Storage

Ceph Distributed Storage

Introduction

Feature Overview

Storage Solution Comparison

Install

Concepts

Architectur

Technical archit

Guides

How To

MinIO Object Storage

Introduction

Install

Prerequisites

Architectur

Core Compone

t Arc

Expa

Conclusion:

Concepts

Guides

How To

TopoLVM Local Storage

Ceph Distributed Storage

Introduction

Install

Architecture

Architecture

Technical architecture

Concepts

Core Concepts

Rook Operator

Ceph CSI

Ceph module functions

Guides

Accessing Storage Services

Prerequisites

Procedure

Follow-up Actions

Managing Storage Pools

Creating a Storage Pool

Deleting a Storage Pool

Viewing Object Storage Pool Addresses

Node-speci

Update Compo

Restart Storage

Monitoring and Alerts

Monitoring

Alerts

Dev

ice

ices

Statu

How To

Introduction

Alauda Build of Rook-Ceph is a hyper-converged storage solution provided by the platform within the cluster. Based on the open-source Rook + Ceph storage solution, distributed storage achieves automatic management, automatic scaling, and automatic repair capabilities, fulfilling the block storage, file storage, and object storage needs of small to medium-sized applications.

> **NOTE**
>
> In this document, **distributed storage** refers to the Ceph storage within this cluster, while **external storage** refers to Ceph storage outside of this cluster.

TOC

Feature Overview

- **Easy Deployment**: Provides graphical automatic deployment and management services for storage clusters; supports both integrated and decoupled deployment modes for compute and storage.

- **Professional Operations**: Offers persistent volume snapshot backup and clone new volume functionalities; visual monitoring of capacity, performance, and component levels; equipped with built-in alert policies to meet the needs of most storage operation scenarios.
- **Secure and Reliable**: Distributed and multi-replica mechanisms ensure data security and reliability; simple and reliable automated management supports online expansion of storage resources.
- **Excellent Performance**: Provides elastic and high-performance storage services; supports the deployment of hybrid disk devices to enhance storage system performance and efficiency.

Storage Solution Comparison

The platform supports the following two types of storage solutions; you can choose one or the other.

Creating a Storage Cluster

Requirement	Advantages
You can choose to create either a standard type cluster **or an** extended type cluster	No need for additional storage solution preparation; configuration can be completed on the business cluster, saving costs.

Accessing External Storage

Option 1: Access the distributed storage resources of other business clusters within the platform to ensure storage and business are isolated for easier management and maintenance.

Option 2: Integrate external Ceph storage resources as distributed storage.

Requirement (choose one)	Advantages
Option 1: Distributed storage already deployed in	Can fully utilize storage resources across clusters and avoid interference from business changes. Ensures data security

Requirement (choose one)	Advantages
other business clusters.	and stability while reducing operational complexity. **Note**: If the storage to be accessed is distributed storage from different platforms, such as a primary/backup platform in a disaster recovery environment, please use the method of integrating external Ceph.
Option 2: External Ceph storage outside the platform, version ≥ 14.2.3.	Compared to directly creating a storage class, this method is more convenient for using the platform's interface for volume snapshots, scaling, and other functions.

Note: If you need to maintain the storage pool, storage device, and other configurations of external storage, operations must be performed in the management interface of the storage cluster.

Install

Create Standard Type Cluster

A standard-type cluster is the most typical deployment method for Ceph storage. It distributes data replicas across hard drives on different hosts, ensuring that if a single host fails, the data copies on other hosts can still maintain service availability.

TOC

Prerequisites

Prepare Package

- **Download** the **Alauda Container Platform Storage Essentials** installation package corresponding to your platform architecture.
- **Upload** the **Alauda Container Platform Storage Essentials** installation package using the Upload Packages mechanism.
- **Download** the **Alauda Build of Rook-Ceph** installation package corresponding to your platform architecture.
- **Upload** the **Alauda Build of Rook-Ceph** installation package using the Upload Packages mechanism.

Prepare Infrastructure

- At least 3 nodes are required in the storage cluster.
- Each node must have at least 1 blank hard disk or 1 unformatted hard disk partition available.
- The available hard disk capacity is recommended to be greater than 50 G.
- If you are using an attached Kubernetes cluster with Containerd as the runtime component, please ensure that the `LimitNOFILE` parameter value in the `/etc/systemd/system/containerd.service` file is configured to `1048576` on all nodes of the cluster, to ensure successful deployment of distributed storage. For configuration instructions, please refer to Modifying Containerd Configuration Information.
  Note: When upgrading from versions earlier than v3.10.2 to the current version, if you need to deploy Ceph distributed storage on your custom Kubernetes cluster with Containerd as the runtime component, you must also set the `LimitNOFILE` parameter value in the `/etc/systemd/system/containerd.service` file to `1048576` on all nodes of the cluster.

Precautions

Creating Storage Service and **Accessing Storage Service** only support selecting one method.

Procedure

1 Deploy Alauda Container Platform Storage Essentials

1. Login, go to the **Administrator** page.
2. Click **Marketplace** > **OperatorHub** to enter the **OperatorHub** page.
3. Find the **Alauda Container Platform Storage Essentials**, click **Install**, and navigate to the **Install Alauda Container Platform Storage Essentials** page.
   Configuration Parameters:

Parameter	Recommended Configuration
Channel	The default channel is `stable`.
Installation Mode	`Cluster`: All namespaces in the cluster share a single Operator instance for creation and management, resulting in lower resource usage.
Installation Place	Select `Recommended`, Namespace only support **acp-storage**.
Upgrade Strategy	`Manual`: When there is a new version in the Operator Hub, manual confirmation is required to upgrade the Operator to the latest version.

2 (Optional) Deploy Alauda Build of LocalStorage

When utilizing the `Selection Device` method to add storage devices to your Ceph cluster, it is necessary to deploy the `Alauda Build of LocalStorage Operator`. This Operator is responsible for automatically discovering all hard disk devices across every node within the Kubernetes cluster and collecting comprehensive device information, thereby streamlining the storage integration process.

1. Login, go to the **Administrator** page.
2. Click **Marketplace** > **OperatorHub** to enter the **OperatorHub** page.

3. Find the **Alauda Build of LocalStorage**, click **Install**, and navigate to the **Install Alauda Build of LocalStorage** page.
   Configuration Parameters:

Parameter	Recommended Configuration
Channel	The default channel is `stable`.
Installation Mode	`Cluster`: All namespaces in the cluster share a single Operator instance for creation and management, resulting in lower resource usage.
Installation Place	Select `Recommended`, Namespace only support **acp-storage**.
Upgrade Strategy	`Manual`: When there is a new version in the Operator Hub, manual confirmation is required to upgrade the Operator to the latest version.

3 Deploy Operator

1. Navigate to **Administrator**.
2. In the left sidebar, click **Storage Management** > **Distributed Storage**.
3. Click **Configure Now**.
4. In the **Deploy Operator** wizard page, click the **Deploy Operator** button at the bottom right.

- When the page automatically advances to the next step, it indicates that the Operator has been deployed successfully.
- If the deployment fails, please refer to the prompt on the interface **Clean Up Deployed Information and Retry**, and redeploy the Operator; if you wish to return to the distributed storage selection page, click **Application Store**, first uninstall the resources in the already deployed **rook-operator**, and then uninstall **rook-operator**.

4 Create Cluster

1. In the **Create Cluster** wizard page, configure the relevant parameters and click the **Create Cluster** button at the bottom right.

Parameter	Explanation
Cluster Type	Select **Standard**.
Device Class Type	Device classes are groupings of hard disks; you can customize device classes according to your storage needs, allocating different storage content to disks of varying performance. • **Default Device Class**: The platform will automatically categorize the types of hard disks in the cluster nodes. For instance, creating device classes named `hdd`, `ssd`, `nvme`. • **Custom Device Class**: Customize the name of the device class for specific combinations of disks in the node; adding multiple device classes is supported. The same hard disk can only belong to one device class.
Device Class - Name	The name of the device class. When selecting **Custom Device Class**, the device class cannot use the following names: `hdd`, `ssd`, `nvme`.

Device Class - Storage Devices	To add storage devices to a device class, you can choose between the `Selection Device` and `Input Device` methods:
	• **Selection Device**:
	Select from available storage devices. A device is considered available if it meets the following criteria:
	  ◦ Device type is either disk or mpath
	  ◦ No file system is detected (fsType is blank)
	  ◦ Capacity exceeds 10 GiB
	Devices such as rbd, nbd, and dm-* will not be displayed in the list of available selectable devices.

Parameter	Explanation
	Note: Requires prior deployment of the Alauda Build of LocalStorage Operator. • **Input Device**: Manually input the names of the blank devices under the node, such as `sda`. **Note**: For optimal performance and management, it is strongly advised to utilize raw disks as storage devices instead of employing individual partitions on a disk.
Snapshot	When enabled, it supports creating PVC snapshots and using snapshots to configure new PVCs for quick backup and recovery of business data. If you did not enable snapshots when creating storage, you can still enable them as needed from the **Operations** section on the storage cluster details page. **Note**: Please ensure that you have deployed volume snapshot plugins for the current cluster before using.
Monitoring Alarm	When enabled, it will provide out-of-the-box monitoring metric collection and alerting capabilities, see Monitoring and Alarming. **Note**: If not enabled at this time, you will need to find alternative solutions for storage monitoring and alarms. For example, manually configuring monitoring dashboards and alert strategies in the operation and maintenance center.

2. Click **Advanced Configuration** for advanced component configuration.

Parameter	Explanation
Network Configuration	• **Host Network**: The storage cluster will use the host network, and you should fill in the relevant network optimization parameters in the optimization parameters

Parameter	Explanation
	column, such as configuring the `public` and `cluster` subnets. If left blank, the default host subnet will be used. **Note**: Using the host network may expose security risks due to unencrypted (plaintext) transmission of data through host ports. Please contact the platform support team to obtain the encrypted transmission solution. • **Container Network**: The storage cluster will use container networking; you can create subnets in network management and assign them to the `rook-ceph` namespace. If left blank, the default subnet will be used. **Note**: IPv6 not supported. When using the container network, storage is only accessible within the cluster. Failures or restarts of the Ceph CSI Pod may result in service interruptions.
Optimization Parameters	Supports filling parameters in Ceph configuration file format; the system will overwrite the default parameters based on the provided content. **Note**: After first filling in or modifying initialization parameters, please click on the initialization parameters; successful initialization is required before a cluster can be created.
Component Fixed-point Deployment	You can deploy components to specified nodes; at least three nodes are required to ensure minimum availability. The components eligible for fixed-point deployment configuration include MON, MGR, MDS, RGW.

- When the page automatically advances to the next step, it indicates that the Ceph cluster has been deployed successfully.
- If the creation fails, you may click to clean up **Created Information or Retry** to automatically clean up the resources and recreate the cluster, or manually clean

up resources according to the documentation Distributed Storage Service Resource Cleanup.

5 Create Storage Pool

1. In the **Create Storage Pool** wizard page, configure the relevant parameters and click the **Create Storage Pool** button at the bottom right.

Parameter	Explanation
Storage Type	• File Storage: Provides secure, reliable, and scalable shared file storage services. Suitable for file sharing, data backup, etc. • Block Storage: Provides high IOPS and low-latency storage services. Suitable for databases, virtualization, etc. • Object Storage: Provides standard S3 interface storage services, suitable for big data, backup archiving, cloud storage, etc.
Replica Count	The larger the number of replicas, the higher the redundancy and data security; however, the utilization rate of storage will decrease. It is usually set to 3 to meet most needs.
Device Class	Uniformly classify types for the same type of device or disks of the same business logic, selecting from the device classes added in the previous step. • When selecting a device class, data will be stored in the chosen device class. • If no device class is selected, data will be randomly stored across all devices in the storage pool.

If it is object storage, you also need to configure the following parameters:

Parameter	Explanation
Region	Specify the region where the storage pool is located.

Parameter	Explanation
Gateway Type	Default is S3 and cannot be modified.
Internal Port	Specify the port for internal access in the cluster.
External Access	Enabling/disabling external access will create/destroy Nodeport type Service.
Instance Count	The number of resource instances for object storage.

- When the page automatically advances to the next step, it indicates that the storage pool has been deployed successfully.
- If the deployment fails, please refer to the interface prompts to check the core components, and then click **Clean Up Created Information and Retry** to recreate the storage pool.

2. Click **Create Storage Pool**. In the **Details** tab, you can view information about the created storage pool.

Related Operations

Create Stretch Type Cluster

For details, please refer to Create Stretch Type Cluster.

Cleanup Distributed Storage

For details, please refer to Cleanup Distributed Storage.

Create Stretch Type Cluster

A stretch cluster can extend across two geographically distinct locations, providing disaster recovery capabilities for storage infrastructure. In the event of a disaster, when one availability zone in the two zones is completely unavailable, Ceph can still maintain availability.

> ⚠️ **IMPORTANT: Alpha Feature Disclaimer**
> This document describes a feature currently in the **Alpha stage**, provided strictly for early technical validation and evaluation. As the feature provider, we offer no warranties regarding its stability, reliability, or data integrity. By configuring and using this feature, you acknowledge and accept the following technical limitations:
>
> - **Not for Production Use:** This feature lacks comprehensive system-level validation. Deploying it in production environments carries a high risk of process failure or data corruption.
> - **No SLA Guarantee:** This feature falls outside standard Service Level Agreements (SLAs). We do not guarantee technical support response times or provide emergency hotfixes.
> - **Breaking Changes & Deprecation:** API endpoints, configuration schemas, and core processing logic are subject to backwards-incompatible changes in future releases. The feature may also be deprecated entirely without prior notice.
> - **No Smooth Upgrade Path:** We do not provide upgrade scripts or data migration tools between versions. Upgrading to subsequent versions typically requires completely purging existing resources and performing a fresh deployment. Any resulting loss of state or data is the sole responsibility of the user.

TOC

Terminology

Term	Explanation
Quorum Availability Zone	Usually located in a separate zone that does not bear primary workloads, focusing on maintaining cluster consistency, and is primarily used for arbitration decisions when a failure occurs in the main data center or a network partition occurs.
Data Availability Zone	The primary area in the Ceph cluster where data is actually stored and processed, bearing operational loads and data storage tasks, forming a complete high-availability storage system together with the quorum zone.

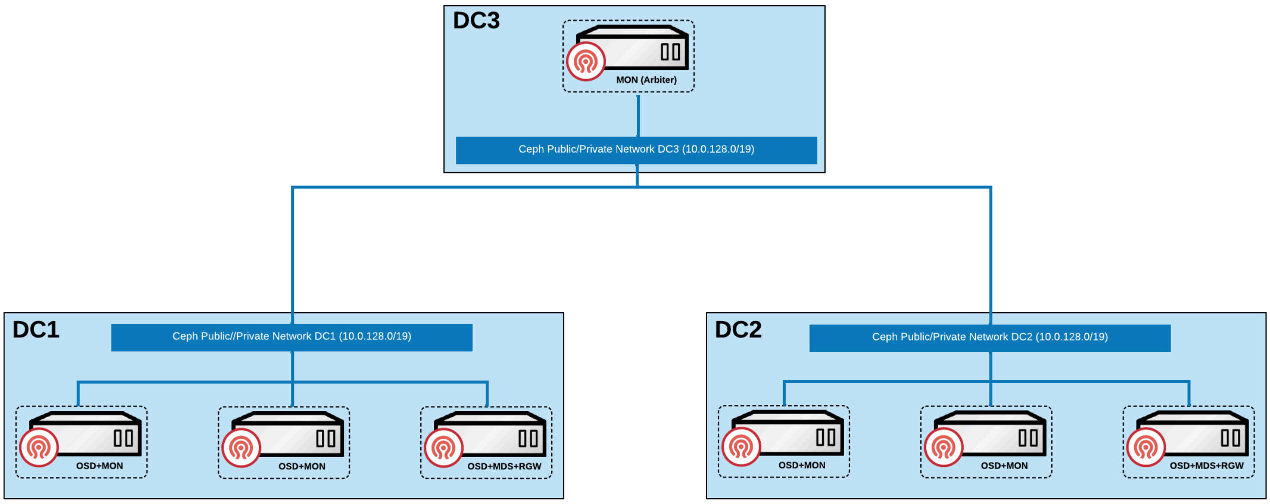
Typical Deployment Scheme

The following content provides a typical deployment scheme for stretch clusters, along with component descriptions and principles of disaster recovery.

Component Description

Nodes need to be distributed across three availability zones, including two data availability zones and one quorum availability zone.

- Both data availability zones need to fully deploy all core Ceph components (MON, OSD, MGR, MDS, RGW), and each data availability zone must configure two MON instances for high availability. When both MON instances in the same data availability zone are unavailable, the system will determine that the availability zone is in a failure state.
- The quorum availability zone only requires the deployment of one MON instance, serving as the arbitration decision node.



Disaster Recovery Explanation

- When a data availability zone completely fails, the Ceph cluster will automatically enter a degraded state and trigger an alarm notification. The system will adjust the minimum number of replicas in the storage pool (min_size) from the default of 2 to 1. Since the other data availability zone still maintains dual replicas, the cluster remains available. When the failed data availability zone recovers, the system will automatically execute data synchronization and return to a healthy state; if the failure cannot be repaired, it is recommended to replace it with a new data availability zone.
- When the network connection between the two data availability zones is interrupted, but they can still connect normally to the quorum availability zone, the quorum availability zone will arbitrate between the two data availability zones based on preset policies, selecting the one in a better state to continue providing services as the primary data zone.

Constraints and Limitations

- **Storage Pool Limitations**: Erasure-coded storage pools are not supported, and only replica mechanisms can be used for data protection.
- **Device Classification Limitations**: Device class functionality is not supported, and storage cannot be stratified based on device characteristics.
- **Regional Deployment Limitations**: Only two data availability zones are supported; no more than two data availability zones can exist.
- **Data Balancing Requirements**: The OSD weights of the two data availability zones must strictly remain consistent to ensure balanced data distribution.
- **Storage Medium Requirements**: Only all-flash (All-Flash) OSD configurations are permitted, minimizing the time required for recovery after a connection is restored, and reducing the potential for data loss as much as possible.
- **Network Latency Requirements**: The RTT (round-trip time) between the two data availability zones must not exceed 10ms, and the quorum availability zone must meet the ETCD specification latency requirements to ensure the reliability of the arbitration mechanism.

Prerequisites

Please classify all or part of the nodes in the cluster into three availability zones in advance, as follows:

- Ensure that at least 5 nodes are distributed among one quorum availability zone and two data availability zones. Among them, the quorum availability zone must have at least one node, which can be a virtual machine or cloud host.
- Ensure that at least one availability zone in the three availability zones contains a Master node (control node).
- Ensure that at least 4 computing nodes are evenly distributed across the 2 data availability zones, with at least 2 computing nodes configured in each data availability zone.
- Try to ensure that the number of nodes and disk configurations in the two data availability zones are consistent.

Procedure

1 Tagging Nodes

1. Access **Administrator**.
2. In the left navigation bar, click **Cluster Management** > **Cluster**.
3. Click on the corresponding cluster name to enter the cluster overview page.
4. Switch to the **Nodes** tab.
5. Based on the planning in the Prerequisites, add the `topology.kubernetes.io/zone=<zone>` label to these nodes to classify them into the specified availability zone. Here, replace <zone> with the name of the availability zone.

2 Create Storage Service

This document only describes the parameters that differ from standard type clusters; for other parameters, please refer to Create Standard Type Cluster.

Create Cluster

Parameter	Description
Cluster Type	Select **Stretch**.
Quorum Availability Zone	Choose the name of the quorum availability zone.
Data Availability Zone	Select the names of the availability zones and choose the nodes.

Create Storage Pool

Parameter	Description
Number of Replicas	Default is 4.

Parameter	Description
Number of Instances	When the storage type is **Object Storage**, to ensure availability, the minimum number of instances is 2 and the maximum is 5.

Related Operations

Create Standard Type Cluster

For details, please refer to Create Standard Type Cluster.

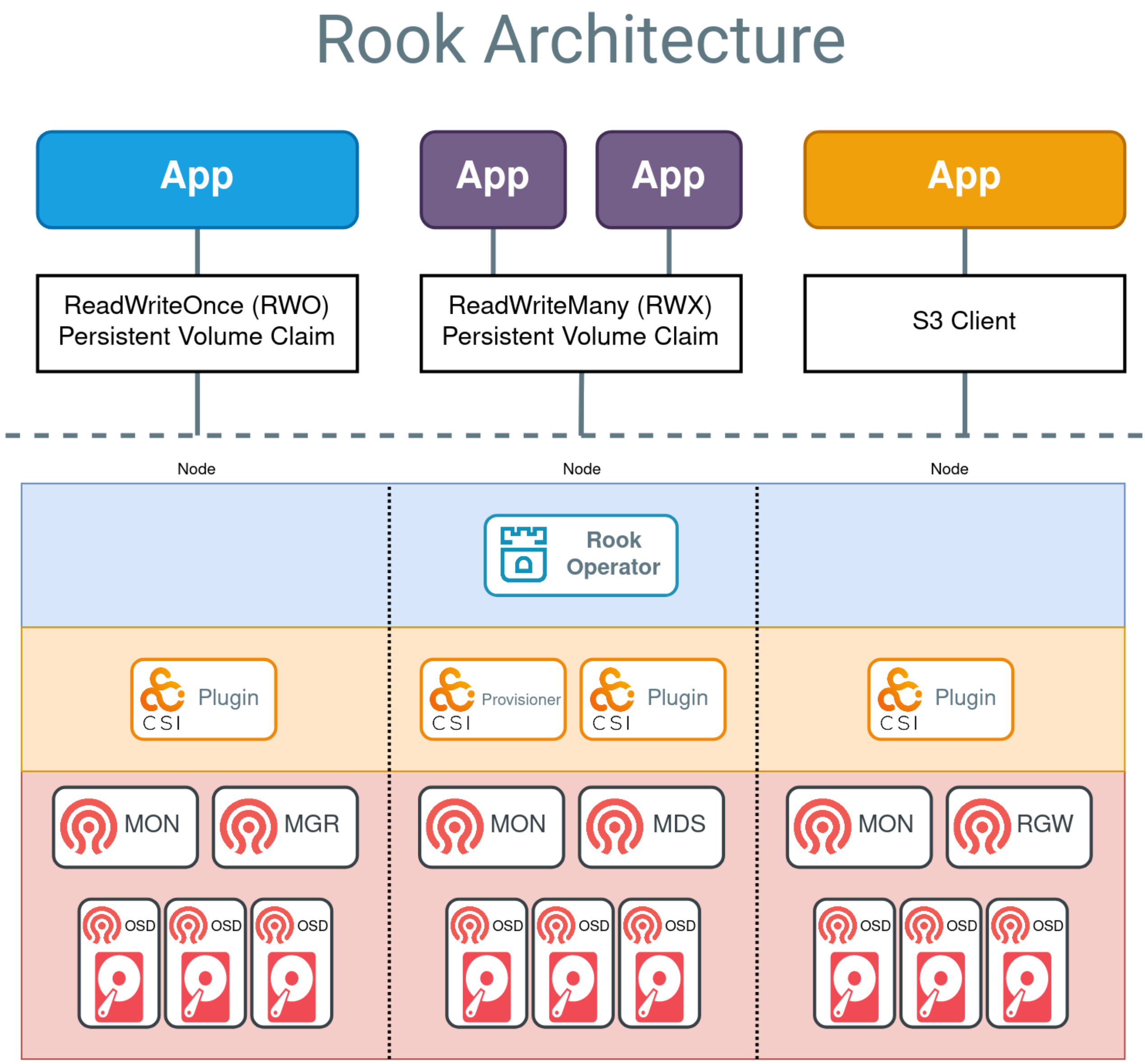
Cleanup Distributed Storage

For details, please refer to Cleanup Distributed Storage.

Architecture

TOC

Technical architecture



Example applications are shown above for the three supported storage types:

- Block Storage is represented with a blue app, which has a ReadWriteOnce (RWO) volume mounted. The application can read and write to the RWO volume, while Ceph manages the IO.
- Shared Filesystem is represented by two purple apps that are sharing a ReadWriteMany (RWX) volume. Both applications can actively read or write simultaneously to the volume. Ceph will ensure the data is safely protected for multiple writers with the MDS daemon.
- Object storage is represented by an orange app that can read and write to a bucket with a standard S3 client.

Below the dotted line in the above diagram, the components fall into three categories:

- Rook operator (blue layer): The operator automates configuration of Ceph
- CSI plugins and provisioners (orange layer): The Ceph-CSI driver provides the provisioning and mounting of volumes
- Ceph daemons (red layer): The Ceph daemons run the core storage architecture. See the Glossary to learn more about each daemon.

Block Storage

In the diagram above, the flow to create an application with an RWO volume is:

- The (blue) app creates a PVC to request storage.
- The PVC defines the Ceph RBD storage class (sc) for provisioning the storage.
- K8s calls the Ceph-CSI RBD provisioner to create the Ceph RBD image.
- The kubelet calls the CSI RBD volume plugin to mount the volume in the app.
- The volume is now available for reads and writes.
- A ReadWriteOnce volume can be mounted on one node at a time.

Shared Filesystem

In the diagram above, the flow to create a applications with a RWX volume is:

- The (purple) app creates a PVC to request storage.
- The PVC defines the CephFS storage class (sc) for provisioning the storage.
- K8s calls the Ceph-CSI CephFS provisioner to create the CephFS subvolume.
- The kubelet calls the CSI CephFS volume plugin to mount the volume in the app.
- The volume is now available for reads and writes.
- A ReadWriteMany volume can be mounted on multiple nodes for your application to use.

Object Storage S3

In the diagram above, the flow to create an application with access to an S3 bucket is:

- The (orange) app creates an BucketClaim to request a bucket.
- The Ceph COSI Driver creates a Ceph RGW bucket.
- The Ceph COSI Driver creates a secret with the credentials for accessing the bucket.
- The app retrieves the credentials from the secret.
- The app can now read and write to the bucket with an S3 client.

Core Concepts

Core Concepts

TOC

Rook Operator

The Rook operator is a simple container that has all that is needed to bootstrap and monitor the storage cluster. The operator will start and monitor Ceph monitor pods, the Ceph OSD daemons to provide RADOS storage, as well as start and manage other Ceph daemons. The operator manages CRDs for pools, object stores (S3/Swift), and filesystems by initializing the pods and other resources necessary to run the services.

The operator will monitor the storage daemons to ensure the cluster is healthy. Ceph mons will be started or failed over when necessary, and other adjustments are made as the cluster grows or shrinks. The operator will also watch for desired state changes specified in the Ceph custom resources (CRs) and apply the changes.

Rook automatically configures the Ceph-CSI driver to mount the storage to your pods. The rook/ceph image includes all necessary tools to manage the cluster.

Ceph CSI

Ceph CSI plugins implement an interface between a CSI-enabled Container Orchestrator (CO) and Ceph clusters. They enable dynamically provisioning Ceph volumes and attaching them to workloads.

Ceph module functions

Module	Function
MON	The monitor (MON) is the most important component in a Ceph cluster. It manages the Ceph cluster and maintains the status of the entire cluster. The MON ensures that related components of a cluster can be synchronized at the same time. It functions as the leader of the cluster and is responsible for collecting, updating, and publishing cluster information.
MGR	The manager (MGR) is a monitoring system that provides collection, storage, analysis (including alarming), and visualization functions. It makes certain cluster parameters available for external systems.
OSD	Object storage daemons (OSDs) store the actual user data. Every OSD is usually bound to one physical drive. The OSDs handle the read/write requests from clients.
MDS	The Ceph Metadata Server (MDS) tracks the file hierarchy and stores metadata used only for CephFS. The RBD and RGW do not require metadata. The MDS does not directly provide data services for clients.
RGW	The RADOS gateway (RGW) is a Ceph object gateway that provides RESTful APIs compatible with S3 and Swift. The RGW also supports multi-tenant and OpenStack Identity service (Keystone).
RADOS	Reliable Autonomic Distributed Object Store (RADOS) is the heart of a Ceph storage cluster. Everything in Ceph is stored by RADOS in the form of objects irrespective of their data types. The RADOS layer ensures data consistency and reliability through data replication, fault detection and recovery, and data recovery across cluster nodes.
LIBRADOS	Librados is a method that simplifies access to RADOS. Currently, it supports programming languages PHP, Ruby, Java, Python, C, and C++. It provides RADOS, a local interface of the Ceph storage cluster, and is the base component of other services such as the RADOS block device (RBD) and RADOS gateway

Module	Function
	(RGW). In addition, it provides the Portable Operating System Interface (POSIX) for the Ceph file system (CephFS). The Librados API can be used to directly access RADOS, enabling developers to create their own interfaces for accessing the Ceph cluster storage.
RBD	The RADOS block device (RBD) is the Ceph block device that provides block storage for external systems. It can be mapped, formatted, and mounted like a drive to a server.
CephFS	The CephFS provides a POSlX-compatible distributed file system of any size. It depends on the Ceph MDS to track the file hierarchy, namely the metadata.

Guides

Accessing Storage Services

Accessing storage services supports two methods of integration: first, integrating distributed storage resources from other business clusters within the platform to ensure storage and business isolation for easier management and maintenance; second, connecting external Ceph storage resources for distributed storage use.

TOC

Prerequisites

Prepare Package

- **Download** the **Alauda Container Platform Storage Essentials** installation package corresponding to your platform architecture.
- **Upload** the **Alauda Container Platform Storage Essentials** installation package using the Upload Packages mechanism.
- **Download** the **Alauda Build of Rook-Ceph** installation package corresponding to your platform architecture.
- **Upload** the **Alauda Build of Rook-Ceph** installation package using the Upload Packages mechanism.

Prepare Storage

Choose one of the following:

- Distributed storage has been deployed in other business clusters, and a storage pool has been created. Please record the name of the storage pool for later integration use.
- External Ceph storage outside the platform (version ≥ 14.2.3) has been created with a storage pool. Please record the name of the storage pool for later integration use.

Open Ports

Destination IP	Destination Port	Source IP	Source Port
IP of Ceph node	3300, 6789, 6800-7300, 7480	IP of all nodes in business cluster	any

Obtain Authentication Information (External Ceph)

If the prepared storage is external Ceph storage, authentication information must be obtained using the following commands.

Parameter	Method of Acquisition
FSID	`ceph fsid`

Parameter	Method of Acquisition
MON Component Information	`ceph mon dump` Must be in {name= IP} format, e.g. *a=192.168.100.100:6789* .
Admin Key	`ceph auth get-key client.admin`
Storage Pool	• File storage: Use `ceph fs ls` command to get the `name` value. • Block storage: `ceph osd dump` | `grep "application rbd"` | `awk '{print $3}'`
Data Storage Pool	(only needed for file storage) Use `ceph fs ls` command to get the `data pools` value.

Procedure

Note: The following steps take **accessing external Ceph storage** as an example, the operations for accessing distributed storage are similar.

1 Deploy Alauda Container Platform Storage Essentials

1. Login, go to the **Administrator** page.
2. Click **Marketplace** > **OperatorHub** to enter the **OperatorHub** page.
3. Find the **Alauda Container Platform Storage Essentials**, click **Install**, and navigate to the **Install Alauda Container Platform Storage Essentials** page.
   Configuration Parameters:

Parameter	Recommended Configuration
Channel	The default channel is `stable`.
Installation Mode	`Cluster`: All namespaces in the cluster share a single Operator instance for creation and management, resulting in lower resource

Parameter	Recommended Configuration
	usage.
Installation Place	Select `Recommended`, Namespace only support **acp-storage**.
Upgrade Strategy	`Manual`: When there is a new version in the Operator Hub, manual confirmation is required to upgrade the Operator to the latest version.

2 Access Storage

1. In the left navigation bar, click **Storage Management** > **Distributed Storage**.
2. Click **Access Storage**.
3. On the **Access Configuration** wizard page, select **External Ceph**.

Parameter	Description
Snapshot	When enabled, supports creating PVC snapshots and using snapshots to configure new PVCs for quick backup and restoration of business data. If snapshots were not enabled during storage access, you can still enable them later in the **Operations** section of the storage cluster details page as needed. **Note**: Please ensure that you have deployed the volume snapshot plugin for the current cluster before use.

Parameter	Description
Network Configuration	• **Host Network**: Computing components in this cluster will access the **storage cluster** using the **host network**. • **Container Network**: Computing components in this cluster will access the **storage cluster** using the **container network**. You can create a subnet in network management and assign the subnet to the `rook-ceph` namespace. If left empty, the default subnet will be used.
Other Parameters	Please fill in the authentication parameters for the external Ceph obtained in the prerequisites.

4. On the **Create Storage Class** wizard page, complete the configuration and click **Access**.

Parameter	Description
Type	Based on the type of storage pool created above, the default corresponding storage class will be: • File Storage: CephFS File Storage • Block Storage: CephRBD Block Storage
Reclaim Policy	Reclaim policy for persistent volumes. • Delete: When the persistent volume claim is deleted, the bound persistent volume will also be deleted. • Retain: Even if the persistent volume claim is deleted, the bound persistent volume will still be retained.
Project Allocation	Projects that can use this type of storage. If there are currently no projects requiring this type of storage, you may choose not to allocate projects for now and update them later.

> 5. Wait approximately 1-5 minutes for the successful integration.

Follow-up Actions

- Create Storage Classes: CephFS File Storage, CephRBD Block Storage
- Developers using the above storage classes to create persistent volume claims can extend usage with volume snapshots and scaling features.

Note: If you need to maintain storage pools, storage device configurations, etc., for external storage, operations must be performed in the management platform of the storage cluster.

Managing Storage Pools

A storage pool refers to a logical partition used for storing data. A single storage cluster supports the simultaneous use of different types of storage pools, such as file storage and block storage, to accommodate various business requirements.

TOC

Creating a Storage Pool

In addition to the storage pools created during the configuration of distributed storage, you can also create additional types of storage pools.

Tip: Within the same storage cluster, only one file storage and one object storage pool are allowed, while up to eight block storage pools can be created.

Procedure

1. Access **Administrator**.
2. In the left navigation bar, click **Storage Management** > **Distributed Storage**.
3. In the **Cluster Information** tab, scroll down to the **Storage Pool** area and click **Create Storage Pool**.
4. Configure the relevant parameters according to the following instructions.

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>Storage Type</td><td>Select the currently undeployed storage type.
- File Storage: Provides secure, reliable, and scalable shared file storage services. Suitable for file sharing, data backup, etc.
- Block Storage: Provides high IOPS and low latency storage services. Suitable for databases, virtualization, etc.
- Object Storage: Provides standard S3 interface storage services, suitable for big data, backup archiving, cloud storage services, etc.</td></tr>
<tr><td>Replica Count</td><td>When the cluster type is Standard: A higher replica count increases redundancy and data security, but it also reduces storage utilization. Usually, a setting of 3 suffices for most needs.
When the cluster type is Extended: The default replica count is 4 and cannot be modified.</td></tr>
<tr><td>Device Class</td><td>When the cluster type is Standard: Choose an already added device class within the created storage pool.
When selecting a device class, data will be stored in the chosen device class.
If no device class is selected, data will be randomly stored in all devices within the storage pool.
When the cluster type is Standard: Adding a device class is not supported.</td></tr>
</table>

If it is an object storage type, you can configure the following parameters as well:

Parameter	Description
Region	Specify the region where the storage pool is located.
Gateway Type	Defaults to S3 and cannot be modified.
Internal Port	Specify the port for internal access to the cluster.
External Access	Enabling/disabling external access will create/destroy a NodePort type Service.
Instance Count	Number of resource instances for object storage.

5. Click **Create**.

Deleting a Storage Pool

If a certain type of storage is no longer required, the storage pool can be deleted after dissociating it from the storage class.

Procedure

1. Access **Administrator**.
2. In the left navigation bar, click **Storage Management** > **Distributed Storage**.
3. In the **Cluster Information** tab, scroll down to the **Storage Pool** area, click on the ⋮ next to the storage pool you wish to delete > **Delete**.
4. Read the prompt information and enter the name of the storage pool.
5. Click **Delete**.

Viewing Object Storage Pool Addresses

After creating an object storage pool, you can view the internal and external access addresses of the storage pool.

Procedure

> 1. Access **Administrator**.
> 2. In the left navigation bar, click **Storage Management** > **Distributed Storage**.
> 3. In the **Cluster Information** tab, scroll down to the **Storage Pool** area, click on the ⋮ next to the object storage pool and select **View Address**.

Node-specific Component Deployment

After creating distributed storage, you can still view and modify the deployment location of components, facilitating storage expansion and maintenance.

TOC

Update Component Deployment Configuration

Precautions

- Updating the configuration will trigger the system to automatically rebuild component instances, which may affect service access to the storage system. It is recommended to perform the update during off-peak hours.
- When the cluster type is **Extend**, the fixed deployment feature for components is not supported.

Procedure

1. Go to **Administrator**.
2. In the left navigation bar, click on **Storage Management** > **Distributed Storage**.
3. Under the **Storage Components** tab, click on **Component Deployment Configuration**.
4. Enable/disable the **Fixed Deployment** switch according to business needs, and deploy components to specified nodes. The number of nodes must be no less than three to ensure minimum availability. The components applicable for fixed deployment configuration include MON, MGR, MDS, RGW.
5. Click **Update**, and the components will begin to be scheduled to the designated nodes.

Restart Storage Components

When you delete the deployed storage components, the system will automatically re-schedule and redeploy components to the nodes according to the current component deployment strategy.

Procedure

1. Go to **Administrator**.
2. In the left navigation bar, click on **Storage Management** > **Distributed Storage**.
3. Under the **Storage Components** tab, click ⋮ next to the component name > **Delete**.

Adding Devices/Device Classes

TOC

Adding Device Classes

Unify the classification of devices of the same type or hard disks with the same business logic in cluster nodes, customize device classes according to storage needs, and allocate different storage contents to different types of storage disks.

Notes

Adding device classes is not supported when the cluster type is **Extend**.

Procedure

1. Enter **Administrator**.
2. In the left navigation bar, click **Storage Management** > **Distributed Storage**.

3. Click the **Device Classes** tab.
4. Click **Add Device Class** and configure the relevant parameters according to the following instructions.

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>Name</td><td>The name of the device class. The following names cannot be used for the device class: <code>hdd</code>, <code>ssd</code>, <code>nvme</code>.</td></tr>
<tr><td>Storage Devices</td><td>To add storage devices to a device class, you can choose between the <code>Selection Device</code> and <code>Input Device</code> methods:
Selection Device:
Select from available storage devices. A device is considered available if it meets the following criteria:
Device type is either disk or mpath
No file system is detected (fsType is blank)
Capacity exceeds 10 GiB
Devices such as rbd, nbd, and dm-* will not be displayed in the list of available selectable devices.

Note: Requires prior deployment of the Alauda Build of LocalStorage Operator.
Input Device:
Manually input the names of the blank devices under the node, such as <code>sda</code>.

Note: For optimal performance and management, it is strongly advised to utilize raw disk as storage devices instead of employing individual partitions on a disk.</td></tr>
</table>

Adding Devices

Map available hard disks to storage devices for usage and management.

Note: Once hard disks are added as storage devices, updating or removing them through the interface is not supported.

Procedure

1. Enter **Administrator**.
2. In the left navigation bar, click **Storage Management** > **Distributed Storage**.
3. Click the **Device Classes** tab.
4. On the right side of the device class, click **Add Device**, and configure the relevant parameters according to the following instructions.

Parameter	Description

Specified Disks	To add storage devices to a device class, you can choose between the `Selection Device` and `Input Device` methods:
	
	• **Selection Device**:
	Select from available storage devices. A device is considered available if it meets the following criteria:
	  ◦ Device type is either disk or mpath
	  ◦ No file system is detected (fsType is blank)
	  ◦ Capacity exceeds 10 GiB
	
	Devices such as rbd, nbd, and dm-* will not be displayed in the list of available selectable devices.
	
	Note: Requires prior deployment of the Alauda Build of LocalStorage Operator.
	
	• **Input Device**:
	Manually input the names of the blank devices under the node, such as `sda`.

Parameter	Description
	Note: For optimal performance and management, it is strongly advised to utilize raw disks as storage devices instead of employing individual partitions on a disk.

5. Click **Add**.

Hard Disk Status

- **Normal**: The corresponding status of the storage device is IN+UP.
- **Abnormal**: The corresponding status of the storage device is IN+DOWN.
- **Offline**: The corresponding status of the storage device is OUT+UP.
- **Fault**: The corresponding status of the storage device is OUT+DOWN.

Monitoring and Alerts

Distributed storage provides out-of-the-box monitoring metrics collection and alert notification capabilities. Once the monitoring and alerting features are enabled, you can monitor and alert on aspects such as the storage cluster, storage performance, and storage components, with support for configuring notification strategies.

The intuitively presented monitoring data can be used to provide decision support for operation and maintenance inspections or performance tuning, and a comprehensive alert and notification mechanism will help ensure the stable operation of the storage system.

Tip: If the monitoring and alerting features were not enabled when creating the distributed storage, you will need to find alternative solutions for storage monitoring and alerting. For example, manually configure monitoring dashboards and alert strategies in the operation and maintenance center.

TOC

Monitoring

The platform automatically collects common monitoring metrics for distributed storage, such as read and write performance, CPU and memory usage. In the **Storage Management** > **Distributed Storage** section under the **Monitoring** tab, you can view real-time monitoring data for these metrics.

Storage Overview

Monitor the health status of the storage, physical capacity usage, and the number of active OSD/MON components. In the event of abnormal storage status, you can check the reason for the alert.

Performance Monitoring

Monitor read and write bandwidth and read and write IOPS from three dimensions: cluster, storage pool, and OSD. Additionally, you can monitor read and write latency specifically for OSD.

Component Monitoring

Monitor CPU usage and memory usage of components such as MON and OSD.

Alerts

The platform has a set of default alert strategies enabled. Once a resource becomes abnormal or monitoring data reaches the warning state, alerts will be automatically triggered. The preset strategies are sufficient for common operational needs such as component and cluster status alerts, device capacity alerts, and user data alerts.

Configure Notifications

To receive alerts in a timely manner, it is recommended that you set up notification strategies in the operation and maintenance center: send alert information via email, SMS, and other means to relevant personnel, reminding them to take necessary measures to resolve issues or prevent failures. Click **Alert Configuration** to switch to the operation and maintenance center to complete the operation, refer to Create Alert Strategies。

Handling Alerts

- If the storage cluster is monitored to be in a `Warning` state, it means an alert has been triggered, and the related anomaly may lead to a failure. Please promptly check the details in **Real-time Alerts** and identify and troubleshoot the fault based on the cause.
- If the storage cluster is monitored to be in a `Failure` state, it indicates that the storage cluster is unable to operate normally. Please locate the issue immediately and carry out troubleshooting.

The table below indicates the meanings of the alert levels used by the preset strategies, which can serve as a reference for you when establishing alert handling principles.

Alert Level	Meaning
Disaster	The resource corresponding to the alert rule has failed, causing platform service interruption, data loss, and significant impact.
Severe	The resource corresponding to the alert rule has known issues, which may lead to platform function failures and affect the normal operation of services.
Warning	The resource corresponding to the alert rule faces operational risks, which could affect the normal operation of services if not dealt with promptly.

Fault Review

The **Alert History** records all alerts that have been triggered and no longer require action. When conducting a fault review using the alert history, to effectively achieve the purpose of summarizing experiences, you may need to answer the following questions.

- What were the specific abnormal conditions at the time of the incident.
- Is there a pattern to a certain alert that appears repeatedly in the alert list, Can it be prevented before it occurs next time.

- Does the timeline show a surge in alerts during a certain period; was it caused by force majeure or an operational accident, Is there a need to adjust the operational plan.

How To

Configure a Dedicated Cluster for Distributed Storage

Cleanup Distributed Storage

Disaster Recovery

Update the optimization parameters

Create Ceph Object Store User

Setting Storage Pool Quotas

Setting Storage Pool Quotas

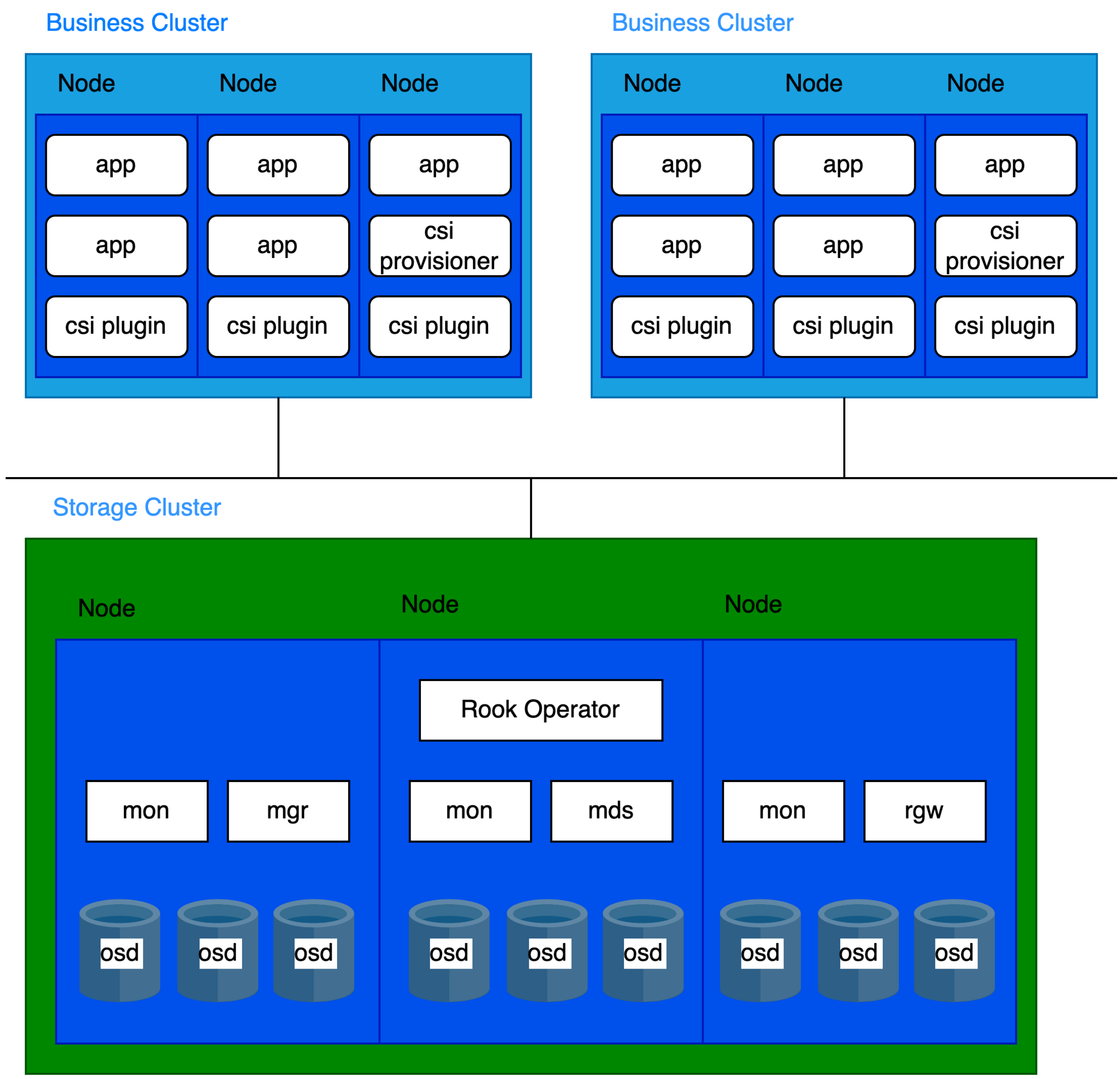
Configure a Dedicated Cluster for Distributed Storage

Dedicated cluster deployment refers to using an independent cluster to deploy the platform's distributed storage, where other business clusters within the platform access and utilize the storage services it provides through integration.
To ensure the performance and stability of the platform's distributed storage, only the platform's core components and distributed storage components are deployed in the dedicated storage cluster, avoiding the co-location of other business workloads. This separated deployment approach is the recommended best practice for the platform's distributed storage.

TOC

Architecture

Storage-Compute Separation Architecture



Infrastructure requirements

Platform requirements

Supported in version 3.18 and later.

Cluster requirements

It is recommended to use bare-metal clusters as dedicated storage clusters.

Resource requirements

Please refer to the Core Concepts for the components of distributed storage deployment.

Each component has distinct CPU and memory requirements. The recommended configurations are as follows:

Process	CPU	Memory
MON	2c	3Gi
MGR	3c	4Gi
MDS	3c	8Gi
RGW	2c	4Gi
OSD	4c	8Gi

A cluster typically runs:

- 3 MON
- 2 MGR
- multiple OSD
- 2 MDS (if using CephFS)
- 2 RGW (if using CephObjectStorage)

Based on the component distribution, the following per-node resource recommendations apply:

CPU	Memory
16c + (4c * OSD per node)	20Gi + (8Gi * OSD per node)

Storage device requirements

It is recommended to deploy 12 or fewer storage devices per node. This helps restrict the recovery time following a node failure.

Storage device type requirements

It is recommended to use enterprise SSDs with a capacity of 10TiB or smaller per device, and ensure all disks are identical in size and type.

Capacity planning

Before deployment, storage capacity must be planned according to specific business requirements. By default, the distributed storage system employs a 3-replica redundancy strategy. Therefore, the usable capacity is calculated by dividing the total raw storage capacity (from all storage devices) by 3.

Example for 30(N) nodes (replica count = 3), The usable capacity scenario is as follows:

Storage device size(D)	Storage device per node(M)	Total Capacity(D*M*N)	Usable Capacity(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

Capacity monitoring and expansion

1. **Proactive Capacity Planning**
   Always ensure usable storage capacity exceeds consumption. If storage is fully exhausted, recovery requires manual intervention and cannot be resolved by simply deleting or migrating data.
2. **Capacity Alerts**
   The cluster triggers alerts at two thresholds:

   - **80% utilization** ("near full"): Proactively **free up space** or scale out the cluster.
   - **95% utilization** ("full"): Storage is fully exhausted, and standard commands cannot free space. Contact platform support immediately.

Always address alerts promptly and monitor storage usage regularly to avoid outages.

3. **Scaling Recommendations**

   - **Avoid**: Adding storage devices to existing nodes.
   - **Recommended**: Scale out by adding new storage nodes.
   - **Requirement**: New nodes must use storage devices identical in size, type, and quantity to existing nodes.

Network requirements

Distributed storage must utilize **HostNetwork**.

Network Isolation

The network is categorized into two types:

- **Public Network**: Used for client-to-storage component interactions (e.g., I/O requests).
- **Cluster Network**: Dedicated to data replication between replicas and data rebalancing (e.g., recovery).

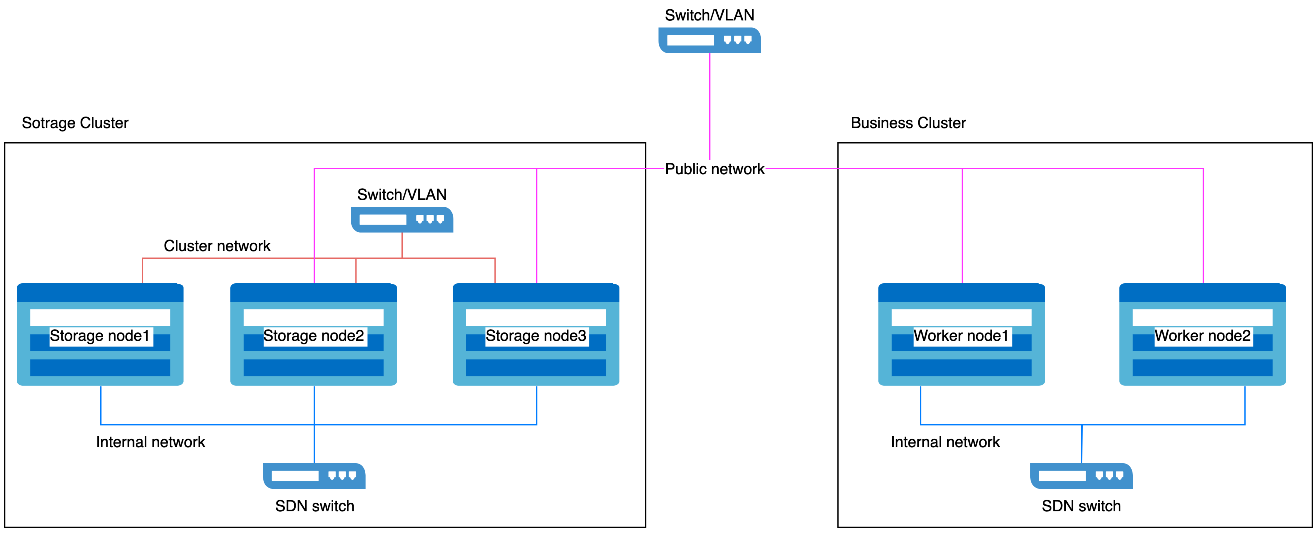
To ensure service quality and performance stability:

1. For Dedicated Storage Clusters:
   Reserve two network interfaces on each host:

   - Public Network: For client and component communication.
   - Cluster Network: For internal replication and rebalancing traffic.

2. For Business Clusters:
   Reserve one network interface on each host to access the storage Public Network.

Example Network Isolation Configuration



Network interface speed requirements

1. **Storage Nodes**

- **Public Network** and **Cluster Network** require 10GbE or higher network interfaces.

2. **Business Cluster Nodes**

- The network interface used to access the storage **Public Network** must be 10GbE or higher.

Procedure

1 Deploy Operator

1. Access **Administrator**.
2. In the left sidebar, click **Storage Management** > **Distributed Storage**.
3. Click **Create Now**.
4. In the **Deploy Operator** wizard page, click the **Deploy Operator** button at the bottom right.
    - When the page automatically advances to the next step, it indicates that the Operator has been deployed successfully.
    - If the deployment fails, please refer to the prompt on the interface **Clean Up Deployed Information and Retry**, and redeploy the Operator; if you wish to return to the distributed storage selection page, click **Application Store**, first uninstall the

resources in the already deployed **rook-operator**, and then uninstall **rook-operator**.

2 Create ceph cluster

Execute commands on the **control node** of the storage cluster.

▶ Click to view

Parameters:

- **public network cidr**: CIDR of the storage **Public Network** (e.g., `- 10.0.1.0/24`).
- **cluster network cidr**: CIDR of the storage **Cluster Network** (e.g., `- 10.0.2.0/24`).
- **storage devices**: Specify the storage devices to be utilized by the distributed storage.
  Example Formatting:

```
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
```

> **Tip**

Uses the disk's World Wide Name (WWN) for stable naming, which avoids reliance on volatile device paths like `sdb` that may change after reboots.

3 Create storage pools

Three storage pool types are available. Select and create the appropriate ones based on your business requirements.

Create file pool

Execute commands on the **control node** of the storage cluster.

▶ Click to view

Create block pool

Execute commands on the **control node** of the storage cluster.

▶ Click to view

Create object pool

Execute commands on the **control node** of the storage cluster.

▶ Click to view

Follow-up Actions

When other clusters need to utilize the distributed storage service, refer to the following guidelines.
Accessing Storage Services

Cleanup Distributed Storage

If you need to delete a rook-ceph cluster and redeploy a new one, you should follow this document to sequentially clean the distributed storage service related resources.

TOC

Precautions

Before cleaning up rook-ceph, ensure that all PVC and PV resources using Ceph storage have been deleted.

Procedure

1 Deleting VolumeSnapshotClasses

1. Delete the VolumeSnapshotClasses.

```
kubectl delete VolumeSnapshotClass csi-cephfs-snapshotclass csi-rb
d-snapshotclass
```

2. Verify that the VolumeSnapshotClasses have been cleaned up.

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

When there is no output from these commands, it indicates that the cleanup is complete.

2 Deleting StorageClasses

1. Go to **Administrator**.
2. In the left navigation bar, click **Storage Management** > **Storage Classes**.
3. Click ⋮ > **Delete**, and delete all StorageClasses that use Ceph storage solutions.

3 Deleting Storage Pools

This step should be performed after the previous step has been completed.

1. Go to **Administrator**.
2. In the left navigation bar, click **Storage Management** > **Distributed Storage**.
3. In the **Storage Pool Area**, click ⋮ > **Delete**, and delete all storage pools one by one. When the storage pool area shows **No Storage Pools**, it indicates successful deletion of the storage pools.
4. (Optional) If the cluster mode is **Extended**, you also need to execute the following command on the Master node of the cluster to delete the created built-in storage pools.

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

Response:

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

4 Deleting ceph-cluster

This step should be performed after the previous step has been completed.

1. Update the ceph-cluster and enable the cleanup policy.

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

2. Delete the ceph-cluster.

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

3. Delete the jobs that perform the cleanup.

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

4. Verify that the ceph-cluster cleanup is complete.

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

When this command has no output, it indicates that the cleanup is complete.

5 Deleting rook-operator

This step should be performed after the previous step has been completed.

1. Delete the rook-operator.

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

2. Verify that the rook-operator cleanup is complete.

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

When this command has no output, it indicates that the cleanup is complete.

3. Verify that all ConfigMaps have been cleaned up.

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

When this command has no output, it indicates that cleanup is complete. If there are output results, execute the following command to clean up, replacing `<configmap>` with the actual output.

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

4. Verify that all Secrets have been cleaned up.

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

When this command has no output, it indicates that cleanup is complete. If there are output results, execute the following command to clean up, replacing `<secret>` with the actual output.

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

5. Verify that the rook-ceph cleanup is complete.

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

When this command has no output, it indicates that cleanup is complete.

6

Execute Cleanup Script

Once the above steps are completed, it indicates that Kubernetes and Ceph related resources have been cleared. Next, you need to clean up any residuals of rook-ceph on the host.

Cleanup Script

The contents of the cleanup script clean-rook.sh are as follows:

▶ Click to view

Precautions

The cleanup script depends on the sgdisk command, so please make sure to have it installed before executing the cleanup script.

- Installation command for Ubuntu: `sudo apt install gdisk`
- Installation command for RedHat or CentOS: `sudo yum install gdisk`

Procedure

1. Execute the cleanup script clean-rook.sh on each machine in the business cluster where distributed storage is deployed.

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

   Example: `sh clean-rook.sh /dev/vdb`
   When executed, you will be prompted to confirm whether to really clear the device. If confirmed, enter yes to begin cleaning.

2. Use the `lsblk -f` command to check the partition information. When the `FSTYPE` column in the output of this command is empty, it indicates that the cleanup is complete.

Disaster Recovery

File Storage Disaster Recovery

CephFS Mirror is a feature of the Ceph file system designed to enable asynchronous data replication between different Ceph clusters, thereby providing cross-cluster disaster recovery. Its core functionality is to synchronize data in a primary-backup mode, ensuring that the backup cluster can rapidly take over services if the primary cluster experiences a failure.

WARNING

- CephFS Mirror performs incremental synchronization based on snapshots, with the default snapshot interval set to once per hour (configurable). The differential data between the primary and backup clusters typically consists of the amount of data written within one snapshot cycle.
- CephFS Mirror solely provides the backup of underlying storage data and is incapable of handling the backup of Kubernetes resources. Please utilize the platform's **Backup and Restore** feature to back up PVC and PV resources in conjunction.

TOC

Terminology

Term	Explanation
Primary Cluster	The cluster currently providing storage services.
Secondary Cluster	Cluster for backup.

Backup Configuration

Prerequisites

- Prepare two clusters suitable for deploying Alauda Build of Rook-Ceph, namely the Primary cluster and the Secondary cluster, ensuring that the networks between the clusters are interconnected.
- The platform versions used by both clusters (v3.12 and above) must be consistent.
- Create a distributed storage service in both the Primary and Secondary clusters
- Create file storage pools with the **same name** in both the Primary and Secondary clusters.

Procedure

1 Enable the Mirror for the file storage pool in the Secondary cluster

Execute the following commands on the Control node of the Secondary cluster:

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
```

Parameters:

- `<fs-name>`: Name of the file storage pool.

2

Obtain the Peer Token

This token is the key credential for establishing a mirroring connection between the two clusters.

Execute the following commands on the Control node of the Secondary cluster:

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

```
# Due to the involvement of sensitive information, the output has
been truncated.
eyJmc2lkIjogImMyYjAyNmMzLTA3ZGQtNDA3Z...
```

Parameters:

- `<fs-name>`: Name of the file storage pool.

3 Create Peer Secret in the Primary cluster

After obtaining the Peer Token from the Secondary cluster, it is necessary to create a Peer Secret in the Primary cluster.

Execute the following commands on the Control node of the Primary cluster:

Command

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

Output

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

Parameters:

- `<token>`: The token obtained in step 2.
- `<fs-name>`:Name of the file storage pool.

4 Enable the Mirror for the file storage pool in the Primary cluster

Execute the following commands on the Control node of the Primary cluster:

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
```

Parameters:

- `<fs-name>`:Name of the file storage pool.
- `<schedule-interval>`:Snapshot execution cycle. For details, please refer to the official documentation ↗.
- `<retention-policy>`: Snapshot retention policy. details, please refer to the official documentation ↗.

5 Deploy the Mirror Daemon in the Primary cluster

The Mirror Daemon continuously monitors data changes in the file storage pool (with Mirror enabled). It periodically creates snapshots and pushes the snapshot differences to the Secondary cluster over the network.

Execute the following commands on the Control node of the Primary cluster:

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
```

Failover

In the event of a Primary cluster failure, you can directly continue using CephFS in the Secondary cluster.

Prerequisites

The Kubernetes resources of the Primary cluster have been backed up and restored to the Secondary cluster, including PVCs, PVs, and workloads of the applications.

Block Storage Disaster Recovery

RBD Mirror is a feature of Ceph Block Storage (RBD) that enables asynchronous data replication between different Ceph clusters, providing cross-cluster Disaster Recovery (DR). Its core function is to synchronize data in a primary-backup mode, ensuring rapid service takeover by the backup cluster when the primary cluster fails.

> **WARNING**
>
> - RBD Mirror performs incremental synchronization based on snapshots, with a default snapshot interval of once per hour (configurable). The differential data between primary and backup clusters typically corresponds to writes within one snapshot cycle.
> - RBD Mirror only provides underlying storage data backup and does not handle Kubernetes resource backups. Please use the platform's **Backup and Restore** feature to back up PVC and PV resources.

TOC

Terminology

Term	Explanation
Primary Cluster	The cluster currently providing storage services.
Secondary Cluster	The standby cluster used for backup purposes.

Backup Configuration

Prerequisites

- Prepare two clusters capable of deploying Alauda Build of Rook-Ceph: a Primary cluster and a Secondary cluster.
- Both clusters must run the same platform version (v3.12 or later).
- Create distributed storage services in both Primary and Secondary clusters.
- Create block storage pools with **identical names** in both Primary and Secondary clusters.

- Please ensure that the following two images have been uploaded to the platform's private image repository:
  - `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`

network Prerequisites

Synchronization between the Primary and Secondary clusters is performed over the Public network of Ceph. Therefore, the inter-site Public network must satisfy the following requirements:

- Each cluster must have a dedicated Public network segment. The Public networks of the Primary and Secondary clusters must be fully routable and mutually reachable.
- The sustained bandwidth utilization of the Public network should not exceed 60% to ensure sufficient headroom for replication bursts and failover scenarios.
- The round-trip time (RTT) between the two sites should be less than 30 ms.
- The packet loss rate between the two Public networks should be lower than 0.05% to guarantee stable and predictable replication performance.

Procedures

Bootstrap Peers(`Primary <-> Secondary`)

1. **Enable Mirroring for Primary Cluster's Block Storage Pool**
   Execute the following command on both Primary and Secondary clusters' Control nodes:

   Command

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

Output

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

Parameters:

- `<block-pool-name>`: Block storage pool name.

2. **Obtained Peer Token**

   This token serves as the critical credential for establishing mirror connections between clusters.

   Execute the following command on both Primary and Secondary clusters' Control nodes:

   Command

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

   Output

```
# Output truncated due to sensitive information
eyJmc2lkIjoiMjc2N2I3ZmEtY2YwYi00N...
```

3. **Create Peer Token Secret in Peer Cluster**

   Execute the following command on both Primary and Secondary cluster's Control node:

   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
```

Parameters:

- `<token>`: Token obtained from Step 2.
  On the Primary cluster, configure this field using the token obtained from the Secondary cluster.
  On the Secondary cluster, configure this field using the token obtained from the Primary cluster.
- `<block-pool-name>`: Block storage pool name.

4. **Patch Peer Secret for Block Storage Pool**
   Execute the following command on both Primary and Secondary cluster's Control node:

   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
```

Parameters:

- `<block-pool-name>`: Block storage pool name.

5. **Deploy Mirror Daemon**

   This daemon is responsible for monitoring and managing RBD mirror synchronization processes, including data synchronization and error handling.

   Execute the following command on both Primary and Secondary cluster's Control node:

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. **Verify Mirror Status**

   Execute the following command on both Primary and Secondary cluster's Control node:

   Command

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

   Output

   ```
   # All "OK" statuses indicate normal operation
   {"daemon_health":"OK","health":"OK","image_health":"OK","states":{}}
   ```

   Parameters:

   - `<block-pool-name>`: Block storage pool name.

Setup Environment For Volume Replication

This feature enables efficient data replication and synchronization without interrupting primary application operations, enhancing system reliability and availability.

1. **Setup CsiAddons Controller**

   Execute the following commands on both Primary and Secondary clusters' Control nodes:

   ```
   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
   ```

   Parameters:

   - `<registry>`: Registry address of platform.

2. **Enable CsiAddons sidecar**

   Execute the following commands on both Primary and Secondary clusters' Control nodes:

```
kubectl patch cm rook-ceph-operator-config -n rook-ceph --type json --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"
  }
]'
```

Wait for all csi pods to restart successfully

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

3. **Create VolumeReplicationClass**

   Execute the following commands on both Primary and Secondary clusters' Control nodes:

   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>`: Scheduling interval, (e.g., schedulingInterval: "1h" indicates execution every 1 hour.)

Enable Mirror for PVC

Execute the following command on the Primary cluster's Control node:

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>`: The name of the VolumeReplication object, recommended to be the same as the PVC name.
2. `<namespace>`: The namespace to which the VolumeReplication belongs, which must be the same as the PVC namespace.
3. `<pvc-name>`: The name of the PVC for which Mirror needs to be enabled.

Note After enabling, the RBD image in the Secondary cluster becomes read-only.

Planned Migration

Use cases: Datacenter maintenance, technology refresh, disaster avoidance, etc.

Relocation

The Relocation operation is the process of switching production to a backup facility(normally your recovery site) or vice versa.

For relocation, access to the image on the primary site should be stopped. The image should now be made primary on the secondary cluster so that the access can be resumed there.

Prerequisites

- The Kubernetes resources of the Primary cluster have been backed up and restored to the Secondary cluster, including PVCs, PVs, application workloads, etc.

Procedures

Follow the below steps for planned migration of workload from the Primary cluster to the Secondary cluster:

1. Scale down all the application pods which are using the mirrored PVC on the Primary cluster.
2. Update VolumeReplications for all the PVCs which mirroring is enabled on the Primary cluster.
   Set `spec.replicationState` to `secondary`.
3. Create VolumeReplications for all the PVCs for which mirroring is enabled on the Secondary.

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>`: The name of the VolumeReplication object, recommended to be the same as the PVC name.

2 `<namespace>`: The namespace to which the VolumeReplication belongs, which must be the same as the PVC namespace.

3 `<pvc-name>`: The name of the PVC for which Mirror needs to be enabled.

4. Check VolumeReplication CR status to verify if the image is marked `primary` on the secondary site.

5. Once the Image is marked as `primary`, the PVC is now ready to be used. Now, we can scale up the applications to use the PVC.

Disaster Recovery

Use cases: Natural disasters, Power failures, System failures, and crashes, etc.

Failover (abrupt shutdown)

In case of Disaster recovery, create VolumeReplication CR at the Secondary Site.

Since the connection to the Primary Site is lost, the operator automatically sends a GRPC request down to the driver to forcefully mark the dataSource as `primary` on the Secondary Site.

Prerequisites

- The Kubernetes resources of the Primary cluster have been backed up and restored to the Secondary cluster, including PVCs, PVs, application workloads, etc.

Procedures

1. Create VolumeReplications for all the PVCs for which mirroring is enabled on the Secondary.

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>`: The name of the VolumeReplication object, recommended to be the same as the PVC name.

2 `<namespace>`: The namespace to which the VolumeReplication belongs, which must be the same as the PVC namespace.

3 `<pvc-name>`: The name of the PVC for which Mirror needs to be enabled.

2. Check VolumeReplication CR status to verify if the image is marked `primary` on the secondary site.
3. Once the Image is marked as `primary`, the PVC is now ready to be used. Now, we can scale up the applications to use the PVC.

Failback (post-disaster recovery)

Once the failed cluster is recovered on the primary site and you want to failback from secondary site, follow the below steps:

Prerequisites

- The Kubernetes resources of the Primary cluster have been backed up and restored to the Secondary cluster, including PVCs, PVs, application workloads, etc.

Procedures

1. Scale down the running applications (if any) on the primary site. Ensure that all persistent volumes in use by the workload are no longer in use on the primary cluster.
2. Update VolumeReplication CR replicationState from primary to secondary on the primary site.
3. Scale down the applications on the secondary site.
4. Update VolumeReplication CR replicationState state from `primary` to `secondary` in secondary site.
5. On the primary site, verify the VolumeReplication status is marked as volume ready to use.
6. Once the volume is marked to ready to use, change the replicationState state from secondary to primary in primary site.
7. Scale up the applications again on the primary site.

Object Storage Disaster Recovery

The Ceph RGW Multi-Site feature is a cross-cluster asynchronous data replication mechanism designed to synchronize object storage data between geographically distributed Ceph clusters, providing High Availability (HA) and Disaster Recovery (DR) capabilities.

TOC

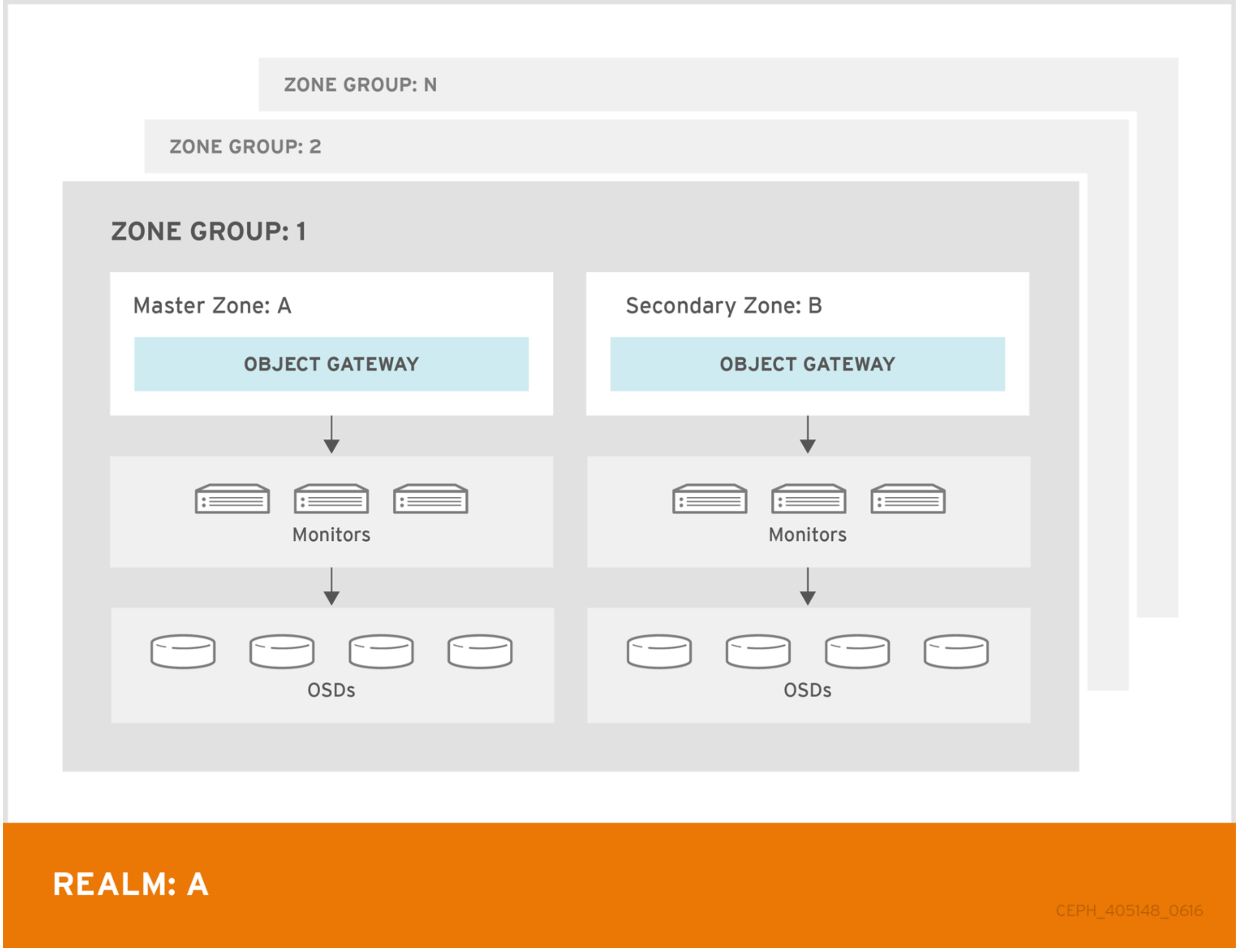
Terminology

Term	Explanation
Primary Cluster	The cluster currently providing storage services.
Secondary Cluster	The standby cluster used for backup purposes.
Realm, ZoneGroup, Zone	• Realm: The highest-level logical grouping in Ceph object storage. It represents a complete object storage namespace, typically used for multi-site replication and synchronization. A Realm can span different geographical locations or data centers. • ZoneGroup: A logical grouping within a Realm, containing multiple Zones. ZoneGroups enable data synchronization and replication across Zones, usually within the same geographical region. • Zone: A logical grouping within a ZoneGroup that physically stores data. Each Zone manages and stores objects independently and can have its own data/metadata pool configurations.

Prerequisites

- Prepare two clusters available for deploying Rook-Ceph (Primary and Secondary clusters) with network connectivity between them.
- Both clusters must use the same platform version (v3.12 or later).
- Ensure no Ceph object storage is deployed on either the Primary or Secondary cluster.
- Refer to the Create Storage Service documentation to deploy Operator and create clusters. Do not proceed with object storage pool creation via the wizard after cluster creation. Instead, use CLI tools for configuration as described below.

Architecture



Procedures

This guide provides a synchronization solution between two Zones in the same ZoneGroup.

1 Create Object Storage in Primary Cluster

This step creates the Realm, ZoneGroup, Primary Zone, and Primary Zone's gateway resources.

Execute the following commands on the Control node of the Primary cluster:

1. **Set Parameters**

```
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>
```

Parameters description:

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

2. **Create Object Storage**

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

Configure External Access for Primary Zone

1. Obtain the UID of the ObjectStore

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

2. Create an external access Service

```
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. Add external endpoints to the 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 Obtain `access-key` and `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 Create Secondary Zone and Configure Realm Sync

This section explains how to create the Secondary Zone and configure synchronization by pulling Realm information from the Primary cluster.

Execute the following commands on the Control node of the Secondary cluster:

1. **Set Parameters**

```
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>
```

Parameters description:

- `<realm-name>`: Realm name.

- `<zone-group-name>`: ZoneGroup name.
- `<realm-endpoint>`: External address obtained from the Primary cluster.
- `<access-key>`: AK obtain from here.
- `<secret-key>`: SK obtain from here.
- `<secondary-zone-name>`: Secondary Zone name.
- `<secondary-object-store-name>`: Secondary Gateway name.

2. **Create Secondary Zone and Configure Realm Sync**

```
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 Configure External Access for Secondary Zone

1. Obtain UID of Secondary Gateway

```
export SECONDARY_OBJECT_STORE_UID=$(kubectl -n rook-ceph get cephobjectstore $SECONDARY_OBJECT_STORE_NAME -o jsonpath='{.metadata.uid}')
```

2. Create an external access Service

```
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. Add external endpoints to the 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-$SECONDARY_OBJECT_STORE_NAME-external -o jsonpath='{.spec.ports[0].nodePort}')
ENDPOINT=http://$IP:$PORT
kubectl -n rook-ceph patch cephobjectzone $SECONDARY_ZONE_NAME --type merge -p "{\"spec\":{\"customEndpoints\":[\"$ENDPOINT\"]}}"
```

6 Check Ceph Object Storage Synchronization Status

Execute the following commands in the `rook-ceph-tools` pod of the Secondary cluster

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

radosgw-admin sync status
```

Output example

```
          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` and `data is caught up with source` means sync status is healthy.

Failover

When the Primary cluster fails, it is necessary to promote the Secondary Zone to the Primary Zone. After the switch, the Secondary Zone's gateway can continue to provide object storage services.

Procedures

Execute the following commands in the `rook-ceph-tools` pod of the Secondary cluster

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

# Make the recovered zone the master and default zone
radosgw-admin zone modify --rgw-realm=<realm-name> --rgw-zonegroup=<zone-group-name> --rgw-zone=<secondary-zone-name> --master

# Update the period so that the changes take effect
radosgw-admin period update --commit --rgw-realm=<realm-name> --rgw-zonegroup=<zone-group-name>
```

Parameters

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

Failback

Once the failed cluster is recovered on the primary site and you want to failback from secondary site, follow the below stepss

Procedures

Execute the following commands in the `rook-ceph-tools` pod of the Primary cluster

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

# check sync status, wait for sync from secondary site
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

# Make the recovered zone the master and default zone
radosgw-admin zone modify --rgw-realm=<realm-name> --rgw-zonegroup=<zone-group-name> --rgw-zone=<primary-zone-name> --master

# Update the period so that the changes take effect
radosgw-admin period update --commit --rgw-realm=<realm-name> --rgw-zonegroup=<zone-group-name>
```

Parameters

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

Update the optimization parameters

The platform supports filling in optimization parameters in Ceph configuration file format when creating a storage cluster, but does not provide a way to modify them through the interface after creation. You need to manually update them according to the following steps.

TOC

Procedure

1. First, update the storage optimization parameters to the Configmap named `rook-config-override-user`, replace the `.data.config` field, and set the value of the `.metadata.annotations[rook.cpaas.io/need-sync]` field to `true`. For example:

```
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. Execute `ceph tell [mon|osd|mgr|mds|rgw].* config set [key] [value]` in the Pod of rook-ceph-tools to apply the configuration in real time.
3. To start the Pod of tools, edit the ClusterServiceVersion (CSV) under the rook-ceph namespace and set the replicas value of rook-ceph-tools in the Deployments section to 1.

Create Ceph Object Store User

We allows creation and customization of object store users through the custom resource definitions (CRDs).

TOC

Prerequisites

- The object storage pool has been created

Procedure

1 Create User

Execute commands on the **control node** of the cluster.

```
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
```

Parameters

Parameters	Description
name	The name of the object store user to create.
namespace	The namespace of the object store user is created.
displayName	The display name.
clusterNamespace	The namespace where the parent `CephCluster` and `CephObjectStore` are found. If not specified, the user must be in the same namespace as the cluster and object store. To enable this feature, the CephObjectStore `allowUsersInNamespaces` must include the namespace of this user.
ObjectStore	The object store in which the user will be created. This matches the name of the object storage pool.
quotas	**optional** This represents quota limitation can be set on the user.

<table>
<tr><th>Parameters</th><th>Description</th></tr>
<tr><td></td><td>maxBuckets: The maximum bucket limit for the user. Defaults to <code>100</code>.
maxSize: Maximum size limit of all objects across all the user's buckets. Set to <code>-1</code> indicates no restriction.
maxObjects: Maximum number of objects across all the user's buckets. Set to <code>-1</code> indicates no restriction.</td></tr>
<tr><td>capabilities</td><td> optional Ceph allows users to be given additional permissions. This setting can currently only be used during the creation of the object store user. If a user's capabilities need modified, the user must be deleted and re-created. See the Ceph docs for more info. We supports adding <code>read</code>, <code>write</code>, <code>read,write</code>, or <code>*</code> permissions for the following resources: user
buckets
usage
metadata
zone
roles
info
amz-cache
bilog
mdlog
datalog
user-policy
odic-provider
ratelimit</td></tr>
</table>

2 Allow create user in other namespaces

If a CephObjectStoreUser is created in a namespace other than the Rook cluster namespace, the namespace must be added to this list of allowed namespaces, or specify "*" to allow all namespaces. This is useful for applications that need object store credentials to be created in their own namespace.

Execute commands on the **control node** of the cluster.

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

3 Get user information

Execute commands on the **control node** of the cluster.

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

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

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

Setting Storage Pool Quotas

Pool quota is a logical data size limit applied at the Ceph storage pool level. It controls how much logical data can be written by the pool, the actual physical space will be **quota * pool replication**.

TOC

Prerequisites

- A Ceph cluster is installed
- Ceph pools are created

Set pool quota for File Storage Pool

Execute commands on the **control node** of the cluster.

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\"}]"
```

Parameters

Parameters	Description
size	quota in bytes as a string with quantity suffixes (e.g. "10Gi")
fs-pool-name	The name of the File Storage Pool.

Set pool quota for Block Storage Pool

Execute commands on the **control node** of the cluster.

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\"}]"
```

Parameters

Parameters	Description
size	quota in bytes as a string with quantity suffixes (e.g. "10Gi")
block-pool-name	The name of the Block Storage Pool.

Set pool quota for Object Storage Pool

Execute commands on the **control node** of the cluster.

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\"}]"
```

Parameters

Parameters	Description
size	quota in bytes as a string with quantity suffixes (e.g. "10Gi")
object-pool-name	The name of the Object Storage Pool.

Validate pool quota via 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
```

MinIO Object Storage

Introduction

Install

Architecture

Conclusion:

Concepts

Guides

How To

Introduction

Alauda Container Platform (ACP) Object Storage with MinIO is an object storage service licensed under the Apache License v2.0. It is compatible with the Amazon S3 cloud storage service interface, making it particularly suitable for storing large volumes of unstructured data, such as images, videos, log files, backup data, and container/virtual machine images. An object file can range in size from a few KB to a maximum of 5T.

The main advantages are as follows:

- **Simplicity**: Minimalism is the guiding design principle of MinIO, allowing for out-of-the-box functionality. Simplicity reduces the chances of errors, increases uptime, and enhances reliability while also boosting performance.
- **High Performance**: MinIO is a world leader in object storage. On standard hardware, read/write speeds can reach up to 183 GB/sec and 171 GB/sec.
- **Scalability**: Multiple small to medium-sized, easily manageable clusters can be established, supporting the aggregation of multiple clusters into a super-large resource pool across data centers, rather than directly adopting a large-scale, centrally managed distributed cluster.
- **Cloud-Native**: Compliant with all native cloud computing architectures and build processes, and incorporates the latest technologies and concepts in cloud computing, making object storage more user-friendly for Kubernetes.

Install

Alauda Container Platform (ACP) Object Storage with MinIO is an object storage service based on the Apache License v2.0 open-source protocol. It is compatible with the Amazon S3 cloud storage service interface and is ideal for storing large volumes of unstructured data, such as images, videos, log files, backup data, and container/virtual machine images. An object file can be of any size, ranging from a few kilobytes to a maximum of 5 terabytes.

TOC

Prerequisites

- MinIO is built on underlying storage, so please ensure that a storage class has been created in the current cluster. TopoLVM is recommended.

- **Download** the **Alauda Container Platform Storage Essentials** installation package corresponding to your platform architecture.
- **Upload** the **Alauda Container Platform Storage Essentials** installation package using the Upload Packages mechanism.
- **Download** the **Alauda Container Platform (ACP) Object Storage with MinIO** installation package corresponding to your platform architecture.
- **Upload** the **Alauda Container Platform (ACP) Object Storage with MinIO** installation package using the Upload Packages mechanism.

Procedure

1 Deploy Alauda Container Platform Storage Essentials

1. Login, go to the **Administrator** page.
2. Click **Marketplace** > **OperatorHub** to enter the **OperatorHub** page.
3. Find the **Alauda Container Platform Storage Essentials**, click **Install**, and navigate to the **Install Alauda Container Platform Storage Essentials** page.
   Configuration Parameters:

Parameter	Recommended Configuration
Channel	The default channel is `stable`.
Installation Mode	`Cluster`: All namespaces in the cluster share a single Operator instance for creation and management, resulting in lower resource usage.
Installation Place	Select `Recommended`, Namespace only support **acp-storage**.
Upgrade Strategy	`Manual`: When there is a new version in the Operator Hub, manual confirmation is required to upgrade the Operator to the latest version.

2 Deploy Operator

1. In the left navigation bar, click **Storage** > **Object Storage**.
2. Click **Configure Now**.
3. On the **Deploy MinIO Operator** wizard page, click **Deploy Operator** at the bottom right.

- Once the page automatically proceeds to the next step, it indicates that the Operator deployment was successful.
- If the deployment fails, refer to the interface prompts to **Clean Up Deployed Information and Retry**, and redeploy the Operator.

3 Create Cluster

1. On the **Create Cluster** wizard page, configure the basic information.

Parameter	Description
Access Key	Access key ID. A unique identifier associated with a private access key; used with the access key ID to encrypt and sign requests.
Secret Key	Private access key used in conjunction with the access key ID to encrypt and sign requests, identify the sender, and prevent request tampering.

2. In the **Resource Configuration** area, configure specifications as per the following instructions.

Parameter	Description
Small scale	Suitable for handling up to 100,000 objects, supporting concurrent access of no more than 50 in test environments or data backup scenarios. The CPU resource request and limit are set to 2 cores by default, and the memory resource request and limit are set to 4 Gi.

Parameter	Description
Medium scale	Designed for enterprise-level applications requiring storage of 1,000,000 objects and capable of handling up to 200 concurrent requests. The CPU resource request and limit are set to 4 cores by default, and the memory resource request and limit are set to 8 Gi.
Large scale	Designed for group users with storage needs of 10,000,000 objects and handling up to 500 concurrent requests, suitable for high-load scenarios. The CPU resource request and limit are set to 8 cores by default, and the memory resource request and limit are set to 16 Gi.
Custom	Offers flexible configuration options for professional users with specific needs, ensuring precise matching of service scale and performance requirements. Note: When configuring custom specifications, ensure that: • The CPU resource request is greater than 100 m. • The memory resource request is greater than or equal to 2 Gi. • The CPU and memory resource limits are greater than or equal to the resource requests.

3. In the **Storage Pool** area, configure related information as per the following instructions.

Parameter	Description
Instance Number	Increasing the number of instances in a MinIO cluster can significantly enhance system performance and reliability, ensuring high data availability. However, too many instances can lead to the following issues: • Increased resource consumption. • If a node hosts multiple instances, a node failure may cause multiple instances to go offline simultaneously, reducing overall cluster reliability.

Parameter	Description
	Note: • The minimum number of instances that can be entered is 4. • If the number of instances is greater than 16, the entered value must be a multiple of 8. • When adding additional storage pools, the number of instances must be no less than the first storage pool's number of instances.
Single Storage Volume	Capacity of a single storage volume PVC. Each storage service manages one storage volume. After entering the capacity of a single storage volume, the platform will automatically calculate the storage pool capacity and other information, which can be viewed in the **Storage Pool Overview**.
Underlying Storage	The underlying storage used by the MinIO cluster. Please select a storage class that has been created in the current cluster. TopoLVM is recommended.
Storage Nodes	Select the storage nodes required by the MinIO cluster. It is recommended to use 4-16 storage nodes. The platform will deploy one storage service for each selected storage node.
Storage Pool Overview	For specific parameters and calculation formulas, please refer to Storage Pool Overview.

4. In the **Access Configuration** area, configure related information as per the following instructions.

Parameter	Description
External Access	When enabled, it supports cross-cluster access to MinIO; when disabled, it only supports access within the cluster.
Protocol	Supports HTTP and HTTPS; when selecting HTTPS, you need to enter the **Domain** and import the **Public Key** and **Private Key** of the domain

Parameter	Description
	name certificate. **Note**: • When the access protocol is HTTP, pods within the cluster can access MinIO directly via the obtained IP or domain name without configuring IP address and domain name mapping; nodes within the cluster can access MinIO directly via the obtained IP, and if domain name access is required, manual configuration of IP address and domain name mapping is needed; external access can be done directly via the obtained IP. • When the access protocol is HTTPS, access to MinIO via IP is not possible both inside and outside the cluster. Manual configuration of the mapping between the obtained IP address and the domain name entered during cluster creation is required to access normally via the domain name.
Access Method	• **NodePort**: Opens a fixed port on each compute node host to expose the service externally. When configuring domain name access, it is recommended to use VIP for domain name resolution to ensure high availability. • **LoadBalancer**: Uses a load balancer to forward traffic to backend services. Before use, please ensure that the MetalLB plugin is deployed in the current cluster and there are available IPs in the external address pool.

5. Click **Create Cluster** at the bottom right.

- Once the page automatically proceeds to **Cluster Details**, it indicates that the cluster creation was successful.
- If the cluster remains in the creation process, you can click **Cancel**. After cancellation, the deployed cluster information will be cleaned up, and you can return to the cluster creation page to recreate the cluster.

4 Create Bucket

Log in to the control node of the cluster and use the command to create a bucket.

1. On the cluster details page, click the **Access Method** tab to view the MinIO access address, or use the following command to query.

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

Note:

- Replace `tenant ns` with the actual namespace `minio-system`.
- Example: `kubectl get svc -n minio-system minio | grep -w minio | awk '{print $3}'`

2. Obtain the mc command.

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

3. Configure MinIO cluster alias.

- IPv4:

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

- IPv6:

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

- Domain Name:

```
mc --insecure alias set <minio cluster alias> http://<domain name>:<port> <accessKey> <secretKey>
mc --insecure alias set <minio cluster alias> https://<domain name>:<port> <accessKey> <secretKey>
```

Note:

- Enter the IP address obtained in step 1 for `minio endpoint`.
- Enter the **Access Key** and **Secret Key** created during cluster creation for `accessKey` and `secretKey`.
- Configuration examples:
  - 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@`
  - Domain Name: `mc --insecure alias set myminio http://test.minio.alauda:80 07Apples@ 07Apples@` or `mc --insecure alias set myminio https://test.minio.alauda:443 07Apples@ 07Apples@`

4. Create a bucket.

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

5 Upload/Download Files

Once the bucket is created, you can use the command line to upload files to the bucket or download existing files from the bucket.

1. Create a file for upload testing. This step can be skipped if uploading an existing file.

```
touch <file name>
```

2. Upload files to the bucket.

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

3. View files in the bucket to confirm successful upload.

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

4. Delete uploaded files.

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

Related Information

Redundancy Factor Mapping Table

Note: When adding additional storage pools, the redundancy factor needs to be calculated based on the number of instances in the first storage pool.

Instance Number	Redundancy Factor
4 - 5	2
6 - 7	3
>= 8	4

Storage Pool Overview

Storage Pool Overview Parameter	Calculation Formula
Usable Capacity	When the Instance Number ≤ 16, Usable Capacity = Single Storage Volume Capacity × (Instance Number - Redundancy Factor).
When the number of instances > 16, Usable Capacity = Single Storage Volume Capacity × (Instance Number - 4 × (Instance Number + 15) / 16). The result of "4 × (Instance Number + 15) / 16" should be rounded down.	
Total Capacity	Total Capacity = Instance Numbers × Single Storage Volume Capacity

Storage Pool Overview Parameter	Calculation Formula
Number of failover storage services tolerated	When the Instance Number > 2 × Redundancy Factor, Number of Tolerable Fault Storage Services = Redundancy Factor.
When the Instance Number = 2 × Redundancy Factor, the number of tolerable fault storage services = Redundancy Factor - 1	

Architecture

Alauda Container Platform (ACP) Object Storage with MinIO is a high-performance, distributed object storage system designed for cloud-native environments. It leverages erasure coding, distributed storage pools, and high-availability mechanisms to ensure data durability and scalability in Kubernetes.

TOC

Core Components:

- **MinIO Operator**: Manages the deployment and upgrade of MinIO clusters.
- **MinIO Peer**: Configures and manages MinIO's site replication functionality.
- **MinIO Pool**: The core component of MinIO, responsible for handling object storage requests. Each pool corresponds to a StatefulSet and provides storage resources.

Deployment Architecture:

Deploying MinIO in Kubernetes requires defining a MinIO tenant, specifying the number of server instances (pods) and the number of volumes (drives) per instance. Each MinIO server is managed via a StatefulSet, ensuring stable identities and persistent storage. MinIO aggregates all drives into one or more erasure sets and applies erasure coding for fault tolerance.

Multi-Pool Expansion:

MinIO clusters can scale by adding additional server pools, each with its own erasure set. While this provides greater storage capacity, it introduces complexity in cluster maintenance and reduces overall cluster reliability. A failure in any server pool can render the entire MinIO cluster unavailable, even if other pools remain operational.

Conclusion:

MinIO is a highly scalable, cloud-native object storage solution that balances performance and reliability. When architecting a MinIO cluster, it is crucial to carefully design storage pools, configure erasure coding settings, and implement high-availability strategies to ensure data integrity and operational stability in Kubernetes environments.

Concepts

Core Concepts

- **Erasure Coding (EC)**: MinIO employs Reed-Solomon erasure coding to break objects into data and parity shards, distributing them across multiple drives to ensure fault tolerance. For example, in a 16-drive setup, data can be split into 12 data shards and 4 parity shards, allowing the system to rebuild data even if up to 4 drives fail.
- **Server Pools & Erasure Sets**: MinIO Server Pools are logical groupings of storage resources, where each pool consists of multiple nodes sharing storage and compute capabilities. Within a pool, drives are automatically organized into one or more **Erasure Sets**.
  - **Data Distribution**: When an object is stored, it is split into data and parity shards and distributed across different drives within an erasure set.
  - **Redundancy Model**: Erasure sets form the fundamental unit of data redundancy, ensuring resiliency based on configured data-to-parity shard ratios.
  - **Scalability**: A single MinIO storage pool can contain multiple erasure sets, and new data is always written to the erasure set with the most available capacity.

Guides

Adding a Storage Pool

A storage pool refers to a logical partition used for storing data. Different types of underlying storage can be used simultaneously within the same storage cluster to meet various business needs.

In addition to the storage pools created during the configuration of object storage, you can also add additional storage pools.

TOC

Notes

Adding a storage pool will cause a brief interruption in the MinIO service, but it will automatically recover to a normal state afterward.

Procedure

1. Go to **Administrator**.
2. Click on **Storage Management** > **Object Storage** in the left navigation bar.
3. Under the **Cluster Information** tab, scroll down to the **Storage Pool** section and click **Add Storage Pool**.

4. Configure the relevant parameters according to the instructions below.

Parameter	Description
Underlying Storage	The underlying storage used by the MinIO cluster. Please select an existing storage class created in the current cluster, with TopoLVM recommended.
Storage Nodes	Select the storage nodes required for the MinIO cluster. It is recommended to use 4-16 storage nodes; the platform will deploy 1 storage service for each selected storage node. **Note**: When using 3 storage nodes, to ensure reliability, 2 storage services will be deployed for each node.
Single Storage Volume	The capacity of a single storage volume PVC. Each storage service manages 1 storage volume, and once the capacity of a single storage volume is entered, the platform will automatically calculate the storage pool capacity and other information, which can be viewed in the **Storage Pool Overview**.

5. Click **Confirm**.

Monitoring & Alerts

The object storage system comes with built-in monitoring and alerting capabilities, covering storage clusters, service health, and resource utilization. It also supports configurable notification policies to keep your operations team informed. Real-time monitoring insights help with performance tuning and operational decision-making, while automated alerts ensure the stability and reliability of your storage system.

TOC

Monitoring

By default, the platform collects key metrics on storage clusters and service status. You can access real-time monitoring data under **Storage Management** > **Object Storage** > **Monitoring**.

Storage Overview

This section provides a high-level view of storage system health, service status, and raw capacity utilization. If the storage status is abnormal, alert details will indicate the root cause, helping you diagnose and resolve issues efficiently.

Cluster Monitoring

Track raw capacity usage and I/O performance trends across your storage cluster. This helps identify storage bottlenecks, optimize resource allocation, and ensure smooth data operations.

Object Monitoring

Monitor access patterns, including total request counts and failed requests. These insights help analyze storage workload and detect anomalies that may indicate service disruptions or security risks.

Alerts

The platform comes with pre-configured alerting policies to detect anomalies and trigger notifications when predefined thresholds are reached. These built-in rules cover essential areas such as component health, capacity usage, and user data integrity.

Configuring Notifications

To ensure timely responses, configure notification policies in the **Operations Center**. Alerts can be sent via email, SMS, or other channels to notify the right personnel. Fine-tune your settings to match your organization's incident response workflow.

Handling Alerts

- **Cluster in "Alert" state**: A warning has been triggered, and system stability may be at risk. Check the **Live Alerts** section for details, identify the root cause, and take corrective actions.

- **Cluster in "Failure" state**: The storage cluster is no longer operational. Immediate intervention is required to restore service availability.

The platform categorizes alerts into different severity levels, helping teams prioritize incident response:

Severity	Description
Critical	A system failure impacting business operations or causing data loss. Immediate action required.
Major	A known issue that may lead to functionality breakdowns, potentially disrupting business processes.
Warning	A potential risk that, if unaddressed, could impact performance or availability.

Post-Incident Analysis

The **Alert History** logs all past incidents, providing valuable data for post-mortem analysis and system improvements. When reviewing past alerts, consider the following:

1. What were the exact symptoms when the incident occurred?
2. Are certain alerts repeating over time? Can proactive measures be taken to prevent recurrence?
3. Did a specific time window show a spike in alerts? Was it caused by an operational issue or an external factor? Should the response strategy be adjusted?

By continuously analyzing alert patterns and refining monitoring strategies, teams can enhance system resilience, minimize downtime, and ensure seamless storage operations.

How To

Data Disaster Recovery

Data Disaster Recovery

MinIO supports the establishment of a disaster recovery center through remote data backup or active-active deployment to ensure that original data is not lost or damaged in the event of a disaster, thereby guaranteeing data security and reliability.

TOC

Applicable Scenarios

- **Hot Backup**: There are two data centers in the same city or in different locations, one primary and one backup. Data is replicated in real-time from the primary cluster to the backup cluster to ensure data consistency. When a disaster occurs in the primary cluster, business traffic can be seamlessly switched to the backup cluster to ensure business continuity.
- **City-Level Active-Active**: In a city-level active-active (multi-cluster) architecture, there are two data centers located in different clusters. Both data centers are active and can receive business traffic simultaneously. When one data center encounters a disaster, business can continue running uninterrupted in the other data center.

Terminology

- **Primary Cluster**: Refers to the cluster that is currently active and processing business requests. It is the source of the data or the initiator of operations. In the primary cluster, data is created, modified, or updated, and business traffic is first sent to this cluster for processing.
- **Target Cluster**: Refers to the cluster that receives data replication, migration, or failover. It is typically in a backup or standby state, waiting to receive data from the primary cluster or take over business traffic. When the primary cluster fails or needs to switch, the target cluster will receive data copies from the primary cluster or take over business traffic to ensure business continuity. In an active-active scenario, both clusters can serve as each other's target cluster.

Prerequisites

- Both the primary cluster and the target cluster must have external network access enabled. For specific configuration methods, please refer to Create Object Storage.
- The primary cluster must use the **LoadBalancer** access method, while the target cluster is recommended to support **load balancing** functionality.
- The primary cluster and the target cluster must use the same access protocol, i.e., either both use HTTP or both use HTTPS.
- When using the HTTPS protocol, both the primary cluster and the target cluster need to configure DNS resolution for themselves and each other.
- When using the HTTPS protocol, it is recommended that both the primary cluster and the target cluster use CA-signed certificates to ensure secure and trusted communication; if self-signed certificates are used, both parties must import and trust each other's self-signed certificates to establish a secure HTTPS connection successfully.

Operation Steps

1. Enter **Administrator**.
2. In the left navigation bar, click **Storage Management** > **Object Storage**.

3. On the **Data Disaster Recovery** tab, click **Add Target Cluster**.
4. Configure the relevant parameters for the target cluster according to the following instructions.

Parameter	Description
Access Address	The external access address of the target cluster, starting with http:// or https://.
Access Key	The Access Key ID for the target cluster. A unique identifier associated with the private access key; used in conjunction with the private access key to encrypt requests.
Secret Key	The private access key used in conjunction with the Access Key ID to encrypt requests, identify the sender, and prevent request modification.

5. Click **Add**.

- Upon successful addition, you will be able to view the status of the target cluster and the synchronization status between clusters.

Parameter	Description
Cluster Status	The status of the target cluster, including **Healthy**, **Abnormal**, or **Unknown**.
Buckets	The number of buckets pending synchronization and those already synchronized. • In hot backup scenarios, pending synchronization refers to the number of buckets that the primary cluster needs to synchronize with the target cluster. • In city-level active-active scenarios, pending synchronization refers to the total number of buckets that need to be synchronized between the primary and target clusters.

Parameter	Description
Objects	The number of objects that failed to synchronize in the bucket. **Note**: This number is for reference only, as MinIO synchronizes related file configurations during synchronization.
Network Traffic Rate	The network ingress and egress rate of the primary cluster. • In hot backup scenarios, the network ingress rate is always 0. • In city-level active-active scenarios, both ingress and egress rates have data.

- If the addition of the target cluster fails, you can click **Re-add** to clear the cluster information and return to the add target cluster page, where you can re-add the target cluster.

Related Operations

When disaster recovery is no longer needed, you can click **Remove Target Cluster**. Removing the target cluster does not delete the data that has been synchronized; if any data is currently synchronizing, it will be interrupted.

TopoLVM Local Storage

Introduction

Install

Guides

How To

Introduction

TopoLVM is a Container Storage Interface (CSI) plugin designed specifically for Kubernetes, aimed at providing efficient and convenient management of local storage volumes.

Key features and advantages:

- **Local Volume Management**: TopoLVM focuses on managing local storage devices (such as disks and SSDs) on Kubernetes nodes. Compared to traditional network storage, local volumes offer lower latency and higher performance.
- **Topology Awareness**: TopoLVM can recognize the topology of Kubernetes clusters (e.g., nodes, availability zones), allowing it to automatically allocate storage volumes to the same node based on the actual scheduling location of Pods, further optimizing performance.
- **Dynamic Volume Allocation**: TopoLVM supports dynamically creating, deleting, and resizing storage volumes without manual intervention, significantly simplifying operations and reducing complexity.
- **Deep Integration with Kubernetes**: As a CSI plugin, TopoLVM seamlessly integrates with Kubernetes storage management APIs, enabling users to manage local volumes directly through standard Kubernetes resource objects such as PersistentVolumeClaims.

In summary, TopoLVM addresses common challenges associated with using local storage in Kubernetes, such as manual management, lack of topology awareness, and insufficient dynamic allocation capabilities. It provides a more efficient and user-friendly solution for applications requiring high-performance local storage, such as databases and caches.

Install

Local storage is a software-defined server-local storage solution that provides a simple, easy-to-maintain, and high-performance local storage service capability. Based on the community's TopoLVM solution, it achieves persistent volume orchestration management of local storage through the system's LVM approach.

TOC

Prerequisites

- The lvm2 package must be installed on each node of the storage cluster. If not installed, please execute the `yum install -y lvm2` command on the node.
- **Download** the **Alauda Container Platform Storage Essentials** installation package corresponding to your platform architecture.
- **Upload** the **Alauda Container Platform Storage Essentials** installation package using the Upload Packages mechanism.
- **Download** the **Alauda Build of TopoLVM** installation package corresponding to your platform architecture.

- **Upload** the **Alauda Build of TopoLVM** installation package using the Upload Packages mechanism.

Procedure

1 Deploy Alauda Container Platform Storage Essentials

1. Login, go to the **Administrator** page.
2. Click **Marketplace** > **OperatorHub** to enter the **OperatorHub** page.
3. Find the **Alauda Container Platform Storage Essentials**, click **Install**, and navigate to the **Install Alauda Container Platform Storage Essentials** page.
   Configuration Parameters:

Parameter	Recommended Configuration
Channel	The default channel is `stable`.
Installation Mode	`Cluster`: All namespaces in the cluster share a single Operator instance for creation and management, resulting in lower resource usage.
Installation Place	Select `Recommended`, Namespace only support **acp-storage**.
Upgrade Strategy	`Manual`: When there is a new version in the Operator Hub, manual confirmation is required to upgrade the Operator to the latest version.

2 Deploy Storage

1. Go to **Administrator**.
2. In the left navigation bar, click **Storage Management** > **Local Storage**.
3. Click **Configure Now**.
4. On the **Install Operator** wizard page, click **Start Deployment**.

- When the page automatically proceeds to the next step, it indicates that the Operator deployment was successful.
- If the deployment fails, please refer to the interface prompts for resolution. Then click **Clean Up** and redeploy the Operator.

5. On the **Create Cluster** wizard page, add devices.

Parameter	Description
Select Node	Node with at least 1 bare disk.
Device Class	Each device class corresponds to a set of storage devices with the same characteristics. It is recommended to fill in the name based on the disk nature, such as *hdd*, *ssd*.
Device Type	Only disk types are supported.
Storage Device	For example, */dev/sda*. If there are multiple disks, they can be added one by one.
Snapshot	When enabled, it supports creating PVC snapshots and using the snapshots to configure new PVCs for quick backup and recovery of business data. If the snapshot was not enabled when creating the storage, you can still enable it as needed in the **Operations** section of the storage cluster details page. **Note**: Before use, please ensure that the Volume Snapshot Plugin has been deployed for the current cluster.

- Click Next. When the page automatically proceeds to the next step, it indicates that the cluster deployment was successful.
- If the creation fails, please refer to the interface prompts and clean up resources in a timely manner.

6. On the **Create Storage Class** wizard page, configure the relevant parameters.

Parameter	Description
Name	The name of the storage class. It must be unique within the current cluster.
Display Name	A name that helps you identify or filter, such as a Chinese description of the storage class.
Device Class	The device class is a way to categorize storage devices in TopoLVM. Each device class corresponds to a set of storage devices with the same characteristics. If there are no special requirements, you can use the **Auto-Allocated** device class from the cluster.
File System	- **XFS** is a high-performance journaling file system adept at handling parallel I/O workloads, supporting large file processing and providing smooth data transfer. - **EXT4** is a journaling file system in Linux, offering extent file storage methods and supporting large file processing. The file system can reach a capacity of 1 EiB, with a maximum supported file size of 16 TiB.
Recycling Policy	The recycling policy for persistent volumes. - Delete: When the persistent volume claim is deleted, the bound persistent volume is also deleted. - Retain: Even if the persistent volume claim is deleted, the bound persistent volume will still be retained.
Access Mode	ReadWriteOnce (RWO): Can be mounted by a single node in read-write mode.
Allocation Project	This type of persistent volume claim can only be created in specific projects. If no project is assigned temporarily, the project can also be **Updated** later.

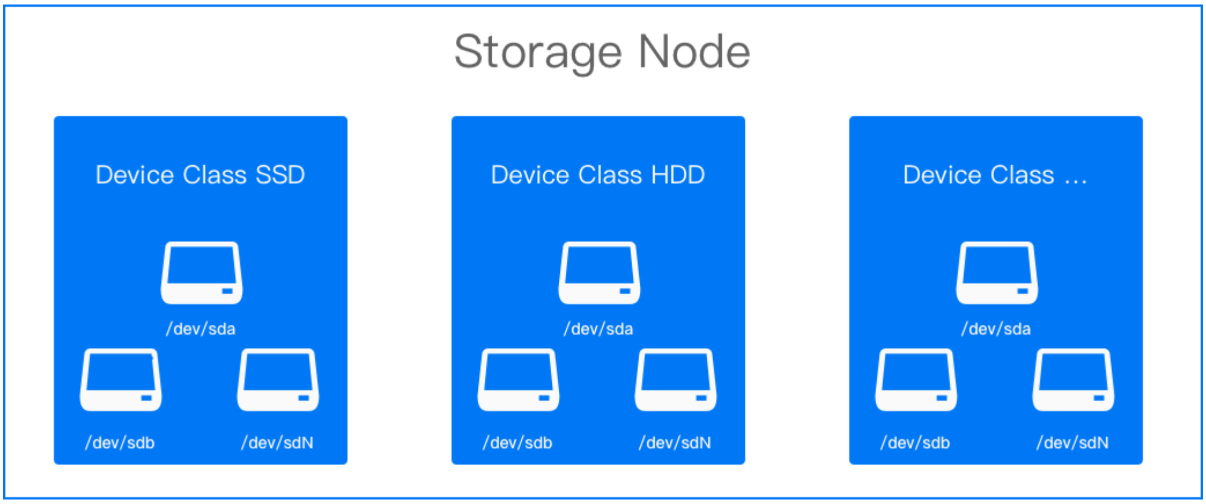
7. Click **Next** and wait for the resource creation to complete.

Guides

Device Management

Whether for initial deployment or resource expansion, you need to map the available disks on the node into storage devices for use and management.

Storage devices with similar characteristics are typically used in a centralized manner, and these devices are categorized under **Device Classes** in local storage. Using device classes is equivalent to directly using disks, ensuring zero loss and high performance, while also reducing application awareness and dependence on specific devices.



TOC

Prerequisites

- At least 1 Device Class (deviceClasses.classes) must have been added when creating the local storage cluster, including devices in the device class.
- There must be at least 1 bare disk present on the node.

Adding Devices

1. Go to **Administrator**.
2. In the left navigation bar, click **Storage Management** > **Local Storage**.
3. In the **Details** tab, click **Add Storage Node**.
4. Configure the related parameters according to the instructions below.

Parameter	Description
Storage Node	A node that has at least 1 bare disk.
Device Class	Each device class corresponds to a group of storage devices with the same characteristics; it is recommended to name it according to the nature of the disks, e.g., *hdd*, *ssd*.
Storage Device	For example, */dev/sda*. If there are multiple disks, they can be added one by one. **Note**: The storage device should be the entire hard disk, not a partition on the hard disk, as this will cause errors.

5. Click **Add**.
   Note: If the device class status is `Unavailable` due to the lack of added devices, you can proceed with the following operations.
6. Switch to the **Storage Devices** tab and click **Add Storage Device**.
7. Add devices according to the prompts on the interface.
8. Click **Add**.

Monitoring and Alerting

Local storage provides out-of-the-box monitoring metrics collection and alerting capabilities. Once the platform monitoring component is enabled, monitoring and alerts can be configured based on storage clusters, storage performance, and storage capacity, with support for configuring notification policies.

The intuitively presented monitoring data can be utilized to support decision making for operational inspections or performance tuning, and a comprehensive alerting mechanism will help ensure the stable operation of the storage system.

TOC

Monitoring

Performance Monitoring

By default, the platform collects commonly used performance monitoring metrics such as read and write bandwidth, IOPS, and latency for local storage. Real-time monitoring data for these metrics can be viewed on the **Monitoring** tab of the **Local Storage** page under **Storage Management**. The platform displays these metrics visually through graphs and charts, allowing administrators to clearly observe current storage performance and quickly identify potential issues.

Capacity Monitoring

Since local storage can only use locally available storage resources on nodes, users must ensure there is sufficient available capacity on the nodes before declaring local storage to avoid issues caused by over-declaring.

To assist with this, the platform provides detailed capacity monitoring in the **Details** section of local storage, categorized by device types. Users can check available storage space clearly displayed in numerical and graphical formats. If any device type shows insufficient available capacity, space should be cleared or additional disk devices added before using local storage.

Alerts

The platform includes a set of default alerting policies. If resources become abnormal or monitoring data reaches a warning threshold, alerts are automatically triggered. The preconfigured alerting policies effectively cover common operational needs, including alerts for cluster health status and device type capacity.

Configuring Notifications

To ensure alerts are received in a timely manner, notification policies should be configured in the operations center. Notifications can be sent through email, SMS, or other methods to relevant personnel, prompting immediate attention to resolve issues or prevent failures. Users can access the notification policy settings directly from the operations center interface. Detailed instructions on configuring alerts can be found in the [Creating Alert Policies] documentation.

Handling Alerts

- If the health status of the storage cluster changes to `Alert`, administrators should investigate immediately. The **Details** section provides information for troubleshooting and resolving these issues. Common causes include abnormal node services or problems with specific device types.

Inspection Item	Corresponding Status	Cause
Health Status	Alert	Caused by abnormal node services or device type issues.
Service Status	Unknown	Node is in a `notready` state, possibly due to network failures or power outages.
Device Type Status	Unavailable	The disk in use may not be a raw disk, or it might be missing.

- Real-time alerts triggered on the **Alert** tab require prompt attention, even if the storage cluster status currently appears `Healthy`. Quick responses prevent escalation into more serious issues. The following table outlines alert levels and their implications:

Alert Level	Meaning
Critical	Indicates significant issues causing platform service interruptions or data loss, with severe impacts.
Major	Known issues potentially affecting platform functionality and normal business operations.
Warning	Risk of operational issues exists; timely intervention needed to avoid impact on normal business operations.

Post-Mortem Analysis

The **Alert History** logs all alerts triggered previously that no longer require immediate action. During post-mortem analysis, consider the following:

- What specific abnormalities were observed at the time of the incident?
- Are there patterns of specific alerts repeatedly occurring? How can these be proactively prevented in the future?
- Was there a surge in alerts during specific periods linked to external factors or operational incidents? Should operational strategies be adjusted accordingly?

Alauda Container Platform

How To

Backup and Restore TopoLVM Filesystem PVCs with Velero

Velero enables backup and restoration of Persistent Volume Claims (PVCs) and Persistent Volumes (PVs) for TopoLVM filesystems. This functionality is integrated into the platform.

This guide applies specifically to TopoLVM filesystem PVCs.

TOC

Prerequisites

1. Deploy the "Alauda Container Platform Data Backup for Velero" via the Marketplace/Cluster Plugins.
2. Configure an S3-compatible storage for Velero's `BackupStorageLocation`. Use platform-provided Ceph or MinIO object storage.

Limitations

1. The S3 storage must have sufficient free space to store all PV data from the target cluster.
2. During restoration, the namespace quota and storage class must support the total capacity of all PVCs.

Procedure

Step 1: Configure Backup Repository

1. Ensure an S3-compatible storage is available.
2. Navigate to **Administrator** > **Cluster Management** > **Backup and Restore** > **Backup Repository**.
3. Create a backup repository using the object storage credentials.

Step 2: Perform Backup

1. Label the PVCs and associated pods to be backed up:
   Velero needs a pod to restore a Filesystem PVC. The pod mounts the PVC for Velero to import data; without a pod, the PVC remains Pending. For complex apps, pause the application and attach the PVC to a lightweight pod (e.g., Nginx) for backup/restore, then restore the original app configuration post-restoration.

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

2. Go to **Backup and Restore** and create a new backup:
   - Select **Backup Kubernetes Resources and PVC Data Volumes**.
   - Choose the namespaces containing the data to back up.
   - Configure the backup with the following settings:

```yaml
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. After the backup completes, verify the data in the S3 bucket (e.g., MinIO):

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

Example output:

```
[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/
```

Step 3: Restore Cluster

1. In the target cluster, configure the same S3 bucket as used for the backup. Velero will automatically detect the existing backup.
2. Navigate to **Backup and Restore** and create a restore task:
   - Select the namespace(s) to restore.
   - In the advanced configuration, map the original namespace to the target namespace if needed.
3. Execute the restore operation.
4. After restoration, verify:
   - PVC names match the original cluster.
   - Application data in the PVCs is intact and accessible.

Configuring Striped Logical Volumes

When you write data to an LVM logical volume, the file system lays the data out across the underlying physical volumes. You can control the way the data is written to the physical volumes by creating a striped logical volume. For large sequential reads and writes, this can improve the efficiency of the data I/O.

Striping enhances performance by writing data to a predetermined number of physical volumes in round-round fashion. With striping, I/O can be done in parallel. In some situations, this can result in near-linear performance gain for each additional physical volume in the stripe.

TopoLVM achieves this by specifying the `lvcreate-option-class` in the `StorageClass`.

TOC

Prerequisites

- The device class must contain **at least two devices** on a single node.

Procedure

1 Using the Default `lvcreate-option-class`

1. Navigate to **Administrator**.
2. In the left sidebar, go to **Storage Management** > **Storage Classes**.
3. Click **Create Storage Class**.
4. Select **Block Storage**.
5. Choose **TopoLVM**, then click **Next**.
6. Configure the required storage class parameters.
7. Switch to **YAML view** and add the `topolvm.io/lvcreate-option-class` parameter:

```yaml
parameters:
  topolvm.io/lvcreate-option-class: striped-default
```

> **NOTE**
>
> `striped-default` is a built-in lvcreate-option-class in Alauda Container Platform. It automatically appends `--stripes=2` `--stripesize=64` to the lvcreate command when the TopoLVM CSI driver provisions logical volumes.

2 Creating a Custom lvcreate-option-class (Optional)

If the built-in striped-default class does not meet your needs, you can define a custom 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
```

This manifest creates a custom LVCreateOptionClass named striped-custom with 3 stripes and a stripe size of 64 KiB. Once applied, you can reference it in your StorageClass using `topolvm.io/lvcreate-option-class: striped-custom`.