Red Hat OpenShift Container Storage 4.8

Red Hat OpenShift Container Storage architecture

Architectural overview of Red Hat OpenShift Container Storage and its components and workflow.
Architectural overview of Red Hat OpenShift Container Storage and its components and workflow.
Abstract

This document provides an overview of the OpenShift Container Storage architecture.
# Table of Contents

- **MAKING OPEN SOURCE MORE INCLUSIVE** .................................................. 3
- **PROVIDING FEEDBACK ON RED HAT DOCUMENTATION** .......................... 4
- **CHAPTER 1. INTRODUCTION TO OPENSHIFT CONTAINER STORAGE** ............ 5
- **CHAPTER 2. AN OVERVIEW OF OPENSHIFT CONTAINER STORAGE ARCHITECTURE** ................................................................. 6
- **CHAPTER 3. OPENSHEET CONTAINER STORAGE OPERATORS** ..................... 7
  - **3.1. OPENSHEET CONTAINER STORAGE OPERATOR** .............................. 7
    - 3.1.1. Components ......................................................................................... 7
    - 3.1.2. Design diagram ................................................................................... 7
    - 3.1.3. Responsibilities ................................................................................... 8
    - 3.1.4. Resources ........................................................................................... 8
    - 3.1.5. Limitation ............................................................................................ 9
    - 3.1.6. High availability .................................................................................. 9
    - 3.1.7. Relevant config files ............................................................................ 9
    - 3.1.8. Relevant log files ................................................................................. 9
    - 3.1.9. Lifecycle .............................................................................................. 10
  - **3.2. ROOK-CEPH OPERATOR** ................................................................. 10
    - 3.2.1. Components ....................................................................................... 10
    - 3.2.2. Design diagram .................................................................................. 11
    - 3.2.3. Responsibilities ................................................................................... 11
    - 3.2.4. Resources ........................................................................................... 12
    - 3.2.5. Lifecycle .............................................................................................. 13
  - **3.3. NOOBAA OPERATOR** ....................................................................... 13
    - 3.3.1. Components ....................................................................................... 14
    - 3.3.2. Responsibilities and resources .............................................................. 14
    - 3.3.3. High availability .................................................................................. 15
    - 3.3.4. Relevant log files ................................................................................. 16
    - 3.3.5. Lifecycle .............................................................................................. 16
MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright’s message.
PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Do let us know how we can make it better. To give feedback:

- For simple comments on specific passages:
  1. Make sure you are viewing the documentation in the *Multi-page HTML* format. In addition, ensure you see the **Feedback** button in the upper right corner of the document.
  2. Use your mouse cursor to highlight the part of text that you want to comment on.
  3. Click the **Add Feedback** pop-up that appears below the highlighted text.
  4. Follow the displayed instructions.

- For submitting more complex feedback, create a Bugzilla ticket:
  1. Go to the **Bugzilla** website.
  2. As the Component, use **Documentation**.
  3. Fill in the **Description** field with your suggestion for improvement. Include a link to the relevant part(s) of documentation.
  4. Click **Submit Bug**.
CHAPTER 1. INTRODUCTION TO OPENSHIFT CONTAINER STORAGE

Red Hat OpenShift Container Storage is a highly integrated collection of cloud storage and data services for Red Hat OpenShift Container Platform. It is available as part of the Red Hat OpenShift Container Platform Service Catalog, packaged as an operator to facilitate simple deployment and management.

Red Hat OpenShift Container Storage services are primarily made available to applications by way of storage classes that represent the following components:

- **Block storage devices**, catering primarily to database workloads. Prime examples include Red Hat OpenShift Container Platform logging and monitoring, and PostgreSQL.

- **Shared and distributed file system**, catering primarily to software development, messaging, and data aggregation workloads. Examples include Jenkins build sources and artifacts, Wordpress uploaded content, Red Hat OpenShift Container Platform registry, and messaging using JBoss AMQ.

- **Multicloud object storage**, featuring a lightweight S3 API endpoint that can abstract the storage and retrieval of data from multiple cloud object stores.

- **On premises object storage**, featuring a robust S3 API endpoint that scales to tens of petabytes and billions of objects, primarily targeting data intensive applications. Examples include the storage and access of row, columnar, and semi-structured data with applications like Spark, Presto, Red Hat AMQ Streams (Kafka), and even machine learning frameworks like TensorFlow and Pytorch.

Red Hat OpenShift Container Storage version 4.x integrates a collection of software projects, including:

- **Ceph**, providing block storage, a shared and distributed file system, and on-premises object storage

- **Ceph CSI**, to manage provisioning and lifecycle of persistent volumes and claims

- **NooBaa**, providing a Multicloud Object Gateway

- **OpenShift Container Storage, Rook-Ceph, and NooBaa operators**, to initialize and manage OpenShift Container Storage services.
CHAPTER 2. AN OVERVIEW OF OPENSIGHT CONTAINER STORAGE ARCHITECTURE

Red Hat OpenShift Container Storage provides services for, and can run internally from Red Hat OpenShift Container Platform.

Red Hat OpenShift Container Storage architecture

Red Hat OpenShift Container Storage supports deployment into Red Hat OpenShift Container Platform clusters deployed on Installer Provisioned Infrastructure or User Provisioned Infrastructure. For details about these two approaches, see OpenShift Container Platform - Installation process. To know more about interoperability of components for the Red Hat OpenShift Container Storage and Red Hat OpenShift Container Platform, see the interoperability matrix.

For information about the architecture and lifecycle of OpenShift Container Platform, see OpenShift Container Platform architecture.
CHAPTER 3. OPENSIFT CONTAINER STORAGE OPERATORS

Red Hat OpenShift Container Storage is comprised of the following three operators, which codify administrative tasks and custom resources so that task and resource characteristics can be easily automated:

- **ocs-operator**
- **Rook-Ceph**
- **NooBaa**

Administrators define the desired end state of the cluster, and the OpenShift Container Storage operators ensure the cluster is either in that state or approaching that state, with minimal administrator intervention.

The deployments for all the three operators are defined in the **ocs-operator** bundle, which is managed by the Operator Lifecycle Manager (OLM).

### 3.1. OPENSIFT CONTAINER STORAGE OPERATOR

The **ocs-operator** is one of the three main operators of OpenShift Container Storage and can be described as a "meta" operator. It is an operator meant to influence other operators and serves as a configuration gateway for the features provided by the other operators. It does not directly manage the other operators.

The **ocs-operator** has the following primary functions:

- Creates Custom Resources (CRs) that trigger the other operators to reconcile against them.
- Abstracts the Ceph and Multicloud Object Gateway configurations and limits them to known best practices that are validated and supported by Red Hat.
- Creates and reconciles the resources required to deploy containerized Ceph and NooBaa according to the support policies.

### 3.1.1. Components

The **ocs-operator** does not have any dependent components. However, the operator has a dependency on the existence of all the custom resource definitions (CRDs) from other operators, which are defined in the **ClusterServiceVersion** (CSV).

### 3.1.2. Design diagram

This diagram illustrates how OpenShift Container Storage is integrated with the OpenShift Container Platform.

**OpenShift Container Storage Operator**
3.1.3. Responsibilities

The two `ocs-operator` CRDs are:

- **OCSInitialization**
- **StorageCluster**

**OCSInitialization** is a singleton CRD used for encapsulating operations that apply at the operator level. The operator takes care of ensuring that one instance always exists. The CR triggers the following:

- Performs initialization tasks required for OpenShift Container Storage. If needed, these tasks can be triggered to run again by deleting the `OCSInitialization` CRD.
  - Ensures that the required Security Context Constraints (SCCs) for OpenShift Container Storage are present.
  - Manages the deployment of the Ceph toolbox Pod, used for performing advanced troubleshooting and recovery operations.

The **StorageCluster** CRD represents the system that provides the full functionality of OpenShift Container Storage. It triggers the operator to ensure the generation and reconciliation of Rook-Ceph and NooBaa CRDs. The `ocs-operator` algorithmically generates the `CephCluster` and `NooBaa` CRDs based on the configuration in the `StorageCluster` spec. The operator also creates additional CRs, such as `CephBlockPools`, `Routes`, and so on. These resources are required for enabling different features of OpenShift Container Storage. Currently, only one StorageCluster CR per OpenShift Container Platform cluster is supported.

3.1.4. Resources

The `ocs-operator` creates the following CRs in response to the spec of the CRDs it defines. The configuration of some of these resources can be overridden, allowing for changes to the generated spec or not creating them altogether.

**General resources**

**Events**

- Creates various events when required in response to reconciliation.
Persistent Volumes (PVs)
PVs are not created directly by the operator. However, the operator keeps track of all the PVs created by the Ceph CSI drivers and ensures that the PVs have appropriate annotations for the supported features.

Quickstarts
Deploys various Quickstart CRs for the OpenShift Container Platform Console.

Rook-Ceph resources

CephBlockPool
Define the default Ceph block pools. CephFileSysPrometheusRulesRoute for the Ceph object store.

StorageClass
Define the default Storage classes. For example, for CephBlockPool and CephFilesystem).

VolumeSnapshotClass
Define the default volume snapshot classes for the corresponding storage classes.

Multicloud Object Gateway resources

NooBaa
Define the default Multicloud Object Gateway system.

Monitoring resources

- Metrics Exporter Service
- Metrics Exporter Service Monitor
- PrometheusRules

3.1.5. Limitation
The ocs-operator neither deploys nor reconciles the other Pods of OpenShift Container Storage. The ocs-operator CSV defines the top-level components such as operator Deployments and the Operator Lifecycle Manager (OLM) reconciles the specified component.

3.1.6. High availability
High availability is not a primary requirement for the ocs-operator Pod similar to most of the other operators. In general, there are no operations that require or benefit from process distribution. OpenShift Container Platform quickly spins up a replacement Pod whenever the current Pod becomes unavailable or is deleted.

3.1.7. Relevant config files
The ocs-operator configuration is entirely specified by the CSV and is not modifiable without a custom build of the CSV.

3.1.8. Relevant log files
To get an understanding of the OpenShift Container Storage and troubleshoot issues, you can look at the following:

- Operator Pod logs
- StorageCluster status and events
- OCSInitialization status

Operator Pod logs

Each operator provides standard Pod logs that include information about reconciliation and errors encountered. These logs often have information about successful reconciliation which can be filtered out and ignored.

StorageCluster status and events

The StorageCluster CR stores the reconciliation details in the status of the CR and has associated events. Status contains a section of the expected container images. It shows the container images that it expects to be present in the pods from other operators and the images that it currently detects. This helps to determine whether the OpenShift Container Storage upgrade is complete.

OCSInitialization status

This status shows whether the initialization tasks are completed successfully.

3.1.9. Lifecycle

The ocs-operator is required to be present as long as the OpenShift Container Storage remains installed. This is managed as part of OLM’s reconciliation of the OpenShift Container Storage CSV. At least one instance of the pod should be in Ready state.

The operator operands such as CRDs should not affect the lifecycle of the operator. An OCSInitialization CR should always exist. The operator creates one if it does not exist. The creation and deletion of StorageClusters is an operation outside the operator’s control and must be initiated by the administrator or automated with the appropriate API calls.

3.2. ROOK-CEPH OPERATOR

Rook-Ceph operator is the Rook operator for Ceph in the OpenShift Container Storage. Rook enables Ceph storage systems to run on the OpenShift Container Platform.

The Rook-Ceph operator is a simple container that automatically bootstraps the storage clusters and monitors the storage daemons to ensure the storage clusters are healthy.

3.2.1. Components

The Rook-Ceph operator manages a number of components as part of the OpenShift Container Storage deployment.

Ceph–CSI Driver

The operator creates and updates the CSI driver, including a provisioner for each of the two drivers, RADOS block device (RBD) and Ceph filesystem (CephFS) and a volume plugin daemonset for each of the two drivers.

Ceph daemons

Mons
The monitors (mons) provide the core metadata store for Ceph.

**OSDs**
The object storage daemons (OSDs) store the data on underlying devices.

**Mgr**
The manager (mgr) collects metrics and provides other internal functions for Ceph.

**RGW**
The RADOS Gateway (RGW) provides the S3 endpoint to the object store.

**MDS**
The metadata server (MDS) provides CephFS shared volumes.

### 3.2.2. Design diagram

The following image illustrates how Ceph Rook integrates with OpenShift Container Platform.

**Rook-Ceph Operator**

With Ceph running in the OpenShift Container Platform cluster, OpenShift Container Platform applications can mount block devices and filesystems managed by Rook-Ceph, or can use the S3/Swift API for object storage.

### 3.2.3. Responsibilities

The Rook-Ceph operator is a container that bootstraps and monitors the storage cluster. It performs the following functions:

- Automates the configuration of storage components
- Starts, monitors, and manages the Ceph monitor pods and Ceph OSD daemons to provide the RADOS storage cluster

- Initializes the pods and other artifacts to run the services to manage:
  - CRDs for pools
  - Object stores (S3/Swift)
  - Filesystems

- Monitors the Ceph mons and OSDs to ensure that the storage remains available and healthy

- Deploys and manages Ceph mons placement while adjusting the mon configuration based on cluster size

- Watches the desired state changes requested by the API service and applies the changes

- Initializes the Ceph-CSI drivers that are needed for consuming the storage

- Automatically configures the Ceph-CSI driver to mount the storage to pods

**Rook-Ceph Operator architecture**

![Rook-Ceph Operator architecture diagram]

The Rook-Ceph operator image includes all required tools to manage the cluster. There is no change to the data path. However, the operator does not expose all Ceph configurations. Many of the Ceph features like placement groups and crush maps are hidden from the users and are provided with a better user experience in terms of physical resources, pools, volumes, filesystems, and buckets.

### 3.2.4. Resources

Rook-Ceph operator adds owner references to all the resources it creates in the `openshift-storage` namespace. When the cluster is uninstalled, the owner references ensure that the resources are all cleaned up. This includes OpenShift Container Platform resources such as `configmaps`, `secrets`, `services`, `deployments`, `daemonsets`, and so on.

The Rook-Ceph operator watches CRs to configure the settings determined by OpenShift Container Storage, which includes `CephCluster`, `CephObjectStore`, `CephFilesystem`, and `CephBlockPool`. 
3.2.5. Lifecycle

Rook-Ceph operator manages the lifecycle of the following pods in the Ceph cluster:

**Rook operator**
A single pod that owns the reconcile of the cluster.

**RBD CSI Driver**
- Two provisioner pods, managed by a single deployment.
- One plugin pod per node, managed by a daemonset.

**CephFS CSI Driver**
- Two provisioner pods, managed by a single deployment.
- One plugin pod per node, managed by a daemonset.

**Monitors (mons)**
Three mon pods, each with its own deployment.

**Stretch clusters**
Contain five mon pods, one in the arbiter zone and two in each of the other two data zones.

**Manager (mgr)**
There is a single mgr pod for the cluster.

**Stretch clusters**
There are two mgr pods (starting with OpenShift Container Storage 4.8), one in each of the two non-arbiter zones.

**Object storage daemons (OSDs)**
At least three OSDs are created initially in the cluster. More OSDs are added when the cluster is expanded.

**Metadata server (MDS)**
The CephFS metadata server has a single pod.

**RADOS gateway (RGW)**
The Ceph RGW daemon has a single pod.

3.3. NOOBA OPERATOR

The NooBaa (also known as Multicloud Object Gateway) operator is an operator for OpenShift Container Storage along with the OpenShift Container Storage operator and the Rook-Ceph operator. The NooBaa operator is available upstream as a standalone operator.

The NooBaa operator performs the following primary functions:

- Controls and reconciles the Multicloud Object Gateway (MCG) component within OpenShift Container Storage.
- Manages new user resources such as object bucket claims, bucket classes, and backing stores.
- Creates the default out-of-the-box resources.
A few configurations and information are passed to the NooBaa operator through the OpenShift Container Storage operator.

### 3.3.1. Components

The NooBaa operator does not have sub-components. However, it consists of a reconcile loop for the different resources that are controlled by it.

The NooBaa operator has a command-line interface (CLI) and is available as a part of OpenShift Container Storage. It enables the creation, deletion, and querying of various resources. This CLI adds a layer of input sanitation and status validation before the configurations are applied unlike applying a YAML file directly.

### 3.3.2. Responsibilities and resources

The NooBaa operator reconciles and is responsible for the custom resource definitions (CRDs) and OpenShift Container Platform entities.

- Backing store
- Namespace store
- Bucket class
- Object bucket claims (OBCs)
- NooBaa, pod stateful sets CRD
- Prometheus Rules and Service Monitoring
- Horizontal pod autoscaler (HPA)

#### Backing store

A resource that the customer has connected to the MCG component. This resource provides MCG the ability to save the data of the provisioned buckets on top of it.

A default backing store is created as part of the deployment depending on the platform that the OpenShift Container Platform is running on. For example, when OpenShift Container Platform or OpenShift Container Storage is deployed on Amazon Web Services (AWS), it results in a default backing store which is an AWS::S3 bucket. Similarly, for Microsoft Azure, the default backing store is a blob container and so on.

The default backing stores are created using CRDs for the cloud credential operator, which comes with OpenShift Container Platform. There is no limit on the amount of the backing stores that can be added to MCG. The backing stores are used in the bucket class CRD to define the different policies of the bucket. Refer the documentation of the specific OpenShift Container Storage version to identify the types of services or resources supported as backing stores.

#### Namespace store

Resources that are used in namespace buckets. No default is created during deployment.

#### Bucketclass

A default or initial policy for a newly provisioned bucket. The following policies are set in a bucketclass:
Placement policy
Indicates the backing stores to be attached to the bucket and used to write the data of the bucket. This policy is used for data buckets and for cache policies to indicate the local cache placement. There are two modes of placement policy:

- Spread. Strips the data across the defined backing stores
- Mirror. Creates a full replica on each backing store

Namespace policy
A policy for the namespace buckets that defines the resources that are being used for aggregation and the resource used for the write target.

Cache Policy
This is a policy for the bucket and sets the hub (the source of truth) and the time to live (TTL) for the cache items.

A default bucket class is created during deployment and it is set with a placement policy that uses the default backing store. There is no limit to the number of bucket class that can be added.

Refer to the documentation of the specific openShift Container Storage version to identify the types of policies that are supported.

Object bucket claims (OBCs)
CRDs that enable provisioning of S3 buckets. With MCG, OBCs receive an optional bucket class to note the initial configuration of the bucket. If a bucket class is not provided, the default bucket class is used.

NooBaa, pod stateful sets CRD
An internal CRD that controls the different pods of the NooBaa deployment such as the DB pod, the core pod, and the endpoints. This CRD must not be changed as it is internal. This operator reconciles the following entities:

- DB pod SCC
- Role Binding and Service Account to allow SSO single sign-on between OpenShift Container Platform and NooBaa user interfaces
- Route for S3 access
- Certificates that are taken and signed by the OpenShift Container Platform and are set on the S3 route

Prometheus rules and service monitoring
These CRDs set up scraping points for Prometheus and alert rules that are supported by MCG.

Horizontal pod autoscaler (HPA)
It is Integrated with the MCG endpoints. The endpoint pods scale up and down according to CPU pressure (amount of S3 traffic).

3.3.3. High availability
As an operator, the only high availability provided is that the OpenShift Container Platform reschedules a failed pod.
3.3.4. Relevant log files

To troubleshoot issues with the NooBaa operator, you can look at the following:

- Operator pod logs, which are also available through the must-gather.
- Different CRDs or entities and their statuses that are available through the must-gather.

3.3.5. Lifecycle

The NooBaa operator runs and reconciles after OpenShift Container Storage is deployed and until it is uninstalled.