Menu

Обзор

Архитектура

Примечания к выпуску

Архитектура

Содержание

Функциональная перспектива

Alauda Container Platform (ACP) включает в себя полный набор функций, состоящий из **ACP Core** и расширений, основанных на двух технических стеках: **Operator** и **Cluster Plugin**.

- **ACP Core**

  Минимальная поставляемая единица ACP , обеспечивающая основные возможности, такие как управление кластерами, оркестрация контейнеров, проекты и администрирование пользователей.

  - Соответствует самым высоким стандартам безопасности
  - Обеспечивает максимальную стабильность
  - Предлагает самый длительный жизненный цикл поддержки

- **Расширения**

  Расширения в стеках Operator и Cluster Plugin можно классифицировать следующим образом:

- **Aligned** – стратегия жизненного цикла, состоящая из нескольких потоков поддержки, синхронизированных с ACP .
- **Agnostic** – стратегия жизненного цикла, состоящая из нескольких потоков поддержки, выпускаемых независимо от ACP .

Подробнее о расширениях см. в разделе Extend.

Перспектива развертывания

ACP состоит из `global` **кластера** и одного или нескольких **кластеров нагрузки**.

- `global` **Кластер**
  - Центральный узел для управления мультикластерной средой
  - Все кластеры должны быть зарегистрированы в `global` перед управлением ими
  - Обеспечивает функциональность мультикластерного и кросс-кластерного взаимодействия
  - Kubernetes развертывается и управляется платформой
- **Кластер нагрузки**
  - Размещает пользовательские нагрузки и сервисы
  - Kubernetes может быть развернут платформой или предоставлен сторонними поставщиками
  - Поддерживает Kubernetes-сервисы основных облачных провайдеров, а также Kubernetes-кластеры, соответствующие CNCF
  - В некоторых сценариях `global` кластер также может размещать бизнес-нагрузки

Техническая перспектива

Выполнение компонентов платформы

Все компоненты платформы работают в контейнерах внутри кластера управления

Kubernetes (кластера `global`).

Архитектура высокой доступности

- Кластер `global` обычно состоит минимум из трёх узлов управления и нескольких рабочих узлов
- Высокая доступность etcd является ключевым элементом HA кластера; подробности см. в разделе *Key Component High Availability Mechanisms*
- Балансировка нагрузки может обеспечиваться внешним балансировщиком или собственным VIP внутри кластера

Маршрутизация запросов

- Клиентские запросы сначала проходят через балансировщик нагрузки или собственный VIP
- Запросы перенаправляются на **ALB** (стандартный Kubernetes Ingress Gateway платформы), работающий на выделенных ingress-узлах (или на узлах управления, если настроено)
- ALB маршрутизирует трафик к целевым подам компонентов согласно настроенным правилам

Стратегия репликации

- Основные компоненты работают минимум с двумя репликами
- Ключевые компоненты (например, registry, MinIO, ALB) работают с тремя репликами

Отказоустойчивость и самовосстановление

- Обеспечивается взаимодействием kubelet, kube-controller-manager, kube-scheduler, kube-proxy, ALB и других компонентов
- Включает проверки состояния, переключение при сбоях и перенаправление трафика

Хранение данных и восстановление

- Конфигурация управляющей плоскости и состояние платформы хранятся в etcd в виде ресурсов Kubernetes
- При катастрофических сбоях восстановление возможно из снимков etcd

Основное / резервное аварийное восстановление

- Два отдельных `global` кластера: **Primary Cluster** и **Standby Cluster**
- Механизм аварийного восстановления основан на синхронизации данных etcd в реальном времени с Primary Cluster на Standby Cluster
- В случае недоступности Primary Cluster из-за сбоя сервисы могут быстро переключиться на Standby Cluster

Механизмы высокой доступности ключевых компонентов

etcd

- Развернут на трёх (или пяти) узлах управления
- Использует протокол RAFT для выбора лидера и репликации данных
- Развертывание на трёх узлах выдерживает отказ одного узла; на пяти — двух
- Поддерживает локальные и удалённые резервные копии снимков S3

Компоненты мониторинга

- **Prometheus**: несколько инстансов, дедупликация с помощью Thanos Query, и отказоустойчивость между регионами
- **VictoriaMetrics**: кластерный режим с распределёнными компонентами VMStorage, VMInsert и VMSelect

Компоненты логирования

- **Nevermore** собирает логи и аудиторские данные
- **Kafka / Elasticsearch / Razor / Lanaya** развернуты в распределённом режиме с несколькими репликами

Компоненты сети (CNI)

- **Kube-OVN / Calico / Flannel**: достигают высокой доступности через безсостояночные DaemonSet или компоненты управляющей плоскости с тройной репликацией

ALB

- Operator развернут с тремя репликами, включён выбор лидера
- Проверки состояния на уровне инстансов и балансировка нагрузки

Собственный VIP

- Виртуальный IP высокой доступности на базе Keepalived
- Поддерживает обнаружение heartbeat и переключение active-standby

Harbor

- Балансировка нагрузки на базе ALB
- PostgreSQL с Patroni для HA
- Redis в режиме Sentinel
- Безсостояночные сервисы развернуты с несколькими репликами

Registry и MinIO

- Registry развернут с тремя репликами
- MinIO в распределённом режиме с кодированием с удалением, избыточностью данных и автоматическим восстановлением

Примечания к выпуску

Содержание

4.0.4

Исправленные ошибки

- 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.

Известные проблемы

No issues in this release.

4.0.3

Исправленные ошибки

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

Известные проблемы

- 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. There is currently no available workaround.
- 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

Исправленные ошибки

- 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.

Известные проблемы

- 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. There is currently no available workaround.

- 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

Исправленные ошибки

- 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.

Известные проблемы

- 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. There is currently no available workaround.

- 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

Новые возможности и улучшения

Установка и обновление: модульная архитектура

Мы полностью переработали архитектуру нашей платформы, чтобы обеспечить беспрецедентную гибкость, ускорить обновления и снизить операционные издержки.

Оптимизированная установка
Наша платформа теперь разворачивается через компактный базовый пакет, содержащий только необходимые компоненты. После установки основы клиенты могут выбирать именно те Operators или плагины кластера, которые им нужны — будь то DevOps, Service Mesh или другие специализированные функции — и загружать, устанавливать их по отдельности.

Целевые патчи

- Патч-релизы включают только те компоненты, которые действительно требуют исправлений.
- Компоненты без исправлений остаются без изменений, что гарантирует сохранность остальной части платформы.

- Клиенты применяют патчи через встроенный стандартизированный механизм обновления платформы, а не вручную обновляют отдельные компоненты, что упрощает сопровождение и отслеживание.

Интеллектуальные обновления

- При обновлении заменяются и перезапускаются только компоненты с новым кодом.
- Неизменённые компоненты сохраняют текущие версии и время работы.
- Это минимизирует время простоя и сокращает окно обслуживания для более плавного обновления.

Независимое версионирование компонентов

- Большинство Operators имеют собственные графики релизов, независимые от основной платформы.
- Новые функции и исправления становятся доступны сразу после готовности — не нужно ждать обновления всей платформы.
- Такой подход ускоряет доставку и позволяет клиентам быстрее получать улучшения.

Кластеры: декларативное управление жизненным циклом с помощью Cluster API

Локальные кластеры теперь используют Kubernetes Cluster API для полностью декларативных операций, включая:

- Создание кластера
- Масштабирование и присоединение узлов

Эта бесшовная интеграция Cluster API напрямую вписывается в ваши IaC-пайплайны, обеспечивая сквозное программное управление жизненным циклом кластера.

Operator & Extension: полный обзор возможностей

Полный каталог Operators

OperatorHub теперь отображает все поддерживаемые Operators, независимо от того, загружены ли их пакеты на платформу. Это улучшение:

- Обеспечивает полный обзор возможностей платформы даже в изолированных средах
- Исключает информационные пробелы между доступным и известным пользователям

- Снижает трение при изучении возможностей платформы

Гибкость выбора версии

Пользователи теперь могут выбирать конкретные версии Operators при установке, а не ограничиваться только последней версией, что даёт больший контроль над совместимостью компонентов и путями обновления.

Расширения Web Console

Operators теперь поддерживают расширения Web Console на основе якорей, позволяя включать функционально-специфичные frontend-образы в Operators и бесшовно интегрировать их в Web Console платформы.

Улучшения плагинов кластера

Все улучшения видимости Operators, выбора версии и возможностей расширения Web Console также применимы к плагинам кластера, обеспечивая единый пользовательский опыт для всех расширений платформы.

Оптимизация логического запроса логов

Страница запроса логов была оптимизирована для решения проблем с удобством и производительностью, с которыми сталкиваются пользователи:

- Оригинальный радиобокс заменён на компонент расширенного поиска. Теперь поиск логов можно использовать так же удобно, как поиск в GIT.
- Независимые условия запроса для содержимого логов
- Положение критериев временного запроса было изменено. Теперь при изменении временного диапазона фильтры логов не сбрасываются.
- Оптимизирован API запроса логов для улучшения общей производительности запросов

Обновление ElasticSearch до версии 8.17

Мы обновили версию ElasticSearch до 8.17, чтобы использовать новые функции и улучшения сообщества.

Аутентификация ALB

ALB теперь поддерживает различные механизмы аутентификации, что позволяет обрабатывать аутентификацию на уровне Ingress вместо реализации в каждом backend-приложении.

ALB поддерживает аннотации ingress-nginx

В этом релизе добавлена поддержка распространённых аннотаций ingress-nginx в ALB, включая настройки keepalive, конфигурации таймаутов и HTTP-перенаправления, что повышает совместимость с ingress-nginx сообщества.

Оптимизация live migration в Kubevirt

Во время процесса live migration время сетевого прерывания сокращено до менее чем 0.5 секунды, и существующие TCP-соединения не разрываются.
Эта оптимизация значительно повышает стабильность и надёжность миграций виртуальных машин в продуктивных средах.

Оптимизация интеграции LDAP/OIDC

Формы интеграции LDAP/OIDC были переработаны, включая удаление ненужных/дублирующих полей и улучшение описаний полей.
Интеграция LDAP/OIDC теперь поддерживает настройку через YAML, позволяя задавать отображение атрибутов пользователя в YAML-файле.

Поддержка Source to Image (S2I)

- Добавлен оператор **Alauda Container Platform Builds** для автоматизированной сборки образов из исходного кода
- Поддержка языковых стеков Java/Go/Node.js/Python
- Упрощение развертывания приложений через репозитории исходного кода

Локальное решение Registry

- **ACP Registry** предоставляет лёгкий Docker Registry с корпоративными возможностями
- Обеспечивает готовые к использованию функции управления образами
- Упрощает доставку приложений

Рефакторинг модуля GitOps

- **ACP GitOps** выделен в отдельную архитектуру плагина кластера
- Обновлён Argo CD до версии v2.14.x
- Улучшено управление жизненным циклом приложений на основе GitOps

Мониторинг на уровне Namespace

- Введены динамические дашборды мониторинга на уровне namespace
- Отображение метрик приложений, workloads и pods

Интеграция Crossplane

- Выпущена дистрибуция **Alauda Build of Crossplane**
- Реализовано приложение-ориентированное предоставление ресурсов через XRD compositions

Обновления виртуализации

- Обновление до KubeVirt 1.4 для расширенных возможностей виртуализации
- Оптимизация работы с образами для ускоренного развертывания BM
- Оптимизация live migration BM с возможностью запуска из UI и отображением статуса миграции
- Улучшена сетевая привязка с поддержкой dual-stack (IPv4/IPv6)
- Добавлена поддержка vTPM для повышения безопасности BM

Обновления Ceph Storage

- Metro-DR с stretch cluster обеспечивает синхронизацию данных в реальном времени между зонами доступности
- Regional-DR с зеркалированием на уровне пулов повышает защиту данных

Обновления TopoLVM

- Добавлена поддержка развертывания multipath устройств, повышающая гибкость и стабильность

Исправленные ошибки

- 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.

Известные проблемы

- 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. There is currently no available workaround.
- 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.
- 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.

- 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.