Хранение данных

Распределённое хранилище Ceph

Введение

Обзор возможностей

Сравнение решений для хранения

Планирование вашего развер

Архитектура развертывания

Вопросы безопасности

Требования к инфраструктуре

Сетевые требования

Архитектур

Техническая ар

Установка

Основные понятия

Руководств

Следующие шаги

Как сделать

MinIO Object Storage

Введение

Установка

Предварительные требования

Архитектур

Основные ком

Основные понятия

ıe:

Руководства

Как сделать

Локальное хранилище TopoLVM

Введение

Установка

Предварительные требования

Руководств

Как сделать

Распределённое хранилище Ceph

Введение

Введение

Обзор возможностей

Сравнение решений для хранения

Планирование вашего развертывания

Планирование вашего развертывания

Архитектура развертывания

Вопросы безопасности

Требования к инфраструктуре

Сетевые требования

Планирование аварийного восстановления

Планирование производительности

Следующие шаги

Архитектура

Архитектура

Техническая архитектура

Установка

Создание кластера стандартн

Предварительные требования

Важные моменты

Порядок действий

Связанные операции

Создание Stretch Type Кластера

Терминология

Типовая схема развертывания

Ограничения и лимитации

Предварительные условия

Процедура

Связанные операции

Основные понятия

Основные концепции

Rook Operator

Ceph CSI

Функции модулей Ceph

Руководства

Доступ к сервисам хранения

- Предварительные требования
- Процедура
- Последующие действия

Управление Storage Pools

- Создание Storage Pool
- Удаление Storage Pool
- Просмотр адресов Object Storage Pool

Развертыв

- Обновление ко
- Перезапуск ко

Добавление устройств/классо

- Добавление классов устройств
- Добавление устройств
- Статус жесткого диска

Мониторинг и оповещения

- Мониторинг
- Оповещения

Как сделать

Замена или удаление устройс

- Предварительные требования
- Ограничения и особенности
- Процедура
- References

Замена или удаление узлов х

- Предварительные условия
- Ограничения и особенности
- Процедура
- References

Настройка

- Архитектура
- Требования к и
- Процедура
- Последующие

Очистка распределённого хра

- Меры предосторожности

Восстановление после сбоев

Обновлени

Процедура

Создание пользователя Ceph Object Store

Предварительные требования

Включение кэша D3N для Ceph RGW

Общая информация

Предварительные требования

Обзор работы

Подготовка локальной файловой систем

Включение кэша D3N в CephObjectStore

Проверка конфигурации D3N

Проверка работы кэша

ка

елы

Настройка шифрования данных в пути

Как включить или отключить шифрование данных в пути н
распределённого хранилища ACP.

Overview

Limitations and prerequisites

Enable in-transit encryption for a new cluster

Enable in-transit encryption after deployment

Disable transport encryption

Verification

Troubleshooting suggestions

Performance impact

Введение

Alauda Build of Rook-Ceph — это гиперконвергентное решение для хранения данных, предоставляемое платформой внутри кластера. Основанное на открытом решении Rook + Ceph, распределённое хранилище обеспечивает автоматическое управление, автоматическое масштабирование и автоматическое восстановление, удовлетворяя потребности малых и средних приложений в блочном, файловом и объектном хранении.

> **NOTE**
>
> В данном документе **распределённое хранилище** означает Ceph-хранилище внутри данного кластера, а **внешнее хранилище** — Ceph-хранилище вне этого кластера.

Содержание

Обзор возможностей

- **Простое развертывание**: Предоставляет графические сервисы автоматического развертывания и управления кластерами хранения; поддерживает как интегрированный, так и раздельный режим развертывания вычислений и хранения.
- **Профессиональные операции**: Обеспечивает функции создания снимков постоянных томов и клонирования новых томов; визуальный мониторинг ёмкости, производительности и состояния компонентов; оснащён встроенными политиками оповещений для удовлетворения большинства сценариев эксплуатации хранилища.
- **Безопасность и надёжность**: Распределённый механизм с мульти-репликами гарантирует безопасность и надёжность данных; простое и надёжное автоматизированное управление поддерживает онлайн-расширение ресурсов хранения.
- **Отличная производительность**: Предоставляет эластичные и высокопроизводительные сервисы хранения; поддерживает развертывание гибридных дисковых устройств для повышения производительности и эффективности системы хранения.

Сравнение решений для хранения

Платформа поддерживает два типа решений для хранения; вы можете выбрать одно из них.

Создание кластера хранения

Требование	Преимущества
Вы можете выбрать создание либо кластера стандартного типа**, либо** кластера расширенного типа	Нет необходимости в дополнительной подготовке решения для хранения; настройка может быть выполнена на бизнес-кластере, что экономит затраты.

Доступ к внешнему хранилищу

Вариант 1: Доступ к распределённым ресурсам хранения других бизнес-кластеров внутри платформы для обеспечения изоляции хранения и бизнеса, что упрощает

управление и сопровождение.

Вариант 2: Интеграция внешних Ceph-ресурсов хранения в качестве распределённого хранилища.

Требование (выберите одно)	Преимущества
Вариант 1: Распределённое хранилище уже развернуто в других бизнес-кластерах.	Позволяет полноценно использовать ресурсы хранения между кластерами и избегать влияния изменений в бизнесе. Обеспечивает безопасность и стабильность данных при снижении сложности эксплуатации. **Примечание**: Если хранилище для доступа — распределённое хранилище с других платформ, например, основная/резервная платформа в среде аварийного восстановления, используйте метод интеграции внешнего Ceph.
Вариант 2: Внешнее Ceph-хранилище вне платформы, версия ≥ 14.2.3.	По сравнению с прямым созданием класса хранения, этот метод удобнее для использования интерфейса платформы для создания снимков томов, масштабирования и других функций.

Примечание: Если необходимо поддерживать пул хранения, устройства хранения и другие настройки внешнего хранилища, операции должны выполняться в интерфейсе управления кластера хранения.

Планирование вашего развертывания

В этой теме представлен контрольный список для планирования развертывания распределённого хранилища Ceph на платформе Alauda Container Platform (ACP). В ней суммируются архитектурные решения, варианты безопасности, размеры инфраструктуры, сетевые ограничения и вопросы аварийного восстановления, чтобы вы могли выбрать модель развертывания до начала фактической установки.

Для ознакомления с продуктом смотрите Introduction и Architecture. Для процедур развертывания смотрите документы в разделах Install и How To.

Содержание

Архитектура развертывания

Распределённое хранилище ACP основано на Ceph и Rook. В общих чертах платформа объединяет следующие уровни:

- Демоны Ceph, такие как MON, MGR, OSD, MDS и RGW, обеспечивающие возможности блочного, файлового и объектного хранения
- Компоненты Rook и CSI для автоматизации развертывания, предоставления ресурсов, масштабирования и управления жизненным циклом
- Интеграция с платформой ACP для предоставления пулов хранения, наблюдаемости и точек входа в операции

Перед развертыванием определитесь, будет ли ваше окружение использовать сервисы хранения из локального кластера или потреблять хранилище из внешней среды Ceph.

Внутренние и внешние модели развертывания

Вы можете спланировать распределённое хранилище ACP одним из следующих способов:

Модель развертывания	Где работают сервисы хранения	Кто управляет кластером хранения	Лучшее применение	Основ компр
Внутреннее, совместное	Компоненты Ceph работают на тех же рабочих узлах ACP, что и бизнес-нагрузки	Команда платформы ACP или администратор кластера	Ранние этапы, bare metal кластеры или ситуации с неясными требованиями к хранению	Проще развер выше конкур ресурс прило храни
Внутреннее, выделенные узлы	Компоненты Ceph работают на выделенных узлах хранения или инфраструктуры внутри того же кластера ACP	Команда платформы ACP или администратор кластера	Производственные среды с предсказуемым спросом на хранение и более строгими требованиями к изоляции	Лучша опера изоля контр но тре зарез узлов плани
Внешнее	ACP потребляет storage классы из внешней среды Ceph	Отдельная команда хранения, SRE или существующий владелец внешнего хранилища	Крупномасштабные среды, несколько потребительских кластеров или организации с уже работающим отдельным кластером Ceph	Чёткое разгра ответс но бол взаим аутент управл зависи

Внутреннее развертывание проще в развертывании и управлении, так как сервисы хранения и потребляющие нагрузки планируются в одном окружении ACP. Внутри внутреннего развертывания первый выбор — использовать ли совместные узлы с бизнес-нагрузками или выделенные узлы. Внешнее развертывание предпочтительно, когда требуется более сильное разделение между кластерами хранения и приложений или когда несколько бизнес-кластеров должны использовать одно и то же хранилище.

Основные точки принятия решения:

- Выбирайте совместное развертывание, если хотите более быстрое развертывание и можете допустить совместное использование рабочих узлов приложениями и хранилищем.
- Выбирайте выделенные узлы, если спрос на хранение известен и нужна более чёткая контроль ёмкости, изоляция сбоев и границы обслуживания.
- Выбирайте внешнее развертывание, если хранилище уже управляется отдельно или если один внешний кластер должен обслуживать несколько ACP кластеров.

Роли узлов

При планировании размещения узлов разделяйте обязанности узлов управляющей плоскости, инфраструктурных узлов и рабочих узлов:

- Узлы управляющей плоскости поддерживают функции управления кластером и не должны рассматриваться как узлы общего назначения для хранения, если модель развертывания явно это не поддерживает.
- Инфраструктурные узлы подходят, если вы хотите изолировать компоненты платформы хранения от бизнес-нагрузок.
- Рабочие узлы могут размещать сервисы хранения в совместных развертываниях, но это увеличивает конкуренцию за ресурсы между приложениями и демонами хранения.

Для производственного использования планируйте минимум три домена отказа для высокодоступных сервисов хранения. Распределяйте узлы хранения по стойкам, зонам или группам хостов, где это возможно.

Вопросы безопасности

Перед развертыванием подтвердите, требуется ли шифрование трафика в пути для дизайна хранения, и оцените операционное влияние перед его включением.

Шифрование трафика в пути

ACP в настоящее время поддерживает шифрование трафика в пути для распределённого хранилища Ceph. Эта функция защищает трафик между компонентами Ceph и клиентами и обычно планируется с учётом Ceph `msgr2` и модели сетевого кластера.

Перед включением шифрования трафика в пути проверьте:

- Поддержку ядра и операционной системы на узлах хранения и клиентах
- Ожидаемую нагрузку на CPU на загруженных узлах хранения
- Влияние на пропускную способность и задержки на целевом оборудовании

Для деталей реализации смотрите Configure in-transit encryption.

Требования к инфраструктуре

Минимальная и рекомендуемая конфигурация

Планируйте количество узлов, устройства хранения и доступные ресурсы до создания кластера.

Параметр	Минимальная конфигурация	Рекомендуемая конфигурация
Узлы хранения	3 узла	3 и более узлов, распределённых по доменам отказа
Устройства хранения	1 доступное устройство хранения на узел	Несколько выделенных устройств на узел, с одинаковым типом и размером
Распределение узлов	3 узла доступны для размещения сервисов Ceph	3 домена отказа, например стойки или зоны
Использование устройств	Отдельный системный диск и диск хранения	Выделенные raw-диски для данных Ceph и запас для будущего расширения

Минимум — это три узла и одно доступное устройство хранения на каждом. Для производственного использования развертывайте кластер минимум в трёх доменах отказа и резервируйте достаточно свободных ресурсов для ребалансировки, ремонта и будущего роста.

Размер ресурсов

Сервисы хранения Ceph постоянно потребляют CPU, память и ёмкость устройств. Планируйте ресурсы сначала для демонов хранения, затем резервируйте дополнительный запас для восстановления, ребалансировки, обновлений и фоновых задач.

В качестве базового ориентира:

- Начинайте с минимум трёх узлов хранения для высокодоступного кластера
- Резервируйте CPU и память для MON, MGR, OSD и любых включённых MDS или RGW сервисов
- Оставляйте запас для новых пулов, дополнительных устройств и событий восстановления кластера
- Избегайте планирования кластера, который уже на старте близок к насыщению ресурсов

Если в дизайне используются выделенные узлы хранения, планирование ресурсов более предсказуемо. Если хранение работает вместе с бизнес-нагрузками, резервируйте дополнительный запас для поглощения конкуренции в пиковые нагрузки и при сбоях узлов.

Общий бюджет планирования кластера

Для раннего планирования начинайте с общего бюджета кластера, а не только с отдельных значений компонентов. Следующая таблица предназначена как ориентир для трёхузлового высокодоступного кластера до настройки под конкретные нагрузки:

Модель развертывания	Общий CPU для хранения	Общая память для хранения	Примечания
Внутреннее, минимальный базовый уровень	24 логических CPU	72 GiB	Базовый уровень планирования для трёх узлов при минимальной цели развертывания
Внутреннее, стандартный базовый уровень	30 логических CPU	72 GiB	Лучшее начальное значение для общего производственного планирования и будущего расширения
Внутреннее, ориентированное на производительность	45 логических CPU	96 GiB	Подходит, если с самого начала требуется высокая пропускная способность или низкая задержка
Внешний потребительский кластер	Размер по требованиям сетевого подключения и доступа клиентов	Размер по требованиям сетевого подключения и доступа клиентов	Демоны хранения работают вне кластера ACP, поэтому кластер ACP в основном нуждается в сетевой доступности, учётных данных и ресурсах клиента

Эти значения следует рассматривать как целевые показатели планирования на уровне кластера, а не как точные резервы планировщика. Чтобы оценить бюджет на узел для трёхузлового кластера, равномерно разделите общие значения между участвующими узлами хранения.

Следующие рекомендации подходят для раннего планирования:

Компонент	Рекомендуемый CPU	Рекомендуемая память
MON	2 ядра	3 GiB
MGR	3 ядра	4 GiB
MDS	3 ядра	8 GiB
RGW	2 ядра	4 GiB
OSD	4 ядра	8 GiB

Эти значения — ориентиры планирования, а не жёсткие гарантии планирования. Фактические требования зависят от количества устройств, включённых сервисов и интенсивности нагрузки.

Как оценить размер кластера

Используйте следующий порядок при оценке размера кластера:

1. Выберите модель развертывания: совместное, выделенные узлы или внешнее.
2. Определите минимальное количество узлов и расположение доменов отказа.
3. Решите, нужны ли блочные, файловые, объектные или смешанные сервисы хранения.
4. Начните с общего бюджета планирования кластера.
5. Добавьте запас для дополнительных наборов устройств, восстановления, мониторинга и ожидаемого роста.

Если требуются и файловые, и объектные сервисы, или если кластер будет одновременно обслуживать тяжёлые бизнес-нагрузки, планируйте выше минимального базового уровня.

Размещение Pod

Правила размещения Pod напрямую влияют на отказоустойчивость. Планируйте кластер так, чтобы:

- Высокодоступные компоненты могли быть распределены по разным доменам отказа
- В каждом домене отказа были доступны устройства хранения и достаточно ресурсов для выделения
- Новые наборы устройств или будущие расширения могли следовать той же схеме размещения

На практике это означает, что просто иметь три узла недостаточно. Узлы должны быть распределены так, чтобы избежать единой точки отказа в стойке, группе хостов или зоне.

Планирование устройств хранения

При выборе устройств хранения стандартизируйте размер и класс устройств насколько возможно. Смешанные устройства усложняют настройку производительности и планирование ёмкости.

Используйте следующие принципы:

- Резервируйте один системный диск для ОС и отдельные устройства для данных Ceph
- Предпочитайте raw-диски или выделенные устройства вместо разделения общих дисков
- Держите количество устройств на узел на управляемом уровне, чтобы восстановление и обслуживание оставались практичными
- Отслеживайте используемую ёмкость, а не сырую, так как репликация уменьшает эффективное пространство хранения

Планирование ёмкости также должно включать пороги оповещений и политику расширения. Планируйте расширение до того, как кластер достигнет почти полного состояния. Работа близко к полной ёмкости увеличивает нагрузку на ребалансировку и усложняет восстановление.

Для связанной операционной информации смотрите Managing Storage Pools и Adding Devices/Device Classes.

Планирование ёмкости

При планировании ёмкости кластера рассчитывайте используемую ёмкость, а не сырую ёмкость дисков. В реплицированном развертывании Ceph часть сырого пространства всегда расходуется на защиту данных.

Используйте следующие принципы планирования:

- Держите доступную ёмкость выше ожидаемого роста бизнеса, а не расширяйте только после почти полного заполнения кластера
- Резервируйте дополнительный запас для восстановления, ребалансировки, снимков и временных всплесков использования данных
- Расширяйте хранилище сбалансированно по узлам и доменам отказа, чтобы новая ёмкость не создавала перекос в использовании
- Анализируйте текущую загрузку и прогнозируемый рост перед добавлением новых нагрузок в кластер

Следующие примеры можно использовать как ориентиры раннего планирования для трёхузлового кластера с одним устройством на узел и политикой защиты данных с 3 репликами:

Размер устройства на узел	**Сырая ёмкость кластера**	**Приблизительная используемая ёмкость с 3 репликами**
0.5 TiB	1.5 TiB	0.5 TiB
2 TiB	6 TiB	2 TiB
4 TiB	12 TiB	4 TiB

Эти значения приведены только как примеры. Используемая ёмкость зависит от фактической политики защиты данных и не должна рассматриваться как универсальное правило для всех дизайнов кластера.

В операциях второго дня ёмкость следует пересматривать до достижения предупреждающих уровней. Если рост прогнозируемый, расширяйте заранее, а не ждите почти полного или полного состояния.

Сетевые требования

Ceph чувствителен к качеству сети. Перед развертыванием проверьте:

- Сеть кластера обеспечивает стабильную пропускную способность для трафика репликации и восстановления
- Задержка между доменами отказа находится в поддерживаемом диапазоне для выбранной модели развертывания
- Необходимые порты открыты между узлами хранения и потребляющими кластерами
- Любая выделенная сетевая архитектура, например разделение на основе Multus, согласована заранее

Если планируете изолировать трафик хранения от общего трафика приложений, подтвердите сетевые интерфейсы, политику маршрутизации и ответственность за эксплуатацию до развертывания. Сетевая изоляция улучшает безопасность и производительность, но увеличивает сложность дизайна.

Поддержка IPv6

Планирование распределённого хранилища ACP должно соответствовать выбранному стеку сети кластера платформы.

- IPv6 поддерживается в одностековых IPv6 окружениях.
- Планирование двойного стека должно быть проверено в соответствии с сетевым дизайном кластера ACP до развертывания хранения.
- Узлы хранения и клиенты должны использовать одну и ту же стратегию адресации, чтобы избежать проблем с подключением и обнаружением сервисов.

Если в вашем окружении используется IPv6, подтвердите перед установкой:

- Сеть кластера ACP уже настроена для работы с IPv6
- Все узлы хранения могут общаться по необходимым IPv6 маршрутам
- Мониторинг, оповещения и внешние интеграции, обращающиеся к конечным точкам хранения, также поддерживают IPv6

IPv6 следует рассматривать как архитектурное решение на этапе установки. Не предполагайте, что существующий дизайн хранения, ориентированный на IPv4, можно будет конвертировать позже без повторной проверки.

Планирование аварийного восстановления

Распределённое хранилище ACP можно планировать с разными целями восстановления. Выберите модель на основе ваших целей по точке восстановления (RPO), времени восстановления (RTO) и топологии площадок.

Regional-DR

ACP поддерживает Regional-DR для сценариев аварийного восстановления между регионами или площадками, где допустима асинхронная репликация и небольшой объём потенциальной потери данных.

При планировании Regional-DR заранее подтвердите:

- Совместимость дизайна хранения и сети исходного и целевого кластеров
- Задержки репликации и ожидания переключения соответствуют бизнес-целям восстановления
- Тип защищаемой нагрузки ясен, например блочные, файловые или объектные данные

Для деталей реализации смотрите Disaster Recovery.

Stretch Cluster

Stretch cluster подходит только при строго контролируемой задержке между площадками и специально спроектированной топологии для этого сценария. В общем случае планируйте:

- Две площадки с данными и одну площадку с кворумом или арбитром
- Минимум пять узлов в трёх зонах
- Ручное и явное назначение меток доменов отказа до создания кластера

- Достаточное количество узлов на каждой площадке с данными для сохранения доступности сервисов хранения
- Межзонная задержка в пределах низколатентного дизайна, обычно не более 10 мс RTT между площадками с данными

> **WARNING**
>
> Не рассматривайте stretch cluster как универсальное решение для развертывания на большие расстояния с высокой задержкой и несколькими датацентрами. Если задержка между площадками не контролируется строго, используйте выделенную архитектуру аварийного восстановления.

Для рекомендаций по развертыванию stretch cluster в ACP смотрите Create Stretch Type Cluster.

Планирование производительности

Производительность следует планировать исходя из характеристик нагрузки, а не только по количеству устройств. Перед развертыванием определите:

- Основной тип нагрузки: блочная, файловая или объектная
- Чувствительность нагрузки к задержкам, пропускной способности или объёму
- Будут ли в кластере доминировать горячие данные, резервное копирование или аналитические задачи

Также подтвердите, требуется ли специальная настройка или дизайн с учётом особенностей. Например, объектные нагрузки могут требовать отдельного планирования ёмкости шлюзов, а некоторые окружения — кэш-ориентированные или выделенные кластеры.

Следующие шаги

После завершения планирования переходите к руководству по развертыванию, соответствующему выбранной модели:

Внутреннее развертывание

- Для совместного развертывания смотрите Create Standard Type Cluster.
- Для stretch cluster смотрите Create Stretch Type Cluster.
- Для развертывания с выделенными узлами смотрите Configure a Dedicated Cluster for Distributed Storage.

Внешнее развертывание

- Для потребления сервисов хранения из другого кластера или внешней среды Ceph смотрите Accessing Storage Services.

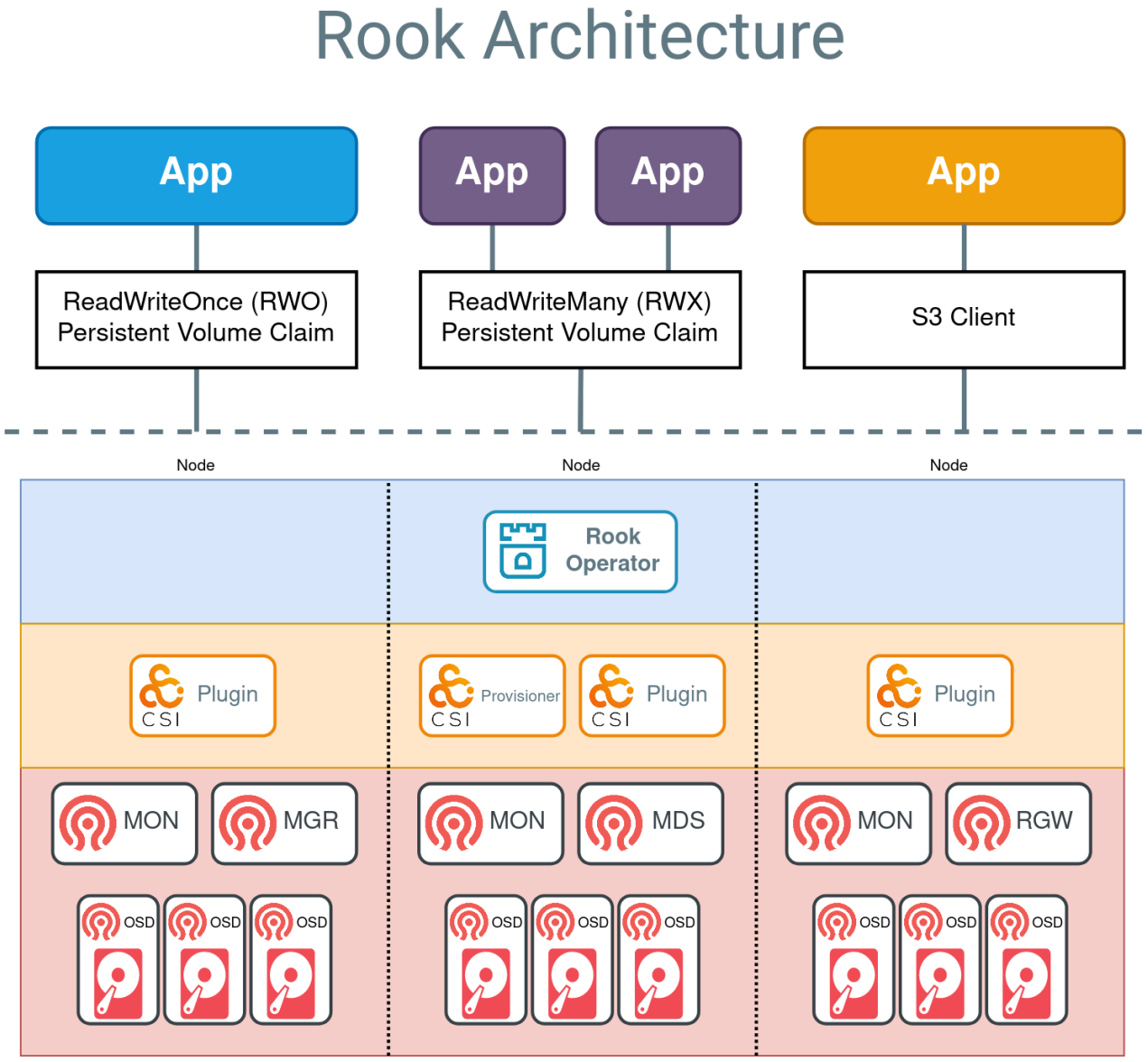
Связанные последующие настройки

- Для включения шифрования сетевого трафика для развернутых сервисов хранения смотрите Configure in-transit encryption.
- Для настройки аварийного восстановления после развертывания смотрите Disaster Recovery.

Архитектура

Содержание

Техническая архитектура



На рисунке выше показаны примерные приложения для трёх поддерживаемых типов хранилищ:

- Блочное хранилище представлено синим приложением, к которому подключён том с режимом ReadWriteOnce (RWO). Приложение может читать и записывать данные в том RWO, в то время как Ceph управляет вводом-выводом.
- Общая файловая система представлена двумя фиолетовыми приложениями, которые совместно используют том с режимом ReadWriteMany (RWX). Оба приложения могут одновременно активно читать и записывать данные в том. Ceph обеспечивает безопасную защиту данных для нескольких писателей с помощью демона MDS.

- Объектное хранилище представлено оранжевым приложением, которое может читать и записывать данные в бакет с помощью стандартного клиента S3.

Ниже пунктирной линии на диаграмме компоненты разделены на три категории:

- Rook operator (синий слой): Оператор автоматизирует конфигурацию Ceph
- CSI plugins and provisioners (оранжевый слой): Драйвер Ceph-CSI обеспечивает создание и монтирование томов
- Ceph daemons (красный слой): Демоны Ceph обеспечивают работу основной архитектуры хранилища. Подробнее о каждом демоне см. в Глоссарии.

Блочное хранилище

На диаграмме выше процесс создания приложения с томом RWO следующий:

- (Синее) приложение создаёт PVC для запроса хранилища.
- PVC определяет класс хранения Ceph RBD (sc) для создания хранилища.
- K8s вызывает provisioner Ceph-CSI RBD для создания образа Ceph RBD.
- kubelet вызывает CSI RBD volume plugin для монтирования тома в приложении.
- Том теперь доступен для чтения и записи.
- Том с режимом ReadWriteOnce может быть смонтирован только на одном узле одновременно.

Общая файловая система

На диаграмме выше процесс создания приложений с томом RWX следующий:

- (Фиолетовое) приложение создаёт PVC для запроса хранилища.
- PVC определяет класс хранения CephFS (sc) для создания хранилища.
- K8s вызывает provisioner Ceph-CSI CephFS для создания подтома CephFS.
- kubelet вызывает CSI CephFS volume plugin для монтирования тома в приложении.
- Том теперь доступен для чтения и записи.
- Том с режимом ReadWriteMany может быть смонтирован на нескольких узлах для использования приложением.

Объектное хранилище S3

На диаграмме выше процесс создания приложения с доступом к бакету S3 следующий:

- (Оранжевое) приложение создаёт BucketClaim для запроса бакета.
- Ceph COSI Driver создаёт бакет Ceph RGW.
- Ceph COSI Driver создаёт секрет с учётными данными для доступа к бакету.
- Приложение получает учётные данные из секрета.
- Приложение теперь может читать и записывать данные в бакет с помощью клиента S3.

Установка

Создание кластера стандартного типа

Кластер стандартного типа — это наиболее типичный способ развертывания хранилища Ceph. Он распределяет реплики данных по жестким дискам на разных хостах, обеспечивая, что при отказе одного хоста копии данных на других хостах смогут поддерживать доступность сервиса.

Содержание

Предварительные требования

Подготовка пакетов

- **Скачайте** установочный пакет **Alauda Container Platform Storage Essentials**, соответствующий архитектуре вашей платформы.
- **Загрузите** установочный пакет **Alauda Container Platform Storage Essentials** с помощью механизма Upload Packages.
- **Скачайте** установочный пакет **Alauda Build of Rook-Ceph**, соответствующий архитектуре вашей платформы.
- **Загрузите** установочный пакет **Alauda Build of Rook-Ceph** с помощью механизма Upload Packages.

Подготовка инфраструктуры

- Для кластера хранения требуется не менее 3 узлов.
- Каждый узел должен иметь как минимум 1 пустой жесткий диск или 1 неформатированный раздел жесткого диска.
- Рекомендуемый объем доступного дискового пространства — более 50 ГБ.
- Если вы используете присоединенный Kubernetes кластер с Containerd в качестве компонента runtime, убедитесь, что параметр `LimitNOFILE` в файле `/etc/systemd/system/containerd.service` на всех узлах кластера установлен в значение `1048576` для успешного развертывания распределенного хранилища. Инструкции по настройке см. в разделе Modifying Containerd Configuration Information. **Примечание**: При обновлении с версий ранее v3.10.2 до текущей версии, если необходимо развернуть распределенное хранилище Ceph на вашем кастомном Kubernetes кластере с Containerd в качестве runtime, также необходимо установить параметр `LimitNOFILE` в файле `/etc/systemd/system/containerd.service` на всех узлах кластера в значение `1048576`.

Важные моменты

Создание сервиса хранения и **Доступ к сервису хранения** поддерживают выбор только одного метода.

Порядок действий

1 Развертывание Alauda Container Platform Storage Essentials

1. Войдите в систему, перейдите на страницу **Administrator**.
2. Нажмите **Marketplace** > **OperatorHub**, чтобы перейти на страницу **OperatorHub**.
3. Найдите **Alauda Container Platform Storage Essentials**, нажмите **Install** и перейдите на страницу **Install Alauda Container Platform Storage Essentials**. Параметры конфигурации:

Параметр	Рекомендуемая конфигурация
Channel	По умолчанию канал `stable`.
Installation Mode	`Cluster`: Все пространства имен в кластере используют один экземпляр Operator для создания и управления, что снижает использование ресурсов.
Installation Place	Выберите `Recommended`, Namespace поддерживается только **acp-storage**.
Upgrade Strategy	`Manual`: При появлении новой версии в Operator Hub требуется ручное подтверждение для обновления Operator до последней версии.

2 (Опционально) Развертывание Alauda Build of LocalStorage

При использовании метода `Selection Device` для добавления устройств хранения в кластер Ceph необходимо развернуть `Alauda Build of LocalStorage`

`Operator`. Этот Operator автоматически обнаруживает все жесткие диски на каждом узле Kubernetes кластера и собирает полную информацию об устройствах, упрощая процесс интеграции хранилища.

1. Войдите в систему, перейдите на страницу **Administrator**.
2. Нажмите **Marketplace** > **OperatorHub**, чтобы перейти на страницу **OperatorHub**.
3. Найдите **Alauda Build of LocalStorage**, нажмите **Install** и перейдите на страницу **Install Alauda Build of LocalStorage**.
   Параметры конфигурации:

Параметр	Рекомендуемая конфигурация
Channel	По умолчанию канал `stable`.
Installation Mode	`Cluster`: Все пространства имен в кластере используют один экземпляр Operator для создания и управления, что снижает использование ресурсов.
Installation Place	Выберите `Recommended`, Namespace поддерживается только **acp-storage**.
Upgrade Strategy	`Manual`: При появлении новой версии в Operator Hub требуется ручное подтверждение для обновления Operator до последней версии.

3 Развертывание Operator

1. Перейдите в раздел **Administrator**.
2. В левой боковой панели нажмите **Storage Management** > **Distributed Storage**.
3. Нажмите **Configure Now**.
4. На странице мастера **Deploy Operator** нажмите кнопку **Deploy Operator** в правом нижнем углу.

   - Автоматический переход страницы к следующему шагу означает успешное развертывание Operator.

- Если развертывание не удалось, следуйте подсказке интерфейса **Clean Up Deployed Information and Retry** и повторите развертывание Operator; чтобы вернуться на страницу выбора распределенного хранилища, нажмите **Application Store**, сначала удалите ресурсы уже развернутого **rook-operator**, затем удалите сам **rook-operator**.

4 Создание кластера

1. На странице мастера **Create Cluster** настройте соответствующие параметры и нажмите кнопку **Create Cluster** в правом нижнем углу.

Параметр	Описание
Cluster Type	Выберите **Standard**.
Device Class Type	Классы устройств — это группировки жестких дисков; вы можете настраивать классы устройств в соответствии с вашими требованиями к хранению, распределяя разные типы данных по дискам с разной производительностью. • **Default Device Class**: Платформа автоматически классифицирует типы жестких дисков на узлах кластера, например, создавая классы устройств с именами `hdd`, `ssd`, `nvme`. • **Custom Device Class**: Позволяет задать имя класса устройств для конкретных комбинаций дисков на узле; поддерживается добавление нескольких классов устройств. Один жесткий диск может принадлежать только одному классу устройств.
Device Class - Name	Имя класса устройств. При выборе **Custom Device Class** имя класса не может быть `hdd`, `ssd`, `nvme`.
Device Class - Storage Devices	Для добавления устройств хранения в класс устройств можно выбрать методы `Selection Device` и `Input Device`:

Параметр	Описание

	• **Selection Device**:
	Выбор из доступных устройств хранения. Устройство считается доступным, если выполняются следующие условия:
	  • Тип устройства — диск или mpath
	  • Файловая система не обнаружена (fsType пустой)
	  • Емкость превышает 10 ГиБ
	Устройства типа rbd, nbd и dm-* не отображаются в списке доступных для выбора.
	
	Примечание: Требуется предварительное развертывание Alauda Build of LocalStorage Operator.
	• **Input Device**:
	Ручной ввод имен пустых устройств на узле, например, `sda`.
	
	Примечание: Для оптимальной производительности и управления настоятельно рекомендуется использовать необработанные диски, а не отдельные разделы на диске.

Snapshot	При включении поддерживается создание снимков PVC и использование снимков для настройки новых PVC для быстрого резервного копирования и восстановления данных. Если снимки не были включены при создании хранилища, их можно включить позже в разделе **Operations** на странице деталей кластера хранения. **Примечание**: Перед использованием убедитесь, что для текущего кластера развернуты плагины volume snapshot.
Monitoring Alarm	При включении обеспечивается готовый сбор метрик мониторинга и возможности оповещений, см. Monitoring and Alarming.

Параметр	Описание
	Примечание: Если не включать сейчас, потребуется использовать альтернативные решения для мониторинга и оповещений хранилища, например, вручную настраивать панели мониторинга и стратегии оповещений в центре эксплуатации.

2. Нажмите **Advanced Configuration** для расширенной настройки компонентов.

Параметр	Описание
Network Configuration	• **Host Network**: Кластер хранения будет использовать сеть хоста, необходимо заполнить соответствующие параметры оптимизации сети в колонке параметров оптимизации, например, указать подсети `public` и `cluster`. Если оставить пустым, будет использована подсеть хоста по умолчанию. **Примечание**: Использование сети хоста может представлять угрозу безопасности из-за передачи данных в незашифрованном виде через порты хоста. Обратитесь в службу поддержки платформы для получения решения по шифрованию передачи. • **Container Network**: Кластер хранения будет использовать контейнерную сеть; можно создать подсети в управлении сетью и назначить их пространству имен `rook-ceph`. Если оставить пустым, будет использована подсеть по умолчанию. **Примечание**: IPv6 не поддерживается. При использовании контейнерной сети доступ к хранилищу возможен только внутри кластера. Сбои или перезапуски Pod Ceph CSI могут привести к прерыванию сервиса.

Параметр	Описание
Optimization Parameters	Поддерживается заполнение параметров в формате конфигурационного файла Ceph; система перезапишет параметры по умолчанию на основе предоставленного содержимого. **Примечание**: После первого заполнения или изменения параметров инициализации необходимо нажать на параметры инициализации; успешная инициализация обязательна для создания кластера.
Component Fixed-point Deployment	Можно развернуть компоненты на указанных узлах; требуется минимум три узла для обеспечения минимальной доступности. Компоненты, поддерживающие настройку фиксированного размещения: MON, MGR, MDS, RGW.

- Автоматический переход страницы к следующему шагу означает успешное развертывание кластера Ceph.
- Если создание не удалось, можно нажать очистку **Created Information or Retry** для автоматической очистки ресурсов и повторного создания кластера, либо вручную очистить ресурсы согласно документации Distributed Storage Service Resource Cleanup.

5 Создание пула хранения

1. На странице мастера **Create Storage Pool** настройте соответствующие параметры и нажмите кнопку **Create Storage Pool** в правом нижнем углу.

Параметр	Описание
Storage Type	• Файловое хранилище: обеспечивает безопасные, надежные и масштабируемые услуги совместного файлового хранения. Подходит для совместного доступа к файлам, резервного копирования и т.д. • Блочное хранилище: обеспечивает высокую производительность IOPS и низкую задержку. Подходит для баз

Параметр	Описание
	данных, виртуализации и т.д. • Объектное хранилище: предоставляет стандартный интерфейс S3, подходит для больших данных, резервного архивирования, облачного хранения и т.д.
Replica Count	Чем больше количество реплик, тем выше избыточность и безопасность данных, однако уменьшается эффективность использования хранилища. Обычно устанавливается значение 3, что удовлетворяет большинству потребностей.
Device Class	Единообразная классификация устройств одного типа или дисков с одинаковой бизнес-логикой, выбирается из классов устройств, добавленных на предыдущем шаге. • При выборе класса устройств данные будут храниться в выбранном классе. • Если класс устройств не выбран, данные будут случайным образом распределены по всем устройствам в пуле хранения.

Для объектного хранилища необходимо также настроить следующие параметры:

Параметр	Описание
Region	Укажите регион, в котором расположен пул хранения.
Gateway Type	По умолчанию S3, изменить нельзя.
Internal Port	Укажите порт для внутреннего доступа в кластере.
External Access	Включение/отключение внешнего доступа создаст/удалит сервис типа Nodeport.

Параметр	Описание
Instance Count	Количество экземпляров ресурсов для объектного хранилища.

- Автоматический переход страницы к следующему шагу означает успешное развертывание пула хранения.
- Если развертывание не удалось, проверьте основные компоненты согласно подсказкам интерфейса, затем нажмите **Clean Up Created Information and Retry** для повторного создания пула хранения.

2. Нажмите **Create Storage Pool**. Во вкладке **Details** можно просмотреть информацию о созданном пуле хранения.

Связанные операции

Создание кластера типа Stretch

Подробности см. в разделе Create Stretch Type Cluster.

Очистка распределенного хранилища

Подробности см. в разделе Cleanup Distributed Storage.

Создание Stretch Type Кластера

Stretch кластер может распространяться на две географически разнесённые локации, обеспечивая возможности аварийного восстановления для инфраструктуры хранения. В случае катастрофы, когда одна зона доступности из двух полностью недоступна, Ceph всё равно может поддерживать доступность.

> ⚠️ **ВАЖНО: Отказ от ответственности за Alpha-функцию**
> В этом документе описывается функция, находящаяся на **этапе Alpha**, предоставляемая исключительно для ранней технической проверки и оценки. Как поставщик функции, мы не даём никаких гарантий относительно её стабильности, надёжности или целостности данных. Настраивая и используя эту функцию, вы признаёте и принимаете следующие технические ограничения:
>
> - **Не для использования в продакшене:** Эта функция не прошла всестороннюю системную проверку. Использование её в продуктивных средах связано с высоким риском сбоев процессов или повреждения данных.
> - **Отсутствие гарантии SLA:** Эта функция не подпадает под стандартные соглашения об уровне обслуживания (SLA). Мы не гарантируем сроки реакции технической поддержки и не предоставляем экстренные хотфиксы.
> - **Нарушающие совместимость изменения и устаревание:** API, схемы конфигураций и основная логика обработки могут претерпевать несовместимые изменения в будущих релизах. Функция также может быть полностью удалена без предварительного уведомления.
> - **Отсутствие плавного пути обновления:** Мы не предоставляем скрипты обновления или инструменты миграции данных между версиями. Обновление обычно требует полного удаления существующих ресурсов и новой установки. Любые потери состояния или данных ложатся на ответственность пользователя.

Содержание

Терминология

Термин	Объяснение
Quorum Availability Zone	Обычно располагается в отдельной зоне, не несущей основные рабочие нагрузки, сосредоточена на поддержании согласованности кластера и используется преимущественно для арбитражных решений при сбоях в основном дата-центре или сетевых разрывах.
Data Availability Zone	Основная зона в кластере Ceph, где фактически хранятся и обрабатываются данные, несущая операционные нагрузки и задачи хранения, вместе с зоной кворума формирующая полноценную систему высокодоступного хранения.

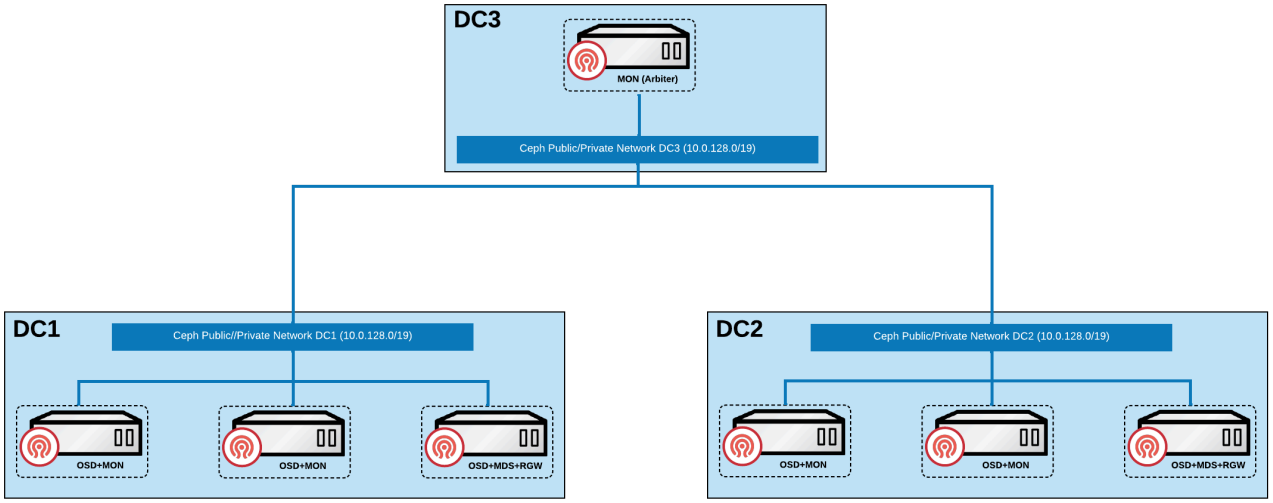
Типовая схема развертывания

Ниже приведена типовая схема развертывания stretch кластеров, а также описание компонентов и принципы аварийного восстановления.

Описание компонентов

Узлы необходимо распределить по трём зонам доступности: две зоны data availability и одна зона quorum availability.

- В обеих зонах data availability необходимо полностью развернуть все основные компоненты Ceph (MON, OSD, MGR, MDS, RGW), при этом в каждой зоне data availability должно быть настроено по два экземпляра MON для обеспечения высокой доступности. Если оба экземпляра MON в одной зоне data availability становятся недоступны, система считает эту зону в состоянии сбоя.
- В зоне quorum availability требуется развернуть только один экземпляр MON, который служит узлом арбитражного решения.



Объяснение аварийного восстановления

- При полном отказе одной из зон data availability Ceph кластер автоматически переходит в деградированное состояние и генерирует уведомление об аварии. Система изменит минимальное количество реплик в пуле хранения (min_size) с дефолтного значения 2 до 1. Поскольку другая зона data availability сохраняет двойные реплики, кластер остаётся доступным. После восстановления отказавшей зоны система автоматически выполнит синхронизацию данных и вернётся в здоровое состояние; если сбой не удаётся устранить, рекомендуется заменить её на новую зону data availability.

- При разрыве сетевого соединения между двумя зонами data availability, но при сохранении нормального соединения с зоной quorum availability, зона quorum будет арбитражно выбирать между двумя зонами data availability на основе заданных политик, выбирая зону в лучшем состоянии для продолжения предоставления сервисов в качестве основной зоны данных.

Ограничения и лимитации

- **Ограничения пулов хранения:** Пулы с кодированием с удалением (erasure-coded) не поддерживаются, допускается только механизм репликации для защиты данных.
- **Ограничения классификации устройств:** Функциональность device class не поддерживается, хранение не может быть стратифицировано по характеристикам устройств.
- **Ограничения регионального развертывания:** Поддерживаются только две зоны data availability; более двух зон data availability быть не может.
- **Требования к балансировке данных:** Вес OSD в двух зонах data availability должен строго совпадать для обеспечения сбалансированного распределения данных.
- **Требования к носителям:** Разрешены только конфигурации All-Flash OSD, что минимизирует время восстановления после восстановления соединения и максимально снижает риск потери данных.
- **Требования к сетевой задержке:** RTT (время кругового прохода) между двумя зонами data availability не должен превышать 10 мс, а зона quorum availability должна соответствовать требованиям по задержкам спецификации ETCD для обеспечения надёжности арбитражного механизма.

Предварительные условия

Заранее классифицируйте все или часть узлов кластера по трём зонам доступности следующим образом:

- Обеспечьте распределение не менее 5 узлов между одной зоной quorum availability и двумя зонами data availability. В зоне quorum availability должен быть минимум один узел, который может быть виртуальной машиной или облачным хостом.

- Обеспечьте наличие как минимум одного Master узла (контрольного узла) в одной из трёх зон доступности.
- Обеспечьте равномерное распределение не менее 4 вычислительных узлов по 2 зонам data availability, с минимум 2 вычислительными узлами в каждой зоне data availability.
- Старайтесь обеспечить одинаковое количество узлов и конфигурации дисков в двух зонах data availability.

Процедура

1 Тегирование узлов

1. Зайдите в **Administrator**.
2. В левой навигационной панели выберите **Cluster Management** > **Cluster**.
3. Нажмите на имя нужного кластера, чтобы перейти на страницу обзора кластера.
4. Перейдите на вкладку **Nodes**.
5. В соответствии с планированием в разделе Предварительные условия добавьте метку `topology.kubernetes.io/zone=<zone>` этим узлам для классификации их в указанную зону доступности. Здесь вместо <zone> укажите имя зоны доступности.

2 Создание сервиса хранения

В этом документе описаны только параметры, отличающиеся от стандартных кластеров; по остальным параметрам обращайтесь к Созданию стандартного кластера.

Создание кластера

Параметр	Описание
Тип кластера	Выберите **Stretch**.
Quorum Availability Zone	Выберите имя зоны quorum availability.

Параметр	Описание
Data Availability Zone	Выберите имена зон доступности и укажите узлы.

Создание пула хранения

Параметр	Описание
Количество реплик	По умолчанию 4.
Количество экземпляров	При типе хранения **Object Storage** для обеспечения доступности минимальное количество экземпляров — 2, максимальное — 5.

Связанные операции

Создание стандартного кластера

Подробности смотрите в Создании стандартного кластера.

Очистка распределённого хранилища

Подробности смотрите в Очистке распределённого хранилища.

Основные концепции

Основные концепции

Rook Operator

Ceph CSI

Функции модулей Ceph

Основные концепции

Содержание

Rook Operator

Оператор Rook — это простой контейнер, который содержит всё необходимое для инициализации и мониторинга кластера хранения данных. Оператор запускает и контролирует поды Ceph monitor, демоны Ceph OSD для предоставления хранилища RADOS, а также запускает и управляет другими демонами Ceph. Оператор управляет CRD для пулов, объектных хранилищ (S3/Swift) и файловых систем, инициализируя поды и другие ресурсы, необходимые для работы сервисов.

Оператор следит за демонами хранения, чтобы обеспечить здоровье кластера. Ceph mons запускаются или переключаются при необходимости, а также вносятся другие корректировки по мере роста или уменьшения кластера. Оператор также отслеживает изменения желаемого состояния, указанные в пользовательских ресурсах Ceph (CR), и применяет эти изменения.

Rook автоматически настраивает драйвер Ceph-CSI для монтирования хранилища к вашим подам. Образ rook/ceph включает все необходимые инструменты для управления

кластером.

Ceph CSI

Плагины Ceph CSI реализуют интерфейс между CSI-совместимым оркестратором контейнеров (CO) и кластерами Ceph. Они обеспечивают динамическое выделение томов Ceph и их подключение к рабочим нагрузкам.

Функции модулей Ceph

Модуль	Функция
MON	Монитор (MON) — самый важный компонент в кластере Ceph. Он управляет кластером Ceph и поддерживает статус всего кластера. MON обеспечивает синхронизацию связанных компонентов кластера одновременно. Он выполняет роль лидера кластера и отвечает за сбор, обновление и публикацию информации о кластере.
MGR	Менеджер (MGR) — это система мониторинга, которая обеспечивает сбор, хранение, анализ (включая оповещения) и визуализацию данных. Он делает определённые параметры кластера доступными для внешних систем.
OSD	Демоны объектного хранилища (OSD) хранят фактические пользовательские данные. Каждый OSD обычно привязан к одному физическому диску. OSD обрабатывают запросы на чтение/запись от клиентов.
MDS	Ceph Metadata Server (MDS) отслеживает иерархию файлов и хранит метаданные, используемые только для CephFS. RBD и RGW не требуют метаданных. MDS не предоставляет клиентам прямые сервисы данных.
RGW	RADOS gateway (RGW) — это шлюз объектов Ceph, который предоставляет RESTful API, совместимые с S3 и Swift. RGW также поддерживает мультиарендность и сервис идентификации OpenStack (Keystone).
RADOS	Reliable Autonomic Distributed Object Store (RADOS) — ядро кластера хранения Ceph. Всё в Ceph хранится с помощью RADOS в виде объектов независимо от типа данных. Слой RADOS обеспечивает согласованность и

Модуль	Функция
	надёжность данных через репликацию, обнаружение и восстановление ошибок, а также восстановление данных между узлами кластера.
LIBRADOS	Librados — это метод, упрощающий доступ к RADOS. В настоящее время поддерживаются языки программирования PHP, Ruby, Java, Python, C и C++. Он предоставляет локальный интерфейс RADOS для кластера Ceph и является базовым компонентом других сервисов, таких как RADOS block device (RBD) и RADOS gateway (RGW). Кроме того, он предоставляет Portable Operating System Interface (POSIX) для файловой системы Ceph (CephFS). API Librados можно использовать для прямого доступа к RADOS, что позволяет разработчикам создавать собственные интерфейсы для доступа к хранилищу кластера Ceph.
RBD	RADOS block device (RBD) — это блочное устройство Ceph, которое предоставляет блочное хранилище для внешних систем. Его можно отображать, форматировать и монтировать как диск на сервере.
CephFS	CephFS предоставляет распределённую файловую систему, совместимую с POSIX, любого размера. Она зависит от Ceph MDS для отслеживания иерархии файлов, то есть метаданных.

Руководства

Доступ к сервисам хранения

- Предварительные требования
- Процедура
- Последующие действия

Управление Storage Pools

- Создание Storage Pool
- Удаление Storage Pool
- Просмотр адресов Object Storage Pool

Развертыв

- Обновление ко
- Перезапуск ко

Добавление устройств/классс

- Добавление классов устройств
- Добавление устройств
- Статус жесткого диска

Мониторинг и оповещения

- Мониторинг
- Оповещения

Доступ к сервисам хранения

Доступ к сервисам хранения поддерживает два метода интеграции: первый — интеграция распределённых ресурсов хранения из других бизнес-кластеров внутри платформы для обеспечения изоляции хранения и бизнеса, что облегчает управление и обслуживание; второй — подключение внешних ресурсов Ceph для использования распределённого хранения.

Содержание

Предварительные требования

Подготовка пакета

- **Скачайте** установочный пакет **Alauda Container Platform Storage Essentials**, соответствующий архитектуре вашей платформы.
- **Загрузите** установочный пакет **Alauda Container Platform Storage Essentials** с помощью механизма Upload Packages.
- **Скачайте** установочный пакет **Alauda Build of Rook-Ceph**, соответствующий архитектуре вашей платформы.
- **Загрузите** установочный пакет **Alauda Build of Rook-Ceph** с помощью механизма Upload Packages.

Подготовка хранилища

Выберите один из вариантов:

- В других бизнес-кластерах уже развернуто распределённое хранилище и создан storage pool. Запишите название storage pool для последующей интеграции.
- Вне платформы создано внешнее хранилище Ceph (версия ≥ 14.2.3) с созданным storage pool. Запишите название storage pool для последующей интеграции.

Открытие портов

IP назначения	Порты назначения	IP источника	Порт источника
IP узла Ceph	3300, 6789, 6800-7300, 7480	IP всех узлов бизнес-кластера	любой

Получение данных аутентификации (внешний Ceph)

Если подготовленное хранилище — внешний Ceph, необходимо получить данные аутентификации с помощью следующих команд.

Параметр	Способ получения
FSID	`ceph fsid`
Информация о компоненте MON	`ceph mon dump` Должна быть в формате {name= IP}, например *a=192.168.100.100:6789* .
Административный ключ	`ceph auth get-key client.admin`
Storage Pool	• Файловое хранилище: используйте команду `ceph fs ls` для получения значения `name`. • Блочное хранилище: `ceph osd dump` | `grep "application rbd"` | `awk '{print $3}'`
Storage Pool для данных	(только для файлового хранилища) используйте команду `ceph fs ls` для получения значения `data pools`.

Процедура

Примечание: В следующих шагах в качестве примера рассматривается **доступ к внешнему Ceph-хранилищу**, операции для доступа к распределённому хранилищу аналогичны.

1 Развертывание Alauda Container Platform Storage Essentials

1. Войдите в систему, перейдите на страницу **Administrator**.
2. Нажмите **Marketplace** > **OperatorHub**, чтобы перейти на страницу **OperatorHub**.
3. Найдите **Alauda Container Platform Storage Essentials**, нажмите **Install** и перейдите на страницу **Install Alauda Container Platform Storage Essentials**. Параметры конфигурации:

Параметр	Рекомендуемая конфигурация
Channel	Канал по умолчанию — `stable`.
Installation Mode	`Cluster`: Все пространства имён в кластере используют один экземпляр Operator для создания и управления, что снижает использование ресурсов.
Installation Place	Выберите `Recommended`, Namespace поддерживается только **acp-storage**.
Upgrade Strategy	`Manual`: При появлении новой версии в Operator Hub требуется ручное подтверждение для обновления Operator до последней версии.

2 Доступ к хранилищу

1. В левой навигационной панели нажмите **Storage Management** > **Distributed Storage**.
2. Нажмите **Access Storage**.
3. На странице мастера **Access Configuration** выберите **External Ceph**.

Параметр	Описание
Snapshot	При включении поддерживается создание снимков PVC и использование снимков для настройки новых PVC, что обеспечивает быстрое резервное копирование и восстановление бизнес-данных. Если при доступе к хранилищу снимки не были включены, их можно включить позже в разделе **Operations** на странице деталей кластера хранения по мере необходимости. **Примечание**: Перед использованием убедитесь, что для текущего кластера развернут плагин volume snapshot.

Параметр	Описание
Network Configuration	• **Host Network**: Вычислительные компоненты в этом кластере будут обращаться к **кластеру хранения** через **host network**. • **Container Network**: Вычислительные компоненты в этом кластере будут обращаться к **кластеру хранения** через **container network**. Можно создать подсеть в управлении сетью и назначить её пространству имён `rook-ceph`. Если оставить пустым, будет использоваться подсеть по умолчанию.
Другие параметры	Заполните параметры аутентификации для внешнего Ceph, полученные на этапе предварительных требований.

4. На странице мастера **Create Storage Class** завершите конфигурацию и нажмите **Access**.

Параметр	Описание
Type	В зависимости от типа созданного выше storage pool, по умолчанию будет соответствующий класс хранения: • Файловое хранилище: CephFS File Storage • Блочное хранилище: CephRBD Block Storage
Reclaim Policy	Политика восстановления для persistent volumes. • Delete: при удалении persistent volume claim будет также удалён связанный persistent volume. • Retain: даже при удалении persistent volume claim связанный persistent volume сохраняется.

Параметр	Описание
Project Allocation	Проекты, которые могут использовать этот тип хранилища. Если в данный момент нет проектов, требующих этот тип хранилища, можно не выделять проекты сейчас и обновить их позже.

5. Дождитесь успешной интеграции, это займет примерно 1-5 минут.

Последующие действия

- Создание классов хранения: CephFS File Storage, CephRBD Block Storage
- Разработчики, использующие вышеуказанные классы хранения для создания persistent volume claims, могут расширить функционал с помощью снимков томов и масштабирования.

Примечание: Если необходимо поддерживать storage pools, конфигурации устройств хранения и т.п. для внешнего хранилища, операции должны выполняться в управляющей платформе кластера хранения.

Управление Storage Pools

Storage pool — это логический раздел, используемый для хранения данных. Один кластер хранения поддерживает одновременное использование различных типов storage pools, таких как файловое хранилище и блочное хранилище, чтобы удовлетворить разные бизнес-требования.

Содержание

Создание Storage Pool

Помимо storage pools, созданных при настройке распределённого хранилища, вы можете создавать дополнительные типы storage pools.

Совет: В рамках одного кластера хранения допускается только один файловый storage pool и один объектный storage pool, при этом можно создать до восьми блочных storage pools.

Порядок действий

1. Перейдите в **Administrator**.
2. В левой навигационной панели выберите **Storage Management** > **Distributed Storage**.
3. На вкладке **Cluster Information** прокрутите вниз до области **Storage Pool** и нажмите **Create Storage Pool**.
4. Настройте соответствующие параметры согласно следующим инструкциям.

Параметр	Описание
Storage Type	Выберите тип хранилища, который ещё не развернут. - File Storage: Обеспечивает безопасные, надёжные и масштабируемые услуги совместного файлового хранения. Подходит для совместного использования файлов, резервного копирования данных и т.д. - Block Storage: Обеспечивает хранилище с высокой производительностью IOPS и низкой задержкой. Подходит для баз данных, виртуализации и т.п. - Object Storage: Обеспечивает хранилище с интерфейсом S3, подходит для больших данных, архивного резервного копирования, облачных сервисов и т.д.
Replica Count	• **Для кластера типа Standard**: Большое количество реплик повышает отказоустойчивость и безопасность данных, но снижает эффективность использования хранилища. Обычно достаточно значения 3. • **Для кластера типа Extended**: Значение реплик по умолчанию — 4, изменить нельзя.
Device Class	• **Для кластера типа Standard**: Выберите уже добавленный класс устройств в созданном storage pool.

Параметр	Описание
	• При выборе класса устройств данные будут храниться в выбранном классе устройств. • Если класс устройств не выбран, данные будут случайным образом распределены по всем устройствам в storage pool. • **Для кластера типа Extended**: Добавление класса устройств не поддерживается.

Если выбран тип object storage, можно также настроить следующие параметры:

Параметр	Описание
Region	Укажите регион, в котором расположен storage pool.
Gateway Type	По умолчанию S3, изменить нельзя.
Internal Port	Укажите порт для внутреннего доступа к кластеру.
External Access	Включение/отключение внешнего доступа создаст/удалит сервис типа NodePort.
Instance Count	Количество экземпляров ресурсов для object storage.

5. Нажмите **Create**.

Удаление Storage Pool

Если определённый тип хранилища больше не нужен, storage pool можно удалить после отвязки от storage class.

Порядок действий

1. Перейдите в **Administrator**.

2. В левой навигационной панели выберите **Storage Management** > **Distributed Storage**.
3. На вкладке **Cluster Information** прокрутите вниз до области **Storage Pool**, нажмите ⋮ рядом с нужным storage pool и выберите **Delete**.
4. Ознакомьтесь с информацией в подсказке и введите имя storage pool.
5. Нажмите **Delete**.

Просмотр адресов Object Storage Pool

После создания object storage pool вы можете просмотреть адреса внутреннего и внешнего доступа к storage pool.

Порядок действий

1. Перейдите в **Administrator**.
2. В левой навигационной панели выберите **Storage Management** > **Distributed Storage**.
3. На вкладке **Cluster Information** прокрутите вниз до области **Storage Pool**, нажмите ⋮ рядом с object storage pool и выберите **View Address**.

Развертывание компонентов на конкретных узлах

После создания распределённого хранилища вы можете просматривать и изменять место развертывания компонентов, что облегчает расширение и обслуживание хранилища.

Содержание

Обновление конфигурации развертывания компонентов

Меры предосторожности

- Обновление конфигурации вызовет автоматическую пересборку экземпляров компонентов системой, что может повлиять на доступ к сервисам хранилища. Рекомендуется выполнять обновление в непиковое время.
- При типе кластера **Extend** функция фиксированного развертывания компонентов не поддерживается.

Порядок действий

1. Перейдите в раздел **Administrator**.
2. В левой навигационной панели нажмите **Storage Management** > **Distributed Storage**.
3. На вкладке **Storage Components** нажмите **Component Deployment Configuration**.
4. Включите/отключите переключатель **Fixed Deployment** в соответствии с бизнес-требованиями и разверните компоненты на указанных узлах. Количество узлов должно быть не менее трёх для обеспечения минимальной доступности. Компоненты, для которых доступна настройка фиксированного развертывания, включают MON, MGR, MDS, RGW.
5. Нажмите **Update**, после чего компоненты начнут планироваться на указанные узлы.

Перезапуск компонентов хранилища

При удалении развернутых компонентов хранилища система автоматически переназначит и повторно развернёт компоненты на узлах в соответствии с текущей стратегией развертывания компонентов.

Порядок действий

1. Перейдите в раздел **Administrator**.
2. В левой навигационной панели нажмите **Storage Management** > **Distributed Storage**.

3. На вкладке **Storage Components** нажмите ⋮ рядом с названием компонента > **Delete**.

Добавление устройств/классов устройств

Содержание

Добавление классов устройств

Объедините классификацию устройств одного типа или жестких дисков с одинаковой бизнес-логикой в узлах кластера, настройте классы устройств в соответствии с требованиями хранения и распределите различные содержимые хранилища по разным типам дисков.

Примечания

Добавление классов устройств не поддерживается, если тип кластера — **Extend**.

Процедура

1. Войдите в **Administrator**.
2. В левой навигационной панели нажмите **Storage Management** > **Distributed Storage**.
3. Перейдите на вкладку **Device Classes**.
4. Нажмите **Add Device Class** и настройте соответствующие параметры согласно следующим инструкциям.

Параметр	Описание
Name	Имя класса устройств. Следующие имена нельзя использовать для класса устройств: `hdd`, `ssd`, `nvme`.

Storage Devices	Для добавления устройств хранения в класс устройств можно выбрать методы `Selection Device` и `Input Device`:
	• **Selection Device**:
	Выберите из доступных устройств хранения. Устройство считается доступным, если оно соответствует следующим критериям:
	  • Тип устройства — disk или mpath
	  • Файловая система не обнаружена (fsType пустой)
	  • Емкость превышает 10 GiB
	Устройства, такие как rbd, nbd и dm-*, не будут отображаться в списке доступных для выбора устройств.
	Примечание: Требуется предварительное развертывание Alauda Build LocalStorage Operator.
	• **Input Device**:
	Вручную введите имена пустых устройств на узле, например `sda`.

Параметр	Описание
	Примечание: Для оптимальной производительности и управления настоятельно рекомендуется использовать необработанные диски в качестве устройств хранения вместо отдельных разделов на диске.

Добавление устройств

Отобразите доступные жесткие диски как устройства хранения для использования и управления.

Примечание: После добавления жестких дисков в качестве устройств хранения обновление или удаление через интерфейс не поддерживается.

Процедура

1. Войдите в **Administrator**.
2. В левой навигационной панели нажмите **Storage Management** > **Distributed Storage**.
3. Перейдите на вкладку **Device Classes**.
4. Справа от класса устройств нажмите **Add Device** и настройте соответствующие параметры согласно следующим инструкциям.

Параметр	Описание
Specified Disks	Для добавления устройств хранения в класс устройств можно выбрать методы `Selection Device` и `Input Device`: • **Selection Device**: Выберите из доступных устройств хранения. Устройство считается доступным, если оно соответствует следующим критериям: • Тип устройства — disk или mpath

Параметр	Описание
	• Файловая система не обнаружена (fsType пустой) • Емкость превышает 10 GiB Устройства, такие как rbd, nbd и dm-*, не будут отображаться в списке доступных для выбора устройств. **Примечание**: Требуется предварительное развертывание Alauda Build LocalStorage Operator. • **Input Device**: Вручную введите имена пустых устройств на узле, например `sda`. **Примечание**: Для оптимальной производительности и управления настоятельно рекомендуется использовать необработанные диски в качестве устройств хранения вместо отдельных разделов на диске.

5. Нажмите **Add**.

Статус жесткого диска

- **Normal**: Соответствующий статус устройства хранения — IN+UP.
- **Abnormal**: Соответствующий статус устройства хранения — IN+DOWN.
- **Offline**: Соответствующий статус устройства хранения — OUT+UP.
- **Fault**: Соответствующий статус устройства хранения — OUT+DOWN.

Мониторинг и оповещения

Распределённое хранилище предоставляет встроенные возможности сбора метрик мониторинга и уведомлений об оповещениях. После включения функций мониторинга и оповещений вы можете отслеживать и получать оповещения по таким аспектам, как кластер хранения, производительность хранилища и компоненты хранилища, с поддержкой настройки стратегий уведомлений.

Интуитивно представленные данные мониторинга могут использоваться для поддержки принятия решений при проведении инспекций эксплуатации и обслуживания или настройке производительности, а комплексный механизм оповещений и уведомлений поможет обеспечить стабильную работу системы хранения.

Совет: Если функции мониторинга и оповещений не были включены при создании распределённого хранилища, вам потребуется найти альтернативные решения для мониторинга и оповещений хранилища. Например, вручную настроить панели мониторинга и стратегии оповещений в центре эксплуатации и обслуживания.

Содержание

Анализ неисправностей

Мониторинг

Платформа автоматически собирает общие метрики мониторинга для распределённого хранилища, такие как производительность чтения и записи, использование CPU и памяти. В разделе **Storage Management** > **Distributed Storage** на вкладке **Monitoring** вы можете просматривать данные мониторинга в реальном времени по этим метрикам.

Обзор хранилища

Отслеживайте состояние здоровья хранилища, использование физической ёмкости и количество активных компонентов OSD/MON. В случае аномального состояния хранилища вы можете проверить причину оповещения.

Мониторинг производительности

Отслеживайте пропускную способность чтения и записи, а также IOPS чтения и записи с трёх уровней: кластер, пул хранения и OSD. Кроме того, можно мониторить задержки чтения и записи специально для OSD.

Мониторинг компонентов

Отслеживайте использование CPU и памяти таких компонентов, как MON и OSD.

Оповещения

В платформе включён набор стандартных стратегий оповещений. Как только ресурс становится аномальным или данные мониторинга достигают состояния предупреждения, оповещения автоматически срабатывают. Предустановленные стратегии достаточно для типичных операционных задач, таких как оповещения о

состоянии компонентов и кластера, оповещения о ёмкости устройств и оповещения по пользовательским данным.

Настройка уведомлений

Для своевременного получения оповещений рекомендуется настроить стратегии уведомлений в центре эксплуатации и обслуживания: отправлять информацию об оповещениях по электронной почте, SMS и другим каналам соответствующим сотрудникам, напоминая им принять необходимые меры для устранения проблем или предотвращения сбоев. Нажмите **Alert Configuration**, чтобы перейти в центр эксплуатации и обслуживания для завершения настройки, см. Create Alert Strategies。

Обработка оповещений

- Если мониторинг кластера хранения показывает состояние `Warning`, это означает, что сработало оповещение, и связанная аномалия может привести к сбою. Пожалуйста, своевременно проверьте детали в разделе **Real-time Alerts** и выявите и устраните неисправность на основе причины.
- Если мониторинг кластера хранения показывает состояние `Failure`, это указывает на то, что кластер хранения не может работать нормально. Немедленно локализуйте проблему и приступайте к устранению неисправности.

В таблице ниже приведены значения уровней оповещений, используемых в предустановленных стратегиях, которые могут служить вам ориентиром при выработке принципов обработки оповещений.

Уровень оповещения	Значение
Disaster	Ресурс, соответствующий правилу оповещения, вышел из строя, что вызывает прерывание работы платформы, потерю данных и значительное влияние.
Severe	Ресурс, соответствующий правилу оповещения, имеет известные проблемы, которые могут привести к сбоям функций платформы и повлиять на нормальную работу.

Уровень оповещения	Значение
Warning	Ресурс, соответствующий правилу оповещения, сталкивается с операционными рисками, которые могут повлиять на нормальную работу сервисов при отсутствии действий.

Анализ неисправностей

В разделе **Alert History** фиксируются все оповещения, которые были сработаны и больше не требуют действий. При проведении анализа неисправностей с использованием истории оповещений для эффективного подведения итогов вам может потребоваться ответить на следующие вопросы.

- Каковы были конкретные аномальные условия во время инцидента.
- Есть ли закономерность в повторяющемся оповещении, можно ли предотвратить его появление в следующий раз.
- Показывает ли временная шкала всплеск оповещений в определённый период; был ли он вызван форс-мажором или операционной ошибкой, требуется ли корректировка плана эксплуатации.

Как сделать

Замена или удаление устройств

Замена или удаление устройств

Предварительные требования

Ограничения и особенности

Процедура

References

Замена или удаление узлов хранения

Замена или удаление узлов хранения

Предварительные условия

Ограничения и особенности

Процедура

References

Настройка выделенного кластера для распределённого хранилища

Очистка распределённого хранилища

Восстановление после сбоев

Переключение при сбое

Восстановление после сбоев

Процедуры

Переключение

Возврат к осно

Обновление параметров оптимизации

Обновление параметров оптимизации

Процедура

Создание пользователя Ceph Object Store

Создание пользователя Ceph Object Store

Предварительные требования

Процедура

Настройка квот для Storage Pool

Настройка квот для Storage Pool

Предварительные требования

Включение кэша D3N для Ceph RGW

Настройка шифрования данных в пути

Как включить или отключить шифрование данных в пути на основе msgr2 для распределённого хранилища ACP.

Замена или удаление устройств

В этом документе описывается, как удалить устройство хранения (диск) из кластера Rook-Ceph, управляемого Container Platform. В зависимости от того, достаточно ли оставшихся OSD для размещения данных с удаляемого диска, возможно, потребуется сначала добавить заменяющий диск.

Содержание

Предварительные требования

- Все компоненты кластера работают корректно.
- Кластер хранения **не был** создан с опцией "add all empty disks". Проверьте, выполнив следующую команду; в выводе должно быть `useAllDevices: false`.

```
kubectl get cephcluster -n rook-ceph ceph-cluster -o yaml | grep useAll
Devices
```

- Применимо для версии платформы 3.8 и выше.

Ограничения и особенности

- Во время перераспределения данных производительность кластера может временно снижаться. Избегайте одновременной работы с несколькими дисками, если это не абсолютно необходимо.
- Не продолжайте, если кластер находится в состоянии `HEALTH_ERR` по причинам, **отличным от** удаления диска. Продолжение в таком состоянии может ухудшить устойчивость данных.
- Если удаляемый диск является последним диском определённого класса устройств, этот класс устройств перестанет существовать. Все пулы хранения или политики, зависящие от него, будут затронуты. Убедитесь, что нет пулов, привязанных исключительно к этому классу устройств, прежде чем продолжать.

Процедура

1 Проверка состояния кластера и ёмкости

1. Проверьте общее состояние кластера.

```
kubectl -n rook-ceph exec -it deploy/rook-ceph-tools -- ceph -s
```

2. Определите ID OSD и использование диска, который нужно удалить.

```
kubectl -n rook-ceph exec -it deploy/rook-ceph-tools -- ceph osd d
f
```

Обратите внимание на значение **USE** целевого OSD. Затем убедитесь, что сумма значений **AVAIL** по всем оставшимся OSD (кроме целевого) больше, чем значение **USE** целевого OSD. Это гарантирует, что оставшиеся OSD имеют достаточно свободного места для размещения данных после удаления.
Если оставшейся ёмкости недостаточно, перейдите к следующему шагу для добавления заменяющего диска. В противном случае перейдите к разделу Scale Down the Rook Operator.

2 Добавление заменяющего диска (если необходимо)

Если оставшиеся OSD не имеют достаточной свободной ёмкости, добавьте заменяющий диск перед удалением старого. Оператор Rook должен работать на этом этапе.

1. Войдите в **Container Platform**.
2. В левой навигационной панели выберите **Storage Management** > **Distributed Storage** > **Device Classes**.
3. Нажмите **Add Device**, выберите узел, на котором установлен заменяющий диск, выберите новый диск и назначьте его тому же классу устройств, что и удаляемый диск.
4. Дождитесь создания нового OSD и завершения перераспределения данных. Отслеживайте прогресс:

   ```
   kubectl -n rook-ceph exec -it deploy/rook-ceph-tools -- ceph -s
   ```

   Ожидайте, пока в выводе не появится `HEALTH_OK` без misplaced или recovering PG.

3 Scale Down the Rook Operator

Уменьшите масштаб оператора Rook, чтобы предотвратить его вмешательство в процесс удаления (например, чтобы он не восстанавливал удалённые развертывания OSD во время процедуры).

```
kubectl -n rook-ceph scale deploy rook-ceph-operator --replicas=0
```

4 Пометить OSD как out и дождаться миграции данных

1. Включите pod rook-ceph-tools, если он ещё не запущен.

   ```
   kubectl -n rook-ceph scale deploy rook-ceph-tools --replicas=1
   ```

2. Войдите в pod tools.

   ```
   kubectl -n rook-ceph exec -it deploy/rook-ceph-tools -- bash
   ```

3. Пометьте OSD как `out`. Это укажет Ceph мигрировать все данные с этого OSD на оставшиеся OSD.

   ```
   ceph osd out osd.<id>
   ```

4. Отслеживайте прогресс перераспределения, пока кластер не вернётся в состояние `HEALTH_OK` без misplaced или recovering PG.

   ```
   ceph -s
   ```

   Не продолжайте, пока миграция данных полностью не завершится. Удаление OSD до окончания миграции приведёт к потере данных.

5 Удаление OSD

1. Отредактируйте ресурс CephCluster, чтобы удалить запись о диске.

   ```
   kubectl edit cephcluster -n rook-ceph ceph-cluster
   ```

   Найдите диск в `spec.storage.nodes` и удалите его запись. Сохраните и выйдите.

2. Удалите развертывание OSD.

```
kubectl -n rook-ceph delete deploy rook-ceph-osd-<osd-id>
```

3. Войдите в pod tools и окончательно удалите OSD из кластера. Замените `<id>` на ID OSD.

```
kubectl -n rook-ceph exec -it deploy/rook-ceph-tools -- bash
```

Внутри pod tools:

```
ceph osd purge osd.<id> --yes-i-really-mean-it
```

6 Очистка диска

Если диск останется физически подключённым к узлу, очистите его метаданные, чтобы Rook случайно не обнаружил его. Выполните следующие команды **на узле**, где расположен диск. Замените `/dev/vdb` на фактический путь устройства.

```
# Удалите любые записи device-mapper, оставшиеся Ceph
dmsetup remove /dev/mapper/ceph--<vg-name>   # Замените на фактическо
е имя mapper, если есть

# Очистите таблицу разделов
sgdisk --zap-all /dev/vdb

# Обнулите первые 100 МБ для удаления остатков метаданных Ceph
dd if=/dev/zero of=/dev/vdb bs=1M count=100 oflag=direct,dsync
```

7 Scale Up the Rook Operator

После восстановления здоровья кластера восстановите оператор Rook.

```
kubectl -n rook-ceph scale deploy rook-ceph-operator --replicas=1
```

8 Проверка состояния кластера

1. Убедитесь, что удалённый OSD больше не отображается в кластере.

```
kubectl -n rook-ceph exec -it deploy/rook-ceph-tools -- ceph osd t
ree
```

2. Проверьте, что кластер вернулся в здоровое состояние.

```
kubectl -n rook-ceph exec -it deploy/rook-ceph-tools -- ceph -s
```

В выводе должно быть `HEALTH_OK` со всеми PG в состоянии `active+clean`.

References

- Rook Ceph OSD Management ↗

Замена или удаление узлов хранения

В этом документе описывается, как удалить узел хранения из кластера Rook-Ceph, управляемого Container Platform. В зависимости от того, достаточно ли оставшихся OSD для размещения данных с удаляемого узла, возможно, потребуется сначала добавить узел-замену.

Содержание

Предварительные условия

- Все компоненты кластера (кроме неисправного узла, если применимо) работают корректно.
- Перед началом зафиксируйте, сколько дисков было на старом узле и к какому классу устройств относится каждый диск.

Ограничения и особенности

- В кластере Ceph из трёх узлов потеря одного узла уже снижает избыточность. Выполните процедуру как можно быстрее, чтобы минимизировать окно риска.
- Во время ребалансировки данных производительность ввода-вывода кластера может временно снизиться.
- Не продолжайте, если кластер находится в состоянии `HEALTH_ERR` по причинам, **отличным от** удаляемого узла. Продолжение в таком состоянии может ещё больше снизить устойчивость данных.

Процедура

1 Проверка состояния кластера и ёмкости

1. Определите все ID OSD, работающие на узле, который нужно удалить, и их использование дисков.

```
kubectl -n rook-ceph get pod -l app=rook-ceph-osd -o wide | grep <
old-node-name>
```

2. Проверьте общее состояние кластера.

```
kubectl -n rook-ceph exec -it deploy/rook-ceph-tools -- ceph -s
```

3. Проверьте ёмкость всех OSD.

```
kubectl -n rook-ceph exec -it deploy/rook-ceph-tools -- ceph osd d
f
```

Сложите значения **USE** всех OSD на удаляемом узле. Затем убедитесь, что сумма **AVAIL** по всем оставшимся OSD (на других узлах) больше этого значения. Это гарантирует, что оставшиеся OSD имеют достаточно свободного места для размещения данных после удаления узла.
Если оставшейся ёмкости недостаточно, перейдите к следующему шагу для добавления узла-замены. В противном случае перейдите к разделу Настройка конфигурации развертывания компонентов.

2 Добавление узла-замены (если необходимо)

Если оставшиеся OSD не имеют достаточной свободной ёмкости, добавьте узел-замену перед удалением старого.

1. Войдите в **Container Platform**.
2. Добавьте заменяющую машину как новый узел кластера с помощью функционала управления узлами платформы.
3. После присоединения узла к кластеру добавьте его как узел хранения. Перейдите в **Storage Management** > **Distributed Storage** > **Device Classes**.
4. Нажмите **Add Device**, выберите новый узел и соответствующие диски. Если на старом узле было несколько дисков разных классов устройств, повторите этот шаг для каждой комбинации диск/класс устройства, пока все диски не будут добавлены.
5. Дождитесь активации новых OSD и завершения ребалансировки данных. Отслеживайте прогресс:

```
kubectl -n rook-ceph exec -it deploy/rook-ceph-tools -- ceph -s
```

Дождитесь, пока кластер не вернётся в состояние `HEALTH_OK` без misplaced или recovering PG, прежде чем продолжать.

3 Настройка конфигурации развертывания компонентов

Демоны Ceph, управляемые Rook (MON, MGR, MDS), могут быть запущены на старом узле. Исключите старый узел из расписания компонентов, чтобы оператор переназначил их на другие узлы.

1. В **Container Platform** перейдите в **Storage Management** > **Distributed Storage** > **Storage Components** > **Component Deployment Configuration**.
2. Включите привязку к узлам и выберите только те узлы, которые должны остаться в кластере (исключая удаляемый узел).
3. Дождитесь, пока все поды MON, MGR и MDS запустятся на оставшихся узлах, прежде чем продолжать.

```
kubectl -n rook-ceph get pod -o wide
```

4 Пометить все OSD как out и дождаться миграции данных

1. Включите под rook-ceph-tools, если он ещё не запущен.

```
kubectl -n rook-ceph scale deploy rook-ceph-tools --replicas=1
```

2. Войдите в под tools.

```
kubectl -n rook-ceph exec -it deploy/rook-ceph-tools -- bash
```

3. Пометьте каждый OSD на старом узле как `out`. Это укажет Ceph мигрировать все данные с этих OSD на оставшиеся.

```
ceph osd out osd.<id>
```

   Повторите для каждого ID OSD на узле.

4. Отслеживайте прогресс ребалансировки, пока кластер не вернётся в состояние `HEALTH_OK` без misplaced или recovering PG.

```
ceph -s
```

   Не продолжайте, пока миграция данных полностью не завершится. Удаление OSD до окончания миграции приведёт к потере данных.

5 Удаление OSD старого узла

1. Отредактируйте ресурс CephCluster, чтобы удалить запись о старом узле.

```
kubectl edit cephcluster -n rook-ceph ceph-cluster
```

   Найдите старый узел в `spec.storage.nodes` и удалите всю запись узла. Сохраните и выйдите.

2. Удалите развертывание OSD для каждого OSD на старом узле.

```
kubectl -n rook-ceph delete deploy rook-ceph-osd-<osd-id>
```

   Повторите для каждого ID OSD на узле.

3. Войдите в под tools и окончательно удалите каждый OSD из кластера.

```
kubectl -n rook-ceph exec -it deploy/rook-ceph-tools -- bash
```

   Внутри пода выполните для **каждого** OSD:

```
ceph osd purge osd.<id> --yes-i-really-mean-it
```

6 Проверка состояния кластера

1. Убедитесь, что все удалённые OSD больше не отображаются в кластере.

```
kubectl -n rook-ceph exec -it deploy/rook-ceph-tools -- ceph osd t
ree
```

2. Проверьте, что кластер вернулся в здоровое состояние.

```
kubectl -n rook-ceph exec -it deploy/rook-ceph-tools -- ceph -s
```

   В выводе должно быть `HEALTH_OK` и все PG в состоянии `active+clean`.

References

- Rook Ceph OSD Management ↗

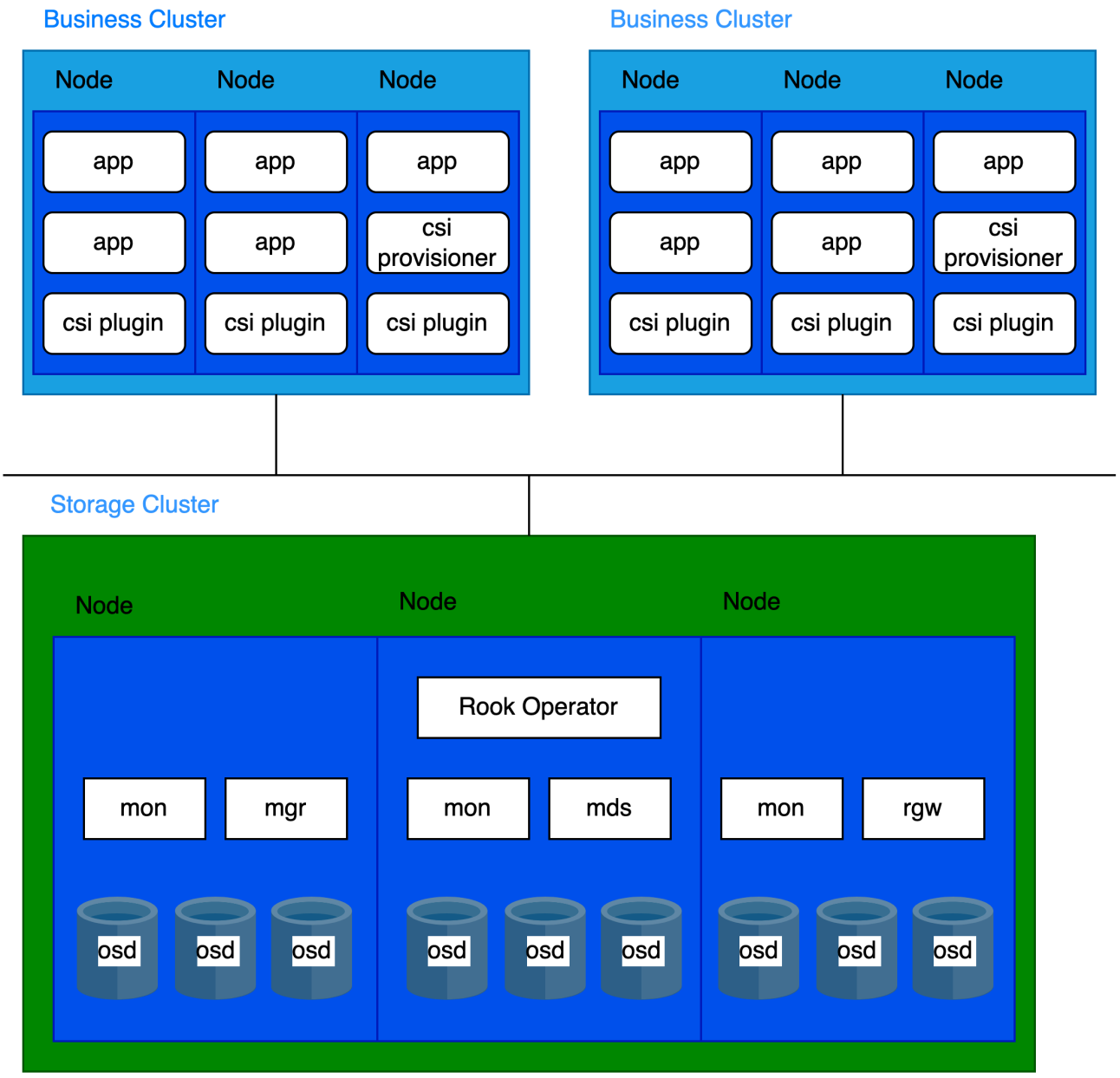
Настройка выделенного кластера для распределённого хранилища

Развёртывание выделенного кластера подразумевает использование отдельного кластера для развертывания распределённого хранилища платформы, при этом другие бизнес-кластеры внутри платформы получают доступ и используют предоставляемые им сервисы хранения через интеграцию.
Для обеспечения производительности и стабильности распределённого хранилища платформы в выделенном кластере развертываются только основные компоненты платформы и компоненты распределённого хранилища, избегая совместного размещения с другими бизнес-нагрузками. Такой раздельный подход к развертыванию является рекомендуемой лучшей практикой для распределённого хранилища платформы.

Содержание

Архитектура

Архитектура разделения хранения и вычислений



Требования к инфраструктуре

Требования к платформе

Поддерживается в версии 3.18 и выше.

Требования к кластеру

Рекомендуется использовать bare-metal кластеры в качестве выделенных кластеров хранения.

Требования к ресурсам

Пожалуйста, ознакомьтесь с Основными понятиями для компонентов развертывания распределённого хранилища.

Каждый компонент имеет свои требования к CPU и памяти. Рекомендуемые конфигурации приведены ниже:

Процесс	CPU	Память
MON	2c	3Gi
MGR	3c	4Gi
MDS	3c	8Gi
RGW	2c	4Gi
OSD	4c	8Gi

В кластере обычно запускаются:

- 3 MON
- 2 MGR
- несколько OSD
- 2 MDS (если используется CephFS)
- 2 RGW (если используется CephObjectStorage)

Исходя из распределения компонентов, применимы следующие рекомендации по ресурсам на узел:

CPU	Память
16c + (4c * OSD на узел)	20Gi + (8Gi * OSD на узел)

Требования к устройствам хранения

Рекомендуется развертывать не более 12 устройств хранения на узел. Это помогает ограничить время восстановления после сбоя узла.

Требования к типу устройств хранения

Рекомендуется использовать корпоративные SSD ёмкостью 10TiB или меньше на устройство, при этом все диски должны быть одинакового размера и типа.

Планирование ёмкости

Перед развертыванием ёмкость хранилища должна быть спланирована в соответствии с конкретными бизнес-требованиями. По умолчанию распределённая система хранения использует стратегию избыточности с 3 репликами. Следовательно, полезная ёмкость рассчитывается как общая сырая ёмкость хранения (суммарно по всем устройствам) делённая на 3.

Пример для 30(N) узлов (число реплик = 3), сценарий полезной ёмкости:

Размер устройства хранения(D)	**Устройств на узел(M)**	**Общая ёмкость(D*M*N)**	**Полезная ёмкость(D*M*N/3)**
0.5 TiB	3	45 TiB	15 TiB
2 TiB	6	360 TiB	120 TiB
4 TiB	9	1080 TiB	360 TiB

Мониторинг ёмкости и масштабирование

1. **Проактивное планирование ёмкости**
   Всегда следите, чтобы полезная ёмкость хранилища превышала текущие потребления. Если хранилище полностью исчерпано, восстановление требует ручного вмешательства и не может быть решено простым удалением или миграцией данных.
2. **Оповещения о ёмкости**
   Кластер генерирует оповещения при достижении двух порогов:

- **80% использования** ("почти заполнено"): проактивно **освободите место** или масштабируйте кластер.
- **95% использования** ("заполнено"): хранилище полностью исчерпано, стандартные команды не могут освободить место. Немедленно обратитесь в службу поддержки платформы.

Всегда оперативно реагируйте на оповещения и регулярно контролируйте использование хранилища, чтобы избежать простоев.

3. **Рекомендации по масштабированию**

- **Избегайте**: добавления устройств хранения к существующим узлам.
- **Рекомендуется**: масштабирование путём добавления новых узлов хранения.
- **Требование**: новые узлы должны использовать устройства хранения идентичного размера, типа и количества с существующими узлами.

Требования к сети

Распределённое хранилище должно использовать **HostNetwork**.

Сетевая изоляция

Сеть делится на два типа:

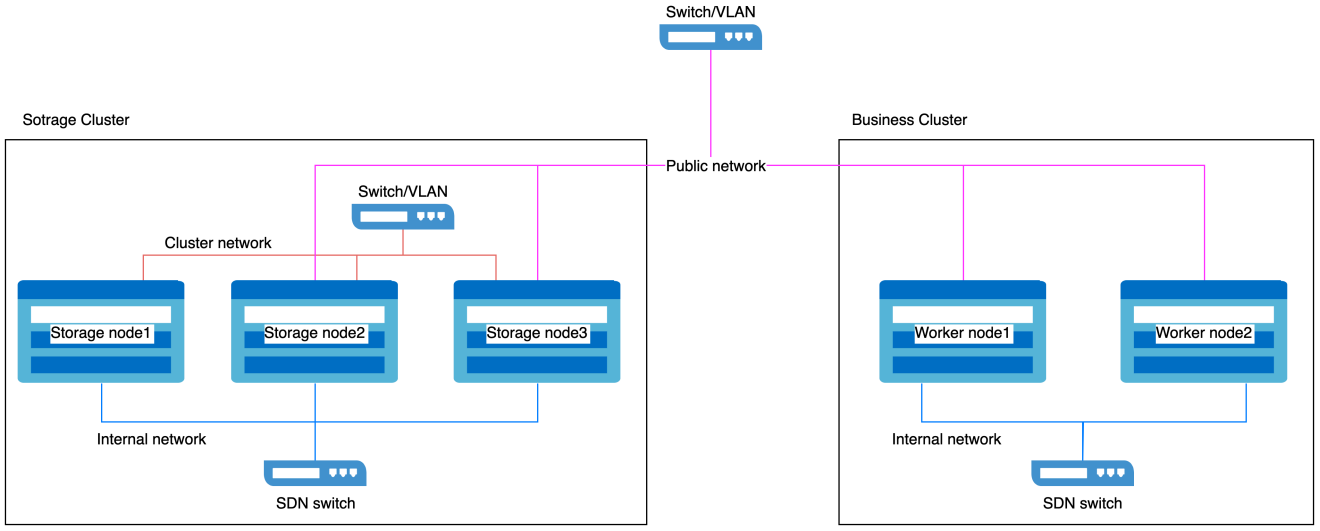
- **Публичная сеть**: используется для взаимодействия клиентов с компонентами хранения (например, I/O запросы).
- **Кластерная сеть**: выделена для репликации данных между репликами и балансировки данных (например, восстановление).

Для обеспечения качества сервиса и стабильности производительности:

1. Для выделенных кластеров хранения:
   Зарезервируйте два сетевых интерфейса на каждом хосте:
   - Публичная сеть: для связи клиентов и компонентов.
   - Кластерная сеть: для внутреннего трафика репликации и балансировки.
2. Для бизнес-кластеров:
   Зарезервируйте один сетевой интерфейс на каждом хосте для доступа к публичной

сети хранения.

Пример конфигурации сетевой изоляции



Требования к скорости сетевых интерфейсов

1. **Узлы хранения**

   - Сетевые интерфейсы для **Публичной сети** и **Кластерной сети** должны быть 10GbE или выше.

2. **Узлы бизнес-кластеров**

   - Сетевой интерфейс, используемый для доступа к публичной сети хранения, должен быть 10GbE или выше.

Процедура

1 Развертывание Operator

1. Зайдите в **Administrator**.
2. В левой боковой панели нажмите **Storage Management** > **Distributed Storage**.
3. Нажмите **Create Now**.
4. На странице мастера **Deploy Operator** нажмите кнопку **Deploy Operator** в правом нижнем углу.

- Если страница автоматически перейдёт к следующему шагу, значит Operator успешно развернут.
- Если развертывание не удалось, следуйте подсказке на интерфейсе **Clean Up Deployed Information and Retry** и повторно разверните Operator; если хотите вернуться на страницу выбора распределённого хранилища, нажмите **Application Store**, сначала удалите ресурсы уже развернутого **rook-operator**, затем удалите сам **rook-operator**.

2 Создание кластера ceph

Выполните команды на **контрольном узле** кластера хранения.

▶ Нажмите для просмотра

Параметры:

- **public network cidr**: CIDR публичной сети хранения (например, `-10.0.1.0/24`).
- **cluster network cidr**: CIDR кластерной сети хранения (например, `-10.0.2.0/24`).
- **storage devices**: Укажите устройства хранения, которые будут использоваться распределённым хранилищем.
  Пример форматирования:

```
nodes:
- name: storage-node-01
  devices:
  - name: /dev/disk/by-id/wwn-0x5000cca01dd27d60
  useAllDevices: false
- name: storage-node-02
  devices:
  - name: sdb
  - name: sdc
  useAllDevices: false
- name: storage-node-03
  devices:
  - name: sdb
  - name: sdc
  useAllDevices: false
```

Совет

Используется World Wide Name (WWN) диска для стабильного именования, что позволяет избежать зависимости от нестабильных путей устройств, таких как `sdb`, которые могут изменяться после перезагрузок.

3 Создание пулов хранения

Доступны три типа пулов хранения. Выберите и создайте подходящие в соответствии с вашими бизнес-требованиями.

Создание файлового пула

Выполните команды на **контрольном узле** кластера хранения.

▶ Нажмите для просмотра

Создание блочного пула

Выполните команды на **контрольном узле** кластера хранения.

▶ Нажмите для просмотра

Создание объектного пула

Выполните команды на **контрольном узле** кластера хранения.

▶ Нажмите для просмотра

Последующие действия

Когда другие кластеры нуждаются в использовании сервиса распределённого хранилища, руководствуйтесь следующими рекомендациями.
Доступ к сервисам хранения

Очистка распределённого хранилища

Если необходимо удалить кластер rook-ceph и развернуть новый, следует последовательно очистить ресурсы, связанные с сервисом распределённого хранилища, согласно данной инструкции.

Содержание

Меры предосторожности

Перед очисткой rook-ceph убедитесь, что все ресурсы PVC и PV, использующие хранилище Ceph, удалены.

Процедура

1 Удаление VolumeSnapshotClasses

1. Удалите VolumeSnapshotClasses.

```
kubectl delete VolumeSnapshotClass csi-cephfs-snapshotclass csi-rbd-snapshotclass
```

2. Проверьте, что VolumeSnapshotClasses удалены.

```
kubectl get VolumeSnapshotClass | grep csi-cephfs-snapshotclass
kubectl get VolumeSnapshotClass | grep csi-rbd-snapshotclass
```

Отсутствие вывода означает, что очистка завершена.

2 Удаление StorageClasses

1. Перейдите в раздел **Administrator**.
2. В левой навигационной панели выберите **Storage Management** > **Storage Classes**.
3. Нажмите ⋮ > **Delete** и удалите все StorageClasses, использующие решения Ceph.

3 Удаление Storage Pools

Этот шаг выполняется после завершения предыдущего.

1. Перейдите в раздел **Administrator**.
2. В левой навигационной панели выберите **Storage Management** > **Distributed Storage**.
3. В области **Storage Pool Area** нажмите ⋮ > **Delete** и удалите все пулы хранения по одному. Когда в области пула хранения отображается сообщение **No Storage**

Pools, это означает успешное удаление.

4. (Опционально) Если режим кластера — **Extended**, выполните следующую команду на Master-узле кластера для удаления созданных встроенных пулов хранения.

```
kubectl -n rook-ceph delete cephblockpool -l cpaas.io/builtin=true
```

Ответ:

```
cephblockpool.ceph.rook.io "builtin-mgr" deleted
```

4 Удаление ceph-cluster

Этот шаг выполняется после завершения предыдущего.

1. Обновите ceph-cluster и включите политику очистки.

```
kubectl -n rook-ceph patch cephcluster ceph-cluster --type merge -
p '{"spec":{"cleanupPolicy":{"confirmation":"yes-really-destroy-da
ta"}}}'
```

2. Удалите ceph-cluster.

```
kubectl delete cephcluster ceph-cluster -n rook-ceph
```

3. Удалите задания, выполняющие очистку.

```
kubectl delete jobs --all -n rook-ceph
```

4. Проверьте, что очистка ceph-cluster завершена.

```
kubectl get cephcluster  -n rook-ceph | grep ceph-cluster
```

Отсутствие вывода означает, что очистка завершена.

5 Удаление rook-operator

Этот шаг выполняется после завершения предыдущего.

1. Удалите rook-operator.

```
kubectl -n rook-ceph delete subscriptions.operators.coreos.com roo
k-operator
```

2. Проверьте, что очистка rook-operator завершена.

```
kubectl get subscriptions.operators.coreos.com -n rook-ceph | grep
rook-operator
```

   Отсутствие вывода означает, что очистка завершена.

3. Проверьте, что все ConfigMaps удалены.

```
kubectl get configmap -n rook-ceph
```

   Отсутствие вывода означает, что очистка завершена. Если есть вывод, выполните следующую команду для очистки, заменив `<configmap>` на фактическое имя.

```
kubectl -n rook-ceph patch configmap <configmap> --type merge -p
'{"metadata":{"finalizers": []}}'
```

4. Проверьте, что все Secrets удалены.

```
kubectl get secret -n rook-ceph
```

   Отсутствие вывода означает, что очистка завершена. Если есть вывод, выполните следующую команду для очистки, заменив `<secret>` на фактическое имя.

```
kubectl -n rook-ceph patch secrets <secret> --type merge -p '{"met
adata":{"finalizers": []}}'
```

5. Проверьте, что очистка rook-ceph завершена.

```
kubectl get all -n rook-ceph
```

Отсутствие вывода означает, что очистка завершена.

6

Выполнение скрипта очистки

После выполнения вышеуказанных шагов Kubernetes и ресурсы Ceph будут очищены. Далее необходимо удалить остатки rook-ceph на хосте.

Скрипт очистки

Содержимое скрипта clean-rook.sh:

▶ Нажмите для просмотра

Меры предосторожности

Скрипт очистки зависит от команды sgdisk, поэтому убедитесь, что она установлена перед запуском скрипта.

- Команда установки для Ubuntu: `sudo apt install gdisk`
- Команда установки для RedHat или CentOS: `sudo yum install gdisk`

Процедура

1. Запустите скрипт очистки clean-rook.sh на каждой машине в бизнес-кластере, где развернуто распределённое хранилище.

   ```
   sh clean-rook.sh /dev/[device_name]
   ```

   Пример: `sh clean-rook.sh /dev/vdb`
   При выполнении будет запрошено подтверждение очистки устройства. Для подтверждения введите yes.

2. Используйте команду `lsblk -f` для проверки информации о разделах. Если в столбце `FSTYPE` вывод пустой, очистка завершена.

Восстановление после сбоев

Восстановление после сбоев файлового хранилища

CephFS Mirror — это функция файловой системы Ceph, предназначенная для асинхронной репликации данных между разными кластерами Ceph, обеспечивая тем самым аварийное восстановление между кластерами. Основная её задача — синхронизация данных в режиме primary-backup, что гарантирует быструю передачу управления на резервный кластер в случае сбоя основного.

WARNING

- CephFS Mirror выполняет инкрементальную синхронизацию на основе снимков (snapshots), при этом интервал создания снимков по умолчанию установлен на один раз в час (настраивается). Дифференциальные данные между основным и резервным кластерами обычно составляют объём данных, записанных за один цикл снимка.
- CephFS Mirror обеспечивает только резервное копирование данных базового хранилища и не может выполнять резервное копирование ресурсов Kubernetes. Для резервного копирования PVC и PV ресурсов используйте функцию платформы **Backup and Restore**.

Содержание

Терминология

Термин	Объяснение
Primary Cluster	Кластер, который в данный момент предоставляет услуги хранения.
Secondary Cluster	Резервный кластер.

Конфигурация резервного копирования

Предварительные условия

- Подготовьте два кластера, подходящих для развертывания Alauda Build Rook-Ceph, а именно Primary и Secondary, убедившись, что сети между кластерами взаимосвязаны.
- Версии платформы, используемые в обоих кластерах (v3.12 и выше), должны совпадать.
- Создайте распределённый сервис хранения в обоих кластерах — Primary и Secondary.
- Создайте пулы файлового хранилища с **одинаковыми именами** в обоих кластерах.

Процедура

1 Включите Mirror для пула файлового хранилища в Secondary кластере

Выполните следующие команды на управляющем узле Secondary кластера:

Command Line

```
kubectl -n rook-ceph patch cephfilesystem <fs-name> \
--type merge -p '{"spec":{"mirroring":{"enabled": true}}}'
```

Output

```
cephfilesystem.ceph.rook.io/<fs-name> patched
```

Параметры:

- `<fs-name>`: Имя пула файлового хранилища.

2 Получите Peer Token

Этот токен является ключевым учётным данным для установления зеркального соединения между двумя кластерами.

Выполните следующие команды на управляющем узле Secondary кластера:

Command

```
kubectl get secret -n rook-ceph \
$(kubectl -n rook-ceph get cephfilesystem <fs-name> -o jsonpath
='{.status.info.fsMirrorBootstrapPeerSecretName}') \
-o jsonpath='{.data.token}' | base64 -d
```

Output

```
# Из-за наличия конфиденциальной информации вывод сокращён.
eyJmc2lkIjogImMyYjAyNmMzLTA3ZGQtNDA3Z...
```

Параметры:

- `<fs-name>`: Имя пула файлового хранилища.

3 Создайте Peer Secret в Primary кластере

После получения Peer Token из Secondary кластера необходимо создать Peer Secret в Primary кластере.

Выполните следующие команды на управляющем узле Primary кластера:

Command

```
kubectl -n rook-ceph create secret generic fs-secondary-site-sec
ret \
--from-literal=token=<token> \
--from-literal=pool=<fs-name>
```

Output

```
secret/fs-secondary-site-secret created
```

Параметры:

- `<token>`: Токен, полученный на шаге 2.
- `<fs-name>`: Имя пула файлового хранилища.

4 Включите Mirror для пула файлового хранилища в Primary кластере

Выполните следующие команды на управляющем узле Primary кластера:

Command

```
kubectl -n rook-ceph patch cephfilesystem <fs-name> --type merge
-p \
'{
  "spec": {
    "mirroring": {
      "enabled": true,
      "peers": {
        "secretNames": [
          "fs-secondary-site-secret"
        ]
      },
      "snapshotSchedules": [
        {
          "path": "/",
          "interval": "<schedule-interval>"
        }
      ],
      "snapshotRetention": [
        {
          "path": "/",
          "duration": "<retention-policy>"
        }
      ]
    }
  }
}'
```

Sample

```
kubectl -n rook-ceph patch cephfilesystem cephfs --type merge -p
\
'{
  "spec": {
    "mirroring": {
      "enabled": true,
      "peers": {
        "secretNames": [
          "fs-secondary-site-secret"
        ]
      },
      "snapshotSchedules": [
        {
          "path": "/",
          "interval": "1h"
        }
      ],
      "snapshotRetention": [
        {
          "path": "/",
          "duration": "h 1"
        }
      ]
    }
  }
}'
```

Output

```
cephfilesystem.ceph.rook.io/<fs-name> patched
```

Параметры:

- `<fs-name>`: Имя пула файлового хранилища.
- `<schedule-interval>`: Интервал выполнения снимков. Подробнее см. в официальной документации↗.

- `<retention-policy>`: Политика хранения снимков. Подробнее см. в официальной документации↗.

5 Разверните Mirror Daemon в Primary кластере

Mirror Daemon непрерывно отслеживает изменения данных в пуле файлового хранилища (с включённым Mirror). Он периодически создаёт снимки и отправляет разницы снимков в Secondary кластер по сети.

Выполните следующие команды на управляющем узле Primary кластера:

Command

```
cat << EOF | kubectl apply -f -
apiVersion: ceph.rook.io/v1
kind: CephFilesystemMirror
metadata:
  name: cephfs-mirror
  namespace: rook-ceph
spec:
  placement:
    tolerations:
    - key: NoSchedule
      operator: Exists
  resources:
    limits:
      cpu: "500m"
      memory: "1Gi"
    requests:
      cpu: "500m"
      memory: "1Gi"
  priorityClassName: system-node-critical
EOF
```

Output

```
cephfilesystemmirror.ceph.rook.io/cephfs-mirror created
```

Переключение при сбое

В случае сбоя Primary кластера вы можете напрямую продолжить использование CephFS в Secondary кластере.

Предварительные условия

Ресурсы Kubernetes Primary кластера были забэкаплены и восстановлены в Secondary кластер, включая PVC, PV и рабочие нагрузки приложений.

Восстановление после сбоев блочного хранилища

RBD Mirror — это функция Ceph Block Storage (RBD), которая обеспечивает асинхронную репликацию данных между разными кластерами Ceph, предоставляя межкластерное восстановление после сбоев (Disaster Recovery, DR). Её основная задача — синхронизация данных в режиме primary-backup, обеспечивая быстрое переключение на резервный кластер при сбое основного.

WARNING

- RBD Mirror выполняет инкрементную синхронизацию на основе снимков (snapshots) с интервалом по умолчанию один раз в час (настраиваемый). Дифференциальные данные между основным и резервным кластерами обычно соответствуют записям, сделанным в течение одного цикла снимка.
- RBD Mirror обеспечивает только резервное копирование данных на уровне хранилища и не занимается резервным копированием ресурсов Kubernetes. Для резервного копирования PVC и PV используйте функцию **Backup and Restore** платформы.

Содержание

Терминология

Термин	Объяснение
Primary Cluster	Кластер, который в данный момент предоставляет услуги хранения.
Secondary Cluster	Резервный кластер, используемый для целей бэкапа.

Конфигурация резервного копирования

Предварительные требования

- Подготовьте два кластера, способных развернуть Alauda Build Rook-Ceph: Primary и Secondary.
- Оба кластера должны работать на одной версии платформы (v3.12 или выше).
- Создайте распределённые сервисы хранения в обоих кластерах — Primary и Secondary.
- Создайте пулы блочного хранилища с **одинаковыми именами** в обоих кластерах.
- Убедитесь, что следующие два образа загружены в приватный репозиторий образов платформы:
  - `quay.io/csiaddons/k8s-controller:v0.12.0` -> `<registry>/csiaddons/k8s-controller:v0.12.0`
  - `quay.io/csiaddons/k8s-sidecar:v0.12.0` -> `<registry>/csiaddons/k8s-sidecar:v0.12.0`

Сетевые требования

Синхронизация между Primary и Secondary кластерами осуществляется через Public network Ceph. Поэтому межсайтовая Public сеть должна соответствовать следующим требованиям:

- Каждый кластер должен иметь выделенный сегмент Public сети. Public сети Primary и Secondary кластеров должны быть полностью маршрутизируемы и доступны друг для друга.
- Постоянная загрузка пропускной способности Public сети не должна превышать 60%, чтобы обеспечить запас для пиковых нагрузок репликации и сценариев переключения.
- Время кругового отклика (RTT) между двумя сайтами должно быть менее 30 мс.
- Потеря пакетов между двумя Public сетями должна быть ниже 0.05% для гарантии стабильной и предсказуемой производительности репликации.

Процедуры

> ### Инициализация пиров (`Primary <-> Secondary`)

1. **Включение зеркалирования для пула блочного хранилища Primary кластера**

   Выполните следующую команду на управляющих узлах обоих кластеров — Primary и Secondary:

   Command

   ```
   kubectl -n rook-ceph patch cephblockpool <block-pool-name> \
     --type merge -p '{"spec":{"mirroring":{"enabled":true,"mod
   e":"image"}}}'
   ```

   Output

   ```
   cephblockpool.ceph.rook.io/<block-pool-name> patched
   ```

   Параметры:

   - `<block-pool-name>`: имя пула блочного хранилища.

2. **Получение токена пира**

   Этот токен является ключевым для установления зеркальных соединений между кластерами.

   Выполните следующую команду на управляющих узлах обоих кластеров:

   Command

   ```
   kubectl get secret -n rook-ceph \
     $(kubectl get cephblockpool.ceph.rook.io <block-pool-name> -
   n rook-ceph -o jsonpath='{.status.info.rbdMirrorBootstrapPeerS
   ecretName}') \
     -o jsonpath='{.data.token}' | base64 -d
   ```

   Output

```
# Вывод сокращён из-за конфиденциальности
eyJmc2lkIjoiMjc2N2I3ZmEtY2YwYi00N...
```

3. **Создание секрета с токеном пира в пировом кластере**

   Выполните следующую команду на управляющем узле каждого кластера:

   Command

   ```
   kubectl -n rook-ceph create secret generic rbd-peer-site-secre
   t \
     --from-literal=token=<token> \
     --from-literal=pool=<block-pool-name>
   ```

   Output

   ```
   secret/rbd-peer-site-secret created
   ```

   Параметры:

   - `<token>`: токен, полученный на Шаге 2.

     В Primary кластере укажите токен, полученный из Secondary кластера.

     В Secondary кластере укажите токен, полученный из Primary кластера.
   - `<block-pool-name>`: имя пула блочного хранилища.

4. **Применение патча секрета пира для пула блочного хранилища**

   Выполните следующую команду на управляющих узлах обоих кластеров:

   Command

```
kubectl -n rook-ceph patch cephblockpool <block-pool-name> --t
ype merge -p \
'{
  "spec": {
    "mirroring": {
      "peers": {
        "secretNames": [
          "rbd-peer-site-secret"
        ]
      }
    }
  }
}'
```

Output

```
cephblockpool.ceph.rook.io/<block-pool-name> patched
```

Параметры:

- `<block-pool-name>`: имя пула блочного хранилища.

5. **Развёртывание демона зеркалирования**

   Этот демон отвечает за мониторинг и управление процессами синхронизации RBD mirror, включая синхронизацию данных и обработку ошибок.

   Выполните следующую команду на управляющих узлах обоих кластеров:

Command

```
cat << EOF | kubectl apply -f -
apiVersion: ceph.rook.io/v1
kind: CephRBDMirror
metadata:
  name: rbd-mirror
  namespace: rook-ceph
spec:
  count: 1
EOF
```

Output

```
cephrbdmirror.ceph.rook.io/rbd-mirror created
```

6. **Проверка статуса зеркалирования**

   Выполните следующую команду на управляющих узлах обоих кластеров:

   Command

   ```
   kubectl get cephblockpools.ceph.rook.io <block-pool-name> -n r
   ook-ceph -o jsonpath='{.status.mirroringStatus.summary}'
   ```

   Output

   ```
   # Все статусы "OK" означают нормальную работу
   {"daemon_health":"OK","health":"OK","image_health":"OK","state
   s":{}}
   ```

   Параметры:

   - `<block-pool-name>`: имя пула блочного хранилища.

Настройка среды для репликации томов

Эта функция позволяет эффективно реплицировать и синхронизировать данные без прерывания работы основных приложений, повышая надёжность и доступность системы.

1. **Установка контроллера CsiAddons**

   Выполните следующие команды на управляющих узлах обоих кластеров:

   ```
   kubectl create -f https://raw.githubusercontent.com/csi-addons/kubernetes-csi-addons/v0.12.0/deploy/controller/crds.yaml
   kubectl create -f https://raw.githubusercontent.com/csi-addons/kubernetes-csi-addons/v0.12.0/deploy/controller/setup-controller.yaml
   kubectl create -f https://raw.githubusercontent.com/csi-addons/kubernetes-csi-addons/v0.12.0/deploy/controller/rbac.yaml
   kubectl create -f https://raw.githubusercontent.com/csi-addons/kubernetes-csi-addons/v0.12.0/deploy/controller/csi-addons-config.yaml

   kubectl -n csi-addons-system set image deployment/csi-addons-controller-manager manager=<registry>/csiaddons/k8s-controller:v0.12.0
   ```

   Параметры:

   - `<registry>`: адрес реестра платформы.

2. **Включение sidecar CsiAddons**

   Выполните следующие команды на управляющих узлах обоих кластеров:

```
kubectl patch cm rook-ceph-operator-config -n rook-ceph --type jso
n --patch \
'[
  {
    "op": "add",
    "path": "/data/CSI_ENABLE_OMAP_GENERATOR",
    "value": "true"
  },
  {
    "op": "add",
    "path": "/data/CSI_ENABLE_CSIADDONS",
    "value": "true"
  },
  {
    "op": "add",
    "path": "/data/ROOK_CSIADDONS_IMAGE",
    "value": "<registry>/csiaddons/k8s-sidecar:v0.12.0"
  }
]'
```

Дождитесь успешного перезапуска всех csi подов:

```
kubectl get po -n rook-ceph -w | grep csi
```

3. **Создание VolumeReplicationClass**

   Выполните следующие команды на управляющих узлах обоих кластеров:

   Command

```
cat << EOF | kubectl apply -f -
apiVersion: replication.storage.openshift.io/v1alpha1
kind: VolumeReplicationClass
metadata:
  name: rbd-volumereplicationclass
spec:
  provisioner: rook-ceph.rbd.csi.ceph.com
  parameters:
    mirroringMode: snapshot
    schedulingInterval: "<scheduling-interval>" 1
    replication.storage.openshift.io/replication-secret-name:
rook-csi-rbd-provisioner
    replication.storage.openshift.io/replication-secret-namesp
ace: rook-ceph
EOF
```

Output

```
volumereplicationclass.replication.storage.openshift.io/rbd-vo
lumereplicationclass created
```

1 `<scheduling-interval>`: Интервал планирования (например, schedulingInterval: "1h" означает выполнение каждые 1 час).

Включение зеркалирования для PVC

Выполните следующую команду на управляющем узле Primary кластера:

Command

```
cat << EOF | kubectl apply -f -
apiVersion: replication.storage.openshift.io/v1alpha1
kind: VolumeReplication
metadata:
  name: <vr-name> 1
  namespace: <namespace> 2
spec:
  autoResync: false
  volumeReplicationClass: rbd-volumereplicationclass
  replicationState: primary
  dataSource:
    apiGroup: ""
    kind: PersistentVolumeClaim
    name: <pvc-name> 3
EOF
```

Output

```
volumereplication.replication.storage.openshift.io/<mirror-pvc-name> created
```

1 `<vr-name>`: имя объекта VolumeReplication, рекомендуется совпадать с именем PVC.

2 `<namespace>`: namespace, к которому принадлежит VolumeReplication, должен совпадать с namespace PVC.

3 `<pvc-name>`: имя PVC, для которого нужно включить зеркалирование.

Примечание

После включения RBD образ в Secondary кластере становится доступен только для чтения.

Плановая миграция

Сценарии использования: обслуживание дата-центра, обновление технологий, предотвращение сбоев и т.п.

Перемещение (Relocation)

Операция Relocation — это процесс переключения производства на резервный объект (обычно сайт восстановления) или обратно.

Для перемещения необходимо прекратить доступ к образу на основном сайте. Затем образ должен быть назначен primary на вторичном кластере, чтобы доступ к нему возобновился там.

Предварительные требования

- Ресурсы Kubernetes основного кластера были сохранены и восстановлены во вторичном, включая PVC, PV, рабочие нагрузки приложений и т.д.

Процедуры

Выполните следующие шаги для плановой миграции рабочих нагрузок с Primary кластера на Secondary:

1. Остановите все поды приложений, использующих зеркалируемые PVC на Primary кластере.
2. Обновите VolumeReplication для всех PVC с включённым зеркалированием на Primary кластере.
   Установите `spec.replicationState` в значение `secondary`.
3. Создайте VolumeReplication для всех PVC с зеркалированием на Secondary кластере.

Example

```
cat << EOF | kubectl apply -f -
apiVersion: replication.storage.openshift.io/v1alpha1
kind: VolumeReplication
metadata:
  name: <vr-name> 1
  namespace: <namespace> 2
spec:
  autoResync: false
  volumeReplicationClass: rbd-volumereplicationclass
  replicationState: primary
  dataSource:
    apiGroup: ""
    kind: PersistentVolumeClaim
    name: <pvc-name> 3
EOF
```

1 `<vr-name>`: имя объекта VolumeReplication, рекомендуется совпадать с именем PVC.

2 `<namespace>`: namespace, к которому принадлежит VolumeReplication, должен совпадать с namespace PVC.

3 `<pvc-name>`: имя PVC, для которого нужно включить зеркалирование.

4. Проверьте статус CR VolumeReplication, чтобы убедиться, что образ помечен как `primary` на вторичном сайте.
5. После того как образ помечен как `primary`, PVC готов к использованию. Теперь можно запустить приложения, использующие этот PVC.

Восстановление после сбоев

Сценарии использования: природные катастрофы, отключения питания, системные сбои и аварии и т.п.

Переключение (Failover) — внезапное отключение

В случае аварийного восстановления создайте CR VolumeReplication на вторичном сайте.

Поскольку связь с основным сайтом потеряна, оператор автоматически отправляет GRPC-запрос драйверу для принудительного назначения dataSource как `primary` на вторичном сайте.

Предварительные требования

- Ресурсы Kubernetes основного кластера были сохранены и восстановлены во вторичном, включая PVC, PV, рабочие нагрузки приложений и т.д.

Процедуры

1. Создайте VolumeReplication для всех PVC с включённым зеркалированием на Secondary кластере.

   Example

   ```
   cat << EOF | kubectl apply -f -
   apiVersion: replication.storage.openshift.io/v1alpha1
   kind: VolumeReplication
   metadata:
     name: <vr-name> 1
     namespace: <namespace> 2
   spec:
     autoResync: false
     volumeReplicationClass: rbd-volumereplicationclass
     replicationState: primary
     dataSource:
       apiGroup: ""
       kind: PersistentVolumeClaim
       name: <pvc-name> 3
   EOF
   ```

   1 `<vr-name>`: имя объекта VolumeReplication, рекомендуется совпадать с именем PVC.

   2 `<namespace>`: namespace, к которому принадлежит VolumeReplication, должен совпадать с namespace PVC.

   3 `<pvc-name>`: имя PVC, для которого нужно включить зеркалирование.

2. Проверьте статус CR VolumeReplication, чтобы убедиться, что образ помечен как `primary` на вторичном сайте.
3. После того как образ помечен как `primary`, PVC готов к использованию. Теперь можно запустить приложения, использующие этот PVC.

Возврат (Failback) — восстановление после аварии

После восстановления сбойного кластера на основном сайте и необходимости возврата с вторичного сайта выполните следующие шаги:

Предварительные требования

- Ресурсы Kubernetes основного кластера были сохранены и восстановлены во вторичном, включая PVC, PV, рабочие нагрузки приложений и т.д.

Процедуры

1. Остановите запущенные приложения (если есть) на основном сайте. Убедитесь, что все используемые рабочей нагрузкой постоянные тома больше не используются на Primary кластере.
2. Обновите CR VolumeReplication, изменив replicationState с primary на secondary на основном сайте.
3. Остановите приложения на вторичном сайте.
4. Обновите CR VolumeReplication, изменив replicationState с `primary` на `secondary` на вторичном сайте.
5. На основном сайте проверьте, что статус VolumeReplication помечен как готовый к использованию том.
6. После того как том помечен как готовый к использованию, измените replicationState с secondary на primary на основном сайте.
7. Запустите приложения снова на основном сайте.

Восстановление после сбоев в объектном хранилище

Функция Ceph RGW Multi-Site представляет собой механизм асинхронной репликации данных между кластерами, предназначенный для синхронизации данных объектного хранилища между географически распределёнными кластерами Ceph, обеспечивая высокую доступность (HA) и возможности восстановления после сбоев (DR).

Содержание

Терминология

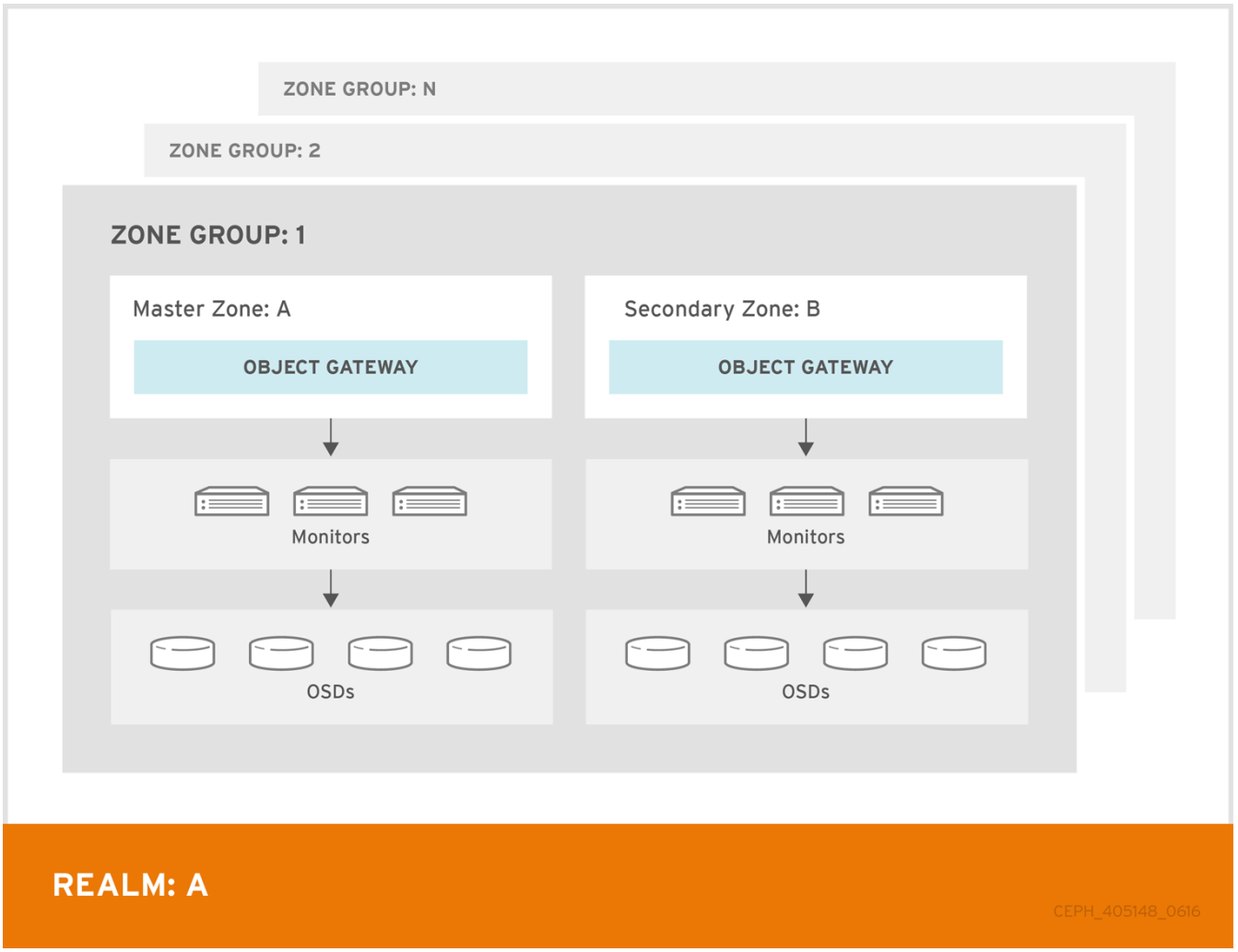
Термин	Объяснение
Primary Cluster	Кластер, который в данный момент предоставляет услуги хранения.
Secondary Cluster	Резервный кластер, используемый для целей резервного копирования.
Realm, ZoneGroup, Zone	• Realm: Наивысший уровень логической группировки в объектном хранилище Ceph. Представляет собой полное пространство имён объектного хранилища, обычно используется для мультисайтовой репликации и синхронизации. Realm может охватывать разные географические локации или дата-центры. • ZoneGroup: Логическая группировка внутри Realm, содержащая несколько зон (Zones). ZoneGroups обеспечивают синхронизацию и репликацию данных между зонами, обычно в пределах одного географического региона. • Zone: Логическая группировка внутри ZoneGroup, которая физически хранит данные. Каждая зона управляет и хранит объекты независимо и может иметь собственные конфигурации пулов данных и метаданных.

Предварительные требования

- Подготовьте два кластера для развертывания Rook-Ceph (Primary и Secondary) с сетевым соединением между ними.
- Оба кластера должны использовать одну и ту же версию платформы (v3.12 или новее).

- Убедитесь, что на кластерах Primary и Secondary не развернуто объектное хранилище Ceph.
- Обратитесь к документации Create Storage Service для развертывания Operator и создания кластеров. Не создавайте пулы объектного хранилища через мастер после создания кластера. Вместо этого используйте CLI-инструменты для настройки, как описано ниже.

Архитектура



Процедуры

В этом руководстве описано решение по синхронизации между двумя зонами в одном ZoneGroup.

1

Создание объектного хранилища в Primary кластере

На этом этапе создаются Realm, ZoneGroup, Primary Zone и ресурсы шлюза Primary Zone.

Выполните следующие команды на управляющем узле Primary кластера:

1. **Установите параметры**

```
export REALM_NAME=<realm-name>
export ZONE_GROUP_NAME=<zonegroup-name>
export PRIMARY_ZONE_NAME=<primary-zone-name>
export PRIMARY_OBJECT_STORE_NAME=<primary-object-store-name>
```

Описание параметров:

- `<realm-name>`: имя Realm.
- `<zonegroup-name>`: имя ZoneGroup.
- `<primary-zone-name>`: имя Primary Zone.
- `<primary-object-store-name>`: имя шлюза.

2. **Создайте объектное хранилище**

Command

```
cat << EOF | kubectl apply -f -
---
apiVersion: ceph.rook.io/v1
kind: CephObjectRealm
metadata:
  name: $REALM_NAME
  namespace: rook-ceph

---
apiVersion: ceph.rook.io/v1
kind: CephObjectZoneGroup
metadata:
  name: $ZONE_GROUP_NAME
  namespace: rook-ceph
spec:
  realm: $REALM_NAME

---
apiVersion: ceph.rook.io/v1
kind: CephObjectZone
metadata:
  name: $PRIMARY_ZONE_NAME
  namespace: rook-ceph
spec:
  zoneGroup: $ZONE_GROUP_NAME
  metadataPool:
    failureDomain: host
    replicated:
      size: 3
      requireSafeReplicaSize: true
  dataPool:
    failureDomain: host
    replicated:
      size: 3
      requireSafeReplicaSize: true
    parameters:
      compression_mode: none
  preservePoolsOnDelete: false

---
apiVersion: ceph.rook.io/v1
kind: CephBlockPool
metadata:
```

```
  labels:
    cpaas.io/builtin: "true"
  name: builtin-rgw-root
  namespace: rook-ceph
spec:
  name: .rgw.root
  application: rgw
  enableCrushUpdates: true
  failureDomain: host
  replicated:
    size: 3
  parameters:
    pg_num: "8"

---
apiVersion: ceph.rook.io/v1
kind: CephObjectStore
metadata:
  name: $PRIMARY_OBJECT_STORE_NAME
  namespace: rook-ceph
spec:
  gateway:
    port: 7480
    instances: 2
  zone:
    name: $PRIMARY_ZONE_NAME
EOF
```

Output

```
cephobjectrealm.ceph.rook.io/<realm-name> created
cephobjectzonegroup.ceph.rook.io/<zonegroup-name> created
cephobjectzone.ceph.rook.io/<primary-zone-name> created
cephobjectstore.ceph.rook.io/<primary-object-store-name> creat
ed
```

2 Настройка внешнего доступа для Primary Zone

1. Получите UID ObjectStore

```
export PRIMARY_OBJECT_STORE_UID=$(kubectl -n rook-ceph get cephobj
ectstore $PRIMARY_OBJECT_STORE_NAME -o jsonpath='{.metadata.uid}')
```

2. Создайте Service для внешнего доступа

```
cat << EOF | kubectl apply -f -
apiVersion: v1
kind: Service
metadata:
  name: rook-ceph-rgw-$PRIMARY_OBJECT_STORE_NAME-external
  namespace: rook-ceph
  labels:
    app: rook-ceph-rgw
    rook_cluster: rook-ceph
    rook_object_store: $PRIMARY_OBJECT_STORE_NAME
  ownerReferences:
    - apiVersion: ceph.rook.io/v1
      kind: CephObjectStore
      name: $PRIMARY_OBJECT_STORE_NAME
      uid: $PRIMARY_OBJECT_STORE_UID
spec:
  ports:
    - name: rgw
      port: 7480
      targetPort: 7480
      protocol: TCP
  selector:
    app: rook-ceph-rgw
    rook_cluster: rook-ceph
    rook_object_store: $PRIMARY_OBJECT_STORE_NAME
  sessionAffinity: None
  type: NodePort
EOF
```

3. Добавьте внешние конечные точки в CephObjectZone.

```
IP=$(kubectl get nodes -l 'node-role.kubernetes.io/control-plane'
-o jsonpath='{.items[0].status.addresses[?(@.type=="InternalIP")].
address}' | cut -d ' ' -f1 | tr -d '\n')
PORT=$(kubectl -n rook-ceph get svc rook-ceph-rgw-$PRIMARY_OBJECT_
STORE_NAME-external -o jsonpath='{.spec.ports[0].nodePort}')
ENDPOINT=http://$IP:$PORT
kubectl -n rook-ceph patch cephobjectzone $PRIMARY_ZONE_NAME --typ
e merge -p "{\"spec\":{\"customEndpoints\":[\"$ENDPOINT\"]}}"
```

3 Получение `access-key` и `secret-key`

```
kubectl -n rook-ceph get secrets $REALM_NAME-keys -o jsonpath='{.dat
a.access-key}'
kubectl -n rook-ceph get secrets $REALM_NAME-keys -o jsonpath='{.dat
a.secret-key}'
```

4 Создание Secondary Zone и настройка синхронизации Realm

В этом разделе описано, как создать Secondary Zone и настроить синхронизацию, подтягивая информацию Realm из Primary кластера.

Выполните следующие команды на управляющем узле Secondary кластера:

1. **Установите параметры**

```
export REALM_NAME=<realm-name>
export ZONE_GROUP_NAME=<zonegroup-name>

export REALM_ENDPOINT=<realm-endpoint>
export ACCESS_KEY=<access-key>
export SECRET_KEY=<secret-key>

export SECONDARY_ZONE_NAME=<secondary-zone-name>
export SECONDARY_OBJECT_STORE_NAME=<secondary-object-store-name>
```

Описание параметров:

- `<realm-name>`: Имя Realm.
- `<zone-group-name>`: Имя ZoneGroup.
- `<realm-endpoint>`: Внешний адрес, полученный из Primary кластера.
- `<access-key>`: AK, полученный здесь.
- `<secret-key>`: SK, полученный здесь.
- `<secondary-zone-name>`: имя Secondary Zone.
- `<secondary-object-store-name>`: имя шлюза Secondary.

2. **Создайте Secondary Zone и настройте синхронизацию Realm**

```
cat << EOF | kubectl apply -f -
apiVersion: v1
kind: Secret
metadata:
  name: $REALM_NAME-keys
  namespace: rook-ceph
data:
  access-key: $ACCESS_KEY
  secret-key: $SECRET_KEY

---
apiVersion: ceph.rook.io/v1
kind: CephObjectRealm
metadata:
  name: $REALM_NAME
  namespace: rook-ceph
spec:
  pull:
    endpoint: $REALM_ENDPOINT

---
apiVersion: ceph.rook.io/v1
kind: CephObjectZoneGroup
metadata:
  name: $ZONE_GROUP_NAME
  namespace: rook-ceph
spec:
  realm: $REALM_NAME

---
apiVersion: ceph.rook.io/v1
kind: CephObjectZone
metadata:
  name: $SECONDARY_ZONE_NAME
  namespace: rook-ceph
spec:
  zoneGroup: $ZONE_GROUP_NAME
  metadataPool:
    failureDomain: host
    replicated:
      size: 3
      requireSafeReplicaSize: true
  dataPool:
```

```
    failureDomain: host
    replicated:
      size: 3
      requireSafeReplicaSize: true
  preservePoolsOnDelete: false

---
apiVersion: ceph.rook.io/v1
kind: CephBlockPool
metadata:
  labels:
    cpaas.io/builtin: "true"
  name: builtin-rgw-root
  namespace: rook-ceph
spec:
  name: .rgw.root
  application: rgw
  enableCrushUpdates: true
  failureDomain: host
  replicated:
    size: 3
  parameters:
    pg_num: "8"

---
apiVersion: ceph.rook.io/v1
kind: CephObjectStore
metadata:
  name: $SECONDARY_OBJECT_STORE_NAME
  namespace: rook-ceph
spec:
  gateway:
    port: 7480
    instances: 2
  zone:
    name: $SECONDARY_ZONE_NAME
EOF
```

5 Настройка внешнего доступа для Secondary Zone

1. Получите UID шлюза Secondary

```
export SECONDARY_OBJECT_STORE_UID=$(kubectl -n rook-ceph get cepho
bjectstore $SECONDARY_OBJECT_STORE_NAME -o jsonpath='{.metadata.ui
d}')
```

2. Создайте Service для внешнего доступа

```
cat << EOF | kubectl apply -f -
apiVersion: v1
kind: Service
metadata:
  name: rook-ceph-rgw-$SECONDARY_OBJECT_STORE_NAME-external
  namespace: rook-ceph
  labels:
    app: rook-ceph-rgw
    rook_cluster: rook-ceph
    rook_object_store: $SECONDARY_OBJECT_STORE_NAME
  ownerReferences:
    - apiVersion: ceph.rook.io/v1
      kind: CephObjectStore
      name: $SECONDARY_OBJECT_STORE_NAME
      uid: $SECONDARY_OBJECT_STORE_UID
spec:
  ports:
    - name: rgw
      port: 7480
      targetPort: 7480
      protocol: TCP
  selector:
    app: rook-ceph-rgw
    rook_cluster: rook-ceph
    rook_object_store: $SECONDARY_OBJECT_STORE_NAME
  sessionAffinity: None
  type: NodePort
EOF
```

3. Добавьте внешние конечные точки в Secondary CephObjectZone

```
IP=$(kubectl get nodes -l 'node-role.kubernetes.io/control-plane'
-o jsonpath='{.items[0].status.addresses[?(@.type=="InternalIP")].
address}' | cut -d ' ' -f1 | tr -d '\n')
PORT=$(kubectl -n rook-ceph get svc rook-ceph-rgw-$SECONDARY_OBJEC
T_STORE_NAME-external -o jsonpath='{.spec.ports[0].nodePort}')
ENDPOINT=http://$IP:$PORT
kubectl -n rook-ceph patch cephobjectzone $SECONDARY_ZONE_NAME --t
ype merge -p "{\"spec\":{\"customEndpoints\":[\"$ENDPOINT\"]}}"
```

6 Проверка статуса синхронизации Ceph Object Storage

Выполните следующие команды в pod `rook-ceph-tools` Secondary кластера

```
# войти в pod rook-ceph-tools
kubectl -n rook-ceph exec -it $(kubectl -n rook-ceph get po -l app=ro
ok-ceph-tools -o jsonpath='{range .items[*]}{@.metadata.name}') -- ba
sh

radosgw-admin sync status
```

Пример вывода

```
          realm 962d6b80-b218-4fc8-8198-e498fabc4e9d (relam-primary)
      zonegroup 9de3acd7-0b01-4a04-ac84-1421c6103a16 (zonegroup-prima
ry)
           zone 7b3ce7f5-7090-46f6-afb1-d1bb156053da (zone-secondary)
   current time 2025-12-04T06:27:15Z
zonegroup features enabled: resharding
                   disabled: compress-encrypted
  metadata sync syncing
                full sync: 0/64 shards
                incremental sync: 64/64 shards
                metadata is caught up with master
      data sync source: 6319ca70-964e-47be-8b96-7c5bf5a128b1 (zone-pr
imary)
                        syncing
                        full sync: 0/128 shards
                        incremental sync: 128/128 shards
                        data is caught up with source
```

`metadata is caught up with master` и `data is caught up with source` означают, что статус синхронизации в норме.

Переключение при сбое (Failover)

При сбое Primary кластера необходимо повысить Secondary Zone до Primary Zone. После переключения шлюз Secondary Zone сможет продолжить предоставлять услуги объектного хранилища.

Процедуры

Выполните следующие команды в pod `rook-ceph-tools` Secondary кластера

```
# войти в pod rook-ceph-tools
kubectl -n rook-ceph exec -it $(kubectl -n rook-ceph get po -l app=rook-c
eph-tools -o jsonpath='{range .items[*]}{@.metadata.name}') -- bash

# Сделать восстановленную зону главной и зоной по умолчанию
radosgw-admin zone modify --rgw-realm=<realm-name> --rgw-zonegroup=<zone-
group-name> --rgw-zone=<secondary-zone-name> --master

# Обновить период, чтобы изменения вступили в силу
radosgw-admin period update --commit --rgw-realm=<realm-name> --rgw-zoneg
roup=<zone-group-name>
```

Параметры

- `<realm-name>`: имя Realm.
- `<zone-group-name>`: имя Secondary Zone Group.
- `<secondary-zone-name>`: имя Secondary Zone.

Возврат к основному кластеру (Failback)

После восстановления сбойного кластера на основном сайте и при необходимости возврата с резервного сайта выполните следующие шаги.

Процедуры

Выполните следующие команды в pod `rook-ceph-tools` Primary кластера

```
# войти в pod rook-ceph-tools
kubectl -n rook-ceph exec -it $(kubectl -n rook-ceph get po -l app=rook-ceph-tools -o jsonpath='{range .items[*]}{@.metadata.name}') -- bash

# проверить статус синхронизации, дождаться синхронизации с резервного сайта
radosgw-admin sync status

#           realm 962d6b80-b218-4fc8-8198-e498fabc4e9d (relam-primary)
#       zonegroup 9de3acd7-0b01-4a04-ac84-1421c6103a16 (zonegroup-primary)
#            zone 6319ca70-964e-47be-8b96-7c5bf5a128b1 (zone-primary)
#    current time 2025-12-04T07:18:26Z
# zonegroup features enabled: resharding
#                   disabled: compress-encrypted
#   metadata sync syncing
#                 full sync: 0/64 shards
#                 incremental sync: 64/64 shards
#                 metadata is caught up with master
#       data sync source: 7b3ce7f5-7090-46f6-afb1-d1bb156053da (zone-secondary)
#                         syncing
#                         full sync: 0/128 shards
#                         incremental sync: 128/128 shards
#                         data is caught up with source

# Сделать восстановленную зону главной и зоной по умолчанию
radosgw-admin zone modify --rgw-realm=<realm-name> --rgw-zonegroup=<zone-group-name> --rgw-zone=<primary-zone-name> --master

# Обновить период, чтобы изменения вступили в силу
radosgw-admin period update --commit --rgw-realm=<realm-name> --rgw-zonegroup=<zone-group-name>
```

Параметры

- `<realm-name>`: имя Realm.
- `<zone-group-name>`: имя Zone Group.
- `<primary-zone-name>`: имя Primary Zone.

Обновление параметров оптимизации

Платформа поддерживает заполнение параметров оптимизации в формате конфигурационного файла Ceph при создании кластера хранения, но не предоставляет способ их изменения через интерфейс после создания. Необходимо вручную обновить их согласно следующим шагам.

Содержание

Процедура

1. Сначала обновите параметры оптимизации хранения в Configmap с именем `rook-config-override-user`, замените поле `.data.config` и установите значение поля `.metadata.annotations[rook.cpaas.io/need-sync]` в `true`. Например:

```
apiVersion: v1
data:
  config: |
    [global]
    mon_memory_target=1073741824
    mds_cache_memory_limit=2147483648
    osd_memory_target=4147483648
kind: ConfigMap
metadata:
  annotations:
    cpaas.io/creator: admin
    cpaas.io/updated-at: "2022-03-01T12:24:04Z"
    rook.cpaas.io/need-sync: "true"
    rook.cpaas.io/sync-status: synced
  creationTimestamp: "2022-03-01T12:24:04Z"
  finalizers:
  - rook.cpaas.io/config-merge
  name: rook-config-override-user
  namespace: default
  resourceVersion: "38816864"
  uid: ce3a8f3e-6453-4bdd-bff0-e16cf7d5d5fa
```

2. Выполните команду `ceph tell [mon|osd|mgr|mds|rgw].* config set [key] [value]` в Pod с rook-ceph-tools для применения конфигурации в реальном времени.
3. Чтобы запустить Pod tools, отредактируйте ClusterServiceVersion (CSV) в пространстве имен rook-ceph и установите значение replicas для rook-ceph-tools в разделе Deployments равным 1.

Создание пользователя Ceph Object Store

Мы предоставляем возможность создания и настройки пользователей объектного хранилища через определения пользовательских ресурсов (CRD).

Содержание

Предварительные требования

- Объектный пул хранения уже создан

Процедура

1 Создание пользователя

Выполняйте команды на **контрольном узле** кластера.

```
cat << EOF | kubectl apply -f -
apiVersion: ceph.rook.io/v1
kind: CephObjectStoreUser
metadata:
  name: <name>
  namespace: <namespace>
spec:
  store: <ObjectStore>
  displayName: <displayName>
  clusterNamespace: <clusterNamespace>
  quotas:
    maxBuckets: 100
    maxSize: -1
    maxObjects: -1
  capabilities:
    user: "*"
    bucket: "*"
EOF
```

Параметры

Параметры	Описание
name	Имя пользователя объектного хранилища, который необходимо создать.
namespace	Пространство имён, в котором создаётся пользователь объектного хранилища.
displayName	Отображаемое имя.
clusterNamespace	Пространство имён, в котором находятся родительские ресурсы `CephCluster` и `CephObjectStore`. Если не указано, пользователь должен находиться в том же пространстве имён, что и кластер и объектное хранилище. Для включения этой возможности в CephObjectStore параметр `allowUsersInNamespaces` должен включать пространство имён данного пользователя.

Параметры	Описание
ObjectStore	Объектное хранилище, в котором будет создан пользователь. Соответствует имени пула объектного хранения.
quotas	**необязательно** Задаёт ограничения квот для пользователя. • maxBuckets: Максимальное количество бакетов для пользователя. По умолчанию `100`. • maxSize: Максимальный общий размер всех объектов во всех бакетах пользователя. Значение `-1` означает отсутствие ограничений. • maxObjects: Максимальное количество объектов во всех бакетах пользователя. Значение `-1` означает отсутствие ограничений.
capabilities	**необязательно** Ceph позволяет назначать пользователям дополнительные права. Эта настройка может использоваться только при создании пользователя объектного хранилища. Для изменения прав пользователя его необходимо удалить и создать заново. Подробнее см. в документации Ceph↗. Поддерживается добавление прав `read`, `write`, `read,write` или `*` для следующих ресурсов: • user • buckets • usage • metadata • zone • roles • info • amz-cache

Параметры	Описание
	• bilog • mdlog • datalog • user-policy • odic-provider • ratelimit

2 Разрешение создания пользователя в других пространствах имён

Если CephObjectStoreUser создаётся в пространстве имён, отличном от пространства имён кластера Rook, это пространство имён должно быть добавлено в список разрешённых, либо указано "*" для разрешения всех пространств имён. Это полезно для приложений, которым необходимо создавать учётные данные объектного хранилища в собственном пространстве имён.

Выполняйте команды на **контрольном узле** кластера.

```
kubectl -n rook-ceph patch cephobjectstore <ObjectStore> --type merge
-p '{"spec":{"allowUsersInNamespaces":["*"]}}'
```

3 Получение информации о пользователе

Выполняйте команды на **контрольном узле** кластера.

```
user_secret=$(kubectl -n <namespace> get cephobjectstoreuser <user-name> -o jsonpath='{.status.info.secretName}')

# ACCESS_KEY
kubectl -n <namespace> get secret $user_secret -o jsonpath='{.data.AccessKey}' | base64 --decode

# SECRET_KEY
kubectl -n <namespace> get secret $user_secret -o jsonpath='{.data.SecretKey}' | base64 --decode
```

Настройка квот для Storage Pool

Квота пула — это логический лимит размера данных, применяемый на уровне пула хранения Ceph. Она контролирует, сколько логических данных может быть записано в пул, фактическое физическое пространство будет равно **квота * репликация пула**.

Содержание

Предварительные требования

- Установлен кластер Ceph
- Созданы пулы Ceph

Установка квоты пула для File Storage Pool

Выполните команды на **контрольном узле** кластера.

Command

```
SIZE="<size>"
POOL_NAME="<fs-pool-name>"

kubectl patch cephfilesystem $POOL_NAME -n rook-ceph --type=json \
    -p "[{\"op\":\"add\",\"path\":\"/spec/dataPools/0/quotas/maxSize
\",\"value\":\"$SIZE\"}]"
```

Example

```
SIZE="100Gi"
POOL_NAME="cephfs"

kubectl patch cephfilesystem $POOL_NAME -n rook-ceph --type=json \
    -p "[{\"op\":\"add\",\"path\":\"/spec/dataPools/0/quotas/maxSize
\",\"value\":\"$SIZE\"}]"
```

Параметры

Параметры	Описание
size	квота в байтах в виде строки с суффиксами количества (например, "10Gi")
fs-pool-name	Имя File Storage Pool.

Установка квоты пула для Block Storage Pool

Выполните команды на **контрольном узле** кластера.

Command

```
SIZE="<size>"
POOL_NAME="<block-pool-name>"

kubectl patch cephblockpool $POOL_NAME -n rook-ceph --type=json \
    -p "[{\"op\":\"add\",\"path\":\"/spec/quotas/maxSize\",\"value
\":\"$SIZE\"}]"
```

Example

```
SIZE="100Gi"
POOL_NAME="rbd"

kubectl patch cephblockpool $POOL_NAME -n rook-ceph --type=json \
    -p "[{\"op\":\"add\",\"path\":\"/spec/quotas/maxSize\",\"value
\":\"$SIZE\"}]"
```

Параметры

Параметры	Описание
size	квота в байтах в виде строки с суффиксами количества (например, "10Gi")
block-pool-name	Имя Block Storage Pool.

Установка квоты пула для Object Storage Pool

Выполните команды на **контрольном узле** кластера.

Command

```
SIZE="<size>"
POOL_NAME="<object-pool-name>"

kubectl patch cephobjectstore $POOL_NAME -n rook-ceph --type=json \
    -p "[{\"op\":\"add\",\"path\":\"/spec/dataPool/quotas/maxSize
\",\"value\":\"$SIZE\"}]"
```

Example

```
SIZE="100Gi"
POOL_NAME="object"

kubectl patch cephobjectstore $POOL_NAME -n rook-ceph --type=json \
    -p "[{\"op\":\"add\",\"path\":\"/spec/dataPool/quotas/maxSize
\",\"value\":\"$SIZE\"}]"
```

Параметры

Параметры	Описание
size	квота в байтах в виде строки с суффиксами количества (например, "10Gi")
object-pool-name	Имя Object Storage Pool.

Проверка квоты пула через Ceph Terminal

```
ceph osd pool ls detail | grep max_bytes

pool 3 'cephfs-data0' replicated size 3 min_size 2 crush_rule 4 object_ha
sh rjenkins pg_num 32 pgp_num 32 autoscale_mode on last_change 100 lfor
0/0/56 flags hashpspool max_bytes 107374182400 stripe_width 0 application
cephfs read_balance_score 1.31
pool 4 'rbd' replicated size 3 min_size 2 crush_rule 5 object_hash rjenki
ns pg_num 32 pgp_num 32 autoscale_mode on last_change 117 lfor 0/0/111 fl
ags hashpspool,selfmanaged_snaps max_bytes 107374182400 stripe_width 0 ap
plication rbd read_balance_score 1.31
pool 12 'object.rgw.buckets.data' replicated size 3 min_size 2 crush_rule
13 object_hash rjenkins pg_num 32 pgp_num 32 autoscale_mode on last_chang
e 304 lfor 0/0/168 flags hashpspool max_bytes 107374182400 stripe_width 0
compression_mode none application rgw read_balance_score 1.59
```

Включение кэша D3N для Ceph RGW

Кэш D3N (Data Delivery Network) ускоряет доступ к горячим объектам, используя **локальные NVMe/SSD диски на узлах RGW в качестве слоя кэша**. Это снижает частоту операций чтения с бэкендных OSD и значительно улучшает производительность чтения горячих данных.

Содержание

Общая информация

Основная идея D3N проста, но эффективна: вместо многократного получения данных объекта из бэкендного хранилища Ceph, RGW обслуживает горячие данные напрямую из **локального постоянного кэша**, расположенного на быстрых дисках (NVMe/SSD), подключённых к узлам RGW.

Данный подход особенно полезен для:

- Загрузки больших объектов
- Повторных чтений горячих объектов
- Нагрузок, чувствительных к пропускной способности

Предварительные требования

- Установлен кластер Ceph
- Развернут Rook-Ceph
- Создан Object Storage (RGW / CephObjectStore)
- На узлах RGW доступны высокопроизводительные локальные диски (NVMe / SSD)

Обзор работы

1. Развернуть Ceph и создать Object Storage
2. Подготовить высокоскоростной локальный диск на узлах RGW и смонтировать его в директорию
3. Настроить `CephObjectStore для`:
   - Монтирования директории через `hostPath`
   - Включения параметров RGW, связанных с D3N

Подготовка локальной файловой системы для кэша

Смонтируйте директорию на каждом узле, где работает сервис RGW. Рекомендуется использовать выделенный высокопроизводительный диск (а не раздел), отформатированный в файловую систему **XFS**, и настроить `/etc/fstab` для сохранения монтирования после перезагрузки. В дальнейшем тексте замените `</path/to/cache/dir>` на фактический путь к директории, которую вы настроили.

Включение кэша D3N в CephObjectStore

Отредактируйте ресурс `CephObjectStore`:

```
kubectl edit cephobjectstore object-store -n rook-ceph
```

Добавьте следующую конфигурацию в раздел `gateway`:

```
gateway:
  additionalVolumeMounts:
  - subPath: cache
    volumeSource:
      hostPath:
        path: </path/to/cache/dir>
  rgwConfig:
    rgw_d3n_l1_datacache_persistent_path: /var/rgw/cache/
    rgw_d3n_l1_datacache_size: "10737418240"
    rgw_d3n_l1_local_datacache_enabled: "true"
```

Параметры

Параметр	Описание
`rgw_d3n_l1_local_datacache_enabled`	Включение локального кэша D3N
`rgw_d3n_l1_datacache_persistent_path`	Директория кэша внутри контейнера RGW
`rgw_d3n_l1_datacache_size`	Максимальный размер кэша в байтах

Проверка конфигурации D3N

Проверка конфигурации RGW через Ceph Tools

В поде `rook-ceph-tools` выполните:

```
ceph config dump | grep d3n
```

Пример вывода:

```
client.rgw.obj.a  advanced  rgw_cache_enabled                    true
client.rgw.obj.a  advanced  rgw_d3n_l1_datacache_persistent_path /var/rg
w/cache/
client.rgw.obj.a  advanced  rgw_d3n_l1_datacache_size            10737418
240
client.rgw.obj.a  advanced  rgw_d3n_l1_local_datacache_enabled   true
```

Проверка логов RGW

После перезапуска пода RGW должны появиться логи инициализации D3N:

```
kubectl logs -n rook-ceph rook-ceph-rgw-object-store-a-xxxx | grep -i d3n
```

Пример вывода:

```
rgw_d3n: rgw_d3n_l1_local_datacache_enabled=1
rgw_d3n: rgw_d3n_l1_datacache_persistent_path='/var/rgw/cache/'
rgw_d3n: rgw_d3n_l1_datacache_size=10737418240
rgw_d3n: rgw_d3n_l1_evict_cache_on_start=1
rgw_d3n: rgw_d3n_l1_eviction_policy=lru
```

Проверка работы кэша

При загрузке объектов через RGW в директории кэша на хосте появятся кэшированные файлы:

```
ls </path/to/cache/dir>
```

Пример вывода:

```
40b934ab-6c7e-45e7-9fed-a35bc143ce95...multipart_v2v-demo.mkv.1
40b934ab-6c7e-45e7-9fed-a35bc143ce95...multipart_v2v-demo.mkv.2
40b934ab-6c7e-45e7-9fed-a35bc143ce95...shadow_v2v-demo.mkv.3_1
...
```

Наличие этих файлов подтверждает, что RGW обслуживает данные из локального кэша D3N.

Configure in-transit encryption

This topic describes how to enable or disable in-transit encryption based on `msgr2` for ACP distributed storage.

Содержание

Overview

Ceph `msgr2` is the second generation of the Ceph messenger protocol. It supports two connection modes:

- `crc`: authenticates the peers and validates data integrity, but does not encrypt payload data.
- `secure`: encrypts traffic on the wire and provides cryptographic integrity protection.

In ACP distributed storage, in-transit encryption is controlled by `CephCluster.spec.network.connections.encryption.enabled`.

Limitations and prerequisites

Before enabling this feature, pay attention to the following restrictions:

- **ACP Version**
  - v4.3.0 and later.
- **OS and Kernel(ceph daemon and client nodes)**
  - kernel `5.11` and later.
  - `Ubuntu 22.04` and later.

> **WARNING**
>
> In-transit encryption increases CPU overhead and may reduce throughput or increase latency, especially on busy storage nodes or low-frequency CPUs. Evaluate the impact in a staging environment first.

Enable in-transit encryption for a new cluster

If the storage cluster has not been created yet, add the following fields to the `CephCluster` manifest before creation:

```
spec:
  network:
    connections:
      encryption:
        enabled: true
```

After the cluster is created, verify that:

- CephFS PVCs can still be mounted successfully
- RBD and CephFS workloads on all nodes use supported kernel versions

Enable in-transit encryption after deployment

If the cluster is already running, changing only the encryption switch is the lowest-risk approach.

Step 1. Confirm node kernel versions

Run the following command on all Kubernetes nodes that mount Ceph volumes and confirm that the kernel version meets the prerequisite:

```
uname -r
```

If some worker nodes do not meet the requirement, do not enable transport encryption on a production cluster until those nodes are upgraded.

Step 2. Enable encryption in CephCluster

```
kubectl patch cephcluster ceph-cluster -n rook-ceph --type merge -p '
spec:
  network:
    connections:
      encryption:
        enabled: true
'
```

Step 3. Wait for the configuration to take effect

After the configuration is updated:

- Check whether related Pods restart normally
- Recreate a test Pod that mounts a CephFS or RBD PVC
- Confirm I/O works as expected

Disable transport encryption

Disable encryption on an existing cluster

To disable only transport encryption and keep `msgr2` available:

```
kubectl patch cephcluster ceph-cluster -n rook-ceph --type merge -p '
spec:
  network:
    connections:
      encryption:
        enabled: false
'
```

Verification

After enabling the feature, verify the cluster from both the Kubernetes side and the Ceph side.

Check the CephCluster settings

```
kubectl get cephcluster ceph-cluster -n rook-ceph -o yaml
```

Confirm that the output contains:

```
spec:
  network:
    connections:
      encryption:
        enabled: true
```

Check client compatibility

After in-transit encryption is enabled:

- clients using `msgr2 secure` should connect normally
- clients configured with non-encrypted modes such as `legacy` or `crc` will fail to connect

Check workload mounts

Create or restart a test workload that mounts:

- a CephFS PVC
- an RBD PVC

Then verify:

- the Pod starts successfully
- the filesystem can be read and written
- no mount-related errors appear in CSI or workload logs

Troubleshooting suggestions

If enabling encryption causes mount failures or service interruptions, check the following items first:

1. Node kernel version does not satisfy the requirement.
2. Some nodes or external clients do not support `msgr2 secure`, or are still configured with `ms_mode=legacy` or `ms_mode=crc`.
3. Network policies, firewalls, or security groups do not allow port `3300`.
4. CPU resources are insufficient after encryption is enabled.

If the change affects production workloads, disable encryption first and then investigate compatibility and performance bottlenecks.

Performance impact

ACP cannot provide a fixed percentage for the overhead of `msgr2 secure`. The actual impact depends on CPU model, whether the CPU provides AES acceleration, network bandwidth, I/O size, and whether the workload is CPU-bound or network-bound.

In practice:

- latency usually increases slightly, and the increase is often more visible on small I/O or latency-sensitive workloads
- CPU usage usually increases on both clients and Ceph daemons because traffic must be encrypted and integrity-protected
- the impact is typically more noticeable on high-throughput workloads, slower CPUs, or environments without strong AES acceleration

As an operational estimate, when modern x86 CPUs with AES-NI are used, a reasonable starting expectation is:

- average latency increase: about `5%` to `15%`
- CPU usage increase on storage and client nodes handling heavy I/O: about `10%` to `30%`

These values are an engineering estimate rather than a product guarantee. Before enabling encryption in production, benchmark a representative workload in a staging environment and compare at least the following metrics:

- average and P99 read/write latency
- client node CPU usage
- OSD node CPU usage
- throughput and IOPS

MinIO Object Storage

Введение

Введение

Установка

Установка

Предварительные требования

Процедура

Связанная информация

Архитектура

Архитектура

Основные компоненты:

Архитектура развертывания:

Масштабирование с помощью нескольких пулов:

Заключение:

Основные понятия

Основные концепции

Руководства

Добавление пула хранения

Примечания

Процедура

Мониторинг и оповещения

Мониторинг

Оповещения

Как сделать

Восстановление данных пос*л*

Применимые сценарии

Терминология

Предварительные требования

Шаги операции

Связанные операции

Руководство по миграции данных из MinIO

1. Обзор и архитектура
2. Требования и подготовка
3. Конфигурация развертывания (манифесты)
4. Порядок выполнения
5. Рекомендации по устранению неполадок и настройке

Введение

Alauda Container Platform (ACP) Object Storage с MinIO — это сервис объектного хранения, лицензированный под Apache License v2.0. Он совместим с интерфейсом облачного хранилища Amazon S3, что делает его особенно подходящим для хранения больших объемов неструктурированных данных, таких как изображения, видео, файлы журналов, резервные копии и образы контейнеров/виртуальных машин. Размер объекта может варьироваться от нескольких КБ до максимума в 5 ТБ.

Основные преимущества следующие:

- **Простота**: Минимализм — главный принцип дизайна MinIO, обеспечивающий функциональность «из коробки». Простота снижает вероятность ошибок, увеличивает время безотказной работы и повышает надежность, а также улучшает производительность.
- **Высокая производительность**: MinIO является мировым лидером в области объектного хранения. На стандартном оборудовании скорость чтения/записи может достигать до 183 ГБ/с и 171 ГБ/с соответственно.
- **Масштабируемость**: Можно создавать несколько небольших и средних кластеров, которые легко управляются, поддерживая агрегацию нескольких кластеров в сверхбольшой пул ресурсов между дата-центрами, вместо прямого использования крупномасштабного централизованного распределенного кластера.
- **Облачная нативность**: Соответствует всем нативным архитектурам и процессам построения облачных вычислений, включает новейшие технологии и концепции облачных вычислений, делая объектное хранилище более удобным для Kubernetes.

Установка

Alauda Container Platform (ACP) Object Storage с MinIO — это сервис объектного хранения, основанный на протоколе с открытым исходным кодом по лицензии Apache License v2.0. Он совместим с интерфейсом облачного хранилища Amazon S3 и идеально подходит для хранения больших объемов неструктурированных данных, таких как изображения, видео, файлы журналов, резервные копии и образы контейнеров/виртуальных машин. Объектный файл может иметь любой размер — от нескольких килобайт до максимума в 5 терабайт.

Содержание

Предварительные требования

- MinIO строится на базовом хранилище, поэтому убедитесь, что в текущем кластере создан класс хранилища. Рекомендуется TopoLVM.
- **Скачайте** установочный пакет **Alauda Container Platform Storage Essentials**, соответствующий архитектуре вашей платформы.
- **Загрузите** установочный пакет **Alauda Container Platform Storage Essentials** с помощью механизма Upload Packages.
- **Скачайте** установочный пакет **Alauda Container Platform (ACP) Object Storage with MinIO**, соответствующий архитектуре вашей платформы.
- **Загрузите** установочный пакет **Alauda Container Platform (ACP) Object Storage with MinIO** с помощью механизма Upload Packages.

Процедура

1 Развертывание Alauda Container Platform Storage Essentials

1. Войдите в систему, перейдите на страницу **Administrator**.
2. Нажмите **Marketplace** > **OperatorHub**, чтобы перейти на страницу **OperatorHub**.
3. Найдите **Alauda Container Platform Storage Essentials**, нажмите **Install** и перейдите на страницу **Install Alauda Container Platform Storage Essentials**. Параметры конфигурации:

Параметр	Рекомендуемая конфигурация
Channel	По умолчанию канал `stable`.
Installation Mode	`Cluster`: Все пространства имён в кластере используют один экземпляр Operator для создания и управления, что снижает использование ресурсов.

Параметр	Рекомендуемая конфигурация
Installation Place	Выберите `Recommended`, Namespace поддерживается только **acp-storage**.
Upgrade Strategy	`Manual`: При наличии новой версии в Operator Hub требуется ручное подтверждение для обновления Operator до последней версии.

2 Развертывание Operator

1. В левой навигационной панели нажмите **Storage** > **Object Storage**.
2. Нажмите **Configure Now**.
3. На странице мастера **Deploy MinIO Operator** нажмите внизу справа **Deploy Operator**.
   - Если страница автоматически переходит к следующему шагу, значит развертывание Operator прошло успешно.
   - Если развертывание не удалось, следуйте подсказкам интерфейса для **Clean Up Deployed Information and Retry** и повторите развертывание Operator.

3 Создание кластера

1. На странице мастера **Create Cluster** настройте базовую информацию.

Параметр	Описание
Access Key	Идентификатор ключа доступа. Уникальный идентификатор, связанный с приватным ключом доступа; используется вместе с Access Key ID для шифрования и подписи запросов.
Secret Key	Приватный ключ доступа, используемый вместе с Access Key ID для шифрования и подписи запросов, идентификации отправителя и предотвращения подделки запросов.

2. В разделе **Resource Configuration** настройте характеристики согласно следующим инструкциям.

Параметр	Описание
Small scale	Подходит для обработки до 100 000 объектов, поддерживает не более 50 одновременных обращений в тестовых средах или сценариях резервного копирования. Запрос и лимит ресурсов CPU по умолчанию — 2 ядра, запрос и лимит памяти — 4 Gi.
Medium scale	Предназначен для корпоративных приложений с хранением 1 000 000 объектов и поддержкой до 200 одновременных запросов. Запрос и лимит CPU по умолчанию — 4 ядра, запрос и лимит памяти — 8 Gi.
Large scale	Предназначен для групп пользователей с потребностями хранения 10 000 000 объектов и обработкой до 500 одновременных запросов, подходит для сценариев с высокой нагрузкой. Запрос и лимит CPU по умолчанию — 8 ядер, запрос и лимит памяти — 16 Gi.
Custom	Предлагает гибкие настройки для профессиональных пользователей с особыми требованиями, обеспечивая точное соответствие масштаба сервиса и требований к производительности. Примечание: при настройке пользовательских характеристик убедитесь, что: • Запрос CPU больше 100 м. • Запрос памяти не меньше 2 Gi. • Лимиты CPU и памяти не меньше соответствующих запросов.

3. В разделе **Storage Pool** настройте соответствующую информацию согласно следующим инструкциям.

Параметр	Описание
Instance Number	Увеличение количества экземпляров в кластере MinIO значительно повышает производительность и надежность системы, обеспечивая высокую доступность данных. Однако слишком большое число экземпляров может привести к следующим проблемам: • Увеличение потребления ресурсов. • Если на одном узле размещено несколько экземпляров, сбой узла может привести к одновременному отключению нескольких экземпляров, снижая общую надежность кластера. **Примечание**: • Минимальное количество экземпляров — 4. • Если количество экземпляров больше 16, введённое значение должно быть кратно 8. • При добавлении дополнительных пулов хранения количество экземпляров не должно быть меньше, чем в первом пуле хранения.
Single Storage Volume	Ёмкость одного тома PVC для хранения. Каждый сервис хранения управляет одним томом. После ввода ёмкости одного тома платформа автоматически рассчитает ёмкость пула хранения и другую информацию, которую можно просмотреть в **Storage Pool Overview**.
Underlying Storage	Базовое хранилище, используемое кластером MinIO. Пожалуйста, выберите класс хранилища, созданный в текущем кластере. Рекомендуется TopoLVM.

Параметр	Описание
Storage Nodes	Выберите узлы хранения, необходимые для кластера MinIO. Рекомендуется использовать от 4 до 16 узлов хранения. Платформа развернёт один сервис хранения на каждом выбранном узле.
Storage Pool Overview	Для конкретных параметров и формул расчёта смотрите Storage Pool Overview.

4. В разделе **Access Configuration** настройте соответствующую информацию согласно следующим инструкциям.

Параметр	Описание
External Access	При включении поддерживается доступ к MinIO из других кластеров; при отключении доступ возможен только внутри кластера.
Protocol	Поддерживает HTTP и HTTPS; при выборе HTTPS необходимо ввести **Domain** и импортировать **Public Key** и **Private Key** сертификата домена. **Примечание**: • При протоколе HTTP поды внутри кластера могут обращаться к MinIO напрямую по полученному IP или доменному имени без настройки сопоставления IP и домена; узлы внутри кластера могут обращаться к MinIO напрямую по IP, а для доступа по домену требуется ручная настройка сопоставления IP и домена; внешний доступ возможен напрямую по IP. • При протоколе HTTPS доступ к MinIO по IP невозможен как внутри, так и снаружи кластера. Для нормального доступа по домену требуется вручную настроить сопоставление между полученным IP и введённым при создании кластера доменом.

Параметр	Описание
Access Method	• **NodePort**: Открывает фиксированный порт на каждом хосте вычислительного узла для внешнего доступа к сервису. При настройке доступа по домену рекомендуется использовать VIP для разрешения доменного имени, чтобы обеспечить высокую доступность. • **LoadBalancer**: Использует балансировщик нагрузки для перенаправления трафика на бэкенд-сервисы. Перед использованием убедитесь, что в текущем кластере развернут плагин MetalLB и в пуле внешних адресов есть доступные IP.

5. Нажмите внизу справа **Create Cluster**.

   - Если страница автоматически перейдёт к **Cluster Details**, значит создание кластера прошло успешно.
   - Если кластер остаётся в процессе создания, можно нажать **Cancel**. После отмены развернутая информация о кластере будет очищена, и вы сможете вернуться на страницу создания кластера для повторного создания.

4 Создание Bucket

Войдите на управляющий узел кластера и используйте команду для создания bucket.

1. На странице сведений о кластере перейдите на вкладку **Access Method**, чтобы посмотреть адрес доступа MinIO, или выполните команду:

```
kubectl get svc -n <tenant ns> minio | grep -w minio | awk '{print $3}'
```

Примечание:

- Замените `tenant ns` на фактическое пространство имён `minio-system`.
- Пример: `kubectl get svc -n minio-system minio | grep -w minio | awk '{print $3}'`

2. Получите команду mc.

```
wget https://dl.min.io/client/mc/release/linux-amd64/mc -O /bin/mc
&& chmod a+x /bin/mc
```

3. Настройте псевдоним кластера MinIO.

- IPv4:

```
mc --insecure alias set <minio cluster alias> http://<minio endp
oint>:<port> <accessKey> <secretKey>
```

- IPv6:

```
mc --insecure alias set <minio cluster alias> http://[<minio end
point>]:<port> <accessKey> <secretKey>
```

- Доменное имя:

```
mc --insecure alias set <minio cluster alias> http://<domain nam
e>:<port> <accessKey> <secretKey>
mc --insecure alias set <minio cluster alias> https://<domain na
me>:<port> <accessKey> <secretKey>
```

Примечание:

- Введите IP-адрес, полученный на шаге 1, для `minio endpoint`.
- Введите **Access Key** и **Secret Key**, созданные при создании кластера, для `accessKey` и `secretKey`.
- Примеры конфигурации:
  - IPv4: `mc --insecure alias set myminio http://12.4.121.250:80 07Apples@ 07Apples@`
  - IPv6: `mc --insecure alias set myminio http://[2004::192:168:143:117]:80 07Apples@ 07Apples@`
  - Доменное имя: `mc --insecure alias set myminio http://test.minio.alauda:80 07Apples@ 07Apples@` или `mc --insecure`

`alias set myminio https://test.minio.alauda:443 07Apples@ 07Apples@`

4. Создайте bucket.

```
mc --insecure mb <minio cluster alias>/<bucket name>
```

5 Загрузка/скачивание файлов

После создания bucket вы можете использовать командную строку для загрузки файлов в bucket или скачивания существующих файлов из bucket.

1. Создайте файл для тестовой загрузки. Этот шаг можно пропустить, если загружаете существующий файл.

```
touch <file name>
```

2. Загрузите файлы в bucket.

```
mc --insecure cp <file name> <minio cluster alias>/<bucket name>
```

3. Просмотрите файлы в bucket, чтобы убедиться в успешной загрузке.

```
mc --insecure ls <minio cluster alias>/<bucket name>
```

4. Удалите загруженные файлы.

```
mc --insecure rm <minio cluster alias>/<bucket name>/<file name>
```

Связанная информация

Таблица соответствия коэффициента избыточности

Примечание: При добавлении дополнительных пулов хранения коэффициент избыточности рассчитывается на основе количества экземпляров в первом пуле хранения.

Количество экземпляров	Коэффициент избыточности
4 - 5	2
6 - 7	3
>= 8	4

Обзор пула хранения

Параметр обзора пула хранения	Формула расчёта
Используемая ёмкость	Если Instance Number ≤ 16, Используемая ёмкость = Ёмкость одного тома × (Количество экземпляров - Коэффициент избыточности).
Если количество экземпляров > 16, Используемая ёмкость = Ёмкость одного тома × (Количество экземпляров - 4 × (Количество экземпляров + 15) / 16). Результат выражения "4 × (Количество экземпляров + 15) / 16" округляется вниз.	
Общая ёмкость	Общая ёмкость = Количество экземпляров × Ёмкость одного тома
Количество отказоустойчивых сервисов хранения	Если Количество экземпляров > 2 × Коэффициент избыточности, Количество отказоустойчивых сервисов = Коэффициент избыточности.
Если Количество экземпляров = 2 × Коэффициент избыточности, количество отказоустойчивых	

Параметр обзора пула хранения	Формула расчёта
сервисов = Коэффициент избыточности - 1	

Архитектура

Alauda Container Platform (ACP) Object Storage с MinIO — это высокопроизводительная распределённая система объектного хранения, разработанная для облачно-нативных сред. Она использует стирающее кодирование, распределённые пулы хранения и механизмы высокой доступности для обеспечения долговечности данных и масштабируемости в Kubernetes.

Содержание

Основные компоненты:

- **MinIO Operator**: Управляет развертыванием и обновлением кластеров MinIO.
- **MinIO Peer**: Настраивает и управляет функциональностью репликации сайтов MinIO.
- **MinIO Pool**: Основной компонент MinIO, отвечающий за обработку запросов объектного хранения. Каждый пул соответствует StatefulSet и предоставляет ресурсы хранения.

Архитектура развертывания:

Для развертывания MinIO в Kubernetes необходимо определить MinIO tenant, указав количество серверных экземпляров (pod) и количество томов (дисков) на каждый экземпляр. Каждый сервер MinIO управляется через StatefulSet, что обеспечивает стабильные идентификаторы и постоянное хранилище. MinIO агрегирует все диски в один или несколько стирающих наборов и применяет стирающее кодирование для обеспечения отказоустойчивости.

Масштабирование с помощью нескольких пулов:

Кластеры MinIO могут масштабироваться путём добавления дополнительных серверных пулов, каждый из которых имеет свой стирающий набор. Хотя это увеличивает ёмкость хранения, такая архитектура усложняет обслуживание кластера и снижает общую надёжность. Сбой в любом серверном пуле может сделать весь кластер MinIO недоступным, даже если другие пулы продолжают работать.

Заключение:

MinIO — это высокомасштабируемое облачно-нативное решение для объектного хранения, которое обеспечивает баланс между производительностью и надёжностью. При проектировании кластера MinIO важно тщательно продумывать пулы хранения, настраивать параметры стирающего кодирования и реализовывать стратегии высокой доступности для обеспечения целостности данных и стабильности работы в Kubernetes.

Основные понятия

Основные концепции

Основные концепции

- **Erasure Coding (EC)**: MinIO использует кодирование с удалением (Reed-Solomon erasure coding) для разбиения объектов на фрагменты данных и контрольные фрагменты (parity shards), распределяя их по нескольким дискам для обеспечения отказоустойчивости. Например, в конфигурации с 16 дисками данные могут быть разбиты на 12 фрагментов данных и 4 контрольных фрагмента, что позволяет системе восстанавливать данные даже при выходе из строя до 4 дисков.
- **Server Pools & Erasure Sets**: Server Pools MinIO — это логические объединения ресурсов хранения, где каждый пул состоит из нескольких узлов, совместно использующих возможности хранения и вычислений. Внутри пула диски автоматически организуются в один или несколько **Erasure Sets**.
  - **Распределение данных**: При сохранении объекта он разбивается на фрагменты данных и контрольные фрагменты, которые распределяются по разным дискам внутри erasure set.
  - **Модель избыточности**: Erasure sets представляют собой базовую единицу избыточности данных, обеспечивая устойчивость на основе настроенного соотношения фрагментов данных и контрольных фрагментов.
  - **Масштабируемость**: Один пул хранения MinIO может содержать несколько erasure sets, и новые данные всегда записываются в erasure set с наибольшей доступной емкостью.

Руководства

Добавление пула хранения

Пул хранения — это логический раздел, используемый для хранения данных. В одном и том же кластере хранения могут одновременно использоваться разные типы базового хранилища для удовлетворения различных бизнес-требований.

Помимо пулов хранения, созданных при настройке объектного хранилища, вы также можете добавить дополнительные пулы хранения.

Содержание

Примечания

Добавление пула хранения вызовет кратковременное прерывание работы сервиса MinIO, но после этого он автоматически восстановится в нормальное состояние.

Процедура

1. Перейдите в раздел **Administrator**.
2. В левой навигационной панели нажмите **Storage Management** > **Object Storage**.

3. На вкладке **Cluster Information** прокрутите вниз до раздела **Storage Pool** и нажмите **Add Storage Pool**.
4. Настройте соответствующие параметры согласно приведённым ниже инструкциям.

Параметр	Описание
Underlying Storage	Базовое хранилище, используемое кластером MinIO. Пожалуйста, выберите существующий класс хранения, созданный в текущем кластере, рекомендуется использовать TopoLVM.
Storage Nodes	Выберите узлы хранения, необходимые для кластера MinIO. Рекомендуется использовать от 4 до 16 узлов хранения; платформа развернёт по 1 сервису хранения на каждый выбранный узел. **Примечание**: при использовании 3 узлов хранения для обеспечения надёжности на каждый узел будет развернуто по 2 сервиса хранения.
Single Storage Volume	Вместимость одного тома хранения PVC. Каждый сервис хранения управляет 1 томом хранения, и после ввода вместимости одного тома платформа автоматически рассчитает ёмкость пула хранения и другую информацию, которую можно просмотреть в **Storage Pool Overview**.

5. Нажмите **Confirm**.

Мониторинг и оповещения

Система объектного хранения данных оснащена встроенными возможностями мониторинга и оповещений, охватывающими кластеры хранения, состояние сервисов и использование ресурсов. Также поддерживаются настраиваемые политики уведомлений для информирования вашей операционной команды. Информация мониторинга в реальном времени помогает в оптимизации производительности и принятии операционных решений, а автоматические оповещения обеспечивают стабильность и надежность вашей системы хранения.

Содержание

Мониторинг

По умолчанию платформа собирает ключевые метрики по кластерам хранения и состоянию сервисов. Вы можете получить доступ к данным мониторинга в реальном времени в разделе **Storage Management** > **Object Storage** > **Monitoring**.

Обзор хранения

Этот раздел предоставляет общий обзор состояния системы хранения, статуса сервисов и использования сырой емкости. Если состояние хранения ненормальное, детали оповещения укажут на первопричину, что поможет эффективно диагностировать и устранять проблемы.

Мониторинг кластера

Отслеживайте использование сырой емкости и тенденции производительности ввода-вывода по всему кластеру хранения. Это помогает выявлять узкие места, оптимизировать распределение ресурсов и обеспечивать бесперебойную работу с данными.

Мониторинг объектов

Отслеживайте модели доступа, включая общее количество запросов и количество неудачных запросов. Эти данные помогают анализировать нагрузку на хранилище и выявлять аномалии, которые могут свидетельствовать о сбоях сервисов или угрозах безопасности.

Оповещения

Платформа поставляется с преднастроенными политиками оповещений для обнаружения аномалий и отправки уведомлений при достижении заданных порогов. Встроенные правила охватывают ключевые области, такие как состояние компонентов, использование емкости и целостность пользовательских данных.

Настройка уведомлений

Для обеспечения своевременного реагирования настройте политики уведомлений в **Operations Center**. Оповещения могут отправляться по электронной почте, SMS или другим каналам, чтобы информировать ответственных сотрудников. Тонко настройте параметры в соответствии с рабочим процессом реагирования вашей организации.

Обработка оповещений

- **Кластер в состоянии "Alert"**: Сработало предупреждение, и стабильность системы может быть под угрозой. Проверьте раздел **Live Alerts** для получения подробностей, определите первопричину и примите корректирующие меры.
- **Кластер в состоянии "Failure"**: Кластер хранения перестал работать. Требуется немедленное вмешательство для восстановления доступности сервиса.

Платформа классифицирует оповещения по уровням серьезности, помогая командам приоритизировать реагирование на инциденты:

Уровень серьезности	Описание
Critical	Сбой системы, влияющий на бизнес-процессы или приводящий к потере данных. Требуется немедленное действие.
Major	Известная проблема, которая может привести к сбоям в функциональности и нарушить бизнес-процессы.
Warning	Потенциальный риск, который при отсутствии реакции может повлиять на производительность или доступность.

Анализ после инцидента

Журнал **Alert History** содержит все прошлые инциденты, предоставляя ценные данные для анализа и улучшения системы. При рассмотрении прошлых оповещений учитывайте следующее:

1. Каковы были точные симптомы во время инцидента?
2. Повторяются ли определённые оповещения со временем? Можно ли принять проактивные меры для предотвращения повторения?

3. Был ли определённый временной интервал с резким увеличением количества оповещений? Был ли он вызван операционной проблемой или внешним фактором? Нужно ли скорректировать стратегию реагирования?

Постоянный анализ шаблонов оповещений и совершенствование стратегий мониторинга позволяют повысить устойчивость системы, минимизировать время простоя и обеспечить бесперебойную работу хранилища.

Как сделать

Восстановление данных после аварий

MinIO поддерживает создание центра аварийного восстановления через удалённое резервное копирование данных или активное-активное развертывание, чтобы обеспечить сохранность исходных данных при возникновении аварийных ситуаций, тем самым гарантируя безопасность и надёжность данных.

Содержание

Применимые сценарии

- **Горячее резервное копирование**: Два дата-центра находятся в одном городе или в разных местах, один из них основной, другой — резервный. Данные в реальном времени реплицируются с основного кластера на резервный, чтобы обеспечить согласованность данных. При аварии в основном кластере бизнес-трафик может быть бесшовно переключён на резервный кластер для обеспечения непрерывности работы.

- **Активное-активное на уровне города**: В архитектуре активное-активное на уровне города (мультикластер) два дата-центра расположены в разных кластерах. Оба дата-центра активны и могут одновременно принимать бизнес-трафик. При аварии в одном дата-центре бизнес продолжает работать без прерывания в другом.

Терминология

- **Основной кластер**: Кластер, который в данный момент активен и обрабатывает бизнес-запросы. Является источником данных или инициатором операций. В основном кластере данные создаются, изменяются или обновляются, и бизнес-трафик сначала направляется именно в этот кластер для обработки.
- **Целевой кластер**: Кластер, который принимает репликацию данных, миграцию или переключение. Обычно находится в резервном или ожидательном состоянии, готовый принять данные с основного кластера или взять на себя бизнес-трафик. При сбое основного кластера или необходимости переключения целевой кластер получает копии данных с основного или принимает бизнес-трафик для обеспечения непрерывности работы. В сценарии активное-активное оба кластера могут выступать в роли целевого друг для друга.

Предварительные требования

- И основной, и целевой кластер должны иметь включённый доступ к внешней сети. Для конкретных методов настройки см. Create Object Storage.
- Основной кластер должен использовать метод доступа **LoadBalancer**, а целевой кластер рекомендуется поддерживать функциональность **балансировки нагрузки**.
- Основной и целевой кластеры должны использовать одинаковый протокол доступа, то есть либо оба HTTP, либо оба HTTPS.
- При использовании протокола HTTPS оба кластера должны настроить DNS-разрешение для себя и друг друга.
- При использовании HTTPS рекомендуется, чтобы и основной, и целевой кластеры использовали сертификаты, подписанные CA, для обеспечения безопасного и доверенного соединения; если используются самоподписанные сертификаты, обе

стороны должны импортировать и доверять сертификатам друг друга для успешного установления защищённого HTTPS-соединения.

Шаги операции

1. Войдите в **Administrator**.
2. В левой панели навигации нажмите **Storage Management** > **Object Storage**.
3. На вкладке **Data Disaster Recovery** нажмите **Add Target Cluster**.
4. Настройте соответствующие параметры целевого кластера согласно следующим инструкциям.

Параметр	Описание
Access Address	Внешний адрес доступа целевого кластера, начинающийся с http:// или https://.
Access Key	Идентификатор Access Key для целевого кластера. Уникальный идентификатор, связанный с приватным ключом доступа; используется вместе с приватным ключом для шифрования запросов.
Secret Key	Приватный ключ доступа, используемый вместе с Access Key ID для шифрования запросов, идентификации отправителя и предотвращения изменения запроса.

5. Нажмите **Add**.

- После успешного добавления вы сможете просматривать статус целевого кластера и статус синхронизации между кластерами.

Параметр	Описание
Cluster Status	Статус целевого кластера, включая **Healthy**, **Abnormal** или **Unknown**.
Buckets	Количество бакетов, ожидающих синхронизации, и уже синхронизированных.

Параметр	Описание
	• В сценариях горячего резервного копирования ожидающая синхронизация — это количество бакетов, которые основной кластер должен синхронизировать с целевым. • В сценариях активное-активное на уровне города ожидающая синхронизация — это общее количество бакетов, которые необходимо синхронизировать между основным и целевым кластерами.
Objects	Количество объектов, неудачно синхронизированных в бакете. **Примечание**: Это число приведено для справки, так как MinIO синхронизирует связанные конфигурации файлов во время синхронизации.
Network Traffic Rate	Скорость входящего и исходящего сетевого трафика основного кластера. • В сценариях горячего резервного копирования скорость входящего трафика всегда равна 0. • В сценариях активное-активное на уровне города данные есть как по входящему, так и по исходящему трафику.

- Если добавление целевого кластера не удалось, вы можете нажать **Re-add**, чтобы очистить информацию о кластере и вернуться на страницу добавления целевого кластера для повторного добавления.

Связанные операции

Когда необходимость в аварийном восстановлении отпадёт, вы можете нажать **Remove Target Cluster**. Удаление целевого кластера не удаляет уже синхронизированные данные; если в данный момент происходит синхронизация данных, она будет прервана.

Руководство по миграции данных из MinIO в Rook Ceph RGW

Содержание

1. Обзор и архитектура

В этом документе объясняется, как использовать встроенный компонент Rclone в установленном образе оператора **Alauda Build of VolSync** для миграции всех данных из высокодоступного (HA) развертывания MinIO в Kubernetes в Rook Ceph RGW.

1.1 Особенности архитектуры

- **Безопасное повторное использование образа**: данный подход запускает Kubernetes Job, который повторно использует образ оператора `build-harbor.alauda.cn/acp/volsync` с исправлениями безопасности, переопределяет стандартную точку входа и напрямую вызывает внутренний бинарник Rclone для выполнения стандартной синхронизации объектов на уровне API **S3-to-S3**.

1.2 Стратегия синхронизации

Используйте трехфазную стратегию: **«Полная -> Инкрементальная -> Переключение»**:

1. **Начальная синхронизация (Полная синхронизация)**: поддерживайте сервисы в онлайн-режиме при миграции исторических данных.
2. **Инкрементальная синхронизация**: поддерживайте сервисы в онлайн-режиме и запускайте несколько раз для догоняющей синхронизации новых или изменённых данных.
3. **Переключение (Cutover)**: остановите записи, выполните строгие проверки согласованности и завершите окончательное переключение.

---

2. Требования и подготовка

2.1 Проверка установки Alauda VolSync и извлечение версии (Критическое требование)

Перед выполнением данного плана убедитесь, что оператор **"Alauda Build of VolSync"** установлен в namespace `volsync-system`. Версия образа для Job миграции должна строго совпадать с версией запущенного оператора.

Подробные инструкции по установке **Alauda Build of VolSync** см. в разделе Configure PVC DR with VolSync.

Как получить версию: Войдите в консоль управления кластером, перейдите в **Marketplace / OperatorHub**, откройте **Installed Operators**, найдите оператор VolSync и запишите отображаемую версию (например, `v0.8.0` или `v0.9.0`). Сохраните это значение как `<OPERATOR_VERSION>` для дальнейшей настройки.

2.2 Сбор информации о MinIO источнике

- **Endpoint**: внутренний адрес сервиса MinIO (например: `http://minio.tenant-ns.svc:9000`).
- **Учетные данные**: Access Key / Secret Key с правами `Read` и `List` для всех бакетов.

2.3 Создание пользователя Ceph RGW (назначение)

Примените следующий YAML в кластере Rook Ceph для создания выделенного пользователя миграции и предоставления прав на создание бакетов.

```yaml
apiVersion: ceph.rook.io/v1
kind: CephObjectStoreUser
metadata:
  name: volsync-migration-user
  namespace: rook-ceph
spec:
  store: object-store
  displayName: "VolSync Migration Admin"
  capabilities:
    bucket: "*"
```

Для минимизации прав оставьте только разрешения на уровне бакетов и не предоставляйте возможность управления пользователями (`user`).

Извлеките и декодируйте учетные данные Ceph для дальнейшего использования:

```
# Получить Access Key
kubectl -n rook-ceph get secret rook-ceph-object-user-object-store-volsyn
c-migration-user -o jsonpath='{.data.AccessKey}' | base64 -d
# Получить Secret Key
kubectl -n rook-ceph get secret rook-ceph-object-user-object-store-volsyn
c-migration-user -o jsonpath='{.data.SecretKey}' | base64 -d
```

---

3. Конфигурация развертывания (манифесты)

Рекомендуется развертывать задачи миграции в namespace на стороне назначения (Ceph RGW) или в выделенном namespace для операций.

Создайте namespace, используемый обоими манифестами, перед их применением:

```
kubectl create ns migration-ops
```

> **Рекомендация по месту развертывания (Важно)**
> Во время миграции Rclone читает данные локально для обработки, а затем загружает их в целевой S3 кластер. Следовательно, сетевое расположение кластера, в котором выполняется задача миграции, напрямую влияет на пропускную способность и время завершения. Рекомендуется развернуть оператор VolSync/Job миграции в **кластере MinIO или Ceph**, чтобы уменьшить сетевые накладные расходы между кластерами и повысить стабильность передачи.

3.1 Создание секрета конфигурации Rclone

Запишите учетные данные источника и назначения S3 в Secret. **Важно: никогда не добавляйте кавычки вокруг `endpoint` или значений секретов.**

```
apiVersion: v1
kind: Secret
metadata:
  name: volsync-rclone-config-secret
  namespace: migration-ops
type: Opaque
stringData:
  rclone.conf: |
    [source-minio]
    type = s3
    provider = Minio
    env_auth = false
    access_key_id = MINIO_ACCESS_KEY_HERE
    secret_access_key = MINIO_SECRET_KEY_HERE
    endpoint = MINIO_ENDPOINT_HERE
    no_check_certificate = true

    [dest-ceph]
    type = s3
    provider = Ceph
    env_auth = false
    access_key_id = CEPH_ACCESS_KEY_HERE
    secret_access_key = CEPH_SECRET_KEY_HERE
    endpoint = CEPH_ENDPOINT_HERE
    list_version = 2
```

3.2 Определение Job миграции

Разверните Job, который вызывает исправленный образ Alauda VolSync для выполнения синхронизации данных S3.

> **О `remote:bucket[/prefix]` vs `remote:` (Важно)**
>
> - Для S3 backend'ов распространённый паттерн Rclone — `remote:bucket` или `remote:bucket/prefix`, что является самым явным и безопасным способом определения области.
> - В этом документе намеренно используется `source-minio:` -> `dest-ceph:`, чтобы поддержать **полную миграцию на уровне инстанса** (все видимые бакеты со

стороны источника).

- **Предупреждение о рисках**: `remote:` охватывает более широкую область. Если учетные данные имеют избыточные права, могут быть мигрированы бакеты вне предполагаемой области. Всегда проверяйте в тестовой среде и рассмотрите возможность перехода на явный allowlist с `remote:bucket[/prefix]` для финального переключения в продакшене.

Обязательно замените `<OPERATOR_VERSION>` в YAML ниже на фактическую версию, полученную в разделе 2.1.

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: volsync-s3-sync-job
  namespace: migration-ops
spec:
  backoffLimit: 0
  template:
    spec:
      restartPolicy: Never
      containers:
        - name: rclone
          # Используйте исправленный образ volsync от Alauda. Версия должна точно совпадать с OperatorHub.
          image: build-harbor.alauda.cn/acp/volsync:<OPERATOR_VERSION>
          # Примечание: в кластерах ACP ссылка на образ автоматически переписывается
          # на адрес внутреннего реестра после создания Pod
          # (например: registry.alauda.cn:60070/acp/volsync:<OPERATOR_VERSION>).
          # Это ожидаемое поведение.
          # Используйте `kubectl describe pod <pod-name>`, чтобы проверить фактический Image / ImageID.
          imagePullPolicy: IfNotPresent
          # Переопределяем стандартную точку входа и напрямую вызываем rclone внутри образа
          command: ["rclone"]
          args:
            - "sync"
            - "source-minio:"
            - "dest-ceph:"
            - "--progress"
            - "--create-empty-src-dirs"
            - "--ignore-errors"
            # --- Флаги настройки производительности ---
            - "--transfers=32"      # параллельные передачи файлов
            - "--checkers=64"       # параллельные проверки
            - "--s3-chunk-size=64M" # размер чанка для больших объектов, чтобы уменьшить фрагментацию RGW
            - "--fast-list"         # использовать ListObjectsV2 для уменьшения количества API-запросов
            - "--metadata"          # синхронизировать метаданные объекто
в
```

```
        resources:
          requests:
            cpu: "2000m"
            memory: "4Gi"
          limits:
            cpu: "4000m"
            memory: "8Gi"
        env:
          - name: RCLONE_CONFIG
            value: "/config/rclone.conf"
        volumeMounts:
          - name: config-volume
            mountPath: /config
            readOnly: true
      volumes:
        - name: config-volume
          secret:
            secretName: volsync-rclone-config-secret
```

4. Порядок выполнения

Фаза 1: Начальная полная синхронизация

1. Создайте namespace для операций: `kubectl create ns migration-ops`
2. Примените Secret: `kubectl apply -f rclone-secret.yaml`
3. Запустите Job: `kubectl apply -f rclone-job.yaml`
4. Отслеживайте прогресс: `kubectl -n migration-ops logs -f job/volsync-s3-sync-job`

Фаза 2: Инкрементальная синхронизация

Для догоняющей синхронизации новых данных, появившихся во время полной синхронизации, повторяйте этот шаг по мере необходимости.

1. Удалите предыдущий Job: `kubectl -n migration-ops delete job volsync-s3-sync-job`
2. Воссоздайте Job: `kubectl apply -f rclone-job.yaml`

Примечание по механизму: по умолчанию Rclone сравнивает Size и ModTime для решения о необходимости передачи. Существующие неизменённые файлы пропускаются.

Фаза 3: Окончательное переключение

1. **Остановите записи**: прекратите записи в MinIO на уровне приложений или заблокируйте запись через балансировщик нагрузки/Ingress.
2. **Строгая проверка синхронизации**:

- Удалите текущий Job: `kubectl -n migration-ops delete job volsync-s3-sync-job`
- Обновите `rclone-job.yaml`: удалите `- "--ignore-errors"` для финального запуска, затем добавьте `- "--checksum"` в `args`. Это предотвратит маскировку ошибок объектов и отдаст приоритет сравнению Size+Checksum (если доступно) для обнаружения различий.
- **Рекомендация по согласованности (Обязательно)**: не полагайтесь только на количество объектов или ETag. Перед переключением выполните `rclone check source-minio: dest-ceph: --download --checksum` (или выполните выборочные загрузки критичных объектов и вычислите SHA256), чтобы снизить ложные срабатывания из-за multipart/ETag различий.
- Повторно примените Job: `kubectl apply -f rclone-job.yaml`

3. **Проверьте и переключитесь**: после того, как Job завершится со статусом `Completed` и в логах не будет ошибок, обновите S3 Endpoint и учетные данные приложения на Ceph RGW.

---

5. Рекомендации по устранению неполадок и настройке

Симптом	Техническая диагностика	Решение
ImagePullBackOff	Узел не может загрузить образ или существует несоответствие версии. В кластерах ACP `build-harbor.alauda.cn` автоматически переписывается на внутренний реестр.	Сначала выполните `kubectl describe pod <pod-name>`, чтобы проверить фактический `Image` (переписан или нет) и `ImageID`. Затем проверьте `<OPERATOR_VERSION>` и сеть/аутентификацию для фактического реестра (`imagePullSecrets`).
Pod OOMKilled	`--fast-list` загружает метаданные пакетно для сценариев с большим количеством мелких объектов, что потребляет много памяти.	Увеличьте лимит памяти Pod до 8Gi и более или удалите `--fast-list` (это увеличит количество API-запросов и замедлит обход).
403 Forbidden	Пользователь Ceph назначения не имеет прав на создание бакетов.	Проверьте конфигурацию `CephObjectStoreUser` и убедитесь, что в `capabilities` есть `bucket: "*"`.
dial tcp: lookup "http: no such host"	Формат секрета конфигурации неверен; URL endpoint заключён в кавычки.	Отредактируйте Secret и **удалите кавычки вокруг URL endpoint**, обеспечив строгий формат: `endpoint = http://...`.
503 Service Unavailable	Слишком высокая параллельность миграции вызывает чрезмерную нагрузку на запись Ceph RGW.	Уменьшите `--transfers` в аргументах Job или добавьте `--bwlimit` для ограничения пропускной способности передачи.

Локальное хранилище TopoLVM

Введение

Введение

Установка

Установка

Предварительные требования

Процедура

Руководства

Управление устройствами

Предварительные требования

Добавление устройств

Мониторинг и оповещения

Мониторинг

Оповещения

Как сделать

Резервное копирование и вос TopoLVM с помощью Velero

Предварительные требования

Ограничения

Процедура

Удаление дисков, групп томов хранилища ACP TopoLVM

Термины

Выбор правильного сценария

Сценарий 1: Удаление volume group device из device-class

Сценарий 2: Удаление device-class volume group из device

Сценарий 3: Удаление узла хранилища TopoLVM

Настройка

Предварительны

Процедура

Введение

TopoLVM — это плагин Container Storage Interface (CSI), разработанный специально для Kubernetes, предназначенный для эффективного и удобного управления локальными томами хранения.

Основные функции и преимущества:

- **Управление локальными томами**: TopoLVM ориентирован на управление локальными устройствами хранения (такими как диски и SSD) на узлах Kubernetes. По сравнению с традиционным сетевым хранилищем, локальные тома обеспечивают меньшую задержку и более высокую производительность.
- **Осведомленность о топологии**: TopoLVM способен распознавать топологию кластера Kubernetes (например, узлы, зоны доступности), что позволяет автоматически выделять тома хранения на том же узле в соответствии с фактическим местоположением планирования Pod, дополнительно оптимизируя производительность.
- **Динамическое выделение томов**: TopoLVM поддерживает динамическое создание, удаление и изменение размера томов хранения без ручного вмешательства, значительно упрощая операции и снижая сложность.
- **Глубокая интеграция с Kubernetes**: Как плагин CSI, TopoLVM бесшовно интегрируется с API управления хранилищем Kubernetes, позволяя пользователям управлять локальными томами напрямую через стандартные объекты ресурсов Kubernetes, такие как PersistentVolumeClaims.

В итоге, TopoLVM решает распространённые проблемы, связанные с использованием локального хранилища в Kubernetes, такие как ручное управление, отсутствие осведомленности о топологии и недостаточные возможности динамического выделения. Он предоставляет более эффективное и удобное решение для приложений, требующих высокопроизводительного локального хранилища, таких как базы данных и кэши.

Установка

Local storage — это программно-определяемое локальное хранилище на сервере, которое обеспечивает простую, удобную в обслуживании и высокопроизводительную возможность локального хранения данных. Основанное на решении сообщества TopoLVM, оно реализует оркестрацию управления постоянными томами локального хранилища через системный подход LVM.

Содержание

Предварительные требования

- На каждом узле кластера хранения должен быть установлен пакет lvm2. Если он не установлен, выполните команду `yum install -y lvm2` на узле.
- **Скачайте** установочный пакет **Alauda Container Platform Storage Essentials**, соответствующий архитектуре вашей платформы.
- **Загрузите** установочный пакет **Alauda Container Platform Storage Essentials** с помощью механизма Upload Packages.

- **Скачайте** установочный пакет **Alauda Build of TopoLVM**, соответствующий архитектуре вашей платформы.
- **Загрузите** установочный пакет **Alauda Build of TopoLVM** с помощью механизма Upload Packages.

Процедура

1 Развертывание Alauda Container Platform Storage Essentials

1. Войдите в систему, перейдите на страницу **Administrator**.
2. Нажмите **Marketplace** > **OperatorHub**, чтобы перейти на страницу **OperatorHub**.
3. Найдите **Alauda Container Platform Storage Essentials**, нажмите **Install** и перейдите на страницу **Install Alauda Container Platform Storage Essentials**. Параметры конфигурации:

Параметр	Рекомендуемая конфигурация
Channel	Канал по умолчанию — `stable`.
Installation Mode	`Cluster`: Все пространства имён в кластере используют один экземпляр Operator для создания и управления, что снижает использование ресурсов.
Installation Place	Выберите `Recommended`, Namespace поддерживается только **acp-storage**.
Upgrade Strategy	`Manual`: При наличии новой версии в Operator Hub требуется ручное подтверждение для обновления Operator до последней версии.

2 Развертывание хранилища

1. Перейдите в **Administrator**.

2. В левой навигационной панели нажмите **Storage Management** > **Local Storage**.
3. Нажмите **Configure Now**.
4. На странице мастера **Install Operator** нажмите **Start Deployment**.
   - Если страница автоматически переходит к следующему шагу, это означает успешное развертывание Operator.
   - Если развертывание не удалось, следуйте подсказкам интерфейса для решения проблемы. Затем нажмите **Clean Up** и повторно разверните Operator.
5. На странице мастера **Create Cluster** добавьте устройства.

Параметр	Описание
Select Node	Узел с минимум одним неразмеченным диском.
Device Class	Каждый класс устройств соответствует набору устройств хранения с одинаковыми характеристиками. Рекомендуется указывать название, исходя из типа диска, например *hdd*, *ssd*.
Device Type	Поддерживаются только типы дисков.
Storage Device	Например, */dev/sda*. Если дисков несколько, их можно добавлять по одному.
Snapshot	При включении поддерживается создание снимков PVC и использование их для настройки новых PVC для быстрого резервного копирования и восстановления данных. Если при создании хранилища снимок не был включён, его можно активировать при необходимости в разделе **Operations** на странице деталей кластера хранения. **Примечание**: Перед использованием убедитесь, что для текущего кластера развернут Volume Snapshot Plugin.

- Нажмите Далее. Если страница автоматически переходит к следующему шагу, это означает успешное развертывание кластера.
- Если создание не удалось, следуйте подсказкам интерфейса и своевременно очистите ресурсы.

6. На странице мастера **Create Storage Class** настройте соответствующие параметры.

Параметр	Описание
Name	Имя класса хранения. Должно быть уникальным в пределах текущего кластера.
Display Name	Имя, помогающее идентифицировать или фильтровать, например, описание класса хранения на русском языке.
Device Class	Класс устройств — способ категоризации устройств хранения в TopoLVM. Каждый класс устройств соответствует набору устройств с одинаковыми характеристиками. При отсутствии особых требований можно использовать класс устройств **Auto-Allocated** из кластера.
File System	- **XFS** — высокопроизводительная журналируемая файловая система, хорошо справляющаяся с параллельными I/O нагрузками, поддерживающая обработку больших файлов и обеспечивающая плавную передачу данных. - **EXT4** — журналируемая файловая система в Linux, использующая методы хранения с экстентами и поддерживающая обработку больших файлов. Максимальная ёмкость файловой системы — 1 EiB, максимальный размер файла — 16 TiB.
Recycling Policy	Политика переработки для постоянных томов. - Delete: При удалении persistent volume claim связанный persistent volume также удаляется. - Retain: Даже при удалении persistent volume claim связанный persistent volume сохраняется.

Параметр	Описание
Access Mode	ReadWriteOnce (RWO): Может быть смонтирован одним узлом в режиме чтения-записи.
Allocation Project	Этот тип persistent volume claim может быть создан только в определённых проектах. Если проект временно не назначен, его можно **Обновить** позже.

7. Нажмите **Next** и дождитесь завершения создания ресурсов.

Руководства

Управление устройствами

Предварительные требования

Добавление устройств

Мониторинг и оповещения

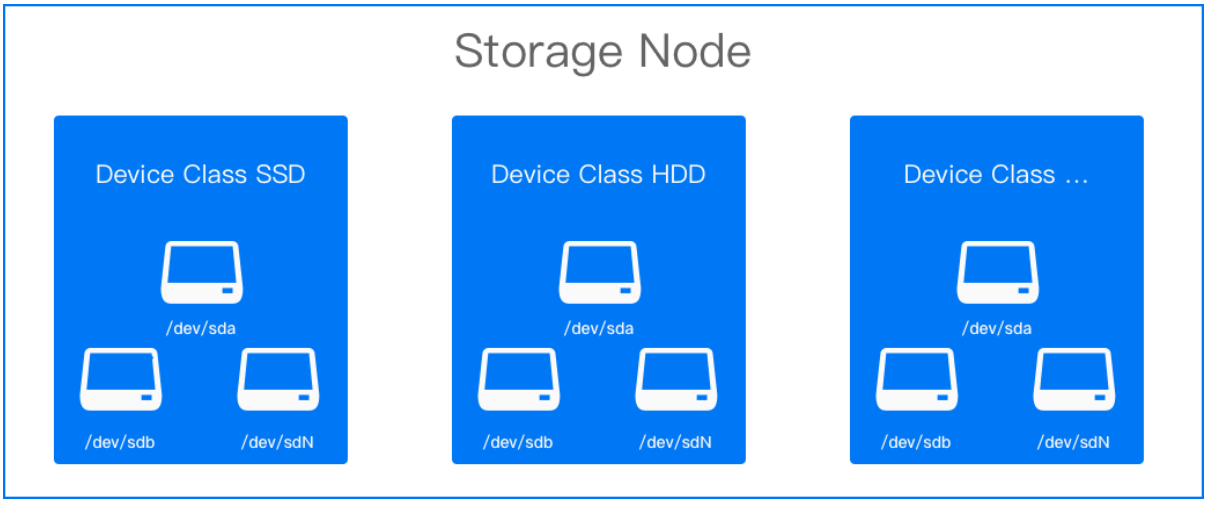
Мониторинг

Оповещения

Управление устройствами

Будь то для первоначального развертывания или расширения ресурсов, необходимо сопоставить доступные диски на узле с устройствами хранения для использования и управления.

Устройства хранения с похожими характеристиками обычно используются централизованно, и такие устройства классифицируются в локальном хранилище под **Device Classes**. Использование device classes эквивалентно прямому использованию дисков, обеспечивая нулевые потери и высокую производительность, а также снижая осведомленность приложений и зависимость от конкретных устройств.



Содержание

Предварительные требования

- При создании кластера локального хранилища должен быть добавлен как минимум 1 Device Class (deviceClasses.classes), включая устройства в этом device class.
- На узле должен присутствовать как минимум 1 голый диск.

Добавление устройств

1. Перейдите в раздел **Administrator**.
2. В левой навигационной панели нажмите **Storage Management** > **Local Storage**.
3. На вкладке **Details** нажмите **Add Storage Node**.
4. Настройте соответствующие параметры согласно приведённым ниже инструкциям.

Параметр	Описание
Storage Node	Узел, на котором имеется как минимум 1 голый диск.
Device Class	Каждый device class соответствует группе устройств хранения с одинаковыми характеристиками; рекомендуется называть его в соответствии с типом дисков, например, *hdd*, *ssd*.
Storage Device	Например, */dev/sda*. Если дисков несколько, их можно добавлять по одному. **Примечание**: Устройство хранения должно быть целым жёстким диском, а не разделом на диске, так как это вызовет ошибки.

5. Нажмите **Add**.
   Примечание: Если статус device class отображается как `Unavailable` из-за отсутствия добавленных устройств, можно продолжить выполнение следующих операций.
6. Перейдите на вкладку **Storage Devices** и нажмите **Add Storage Device**.
7. Добавьте устройства согласно подсказкам на интерфейсе.

8. Нажмите **Add**.

Мониторинг и оповещения

Локальное хранилище предоставляет готовые возможности для сбора метрик мониторинга и настройки оповещений. После включения компонента мониторинга платформы можно настроить мониторинг и оповещения на основе кластеров хранения, производительности и ёмкости хранилища с поддержкой настройки политик уведомлений.

Интуитивно представленные данные мониторинга можно использовать для поддержки принятия решений при операционных проверках или настройке производительности, а комплексный механизм оповещений поможет обеспечить стабильную работу системы хранения.

Содержание

Мониторинг

Мониторинг производительности

По умолчанию платформа собирает часто используемые метрики мониторинга производительности, такие как пропускная способность чтения и записи, IOPS и задержки для локального хранилища. Данные мониторинга в реальном времени по этим метрикам можно просмотреть на вкладке **Monitoring** страницы **Local Storage** в разделе **Storage Management**. Платформа визуально отображает эти метрики с помощью графиков и диаграмм, что позволяет администраторам чётко наблюдать текущую производительность хранилища и быстро выявлять потенциальные проблемы.

Мониторинг ёмкости

Поскольку локальное хранилище может использовать только локально доступные ресурсы хранения на узлах, пользователи должны убедиться в наличии достаточного объёма свободной ёмкости на узлах перед объявлением локального хранилища, чтобы избежать проблем, вызванных чрезмерным объявлением.

Для помощи в этом платформа предоставляет подробный мониторинг ёмкости в разделе **Details** локального хранилища, классифицированный по типам устройств. Пользователи могут проверить доступное пространство для хранения, чётко отображаемое в числовом и графическом форматах. Если у какого-либо типа устройств наблюдается недостаток доступной ёмкости, следует освободить место или добавить дополнительные дисковые устройства перед использованием локального хранилища.

Оповещения

Платформа включает набор предустановленных политик оповещений. Если ресурсы становятся аномальными или данные мониторинга достигают порога предупреждения, оповещения автоматически срабатывают. Преднастроенные политики оповещений эффективно покрывают распространённые операционные потребности, включая оповещения о состоянии здоровья кластера и ёмкости по типам устройств.

Настройка уведомлений

Для своевременного получения оповещений следует настроить политики уведомлений в центре операций. Уведомления могут отправляться по электронной почте, SMS или другими способами соответствующим сотрудникам, что побуждает к немедленному реагированию для устранения проблем или предотвращения сбоев. Пользователи могут получить доступ к настройкам политик уведомлений напрямую из интерфейса центра операций. Подробные инструкции по настройке оповещений доступны в документации [Creating Alert Policies].

Обработка оповещений

- Если состояние здоровья кластера хранения изменяется на `Alert`, администраторы должны немедленно провести расследование. Раздел **Details** предоставляет информацию для устранения и решения этих проблем. Распространённые причины включают аномалии в службах узлов или проблемы с конкретными типами устройств.

Пункт проверки	Соответствующее состояние	Причина
Состояние здоровья	Alert	Вызвано аномалиями в службах узлов или проблемами с типами устройств.
Состояние службы	Unknown	Узел находится в состоянии `notready`, возможно из-за сбоев сети или отключения питания.
Состояние типа устройства	Unavailable	Используемый диск может не быть raw-диском или отсутствовать.

- Оповещения в реальном времени, срабатывающие на вкладке **Alert**, требуют оперативного внимания, даже если текущее состояние кластера хранения отображается как `Healthy`. Быстрая реакция предотвращает развитие более серьёзных проблем. В следующей таблице приведены уровни оповещений и их значение:

Уровень оповещения	Значение
Critical	Указывает на серьёзные проблемы, вызывающие перебои в работе платформы или потерю данных с тяжёлыми последствиями.
Major	Известные проблемы, которые могут повлиять на функциональность платформы и нормальное ведение бизнеса.
Warning	Существует риск операционных проблем; требуется своевременное вмешательство, чтобы избежать влияния на нормальную работу.

Анализ после инцидента

В журнале **Alert History** фиксируются все ранее сработавшие оповещения, которые больше не требуют немедленных действий. При проведении анализа после инцидента следует учитывать следующее:

- Какие конкретные аномалии наблюдались во время инцидента?
- Есть ли повторяющиеся шаблоны определённых оповещений? Как можно проактивно предотвратить их в будущем?
- Был ли всплеск оповещений в определённые периоды, связанный с внешними факторами или операционными инцидентами? Следует ли скорректировать операционные стратегии соответственно?

Как сделать

Резервное копирование и вос TopoLVM с помощью Velero

Предварительные требования

Ограничения

Процедура

Удаление дисков, групп томов хранилища ACP TopoLVM

Термины

Выбор правильного сценария

Сценарий 1: Удаление volume group device из device-class

Сценарий 2: Удаление device-class volume group из device

Сценарий 3: Удаление узла хранилища TopoLVM

Настройка

Предварительны

Процедура

Резервное копирование и восстановление PVC файловой системы TopoLVM с помощью Velero

Velero позволяет выполнять резервное копирование и восстановление Persistent Volume Claims (PVC) и Persistent Volumes (PV) для файловых систем TopoLVM. Эта функциональность интегрирована в платформу.

Данное руководство применимо исключительно к PVC файловой системы TopoLVM.

Содержание

Предварительные требования

1. Разверните «Alauda Container Platform Data Backup for Velero» через Marketplace/Cluster Plugins.
2. Настройте S3-совместимое хранилище для `BackupStorageLocation` Velero. Используйте предоставленное платформой объектное хранилище Ceph или MinIO.

Ограничения

1. В S3-хранилище должно быть достаточно свободного места для хранения всех данных PV из целевого кластера.
2. При восстановлении квота namespace и класс хранения должны поддерживать общий объем всех PVC.

Процедура

Шаг 1: Настройка репозитория резервных копий

1. Убедитесь, что доступно S3-совместимое хранилище.
2. Перейдите в **Administrator** > **Cluster Management** > **Backup and Restore** > **Backup Repository**.
3. Создайте репозиторий резервных копий, используя учетные данные объектного хранилища.

Шаг 2: Выполнение резервного копирования

1. Пометьте PVC и связанные с ними pod’ы, которые необходимо сохранить:
   Velero для восстановления PVC файловой системы требует pod, который монтирует PVC для импорта данных; без pod PVC останется в состоянии Pending. Для сложных приложений приостановите работу приложения и присоедините PVC к легковесному pod (например, Nginx) для резервного копирования/восстановления, затем после восстановления верните исходную конфигурацию приложения.

```
kubectl label pvc -n <namespace> <pvc-name> velero-backup=true
kubectl label pod -n <namespace> <pod-name> velero-backup=true
```

2. Перейдите в **Backup and Restore** и создайте новую резервную копию:

   - Выберите **Backup Kubernetes Resources and PVC Data Volumes**.
   - Укажите namespaces, содержащие данные для резервного копирования.
   - Настройте резервное копирование с параметрами:

```
apiVersion: velero.io/v1
kind: Schedule
metadata:
  name: <backup-name>
  namespace: cpaas-system
  annotations:
    cpaas.io/description: ''
spec:
  template:
    includedNamespaces:
      - <namespace>
    includedResources:
      - '*'
    labelSelector:
      matchLabels:
        velero-backup: 'true'
    excludedNamespaces: []
    excludedResources: []
    defaultVolumesToFsBackup: true
    storageLocation: default
    ttl: 720h
  schedule: '@every 876000h'
  skipImmediately: false
status:
  phase: Enabled
```

3. После завершения резервного копирования проверьте данные в S3-бакете (например, MinIO):

```
mc ls <minio-alias>/<bucket-name>/<backup-path>/<namespace>/
```

Пример вывода:

```
[2025-03-14 00:18:33 CST]  155B STANDARD config
[2025-03-14 09:04:56 CST]   0B data/
[2025-03-14 09:04:56 CST]   0B index/
[2025-03-14 09:04:56 CST]   0B keys/
[2025-03-14 09:04:56 CST]   0B snapshots/
```

Шаг 3: Восстановление кластера

1. В целевом кластере настройте тот же S3-бакет, который использовался для резервного копирования. Velero автоматически обнаружит существующую резервную копию.
2. Перейдите в **Backup and Restore** и создайте задачу восстановления:
   - Выберите namespace(ы) для восстановления.
   - В расширенных настройках при необходимости сопоставьте исходный namespace с целевым.
3. Запустите операцию восстановления.
4. После восстановления проверьте:
   - Совпадение имен PVC с исходным кластером.
   - Целостность и доступность данных приложений в PVC.

Удаление дисков, групп томов device-class или узлов из локального хранилища ACP TopoLVM

В этом документе описывается, как удалить неисправные диски, удалить группы томов device-class из device class и удалить узлы хранилища в локальном хранилище ACP TopoLVM.

В зависимости от области сбоя, в документе рассматриваются следующие сценарии:

- Удаление `volume group device` из `device-class volume group`
- Удаление `device-class volume group` из `device class`
- Удаление узла хранилища TopoLVM

Предупреждение о рисках

- Процедура в этом документе напрямую удаляет ресурсы хранилища, такие как PVC, PV и `logicalvolumes.topolvm.cybozu.com`. После удаления этих ресурсов данные на затронутых томах обычно невозможно восстановить. Сделайте резервную копию данных заранее.
- Перед началом убедитесь, что вы правильно определили диск, группу томов device-class или узел, и запланировали окно обслуживания для затронутых рабочих нагрузок.

Содержание

Термины

Термин	Описание
device class	Логический класс хранилища, состоящий из одной или нескольких групп томов device-class на разных узлах.
device-class volume group	`LVM volume group` на узле, представляющая ресурсы хранилища device class на этом узле.
volume group device	Диск на узле. В LVM соответствует физическому тому (physical volume).

Выбор правильного сценария

Используйте следующую таблицу, чтобы определить, какой сценарий применим к вашей среде.

Условие	Сценарий 1: Удаление volume group device	Сценарий 2: Удаление device-class volume group	Сценарий 3: Удаление узла хранилища TopoLVM
Доступность узла	Узел всё ещё доступен.	Узел всё ещё доступен.	Узел больше не подлежит восстановлению или вы решили навсегда удалить его из TopolvmCluster.
Статус группы томов	Целевая `LVM volume group` всё ещё существует и распознаётся.	Целевая `LVM volume group` больше не существует или не распознаётся, но узел сохраняется.	Узел и все группы томов device-class на этом узле будут удалены вместе.
Область удаления	Удаление одного или нескольких неисправных дисков из группы томов.	Удаление одной группы томов device-class из сохраняемого узла.	Удаление всего узла из TopolvmCluster.
План сохранения	Узел и группа томов device-class сохраняются.	Узел сохраняется, но целевая группа томов device-class не сохраняется.	Ни узел, ни какие-либо группы томов device-class на этом узле не сохраняются.

Сценарий 1: Удаление volume group device из device-class volume group

Сценарий пользователя

- Используйте эту процедуру, когда `device-class volume group` не полностью повреждена, но один или несколько `volume group devices` в группе томов вышли из строя и должны быть удалены.

Предварительные условия

- Целевой узел всё ещё в кластере и доступен.
- `Provision Type` целевого device class — `Thick`.
- Целевая группа томов device-class не полностью повреждена, и в группе томов остаётся хотя бы один исправный volume group device.

Проверка возможности восстановления device-class volume group

Выполните следующую команду на целевом узле:

```
vgs <vg-name>
```

Параметры:

- `<vg-name>`: имя целевой LVM volume group.

Если команда возвращает нормальный вывод или сообщает только о пропавших некоторых PV, при этом группа томов всё ещё существует, значит device-class volume group не полностью повреждена, и вы можете следовать этой процедуре.

Пример вывода:

```
WARNING: Couldn't find device with uuid VJes6j-2a8V-8Cxf-eW84-yEJK-K24A-y
Bc9OD.
WARNING: VG hdd-2ab8f0a2-7d1d-42d7-ba6b-da94c6185c33 is missing PV VJes6j
-2a8V-8Cxf-eW84-yEJK-K24A-yBc9OD (last written to /dev/vdb).
WARNING: Couldn't find all devices for LV hdd-2ab8f0a2-7d1d-42d7-ba6b-da9
4c6185c33/b6b331f0-5242-420f-a531-df90628bef80 while checking used and as
sumed devices.
```

Процедура

Шаг 1: Остановите topolvm-operator

Выполните следующую команду на управляющем узле, чтобы предотвратить согласование ресурсов оператором во время очистки:

```
kubectl -n nativestor-system scale --replicas 0 deployment topolvm-op
erator
```

Шаг 2: Найдите затронутый LVM логический том

Выполните следующую команду на целевом узле, чтобы найти логические тома, использующие неисправные диски:

```
lvs -a -o +devices <vg-name> | egrep "<path-to-disks>|unknown device"
| awk '{print $1}'
```

Параметры:

- `<vg-name>`: имя целевой LVM volume group.
- `<path-to-disks>`: пути к устройствам неисправных дисков, например `/dev/vdb`. Для нескольких дисков разделяйте через `|`.

Например:

```
lvs -a -o +devices hdd-2ab8f0a2-7d1d-42d7-ba6b-da94c6185c33 2>/dev/nu
ll | egrep "/dev/vdb|unknown" | awk '{print $1}'
```

Пример вывода:

```
b6b331f0-5242-420f-a531-df90628bef80
```

Шаг 3: Найдите затронутые PVC и PV

Выполните следующую команду на управляющем узле, чтобы найти связанные PV и PVC по имени логического тома:

```
kubectl get pv -o json | jq -r --arg HANDLE <lv-name> '
  .items[]
  | select(.spec.csi.volumeHandle == $HANDLE)
  | [.metadata.name, .spec.claimRef.namespace, .spec.claimRef.name]
  | @tsv
'
```

Параметры:

- `<lv-name>`: имя затронутого LVM логического тома.

Пример вывода:

```
pvc-e11f6c18-0e15-4c70-9a24-e7136fabfb2f    demo-space  pvc-topolvm
```

В выводе:

- Колонка 1 — имя `PersistentVolume`.
- Колонка 2 — namespace `PersistentVolumeClaim`.
- Колонка 3 — имя `PersistentVolumeClaim`.

Шаг 4: Остановите рабочие нагрузки, использующие затронутые PVC

После определения затронутых PVC остановите рабочие нагрузки, которые их используют, и убедитесь, что все связанные Pod остановлены, прежде чем продолжить.

Шаг 5: Удалите затронутые ресурсы хранилища Kubernetes

Выполните следующие команды на управляющем узле:

```
kubectl delete pvc -n <pvc-namespace> <pvc-name>
kubectl delete pv <pv-name>
kubectl delete logicalvolumes.topolvm.cybozu.com <logicalvolume-name>
```

Параметры:

- `<pvc-namespace>`: namespace затронутого PVC.
- `<pvc-name>`: имя затронутого PVC.
- `<pv-name>`: имя затронутого PV.
- `<logicalvolume-name>`: имя ресурса TopoLVM `logicalvolumes.topolvm.cybozu.com`. Совпадает с `<pv-name>`.

Если в результате запроса несколько ресурсов, удаляйте их по одному согласно сопоставлению.

Если ресурс не удаётся удалить обычным способом, добавьте `--force` по необходимости.

Шаг 6: Очистите остаточные LVM логические тома

Выполните следующую команду на целевом узле, чтобы проверить, остались ли логические тома:

```
lvs -a -o +devices <vg-name> 2>/dev/null | egrep "<path-to-disks>|unk
nown device" | awk '{print $1}'
```

Параметры:

- `<vg-name>`: имя целевой LVM volume group.
- `<path-to-disks>`: пути к устройствам неисправных дисков, например `/dev/vdb`. Для нескольких дисков разделяйте через `|`.

Если команда возвращает вывод, удалите оставшиеся логические тома по одному:

```
lvremove <vg-name>/<lv-name>
```

Параметры:

- `<vg-name>`: имя целевой LVM volume group.
- `<lv-name>`: имя логического тома для удаления.

При необходимости добавьте `--force`.

Шаг 7: Удалите отсутствующий физический том из группы томов LVM

Выполните следующую команду на целевом узле:

```
vgreduce --removemissing <vg-name>
```

Параметры:

- `<vg-name>`: имя целевой LVM volume group.

При необходимости добавьте `--force`.

Шаг 8: Обновите ресурс TopolvmCluster

Выполните следующую команду на управляющем узле:

```
kubectl -n nativestor-system edit topolvmclusters.topolvm.cybozu.com topolvm
```

В редакторе удалите неисправный volume group device из списка `devices` целевого узла. Например, удалите `/dev/vdb` из конфигурации для `nodeName: 192.168.133.50`.

До:

```
spec:
  storage:
    deviceClasses:
      - classes:
          - className: hdd
            default: true
            devices:
              - name: /dev/vdb
                type: disk
              - name: /dev/vdc
                type: disk
            volumeGroup: hdd-2ab8f0a2-7d1d-42d7-ba6b-da94c6185c33
        nodeName: 192.168.133.50
```

После:

```
spec:
  storage:
    deviceClasses:
      - classes:
          - className: hdd
            default: true
            devices:
              - name: /dev/vdc
                type: disk
            volumeGroup: hdd-2ab8f0a2-7d1d-42d7-ba6b-da94c6185c33
        nodeName: 192.168.133.50
```

Шаг 9: Обновите ConfigMap lvmdconfig

Выполните следующую команду на управляющем узле:

```
kubectl -n nativestor-system edit configmaps lvmdconfig-<node-name>
```

Параметры:

- `<node-name>`: имя целевого узла.

В редакторе удалите статус неисправного volume group device из `status.json`. Например, удалите запись `deviceStates` для `/dev/vdb`.

До:

```
status.json: '{"node":"192.168.133.50","phase":"","failClasses":[],"successClasses":[{"className":"hdd","vgName":"hdd-2ab8f0a2-7d1d-42d7-ba6b-da94c6185c33","state":"Ready","message":"create successful","deviceStates":[{"name":"/dev/vdb","state":"Online"},{"name":"/dev/vdc","state":"Online"}]}],"loops":[],"raids":[]}'
```

После:

```
status.json: '{"node":"192.168.133.50","phase":"","failClasses":[],"successClasses":[{"className":"hdd","vgName":"hdd-2ab8f0a2-7d1d-42d7-ba6b-da94c6185c33","state":"Ready","message":"create successful","deviceStates":[{"name":"/dev/vdc","state":"Online"}]}],"loops":[],"raids":[]}'
```

Шаг 10: Запустите topolvm-operator

Выполните следующую команду на управляющем узле:

```
kubectl -n nativestor-system scale --replicas 1 deployment topolvm-operator
```

Шаг 11: Проверьте, что volume group device удалён

Выполните следующую команду на управляющем узле:

```
kubectl -n nativestor-system get topolvmclusters.topolvm.cybozu.com topolvm -o jsonpath="{.status.nodeStorageState}" | jq
```

Убедитесь, что удалённый volume group device больше не отображается в `deviceStates` целевого узла.

Сценарий 2: Удаление device-class volume group из device class

Сценарий пользователя

- Используйте эту процедуру, когда группа томов device-class на узле полностью повреждена и не может быть восстановлена удалением только одного volume group device.

Предварительные условия

- Целевой узел всё ещё в кластере и доступен.
- Целевая группа томов device-class полностью повреждена.
- После удаления целевой группы томов device-class на узле остаётся хотя бы одна другая группа томов device-class.

Проверка полной повреждённости device-class volume group

Выполните следующую команду на целевом узле:

```
vgs
```

Если целевая `LVM volume group` больше не отображается в выводе, значит device-class volume group полностью повреждена, и вы можете следовать этой процедуре.

> **Примечание**
>
> В этом сценарии удаляется вся группа томов device-class с узла, а не только один volume group device в этой группе.

Процедура

Шаг 1: Остановите topolvm-operator

Выполните следующую команду на управляющем узле:

```
kubectl -n nativestor-system scale --replicas 0 deployment topolvm-op
erator
```

Шаг 2: Найдите затронутые PVC и PV

Выполните следующую команду на управляющем узле, чтобы найти PV, PVC и ресурсы `logicalvolumes.topolvm.cybozu.com`, связанные с указанным классом хранилища на целевом узле:

```
kubectl get pv -o json | jq -r --arg NODE <node-name> --arg SC <stora
geclass-name> '
  .items[]
  | select(.spec.nodeAffinity.required.nodeSelectorTerms[]?.matchExpr
essions[]? | select(.key=="topology.topolvm.cybozu.com/node") | .valu
es[]? == $NODE)
  | select(.spec.storageClassName == $SC)
  | [.metadata.name, .spec.claimRef.namespace, .spec.claimRef.name]
  | @tsv
'
```

Параметры:

- `<node-name>`: имя целевого узла.
- `<storageclass-name>`: имя класса хранилища, связанного с целевой группой томов device-class. Если задействовано несколько классов, выполните запрос отдельно для каждого.

Пример вывода:

```
pvc-e11f6c18-0e15-4c70-9a24-e7136fabfb2f    demo-space  pvc-topolvm
```

В выводе:

- Колонка 1 — имя и `PersistentVolume`, и ресурса `logicalvolumes.topolvm.cybozu.com`.
- Колонки 2 и 3 — namespace и имя `PersistentVolumeClaim`.

Если с целевой группой томов device-class связаны несколько классов хранилища, повторите этот запрос и последующие шаги очистки для каждого класса.

Шаг 3: Остановите рабочие нагрузки, использующие затронутые PVC

После определения затронутых PVC остановите рабочие нагрузки, которые их используют, и убедитесь, что все связанные Pod остановлены, прежде чем продолжить.

Шаг 4: Удалите затронутые ресурсы хранилища Kubernetes

Выполните следующие команды на управляющем узле:

```
kubectl delete pvc -n <pvc-namespace> <pvc-name>
kubectl delete pv <pv-name>
kubectl delete logicalvolumes.topolvm.cybozu.com <logicalvolume-name>
```

Параметры:

- `<pvc-namespace>`: namespace затронутого PVC.
- `<pvc-name>`: имя затронутого PVC.
- `<pv-name>`: имя затронутого PV.
- `<logicalvolume-name>`: имя ресурса TopoLVM `logicalvolumes.topolvm.cybozu.com`. Совпадает с `<pv-name>`.

Если в результате запроса несколько ресурсов, удаляйте их по одному согласно сопоставлению.

Если ресурс не удаётся удалить обычным способом, добавьте `--force` по необходимости.

Шаг 5: Обновите ресурс TopolvmCluster

Выполните следующую команду на управляющем узле:

```
kubectl -n nativestor-system edit topolvmclusters.topolvm.cybozu.com
topolvm
```

В редакторе удалите группу томов device-class с целевого узла. Например, в конфигурации для `nodeName: 192.168.140.13` удалите запись `className: hdd` и связанные с ней `volumeGroup` и `devices`.

До:

```
spec:
  storage:
    deviceClasses:
      - classes:
          - className: ssd
            default: true
            devices:
              - name: /dev/vdc
                type: disk
            volumeGroup: ssd-4a8737fc-48d3-4c61-882d-0a5bcc6f77a1
          - className: hdd
            devices:
              - name: /dev/vdb
                type: disk
            volumeGroup: hdd-97dc00f3-1df6-4f64-8ddc-7b0b6c5d6de5
        nodeName: 192.168.140.13
```

После:

```
spec:
  storage:
    deviceClasses:
      - classes:
          - className: ssd
            default: true
            devices:
              - name: /dev/vdc
                type: disk
            volumeGroup: ssd-4a8737fc-48d3-4c61-882d-0a5bcc6f77a1
        nodeName: 192.168.140.13
```

Если удалённая запись `className` имела `default: true`, назначьте другой оставшийся класс как класс по умолчанию.

Шаг 6: Обновите ConfigMap lvmdconfig

Выполните следующую команду на управляющем узле:

```
kubectl -n nativestor-system edit configmaps lvmdconfig-<node-name>
```

Параметры:

- `<node-name>`: имя целевого узла.

В редакторе удалите конфигурацию, соответствующую целевой группе томов device-class, из `lvmd.yaml` и `status.json`. Например, удалите конфигурацию для `className: hdd`.

До:

```
lvmd.yaml: |
  socket-name: /run/topolvm/lvmd.sock
  device-classes:
  - name: ssd
    volume-group: ssd-4a8737fc-48d3-4c61-882d-0a5bcc6f77a1
    default: true
    type: thick
  - name: hdd
    volume-group: hdd-97dc00f3-1df6-4f64-8ddc-7b0b6c5d6de5
    default: false
    type: thick
status.json: '{"node":"192.168.140.13","phase":"","failClasses":[],"s
uccessClasses":[{"className":"hdd","vgName":"hdd-97dc00f3-1df6-4f64-8
ddc-7b0b6c5d6de5","state":"Ready","message":"create successful","devi
ceStates":[{"name":"/dev/vdb","state":"Online"}]},{"className":"ss
d","vgName":"ssd-4a8737fc-48d3-4c61-882d-0a5bcc6f77a1","state":"Read
y","message":"create successful","deviceStates":[{"name":"/dev/vd
c","state":"Online"}]}],"loops":[],"raids":[]}'
```

После:

```
lvmd.yaml: |
  socket-name: /run/topolvm/lvmd.sock
  device-classes:
  - name: ssd
    volume-group: ssd-4a8737fc-48d3-4c61-882d-0a5bcc6f77a1
    default: true
    type: thick
status.json: '{"node":"192.168.140.13","phase":"","failClasses":[],"s
uccessClasses":[{"className":"ssd","vgName":"ssd-4a8737fc-48d3-4c61-8
82d-0a5bcc6f77a1","state":"Ready","message":"create successful","devi
ceStates":[{"name":"/dev/vdc","state":"Online"}]}],"loops":[],"raid
s":[]}'
```

Если удалённая запись `className` имела `default: true`, назначьте другой оставшийся класс как класс по умолчанию.

Шаг 7: Запустите topolvm-operator

Выполните следующую команду на управляющем узле:

```
kubectl -n nativestor-system scale --replicas 1 deployment topolvm-op
erator
```

Шаг 8: Проверьте, что device-class volume group удалена

Выполните следующую команду на управляющем узле:

```
kubectl -n nativestor-system get topolvmclusters.topolvm.cybozu.com t
opolvm -o jsonpath="{.status.nodeStorageState}" | jq
```

Убедитесь, что удалённая группа томов device-class больше не отображается на целевом узле, а оставшиеся группы томов device-class находятся в здоровом состоянии.

Сценарий 3: Удаление узла хранилища TopoLVM

Сценарий пользователя

- Целевой узел больше не подлежит восстановлению, или вы решили навсегда удалить его из TopolvmCluster.

Предварительные условия

- Вы решили не сохранять никакие группы томов device-class на целевом узле.
- Рабочие нагрузки, использующие локальные тома на целевом узле, остановлены или перенесены на другие узлы.
- Оставшиеся узлы могут продолжать размещать необходимые рабочие нагрузки после удаления целевого узла.

Процедура

Шаг 1: Остановите topolvm-operator

Выполните следующую команду на управляющем узле:

```
kubectl -n nativestor-system scale --replicas 0 deployment topolvm-op
erator
```

Шаг 2: Найдите затронутые PVC и PV

Выполните следующую команду на управляющем узле, чтобы найти PV, PVC и ресурсы `logicalvolumes.topolvm.cybozu.com`, связанные с целевым узлом:

```
kubectl get pv -o json | jq -r --arg NODE <node-name> '
  .items[]
  | select(.spec.nodeAffinity.required.nodeSelectorTerms[]?.matchExpr
essions[]? | select(.key=="topology.topolvm.cybozu.com/node") | .valu
es[]? == $NODE)
  | [.metadata.name, .spec.claimRef.namespace, .spec.claimRef.name]
  | @tsv
'
```

Параметры:

- `<node-name>`: имя целевого узла.

Шаг 3: Остановите рабочие нагрузки, использующие затронутые PVC

После определения затронутых PVC остановите рабочие нагрузки, которые их используют, и убедитесь, что все связанные Pod остановлены, прежде чем продолжить.

Шаг 4: Удалите затронутые ресурсы хранилища Kubernetes

Выполните следующие команды на управляющем узле:

```
kubectl delete pvc -n <pvc-namespace> <pvc-name>
kubectl delete pv <pv-name>
kubectl delete logicalvolumes.topolvm.cybozu.com <logicalvolume-name>
```

Параметры:

- `<pvc-namespace>`: namespace затронутого PVC.
- `<pvc-name>`: имя затронутого PVC.
- `<pv-name>`: имя затронутого PV.
- `<logicalvolume-name>`: имя ресурса TopoLVM `logicalvolumes.topolvm.cybozu.com`. Совпадает с `<pv-name>`.

Если в результате запроса несколько ресурсов, удаляйте их по одному согласно сопоставлению.

Если ресурс не удаётся удалить обычным способом, добавьте `--force` по необходимости.

Шаг 5: Обновите ресурс TopolvmCluster

Выполните следующую команду на управляющем узле:

```
kubectl -n nativestor-system edit topolvmclusters.topolvm.cybozu.com topolvm
```

В редакторе удалите весь блок конфигурации для целевого узла. Например, удалите блок для `nodeName: 192.168.140.13`.

До:

```
spec:
  storage:
    deviceClasses:
      - classes:
          - className: hdd
            default: true
            devices:
              - name: /dev/vdc
                type: disk
            volumeGroup: hdd-2ab8f0a2-7d1d-42d7-ba6b-da94c6185c33
        nodeName: 192.168.133.50
      - classes:
          - className: ssd
            default: true
            devices:
              - name: /dev/vdc
                type: disk
            volumeGroup: ssd-4a8737fc-48d3-4c61-882d-0a5bcc6f77a1
          - className: hdd
            devices:
              - name: /dev/vdb
                type: disk
            volumeGroup: hdd-97dc00f3-1df6-4f64-8ddc-7b0b6c5d6de5
        nodeName: 192.168.140.13
```

После:

```
spec:
  storage:
    deviceClasses:
      - classes:
          - className: hdd
            default: true
            devices:
              - name: /dev/vdc
                type: disk
            volumeGroup: hdd-2ab8f0a2-7d1d-42d7-ba6b-da94c6185c33
        nodeName: 192.168.133.50
```

Шаг 6: Обновите ConfigMap lvmdconfig

Выполните следующую команду на управляющем узле:

```
kubectl -n nativestor-system edit configmaps lvmdconfig-<node-name>
```

Параметры:

- `<node-name>`: имя целевого узла.

В редакторе удалите полностью секции `lvmd.yaml` и `status.json`. Не оставляйте никакой конфигурации или статуса для удалённого узла.

Шаг 7: Запустите topolvm-operator

Выполните следующую команду на управляющем узле:

```
kubectl -n nativestor-system scale --replicas 1 deployment topolvm-op
erator
```

Шаг 8: Проверьте, что узел удалён

Выполните следующую команду на управляющем узле:

```
kubectl -n nativestor-system get topolvmclusters.topolvm.cybozu.com t
opolvm -o jsonpath="{.status.nodeStorageState}" | jq
```

Убедитесь, что целевой узел больше не отображается в `status.nodeStorageState`. Например, `192.168.140.13` не должен появляться в выводе.

Настройка полосатых логических томов

Когда вы записываете данные в логический том LVM, файловая система распределяет данные по базовым физическим томам. Вы можете контролировать способ записи данных на физические тома, создав полосатый логический том. Для больших последовательных операций чтения и записи это может повысить эффективность ввода-вывода данных.

Полосатость улучшает производительность за счёт записи данных на заранее определённое количество физических томов поочерёдно. При полосатости ввод-вывод может выполняться параллельно. В некоторых случаях это может привести к почти линейному увеличению производительности с каждым дополнительным физическим томом в полосе.

TopoLVM достигает этого, задавая параметр `lvcreate-option-class` в `StorageClass`.

Содержание

Предварительные требования

- Класс устройств должен содержать **не менее двух устройств** на одном узле.

Процедура

1 Использование параметра `lvcreate-option-class` по умолчанию

1. Перейдите в раздел **Administrator**.
2. В левой боковой панели выберите **Storage Management** > **Storage Classes**.
3. Нажмите **Create Storage Class**.
4. Выберите **Block Storage**.
5. Выберите **TopoLVM**, затем нажмите **Next**.
6. Настройте необходимые параметры класса хранения.
7. Переключитесь в **YAML view** и добавьте параметр `topolvm.io/lvcreate-option-class`:

```
parameters:
  topolvm.io/lvcreate-option-class: striped-default
```

> **NOTE**
>
> `striped-default` — встроенный lvcreate-option-class в Alauda Container Platform. Он автоматически добавляет `--stripes=2` и `--stripesize=64` к команде lvcreate при создании логических томов драйвером TopoLVM CSI.

2 Создание пользовательского lvcreate-option-class (необязательно)

Если встроенный класс striped-default не соответствует вашим требованиям, вы можете определить собственный LVCreateOptionClass:

```
cat << EOF | kubectl apply -f -
apiVersion: topolvm.cybozu.com/v2
kind: LVCreateOption
metadata:
  name: custom
  namespace: nativestor-system
spec:
  options:
  - name: striped-custom
    type: striped
    options:
    - --stripes=3
    - --stripesize=64
EOF
```

Этот манифест создаёт пользовательский LVCreateOptionClass с именем striped-custom, который задаёт 3 полосы и размер полосы 64 КиБ. После применения вы можете ссылаться на него в вашем StorageClass с помощью `topolvm.io/lvcreate-option-class: striped-custom`.