# Overview

# Architecture

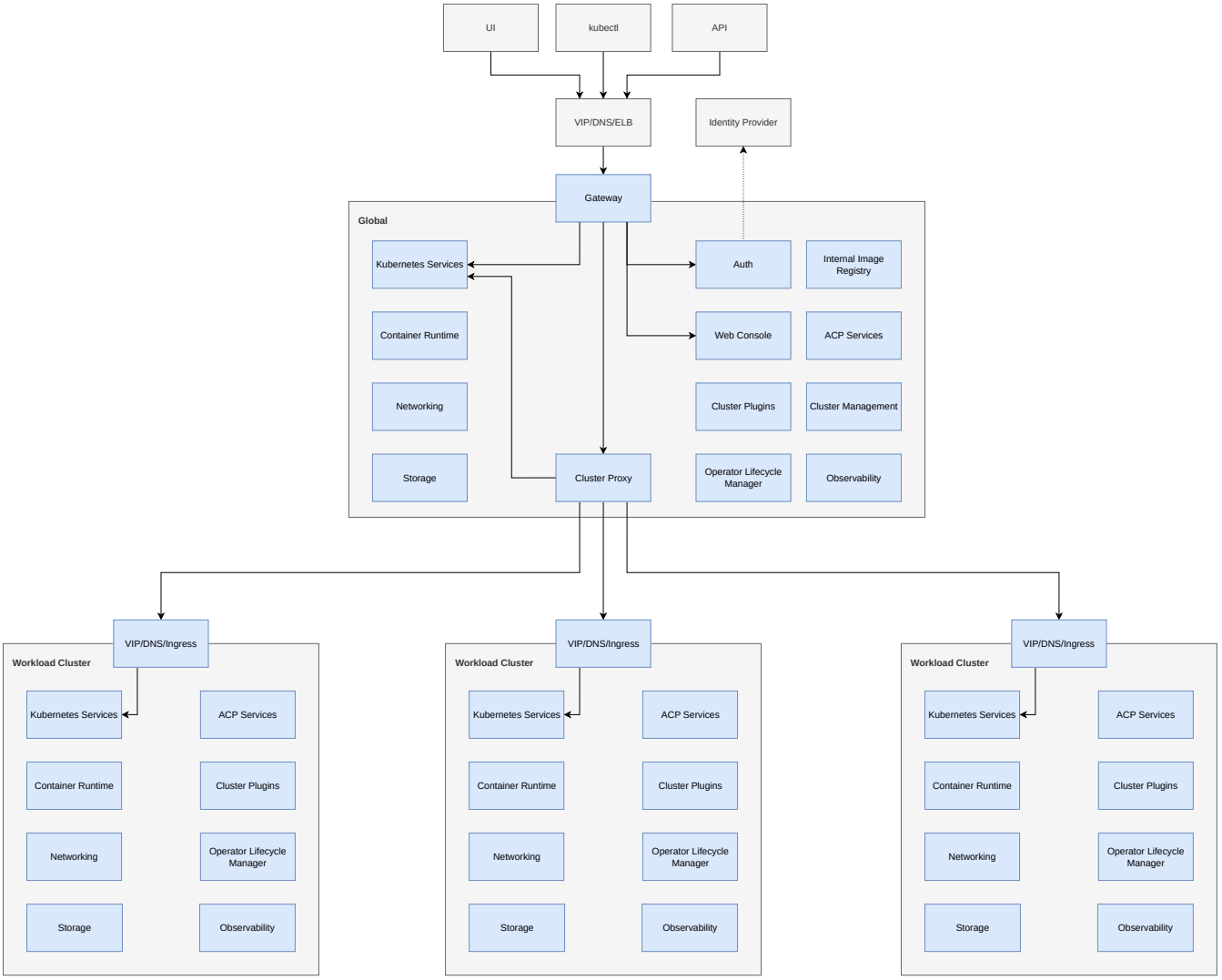
## TOC



## Introduction to Alauda Container Platform

The Alauda Container Platform (ACP) provides an enterprise-grade Kubernetes-based platform that enables organizations to build, deploy, and manage applications consistently across hybrid and multi-cloud environments. ACP integrates core Kubernetes capabilities with enhanced management, observability, and security services, offering a unified control plane and flexible workload clusters.

The architecture follows a **hub-and-spoke** model, consisting of a `global` cluster and multiple workload clusters. This design provides centralized governance while allowing independent workload execution and scalability.




# Core Architectural Components

## Global Cluster

The `global` cluster serves as the centralized management and control hub of ACP. It provides platform-wide services such as authentication, policy management, cluster lifecycle operations, and observability. It's also a central hub for multi-cluster management and provides cross-cluster functionality.

Key components include:

- **Gateway** Acts as the main entry point to the platform. It manages API requests from the UI, CLI (kubectl), and automation tools, routing them to appropriate backend services.
- **Authentication and Authorization (Auth)** Integrates with external Identity Providers (**IdPs**) to provide Single Sign-On (SSO) and RBAC-based access control.

- **Web Console** Provides a web-based interface for ACP. It interfaces with platform APIs through the gateway.
- **Cluster Management** Handles the registration, provisioning, and lifecycle management of workload clusters.
- **ACP Services**
- **Operator Lifecycle Manager (OLM) and Cluster Plugins** Manages the installation, updates, and lifecycle of operators and cluster extensions.
- **Internal Image Registry** Offers an out-of-box integrated container image repository with role-based access.
- **Observability** Provides centralized logging, metrics, and tracing for both the `global` and workload clusters.
- **Cluster Proxy** Enables secure communication between the `global` cluster and workload clusters.

## Workload Cluster

Workload clusters are Kubernetes-based environments managed by the `global` cluster. Each workload cluster runs isolated application workloads and inherits governance and configuration from the central control plane.

## External Integrations

- **Identity Provider (IdP)** Supports federated authentication via standard protocols (OIDC, SAML) for unified user management.
- **API and CLI Access** Users can interact with ACP through RESTful APIs, the web console, or command-line tools like `kubectl` and `ac`.
- **Load Balancer (VIP/DNS/SLB)** Provides high availability and traffic distribution to the Gateway and ingress endpoints of the `global` and workload Clusters.

# Scalability and High Availability

ACP is designed for horizontal scalability and high availability:

- Each component can be deployed redundantly to eliminate single points of failure.
- The `global` cluster supports managing dozens to hundreds of workload clusters.
- Workload clusters can scale independently according to workload demand.
- The use of VIP/DNS/Ingress ensures seamless routing and failover.

# Functional Perspective

Alauda Container Platform (ACP)'s complete functionality consists of **ACP Core** and extensions based on two technical stacks: **Operator** and **Cluster Plugin**.

- **ACP Core**

  The minimal deliverable unit of ACP, providing core capabilities such as cluster management, container orchestration, projects, and user administration.

  - Meets the highest security standards
  - Delivers maximum stability
  - Offers the longest support lifecycle

- **Extensions**

  Extensions in both the Operator and Cluster Plugin stacks can be classified into:

  - **Aligned** – Life cycle strategy consisting of multiple maintenance streams, with alignment to ACP.
  - **Agnostic** – Life cycle strategy consisting of multiple maintenance streams, released independently from ACP.

  For more details about extensions, see Extend.

# Technical Perspective

**Platform Component Runtime** All platform components run as containers within a Kubernetes management cluster (the `global` cluster).

### High Availability Architecture

- The `global` cluster typically consists of at least three control plane nodes and multiple worker nodes
- High availability of etcd is central to cluster HA; see *Key Component High Availability Mechanisms* for details
- Load balancing can be provided by an external load balancer or a self-built VIP inside the cluster

### Request Routing

- Client requests first pass through the load balancer or self-built VIP
- Requests are forwarded to **ALB** (the platform's default Kubernetes Ingress Gateway) running on designated ingress nodes (or control-plane nodes if configured)
- ALB routes traffic to the target component pods according to configured rules

### Replica Strategy

- Core components run with at least two replicas
- Key components (such as registry, MinIO, ALB) run with three replicas

### Fault Tolerance & Self-healing

- Achieved through cooperation between kubelet, kube-controller-manager, kube-scheduler, kube-proxy, ALB, and other components
- Includes health checks, failover, and traffic redirection

### Data Storage & Recovery

- Control-plane configuration and platform state are stored in etcd as Kubernetes resources
- In catastrophic failures, recovery can be performed from etcd snapshots

### Primary / Standby Disaster Recovery

- Two separate `global` clusters: **Primary Cluster** and **Standby Cluster**
- The disaster recovery mechanism is based on real-time synchronization of etcd data from the Primary Cluster to the Standby Cluster.

- If the Primary Cluster becomes unavailable due to a failure, services can quickly switch to the Standby Cluster.

## Key Component High Availability Mechanisms

### etcd

- Deployed on three (or five) control plane nodes
- Uses the RAFT protocol for leader election and data replication
- Three-node deployments tolerate up to one node failure; five-node deployments tolerate up to two
- Supports local and remote S3 snapshot backups

### Monitoring Components

- **Prometheus**: Multiple instances, deduplication with Thanos Query, and cross-region redundancy
- **VictoriaMetrics**: Cluster mode with distributed VMStorage, VMInsert, and VMSelect components

### Logging Components

- **Nevermore** collects logs and audit data
- **Kafka / Elasticsearch / Razor / Lanaya** are deployed in distributed and multi-replica modes

### Networking Components (CNI)

- **Kube-OVN / Calico / Flannel**: Achieve HA via stateless DaemonSets or triple-replica control plane components

### ALB

- Operator deployed with three replicas, leader election enabled
- Instance-level health checks and load balancing

### Self-built VIP

- High-availability virtual IP based on Keepalived
- Supports heartbeat detection and active-standby failover

### Harbor

- ALB-based load balancing
- PostgreSQL with Patroni HA
- Redis Sentinel mode
- Stateless services deployed in multiple replicas

### Registry and MinIO

- Registry deployed with three replicas
- MinIO in distributed mode with erasure coding, data redundancy, and automatic recovery

# Kubernetes Support Matrix

This document provides the Kubernetes version support matrix for ACP. This information is critical when creating clusters, upgrading ACP, and managing third-party clusters.

## TOC



## Overview

ACP supports multiple Kubernetes versions across different ACP releases. Understanding the supported versions is essential for:

- **Creating clusters** – Determine which Kubernetes versions can be used when provisioning new clusters
- **Upgrading ACP** – Ensure all workload clusters meet compatibility requirements before upgrading the global cluster
- **Managing third-party clusters** – Verify that public cloud or CNCF-compliant Kubernetes clusters are within the supported version range

## Version Support Matrix

The following table shows the Kubernetes version support for each ACP release.

> **INFO**
>
> The table lists ACP minor versions and does not distinguish between patch versions. Patch versions only include bug fixes and security updates, so the Kubernetes minor versions remain consistent across all patch versions within the same minor release.
>
> **Starting from ACP 4.1**, each ACP release supports only **one Kubernetes version** for cluster creation. This ensures consistency and simplifies the upgrade path for new clusters.

| ACP Version | Supported for Cluster Creation | Compatible Versions |
|---|---|---|
| ACP 4.2 | 1.33 | 1.33, 1.32, 1.31, 1.30 |
| ACP 4.1 | 1.32 | 1.32, 1.31, 1.30, 1.29 |
| ACP 4.0 | 1.31, 1.30, 1.29, 1.28 | 1.31, 1.30, 1.29, 1.28 |

# Upgrade Requirements

For ACP 4.2 and earlier, **all** workload clusters must be upgraded to the **latest** Kubernetes version in the compatible versions list **before** upgrading the ACP global cluster.

In future releases, workload clusters will only need to be within the compatible versions range to upgrade the ACP global cluster.

# Release Notes

## TOC

---

# 4.0.8

## Fixed Issues

- When a user is automatically disabled by the system due to long-term lack of login, it will be automatically disabled again after being manually activated by the administrator. This

issue has been fixed.

- Fix ovn-central can not auto recover if one ovn-db file is deleted manually.
- Fix ovn-central race with two leaders that interrupts cluster network.

## Known Issues

- In certain cases, users may find that the operations automatically recorded by the platform in a ResourcePatch resource do not match the actual modifications they made to the component. As a result, when the ResourcePatch controller applies the patch, the component may undergo unexpected changes.
  Workaround: Users should manually modify the ResourcePatch resource to ensure it reflects the intended changes.
- Previously, if a cluster contained nodes with an empty Display Name, users were unable to filter nodes by typing in the node selector dropdown on the node details page. This issue has been fixed in ACP 4.2.0.
- The temporary files were not deleted after log archiving, preventing disk space from being reclaimed. This issue has been fixed.
- When upgrading from 3.18.0 to 4.0.1, running the upgrade script may fail with a timeout if the global cluster uses the built-in image registry with the protect-secret-files feature enabled. This issue has been fixed in ACP 4.1.0.
- When upgrading from 3.18.0 to 4.0.1, running the upgrade script may fail with a timeout if the global cluster uses the built-in image registry with the protect-secret-files feature enabled. This issue has been fixed in ACP 4.1.0.
- Occasionally, a pod may become stuck in the Terminating state and cannot be deleted by containerd. Although containerd attempts the deletion operation, the container remains in a pseudo-running state. The containerd logs show OCI "runtime exec failed: exec failed: cannot exec in a stopped container: unknown" while the container status appears as Running. This issue occurs very rarely in containerd 1.7.23 (observed only once) and affects only individual pods when triggered. If encountered, restart containerd as a temporary workaround. This is a known issue in the containerd community, tracked at https://github.com/containerd/containerd/issues/6080.
- When the amount of logs in a single container is too large (standard output or file logs), it can happen that a log file reaches the rotate threshold and triggers a rotate, but the

contents of the logs in it have not been captured yet, which results in the simultaneous capture of the old and new log files, and a chaotic log order.

- When a StatefulSet's Pod is stopped and then restarted, the platform takes the earliest runtime of the Pod's daily operation as the start time and the latest runtime as the end time, ignoring any intermediate periods when it was not running. This results in the Pod being metered for more time than its actual operational hours.
- When upgrading clusters to Kubernetes 1.31, all pods in the cluster will restart. This behavior is caused by changes to the Pod spec fields in Kubernetes 1.31 and cannot be avoided. For more details, please refer to the Kubernetes issue: https://github.com/kubernetes/kubernetes/issues/129385
- Application creation failure triggered by the defaultMode field in YAML.
  Affected Path: Alauda Container Platform → Application Management → Application List → Create from YAML. Submitting YAML containing the defaultMode field (typically used for ConfigMap/Secret volume mount permissions) triggers validation errors and causes deployment failure.
  Workaround: Manually remove all defaultMode declarations before application creation.
- The default pool .mgr created by ceph-mgr uses the default Crush Rule, which may fail to properly select OSDs in a stretched cluster. To resolve this, the .mgr pool must be created using CephBlockPool. However, due to timing uncertainties, ceph-mgr might attempt to create the .mgr pool before the Rook Operator completes its setup, leading to conflicts.
  If encountering this issue, restart the rook-ceph-mgr Pod to trigger reinitialization.
  If unresolved, manually clean up the conflicting .mgr pool and redeploy the cluster to ensure proper creation order.
- When pre-delete post-delete hook is set in helm chart.
  When the delete template application is executed and the chart is uninstalled, the hook execution fails for some reasons, thus the application cannot be deleted. It is necessary to investigate the cause and give priority to solving the problem of hook execution failure.

# 4.0.7

## Fixed Issues

- Before this update, the Tekton Pipeline component had a Kubernetes STIG security vulnerability, where secrets were exposed through environment variables in the tekton-hub-

api deployment, violating security best practices. With this update, the secret mounting logic in environment variables has been completely removed to ensure that the tekton-hub-api deployment no longer exposes any credentials, complying with Kubernetes STIG security requirements.

- Before this update, the tekton-results-retention-policy-agent container in Tekton Results included sensitive information in environment variables, posing a security risk of exposing credentials in plaintext during container operations and logging scenarios. With this update, sensitive information has been properly secured and removed from environment variables to prevent credential leakage, ensuring that the retention-policy-agent container no longer contains plaintext passwords or tokens in its configuration, thereby enhancing the overall security posture of the Tekton Results system.
- Before this update, the PostgreSQL container in tekton-results-postgres-0 contained environment variables with sensitive information such as PASSWORD, password, TOKEN, and token, which posed a security risk when these credentials were exposed in plain text. With this update, the sensitive environment variables have been properly secured and no longer contain plain text passwords or tokens, ensuring that sensitive credentials are handled securely and not exposed in container environment variables.
- Before this update, the environment variables of the tekton-results-api container contained sensitive information, posing security risks when these credentials were exposed in plain text. With this update, sensitive environment variables have been properly protected, and passwords and token information are no longer exposed in plain text, enhancing the security of the tekton-results-api component.
- Before this update, the pipeline interface experienced multiple display issues including text display problems, poor user experience with variable completion multi-line functionality, and unstable behavior when updating triggers where parameters and workspace would sometimes appear and sometimes disappear, requiring users to reselect the pipeline to make them appear (including the Pipeline list). With this update, these display issues have been resolved. The pipeline and pipelinerun pages now display correctly with improved text rendering, enhanced variable completion multi-line functionality for better user experience, and stable trigger update behavior where parameters and workspace consistently appear without requiring pipeline reselection.
- Previously, the status field of an upmachinepool resource stored the associated machine resources without any ordering. This caused the resource to be updated on every reconcile loop, resulting in excessively large audit logs. This issue has now been fixed.

- Fixed an issue where modifying the Pod Security Policy when importing a namespace into a project did not take effect.
- Fixed an issue in the console where updating a Deployment in a specific sequence could cause the container lifecycle configuration to be unintentionally removed.

## Known Issues

- Previously, if a cluster contained nodes with an empty Display Name, users were unable to filter nodes by typing in the node selector dropdown on the node details page. This issue has been fixed in ACP 4.2.0.
- The temporary files were not deleted after log archiving, preventing disk space from being reclaimed. This issue has been fixed.
- When upgrading from 3.18.0 to 4.0.1, running the upgrade script may fail with a timeout if the global cluster uses the built-in image registry with the protect-secret-files feature enabled. This issue has been fixed in ACP 4.1.0.
- When upgrading from 3.18.0 to 4.0.1, running the upgrade script may fail with a timeout if the global cluster uses the built-in image registry with the protect-secret-files feature enabled. This issue has been fixed in ACP 4.1.0.
- Occasionally, a pod may become stuck in the Terminating state and cannot be deleted by containerd. Although containerd attempts the deletion operation, the container remains in a pseudo-running state. The containerd logs show OCI "runtime exec failed: exec failed: cannot exec in a stopped container: unknown" while the container status appears as Running. This issue occurs very rarely in containerd 1.7.23 (observed only once) and affects only individual pods when triggered. If encountered, restart containerd as a temporary workaround. This is a known issue in the containerd community, tracked at https://github.com/containerd/containerd/issues/6080.
- When the amount of logs in a single container is too large (standard output or file logs), it can happen that a log file reaches the rotate threshold and triggers a rotate, but the contents of the logs in it have not been captured yet, which results in the simultaneous capture of the old and new log files, and a chaotic log order.
- When a StatefulSet's Pod is stopped and then restarted, the platform takes the earliest runtime of the Pod's daily operation as the start time and the latest runtime as the end time, ignoring any intermediate periods when it was not running. This results in the Pod being metered for more time than its actual operational hours.

- When upgrading clusters to Kubernetes 1.31, all pods in the cluster will restart. This behavior is caused by changes to the Pod spec fields in Kubernetes 1.31 and cannot be avoided. For more details, please refer to the Kubernetes issue: https://github.com/kubernetes/kubernetes/issues/129385
- Fix ovn-central can not auto recover if one ovn-db file is deleted manually.
- Fix ovn-central race with two leaders that interrupts cluster network.
- Application creation failure triggered by the defaultMode field in YAML.
  Affected Path: Alauda Container Platform → Application Management → Application List → Create from YAML. Submitting YAML containing the defaultMode field (typically used for ConfigMap/Secret volume mount permissions) triggers validation errors and causes deployment failure.
  Workaround: Manually remove all defaultMode declarations before application creation.
- The default pool .mgr created by ceph-mgr uses the default Crush Rule, which may fail to properly select OSDs in a stretched cluster. To resolve this, the .mgr pool must be created using CephBlockPool. However, due to timing uncertainties, ceph-mgr might attempt to create the .mgr pool before the Rook Operator completes its setup, leading to conflicts.
  If encountering this issue, restart the rook-ceph-mgr Pod to trigger reinitialization.
  If unresolved, manually clean up the conflicting .mgr pool and redeploy the cluster to ensure proper creation order.
- When pre-delete post-delete hook is set in helm chart.
  When the delete template application is executed and the chart is uninstalled, the hook execution fails for some reasons, thus the application cannot be deleted. It is necessary to investigate the cause and give priority to solving the problem of hook execution failure.

# 4.0.6

## Fixed Issues

- Fixed an issue where, after upgrading the platform from v3.x to v4.x, if a business cluster was not upgraded, the metrics created in its new custom monitoring dashboard could not be used by HPA.

## Known Issues

- Previously, the status field of an upmachinepool resource stored the associated machine resources without any ordering. This caused the resource to be updated on every reconcile loop, resulting in excessively large audit logs. This issue has now been fixed.
- Previously, if a cluster contained nodes with an empty Display Name, users were unable to filter nodes by typing in the node selector dropdown on the node details page. This issue has been fixed in ACP 4.2.0.
- The temporary files were not deleted after log archiving, preventing disk space from being reclaimed. This issue has been fixed.
- When upgrading from 3.18.0 to 4.0.1, running the upgrade script may fail with a timeout if the global cluster uses the built-in image registry with the protect-secret-files feature enabled. This issue has been fixed in ACP 4.1.0.
- When upgrading from 3.18.0 to 4.0.1, running the upgrade script may fail with a timeout if the global cluster uses the built-in image registry with the protect-secret-files feature enabled. This issue has been fixed in ACP 4.1.0.
- Occasionally, a pod may become stuck in the Terminating state and cannot be deleted by containerd. Although containerd attempts the deletion operation, the container remains in a pseudo-running state. The containerd logs show OCI "runtime exec failed: exec failed: cannot exec in a stopped container: unknown" while the container status appears as Running. This issue occurs very rarely in containerd 1.7.23 (observed only once) and affects only individual pods when triggered. If encountered, restart containerd as a temporary workaround. This is a known issue in the containerd community, tracked at https://github.com/containerd/containerd/issues/6080.
- When the amount of logs in a single container is too large (standard output or file logs), it can happen that a log file reaches the rotate threshold and triggers a rotate, but the contents of the logs in it have not been captured yet, which results in the simultaneous capture of the old and new log files, and a chaotic log order.
- When a StatefulSet's Pod is stopped and then restarted, the platform takes the earliest runtime of the Pod's daily operation as the start time and the latest runtime as the end time, ignoring any intermediate periods when it was not running. This results in the Pod being metered for more time than its actual operational hours.
- When upgrading clusters to Kubernetes 1.31, all pods in the cluster will restart. This behavior is caused by changes to the Pod spec fields in Kubernetes 1.31 and cannot be avoided. For more details, please refer to the Kubernetes issue: https://github.com/kubernetes/kubernetes/issues/129385

- Fix ovn-central can not auto recover if one ovn-db file is deleted manually.
- Fix ovn-central race with two leaders that interrupts cluster network.
- Application creation failure triggered by the defaultMode field in YAML.
  Affected Path: Alauda Container Platform → Application Management → Application List → Create from YAML. Submitting YAML containing the defaultMode field (typically used for ConfigMap/Secret volume mount permissions) triggers validation errors and causes deployment failure.
  Workaround: Manually remove all defaultMode declarations before application creation.
- Fixed an issue in the console where updating a Deployment in a specific sequence could cause the container lifecycle configuration to be unintentionally removed.
- The default pool .mgr created by ceph-mgr uses the default Crush Rule, which may fail to properly select OSDs in a stretched cluster. To resolve this, the .mgr pool must be created using CephBlockPool. However, due to timing uncertainties, ceph-mgr might attempt to create the .mgr pool before the Rook Operator completes its setup, leading to conflicts.
  If encountering this issue, restart the rook-ceph-mgr Pod to trigger reinitialization.
  If unresolved, manually clean up the conflicting .mgr pool and redeploy the cluster to ensure proper creation order.
- When pre-delete post-delete hook is set in helm chart.
  When the delete template application is executed and the chart is uninstalled, the hook execution fails for some reasons, thus the application cannot be deleted. It is necessary to investigate the cause and give priority to solving the problem of hook execution failure.

# 4.0.5

## Fixed Issues

- Previously, after uninstalling an Operator, the Operator status was incorrectly displayed as Absent, even though the Operator was actually Ready. Users had to manually re-upload the Operator using violet upload. This issue has now been resolved, and the Operator correctly appears as Ready after uninstallation.
- In some cases, installing a new Operator version after uploading it via violet upload would fail unexpectedly. This intermittent issue has been fixed.

- When an Operator or Cluster Plugin included multiple frontend extensions, the left-side navigation of these extensions could become unresponsive. The temporary workaround required users to add the annotation cpaas.io/auto-sync: "false" to the extension’s ConfigMap. This behavior has now been permanently fixed in the code.

## Known Issues

- When upgrading a Redis Sentinel instance from v5 to v7, occasional brain split incidents may occur, potentially leading to data loss.
  Solution: Back up the Redis instance data before performing a cross-version upgrade.
- When cluster network anomalies occur, failure to update the primary node label of a PostgreSQL instance may result in abnormal instance status, potentially causing partial new connection failures.
- Before this update, the pipeline interface experienced multiple display issues including text display problems, poor user experience with variable completion multi-line functionality, and unstable behavior when updating triggers where parameters and workspace would sometimes appear and sometimes disappear, requiring users to reselect the pipeline to make them appear (including the Pipeline list). With this update, these display issues have been resolved. The pipeline and pipelinerun pages now display correctly with improved text rendering, enhanced variable completion multi-line functionality for better user experience, and stable trigger update behavior where parameters and workspace consistently appear without requiring pipeline reselection.
- Previously, the status field of an upmachinepool resource stored the associated machine resources without any ordering. This caused the resource to be updated on every reconcile loop, resulting in excessively large audit logs. This issue has now been fixed.
- Previously, if a cluster contained nodes with an empty Display Name, users were unable to filter nodes by typing in the node selector dropdown on the node details page. This issue has been fixed in ACP 4.2.0.
- The temporary files were not deleted after log archiving, preventing disk space from being reclaimed. This issue has been fixed.
- When upgrading from 3.18.0 to 4.0.1, running the upgrade script may fail with a timeout if the global cluster uses the built-in image registry with the protect-secret-files feature enabled. This issue has been fixed in ACP 4.1.0.

- When upgrading from 3.18.0 to 4.0.1, running the upgrade script may fail with a timeout if the global cluster uses the built-in image registry with the protect-secret-files feature enabled. This issue has been fixed in ACP 4.1.0.
- Occasionally, a pod may become stuck in the Terminating state and cannot be deleted by containerd. Although containerd attempts the deletion operation, the container remains in a pseudo-running state. The containerd logs show OCI "runtime exec failed: exec failed: cannot exec in a stopped container: unknown" while the container status appears as Running. This issue occurs very rarely in containerd 1.7.23 (observed only once) and affects only individual pods when triggered. If encountered, restart containerd as a temporary workaround. This is a known issue in the containerd community, tracked at https://github.com/containerd/containerd/issues/6080.
- When the amount of logs in a single container is too large (standard output or file logs), it can happen that a log file reaches the rotate threshold and triggers a rotate, but the contents of the logs in it have not been captured yet, which results in the simultaneous capture of the old and new log files, and a chaotic log order.
- When a StatefulSet's Pod is stopped and then restarted, the platform takes the earliest runtime of the Pod's daily operation as the start time and the latest runtime as the end time, ignoring any intermediate periods when it was not running. This results in the Pod being metered for more time than its actual operational hours.
- When upgrading clusters to Kubernetes 1.31, all pods in the cluster will restart. This behavior is caused by changes to the Pod spec fields in Kubernetes 1.31 and cannot be avoided. For more details, please refer to the Kubernetes issue: https://github.com/kubernetes/kubernetes/issues/129385
- Fix ovn-central can not auto recover if one ovn-db file is deleted manually.
- Fix ovn-central race with two leaders that interrupts cluster network.
- Fixed an issue where, after upgrading the platform from v3.x to v4.x, if a business cluster was not upgraded, the metrics created in its new custom monitoring dashboard could not be used by HPA.
- Application creation failure triggered by the defaultMode field in YAML.
  Affected Path: Alauda Container Platform → Application Management → Application List → Create from YAML. Submitting YAML containing the defaultMode field (typically used for ConfigMap/Secret volume mount permissions) triggers validation errors and causes deployment failure.
  Workaround: Manually remove all defaultMode declarations before application creation.

- Fixed an issue in the console where updating a Deployment in a specific sequence could cause the container lifecycle configuration to be unintentionally removed.
- The default pool .mgr created by ceph-mgr uses the default Crush Rule, which may fail to properly select OSDs in a stretched cluster. To resolve this, the .mgr pool must be created using CephBlockPool. However, due to timing uncertainties, ceph-mgr might attempt to create the .mgr pool before the Rook Operator completes its setup, leading to conflicts.
  If encountering this issue, restart the rook-ceph-mgr Pod to trigger reinitialization.
  If unresolved, manually clean up the conflicting .mgr pool and redeploy the cluster to ensure proper creation order.
- When pre-delete post-delete hook is set in helm chart.
  When the delete template application is executed and the chart is uninstalled, the hook execution fails for some reasons, thus the application cannot be deleted. It is necessary to investigate the cause and give priority to solving the problem of hook execution failure.

# 4.0.4

## Fixed Issues

- Previously, upgrading the cluster would leave behind CRI (Container Runtime Interface) Pods, which blocked further upgrades to version 4.1. This issue has been fixed in version 4.0.4.

## Known Issues

No issues in this release.

# 4.0.3

## Fixed Issues

- Fixed an issue where master nodes in HA clusters using Calico could not be deleted.

## Known Issues

- Previously, upgrading the cluster would leave behind CRI (Container Runtime Interface) Pods, which blocked further upgrades to version 4.1. This issue has been fixed in version 4.0.4.
- When upgrading from 3.18.0 to 4.0.1, running the upgrade script may fail with a timeout if the global cluster uses the built-in image registry with the protect-secret-files feature enabled. This issue has been fixed in ACP 4.1.0.
- When upgrading from 3.18.0 to 4.0.1, running the upgrade script may fail with a timeout if the global cluster uses the built-in image registry with the protect-secret-files feature enabled. This issue has been fixed in ACP 4.1.0.
- Occasionally, a pod may become stuck in the Terminating state and cannot be deleted by containerd. Although containerd attempts the deletion operation, the container remains in a pseudo-running state. The containerd logs show OCI "runtime exec failed: exec failed: cannot exec in a stopped container: unknown" while the container status appears as Running. This issue occurs very rarely in containerd 1.7.23 (observed only once) and affects only individual pods when triggered. If encountered, restart containerd as a temporary workaround. This is a known issue in the containerd community, tracked at https://github.com/containerd/containerd/issues/6080.
- When upgrading clusters to Kubernetes 1.31, all pods in the cluster will restart. This behavior is caused by changes to the Pod spec fields in Kubernetes 1.31 and cannot be avoided. For more details, please refer to the Kubernetes issue: https://github.com/kubernetes/kubernetes/issues/129385

# 4.0.2

## Fixed Issues

- Fixed an issue where performing a node drain on a public cloud Kubernetes cluster (such as ACK) managed by the platform failed with a 404 error.

## Known Issues

- Fixed an issue where master nodes in HA clusters using Calico could not be deleted.
- When upgrading from 3.18.0 to 4.0.1, running the upgrade script may fail with a timeout if the global cluster uses the built-in image registry with the protect-secret-files feature enabled. This issue has been fixed in ACP 4.1.0.
- When upgrading from 3.18.0 to 4.0.1, running the upgrade script may fail with a timeout if the global cluster uses the built-in image registry with the protect-secret-files feature enabled. This issue has been fixed in ACP 4.1.0.
- Occasionally, a pod may become stuck in the Terminating state and cannot be deleted by containerd. Although containerd attempts the deletion operation, the container remains in a pseudo-running state. The containerd logs show OCI "runtime exec failed: exec failed: cannot exec in a stopped container: unknown" while the container status appears as Running. This issue occurs very rarely in containerd 1.7.23 (observed only once) and affects only individual pods when triggered. If encountered, restart containerd as a temporary workaround. This is a known issue in the containerd community, tracked at https://github.com/containerd/containerd/issues/6080.
- When upgrading clusters to Kubernetes 1.31, all pods in the cluster will restart. This behavior is caused by changes to the Pod spec fields in Kubernetes 1.31 and cannot be avoided. For more details, please refer to the Kubernetes issue: https://github.com/kubernetes/kubernetes/issues/129385

# 4.0.1

## Fixed Issues

- Under high api-server pressure, the aggregate worker in kyverno-report-controller may occasionally fail to start, preventing proper creation of compliance reports. This results in PolicyReport resources not being created, causing the Web Console to either display no compliance violation information or only partial report data. To troubleshoot, check the kyverno-report-controller pod logs for the presence of "starting worker aggregate-report-controller/worker" messages to verify proper operation. If the worker is not running, manually restart the kyverno-report-controller as a temporary solution.

## Known Issues

- Fixed an issue where master nodes in HA clusters using Calico could not be deleted.
- Fixed an issue where performing a node drain on a public cloud Kubernetes cluster (such as ACK) managed by the platform failed with a 404 error.
- When upgrading from 3.18.0 to 4.0.1, running the upgrade script may fail with a timeout if the global cluster uses the built-in image registry with the protect-secret-files feature enabled. This issue has been fixed in ACP 4.1.0.
- When upgrading from 3.18.0 to 4.0.1, running the upgrade script may fail with a timeout if the global cluster uses the built-in image registry with the protect-secret-files feature enabled. This issue has been fixed in ACP 4.1.0.
- Occasionally, a pod may become stuck in the Terminating state and cannot be deleted by containerd. Although containerd attempts the deletion operation, the container remains in a pseudo-running state. The containerd logs show OCI "runtime exec failed: exec failed: cannot exec in a stopped container: unknown" while the container status appears as Running. This issue occurs very rarely in containerd 1.7.23 (observed only once) and affects only individual pods when triggered. If encountered, restart containerd as a temporary workaround. This is a known issue in the containerd community, tracked at https://github.com/containerd/containerd/issues/6080.
- When upgrading clusters to Kubernetes 1.31, all pods in the cluster will restart. This behavior is caused by changes to the Pod spec fields in Kubernetes 1.31 and cannot be avoided. For more details, please refer to the Kubernetes issue: https://github.com/kubernetes/kubernetes/issues/129385

# 4.0.0

## Features and Enhancements

### Installation and Upgrade: Modular Architecture

We've completely redesigned our platform's architecture to provide unprecedented flexibility, faster updates, and reduced operational overhead.

**Streamlined Installation** Our platform is now deployed via a lean core package containing only the essential components. Once the foundation is in place, customers can pick and

choose exactly which Operators or cluster plugins they need—whether DevOps, Service Mesh, or other specialized features—and download, upload, and install them individually.

**Targeted Patches**

- Patch releases include only those components that actually require bug fixes.
- Components without fixes remain exactly as they are, ensuring the rest of the platform stays untouched.
- Customers apply patches through the platform's built-in, standardized upgrade mechanism—rather than manually updating individual components—making maintenance and tracking far more straightforward.

**Intelligent Upgrades**

- During an upgrade, only components with new code are replaced and restarted.
- Unmodified components retain their existing versions and uptime.
- This minimizes downtime and shortens the maintenance window for a smoother upgrade experience.

**Independent Component Versioning**

- Most Operators follow their own release schedules, separate from the core platform.
- New features and fixes go live as soon as they're ready—no need to wait for a full-platform update.
- This approach accelerates delivery and lets customers benefit from improvements faster.

## Clusters: Declarative Cluster Lifecycle Management with Cluster API

On-premises clusters now leverage the Kubernetes Cluster API for fully declarative operations, including:

- Cluster creation
- Node scaling and joining

This seamless Cluster API integration fits directly into your IaC pipelines, enabling end-to-end, programmatic control over your cluster lifecycle.

## Operator & Extension: Comprehensive Capability Visibility

### Complete Operator Catalog

The OperatorHub now displays all supported Operators regardless of whether their packages have been uploaded to the platform. This enhancement:

- Provides full visibility into platform capabilities even in air-gapped environments
- Eliminates information gaps between what's available and what's known to users
- Reduces discovery friction when exploring platform capabilities

### Version Flexibility

Users can now select specific Operator versions during installation rather than being limited to only the latest version, providing greater control over component compatibility and upgrade paths.

### Web Console Extensions

Operators now support anchor-based Web Console extensions, allowing functionality-specific frontend images to be included within Operators and seamlessly integrated into the platform's Web Console.

### Cluster Plugin Enhancements

All improvements to Operator visibility, version selection, and Web Console extension capabilities also apply to cluster plugins, ensuring consistent user experience across all platform extensions.

## Log query logic optimization

The log query page has been optimized to solve the experience and performance problems users encounter when using the log query function:

- The original radio box has been replaced with the advanced search component. Now you can use the log search as you use the GIT search.
- Independent query conditions for log content
- The location of the time query criteria has been adjusted. Now you will not reset your log filter criteria when you adjust the time range.

- Optimized the log query API to improve the overall query performance

## ElasticSearch upgrade to 8.17

We upgraded the version of ElasticSearch to 8.17 to follow up the functions and improvements of the community.

## ALB authentication

ALB now support various authentication mechanism, which allows user to handle authentication at Ingress level instead of implementing it in each backend application.

## ALB supports ingress-nginx annotations

This release adds support for common ingress-nginx annotations in ALB, including keepalive settings, timeout configurations, and HTTP redirects, enhancing compatibility with the community ingress-nginx.

## Kubevirt live migration optimization

During the live migration process, the network interruption time has been reduced to less than 0.5 seconds, and existing TCP connections will not be disconnected. This optimization significantly improves the stability and reliability of virtual machine migrations in production environments.

## LDAP/OIDC integration optimization

The LDAP/OIDC integration form fields have been adjusted, mainly including removal of unnecessary/duplicate fields and optimization of field descriptions. LDAP/OIDC integration now supports configuration through YAML, allowing user attribute mapping within the YAML file.

## Source to Image (S2I) Support

- Added **Alauda Container Platform Builds** operator for automated image building from source code
- Supports Java/Go/Node.js/Python language stacks
- Streamlines application deployment via source code repositories

## On-prem Registry Solution

- **ACP Registry** delivered lightweight Docker Registry with enterprise-ready features
- Provides out-of-the-box image management capabilities
- Simplifies application delivery

## GitOps Module Refactoring

- Decoupled **ACP GitOps** into standalone cluster plugin architecture
- Upgraded Argo CD to v2.14.x version
- Enhanced GitOps-based application lifecycle management

## Namespace-level Monitoring

- Introduced dynamic monitoring dashboards at namespace level
- Provides Applications/Workloads/Pods metrics visualization

## Crossplane Integration

- Released **Alauda Build of Crossplane** distribution
- Implements app-centric provisioning via XRD compositions

## Virtualization Updates

- Upgraded to KubeVirt 1.4 for enhanced virtualization capabilities
- Optimized image handling for faster VM provisioning
- Optimized VM live migration, now initiable directly from the UI with visible migration status
- Improved binding networking with dual-stack (IPv4/IPv6) support
- Added vTPM support to enhance VM security

## Ceph Storage Updates

- Metro-DR with stretch cluster enables real-time data synchronization across availability zones
- Regional-DR with pool-based mirroring enhances data protection

### TopoLVM Updates

- Added support for multipath device deployment, improving flexibility and stability

## Fixed Issues

- Previously, after publishing a new Operator version, users had to wait 10 minutes before installing it. This waiting period has been reduced to 2 minutes, allowing faster installation of new Operator versions.
- On gpu nodes with multiple cards on a single node, gpu-manager occasionally exists, with unsuccessful scheduling issues for applications using vgpu.
- When using the pgpu plugin, you need to set the default runtimeclass on the gpu node to nvidia. if you don't, it may cause the application to not be able to request gpu resources properly.
- On a single GPU card, gpu-manager cannot create multiple inference services based on vllm, mlserver at the same time.
  On AI platforms, this issue occurs when gpu-manager is used to create multiple inference services; on container platforms, this issue does not occur when gpu-manager is used to create multiple smart applications.
- With mps, pods restart indefinitely when nodes are low on resources.

## Known Issues

- Fixed an issue where master nodes in HA clusters using Calico could not be deleted.
- When upgrading from 3.18.0 to 4.0.1, running the upgrade script may fail with a timeout if the global cluster uses the built-in image registry with the protect-secret-files feature enabled. This issue has been fixed in ACP 4.1.0.
- Occasionally, a pod may become stuck in the Terminating state and cannot be deleted by containerd. Although containerd attempts the deletion operation, the container remains in a pseudo-running state. The containerd logs show OCI "runtime exec failed: exec failed: cannot exec in a stopped container: unknown" while the container status appears as Running. This issue occurs very rarely in containerd 1.7.23 (observed only once) and affects only individual pods when triggered. If encountered, restart containerd as a

temporary workaround. This is a known issue in the containerd community, tracked at https://github.com/containerd/containerd/issues/6080.

- When upgrading clusters to Kubernetes 1.31, all pods in the cluster will restart. This behavior is caused by changes to the Pod spec fields in Kubernetes 1.31 and cannot be avoided. For more details, please refer to the Kubernetes issue: https://github.com/kubernetes/kubernetes/issues/129385
- Under high api-server pressure, the aggregate worker in kyverno-report-controller may occasionally fail to start, preventing proper creation of compliance reports. This results in PolicyReport resources not being created, causing the Web Console to either display no compliance violation information or only partial report data. To troubleshoot, check the kyverno-report-controller pod logs for the presence of "starting worker aggregate-report-controller/worker" messages to verify proper operation. If the worker is not running, manually restart the kyverno-report-controller as a temporary solution.
- Fixed an inconsistency where Secrets created through the web console only stored the Username and Password, and lacked the complete authentication field (auth) compared to those created via kubectl create. This issue previously caused authentication failures for build tools (e.g., buildah) that rely on the complete auth data.
- The default pool .mgr created by ceph-mgr uses the default Crush Rule, which may fail to properly select OSDs in a stretched cluster. To resolve this, the .mgr pool must be created using CephBlockPool. However, due to timing uncertainties, ceph-mgr might attempt to create the .mgr pool before the Rook Operator completes its setup, leading to conflicts.
  If encountering this issue, restart the rook-ceph-mgr Pod to trigger reinitialization.
  If unresolved, manually clean up the conflicting .mgr pool and redeploy the cluster to ensure proper creation order.

No issues in this release.

- The temporary files were not deleted after log archiving, preventing disk space from being reclaimed. This issue has been fixed.
- When the amount of logs in a single container is too large (standard output or file logs), it can happen that a log file reaches the rotate threshold and triggers a rotate, but the contents of the logs in it have not been captured yet, which results in the simultaneous capture of the old and new log files, and a chaotic log order.