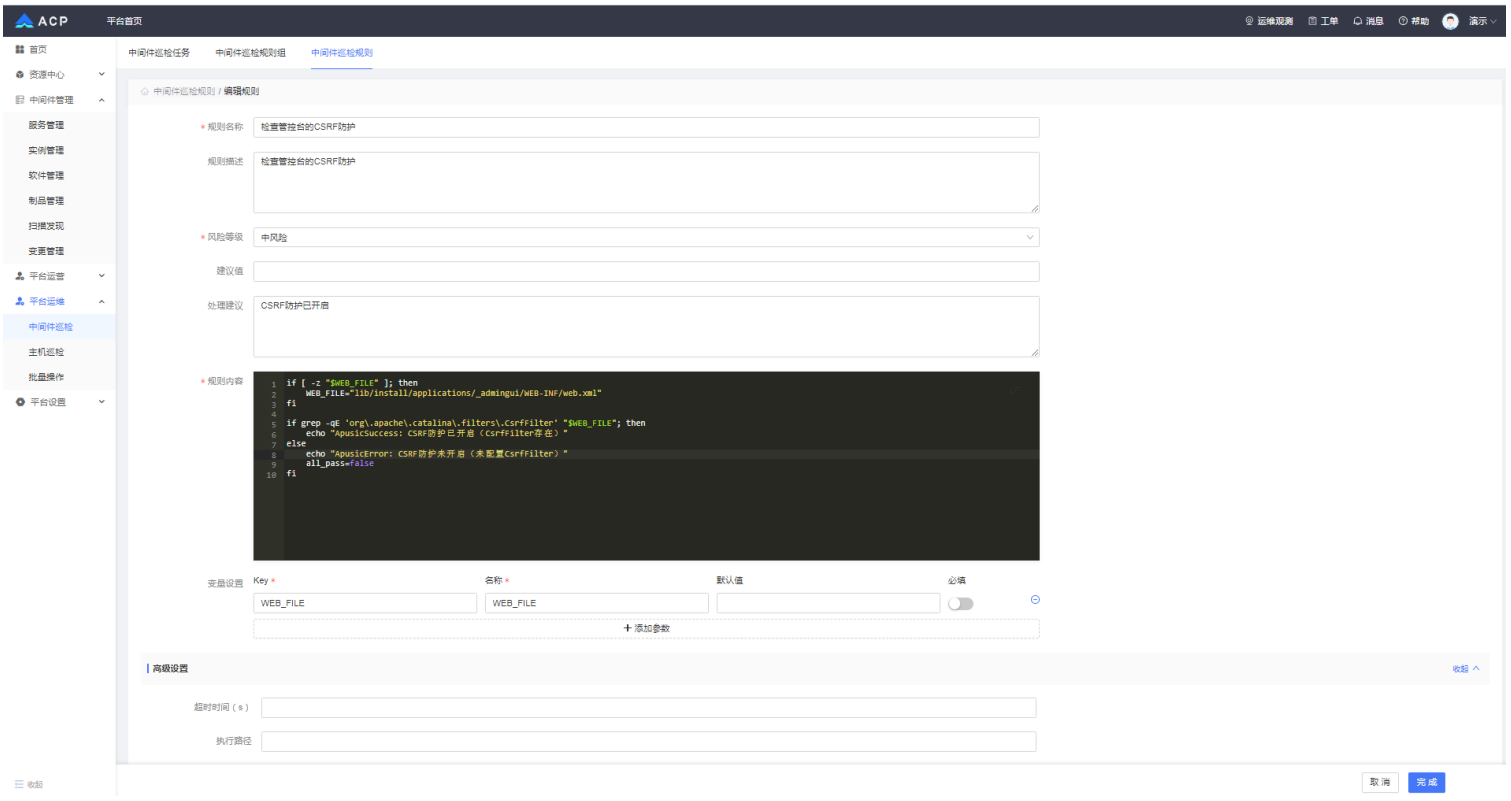
巡检规则即巡检脚本，我们建议每条规则仅做单一项的巡检。

点击【平台运维】-【中间件巡检】-【中间件巡检规则】-【创建巡检规则】可以进入规则创建页面，页面包含：

- 规则名称：规则名称，建议与实际巡检项相关联
- 规则描述：巡检项描述，建议描述清楚
- 风险等级：巡检项异常的影响等级
- 建议值：该巡检项的建议值。如检查是否配置密码，则建议值为开启
- 处理建议：当巡检出异常时的处理建议
- 规则内容：即巡检脚本，ACP要求巡检成功/失败都需要按照指定格式输出结果信息
- 变量设置：脚本中的变量声明。ACP执行巡检时会将这些变量以环境变量形式注入。请使用[$变量]名引用。

- 高级设置：设置脚本超时时间、执行路径、用户等

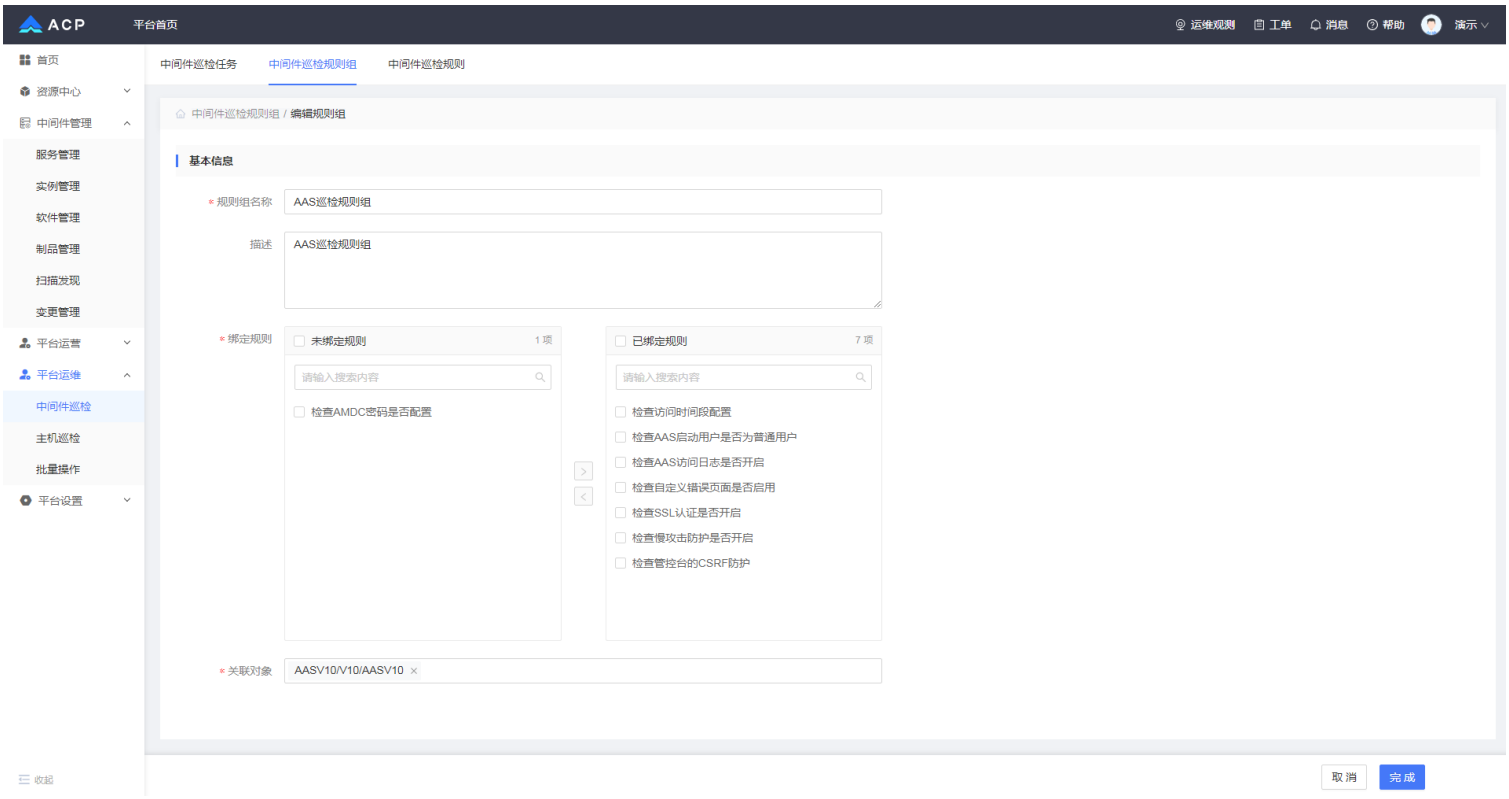
> *提示：ACP会根据巡检规则执行结果的标准输出中是否包含ApusicSuccess或者ApusicError判断巡检项是否正常，所以请务必保证以上关键字的输出*



巡检规则组

巡检规则组即完成同一目的的巡检规则的集合，如Linux网络配置检测组等。

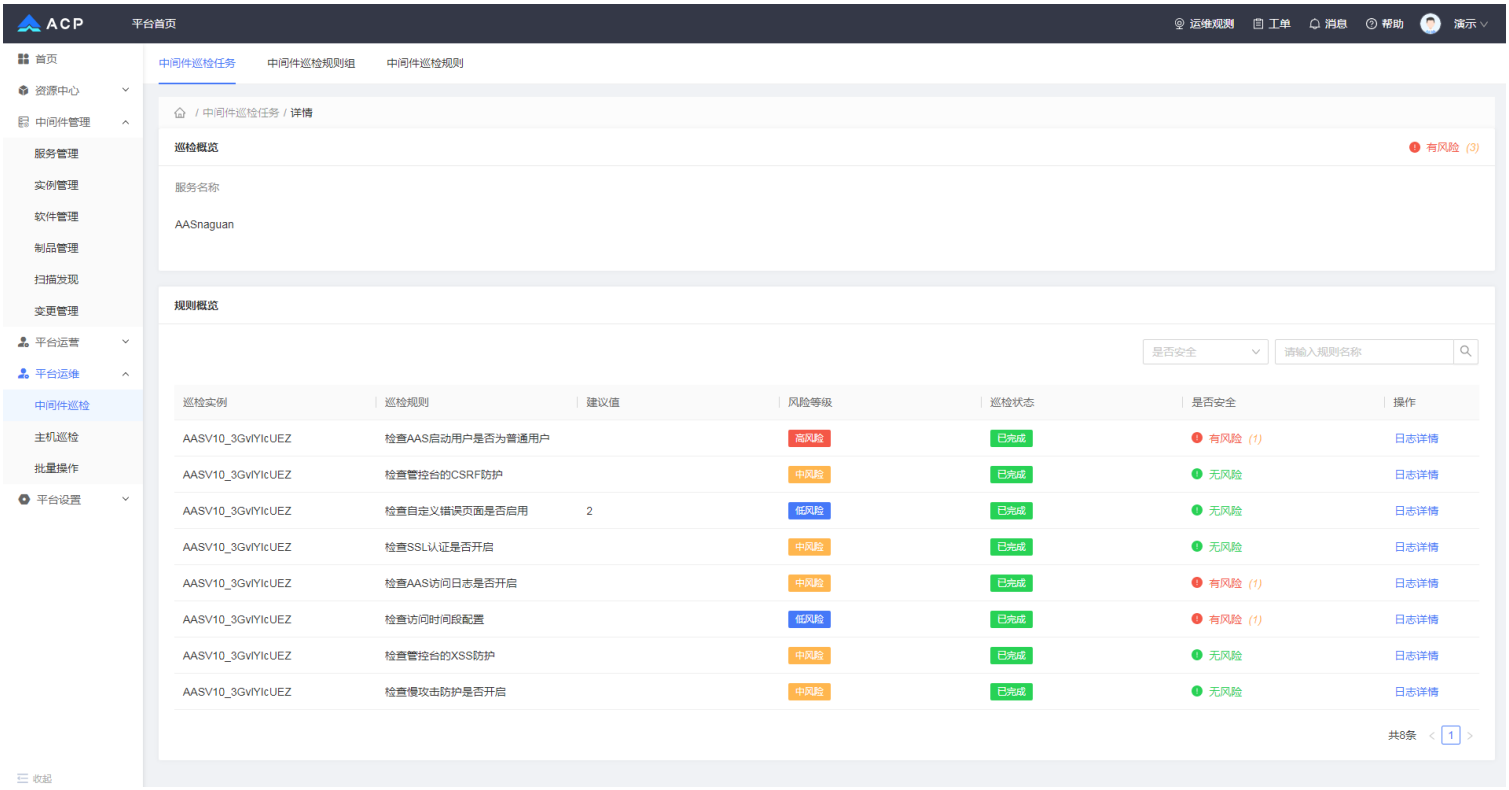
点击【平台运维】-【中间件巡检】-【中间件巡检规则】-【创建规则组】可以进入规则创建页面：



6.6.4.2 巡检任务

点击【平台运维】-【中间件巡检】-【中间件巡检任务】-【创建巡检任务】可以进入任务创建页面。

巡检任务执行完成后，即可查看本次巡检结果。



7 产品所有配置参数

7.1 中间件云平台配置说明

application-prod.properties 配置说明:

key	默认值	类型	说明	
apusic.block.type	minio	string	软件包存储类型	当
apusic.block.minio.bucket	acp	string	存储的桶名称	不
apusic.block.minio.accessKey	minioadmin	string	minio的用户名	
apusic.block.minio.secretKey	minioadmin	string	minio的密码	
apusic.block.minio.endpoint	http://localhost:9000	string	minio的访问地址	
apusic.prometheus.addr	http://localhost:9090	string	prometheus的访问地址	打
apusic.gitlab.*	\	string	acp针对gitlab的单点登录页面	这
apusic.datasource.encrypt.enabled	false	bool	数据库密码是否加密	加
apusic.datasource.database	maas	string	maas连接的数据库的库名	填
apusic.datasource.driver-class-name	org.postgresql.Driver	string	数据库驱动名称	pg or
apusic.datasource.type	postgresql	string	数据库类型	po
mybatis-plus.configuration.log-impl	""	string	数据库查询日志输出日志	这 or
apusic.datasource.data-init.enabled	\	bool	是否初始化数据	只

elasticsearch.*	\	string	连接elastic地址	连
apusic.registry.server.address	http://localhost:8761/eureka	string	服务端注册地址	
apusic.registry.server.port	8761	int	服务端注册端口	
eureka.instance.metadata-map.zone	dev	string	设置当前实例的逻辑区域（zone）	标
eureka.instance.lease-expiration-duration-in-seconds	10	int	指定实例在未发送心跳续约时	Eu
eureka.instance.lease-renewal-interval-in-seconds	3	int	客户端向 Eureka 服务器发送心跳续约的时间间隔	用
eureka.client.service-url.defaultZone	http://localhost:8761/eureka	string	连接eureka地址	连接 ht
apusic.acp.apusiclet.server	http://localhost:9840	string		修 re
apusic.acp.apusiclet.executor.ip =	localhost	string	文件导入暗转虚拟机来管理虚拟机: 见主机管理目录	本量
apusic.acp.apusiclet.agent.repositories.x86_64	\	string	x86_64下apusiclet下载地址	
apusic.acp.apusiclet.agent.repositories.aarch64	\	string	aarch64下apusiclet下载地址	
apusic.prometheus.sd.addr	http://localhost:9830	string	安装 acp 的服务器地址+9830	端

apusic.prometheus.apusiclet.ip	localhost	string	prometheus 地址	安
apusic.prometheus.deploy.dir	/home/apusic/maas-8.0.4/prometheus	string	Prometheus 部署目录	
apusic.alert.manager.add	9093	int	prometheus alert manager 地址	
apusic.console.addr	http://localhost:9801	string	前端服务的地址	
apusic.auth.center.addr	http://localhost:6866	string	授权中心的地址	
apusic.gateway.addr	http://localhost:9880	string	网关地址(可以理解当前用户访问acp管控台的地址)	H
apusic.dependency-check.path	./conf/aump/dependency-check	string	dependency-check 存放目录	

8 产品常见问题

8.1 常见产品知识问题

8.1.1 ACP上支持那些中间件?

ACP默认提供如下的中间件服务，用户可以根据平台的中间件软件接入规范标准，自定义接入三方的中间件等软件产品。

序号	软件分类	软件	版本	部署形态	环境支持
1	应用服务器	AAS	V9、V10	单机、集群	物理机、虚拟化、Kubernetes
2		Tomcat	V8.5、V9.0	单机	物理机、虚拟化
3	缓存	AMDC	V2.0	单机、主从、哨兵、集群	物理机、虚拟化、Kubernetes
4		Redis	V5.0	单机、主从	物理机、虚拟化
5	消息	ADMQ	V2.3	单机、集群	物理机、虚拟化、Kubernetes
6		Kafka	V0.10、V1.0	单机、集群	物理机、虚拟化
7		RabbitMQ	V3.8、V3.9	单机	物理机、虚拟化
8		RocketMQ	V4.9	单机	物理机、虚拟化
9	服务代理	Nginx	V1.8	单机	物理机、虚拟化
10		ALB	V2.0	标准	物理机、虚拟化、Kubernetes
11	微服务中间件	ADCC	V1.0	单机、集群	物理机、虚拟化、Kubernetes
12		Nacos	V2.1	单机	物理机、虚拟化
14		Zookeeper	V3.4	单机、集群	物理机、虚拟化
15	数据库服务	openGauss	V5.1	单机	物理机、虚拟化

16		MySQL	V5.7、V8.0	单机	物理机、虚拟化
17		PostgreSQL	V11、V12、V13、V14	单机	物理机、虚拟化

8.1.2 ACP的角色如何划分?

ACP角色划分如下:

- 平台用户
  - 平台管理员:负责访问控制管理,如用户、平台角色、平台权限
  - 平台运营人员:负责运营相关模块,如资源规划、产品管理、租户管理、租户配额、费用计费等
  - 平台运维人员:负责运维相关模块,如资源接入、软件仓库管理、配置管理、监控运维等
- 租户用户
  - 租户管理员:通过租户主页进入我的项目,在管理菜单中进行项目管理、项目配额、租户成员、角色管理
  - 租户用户:通过租户主页进入我的项目;其中租户用户在项目中又分为两种角色:
    - 项目管理员:在项目中进行中间件管理、业务管理、云原生中间件管理,通过项目配置菜单中进行项目配置
    - 项目成员:在项目中进行中间件管理、业务管理、云原生中间件管理

8.1.3 ACP的用户账号怎么注册?

联系平台管理员,由平台管理员统一创建并开通账号。

8.1.4 ACP的用户忘记密码怎么办?

普通用户忘记密码,可联系平台管理员进行密码重置。平台管理员忘记密码,则需要修改数据库,进行重置。

8.1.5 用户计算资源配额不够了,如何扩容?

目前平台支持按照服务器规格(CPU,内存容量,磁盘容量)进行配额授权,用户可在“个人中心-租户信息”页面查看总配额以及剩余配额。资源配额不足时可线下联系平台管理员申请新增计算资源额度。

8.2 常见技术问题

8.2.1 ACP有哪些部署方式?

支持单机部署和集群部署两种方式,单机部署主要用于测试和演示环境,生产环境建议使用集群部署方案。

8.2.2 部署ACP需要的服务器等软硬件资源?

软件及操作系统环境要求,如下表:

组件	要求
操作系统	Linux Red Hat 5.2或以上； 国产操作系统如银河麒麟系列、中标麒麟系列、普华、中科红旗、深度等。
CPU	Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz或以上； HUAWEI,Kunpeng 920; phytium FT1500a等。
浏览器	FireFox 70及以上、Chrome 60及以上、IE 11及以上

对应的配置要求，如下表：

部署模式	操作系统	硬件规格 (CPU/内存/硬盘)	服务器台数
单机	Linux	8核/16G/500G	1
集群	Linux	8核/16G/1T	4

8.2.3 验证图片出不来或者maas-auth报“读取默认字体包报错”

当访问登录页面，验证图片显示异常

或者当 `maas-auth` 启动时候 报错, 读取默认字体包报错时,是因为当前服务器缺少字体渲染库。

```
egy.java:94)
        at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.instantiateBean(AbstractAutow
ireCapableBeanFactory.java:1331)
        ... 29 common frames omitted
Caused by: java.lang.RuntimeException: 读取默认字体包报错
        at cloud.tianai.captcha.application.TACBuilder.build(TACBuilder.java:159)
        at com.apusic.oauth.web.controller.GenImageCaptchaApplication.<init>(GenImageCaptchaApplication.java:49)
        at java.base/jdk.internal.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
        at java.base/jdk.internal.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.jav
a:77)
        at java.base/jdk.internal.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessor
Impl.java:45)
        at java.base/java.lang.reflect.Constructor.newInstanceWithCaller(Constructor.java:499)
        at java.base/java.lang.reflect.Constructor.newInstance(Constructor.java:480)
        at org.springframework.beans.BeanUtils.instantiateClass(BeanUtils.java:195)
        ... 31 common frames omitted
Caused by: java.io.IOException: Problem reading font data.
        at java.desktop/java.awt.Font.createFont0(Font.java:1208)
        at java.desktop/java.awt.Font.createFont(Font.java:1076)
        at cloud.tianai.captcha.application.TACBuilder.build(TACBuilder.java:155)
        ... 38 common frames omitted
```

安装字体渲染库

```
$ yum install fontconfig -y
```

```
[root@linux-4-105 maas-8.0.4]# yum install fontconfig
Loaded plugins: fastestmirror
Repository base is listed more than once in the configuration
Repository extras is listed more than once in the configuration
Repository updates is listed more than once in the configuration
Loading mirror speeds from cached hostfile
base                                                                    | 3.6 kB  00:00:00
epel                                                                    | 4.3 kB  00:00:00
extras                                                                  | 2.9 kB  00:00:00
updates                                                                 | 2.9 kB  00:00:00
Resolving Dependencies
--> Running transaction check
---> Package fontconfig.x86_64 0:2.13.0-4.3.el7 will be installed
--> Processing Dependency: fontpackages-filesystem for package: fontconfig-2.13.0-4.3.el7.x86_64
--> Processing Dependency: dejavu-sans-fonts for package: fontconfig-2.13.0-4.3.el7.x86_64
--> Running transaction check
---> Package dejavu-sans-fonts.noarch 0:2.33-6.el7 will be installed
--> Processing Dependency: dejavu-fonts-common = 2.33-6.el7 for package: dejavu-sans-fonts-2.33-6.el7.noarch
---> Package fontpackages-filesystem.noarch 0:1.44-8.el7 will be installed
--> Running transaction check
---> Package dejavu-fonts-common.noarch 0:2.33-6.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================================================
 Package                       Arch              Version                    Repository          Size
================================================================================================================
Installing:
 fontconfig                    x86_64            2.13.0-4.3.el7             base               254 k
Installing for dependencies:
 dejavu-fonts-common           noarch            2.33-6.el7                 base                64 k
 dejavu-sans-fonts             noarch            2.33-6.el7                 base               1.4 M
 fontpackages-filesystem       noarch            1.44-8.el7                 base               9.9 k

Transaction Summary
================================================================================================================
Install  1 Package (+3 Dependent packages)
```

8.2.4 max file descriptions或max_map_count错误

当安装日志服务，报max file descriptions too low或者max_map_count错误，如下图，此时需要修改内核参数。

```
[apusic@linux-4-105 bin]$
ERROR: [1] bootstrap checks failed. You must address the points described in the following [1] lines before starting Elasticsearch.
bootstrap check failure [1] of [1]: max file descriptors [4096] for elasticsearch process is too low, increase to at least [65535]
ERROR: Elasticsearch did not exit normally - check the logs at /home/apusic/maas-8.0.3/service/elasticsearch/logs/elasticsearch.log
```

编辑 /etc/sysctl.conf 文件

```
vim /etc/sysctl.conf
```

在文件末尾添加以下行：

```
vm.max_map_count=262144
fs.file-max = 65536
```

保存文件并运行以下命令使配置生效：

```
sysctl -p
```

编辑用户资源限制配置文件 /etc/security/limits.conf ， 添加或修改以下内容：

```
* hard nofile 65536
* soft nofile 65536
```

> *注：上述修改需要重新登录才生效*

8.2.5 No private IP address found

当启动alertmanager时，可能会遇到No private IP address found 错误。

该问题是因为alertmanger的默认监听地址:9093导致。遇到上述问题，需要修改alertmanager启动脚本，绑定IP地址。

```
$ vi bin/script/middleware/alertmanager.sh

#在启动命令上加上，IP修改为所在机器的IP
--web.listen-address=172.20.240.228:9093 --cluster.listen-
address=172.20.240.228:9094
```

8.2.6 Could not create the Java Virtual Machine

ACP管控台依赖JDK17，当系统JAVA环境冲突时，可能会出现如下报错：

```
[apusic@linux-4-234 maas-8.0.4]$ tail -f logs/maas-manager-8.0.4.
tail: cannot open 'logs/maas-manager-8.0.4.' for reading: No such file or directory
tail: no files remaining
[apusic@linux-4-234 maas-8.0.4]$ tail -f logs/maas-manager-8.0.4.log
2026-04-15 17:29:34,831 INFO [AsyncResolver-bootstrap-executor-%d] c.n.d.s.r.a.ConfigClusterResolver [ConfigClusterResolver.java : 43] R
esolving eureka endpoints via configuration
2026-04-15 17:34:34,831 INFO [AsyncResolver-bootstrap-executor-%d] c.n.d.s.r.a.ConfigClusterResolver [ConfigClusterResolver.java : 43] R
esolving eureka endpoints via configuration
2026-04-15 17:39:34,832 INFO [AsyncResolver-bootstrap-executor-%d] c.n.d.s.r.a.ConfigClusterResolver [ConfigClusterResolver.java : 43] R
esolving eureka endpoints via configuration
2026-04-15 17:44:34,832 INFO [AsyncResolver-bootstrap-executor-%d] c.n.d.s.r.a.ConfigClusterResolver [ConfigClusterResolver.java : 43] R
esolving eureka endpoints via configuration
2026-04-15 17:49:34,833 INFO [AsyncResolver-bootstrap-executor-%d] c.n.d.s.r.a.ConfigClusterResolver [ConfigClusterResolver.java : 43] R
esolving eureka endpoints via configuration
2026-04-15 17:54:34,833 INFO [AsyncResolver-bootstrap-executor-%d] c.n.d.s.r.a.ConfigClusterResolver [ConfigClusterResolver.java : 43] R
esolving eureka endpoints via configuration
2026-04-15 17:59:34,834 INFO [AsyncResolver-bootstrap-executor-%d] c.n.d.s.r.a.ConfigClusterResolver [ConfigClusterResolver.java : 43] R
esolving eureka endpoints via configuration
Unrecognized option: --add-opens
Error: Could not create the Java Virtual Machine.
Error: A fatal exception has occurred. Program will exit.
```

可以通过指定JAVA_HOME的方式，解决此类问题：

```
# 指定JAVA_HOME为ACP内置JDK，并启动
export JAVA_HOME=/home/apusic/maas-8.0.4/jdk
./bin/maas.sh start all -d
```

8.2.7 java not found

如果中间件依赖iava环境，有时可能会遇到如下错误，造成该问题的原因有两种：

```
1  ./aas/bin/startserv: line 25: exec: java: not found
2  ./aas/bin/startserv: line 25: exec: java: not found
3  ./aas/bin/startserv: line 25: exec: java: not found
4  ./aas/bin/startserv: line 25: exec: java: not found
5  ./aas/bin/startserv: line 25: exec: java: not found
6  ./aas/bin/startserv: line 25: exec: java: not found
7  ./aas/bin/startserv: line 25: exec: java: not found
8  ./aas/bin/startserv: line 25: exec: java: not found
9  ./aas/bin/startserv: line 25: exec: java: not found
10 ./aas/bin/startserv: line 25: exec: java: not found
11 ./aas/bin/startserv: line 25: exec: java: not found
12
```

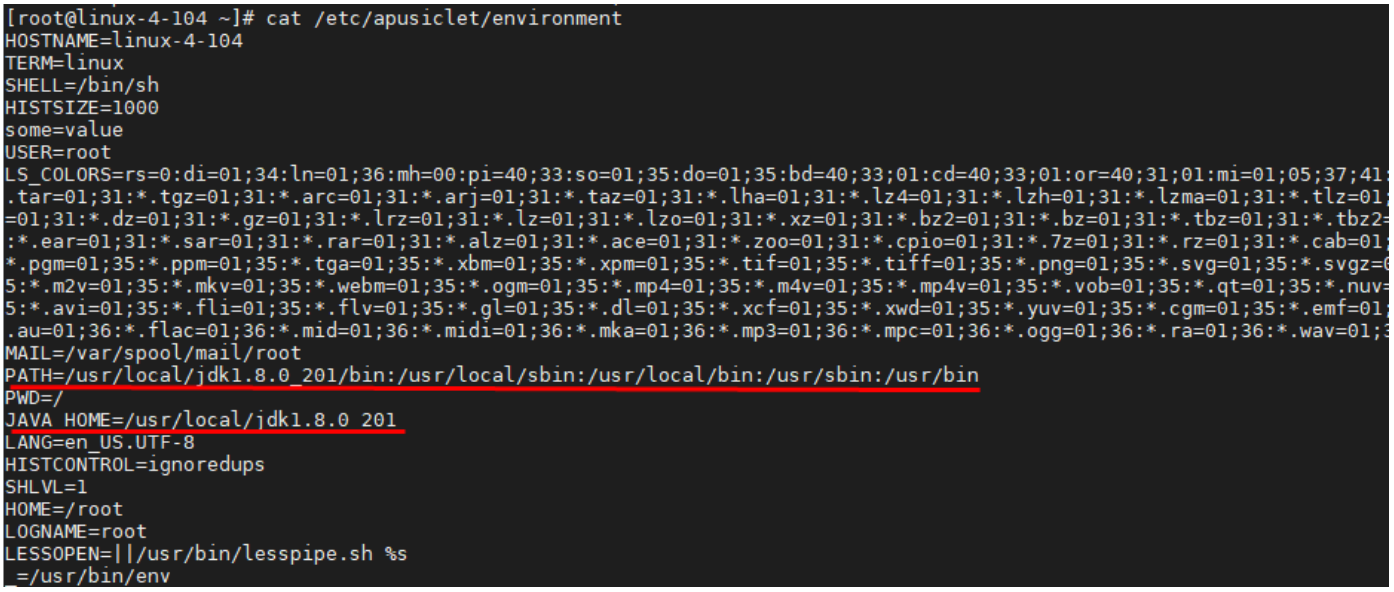
- 服务器上未安装java环境，请正确安装java环境，设置JAVA_HOME、PATH等环境变量，并重启apusiclet

```
systemctl restart apusiclet
```

- 若服务器上已经安装java，并且通过java -version能够正确获取java环境信息，那大概率是环境JAVA_HOME、PATH设置的有问题。你可以先检查apusiclet的环境变量，路径为/etc/apusiclet/environment

```
cat /etc/apusiclet/environment
```

通常情况下，JAVA_HOME以及PATH环境变量应该正确写入。若未写入或者写入错误，你需要检查系统环境变量是否设置正确，检查顺序为 /etc/profile -> /root/.bashrc。



9 附录

9.1 操作命令

9.1.1 postgresql操作命令

```
# 启动
./bin/maas.sh middleware postgresql start

# 停止
./bin/maas.sh middleware postgresql stop

# 查看状态
./bin/maas.sh middleware postgresql status

# 查看日志
./bin/maas.sh middleware postgresql logs
```

9.1.2 Minio操作命令

```
# 启动
./bin/maas.sh middleware minio start

# 停止
./bin/maas.sh middleware minio stop

# 查看状态
./bin/maas.sh middleware minio status

# 查看日志
./bin/maas.sh middleware minio logs
```

9.2 Apusiclet介绍

9.2.1 主要功能

1. 对进程操作和管理
    - apusiclet 通过对进程配合，确保在节点上启动、停止、重启等操作。
2. 对Pescod资源的管理
    - 定时同步与acp平台进程的状态，虚拟机状态。
3. 健康检查（Liveness/Readiness）：
    - apusiclet 会定期检查容器的健康状况，执行 liveness（存活）和 readiness（就绪）探针。如果容器健康检查失败，apusiclet 会重新启动容器。
4. 日志收集
    - apusiclet 会处理进程日志的收集。
5. 资源监控
    - apusiclet 会定期报告节点资源使用情况（如 CPU、内存、磁盘等）给 ACP 控制平面，以便集群管理员可以监控节点健康和负载。
6. 与 ACP 服务器通信：
    - apusiclet 定期向 ACP API 服务器报告节点和 进程 的状态，并接收新的调度指令。它负责与 API 服务器的实时通信，确保进程与集群的状态一致。

9.2.2 工作流程

9.2.2.1 操作说明

1. 获取当前虚拟机中所有的 进程对象。

   名词解释:

    1. pc(进程组): 由一个或者多个进程的集合.
    2. NAMESPACE: ACP中 租户 ID.
    3. CONTEXT: 当前实例集群的名称.
    4. READY: 进程组正常能访问的个数(正常进程个数/进程组总数).
    5. RESTART: 重启次数, 当当前进程关闭或者自动退出 `apusiclet` 会自动拉起当前被纳管的进程,重启次数并依次递增,重启次数会随着 重启次数指数递增,最大重启时间为5min.
    6. STATUS: 当前进程组的状态(INITINALIZED(初始化), CREATED(已创建),
    7. RUNNING(运行中), CHECKING(状态校验), FAILED(已失败),UPGRADE(升级中),EXISTED(进程退出),SUCCESS(成功),STOPPED(已停止))。

```
$ apusiclet pc ls

ID       NAME                                  CONTEXT     NAMESPACE
```

```
READY   RESTART STATUS  TIME
0       adcc_exporter-ayynukaq-0        adcc集群    1
1/1     0       SUCCESS 2024-11-25 15:25:09
1       adcc-omgyciiw-0                 adcc集群    1
1/1     0       SUCCESS 2024-11-25 15:25:03
```

2. 获取当前进程组资源状态及对象yaml, 按住上下选中你想要的容器组.

```
$ apusiclet pc get
Use the arrow keys to navigate: ↓ ↑ → ←  and / toggles search
select peasecod Output format. One of: yaml
😾 adcc_exporter-ayynukaq-0                 adcc集群
1
adcc-omgyciiw-0                     1


--------- Info ----------
Name:           adcc_exporter-ayynukaq-0
Context:        adcc集群
```

3. 获取当前实例的进程日志.

| 和上面的逻辑是一样的，同样选择对应的需要查看的日志。

```
$ apusiclet logs

Use the arrow keys to navigate: ↓ ↑ → ←  and / toggles search
select peasecod Output format. One of: yaml
😾 adcc_exporter-ayynukaq-0                 adcc集群
1
adcc-omgyciiw-0                     1


--------- Info ----------
Name:           adcc_exporter-ayynukaq-0
```

```
Context:        adcc集群

😸 Select:adcc_exporter-ayynukaq-0
server address is 127.0.0.1:2185
exporter_port is 9145
2024/11/25 15:25:09 info: zookeeper hosts: [127.0.0.1:2185]
2024/11/25 15:25:09 info: serving metrics at :9145/metrics
server address is 127.0.0.1:2185
exporter_port is 9145
2024/11/28 14:02:33 info: zookeeper hosts: [127.0.0.1:2185]
2024/11/28 14:02:33 info: serving metrics at :9145/metrics
server address is 127.0.0.1:2185
exporter_port is 9145
2024/12/02 14:53:50 info: zookeeper hosts: [127.0.0.1:2185]
2024/12/02 14:53:50 info: serving metrics at :9145/metrics
2024/12/02 14:53:50 fatal: shutting down exporter: listen tcp
:9145: bind: address already in use
```

4. 查看当前volumes 挂载的卷

```
$ apusiclet volume ls
ID      NAME    PATH    MOUNT   SIZE    STATUS  TIME
```

9.2.2.2 apusiclet 制品包的制作

方式一:

1. 构建apusiclet 构建 apusic 制品包

```
$ apusiclet preset build <REPOSITORY_NAME>:<VERSION>
✓ Name (软件包名称): minio
✓ Version (软件包版本): 20210422
✓ Maintainer (作者): apusic
Preset (软件包): minio-20210422-linux-amd64.tar.gz
```

```
WorkDir (工作路径): /root/minio-test
Volumes (Bind mount a volume [hostPath:containerPath],
[hostPath1:containerPath1]): ./data:./data
Environment (环境变量):
MINIO_ROOT_USER=minioadmin,MINIO_ROOT_PASSWORD=minioadmin
Command (执行命令(输入 'exit' 退出))
请输入command:: ./minio server ./data/minio
请输入command:: exit
```

方式二:

1. 初始化 `metadata.json`，初始化成功后会在当前目录生成一个 `metadata.json` 文件。

```
apusiclet preset init
```

2. 修改当前 `metadata.json`

```
{
  "metadata": {},
  "raw": {
    "name": "minio", # 当前服务得名称
    "version": "20210422", # 当前服务得版本号
    "maintainer": "apusic" # 当前服务得作者
  },
  "images": {
    "preset": "minio-20210422-linux-amd64.tar.gz" # 软件包名称，当前软件包必须在当前目录必须存在
  },
  "os": "linux", # 部署到当前操作系统 支持什么 linux windows
  "command": [
    "./minio server /root/minio-dev/data" # 启动命令
  ],
  "create_time": "0001-01-01T00:00:00Z"
}
```

3. 构建preset 中间件安装包

```
apusiclet preset build minio:v1 -f metadata.json
=> naming to minio:v1
```

5. 关于apusiclet 升级

```
# 查看当前虚拟机中的apusiclet 当前的版本
apusiclet version

APUSICLET
apusic 中间件生命周期管理
Version:        8.0.4
GitRevision:    075de3cd90e7d279657267cfb1d47a7093cc877a
GolangVersion:  go1.23.4
BuildStatus:    develop
GitTag:         v8.0.4-22-g075de3c
Platform:       linux/amd64
BuildDate:      2025-06-17-14:31:53

# 如果有最新的版本需要更新的问题的时候，一种方式是通过页面上更新apusiclet 还有一种方式通过命令行更新

apusiclet upgrade
```

6. 手动安装apusiclet

```
# 手动卸载apusiclet 软件包
systemctl stop apusiclet
yum remove -y apusiclet

# 安装apusiclet
rpm -ivh apusiclet.rpm
rm apusiclet-8.0.4-Linux-x86_64.rpm -rf
```

```
systemctl daemon-reload
systemctl start apusiclet
```

如果当前更新的时候出现问题使用当前命令查看日志

```
$ cat /var/log/messages |grep upgrade.sh


Jun 17 02:49:17 linux-3-61 upgrade.sh: --2025-06-17 14:49:17--
http://172.24.3.61/acp/resource-
manager/apis/v1alpha1/openapi/apusiclet/rpm?arch=x86_64
Jun 17 02:49:17 linux-3-61 upgrade.sh: Connecting to
172.24.3.61:80... connected.
Jun 17 02:49:17 linux-3-61 upgrade.sh: HTTP request sent, awaiting
response... 200 OK
Jun 17 02:49:17 linux-3-61 upgrade.sh: Length: unspecified
[application/octet-stream]
Jun 17 02:49:17 linux-3-61 upgrade.sh: Saving to: ‘apusiclet.rpm’
Jun 17 02:49:17 linux-3-61 upgrade.sh: 27650K .......... ......
170M=0.2s
Jun 17 02:49:17 linux-3-61 upgrade.sh: 2025-06-17 14:49:17 (172
MB/s) - ‘apusiclet.rpm’ saved [28330211]
Jun 17 02:49:18 linux-3-61 upgrade.sh: Loaded plugins:
fastestmirror, product-id, search-disabled-repos, subscription-
Jun 17 02:49:18 linux-3-61 upgrade.sh: : manager
Jun 17 02:49:18 linux-3-61 upgrade.sh: This system is not registered
with an entitlement server. You can use subscription-manager to
register.
Jun 17 02:49:18 linux-3-61 upgrade.sh: Resolving Dependencies
Jun 17 02:49:18 linux-3-61 upgrade.sh: --> Running transaction check
Jun 17 02:49:18 linux-3-61 upgrade.sh: ---> Package apusiclet.x86_64
0:8.0.4-1 will be erased
Jun 17 02:49:19 linux-3-61 upgrade.sh: --> Finished Dependency
Resolution
Jun 17 02:49:19 linux-3-61 upgrade.sh: Dependencies Resolved
Jun 17 02:49:19 linux-3-61 upgrade.sh:
```

```
================================================================================
Jun 17 02:49:19 linux-3-61 upgrade.sh: Package            Arch
Version            Repository          Size
Jun 17 02:49:19 linux-3-61 upgrade.sh:
================================================================================
Jun 17 02:49:19 linux-3-61 upgrade.sh: Removing:
Jun 17 02:49:19 linux-3-61 upgrade.sh: apusiclet          x86_64
8.0.4-1            installed           94 M
Jun 17 02:49:19 linux-3-61 upgrade.sh: Transaction Summary
Jun 17 02:49:19 linux-3-61 upgrade.sh:
================================================================================
Jun 17 02:49:19 linux-3-61 upgrade.sh: Remove  1 Package
Jun 17 02:49:19 linux-3-61 upgrade.sh: Installed size: 94 M
Jun 17 02:49:19 linux-3-61 upgrade.sh: Downloading packages:
Jun 17 02:49:19 linux-3-61 upgrade.sh: Running transaction check
Jun 17 02:49:19 linux-3-61 upgrade.sh: Running transaction test
Jun 17 02:49:19 linux-3-61 upgrade.sh: Transaction test succeeded
Jun 17 02:49:19 linux-3-61 upgrade.sh: Running transaction
Jun 17 02:49:19 linux-3-61 upgrade.sh: Warning: RPMDB altered
outside of yum.
Jun 17 02:49:20 linux-3-61 upgrade.sh: ** Found 2 pre-existing rpmdb
problem(s), 'yum check' output follows:
Jun 17 02:49:20 linux-3-61 upgrade.sh: 2:postfix-2.10.1-9.el7.x86_64
has missing requires of libmysqlclient.so.18()(64bit)
Jun 17 02:49:20 linux-3-61 upgrade.sh: 2:postfix-2.10.1-9.el7.x86_64
has missing requires of libmysqlclient.so.18(libmysqlclient_18)
(64bit)
Jun 17 02:49:20 linux-3-61 upgrade.sh: Erasing    : apusiclet-8.0.4-
1.x86_64                                   1/1
Jun 17 02:49:20 linux-3-61 upgrade.sh: Verifying  : apusiclet-8.0.4-
1.x86_64
1/1file:///tmp/ansible/repodata/repomd.xml: [Errno 14] curl#37 -
"Couldn't open file /tmp/ansible/repodata/repomd.xml"
Jun 17 02:49:20 linux-3-61 upgrade.sh: Trying other mirror.
Jun 17 02:49:20 linux-3-61 upgrade.sh:
https://nexus.apusic.com/repository/yum-local/repodata/repomd.xml:
```

```
[Errno 14] HTTPS Error 500 - Internal Server Error
Jun 17 02:49:20 linux-3-61 upgrade.sh: Trying other mirror.
Jun 17 02:49:20 linux-3-61 upgrade.sh:
https://nexus.apusic.com/repository/docker-
ce/linux/centos/7/x86_64/stable/repodata/repomd.xml: [Errno 14]
HTTPS Error 500 - Internal Server Error
Jun 17 02:49:20 linux-3-61 upgrade.sh: Trying other mirror.
Jun 17 02:49:20 linux-3-61 upgrade.sh: Removed:
Jun 17 02:49:20 linux-3-61 upgrade.sh: apusiclet.x86_64 0:8.0.4-1
Jun 17 02:49:20 linux-3-61 upgrade.sh: Complete!
Jun 17 02:49:20 linux-3-61 upgrade.sh: Preparing...
########################################
Jun 17 02:49:20 linux-3-61 upgrade.sh: Updating / installing...
Jun 17 02:49:22 linux-3-61 upgrade.sh: apusiclet-8.0.4-1
########################################
Jun 17 02:49:22 linux-3-61 systemd: Stopping /tmp/upgrade.sh...
Jun 17 02:49:22 linux-3-61 systemd: Stopped /tmp/upgrade.sh.
```

9.2.3 apusiclet 制品库管理

2. 查询制品包列表

```
$ apusiclet preset ls

REPOSITORY                      TAG     ID              CREATED
SIZE
172.24.3.61:9010/acp/minio      latest  c4d70f148d      2025-
03-11 15:50:11  17 MB
```

3. 删除本地制品包

```
$ apusiclet preset rm 172.24.3.61:9010/acp/minio:latest
remove preset down ...
```

4. 拉取仓库的制品包

```
$ apusiclet pull 172.24.3.61:9010/acp/minio:latest
Connecting to 172.24.3.61:9010 ... connected.
Length: 16690390 (17 MB) [application/x-rpm]
minio-latest.tar.gz 100% [==============================] (616
MB/s)
2025-03-11 18:15:16 - ‘minio-latest.tar.gz’ saved
[16690390/16690390]
```

5. 推送仓库的制品包

```
$ apusiclet push 172.24.3.61:9010/acp/minio:latest
Length: 16690388 (17 MB)              [application/octet-
stream]
minio-latest.tar.gz 100% [==============================] (204
MB/s)
2025-03-11 18:15:59  - ‘minio-latest.tar.gz’ pushed...
[16690388:16690388]
```

6. 推送镜像和拉取镜像的时候我们需要登录对应的仓库

```
$ apusiclet login  172.24.3.61:9010
 Username:: minioadmin
 ✓ Password:: **********
 Login Succeeded
```

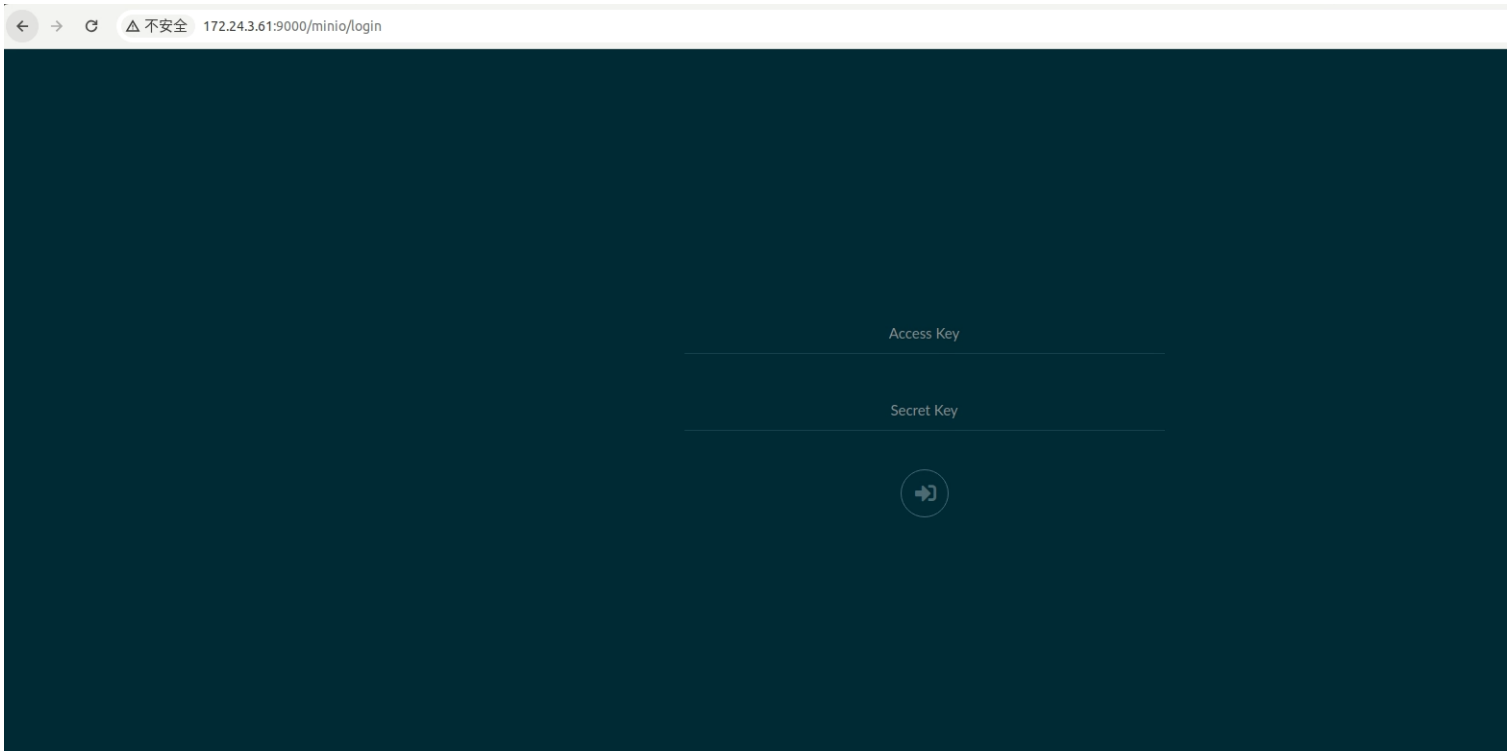
7. 运行指定的制品包服务

```
$ apusiclet preset run --name minio-demo
172.24.3.61:9010/acp/minio:latest
peasecod minio-demo created

# 查询当前服务运行状态
```

```
$ apusiclet pc ls
ID      NAME                    CONTEXT         NAMESPACE
READY   RESTART STATUS  TIME
0       minio-demo              minio-demo      default
1/1     0       SUCCESS 2025-03-11 18:20:38

# 获取当前服务的日志
# apusiclet pc logs <NAME> <NAMESPACE>
$ apusiclet pc logs minio-demo  default

Endpoint: http://172.24.3.61:9000  http://192.168.29.1:9000
http://192.168.0.1:9000  http://192.168.1.1:9000
http://192.168.18.1:9000  http://192.168.26.1:9000
http://192.168.67.1:9000  http://192.168.70.1:9000
http://192.18.0.1:9000  http://127.0.0.1:9000

Browser Access:
http://172.24.3.61:9000  http://192.168.29.1:9000
http://192.168.0.1:9000  http://192.168.1.1:9000
http://192.168.18.1:9000  http://192.168.26.1:9000
http://192.168.67.1:9000  http://192.168.70.1:9000
http://192.18.0.1:9000  http://127.0.0.1:9000

Object API (Amazon S3 compatible):
Go:         https://docs.min.io/docs/golang-client-quickstart-
guide
Java:       https://docs.min.io/docs/java-client-quickstart-
guide
Python:     https://docs.min.io/docs/python-client-quickstart-
guide
JavaScript: https://docs.min.io/docs/javascript-client-
quickstart-guide
```

```
.NET:       https://docs.min.io/docs/dotnet-client-quickstart-
guide
Detected default credentials 'minioadmin:minioadmin', please
change the credentials immediately using 'MINIO_ROOT_USER' and
'MINIO_ROOT_PASSWORD'
IAM initialization complete
```

访问当前minio是否正常运行(可以看到minio正常运行):



全国统一服务热线

4008-555-800

金蝶天燕云计算股份有限公司(简称“金蝶天燕云”)成立于2000年，前身为“金蝶中间件公司”，是金蝶集团旗下新一代软件基础云平台服务商，云计算国家标准制定企业，国家信创产业核心软件企业。金蝶天燕是国家863重点研发计划与核高基重大专项承接企业，也是“两网一站四库十二金”国家重点工程的基础平台提供商，产品广泛应用于政府、军工、金融、能源等关键行业，累计服务客户总数超过10万家。

Apusic 金蝶天燕

云计算国家标准制定企业

金蝶集团旗下基础软件企业

信息技术应用创新核心企业

官网：www.apusic.com