Menu

产品概览

架构

功能视角

部署视角

技术视角

发版日志

4.0.4

4.0.3

4.0.2

4.0.1

4.0.0

架构

目录

功能视角

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

- **ACP Core**

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

  - 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 两大技术栈中的扩展可分为：

  - **Aligned** —— 由多个维护流组成的生命周期策略，与 ACP 保持一致。
  - **Agnostic** —— 由多个维护流组成的生命周期策略，独立于 ACP 发布。

  有关扩展的更多详情，请参见 Extend。

部署视角

ACP 由一个 `global` 集群 和一个或多个 业务集群 组成。

- `global` 集群
  - 多集群管理的中心枢纽
  - 所有集群必须先注册到 `global` ，才能被管理
  - 承载多集群及跨集群功能
  - Kubernetes 由平台部署和管理
- 业务集群
  - 承载用户工作负载和服务
  - Kubernetes 可能由平台部署，也可能由第三方提供
  - 支持主流云厂商的 Kubernetes 服务以及 CNCF 兼容的 Kubernetes 集群
  - 某些场景下， `global` 集群也可能承载业务工作负载

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）内。

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 ingress 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至目标组件 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
- 包含健康检查、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 资源形式存储在 etcd 中
- 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
- 主集群发生故障不可用时，可快速切换至备集群提供服务

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 协议进行领导者选举和数据复制
- 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**：多实例部署，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** 负责采集日志和审计数据
- **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高可用

ALB

- Operator 以三副本部署，启用领导者选举
- 支持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 实现高可用
- Redis 采用哨兵模式
- 无状态服务以多副本部署

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

4.0.4

修复的问题

- 以前在升级集群时，会遗留 CRI（Container Runtime Interface）Pod，从而阻塞集群继续升级到 4.1。该问题已在 4.0.4 中修复。

已知问题

此次发版无相关问题。

4.0.3

修复的问题

- 修复了无法删除使用 Calico 的高可用集群 master 节点的问题。

已知问题

- 以前在升级集群时，会遗留 CRI（Container Runtime Interface）Pod，从而阻塞集群继续升级到 4.1。该问题已在 4.0.4 中修复。

- 在从 3.18.0 升级到 4.0.1 时，如果 global 集群使用内置镜像仓库且开启了 protect-secret-files 开关，执行升级脚本可能会超时报错。目前暂无可用的临时解决方案。
- 偶发情况下，Pod 可能卡在 Terminating 状态且无法被 containerd 删除。尽管 containerd 执行了删除操作，但容器仍处于伪运行状态。containerd 日志显示 "OCI runtime exec failed: exec failed: cannot exec in a stopped container: unknown"，而容器状态显示为 Running。此问题在 containerd 1.7.23 版本中极少发生（仅观察到一次），触发时只影响单个 Pod。如遇此情况，可通过重启 containerd 作为临时解决方法。这是 containerd 社区已知问题，详见 https://github.com/containerd/containerd/issues/6080。
- 将集群升级到 Kubernetes 1.31 时，集群中的所有 Pod 将会重启。这是由于 Kubernetes 1.31 中对 Pod Spec 字段的变更所导致，无法避免。更多详情，请参阅 Kubernetes 问题报告：https://github.com/kubernetes/kubernetes/issues/129385

4.0.2

修复的问题

- 修复了在托管到平台的公有云 Kubernetes 集群（如 ACK）上执行节点 Drain 操作失败并返回 404 的问题。

已知问题

- 修复了无法删除使用 Calico 的高可用集群 master 节点的问题。
- 在从 3.18.0 升级到 4.0.1 时，如果 global 集群使用内置镜像仓库且开启了 protect-secret-files 开关，执行升级脚本可能会超时报错。目前暂无可用的临时解决方案。
- 偶发情况下，Pod 可能卡在 Terminating 状态且无法被 containerd 删除。尽管 containerd 执行了删除操作，但容器仍处于伪运行状态。containerd 日志显示 "OCI runtime exec failed: exec failed: cannot exec in a stopped container: unknown"，而容器状态显示为 Running。此问题在 containerd 1.7.23 版本中极少发生（仅观察到一次），触发时只影响单个 Pod。如遇此情况，可通过重启 containerd 作为临时解决方法。这是 containerd 社区已知问题，详见 https://github.com/containerd/containerd/issues/6080。
- 将集群升级到 Kubernetes 1.31 时，集群中的所有 Pod 将会重启。这是由于 Kubernetes 1.31 中对 Pod Spec 字段的变更所导致，无法避免。更多详情，请参阅 Kubernetes 问题报告：https://github.com/kubernetes/kubernetes/issues/129385

4.0.1

修复的问题

- 在集群 api-server 压力过大的场景下，kyverno-report-controller 中的 aggregate worker 有概率无法启动，导致合规报告无法正常创建。这会阻止 PolicyReport 资源的创建，使 Web Console 无法展示违规资源信息或只能显示部分报告数据。排查方法是检查 kyverno-report-controller pod 日志中是否存在 "starting worker aggregate-report-controller/worker" 信息以验证其是否正常运行。如未正常运行，临时解决方案是手动重启 kyverno-report-controller。

已知问题

- 修复了无法删除使用 Calico 的高可用集群 master 节点的问题。
- 修复了在托管到平台的公有云 Kubernetes 集群（如 ACK）上执行节点 Drain 操作失败并返回 404 的问题。
- 在从 3.18.0 升级到 4.0.1 时，如果 global 集群使用内置镜像仓库且开启了 protect-secret-files 开关，执行升级脚本可能会超时报错。目前暂无可用的临时解决方案。
- 偶发情况下，Pod 可能卡在 Terminating 状态且无法被 containerd 删除。尽管 containerd 执行了删除操作，但容器仍处于伪运行状态。containerd 日志显示 "OCI runtime exec failed: exec failed: cannot exec in a stopped container: unknown"，而容器状态显示为 Running。此问题在 containerd 1.7.23 版本中极少发生（仅观察到一次），触发时只影响单个 Pod。如遇此情况，可通过重启 containerd 作为临时解决方法。这是 containerd 社区已知问题，详见 https://github.com/containerd/containerd/issues/6080。
- 将集群升级到 Kubernetes 1.31 时，集群中的所有 Pod 将会重启。这是由于 Kubernetes 1.31 中对 Pod Spec 字段的变更所导致，无法避免。更多详情，请参阅 Kubernetes 问题报告：https://github.com/kubernetes/kubernetes/issues/129385

4.0.0

新功能与增强

安装与升级：模块化架构

我们全面重构了平台架构，带来前所未有的灵活性、更快的更新速度以及更低的运维成本。

简化安装流程
平台现通过一个精简的核心包部署，仅包含必要组件。基础搭建完成后，客户可根据需求选择所需的 Operators 或集群插件——无论是 DevOps、Service Mesh 还是其他专业功能——并单独下载、上传和安装。

针对性补丁

- 补丁版本仅包含实际需要修复的组件。
- 未修复的组件保持不变，确保平台其他部分不受影响。
- 客户通过平台内置的标准升级机制应用补丁，而非手动更新单个组件，使维护和追踪更加简便。

智能升级

- 升级时仅替换并重启有新代码的组件。
- 未修改的组件保持现有版本和运行时间。
- 最大限度减少停机时间，缩短维护窗口，提升升级体验。

独立组件版本管理

- 大多数 Operators 遵循独立于核心平台的发布节奏。
- 新功能和修复一经准备即可上线，无需等待整个平台更新。
- 该方式加快交付速度，让客户更快享受改进成果。

集群：基于 **Cluster API** 的声明式集群生命周期管理

本地集群现利用 Kubernetes Cluster API 实现全声明式操作，包括：

- 集群创建
- 节点扩缩容与加入

该无缝集成的 Cluster API 直接融入您的 IaC 流水线，实现集群生命周期的端到端编程式控制。

Operator 与扩展：全面能力可见性

完整 **Operator** 目录

OperatorHub 现展示所有支持的 Operators，无论其包是否已上传至平台。此改进：

- 即使在隔离环境中也能完整查看平台能力
- 消除用户对可用功能的认知差距
- 降低探索平台功能时的发现阻力

版本灵活性

用户安装时可选择特定 Operator 版本，而非仅限最新版本，增强组件兼容性与升级路径的控制。

Web Console 扩展

Operators 现支持基于锚点的 Web Console 扩展，允许将功能特定的前端镜像包含在 Operators 中，并无缝集成至平台 Web Console。

集群插件增强

所有关于 Operator 可见性、版本选择及 Web Console 扩展的改进同样适用于集群插件，确保平台扩展的一致用户体验。

日志查询逻辑优化

日志查询页面优化解决了用户使用日志查询功能时遇到的体验和性能问题：

- 原有单选框替换为高级搜索组件，日志搜索体验类似于 GIT 搜索。
- 日志内容查询条件独立。
- 时间查询条件位置调整，调整时间范围时不会重置日志过滤条件。
- 优化日志查询 API，提升整体查询性能。

ElasticSearch 升级至 8.17

我们将 ElasticSearch 版本升级至 8.17，以跟进社区功能和改进。

ALB 认证

ALB 现支持多种认证机制，允许用户在 Ingress 层处理认证，而无需在每个后端应用中实现。

ALB 支持 ingress-nginx 注解

本版本新增对 ALB 中常用 ingress-nginx 注解的支持，包括 keepalive 设置、超时配置和 HTTP 重定向，提升与社区 ingress-nginx 的兼容性。

Kubevirt 实时迁移优化

实时迁移过程中，网络中断时间缩短至小于 0.5 秒，且现有 TCP 连接不会断开。
此优化显著提升了虚拟机迁移在生产环境中的稳定性和可靠性。

LDAP/OIDC 集成优化

LDAP/OIDC 集成表单字段调整，主要包括去除不必要/重复字段及优化字段描述。
LDAP/OIDC 集成现支持通过 YAML 配置，允许在 YAML 文件中进行用户属性映射。

源代码构建（**S2I**）支持

- 新增 **Alauda Container Platform Builds** operator，实现从源代码自动构建镜像。
- 支持 Java/Go/Node.js/Python 语言栈。
- 通过源代码仓库简化应用部署流程。

本地 **Registry** 解决方案

- **ACP Registry** 提供轻量级 Docker Registry，具备企业级功能。
- 开箱即用的镜像管理能力。
- 简化应用交付流程。

GitOps 模块重构

- 将 **ACP GitOps** 解耦为独立的集群插件架构。
- 升级 Argo CD 至 v2.14.x 版本。
- 增强基于 GitOps 的应用生命周期管理。

命名空间级监控

- 引入命名空间级动态监控仪表盘。
- 提供 Applications/Workloads/Pods 指标可视化。

Crossplane 集成

- 发布 **Alauda Build of Crossplane** 发行版。
- 通过 XRD 组合实现以应用为中心的资源配置。

虚拟化更新

- 升级至 KubeVirt 1.4，增强虚拟化能力。
- 优化镜像处理，加快虚拟机部署速度。
- 优化虚拟机实时迁移，支持直接从 UI 发起并显示迁移状态。
- 改进绑定网络，支持双栈（IPv4/IPv6）。
- 增加 vTPM 支持，提升虚拟机安全性。

Ceph 存储更新

- Metro-DR 通过拉伸集群实现跨可用区实时数据同步。
- Regional-DR 通过基于池的镜像增强数据保护。

TopoLVM 更新

- 新增多路径设备部署支持，提升灵活性和稳定性。

修复的问题

- 以前，Operator 上架新版本后，用户需等待 10 分钟才能安装新版本。现在等待时间已缩短至 2 分钟，使用户能更快地安装 Operator 的新版本。
- 在单节点多张卡的gpu节点上，gpu-manager 偶尔会存在，针对使用 vgpu 的应用调度不成功问题。
- 使用pgpu插件时，需要将gpu节点上的默认 runtimeclass设置为nvidia。如果不设置，可能会导致应用无法正常请求gpu资源。
- 在单张 GPU 卡上，使用 gpu-manager 无法同时创建基于 vllm、mlserver 的多个推理服务在 AI 平台上，使用 gpu-manager 创建多个推理服务时，会出现该问题；在容器平台上，使用 gpu-manager 创建多个智能应用时，不会出现该问题。
- 使用mps时，当节点资源不足时，pod会无限重启。

已知问题

- 修复了无法删除使用 Calico 的高可用集群 master 节点的问题。
- 在从 3.18.0 升级到 4.0.1 时，如果 global 集群使用内置镜像仓库且开启了 protect-secret-files 开关，执行升级脚本可能会超时报错。目前暂无可用的临时解决方案。
- 偶发情况下，Pod 可能卡在 Terminating 状态且无法被 containerd 删除。尽管 containerd 执行了删除操作，但容器仍处于伪运行状态。containerd 日志显示 "OCI runtime exec failed: exec failed: cannot exec in a stopped container: unknown"，而容器状态显示为 Running。此问题在 containerd 1.7.23 版本中极少发生（仅观察到一次），触发时只影响单个 Pod。如遇此情况，可通过重启 containerd 作为临时解决方法。这是 containerd 社区已知问题，详见 https://github.com/containerd/containerd/issues/6080。
- 将集群升级到 Kubernetes 1.31 时，集群中的所有 Pod 将会重启。这是由于 Kubernetes 1.31 中对 Pod Spec 字段的变更所导致，无法避免。更多详情，请参阅 Kubernetes 问题报告：https://github.com/kubernetes/kubernetes/issues/129385
- 在集群 api-server 压力过大的场景下，kyverno-report-controller 中的 aggregate worker 有概率无法启动，导致合规报告无法正常创建。这会阻止 PolicyReport 资源的创建，使 Web Console 无法展示违规资源信息或只能显示部分报告数据。排查方法是检查 kyverno-report-controller pod 日志中是否存在 "starting worker aggregate-report-controller/worker" 信息以验证其是否正常运行。如未正常运行，临时解决方案是手动重启 kyverno-report-controller。
- 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。

此次发版无相关问题。

- 当单容器的日志量过大时(标准输出或者文件日志)，会出现一个日志文件达到了rotate的阈值，触发了rotate，但是其中的日志内容还没有采集完毕，从而导致新老日志文件同时采集，日志顺序混乱。