Menu

概览

架构

发版日志

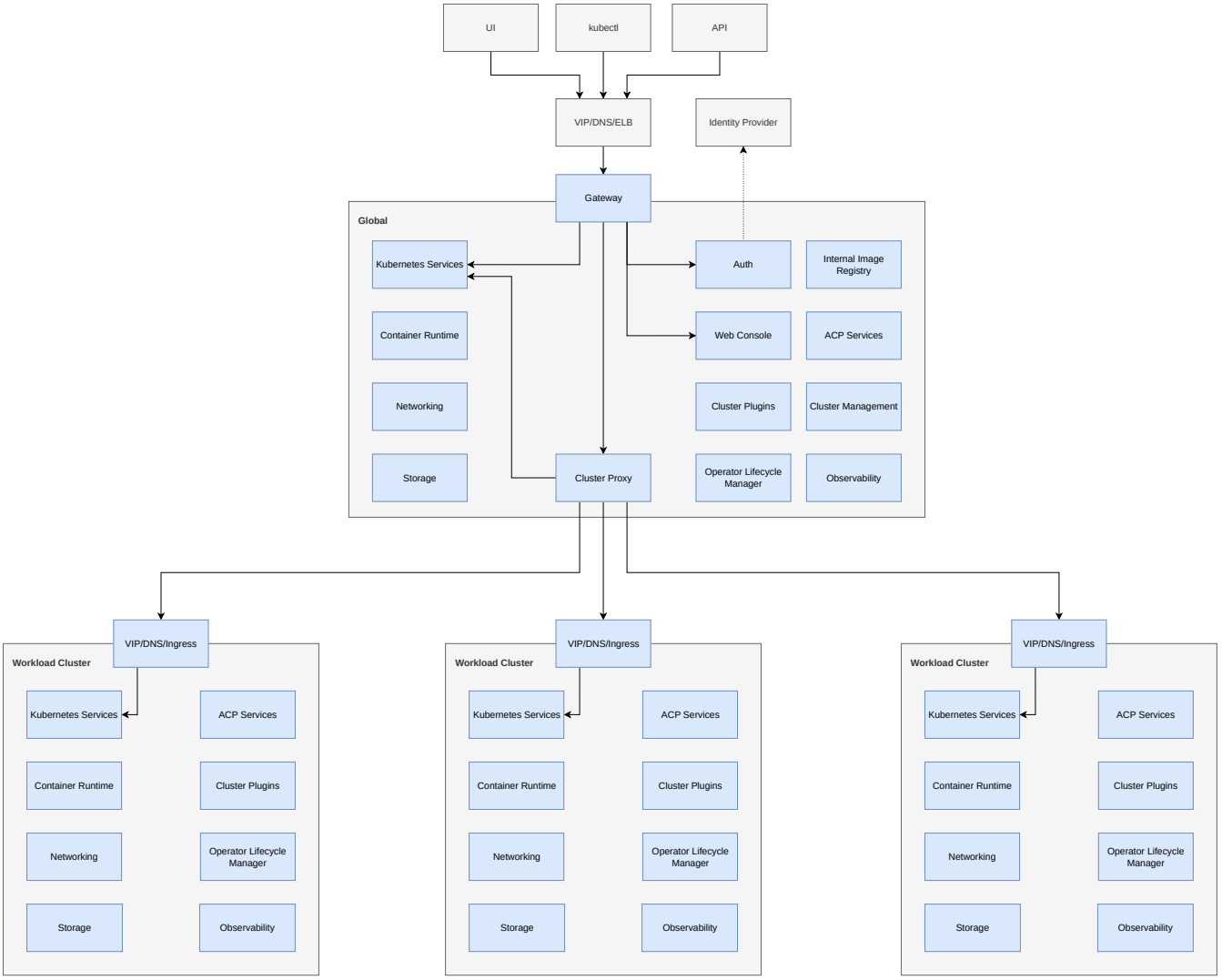
架构

目录

灵雀云容器平台 简介

灵雀云容器平台（ACP）提供了一个企业级的基于 Kubernetes 的平台，使组织能够在混合云和多云环境中一致地构建、部署和管理应用。ACP 集成了核心 Kubernetes 功能，并增强了管理、可观测性和安全服务，提供统一的控制平面和灵活的业务集群。

该架构遵循**中心辐射（hub-and-spoke）**模型，由一个 `global` 集群和多个业务集群组成。此设计既提供了集中治理，又允许独立的工作负载执行和可扩展性。



核心架构组件

Global Cluster

`global` 集群作为 ACP 的集中管理和控制中心，提供平台范围的服务，如认证、策略管理、集群生命周期操作和可观测性。它也是多集群管理的核心枢纽，支持跨集群功能。

关键组件包括：

- **Gateway**
  作为平台的主要入口，管理来自 UI、CLI（kubectl）和自动化工具的 API 请求，并将其路由到相应的后端服务。
- **Authentication and Authorization (Auth)**
  集成外部 Identity Providers (**IdPs**)，提供单点登录（SSO）和基于 RBAC 的访问控制。

- **Web Console**
  提供基于 Web 的 ACP 界面，通过 Gateway 访问平台 API。
- **Cluster Management**
  负责业务集群的注册、配置和生命周期管理。
- **ACP Services**
- **Operator Lifecycle Manager (OLM)** 和集群插件
  管理 operator 和集群扩展的安装、更新及生命周期。
- **Internal Image Registry**
  提供开箱即用的集成容器镜像仓库，支持基于角色的访问控制。
- **Observability**
  为 `global` 集群和业务集群提供集中式日志、指标和追踪。
- **Cluster Proxy**
  实现 `global` 集群与业务集群之间的安全通信。

Workload Cluster

业务集群是由 `global` 集群管理的基于 Kubernetes 的环境。每个业务集群运行隔离的应用工作负载，并继承来自中央控制平面的治理和配置。

外部集成

- **Identity Provider (IdP)**
  支持通过标准协议（OIDC、SAML）进行联合认证，实现统一的用户管理。
- **API** 和 **CLI** 访问
  用户可通过 RESTful API、Web 控制台或命令行工具（如 `kubectl` 和 `ac`）与 ACP 交互。
- 负载均衡器 **(VIP/DNS/SLB)**
  为 Gateway 及 `global` 和业务集群的入口端点提供高可用性和流量分发。

可扩展性与高可用性

ACP 设计支持水平扩展和高可用性：

- 各组件均可冗余部署，消除单点故障。

- `global` 集群支持管理数十至数百个业务集群。
- 业务集群可根据工作负载需求独立扩展。
- 通过 VIP/DNS/Ingress 实现无缝路由和故障切换。

功能视角

灵雀云容器平台（ACP）的完整功能由**ACP Core**和基于两大技术栈的扩展组成：**Operator** 和 **Cluster Plugin**。

- **ACP Core**

  ACP 的最小交付单元，提供核心能力，如集群管理、容器编排、项目和用户管理。

  - 满足最高安全标准
  - 提供最大稳定性
  - 支持最长生命周期

- 扩展

  Operator 和 Cluster Plugin 两个技术栈中的扩展可分为：

  - **Aligned** – 生命周期策略包含多个维护流，与 ACP 保持一致。
  - **Agnostic** – 生命周期策略包含多个维护流，独立于 ACP 发布。

  更多扩展详情，请参见 Extend。

技术视角

平台组件运行时
所有平台组件均作为容器运行在 Kubernetes 管理集群（即 `global` 集群）中。

高可用架构

- `global` 集群通常由至少三个控制平面节点和多个工作节点组成
- etcd 的高可用性是集群 HA 的核心；详情见*关键组件高可用机制*

- 负载均衡可由外部负载均衡器或集群内自建 VIP 提供

请求路由

- 客户端请求首先经过负载均衡器或自建 VIP
- 请求转发至运行在指定入口节点（或配置为控制平面节点）的 **ALB**（平台默认 Kubernetes Ingress Gateway）
- ALB 根据配置规则将流量路由到目标组件 Pod

副本策略

- 核心组件至少运行两个副本
- 关键组件（如 registry、MinIO、ALB）运行三个副本

容错与自愈

- 通过 kubelet、kube-controller-manager、kube-scheduler、kube-proxy、ALB 等组件协作实现
- 包括健康检查、故障切换和流量重定向

数据存储与恢复

- 控制平面配置和平台状态以 Kubernetes 资源形式存储于 etcd
- 在灾难性故障时，可通过 etcd 快照进行恢复

主备灾备

- 两个独立的 `global` 集群：主集群 和 备集群
- 灾备机制基于主集群到备集群的 etcd 数据实时同步
- 主集群故障不可用时，服务可快速切换至备集群

关键组件高可用机制

etcd

- 部署在三个（或五个）控制平面节点上
- 使用 RAFT 协议进行领导选举和数据复制

- 三节点部署可容忍最多一个节点故障；五节点部署可容忍最多两个节点故障
- 支持本地和远程 S3 快照备份

监控组件

- **Prometheus**：多实例，结合 Thanos Query 实现去重和跨地域冗余
- **VictoriaMetrics**：集群模式，包含分布式 VMStorage、VMInsert 和 VMSelect 组件

日志组件

- **Nevermore** 收集日志和审计数据
- **Kafka / Elasticsearch / Razor / Lanaya** 以分布式和多副本模式部署

网络组件（**CNI**）

- **Kube-OVN / Calico / Flannel**：通过无状态 DaemonSet 或三副本控制平面组件实现 HA

ALB

- Operator 以三副本部署，启用领导选举
- 实例级健康检查和负载均衡

自建 **VIP**

- 基于 Keepalived 的高可用虚拟 IP
- 支持心跳检测和主备切换

Harbor

- 基于 ALB 的负载均衡
- PostgreSQL 采用 Patroni 实现 HA
- Redis 采用哨兵模式
- 无状态服务多副本部署

Registry 和 **MinIO**

- Registry 以三副本部署
- MinIO 采用分布式模式，支持纠删码、数据冗余和自动恢复

发版日志

目录

已知问题

4.2.0

新功能与增强

支持 **Kubernetes 1.33**

ACP 现已支持 **Kubernetes 1.33**，带来来自 Kubernetes 社区的最新上游功能、性能提升和安全增强。

ACP CLI (ac)

全新的 **ACP CLI (ac)** 让您能够通过无缝的命令行体验，在 ACP 上开发、构建、部署和运行应用。

主要功能包括：

- 兼容 **kubectl** 的命令
- 与 **ACP** 平台环境集成的身份认证
- 跨多个环境的统一会话管理
- **ACP** 特有的扩展，支持平台访问和跨环境工作流

完整功能详情请参见： ACP CLI (ac)

托管控制平面 **(HCP)**

发布内容：

- **Alauda Container Platform Kubeadm Provider**
- **Alauda Container Platform Hosted Control Plane**
- **Alauda Container Platform SSH Infrastructure Provider**

生命周期： *Agnostic*（与 ACP 异步发布）

托管控制平面通过将每个集群的控制平面作为容器化组件托管在管理集群中，实现了控制平面与工作节点的解耦。该架构降低了资源消耗，加快了集群创建和升级速度，并为大型多集群环境提供了更好的可扩展性。

更多信息请参见： 关于托管控制平面

用户权限管理增强

我们优化了 RBAC 管理，带来以下改进以提升可用性和可维护性：

- 平台角色管理：
  - 基于 **UI** 的权限自定义已废弃： 平台角色不再支持通过 Web 控制台自定义权限配置，所有角色权限必须通过 YAML 文件配置。
  - 保持向后兼容： 现有平台预设角色及之前版本定义的角色仍然完全可用，用户可继续为用户分配这些角色并授权。
- **Kubernetes** 角色管理：
  - 原生 **Kubernetes** 角色管理： 平台控制台新增 Kubernetes Role 和 ClusterRole 资源的专用管理界面，支持直接关联 Kubernetes 角色与用户并分配权限。
  - 模块化权限定义： 平台插件资源权限将逐步迁移到独立的 Role 和 ClusterRole 资源，实现更好的隔离和更易管理。

通过 **Kyverno** 增强 **Pod** 安全策略

我们通过 Kyverno 策略引擎强化了工作负载安全能力：

- 即用型安全模板：控制台内置 8 个经过验证的安全策略模板，涵盖 Pod Security Standards 的 Privileged、Baseline、Nonroot 和 Restricted 等级别
- 一键配置：业务视图中可快速从模板创建策略，无需手写 YAML，立即在指定命名空间生效

基于 **Envoy Gateway** 的下一代 **Gateway API**

本次发布引入了基于 Envoy Gateway 的全新 Gateway API 实现，提供统一的 L7 流量入口，保持与社区 Gateway API 规范一致，并为更丰富的流量策略和生态集成奠定基础。

出口防火墙支持基于域名的规则

出口防火墙新增基于域名的允许/拒绝规则，突破仅支持 IP 地址的限制，实现对公共 SaaS 服务和 IP 地址频繁变更的外部资源的细粒度出站访问控制。

新增端点健康检查器，实现更快故障切换

引入新的端点健康检查器，能够更快检测节点崩溃、网络分区等故障，及时剔除不健康后端，大幅缩短流量故障切换时间，降低服务中断风险。

新的本地存储 **Operator**，简化 **Ceph/TopoLVM** 管理

新推出的 **Local Storage**（Alauda Build of Local Storage）Operator 大幅简化了 Ceph 和 TopoLVM 的部署与磁盘管理。部署时可列出集群内所有可用磁盘，包括型号、容量等关键属性，并选择纳管磁盘。 磁盘绑定方面，Ceph 和 TopoLVM 优先使用设备 ID 而非挂载路径，避免因节点重启或设备重新识别导致的设备名变化引发存储问题。

其他重要变更

日志插件生命周期变更

日志相关插件的生命周期状态由 *Aligned* 变更为 *Agnostic*（与 ACP 异步发布）。

受影响插件：

- **Alauda Container Platform Log Essentials**（本版本新增）
- **Alauda Container Platform Log Storage for ClickHouse**
- **Alauda Container Platform Log Storage for Elasticsearch**
- **Alauda Container Platform Log Collector**

更多信息请参见： 关于日志服务

命名空间默认安全级别提升

自 v4.2.0 起，新创建的命名空间（通过 Web 控制台或 CLI）默认 PSA 策略由 Baseline 改为 Restricted。

- Baseline：禁止已知的权限提升，提供中等安全性
- Restricted：遵循 Pod 安全最佳实践，要求最严格

WARNING

Restricted 策略对 Pod 配置要求非常严格。如果您的业务需要特权模式、以 root 用户运行、挂载 hostPath 或使用 host 网络等能力，这些工作负载将无法在默认 Restricted 策略的命名空间中运行。

影响分析：

- 仅影响新创建的命名空间
- 需要特权能力（如 root 用户、hostPath 挂载）的工作负载无法直接运行

推荐方案：

- 修改应用配置以满足 Restricted 策略的安全要求
- 必要时手动将命名空间策略设置回 Baseline

MinIO 进入维护模式

MinIO（Alauda Build of MinIO）已进入 维护模式，未来仅提供安全修复，不再新增功能。现有 MinIO 集群可继续运行，新建对象存储需求建议优先采用 Ceph Object 作为推荐方案。

Calico 进入维护模式

Calico（Alauda Container Platform Networking for Calico）CNI 插件已进入 维护模式，仅处理安全相关问题，不再作为默认推荐网络方案。现有 Calico 集群继续支持，新集群建议使用 kube-ovn 作为标准 CNI。

Ingress Nginx 切换为 Operator 管理

Ingress Nginx（Alauda Build of Ingress Nginx）已从集群插件迁移为基于 Operator 的部署和管理模式。升级后现有 Ingress 资源继续生效，后续操作预计通过 Operator 进行。尽管上游社区版本不再更新，我们将继续为该发行版提供漏洞修复和安全补丁。

弃用与移除功能

Kubernetes 版本升级策略更新

自 **ACP 4.2** 起，Kubernetes 版本升级 不再可选。集群升级时，必须与其他平台组件一同升级 Kubernetes 版本。 此变更确保集群版本一致性，减少未来维护窗口。

ALB 自 **v4.2.0** 起弃用

ALB（Alauda Container Platform Ingress Gateway）自 v4.2.0 起标记为弃用。新集群和新用户应采用基于 Envoy Gateway 的 Gateway API 作为首选方案。现有使用 ALB 的集群升级后仍可继续使用，但强烈建议规划并执行向 Gateway API 的迁移，以获得长期支持和功能演进。

Flannel 完全移除

Flannel（Alauda Container Platform Networking for Flannel）CNI 插件已从平台中完全移除。仍在使用 Flannel 的集群必须在升级到本版本或更高版本前迁移至 kube-ovn。请提前规划并完成迁移，避免因切换 CNI 导致服务中断。

修复的问题

- 之前 upmachinepool 资源的 status 字段会保存对应的 machine 资源，但未进行排序，导致在每次 reconcile 时都被判定为需要更新，从而造成审计数据过大。该问题现已修复。
- 平台集群数量较多时，若使用批量设置项目配额功能为项目设置了配额后，将无法对单个集群的项目配额进行更新，此问题已修复。
- 更新 LDAP 绑定账号的密码时，提交配置会返回网络检测错误，导致变更无法生效。此问题已经修复。
- 之前在 OperatorHub 中创建集群级别的 Instance 时，由于 web console 自动添加了 metadata.namespace 字段，导致出现 404 报错。此问题现已修复。
- 长期未登录而被系统自动禁用的用户，在管理员手动激活后会被再次自动禁用，导致激活操作无法生效。此问题已经解决。
- 之前从集群卸载 Operator 后，其状态会错误地显示为 Absent，尽管实际状态仍为 Ready。用户需要通过 violet upload 手动重新上传才能恢复。此问题现已修复，Operator 在卸载后将正确显示为 Ready。
- 使用 violet upload 上传 Operator 新版本后，偶尔会出现无法安装新版本的情况。该问题已被修复。
- 当 Operator 或 Cluster Plugin 中包含多个前端扩展时，前端扩展的左侧导航可能会出现点击无反应的问题。此前的临时解决方法是为扩展的 ConfigMap 添加注解 cpaas.io/auto-sync: "false"。该问题现已在代码层面正式修复。

- 之前当集群中存在 Display Name 为空的节点时，用户在节点详情页面打开面包屑的节点下拉筛选框，会无法通过输入内容来筛选节点。该问题现已修复。
- 日志归档结束后未删除临时文件，导致磁盘无法得到释放，此问题已修复
- 使用 violet upload 对文件夹中的多个 package 进行上传时，以前会因磁盘空间不足而失败。现已优化为自动及时清理已上传的 package，避免出现该错误。
- 修复了在项目导入命名空间过程中，修改Pod 安全策略配置不生效的问题。
- 修复了 global 集群升级而 Workload 集群未升级时，Workload 集群内 Application、Deployment 等负载监控面板无法显示的问题。
- 修复了在低于 1.30 版本的 Kubernetes 环境中进行升级时，KubeVirt Operator 可能部署失败的问题。
- 修复了通过 UI 界面创建的 Secret 仅包含 Username 和 Password，而缺少 auth 认证信息，与 kubectl create 行为不一致的问题。该问题曾导致依赖完整认证信息的构建工具（如 buildah）出现认证失败。

已知问题

- 使用 violet push 推送 chart package 时，虽然 push 显示成功，但该 package 在 public-charts 仓库中可能无法看到。
  临时解决方案： 重新 push 一次。
- 虽然安装平台的页面中提供了 global 集群的 label 和 annotation 配置项，但实际不会生效。
- 如果自定义应用包含告警资源，且该告警资源中的指标表达式使用了自定义指标，那么当将该应用导出为 Chart 或应用 YAML 后，无论是通过导入 Chart 至平台，还是直接使用应用 YAML 创建应用，只要将其部署到与原应用命名空间名称不同的环境中，都会导致应用部署失败。

  解决方法为：手动修改 Chart 或 YAML 文件中告警资源内的指标表达式，将其中的命名空间标签值改为目标部署环境的命名空间。
- ceph-mgr 创建的默认存储池 .mgr 会使用默认的 Crush Rule，在延伸集群中不能正常选出 osd，所以必须使用 CephBlockPool 创建名为 .mgr 的存储池，但是因为时序上的不确定导致 mgr 可能先于 Rook Operator 去创建 .mgr 存储池导致出现该问题。
  遇到该问题后可以尝试重启 rook-ceph-mgr 的 pod，如果不能恢复需要清理后重新部署。
- 通过 YAML 创建应用时使用 defaultMode 字段导致应用创建失败。
  操作路径：容器平台 → 应用管理 → 应用列表 → 通过 YAML 创建（Create from YAML），当提交的 YAML 文件中包含 defaultMode 字段（通常用于 ConfigMap/Secret 的卷挂载权限

配置）时，应用创建会失败并返回校验错误。
解决方案：创建应用前手动移除 YAML 中所有 defaultMode 声明。

- 当 Helm Chart 中设置了 pre-delete post-delete hook。
  执行删除模板应用，卸载 Chart 时，遇到某些原因导致 hook 执行失败，进而导致应用无法删除。需要排查原因，并优先解决 hook 执行失败的问题。