Red Hat Ceph Storage Ceph Object Gateway

Ceph Object Gateway, also known as RADOS Gateway (RGW), is an object storage interface built on top of the `librados` library to provide applications with a RESTful gateway to Ceph storage clusters. Use this information to understand how to deploy, configure, and administer a Ceph Object Gateway environment.

This uses a "Day Zero", "Day One", and "Day Two" organizational methodology, providing readers with a logical progression path.

Day Zero is where research and planning are done before implementing a potential solution.

Day One is where the actual deployment, and installation of the software happens.

Day Two is where all the basic, and advanced configuration happens.

Ceph Object Gateway supports three interfaces:

S3-compatibility

Provides object storage functionality with an interface that is compatible with a large subset of the Amazon S3 RESTful API.

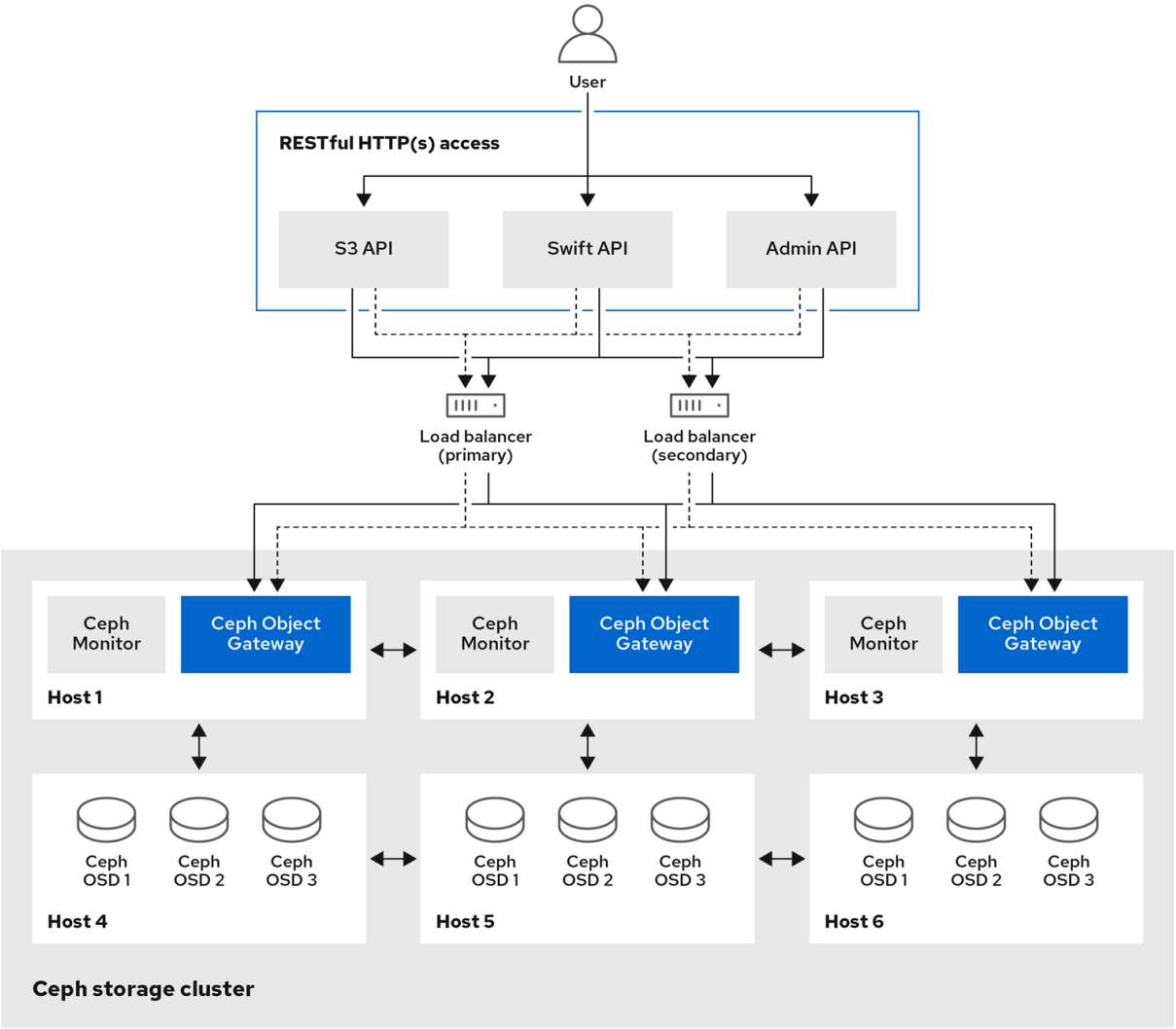
Swift-compatibility

Provides object storage functionality with an interface that is compatible with a large subset of the OpenStack Swift API.
The Ceph Object Gateway is a service interacting with a Ceph storage cluster. Since it provides interfaces compatible with OpenStack Swift and Amazon S3, the Ceph Object Gateway has its own user management system. Ceph Object Gateway can store data in the same Ceph storage cluster used to store data from Ceph Block Device clients; however, it would involve separate pools and likely a different CRUSH hierarchy. The S3 and Swift APIs share a common namespace, so you can write data with one API and retrieve it with the other.

Administrative API

Provides an administrative interface for managing the Ceph Object Gateways.
Administrative API requests are done on a URI that starts with the `admin` resource end point. Authorization for the administrative API mimics the S3 authorization convention. Some operations require the user to have special administrative capabilities. The response type can be either XML or JSON by specifying the format option in the request, but defaults to the JSON format.

Figure: Basic access diagram



Introduction to WORM

Write-Once-Read-Many (WORM) is a secured data storage model that is used to guarantee data protection and data retrieval even in cases where objects and buckets are compromised in production zones.

In Red Hat Ceph Storage, data security is achieved through the use of S3 Object Lock with read-only capability that is used to store objects and buckets using a Write-Once-Read-Many (WORM) model, preventing them from being deleted or overwritten. They cannot be deleted even by the Red Hat Ceph Storage administrator.

S3 Object Lock provides two retention modes:

- GOVERNANCE
- COMPLIANCE

These retention modes apply different levels of protection to your objects. You can apply either retention mode to any object version that is protected by Object Lock.

In GOVERNANCE mode, users cannot overwrite or delete an object version or alter its lock settings unless they have special permissions. With GOVERNANCE mode, you can protect objects against deletion by most users, although you can still grant some users permission to alter the retention settings or delete the object if necessary.

In COMPLIANCE mode, a protected object version cannot be overwritten or deleted by any user. When an object is locked in COMPLIANCE mode, its retention mode cannot be changed or shortened.

Important:
Consulting company Cohasset concludes that Red Hat Ceph Storage, when properly configured and upon satisfying the additional considerations, meets the electronic record-keeping system requirements of **SEC Rules 17a-4(f)(2)**, **18a-6(e)(2)**, and **FINRA Rule 4511(c)**, as well as, supports the regulated entity in its compliance with the audit system requirements in **SEC Rules 17a-4(f)(3)(iii)** and **18a-6(e)(3)(iii)**. In addition, the assessed capabilities meet the principles-based electronic records requirements of **CFTC Rule 1.31(c)-(d)**. See the Cohasset certification for more information.

Reference

For more details about S3 Object Lock, see the following:

- "Enabling object lock for S3" on page 149

Key Factors for deploying Ceph Object Gateway

As a storage administrator, a basic understanding about what to consider before running a Ceph Object Gateway and implementing a multi-site Ceph Object Gateway solution is important. You can learn the hardware and network requirements, knowing what type of workloads work well with a Ceph Object Gateway, and Red Hat's recommendations.

Prerequisites

- Time to understand, consider, and plan a storage solution.

Reference

For more details about Ceph's various internal components and the strategies around those components, see "Storage strategies" on page 0.

Network Requirements in Red Hat Ceph Storage

An important aspect of a cloud storage solution is that storage clusters can run out of IOPS due to network latency, and other factors. Also, the storage cluster can run out of throughput due to bandwidth constraints long before the storage clusters run out of storage capacity. This means that the network hardware configuration must support the chosen workloads to meet price versus performance requirements.

Storage administrators prefer that a storage cluster recovers as quickly as possible. Carefully consider bandwidth requirements for the storage cluster network, be mindful of network link oversubscription, and segregate the intra-cluster traffic from the client-to-cluster traffic. Also consider that network performance is increasingly important when considering the use of Solid State Disks (SSD), flash, NVMe, and other high performing storage devices.

Ceph supports a public network and a storage cluster network. The public network handles client traffic and communication with Ceph Monitors. The storage cluster network handles Ceph OSD heartbeats, replication, backfilling, and recovery traffic. At a **minimum**, a single 10 GB Ethernet link should be used for storage hardware, and you can add additional 10 GB Ethernet links for connectivity and throughput.

Important: Red Hat recommends allocating bandwidth to the storage cluster network, such that it is a multiple of the public network using the `osd_pool_default_size` as the basis for the multiple on replicated pools. Red Hat also recommends running the public and storage cluster networks on separate network cards.

Important:

A 1 GB Ethernet network is not suitable for production storage clusters.

In the case of a drive failure, replicating 1 TB of data across a 1 GB Ethernet network takes 3 hours, and 3 TB takes 9 hours. Using 3 TB is the typical drive configuration. By contrast, with a 10 GB Ethernet network, the replication times would be 20 minutes and 1 hour. Remember that when a Ceph OSD fails, the storage cluster recovers by replicating the data it contained to other OSDs within the same failure domain and device class as the failed OSD.

The failure of a larger domain such as a rack means that the storage cluster utilizes considerably more bandwidth. When building a storage cluster consisting of multiple racks, which is common for large storage implementations, consider utilizing as much network bandwidth between switches in a "fat tree" design for optimal performance. A typical 10 GB Ethernet switch has 48 10 GB ports and four 40 GB ports. Use the 40 GB ports on the spine for maximum throughput. Alternatively, consider aggregating unused 10 GB ports with QSFP+ and SFP+ cables into more 40 GB ports to connect to other rack and spine routers. Also, consider using LACP mode 4 to bond network interfaces. Additionally, use jumbo frames, with a maximum transmission unit (MTU) of 9000, especially on the backend or cluster network.

Most performance-related problems in Ceph usually begin with a networking issue. Simple network issues like a kinked or bent Cat-6 cable could result in degraded bandwidth. Use a minimum of 10 GB ethernet for the front side network. For large clusters, consider using 40 GB ethernet for the backend or cluster network.

For network optimization, Red Hat recommends using jumbo frames for a better CPU per bandwidth ratio, and a non-blocking network switch back-plane.Red Hat Ceph Storage requires the same MTU value throughout all networking devices in the communication path, end-to-end for both public and cluster networks. Verify that the MTU value is the same on all hosts and networking equipment in the environment before using a Red Hat Ceph Storage cluster in production.

Workload Management in Red Hat Ceph Storage

One of the key benefits of a Ceph storage cluster is the ability to support different types of workloads within the same storage cluster by using performance domains. Different hardware configurations can be associated with each performance domain. Storage administrators can deploy storage pools on the appropriate performance domain, providing applications with storage tailored to specific performance and cost profiles. Selecting appropriately sized and optimized servers for these performance domains is an essential aspect of designing a Red Hat Ceph Storage cluster.

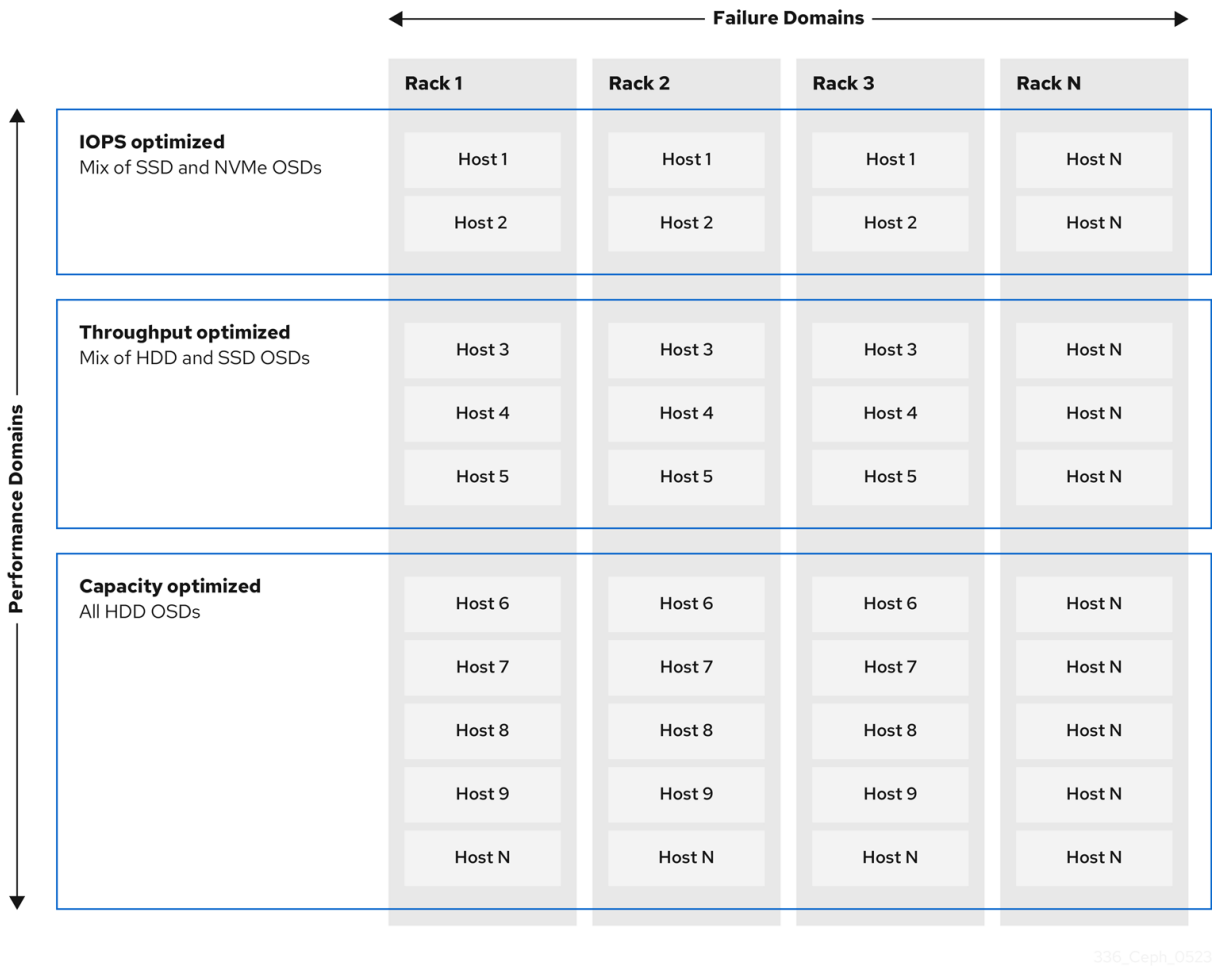
To the Ceph client interface that reads and writes data, a Ceph storage cluster appears as a simple pool where the client stores data. However, the storage cluster performs many complex operations in a manner that is visible to the client interface. Ceph clients and Ceph object storage daemons, referred to as *Ceph OSDs* or *OSDs*, both use the Controlled Replication Under Scalable Hashing (CRUSH) algorithm for the storage and retrieval of objects. Ceph OSDs can run in containers within the storage cluster.

A CRUSH map describes a topography of cluster resources, and the map exists both on client hosts and Ceph Monitor hosts within the cluster. Ceph clients and Ceph OSDs both use the CRUSH map and the CRUSH algorithm. Ceph clients communicate directly with OSDs, eliminating a centralized object lookup and a potential performance bottleneck. With awareness of the CRUSH map and communication with their peers, OSDs can handle replication, backfilling, and recovery, allowing for dynamic failure recovery.

Ceph uses the CRUSH map to implement failure domains. Ceph also uses the CRUSH map to implement performance domains, which takes the performance profile of the underlying hardware into consideration. The CRUSH map describes how Ceph stores data, and it is implemented as a simple hierarchy, specifically an acyclic graph, and a ruleset. The CRUSH map can support multiple hierarchies to separate one type of hardware performance profile from another. Ceph implements performance domains with device "classes".

- Hard disk drives (HDDs) are typically appropriate for cost and capacity-focused workloads.
- Throughput-sensitive workloads typically use HDDs with Ceph write journals on solid state drives (SSDs).
- IOPS-intensive workloads, such as MySQL and MariaDB, often use SSDs.

Figure: Performance and failure domains



Workloads

Red Hat Ceph Storage is optimized for three primary workloads.

> **Important:** Carefully consider the workload being run by Red Hat Ceph Storage clusters BEFORE considering what hardware to purchase because it can significantly impact the price and performance of the storage cluster. For example, if the workload is capacity-optimized and the hardware is better suited to a throughput-optimized workload, then hardware is more expensive than necessary. Conversely, if the workload is throughput-optimized and the hardware is better suited to a capacity-optimized workload, then the storage cluster can suffer from poor performance.

IOPS optimized

Input, output per second (IOPS) optimization deployments are suitable for cloud computing operations, such as running MYSQL or MariaDB instances as virtual machines on OpenStack. IOPS optimized deployments require higher performance storage such as 15k RPM SAS drives and separate SSD journals to handle frequent write operations. Some high IOPS scenarios use all flash storage to improve IOPS and total throughput.
An IOPS-optimized storage cluster has the following properties:

- Lowest cost per IOPS.
- Highest IOPS per GB.
- 99th percentile latency consistency.

Uses for an IOPS-optimized storage cluster are:

- Typically block storage.

- 3x replication for hard disk drives (HDDs) or 2x replication for solid state drives (SSDs).
- MySQL on OpenStack clouds.

Throughput optimized

Throughput-optimized deployments are suitable for serving up significant amounts of data, such as graphic, audio, and video content. Throughput-optimized deployments require high bandwidth networking hardware, controllers, and hard disk drives with fast sequential read and write characteristics. If fast data access is a requirement, then use a throughput-optimized storage strategy. Also, if fast write performance is a requirement, using Solid State Disks (SSD) for journals substantially improves write performance.
A throughput-optimized storage cluster has the following properties:

- Lowest cost per MBps (throughput).
- Highest MBps per TB.
- Highest MBps per BTU.
- Highest MBps per Watt.
- 97th percentile latency consistency.

Uses for a throughput-optimized storage cluster are as follows:

- Block or object storage.
- 3x replication.
- Active performance storage for video, audio, and images.
- Streaming media, such as 4k video.

Capacity optimized

Capacity-optimized deployments are suitable for storing significant amounts of data as inexpensively as possible. Capacity-optimized deployments typically trade performance for a more attractive price point. For example, capacity-optimized deployments often use slower and less expensive SATA drives and colocate journals rather than using SSDs for journaling.
A cost and capacity-optimized storage cluster has the following properties:

- Lowest cost per TB.
- Lowest BTU per TB.
- Lowest Watts required per TB.

Uses for a cost and capacity-optimized storage cluster are as follows:

- Typically object storage.
- Erasure coding for maximizing usable capacity.
- Object archive.
- Video, audio, and image object repositories.

Basic Considerations for Red Hat Ceph Storage Deployment

A storage strategy is a method of storing data that serves a particular use case. If you need to store volumes and images for a cloud platform like OpenStack, you can choose to store data on faster Serial Attached SCSI (SAS) drives with Solid State Drives (SSD) for journals. By contrast, if you need to store object data for an S3- or Swift-compliant gateway, you can choose to use something more economical, like traditional Serial Advanced Technology Attachment (SATA) drives.

One of the most important steps in a successful Ceph deployment is identifying a price-to-performance profile suitable for the storage cluster's use case and workload. It is important to choose the correct hardware for the use case. For example, choosing IOPS-optimized hardware for a cold storage application increases hardware costs unnecessarily. Whereas, choosing capacity-optimized hardware for its more attractive price point in an IOPS-intensive workload will likely lead to unhappy users complaining about slow performance.

Use cases, cost versus benefit performance tradeoffs, and data durability are the primary considerations that help develop a sound storage strategy.

Use cases

Ceph provides massive storage capacity, and it supports numerous use cases, such as:

- The Ceph Block Device client is a leading storage backend for cloud platforms that provides limitless storage for volumes and images with high performance features like copy-on-write cloning.
- The Ceph Object Gateway client is a leading storage backend for cloud platforms that provides a RESTful S3-compliant and Swift-compliant object storage for objects like audio, bitmap, video, and other data.
- The Ceph File System for traditional file storage.

Cost vs. benefit of performance

Faster is better. Bigger is better. High durability is better. However, there is a price for each superlative quality, and a corresponding cost versus benefit tradeoff. Consider the following use cases from a performance perspective: SSDs can provide very fast storage for relatively small amounts of data and journaling. Storing a database or object index can benefit from a pool of very fast SSDs, but proves too expensive for other data. SAS drives with SSD journaling provide fast performance at an economical price for volumes and images. SATA drives without SSD journaling provide cheap storage with lower overall performance. When you create a CRUSH hierarchy of OSDs, you need to consider the use case and an acceptable cost versus performance tradeoff.

Data durability

In large scale storage clusters, hardware failure is an expectation, not an exception. However, data loss and service interruption remain unacceptable. For this reason, data durability is very important. Ceph addresses data durability with multiple replica copies of an object or with erasure coding and multiple coding chunks. Multiple copies or multiple coding chunks present an additional cost versus benefit tradeoff: it is cheaper to store fewer copies or coding chunks, but it can lead to the inability to service write requests in a degraded state. Generally, one object with two additional copies, or two coding chunks can allow a storage cluster to service writes in a degraded state while the storage cluster recovers.

Replication stores one or more redundant copies of the data across failure domains in case of a hardware failure. However, redundant copies of data can become expensive at scale. For example, to store 1 petabyte of data with triple replication would require a cluster with at least 3 petabytes of storage capacity.

Erasure coding stores data as data chunks and coding chunks. In the event of a lost data chunk, erasure coding can recover the lost data chunk with the remaining data chunks and coding chunks. Erasure coding is substantially more economical than replication. For example, using erasure coding with 8 data chunks and 3 coding chunks provides the same redundancy as 3 copies of the data. However, such an encoding scheme uses approximately 1.5 times the initial data stored compared to 3 times with replication.

The CRUSH algorithm aids this process by ensuring that Ceph stores additional copies or coding chunks in different locations within the storage cluster. This ensures that the failure of a single storage device or host does not lead to a loss of all of the copies or coding chunks necessary to preclude data loss. You can plan a storage strategy with cost versus benefit tradeoffs, and data durability in mind, then present it to a Ceph client as a storage pool.

Important:

- ONLY the data storage pool can use erasure coding. Pools storing service data and bucket indexes use replication.
- Ceph’s object copies or coding chunks make RAID solutions obsolete. Do not use RAID, because Ceph already handles data durability, a degraded RAID has a negative impact on performance, and recovering data using RAID is substantially slower than using deep copies or erasure coding chunks.

Deployment Guidelines

An important aspect of designing a storage cluster is to determine if the storage cluster will be in one data center site or span multiple data center sites. Use this information to learn more about considerations for setting up Ceph Object Gateway.

Multi-site storage clusters benefit from geographically distributed failover and disaster recovery, such as long-term power outages, earthquakes, hurricanes, floods or other disasters. Additionally, multi-site storage clusters can have an active-active configuration, which can direct client applications to the closest available storage cluster. This is a good storage strategy for content delivery networks. Consider placing data as close to the client as possible. This is important for throughput-intensive workloads, such as streaming 4k video.

Important: Red Hat recommends identifying realm, zone group and zone names BEFORE creating Ceph's storage pools. Prepend some pool names with the zone name as a standard naming convention.

Reference

For more information, see "Multi-site configuration and administration" on page 71.

Administrative data storage

A Ceph Object Gateway stores administrative data in a series of pools that are defined in an instance's zone configuration. For example, the buckets, users, user quotas, and usage statistics that are discussed in the subsequent sections are stored in pools in the Ceph storage cluster.

By default, Ceph Object Gateway creates the following pools and maps them to the default zone.

- `.rgw.root`
- `.default.rgw.control`
- `.default.rgw.meta`
- `.default.rgw.log`
- `.default.rgw.buckets.index`
- `.default.rgw.buckets.data`
- `.default.rgw.buckets.non-ec`

Note: The `.default.rgw.buckets.index` pool is created only after the bucket is created in Ceph Object Gateway, while the `.default.rgw.buckets.data` pool is created after the data is uploaded to the bucket.

Consider creating these pools manually so you can set the CRUSH ruleset and the number of placement groups. In a typical configuration, the pools that store the Ceph Object Gateway's administrative data usually use the same CRUSH ruleset and fewer placement groups. This is because there are 10 pools for the administrative data.

Have the `.rgw.root` pool and the service pools use the same CRUSH hierarchy, and use at least `node` as the failure domain in the CRUSH rule. Use `replicated` for data durability, and NOT `erasure` for the `.rgw.root` pool, and the service pools.

The `mon_pg_warn_max_per_osd` setting warns you if you assign too many placement groups to a pool, `300` by default. You can adjust the value to suit your needs and the capabilities of your hardware where `n` is the maximum number of placement groups (PGs) per OSD.

```
mon_pg_warn_max_per_osd = n
```

Note: For service pools, including `.rgw.root`, the suggested PG count from the Ceph placement groups (PGs) per pool calculator is substantially less than the target PGs per Ceph OSD. Also, ensure the number of Ceph OSDs is set in step 4 of the calculator.

Important: Garbage collection uses the `.log` pool with regular RADOS objects instead of OMAP. Use NVMe/SSD Ceph OSDs for the `.log` pool.

`.rgw.root` pool

The pool where the Ceph Object Gateway configuration is stored. This includes realms, zone groups, and zones. Conventionally, the pool name does not include the zone name as a prefix.

Service pools

The service pools store objects that are related to service control, garbage collection, logging, user information, and usage. Conventionally, the pool name includes the zone name as a prefix, as described in "Service pools with zone names" on page 9.

Service pools with zone names

Zone name	Description
`.ZONE_NAME.rgw.control`	The control pool.
`.ZONE_NAME.log`	The log pool contains logs of all bucket, container, and object actions, such as create, read, update, and delete.
`.ZONE_NAME.rgw.buckets.index`	This pool stores index of the buckets.
`.ZONE_NAME.rgw.buckets.data`	This pool stores data of the buckets.
`.ZONE_NAME.rgw.meta`	The metadata pool stores user_keys and other critical metadata.
`.ZONE_NAME.rgw.meta:users.uid`	The user ID pool contains a map of unique user IDs.
`.ZONE_NAME.rgw.meta:users.keys`	The keys pool contains access keys and secret keys for each user ID.
`.ZONE_NAME.rgw.meta:users.email`	The email pool contains email addresses that are associated to a user ID.
`.ZONE_NAME.rgw.meta:users.swift`	The Swift pool contains the Swift subuser information for a user ID.

For more information, see "Pools" on page 75 and Storage strategies.

Index pool

When selecting OSD hardware for use with a Ceph Object Gateway--irrespective of the use case--an OSD node that has at least one high performance drive, either an SSD or NVMe drive, is required for storing the index pool. This is particularly important when buckets contain a large number of objects.

For Red Hat Ceph Storage running BlueStore, Red Hat recommends deploying an NVMe drive as a `block.db` device, rather than as a separate pool.

Ceph Object Gateway index data is written only into an object map (OMAP). OMAP data for BlueStore resides on the `block.db` device on an OSD. When an NVMe drive functions as a `block.db` device for an HDD OSD and when the index pool is backed by HDD OSDs, the index data will ONLY be written to the `block.db` device. As long as the `block.db` partition/lvm is sized properly at 4% of block, this configuration is all that is needed for BlueStore.

Note: Red Hat does not support HDD devices for index pools. For more information on supported configurations, see the *Red Hat Ceph Storage:Supported configurations* article.

An index entry is approximately 200 bytes of data, stored as an OMAP in `rocksdb`. While this is a trivial amount of data, some uses of Ceph Object Gateway can result in tens or hundreds of millions of objects in a single bucket. By mapping the index pool to a CRUSH hierarchy of high performance storage media, the reduced latency provides a dramatic performance improvement when buckets contain very large numbers of objects.

Important: In a production cluster, a typical OSD node will have at least one SSD or NVMe drive for storing the OSD journal and the index pool or `block.db` device, which use separate partitions or logical volumes for the same physical drive.

Data pool

The data pool is where the Ceph Object Gateway stores the object data for a particular storage policy. The data pool has a full complement of placement groups (PGs), not the reduced number of PGs for service pools. Consider using erasure coding for the data pool, as it is substantially more efficient than replication, and can significantly reduce the capacity requirements while maintaining data durability.

To use erasure coding, create an erasure code profile. See *Erasure code profiles*

Important: Choosing the correct profile is important because you cannot change the profile after you create the pool. To modify a profile, you must create a new pool with a different profile and migrate the objects from the old pool to the new pool.

The default configuration is two data chunks(**k**) and two encoding chunks(**m**), which means only one OSD can be lost. For higher resiliency, consider a larger number of data and encoding chunks. For example, some large scale systems use 8 data chunks and 3 encoding chunks, which allows 3 OSDs to fail without losing data.

Important: Each data and encoding chunk SHOULD get stored on a different node or host at a minimum. For smaller storage clusters, this makes using `rack` impractical as the minimum CRUSH failure domain for a larger number of data and encoding chunks. Consequently, it is common for the data pool to use a separate CRUSH hierarchy with `host` as the minimum CRUSH failure domain. Red Hat recommends `host` as the minimum failure domain. If erasure code chunks get stored on Ceph OSDs within the same host, a host failure, such as a failed journal or network card, could lead to data loss.

To create a data pool, run the `ceph osd pool create` command with the pool name, the number of PGs and PGPs, the `erasure` data durability method, the erasure code profile, and the name of the rule.

Data extra pool

The `data_extra_pool` is for data that cannot use erasure coding. For example, multi-part uploads allow uploading a large object, such as a movie in multiple parts. These parts must first be stored without erasure coding. Erasure coding applies to the whole object, not the partial uploads.

Note: The placement group (PG) per Pool Calculator recommends a smaller number of PGs per pool for the `data_extra_pool`; however, the PG count is approximately twice the number of PGs as the service pools and the same as the bucket index pool.

To create a data extra pool, run the `ceph osd pool create` command with the pool name, the number of PGs and PGPs, the `replicated` data durability method, and the name of the rule. For example:

```
# ceph osd pool create .us-west.rgw.buckets.non-ec 64 64 replicated rgw-service
```

Developing CRUSH hierarchies

As a storage administrator, when deploying a Ceph storage cluster and an object gateway, typically the Ceph Object Gateway has a default zone group and zone. The Ceph storage cluster will have default pools, which in turn will use a CRUSH map with a default CRUSH hierarchy and a default CRUSH rule.

Important: The default `rbd` pool can use the default CRUSH rule. **DO NOT** delete the default rule or hierarchy if Ceph clients have used them to store client data.

Production gateways typically use a custom realm, zone group and zone named according to the use and geographic location of the gateways. Additionally, the Ceph storage cluster will have a CRUSH map that has multiple CRUSH hierarchies.

- **Service Pools:** At least one CRUSH hierarchy will be for service pools and potentially for data. The service pools include `.rgw.root` and the service pools associated with the zone. Service pools typically fall under a single CRUSH hierarchy, and use replication for data durability. A data pool may also use the CRUSH hierarchy, but the pool will usually be configured with erasure coding for data durability.
- **Index:** At least one CRUSH hierarchy **SHOULD** be for the index pool, where the CRUSH hierarchy maps to high performance media, such as SSD or NVMe drives. Bucket indices can be a performance bottleneck. Red Hat recommends to use SSD or NVMe drives in this CRUSH hierarchy. Create partitions for indices on SSDs or NVMe drives used for Ceph OSD journals. Additionally, an index should be configured with bucket sharding.
- **Placement Pools:** The placement pools for each placement target include the bucket index, the data bucket, and the bucket extras. These pools can fall under separate CRUSH hierarchies. Since the Ceph Object Gateway can support multiple storage policies, the bucket pools of the storage policies may be associated with different CRUSH hierarchies, reflecting different use cases, such as IOPS-optimized, throughput-optimized, and capacity-optimized. The bucket index pool **SHOULD** use its own CRUSH hierarchy to map the bucket index pool to higher performance storage media, such as SSD or NVMe drives.

Creating CRUSH roots

From the command line on the administration node, create CRUSH roots in the CRUSH map for each CRUSH hierarchy. There **MUST** be at least one CRUSH hierarchy for service pools that may also potentially serve data storage pools. There **SHOULD** be at least one CRUSH hierarchy for the bucket index pool, mapped to high performance storage media, such as SSDs or NVMe drives.

In the following examples, the hosts named `data0`, `data1`, and `data2` use extended logical names, such as `data0-sas-ssd`, `data0-index`, and so forth in the CRUSH map, because there are multiple CRUSH hierarchies pointing to the same physical hosts.

A typical CRUSH root might represent nodes with SAS drives and SSDs for journals. For example:

```
##
# SAS-SSD ROOT DECLARATION
##

root sas-ssd {
  id -1   # do not change unnecessarily
  # weight 0.000
  alg straw
  hash 0  # rjenkins1
  item data2-sas-ssd weight 4.000
  item data1-sas-ssd weight 4.000
  item data0-sas-ssd weight 4.000
}
```

A CRUSH root for bucket indexes **SHOULD** represent high performance media, such as SSD or NVMe drives. Consider creating partitions on SSD or NVMe media that store OSD journals. For example:

```
##
# INDEX ROOT DECLARATION
##

root index {
  id -2    # do not change unnecessarily
  # weight 0.000
  alg straw
  hash 0  # rjenkins1
  item data2-index weight 1.000
  item data1-index weight 1.000
```

```
  item data0-index weight 1.000
}
```

Using logical host names in a CRUSH map

CRUSH supports the notion of a storage device "class". The clusters with hosts or nodes that contain multiple classes of a storage device, such as NVMe, SSD, or HDD, use a single CRUSH hierarchy with device classes to distinguish different classes of a storage device.

In the CRUSH map, host names must be unique and used only once. When the host serves multiple CRUSH hierarchies and use cases, a CRUSH map may use logical host names instead of the actual host name in order to ensure the host name is only used once. For example, a node may have multiple classes of drives such as SSDs, SAS drives with SSD journals, and SATA drives with co-located journals. For example, if the host name is `data2`, the CRUSH hierarchy might use logical names, such as `data2-sas-ssd` and `data2-index`:

```
host data2-sas-ssd {
  id -11   # do not change unnecessarily
  # weight 0.000
  alg straw
  hash 0  # rjenkins1
  item osd.0 weight 1.000
  item osd.1 weight 1.000
  item osd.2 weight 1.000
  item osd.3 weight 1.000
}
```

In the foregoing example, the host `data2` uses the logical name `data2-sas-ssd` to map the SAS drives with journals on SSDs into one hierarchy. The OSD IDs `osd.0` through `osd.3` represent SAS drives using SSD journals in a high throughput hardware configuration. These OSD IDs differ from the OSD ID in the following example.

In the following example, the host `data2` uses the logical name `data2-index` to map the SSD drive for a bucket index into a second hierarchy. The OSD ID `osd.4` represents an SSD drive or other high speed storage media used exclusively for a bucket index pool.

```
host data2-index {
  id -21   # do not change unnecessarily
  # weight 0.000
  alg straw
  hash 0  # rjenkins1
  item osd.4 weight 1.000
}
```

> **Important:** When using logical host names, ensure that one of the following settings is present in the Ceph configuration file to prevent the OSD startup scripts from using the actual host names upon startup and thereby, failing to locate data in the CRUSH map.

When the CRUSH map uses logical host names, as in the foregoing examples, prevent the OSD startup scripts from identifying the hosts according to their actual host names at initialization. In the `[global]` section of the Ceph configuration file, add the following setting:

```
osd_crush_update_on_start = false
```

An alternative method of defining a logical host name is to define the location of the CRUSH map for each OSD in the `[osd.<ID>]` sections of the Ceph configuration file. This will override any locations the OSD startup script defines. From the foregoing examples, the entries might look like the following:

```
[osd.0]
osd crush location = "host=data2-sas-ssd"

[osd.1]
osd crush location = "host=data2-sas-ssd"

[osd.2]
osd crush location = "host=data2-sas-ssd"
```

```
[osd.3]
osd crush location = "host=data2-sas-ssd"

[osd.4]
osd crush location = "host=data2-index"
```

Important: If one of the foregoing approaches is not taken when a CRUSH map uses logical host names rather than actual host names, on restart, the Ceph Storage Cluster assumes that the OSDs map to the actual host names, the actual host names are not found in the CRUSH map, and Ceph Storage Cluster clients will not find the OSDs and their data.

Creating CRUSH rules

Like the default CRUSH hierarchy, the CRUSH map also contains a default CRUSH rule. For each CRUSH hierarchy, create a CRUSH rule.

Note: The default `rbd` pool may use this rule. Do not delete the default rule if other pools have used it to store customer data.

- For general details on CRUSH rules, see “CRUSH rules” on page 0
- To manually edit a CRUSH map, see “Edit a CRUSH map” on page 0
- For more information about administrating, see “CRUSH admin overview” on page 0

The following example illustrates a rule for the CRUSH hierarchy that will store the service pools, including `.rgw.root`. In this example, the root `sas-ssd` serves as the main CRUSH hierarchy. It uses the name `rgw-service` to distinguish itself from the default rule. The `step take sas-ssd` line tells the pool to use the `sas-ssd` root created in “Creating CRUSH roots” on page 11 whose child buckets contain OSDs with SAS drives and high performance storage media, such as SSD or NVMe drives, for journals in a high throughput hardware configuration. The `type rack` portion of `step chooseleaf` is the failure domain. In this example, the failure domain is `rack`.

```
##
# SERVICE RULE DECLARATION
##

rule rgw-service {
 type replicated
 min_size 1
 max_size 10
 step take sas-ssd
 step chooseleaf firstn 0 type rack
 step emit
}
```

Note: In the following example, if data gets replicated three times, there should be at least three racks in the cluster containing a similar number of OSD nodes.

Tip: The `type replicated` setting has nothing to do with data durability, the number of replicas, or the erasure coding. Only `replicated` is supported.

The following example illustrates a rule for the CRUSH hierarchy that will store the data pool. In this example, the root `sas-ssd` serves as the main CRUSH hierarchy—the same CRUSH hierarchy as the service rule. It uses `rgw-throughput` to distinguish itself from the default rule and `rgw-service`. The `step take sas-ssd` line tells the pool to use the `sas-ssd` root created in “Creating CRUSH roots” on page 11, whose child buckets contain OSDs with SAS drives and high performance storage media, such as SSD or NVMe drives, in a high throughput

hardware configuration. The `type host` portion of `step chooseleaf` is the failure domain. In the following example, it is a host. Notice that the rule uses the same CRUSH hierarchy, but a different failure domain.

```
##
# THROUGHPUT RULE DECLARATION
##

rule rgw-throughput {
 type replicated
 min_size 1
 max_size 10
 step take sas-ssd
 step chooseleaf firstn 0 type host
 step emit
}
```

> **Note:** In the following example, if the pool uses erasure coding with a larger number of data and encoding chunks than the default, there should be at least as many racks in the cluster containing a similar number of OSD nodes to facilitate the erasure coding chunks. For smaller clusters, this may not be practical, so the foregoing example uses `host` as the CRUSH failure domain.

The following example illustrates a rule for the CRUSH hierarchy that stores the index pool. In this example, the root index serves as the main CRUSH hierarchy. It uses rgw-index to distinguish itself from rgw-service and rgw-throughput. The step take index line tells the pool to use the index root created in *Creating CRUSH roots*, whose child buckets contain high performance storage media, such as SSD or NVMe drives, or partitions on SSD or NVMe drives that also store OSD journals. The `type rack` portion of `step chooseleaf` is the failure domain. In the following example, it is a rack.

```
##
# INDEX RULE DECLARATION
##

rule rgw-index {
 type replicated
 min_size 1
 max_size 10
 step take index
 step chooseleaf firstn 0 type rack
 step emit
}
```

CRUSH Multi-Step Retry (MSR) rules

In Ceph, the **CRUSH** algorithm manages data placement across the cluster. However, conventional CRUSH rules have limitations when handling `out` OSDs (Object Storage Daemons) in multi-step configurations. Specifically, when an OSD is marked as `out` it is unavailable and conventional CRUSH rules cannot retry earlier steps in the rule, leading to potential issues such as uneven load distribution and undersized Placement Groups (PGs).

In standard CRUSH rules, if an OSD in a multi-step rule is marked as "out", subsequent steps may fail to find suitable OSDs for data placement. This can lead to:

Uneven load distribution

When an OSD is marked out on a particular host, other OSDs on the same host may become over-utilized, resulting in load imbalances.

Undersized placement groups (PGs)

If all OSDs on a host are marked out, the PGs mapped to that host may fail to find a valid location for data, causing them to be undersized or unable to map correctly.

To address these limitations, CRUSH MSR (Multi-Step Retry) is introduced. CRUSH MSR improves data remapping and load balancing by retrying each step of the CRUSH rule sequence whenever an `out` OSD is encountered. This ensures that data placement remains balanced, even when some OSDs are unavailable.

Consider an erasure-coded pool with an 8+6 coding scheme. The goal is to place 16 OSDs across four hosts, allowing the system to tolerate the loss of one entire host and one OSD on another. Without MSR, out OSDs on one host would result in PGs remapping to other OSDs on the same host, potentially causing load imbalance.

Example

```
rule ecpool-86 {
    type msr_indep
    step take default class hdd
    step choosemsr 4 type host
    step choosemsr 4 type osd
    step emit
}
```

In this example:

- `take defaultclass hdd` specifies the root bucket.
- `choosemsr 4 type host` selects 4 hosts, retrying previous steps if any host is unavailable.
- `choosemsr 4 type osd` selects 4 OSD from each chosen host, retrying as needed to ensure all hosts are utilized.
- `emit` finalizes the rule, mapping the PGs across chosen OSDs.

Before implementing CRUSH MSR, there are a few key considerations:

Compatibility

Before implementing CRUSH MSR ensure that your Red Hat Ceph Storage version supports CRUSH MSR. Older versions do not support this feature. Enable MSR for erasure-coded pools by restricting the cluster to support only Squid and later version clients.

```
ceph osd set-require-min-compat-client squid
```

Example for creating an MSR-based EC pool with a 4 node count

Create an MSR-based erasure coded pool for a 4 node count by creating a new erasure coding profile and then creating a pool with the new profile.

1. Create a new the erasure coding profile.
2. Create pool with the above profile.

The following procedure is an example to create a MSR-based Erase-Coded Pool for 4 Node Count.

1. Create an erasure code profile that defines the desired data (k) and parity (m) chunks, failure domain, and chunk distribution parameters.

   ```
   ceph osd erasure-code-profile set ec86-profile \
      k=8 \
      m=6 \
      crush-failure-domain=host \
      crush-osds-per-failure-domain=4 \
      crush-num-failure-domains=4
   ```

   The following lists the command examples and their descriptions:

   k=8,m=6

   Defines an 8+6 erasure coding scheme, with eight data chunks and six parity chunks.

   crush-failure-domain=host

   Sets each host as a failure domain.

   crush-osds-per-failure-domain=4

   Allows each host to store up to four chunks.

   crush-num-failure-domains=4

   Limits the mapping to four hosts, each with multiple OSDs, to meet the resilience requirements.

2. Verify that the profile and the CRUSH rule were created successfully in one of the following ways.
   - List and check the new profile created.

     ```
     ceph osd erasure-code-profile ls
     ceph osd erasure-code-profile get PROFILE_NAME
     ```

   - List and check the CRUSH rule created.

```
ceph osd crush rule ls
ceph osd crush rule dump RULE_NAME
```

3. Create the erasure-coded pool with the MSR rule. This command creates the ec86-pool by using the `ec86-profile` profile and the custom MSR rule `ecpool-86`.

```
ceph osd pool create ec86-pool erasure ec86-profile
```

4. Verify that the pool was created with the correct CRUSH rule profile.

```
ceph osd pool ls detail
```

Multi-site considerations

A Ceph Object Gateway multi-site configuration requires at least two Red Hat Ceph Storage clusters, and at least two Ceph Object Gateway instances, one for each Red Hat Ceph Storage cluster. Typically, the two Red Hat Ceph Storage clusters are in geographically separate locations; however, this same multi-site configuration can work on two Red Hat Ceph Storage clusters at the same physical site.

Multi-site configurations require a primary zone group and a primary zone. Additionally, each zone group requires a primary zone. Zone groups might have one or more secondary zones.

Configure multi-site either through the CLI or through the dashboard. See Configuring a multi-site for more details.

> **Important:** The primary zone within the primary zone group of a realm is responsible for storing the primary copy of the realm's metadata, including users, quotas, and buckets. This metadata gets synchronized to secondary zones and secondary zone groups automatically. Metadata operations issued with the **`radosgw-admin`** command line interface (CLI) *MUST* be issued on a node within the primary zone of the primary zone group to ensure that they synchronize to the secondary zone groups and zones. Currently, it is possible to issue metadata operations on secondary zones and zone groups, but it is *NOT* recommended because they *WILL NOT* be synchronized, which can lead to fragmentation of the metadata.

The following diagrams illustrate the possible one and two realm configurations in multi-site Ceph Object Gateway environments.

Figure: One realm configuration

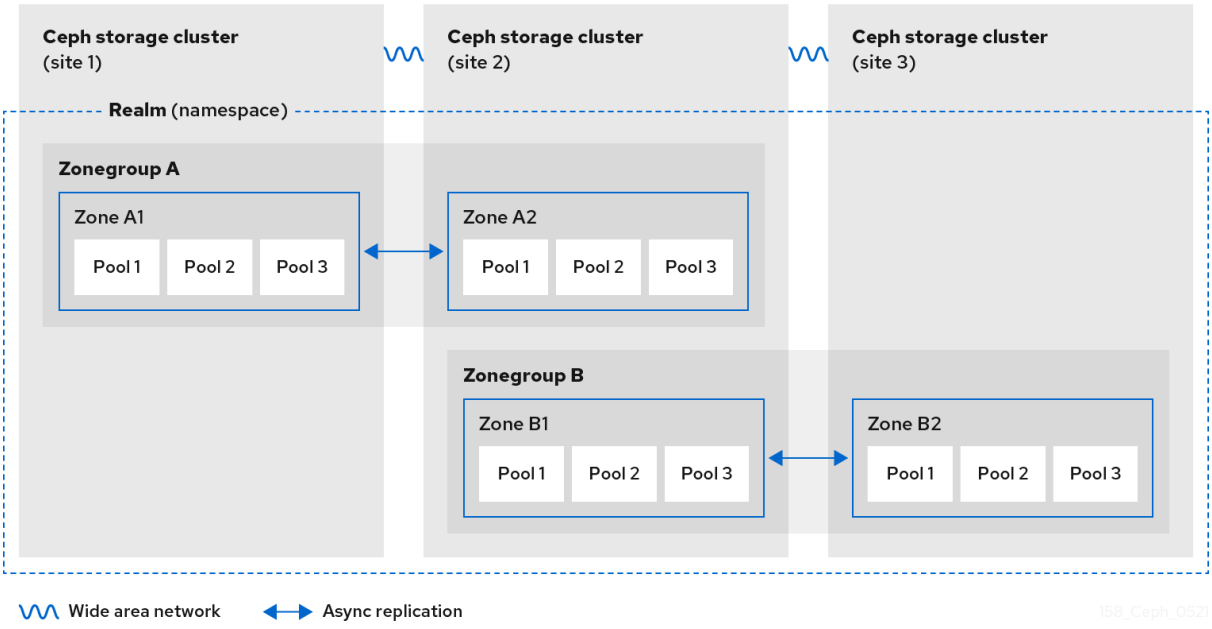


Figure: Two realm configuration

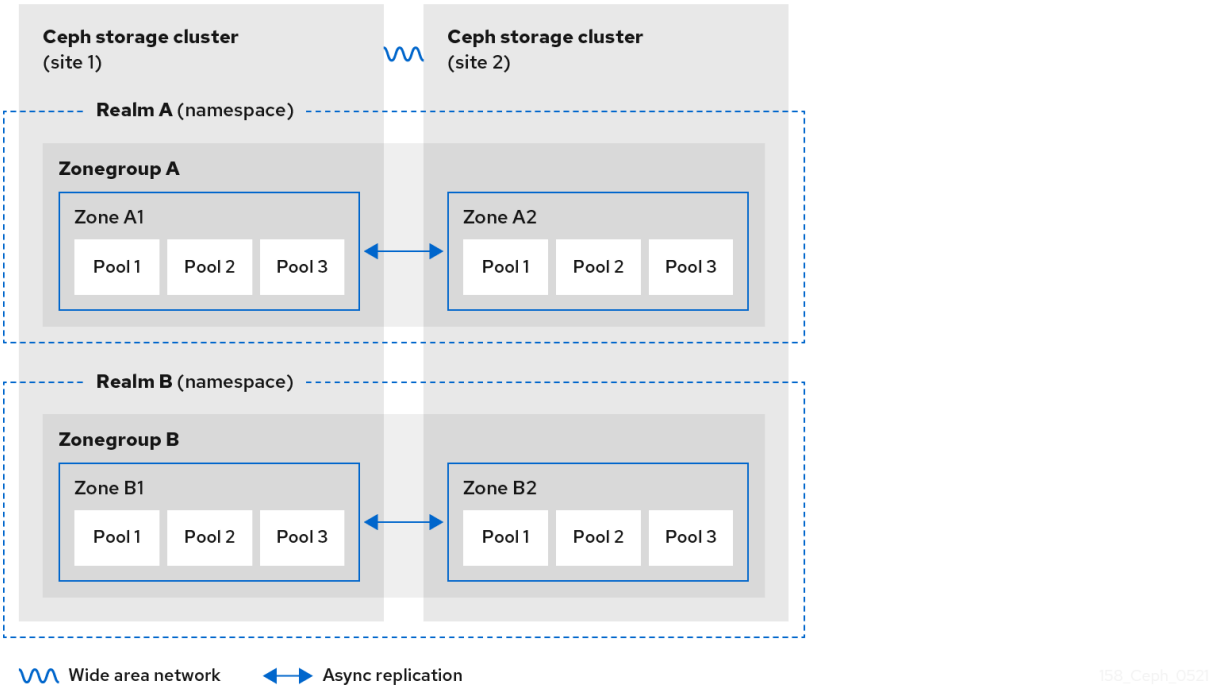
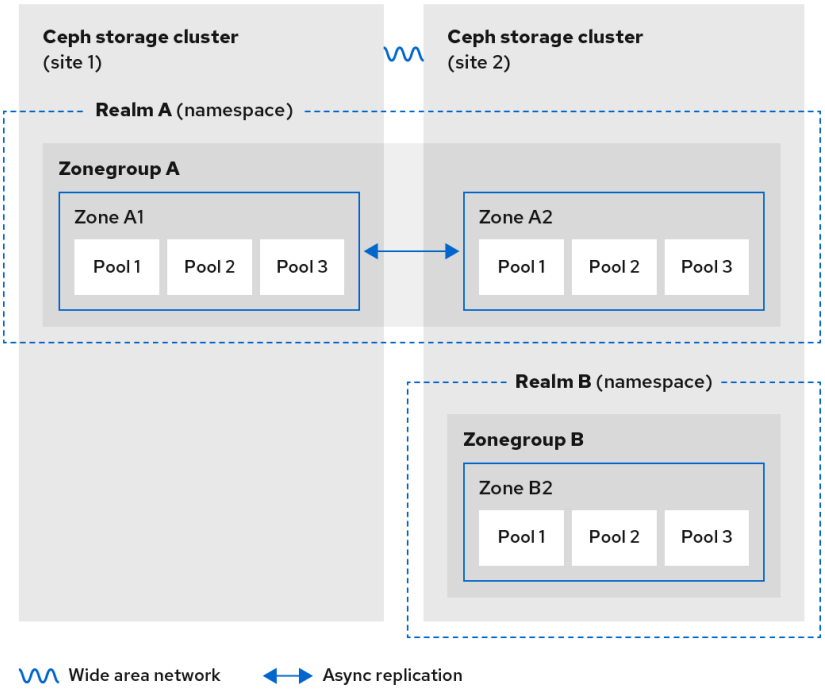


Figure: Two realms variant



Storage sizing

One of the most important factors in designing a cluster is to determine the storage requirements (sizing). Ceph Storage is designed to scale into petabytes and beyond.

The following examples are common sizes for Ceph storage clusters.

- **Small:** 250 terabytes
- **Medium:** 1 petabyte

- **Large:** 2 petabytes or more

Sizing includes current needs and near future needs. Consider the rate at which the gateway client will add new data to the cluster. That can differ from use-case to use-case. For example, recording 4k videos or storing medical images can add significant amounts of data faster than less storage-intensive information, such as financial market data. Additionally, consider that the data durability methods, such as replication versus erasure coding, can have a significant impact on the storage media required.

Reference

For more information, see the following:

- Hardware
-

Storage density

An important aspect of Ceph's design, includes storage density.

Generally, a storage cluster stores data across at least 10 nodes to ensure reasonable performance when replicating, backfilling, and recovery. If a node fails, with at least 10 nodes in the storage cluster, only 10% of the data has to move to the surviving nodes. If the number of nodes is substantially less, a higher percentage of the data must move to the surviving nodes. Additionally, the `full_ratio` and `near_full_ratio` options need to be set to accommodate a node failure to ensure that the storage cluster can write data. For this reason, it is important to consider storage density. Higher storage density is not necessarily a good idea.

Another factor that favors more nodes over higher storage density is erasure coding. When writing an object using erasure coding and using `node` as the minimum CRUSH failure domain, the Ceph storage cluster will need as many nodes as data and coding chunks. For example, a cluster using `k=8, m=3` should have at least 11 nodes so that each data or coding chunk is stored on a separate node.

Hot-swapping is also an important consideration. Most modern servers support drive hot-swapping. However, some hardware configurations require removing more than one drive to replace a drive. Red Hat recommends avoiding such configurations, because they can bring down more Ceph OSDs than required when swapping out failed disks.

Disks for the Ceph Monitor nodes

Ceph Monitors use `rocksdb`, which is sensitive to synchronous write latency. Red Hat strongly recommends using SSD disks to store the Ceph Monitor data. Choose SSD disks that have sufficient sequential write and throughput characteristics.

Performance adjustments

To optimize the performance of the Ceph Object Gateway, several adjustments can be made in the configuration.

Following are the key factors to consider for performance adjustments.

- Load balancing
- OSD configuration
- Network optimization

Adjusting backfill and recovery settings

I/O is negatively impacted by both backfilling and recovery operations, leading to poor performance and unhappy end users. To help accommodate I/O demand during a cluster expansion or recovery, adjust the backfill and recovery values in the Ceph configuration file.

Set the following options and values in the Ceph Configuration file:

```
[osd]
osd_max_backfills = 1
osd_recovery_max_active = 1
osd_recovery_op_priority = 1
```

Adjusting the cluster map size

Adjusting the cluster map size in the Ceph configuration can help reduce memory consumption, when needed.

By default, the `ceph-osd` daemon caches 500 previous osdmaps. Even with deduplication, the map might consume a lot of memory per daemon. Tuning the cache size in the Ceph configuration might help reduce memory consumption significantly. For example:

```
[ceph: root@host01 /]# ceph config set global osd_map_message_max 10
[ceph: root@host01 /]# ceph config set osd osd_map_cache_size 20
[ceph: root@host01 /]# ceph config set osd osd_map_share_max_epochs 10
[ceph: root@host01 /]# ceph config set osd osd_pg_epoch_persisted_max_stale 10
```

The `ceph-manager` daemon handles PG queries, so the cluster map should not impact performance.

Adjusting scrubbing

By default, Ceph performs light scrubbing daily and deep scrubbing weekly. Adjust the scrubbing schedules if the scrubbing schedule takes place at inopportune times.

Light scrubbing checks object sizes and checksums to ensure that PGs are storing the same object data. Over time, disk sectors can go bad irrespective of object sizes and checksums. Deep scrubbing checks an object's content with that of its replicas to ensure that the actual contents are the same. In this respect, deep scrubbing ensures data integrity in the manner of `fsck`, but the procedure imposes an I/O penalty on the cluster. Even light scrubbing can impact I/O.

The default settings may allow Ceph OSDs to initiate scrubbing at inopportune times, such as peak operating times or periods with heavy loads. End users may experience latency and poor performance when scrubbing operations conflict with end user operations.

To prevent end users from experiencing poor performance, Ceph provides a number of scrubbing settings that can limit scrubbing to periods with lower loads or during off-peak hours. For more information, see Scrubbing the OSD.

If the cluster experiences high loads during the day and low loads late at night, consider restricting scrubbing to night time hours. For example,

```
[osd]
osd_scrub_begin_hour = 23   #23:01H, or 10:01PM.
osd_scrub_end_hour = 6      #06:01H or 6:01AM.
```

If time constraints are not an effective method of determining a scrubbing schedule, consider using the **`osd_scrub_load_threshold`** command. The default value is `0.5`, but can be modified for low load conditions. For example,

```
[osd]
osd_scrub_load_threshold = 0.25
```

Increasing scalability

Improve scalability by editing Ceph Object Gateway configuration parameters.

For more information about configuring parameters, see "Configuration reference" on page 297.

Increase the thread pool

Change the size of the thread pool by editing the value of the **`rgw_thread_pool_size`** parameter.

> **Note:** The `beast` frontend is not restricted by the thread pool size to accept new connections.

For example,

```
rgw_thread_pool_size = 512
```

Increase the maximum number of allowed unsent I/O requests

Change the maximum number of allowed unsent I/O requests by editing the value of the **`--objecter_inflight_ops`** parameter. The **`--objecter_inflight_ops`** parameter is used for client traffic control.

For example,

```
objecter_inflight_ops = 24576
```

Tuning considerations for the Linux kernel when running Ceph

Tune the operating system, specifically for limits and memory allocation.

Production Red Hat Storage clusters generally benefit from tuning the operating system, specifically around limits and memory allocation. Ensure that adjustments are set for all hosts within the storage cluster. For more guidance, contact Red Hat Support.

Increasing the file descriptors

The Ceph Object Gateway can hang if it runs out of file descriptors. You can modify the `/etc/security/limits.conf` file on Ceph Object Gateway hosts to increase the file descriptors for the Ceph Object Gateway.

```
ceph       soft    nofile     unlimited
```

Adjusting the `ulimit` value for large storage clusters

Create a `/etc/security/limits.d/50-ceph.conf` file on each host that runs administrative commands, when running Ceph administrative commands on large storage clusters. An example of a large storage cluster is one with 1024 or more Ceph OSDs.

> **Note:** The root user's `ulimit` value is already set to `unlimited` by default on Red Hat Enterprise Linux.

Include the following contents to the configuration file:

```
USER_NAME       soft    nproc     unlimited
```

Replace *USER_NAME* with the name of the non-root user account that runs the Ceph administrative commands.

Advantages of colocating Ceph daemons

Colocate containerized Ceph daemons on the same host to optimize resource usage, reduce hardware costs, and improve performance.

Colocating Ceph daemons helps you achieve better efficiency and scalability. This topic explains the advantages, best practices, and examples for colocating daemons using YAML files or the command-line interface (CLI).

The following are advantages of colocating Red Hat Ceph Storage daemons:

- Significantly improves total cost of ownership (TCO) for small-scale deployments.
- Increases overall performance.

- Reduces the number of physical hosts in a minimum configuration.
- Optimizes resource usage.
- Simplifies Red Hat Ceph Storage upgrades.

For detailed information about daemon-specific colocation considerations, see Red Hat Ceph Storage: Supported configurations. Be sure to avoid single points of failure and to distribute CPU and memory demands evenly.

“Daemon placement example” on page 21 provides an example configuration.

Note:

- The only clusters with 3 Ceph Monitors should be those that only have 3 or 4 nodes total.
- The Ceph Monitor (`ceph-mon`) and Ceph Manager (`ceph-mgr`) work closely together and therefore are not considered two separate daemons for the purposes of colocation.

Daemon placement example

Host Name	Daemon	Daemon	Daemon
host01	OSD	Monitor and Manager	Prometheus
host02	OSD	Monitor and Manager	RGW
host03	OSD	Monitor and Manager	RGW
host04	OSD	Metadata Server, Monitor, and Manager	Prometheus or Monitoring Stack
host5	OSD	Metadata Server, Monitor, and Manager	Prometheus or Monitoring Stack

Colocate Ceph daemons, either by using a service specification YAML file or with the CLI.

Colocating Ceph daemons with a service specification YAML file

Colocate Ceph daemons by using a service specification YAML file, based on host labels. Using a service specification file is best practice as it avoids explicitly placing daemons on specific hostnames, increasing resilience during host changes. For more information, see “Labeling hosts” on page 0.
For example,

```
service_type: mon
placement:
  hosts:
   - host01
   - host02
   - host03
   - host04
   - host05

[ceph: root@host01 /]# ceph orch apply -i mon.yml
```

Colocating Ceph daemons with the CLI

Colocate Ceph daemons by using the CLI, by using the **`ceph orch`** command with the **`--placement`** option.
For example,

```
[ceph: root@host01 /]# ceph orch apply mon --placement="host01 host02 host03 host04
host05"
```

To increase performance, colocate the Ceph Object Gateway with Ceph OSD. To increase performance without incurring extra hardware costs, deploy two Ceph Object Gateway daemons per host.

“Figure: Colocated daemons” on page 21 and “Figure: Non-colocated daemons” on page 22 show the difference between storage clusters with colocated and non-colocated daemons.

Figure: Colocated daemons

Figure: Non-colocated daemons

For more information, see Managing services.

Related information

Red Hat Ceph Storage RGW deployment strategies and sizing guidance

Deployment

As a storage administrator, you can deploy the Ceph Object Gateway using the Ceph Orchestrator with the command line interface or the service specification. You can also configure multi-site Ceph Object Gateways, and remove the Ceph Object Gateway using the Ceph Orchestrator.

The `cephadm` command deploys the Ceph Object Gateway as a collection of daemons that manages a single-cluster deployment or a particular realm and zone in a multi-site deployment.

> **Note:** With `cephadm`, the Ceph Object Gateway daemons are configured using the Ceph Monitor configuration database instead of the `ceph.conf` file or the command line options. If the configuration is

not in the `client.rgw` section, then the Ceph Object Gateway daemons start up with default settings and bind to port `80`.

Warning: If you want Cephadm to handle the setting of a realm and zone, specify the realm and zone in the service specification during the deployment of the Ceph Object Gateway. If you want to change that realm or zone at a later point, ensure to update and reapply the `rgw_realm` and `rgw_zone` parameters in the specification file. If you want to handle these options manually without Cephadm, do not include them in the service specification. Cephadm still deploys the Ceph Object Gateway daemons without setting the configuration option for which realm or zone the daemons should use. In this case, the update of the specification file is not necessary.

Prerequisites

- A running, and healthy Red Hat Ceph Storage cluster.
- Root-level access to all the nodes.
- Available nodes on the storage cluster.
- All the managers, monitors, and OSDs are deployed in the storage cluster.

Deploying the Ceph Object Gateway using the command line interface

Deploy the Ceph Object Gateway by using the Ceph Orchestrator with the **ceph orch** command in the command line interface.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Root-level access to all the nodes.
- Hosts are added to the cluster.
- All manager, monitor, and OSD daemons are deployed.

Log in to the Cephadm shell by using the **cephadm shell** to deploy Ceph Object Gateway daemons. Deploy using one of the following methods:

- “Creating a realm, zone group, and zone” on page 24
- “Using an arbitrary service name for a single cluster deployment” on page 25
- “Using an arbitrary service name on a labeled set of hosts” on page 25

Creating a realm, zone group, and zone

Create a realm, zone group, zone, and then use the placement specification with the hostname.

1. Create a realm.

```
radosgw-admin realm create --rgw-realm=REALM_NAME --default
```

For example,

```
[ceph: root@host01 /]# radosgw-admin realm create --rgw-realm=test_realm --default
```

2. Create a zone group.

```
radosgw-admin zonegroup create --rgw-zonegroup=ZONE_GROUP_NAME  --master --default
```

For example,

```
[ceph: root@host01 /]# radosgw-admin zonegroup create --rgw-zonegroup=default  --
master --default
```

3. Create a zone.

```
radosgw-admin zone create --rgw-zonegroup=ZONE_GROUP_NAME --rgw-zone=ZONE_NAME --
master --default
```

For example,

```
[ceph: root@host01 /]# radosgw-admin zone create --rgw-zonegroup=default --rgw-
zone=test_zone --master --default
```

4. Commit the changes.

```
radosgw-admin period update --rgw-realm=REALM_NAME --commit
```

For example,

```
[ceph: root@host01 /]# radosgw-admin period update --rgw-realm=test_realm --commit
```

5. Apply the changes by using the **ceph orch apply** command.

```
ceph orch apply rgw NAME [--realm=REALM_NAME] [--zone=ZONE_NAME] [--
zonegroup=ZONE_GROUP_NAME] --placement="NUMBER_OF_DAEMONS [HOST_NAME_1 HOST_NAME_2]"
```

For example,

```
[ceph: root@host01 /]# ceph orch apply rgw test --realm=test_realm --zone=test_zone --
zonegroup=default --placement="2 host01 host02"
```

Using an arbitrary service name for a single cluster deployment

Use an arbitrary service name to deploy two Ceph Object Gateway daemons for a single cluster deployment.

```
ceph orch apply rgw SERVICE_NAME
```

For example,

```
[ceph: root@host01 /]# ceph orch apply rgw foo
```

Using an arbitrary service name on a labeled set of hosts

Use an arbitrary service name on a labeled set of hosts.

```
ceph orch host label add HOST_NAME_1 LABEL_NAME
ceph orch host label add HOSTNAME_2 LABEL_NAME
ceph orch apply rgw SERVICE_NAME --placement="label:LABEL_NAME count-per-
host:NUMBER_OF_DAEMONS" --port=8000
```

NUMBER_OF_DAEMONS controls the number of Ceph Object Gateways that are deployed on each host. To achieve the highest performance without incurring any extra cost, set this value to 2.
For example,

```
[ceph: root@host01 /]# ceph orch host label add host01 rgw  # the 'rgw' label can be
anything
[ceph: root@host01 /]# ceph orch host label add host02 rgw
[ceph: root@host01 /]# ceph orch apply rgw foo --placement="label:rgw count-per-host:2" --
port=8000
```

AFTER COMPLETING THE TASK

Verify the Ceph Object Gateway deployment.

- List the service, by using the **ceph orch ls** command.

For example,

```
[ceph: root@host01 /]# ceph orch ls
```

- List the hosts, daemons, and processes.

```
ceph orch ps --daemon_type=DAEMON_NAME
```

For example,

```
[ceph: root@host01 /]# ceph orch ps --daemon_type=rgw
```

Deploying the Ceph Object Gateway using the service specification

Deploy the Ceph Object Gateway using the service specification with either the default or the custom realms, zones, and zone groups.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the bootstrapped host.
- Hosts are added to the cluster.
- All manager, monitor, and OSD daemons are deployed.

Procedure

1. As a root user, create a specification file.

```
touch rgw_spec.yaml
```

For example,

```
[root@host01 ~]# touch rgw_spec.yaml
```

2. Configure S3 requests to wait for the duration defined in the **`rgw_exit_timeout_secs`** parameter for all outstanding requests to complete by setting **`rgw_graceful_stop`** to *true* during Ceph Object gateway shutdown/restart.

```
ceph config set client.rgw rgw_graceful_stop true

ceph config set client.rgw rgw_exit_timeout_secs 120
```

Note: In containerized deployments, an additional `extra_container_agrs` configuration of `--stop-timeout=120` (or the value of **`rgw_exit_timeout_secs`** configuration, if not default) is also necessary in order for it to work as expected with `ceph orch stop/restart` commands.
For example,

```
[root@host1 ~]$ cat rgw_spec.yaml
service_type: rgw
service_id: foo
placement:
  count_per_host: 1
  hosts:
    - rgw_node
spec:
  rgw_frontend_port: 8081
extra_container_args:
  - "--stop-timeout=120"
```

3. Edit the `rgw_spec.yaml` file to include the following details for the default realm, zone, and zone group.

```
service_type: rgw
service_id: REALM_NAME.ZONE_NAME
placement:
  hosts:
  - HOST_NAME_1
  - HOST_NAME_2
  count_per_host: NUMBER_OF_DAEMONS
spec:
  rgw_realm: REALM_NAME
  rgw_zone: ZONE_NAME
  rgw_zonegroup: ZONE_GROUP_NAME
  rgw_frontend_port: FRONT_END_PORT
networks:
  -  NETWORK_CIDR # Ceph Object Gateway service binds to a specific network
```

NUMBER_OF_DAEMONS controls the number of Ceph Object Gateways deployed on each host. To achieve the highest performance without incurring an additional cost, set this value to 2.

For example,

```
service_type: rgw
service_id: default
placement:
  hosts:
  - host01
  - host02
  - host03
  count_per_host: 2
spec:
  rgw_realm: default
  rgw_zone: default
  rgw_zonegroup: default
  rgw_frontend_port: 1234
networks:
  - 192.169.142.0/24
```

4. Optional: For custom realm, zone, and zone group, create the resources and then create the `rgw_spec.yaml` file.

   a. Create the custom realm, zone, and zone group.

      For example,

```
[root@host01 ~]# radosgw-admin realm create --rgw-realm=test_realm --default
[root@host01 ~]# radosgw-admin zonegroup create --rgw-zonegroup=test_zonegroup --
default
[root@host01 ~]# radosgw-admin zone create --rgw-zonegroup=test_zonegroup --rgw-
zone=test_zone --default
[root@host01 ~]# radosgw-admin period update --rgw-realm=test_realm --commit
```

   b. Create the `rgw_spec.yaml` file with the following details.

      For example,

```
service_type: rgw
service_id: test_realm.test_zone
placement:
  hosts:
  - host01
  - host02
  - host03
  count_per_host: 2
spec:
  rgw_realm: test_realm
  rgw_zone: test_zone
  rgw_zonegroup: test_zonegroup
  rgw_frontend_port: 1234
networks:
  - 192.169.142.0/24
```

5. Optional: Deploy HA Proxy concentrator for local RGW load balancing. To load balance requests across RGW daemons on the same host, configure the RGW service with a local HA Proxy concentrator.

> **Note:** HA Proxy concentrator does not currently support SSL.

> **Note:** Use this only when `count_per_host > 1`.

For example,

```
service_type: rgw
service_id: foo
service_name: rgw.foo
placement:
  count_per_host: 2
  hosts:
    - host1
    - host2
spec:
  concentrator: haproxy
  concentrator_frontend_port: 8080
  concentrator_monitor_port: 1967
  concentrator_monitor_user: admin
```

6. Mount the `rgw_spec.yaml` file under a directory in the container.

   For example,

   ```
   [root@host01 ~]# cephadm shell --mount rgw_spec.yaml:/var/lib/ceph/radosgw/
   rgw_spec.yaml
   ```

   > **Note:** Every time you exit the shell, you have to mount the file in the container before deploying the daemon.

7. Deploy the Ceph Object Gateway using the service specification.

   ```
   ceph orch apply -i FILE_NAME.yaml
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph orch apply -i /var/lib/ceph/radosgw/rgw_spec.yaml
   ```

> **Note:** In containerized deployments, an additional `extra_container_args` configuration of `--stop-timeout=120` (or the value of **`rgw_exit_timeout_secs`** configuration, if not default) is also necessary. For example,
>
> ```
> [root@host1 ~]$ cat rgw_spec.yaml
> service_type: rgw
> service_id: foo
> placement:
>   count_per_host: 1
>   hosts:
>     - rgw_node
> spec:
>   rgw_frontend_port: 8081
> extra_container_args:
>   - "--stop-timeout=120"
> ```

Verification

- List the service.

```
ceph orch ls
```

- List the hosts, daemons, and processes.

```
ceph orch ps --daemon_type=DAEMON_NAME
```

For example,

```
[ceph: root@host01 /]# ceph orch ps --daemon_type=rgw
```

Deploying a multi-site Ceph Object Gateway using the Ceph Orchestrator

Ceph Orchestrator supports multi-site configuration options for the Ceph Object Gateway.

You can configure each object gateway to work in an active-active zone configuration allowing writes to a non-primary zone. The multi-site configuration is stored within a container called a realm.

The realm stores zone groups, zones, and a time period. The `rgw` daemons handle the synchronization eliminating the need for a separate synchronization agent, thereby operating with an active-active configuration.

You can also deploy multi-site zones using the command line interface (CLI).

> **Note:** The following configuration assumes at least two Red Hat Ceph Storage clusters are in geographically separate locations. However, the configuration also works on the same site.

Prerequisites

- At least two running Red Hat Ceph Storage clusters.
- At least two Ceph Object Gateway instances, one for each Red Hat Ceph Storage cluster.
- Root-level access to all the nodes.
- Nodes or containers are added to the storage cluster.
- All Ceph Manager, Monitor and OSD daemons are deployed.

Procedure

1. In the `cephadm` shell, configure the primary zone:
   a. Create a realm:

      Syntax

      ```
      radosgw-admin realm create --rgw-realm=REALM_NAME --default
      ```

      Example

      ```
      [ceph: root@host01 /]# radosgw-admin realm create --rgw-realm=test_realm --
      default
      ```

      If the storage cluster has a single realm, then specify the `--default` flag.

   b. Create a primary zone group:

      Syntax

      ```
      radosgw-admin zonegroup create --rgw-zonegroup=_ZONE_GROUP_NAME_ --
      endpoints=http://RGW_PRIMARY_HOSTNAME:RGW_PRIMARY_PORT_NUMBER_1 --master --
      default
      ```

      Example

```
[ceph: root@host01 /]# radosgw-admin zonegroup create --rgw-zonegroup=us --
endpoints=http://rgw1:80 --master --default
```

c. Create a primary zone:

Syntax

```
radosgw-admin zone create --rgw-zonegroup=PRIMARY_ZONE_GROUP_NAME --rgw-
zone=PRIMARY_ZONE_NAME --endpoints=http://
RGW_PRIMARY_HOSTNAME:RGW_PRIMARY_PORT_NUMBER_1 --access-key=_SYSTEM_ACCESS_KEY_
--secret=_SYSTEM_SECRET_KEY_
```

Example

```
[ceph: root@host01 /]# radosgw-admin zone create --rgw-zonegroup=us --rgw-
zone=us-east-1 --endpoints=http://rgw1:80 --access-key=LIPEYZJLTWXRKXS9LPJC --
secret-key=IsAje0AVDNXNw48LjMAimpCpI7VaxJYSnfD0FFKQ
```

d. Optional: Delete the default zone, zone group, and the associated pools.

> **Important:**
> Do not delete the default zone and its pools if you are using the default zone and zone group to store data. Also, removing the default zone group deletes the system user.
>
> To access old data in the `default` zone and zonegroup, use `--rgw-zone default` and `--rgw-zonegroup default` in `radosgw-admin` commands.

Example

```
[ceph: root@host01 /]# radosgw-admin zonegroup delete --rgw-zonegroup=default
[ceph: root@host01 /]# ceph osd pool rm default.rgw.log default.rgw.log --yes-i-
really-really-mean-it
[ceph: root@host01 /]# ceph osd pool rm default.rgw.meta default.rgw.meta --yes-
i-really-really-mean-it
[ceph: root@host01 /]# ceph osd pool rm default.rgw.control default.rgw.control
--yes-i-really-really-mean-it
[ceph: root@host01 /]# ceph osd pool rm default.rgw.data.root
default.rgw.data.root --yes-i-really-really-mean-it
[ceph: root@host01 /]# ceph osd pool rm default.rgw.gc default.rgw.gc --yes-i-
really-really-mean-it
```

e. Create a system user:

Syntax

```
radosgw-admin user create --uid=USER_NAME --display-name="USER_NAME" --access-
key=SYSTEM_ACCESS_KEY --secret=SYSTEM_SECRET_KEY --system
```

Example

```
[ceph: root@host01 /]# radosgw-admin user create --uid=zone.user --display-
name="Zone user" --system
```

Make a note of the `access_key` and `secret_key`.

f. Add the access key and system key to the primary zone:

Syntax

```
radosgw-admin zone modify --rgw-zone=PRIMARY_ZONE_NAME --access-key=ACCESS_KEY --
secret=SECRET_KEY
```

Example

```
[ceph: root@host01 /]# radosgw-admin zone modify --rgw-zone=us-east-1 --access-
key=NE48APYCAODEPLKBCZVQ --secret=u24GHQWRE3yxxNBnFBzjM4jn14mFIckQ4EKL6LoW
```

g. Commit the changes:

Syntax

```
radosgw-admin period update --commit
```

Example

```
[ceph: root@host01 /]# radosgw-admin period update --commit
```

h. Outside the `cephadm` shell, fetch the `FSID` of the storage cluster and the processes:

Example

```
[root@host01 ~]#  systemctl list-units | grep ceph
```

i. Start the Ceph Object Gateway daemon:

Syntax

```
systemctl start ceph-FSID@DAEMON_NAME
systemctl enable ceph-FSID@DAEMON_NAME
```

Example

```
[root@host01 ~]# systemctl start ceph-62a081a6-88aa-11eb-
a367-001a4a000672@rgw.test_realm.us-east-1.host01.ahdtsw.service
[root@host01 ~]# systemctl enable ceph-62a081a6-88aa-11eb-
a367-001a4a000672@rgw.test_realm.us-east-1.host01.ahdtsw.service
```

2. In the Cephadm shell, configure the secondary zone.

a. Pull the primary realm configuration from the host:

Syntax

```
radosgw-admin realm pull --rgw-realm=PRIMARY_REALM --
url=URL_TO_PRIMARY_ZONE_GATEWAY --access-key=ACCESS_KEY --secret-key=SECRET_KEY
--default
```

Example

```
[ceph: root@host04 /]# radosgw-admin realm pull --rgw-realm=test_realm --
url=http://10.74.249.26:80 --access-key=LIPEYZJLTWXRKXS9LPJC --secret-
key=IsAje0AVDNXNw48LjMAimpCpI7VaxJYSnfD0FFKQ --default
```

b. Pull the primary period configuration from the host:

Syntax

```
radosgw-admin period pull --url=URL_TO_PRIMARY_ZONE_GATEWAY --access-
key=ACCESS_KEY --secret-key=SECRET_KEY
```

Example

```
[ceph: root@host04 /]# radosgw-admin period pull --url=http://10.74.249.26:80 --
access-key=LIPEYZJLTWXRKXS9LPJC --secret-
key=IsAje0AVDNXNw48LjMAimpCpI7VaxJYSnfD0FFKQ
```

c. Configure a secondary zone:

Syntax

```
radosgw-admin zone create --rgw-zonegroup=ZONE_GROUP_NAME
             --rgw-zone=SECONDARY_ZONE_NAME
             --access-key=SYSTEM_ACCESS_KEY --secret=SYSTEM_SECRET_KEY
```

```
            --endpoints=http://FQDN:80
            [--read-only]
```

Example

```
[ceph: root@host04 /]# radosgw-admin zone create --rgw-zonegroup=us --rgw-
zone=us-east-2 --endpoints=http://rgw2:80 --access-key=LIPEYZJLTWXRKXS9LPJC --
secret-key=IsAje0AVDNXNw48LjMAimpCpI7VaxJYSnfD0FFKQ
```

d. Optional: Delete the default zone.

> **Important:** Do not delete the default zone and its pools if you are using the default zone and zone group to store data. To access old data in the `default` zone and zonegroup, use `--rgw-zone default` and `--rgw-zonegroup default` in `radosgw-admin` commands.

Example

```
[ceph: root@host04 /]# radosgw-admin zone rm --rgw-zone=default
[ceph: root@host04 /]# ceph osd pool rm default.rgw.log default.rgw.log --yes-i-
really-really-mean-it
[ceph: root@host04 /]# ceph osd pool rm default.rgw.meta default.rgw.meta --yes-
i-really-really-mean-it
[ceph: root@host04 /]# ceph osd pool rm default.rgw.control default.rgw.control
--yes-i-really-really-mean-it
[ceph: root@host04 /]# ceph osd pool rm default.rgw.data.root
default.rgw.data.root --yes-i-really-really-mean-it
[ceph: root@host04 /]# ceph osd pool rm default.rgw.gc default.rgw.gc --yes-i-
really-really-mean-it
```

e. Update the Ceph configuration database:

Syntax

```
ceph config set SERVICE_NAME rgw_zone SECONDARY_ZONE_NAME
```

Example

```
[ceph: root@host04 /]# ceph config set rgw rgw_zone us-east-2
```

f. Commit the changes:

Syntax

```
radosgw-admin period update --commit
```

Example

```
[ceph: root@host04 /]# radosgw-admin period update --commit
```

g. Outside the Cephadm shell, fetch the FSID of the storage cluster and the processes:

Example

```
[root@host04 ~]#  systemctl list-units | grep ceph
```

h. Start the Ceph Object Gateway daemon:

Syntax

```
systemctl start ceph-FSID@DAEMON_NAME
systemctl enable ceph-FSID@DAEMON_NAME
```

Example

```
[root@host04 ~]# systemctl start ceph-62a081a6-88aa-11eb-
a367-001a4a000672@rgw.test_realm.us-east-2.host04.ahdtsw.service
[root@host04 ~]# systemctl enable ceph-62a081a6-88aa-11eb-
a367-001a4a000672@rgw.test_realm.us-east-2.host04.ahdtsw.service
```

3. Optional: Deploy multi-site Ceph Object Gateways using the placement specification:

   Syntax

   ```
   ceph orch apply rgw NAME --realm=REALM_NAME --zone=PRIMARY_ZONE_NAME --
   placement="NUMBER_OF_DAEMONS HOST_NAME_1 HOST_NAME_2"
   ```

   Example

   ```
   [ceph: root@host04 /]# ceph orch apply rgw east --realm=test_realm --zone=us-east-1 --
   placement="2 host01 host02"
   ```

Verification

- Check the synchronization status to verify the deployment:

  Example

  ```
  [ceph: root@host04 /]# radosgw-admin sync status
  ```

Removing the Ceph Object Gateway using the Ceph Orchestrator

Remove the Ceph Object Gateway daemons using the **`ceph orch rm`** command.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to all the nodes.
- Hosts are added to the cluster.
- At least one Ceph Object Gateway daemon deployed on the hosts.

Procedure

1. Log into the Cephadm shell:

   Example

   ```
   [root@host01 ~]# cephadm shell
   ```

2. List the service:

   Example

   ```
   [ceph: root@host01 /]# ceph orch ls
   ```

3. Remove the service:

   Syntax

   ```
   ceph orch rm SERVICE_NAME
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph orch rm rgw.test_realm.test_zone_bb
   ```

Verification

- List the hosts, daemons, and processes:

  Syntax

  ```
  ceph orch ps
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph orch ps
  ```

Reference

For more information, see the following:

- Deploying the Ceph Object Gateway using the command line interface
- Deploying the Ceph Object Gateway using the service specification

Using the Ceph Manager `rgw` module

As a storage administrator, you can deploy Ceph Object Gateway, single site and multi-site, using the `rgw` module. It helps with bootstrapping and configuring Ceph Object realm, zonegroup, and the different related entities.

You can use the available tokens for the newly created or existing realms. This token is a base64 string that encapsulates the realm information and its master zone endpoint authentication data.

In a multi-site configuration, these tokens can be used to pull a realm to create a secondary zone on a different cluster that syncs with the master zone on the primary cluster by using the `rgw zone create` command.

Deploying Ceph Object Gateway using the `rgw` module

Bootstrapping the Ceph Object Gateway realm creates a new realm entity, a new zonegroup, and a new zone. The `rgw` module instructs the orchestrator to create and deploy the corresponding Ceph Object Gateway daemons.

PREREQUISITE

- A running Red Hat Ceph Storage cluster.

Enable the `rgw` module by using the `ceph mgr module enable rgw` command. Pass the arguments in the CLI or use the YAML specification file to bootstrap the realm.

1. Log in to the Cephadm shell.

   ```
   cephadm shell
   ```

   For example,

   ```
   [root@host01 ~]# cephadm shell
   ```

2. Enable the `rgw` module.

   ```
   ceph mgr module enable rgw
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph mgr module enable rgw
   ```

3. Bootstrap the Ceph Object Gateway realm by using either the CLI or the YAML specification file.
   - Use the command-line interface.

```
ceph rgw realm bootstrap [--realm name REALM_NAME] [--zonegroup-
name ZONEGROUP_NAME] [--zone-name ZONE_NAME] [--port PORT_NUMBER] [--
placement HOSTNAME] [--start-radosgw]
```

Foe example,

```
[ceph: root@host01 /]# ceph rgw realm bootstrap --realm-name myrealm --zonegroup-
name myzonegroup --zone-name myzone --port 5500 --placement="host01 host02" --
start-radosgw
Realm(s) created correctly. Please, use 'ceph rgw realm tokens' to get the token.
```

- Use YAML specification file.

  a. As a root user, create the YAML file:

  ```
  rgw_realm: REALM_NAME
  rgw_zonegroup: ZONEGROUP_NAME
  rgw_zone: ZONE_NAME
  placement:
    hosts:
      - HOSTNAME_1
      - HOSTNAME_2
  ```

  For example,

  ```
  [root@host01 ~]# cat rgw.yaml

  rgw_realm: myrealm
  rgw_zonegroup: myzonegroup
  rgw_zone: myzone
  placement:
    hosts:
      - host01
      - host02
  ```

  b. You can add *hostnames* to the zonegroup during realm bootstrap:

  ```
  service_type: rgw
  placement:
    hosts:
    - host1
    - host2
  spec:
    rgw_realm: my_realm
    rgw_zonegroup: my_zonegroup
    rgw_zone: my_zone
    zonegroup_hostnames:
    - hostname1
    - hostname2
  ```

  For example,

  ```
  service_type: rgw
  placement:
    hosts:
    - _host1_
    - _host2_
  spec:
    rgw_realm: my_realm
    rgw_zonegroup: my_zonegroup
    rgw_zone: my_zone
    zonegroup_hostnames:
    - foo
    - bar
  ```

  c. Mount the YAML file under a directory in the container.

  ```
  cephadm shell --mount rgw.yaml:/var/lib/ceph/rgw/rgw.yaml
  ```

  For example,

  ```
  [root@host01 ~]# cephadm shell --mount rgw.yaml:/var/lib/ceph/rgw/rgw.yaml
  ```

d. Bootstrap the realm.

> **Note:** The specification file that is used by the `rgw` module has the same format as the one used by the orchestrator. Therefore, you can provide any orchestration supported Ceph Object Gateway parameters, including advanced configuration features such as SSL certificates.

```
ceph rgw realm bootstrap -i /var/lib/ceph/rgw/rgw.yaml
```

For example,

```
[ceph: root@host01 /]# ceph rgw realm bootstrap -i /var/lib/ceph/rgw/
rgw.yaml
```

4. List the available tokens.

> **Note:** If you run this command before the Ceph Object Gateway daemons are fully deployed, a 'no tokens' message is displayed. This is because of no endpoints existing yet.

```
ceph rgw realm tokens | jq
```

For example,

```
[ceph: root@host01 /]# ceph rgw realm tokens | jq

[
  {
    "realm": "myrealm",
    "token":
"ewogICAgInJlYWxtX25hbWUiOiAibXlyZWFsbSIsCiAgICAicmVhbG1faWQiOiAiZDA3YzAwZWYtOTA0MS00Z
jZlLTg4MDQtN2Q0MDI0MDU1NmFlIiwKICAgICJlbmRwb2ludCI6ICJodHRwOi8vdm0tMDA6NDMyMSIsCiAgICA
iYWNjZXNzX2tleSI6ICI5NTY1VFZSMVFWTExFRzdVNFIxRCIsCiAgICAic2VjcmV0IjogImQ3b0FJQXZrNEdYe
Xpyd3Q2QVZ6bEZNQmNnRG53RVdMMHFDenE3cjUiCn1="
  }
]
```

AFTER COMPLETING THE TASK

- Verify the Ceph Object Gateway deployment.
  For example,

```
[ceph: root@host01 /]# ceph orch list --daemon-type=rgw
NAME                                                  HOST                    PORTS
 STATUS          REFRESHED  AGE  MEM USE  MEM LIM  VERSION          IMAGE ID
 CONTAINER ID
rgw.myrealm.myzonegroup.ceph-saya-6-osd-host01.eburst  ceph-saya-6-osd-host01  *:80
running (111m)  9m ago     111m  82.3M   -        17.2.6-22.el9cp  2d5b080de0b0
 2f3eaca7e88e
```

- Verify the *hostnames* added via realm bootstrap:
  For example,

```
radosgw-admin zonegroup get --rgw-zonegroup zone_group_name
```

Foe example,

```
[ceph: root@host01 /]# radosgw-admin zonegroup get --rgw-zonegroup my_zonegroup

{
    "id": "02a175e2-7f23-4882-8651-6fbb15d25046",
    "name": "my_zonegroup_ck",
    "api_name": "my_zonegroup_ck",
    "is_master": true,
    "endpoints": [
        "http://vm-00:80"
    ],
    "hostnames": [
        "foo"
        "bar"
    ],
    "hostnames_s3website": [],
    "master_zone": "f42fea84-a89e-4995-996e-61b7223fb0b0",
    "zones": [
        {
            "id": "f42fea84-a89e-4995-996e-61b7223fb0b0",
            "name": "my_zone_ck",
            "endpoints": [
                "http://vm-00:80"
            ],
            "log_meta": false,
            "log_data": false,
            "bucket_index_max_shards": 11,
            "read_only": false,
            "tier_type": "",
            "sync_from_all": true,
            "sync_from": [],
            "redirect_zone": "",
            "supported_features": [
                "compress-encrypted",
                "resharding"
            ]
        }
    ],
    "placement_targets": [
        {
            "name": "default-placement",
            "tags": [],
            "storage_classes": [
                "STANDARD"
            ]
        }
    ],
    "default_placement": "default-placement",
    "realm_id": "439e9c37-4ddc-43a3-99e9-ea1f3825bb51",
    "sync_policy": {
        "groups": []
    },
    "enabled_features": [
        "resharding"
    ]
}
```

See the *hostnames* section of the zonegroup for the list of hostnames. They are specified in *zonegroup_hostnames* in the Ceph Object Gateway specification file.

Deploying Ceph Object Gateway multi-site using the `rgw` module

Bootstrapping the Ceph Object Gateway realm creates a new realm entity, a new zonegroup, and a new zone. It configures a new system user that can be used for multi-site sync operations. The `rgw` module instructs the orchestrator to create and deploy the corresponding Ceph Object Gateway daemons.

Enable the `rgw` module by using the **`ceph mgr module enable rgw`** command. After enabling the `rgw` module, either pass the arguments in the command line or use the YAML specification file to bootstrap the realm.

PREREQUISITE

- A running Red Hat Ceph Storage cluster with at least one OSD deployed.

1. Log in to the Cephadm shell.

```
cephadm shell
```

2. Enable the `rgw` module.

```
ceph mgr module enable rgw
```

For example,

```
[ceph: root@host01 /]# ceph mgr module enable rgw
```

3. Bootstrap the Ceph Object Gateway realm.
Use either the command-line or the YAML specification file to bootstrap the realm.

- Use the command-line.

```
ceph rgw realm bootstrap [--realm name REALM_NAME] [--zonegroup-
name ZONEGROUP_NAME] [--zone-name ZONE_NAME] [--port PORT_NUMBER] [--
placement HOSTNAME] [--start-radosgw]
```

For example,

```
[ceph: root@host01 /]# ceph rgw realm bootstrap --realm-name myrealm --zonegroup-
name myzonegroup --zone-name myzone --port 5500 --placement="host01 host02" --
start-radosgw
Realm(s) created correctly. Please, use 'ceph rgw realm tokens' to get the token.
```

- Use the YAML specification file.

a. As a root user, create the YAML file.

```
rgw_realm: REALM_NAME
rgw_zonegroup: ZONEGROUP_NAME
rgw_zone: ZONE_NAME
placement:
  hosts:
   - HOSTNAME_1
   - HOSTNAME_2
spec:
  rgw_frontend_port: PORT_NUMBER
zone_endpoints:http://RGW_HOSTNAME_1:RGW_PORT_NUMBER_1, http://
RGW_HOSTNAME_2:RGW_PORT_NUMBER_2
```

For example,

```
[root@host01 ~]# cat rgw.yaml

rgw_realm: myrealm
rgw_zonegroup: myzonegroup
rgw_zone: myzone
placement:
  hosts:
   - host01
   - host02
spec:
  rgw_frontend_port: 5500
zone_endpoints: http://<rgw_host1>:<rgw_port1>, http://
<rgw_host2>:<rgw_port2>
```

b. [Optional]. To deploy multiple RGW instances and dedicate some for client IO only, disable sync traffic during deployment. This ensures those instances handle client IO but do not participate in sync operations.
For those client rgw's, set **disable_multisite_sync_traffic:true** under the `spec` section of the Ceph Object Gateway specification file. Cephadm then sets the **rgw_run_sync_thread** configuration to **false** for those RGW daemons, preventing them to participate in sync operation.

This automation eliminates the need to manually configure **rgw_run_sync_thread** for those RGW users.

```
ceph config set <client_rgw_daemon> rgw_run_sync_thread false
```

In the following example, the RGW service `io.rgw` is configured with **disable_multisite_sync_traffic: true**, so it handles only client IO. The RGW

service `sync.rgw` is configured to participate in multisite sync operations as the **`disable_multisite_sync_traffic`** is false, by default.

Example

```
[root@host01 ~]# cat rgw.yaml
service_id: io.rgw
service_name: rgw.io.rgw
rgw_realm: myrealm
rgw_zonegroup: myzonegroup
rgw_zone: myzone
placement:
  hosts:
  - rgw_host1
  - rgw_host2
spec:
   rgw_frontend_port: <rgw_port>
   disable_multisite_sync_traffic: true
---
service_id: sync.rgw
service_name: rgw.sync.rgw
rgw_realm: myrealm
rgw_zonegroup: myzonegroup
rgw_zone: myzone
placement:
  hosts:
  - rgw_host3
  - rgw_host4
spec:
   rgw_frontend_port: <rgw_port>
zone_endpoints: http://<rgw_host3>:<rgw_port>, http://
<rgw_host4>:<rgw_port>
```

c. Mount the YAML file under a directory in the container, by using the **`cephadm shell --mount`** command.
For example,

```
[root@host01 ~]# cephadm shell --mount rgw.yaml:/var/lib/ceph/rgw/rgw.yaml
```

d. Bootstrap the realm, by using the **`ceph rgw realm bootstrap`** command.

> **Note:** The specification file that is used by the `rgw` module has the same format as the one used by the orchestrator. As a result, you can provide any orchestration supported Ceph Object Gateway parameters, including advanced configuration features such as SSL certificates.

For example,

```
[ceph: root@host01 /]# ceph rgw realm bootstrap -i /var/lib/ceph/rgw/
rgw.yaml
```

4. List the available tokens, by using the **`ceph rgw realm tokens`** command.

> **Note:** If you run this command before the Ceph Object Gateway daemons get deployed, it displays a message that there are no tokens as there are no endpoints yet.

For example,

```
[ceph: root@host01 /]# ceph rgw realm tokens | jq

[
  {
    "realm": "myrealm",
    "token":
"ewogICAgInJlYWxtX25hbWUiOiAibXlyZWFsbSIsCiAgICAicmVhbG1faWQiOiAiZDA3YzAwZWYtOTA0MS00Z
jZlLTg4MDQtN2Q0MDI0MDU1NmFlIiwKICAgICJlbmRwb2ludCI6ICJodHRwOi8vdm0tMDA6NDMyMSIsCiAgICA
iYWNjZXNzX2tleSI6ICI5NTY1VFZSMVFWTExFRzdVNFIxRCIsCiAgICAic2VjcmV0IjogImQ3b0FJQXZrNEdYe
Xpyd3Q2QVZ6bEZNQmNnRG53RVdMMHFDenE3cjUiCn1="
  }
]
```

5. Create the secondary zone by using these tokens and join the existing realms.

   a. As a root user, create the YAML file.
   For example,

   ```
   [root@host01 ~]# cat zone-spec.yaml
   rgw_zone: my-secondary-zone
   rgw_realm_token: <token>
   placement:
     hosts:
      - ceph-node-1
      - ceph-node-2
   spec:
     rgw_frontend_port: 5500
   ```

   To disable **multisite sync traffic** parameter for specific RGW services during deployment follow step 3(b optional). This setting disables multisite sync traffic for specific RGW services during deployment, so they handle only client IO.

   b. Mount the `zone-spec.yaml` file under a directory in the container, by using the **cephadm shell --mount** command.
   For example,

   ```
   [root@host01 ~]# cephadm shell --mount zone-spec.yaml:/var/lib/ceph/radosgw/zone-
   spec.yaml
   ```

   c. Enable the `rgw` module on the secondary zone.

   ```
   ceph mgr module enable rgw
   ```

   d. Create the secondary zone, by using the **ceph rgw zone create** command.
   For example,

   ```
   [ceph: root@host01 /]# ceph rgw zone create -i /var/lib/ceph/radosgw/zone-
   spec.yaml
   ```

AFTER COMPLETING THE TASK

Verify Object Gateway multi-site deployment, by using the **radosgw-admin realm list** command.
For example,

```
[ceph: root@host01 /]# radosgw-admin realm list
{
"default_info": "d07c00ef-9041-4f6e-8804-7d40240556ae",
"realms": [
"myrealm"
]
}
```

High availability for the Ceph Object Gateway

As a storage administrator, you can assign many instances of the Ceph Object Gateway to a single zone. This allows you to scale out as the load increases, that is, the same zone group and zone; however, you do not need a federated architecture to use a highly available proxy. Since each Ceph Object Gateway daemon has its own IP address, you can use the `ingress` service to balance the load across many Ceph Object Gateway daemons or nodes. The `ingress` service manages HAProxy and `keepalived` daemons for the Ceph Object Gateway

environment. You can also terminate HTTPS traffic at the HAProxy server, and use HTTP between the HAProxy server and the Beast front-end web server instances for the Ceph Object Gateway.

Prerequisites

- At least two Ceph Object Gateway daemons running on different hosts.
- Capacity for at least two instances of the ingress service running on different hosts

High availability service

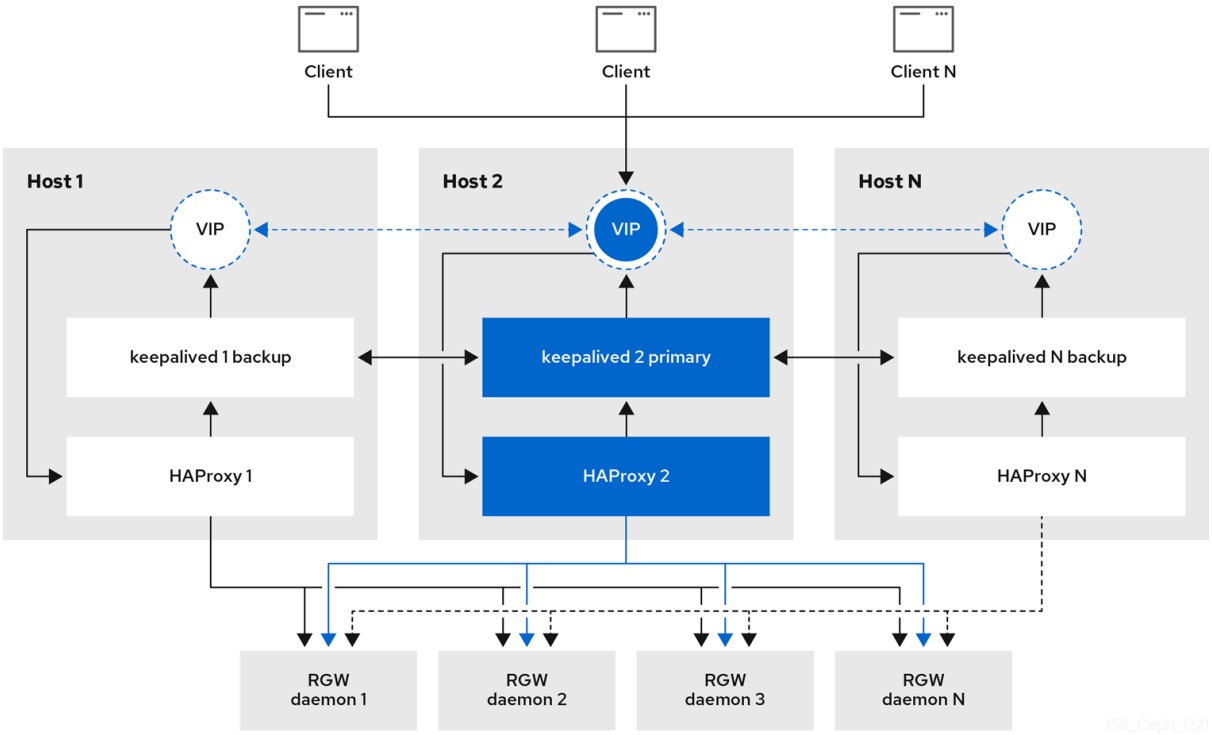
The `ingress` service provides a highly available endpoint for the Ceph Object Gateway. The `ingress` service can be deployed to any number of hosts as needed. Red Hat recommends having at least two supported Red Hat Enterprise Linux servers, each server configured with the `ingress` service. You can run a high availability (HA) service with a minimum set of configuration options. The Ceph orchestrator deploys the `ingress` service, which manages the `haproxy` and `keepalived` daemons, by providing load balancing with a floating virtual IP address. The active `haproxy` distributes all Ceph Object Gateway requests to all the available Ceph Object Gateway daemons.

A virtual IP address is automatically configured on one of the `ingress` hosts at a time, known as the primary host. The Ceph orchestrator selects the first network interface based on existing IP addresses that are configured as part of the same subnet. In cases where the virtual IP address does not belong to the same subnet, you can define a list of sub-nets for the Ceph orchestrator to match with existing IP addresses. If the `keepalived` daemon and the active `haproxy` are not responding on the primary host, then the virtual IP address moves to a backup host. This backup host becomes the new primary host.

> **Warning:** Currently, you can not configure a virtual IP address on a network interface that does not have a configured IP address.

> **Important:** To use the secure socket layer (SSL), SSL must be terminated by the `ingress` service and not at the Ceph Object Gateway.

Figure: High availability architecture



Reference

For more information, see “Configuring high availability for the Ceph Object Gateway” on page 42.

Configuring high availability for the Ceph Object Gateway

To configure high availability (HA) for the Ceph Object Gateway you write a YAML configuration file, and the Ceph orchestrator does the installation, configuration, and management of the `ingress` service. The `ingress` service uses the `haproxy` and `keepalived` daemons to provide high availability for the Ceph Object Gateway.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A minimum of two hosts running Red Hat Enterprise Linux 9 or higher, for installing the `ingress` service on.
- A healthy running Red Hat Ceph Storage cluster.
- A minimum of two Ceph Object Gateway daemons running on different hosts.
- Root-level access to the host running the `ingress` service.
- If using a firewall, then open port 80 for HTTP and port 443 for HTTPS traffic.

You can deploy an ingress service with Ceph Object Gateway as the backend, where the `use_tcp_mode_over_rgw` option is set to true in the `spec` section of the ingress specification.
For more information, see “High availability service” on page 41, and *Performing a standard RHEL 9 installation* within the product documentation for your supported Red Hat Enterprise Linux version on Red Hat Documentation.

1. Create a new `ingress.yaml` file.

   ```
   touch ingress.yaml
   ```

   For example,

   ```
   [root@host01 ~] touch ingress.yaml
   ```

2. Open the `ingress.yaml` file for editing.
   Add in the following options, and add values applicable to the environment.

```
service_type: ingress
service_id: SERVICE_ID    # adjust to match your existing RGW service
placement:
 hosts:
   - HOST1
   - HOST2
   - HOST3
spec:
 backend_service: SERVICE_ID              # adjust to match your existing RGW service
 virtual_ip: IP_ADDRESS/CIDR              # ex: 192.168.20.1/24
 frontend_port: INTEGER                   # ex: 8080
 monitor_port: INTEGER                    # ex: 1967, used by haproxy for load
balancer status
 virtual_interface_networks: [ ... ]      # optional: list of CIDR networks
 use_keepalived_multicast: TRUE/FALSE     # optional: Default is False.
 vrrp_interface_network: IP_ADDRESS/CIDR  # optional: ex: 192.168.20.0/24
 health_check_interval: TIME              # optional: Default is 2s.
 ssl: true
 certificate_source: inline               # optional: Default is cephadm-signed
 ssl_cert: |                              # optional: SSL certificate and key
   -----BEGIN CERTIFICATE-----
   ...
   -----END CERTIFICATE-----
 ssl_key: |
   -----BEGIN PRIVATE KEY-----
   ...
   -----END PRIVATE KEY-----
 enable_stats: true
 monitor_ssl: BOOLEAN
 monitor_cert_source: inline               # optional: default is reuse_service_cert
 monitor_ssl_cert: |                       # optional: SSL certificate and key
   -----BEGIN CERTIFICATE-----
   ...
   -----END CERTIFICATE-----
 monitor_ssl_key: |
   -----BEGIN PRIVATE KEY-----
   ...
   -----END PRIVATE KEY-----
 monitor_networks: [..]
 monitor_ip_addrs:
   host: IP_ADDRESSSS/CIDR
```

"Ingress service configuration" on page 43 provides the ingress service configuration parameters and descriptions.

Ingress service configuration

Parameter	Description
service_type	Mandatory and set to `ingress`.
service_id	Name of the service, for example, `rgw.foo`.
placement hosts	Hosts for HA daemons (haproxy and keepalived). Need not match RGW nodes.
virtual_ip	Virtual IP in CIDR format for ingress service.
virtual_ips_list	List of virtual IPs in CIDR format. Each IP is primary on one ingress node.
virtual_interface_networks	Networks to identify ethernet interface for virtual IP.
frontend_port	Port used to access the ingress service.
ssl_cert	SSL certificate in .pem format with certificate and key blocks.
ssl	Enable SSL for ingress service.
certificate_source	Source of certificate: `inline`, `reference`, or `cephadm-signed`. Default is `cephadm-signed`.

Parameter	Description
ssl_cert	SSL certificate in `.pem` format if **`certificate_source`** is not `cephadm-signed`.
ssl_key	SSL key in `.pem` format if **`certificate_source`** is not `cephadm-signed`.
use_keepalived_multicast	Default is `False`. Uses unicast IPs for keepalived config.
health_check_interval	Interval between haproxy health checks. Default is 2 seconds.
enable_stats	Enable haproxy stats. Default is `False`.
monitor_ssl	Enable SSL for monitoring. Requires service SSL enabled.
monitor_cert_source	Source of monitor certificate: `reuse_service_cert`, `inline`, `reference`, or `cephadm-signed`. Default is `reuse_service_cert`.
monitor_ssl_cert	Monitor SSL certificate in .pem format if **`monitor_cert_source`** is not`cephadm-signed`.
monitor_ssl_key	Monitor SSL key in `.pem` format if **`monitor_cert_source`** is not`cephadm-signed`.
monitor_ip_addrs	If assigned to host, it will be used. Otherwise, **`monitor_networks`** will be checked.
monitor_networks	If specified, an IP matching one of the networks will be used. Otherwise, default host IP is used.

Following is an example of providing an SSL cert:

```
service_type: ingress
service_id: rgw.foo
placement:
  hosts:
    - host01.example.com
    - host02.example.com
    - host03.example.com
spec:
  backend_service: rgw.foo
  virtual_ip: 192.168.1.2/24
  frontend_port: 8080
  monitor_port: 1967
  virtual_interface_networks:
    - 10.10.0.0/16
  ssl_cert: |
    -----BEGIN CERTIFICATE-----
    MIIEpAIBAAKCAQEA+Cf4l9OagD6x67HhdCy4Asqw89Zz9ZuGbH50/7ltIMQpJJU0
    gu9ObNtIoC0zabJ7n1jujueYgIpOqGnhRSvsGJiEkgN81NLQ9rqAVaGpadjrNLcM
    bpgqJCZj0vzzmtFBCtenpb5l/EccMFcAydGtGeLP33SaWiZ4Rne56GBInk6SATI/
    JSKweGD1y5GiAWipBR4C74HiAW9q6hCOuSdp/2WQxWT3T1j2sjlqxkHdtInUtwOm
    j5Ism276IndeQ9hR3reFR8PJnKIPx73oTBQ7p9CMR1J4ucq9Ny0J12wQYT00fmJp
    -----END CERTIFICATE-----
    -----BEGIN PRIVATE KEY-----
    MIIEBTCCAu2gAwIBAgIUGfYFsj8HyA9Zv2l600hxzT8+gG4wDQYJKoZIhvcNAQEL
    BQAwgYkxCzAJBgNVBAYTAklOMQwwCgYDVQQIDANLQVIxDDAKBgNVBAcMA0JMUjEM
    MAoGA1UECgwDUkhUMQswCQYDVQQLDAJCVTEkMCIGA1UEAwwbY2VwaC1zc2wtcmhj
    czUtOGRjeHY2LW5vZGU1MR0wGwYJKoZIhvcNAQkBFg5hYmNAcmVkaGF0LmNvbTAe
    -----END PRIVATE KEY-----
```

Following is an example of an ingress file without providing an SSL cert:

```
service_type: ingress

service_id: rgw.ssl    # adjust to match your existing RGW service

placement:
hosts:
- hostname1
- hostname2
```

```
spec:
backend_service: rgw.rgw.ssl.ceph13   # adjust to match your existing RGW service
virtual_ip: IP_ADDRESS/CIDR           # ex: 192.168.20.1/24
frontend_port: INTEGER                # ex: 443
monitor_port: INTEGER                 # ex 1969
use_tcp_mode_over_rgw: True
```

3. Launch the `cephadm` shell.

```
cephadm shell --mount ingress.yaml:FILEPATH/ingress.yaml
```

For example,

```
[root@host01 ~]# cephadm shell --mount ingress.yaml:/var/lib/ceph/radosgw/ingress.yaml
```

4. Configure the latest `haproxy` and `keepalived` images.

```
ceph config set mgr mgr/cephadm/container_image_haproxy HAPROXY_IMAGE_ID
ceph config set mgr mgr/cephadm/container_image_keepalived KEEPALIVED_IMAGE_ID
```

For example,

```
[ceph: root@host01 /]# ceph config set mgr mgr/cephadm/container_image_haproxy
registry.redhat.io/rhceph/rhceph-haproxy-rhel9:latest
[ceph: root@host01 /]# ceph config set mgr mgr/cephadm/container_image_keepalived
registry.redhat.io/rhceph/keepalived-rhel9:latest
```

5. Install and configure the new `ingress` service using the Ceph orchestrator.

```
ceph orch apply -i FILEPATH/ingress.yaml
```

For example,

```
[ceph: root@host01 /]# ceph orch apply -i /var/lib/ceph/radosgw/ingress.yaml
```

6. After the Ceph orchestrator completes, verify the HA configuration.
   a. On the host running the `ingress` service, check that the virtual IP address displays.

```
ip addr show
```

For example,

```
[root@host01 ~]# ip addr show
```

   b. Try reaching the Ceph Object Gateway from a Ceph client.

```
wget HOST_NAME
```

For example,

```
[root@client ~]# wget host01.example.com
```

The HA configuration for the Ceph Object Gateway is working properly if this returns an `index.html` file, similar to the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
        <ListAllMyBucketsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
                <Owner>
                        <ID>anonymous</ID>
                        <DisplayName></DisplayName>
                </Owner>
                <Buckets>
                </Buckets>
        </ListAllMyBucketsResult>
```

Adjusting logging and debugging output

Debug output can be changed and log output can be adjusted to save as files.

After finishing the setup procedure, check your logging output to ensure it meets your needs. By default, the Ceph daemons log to `journald`, and you can view the logs using the `journalctl` command. Alternatively, you can also have the Ceph daemons log to files, which are located under the `/var/log/ceph/CEPH_CLUSTER_ID/` directory.

> **Important:** Verbose logging can generate over 1 GB of data per hour. This type of logging can potentially fill up the operating system's disk, causing the operating system to stop functioning.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway software.

Procedure

1. Set the following parameter to increase the Ceph Object Gateway logging output:

   Syntax

   ```
   ceph config set client.rgw debug_rgw VALUE
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph config set client.rgw debug_rgw 20
   ```

   a. You can also modify these settings at runtime:

      Syntax

      ```
      ceph --admin-daemon /var/run/ceph/CEPH_CLUSTER_ID/ceph-client.rgw.NAME.asok
      config set debug_rgw VALUE
      ```

      Example

      ```
      [ceph: root@host01 /]# ceph --admin-daemon /var/run/ceph/62a081a6-88aa-11eb-
      a367-001a4a000672/ceph-client.rgw.rgw.asok config set debug_rgw 20
      ```

2. Optionally, you can configure the Ceph daemons to log their output to files. Set the `log_to_file`, and `mon_cluster_log_to_file` options to `true`:

   Example

   ```
   [ceph: root@host01 /]# ceph config set global log_to_file true
   [ceph: root@host01 /]# ceph config set global mon_cluster_log_to_file true
   ```

Reference

For more information, see the following:

- Debugging and logging configuration options

Configuration

As a storage administrator, learning the basics of configuring the Ceph Object Gateway is important. You can learn about the defaults and the embedded web server called Beast.

For troubleshooting issues with the Ceph Object Gateway, you can adjust the logging and debugging output generated by the Ceph Object Gateway. Also, you can provide a High-Availability proxy for storage cluster access using the Ceph Object Gateway.

Add a wildcard to the DNS

Add the wildcard such as hostname to the DNS record of the DNS server.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Ceph Object Gateway installed.
- Root-level access to the admin node.

Procedure

1. To use Ceph with S3-style sub-domains, add a wildcard to the DNS record of the DNS server that the `ceph-radosgw` daemon uses to resolve domain names:

   Syntax

   ```
   bucket-name.domain-name.com
   ```

   For `dnsmasq`, add the following address setting with a dot (.) prepended to the host name:

   Syntax

   ```
   address=/.HOSTNAME_OR_FQDN/HOST_IP_ADDRESS
   ```

   Example

   ```
   address=/.gateway-host01/192.168.122.75
   ```

   For `bind`, add a wildcard to the DNS record:

   Example

   ```
   $TTL    604800
   @       IN      SOA     gateway-host01. root.gateway-host01. (
                                 2         ; Serial
                            604800         ; Refresh
                             86400         ; Retry
                           2419200         ; Expire
                            604800 )       ; Negative Cache TTL
   ;
   @       IN      NS      gateway-host01.
   @       IN      A       192.168.122.113
   *       IN      CNAME   @
   ```

2. Restart the DNS server and ping the server with a subdomain to ensure that the `ceph-radosgw` daemon can process the subdomain requests:

   Syntax

   ```
   ping mybucket.HOSTNAME
   ```

   Example

   ```
   [root@host01 ~]# ping mybucket.gateway-host01
   ```

3. If the DNS server is on the local machine, you might need to modify `/etc/resolv.conf` by adding a nameserver entry for the local machine.
4. Add the host name in the Ceph Object Gateway zone group:

a. Get the zone group:

Syntax

```
radosgw-admin zonegroup get --rgw-zonegroup=ZONEGROUP_NAME > zonegroup.json
```

Example

```
[ceph: root@host01 /]# radosgw-admin zonegroup get --rgw-zonegroup=us >
zonegroup.json
```

b. Take a back-up of the JSON file:

Example

```
[ceph: root@host01 /]# cp zonegroup.json zonegroup.backup.json
```

c. View the `zonegroup.json` file:

Example

```
[ceph: root@host01 /]# cat zonegroup.json
{
    "id": "d523b624-2fa5-4412-92d5-a739245f0451",
    "name": "asia",
    "api_name": "asia",
    "is_master": "true",
    "endpoints": [],
    "hostnames": [],
    "hostnames_s3website": [],
    "master_zone": "d2a3b90f-f4f3-4d38-ac1f-6463a2b93c32",
    "zones": [
        {
            "id": "d2a3b90f-f4f3-4d38-ac1f-6463a2b93c32",
            "name": "india",
            "endpoints": [],
            "log_meta": "false",
            "log_data": "false",
            "bucket_index_max_shards": 11,
            "read_only": "false",
            "tier_type": "",
            "sync_from_all": "true",
            "sync_from": [],
            "redirect_zone": ""
        }
    ],
    "placement_targets": [
        {
            "name": "default-placement",
            "tags": [],
            "storage_classes": [
                "STANDARD"
            ]
        }
    ],
    "default_placement": "default-placement",
    "realm_id": "d7e2ad25-1630-4aee-9627-84f24e13017f",
    "sync_policy": {
        "groups": []
    }
}
```

d. Update the `zonegroup.json` file with new host name:

Example

```
"hostnames": ["host01", "host02","host03"],
```

e. Set the zone group back in the Ceph Object Gateway:

Syntax

```
radosgw-admin zonegroup set --rgw-zonegroup=ZONEGROUP_NAME --
infile=zonegroup.json
```

Example

```
[ceph: root@host01 /]# radosgw-admin zonegroup set --rgw-zonegroup=us --
infile=zonegroup.json
```

f. Update the period:

Example

```
[ceph: root@host01 /]# radosgw-admin period update --commit
```

g. Restart the Ceph Object Gateway so that the DNS setting takes effect.

Reference

For more information, see the following:

- "Configuration database" on page 0

Beast front-end web server

The Ceph Object Gateway provides Beast, a C/C embedded front-end web server. Beast uses the `Boost.Beast` C library to parse HTTP, and `Boost.Asio` for asynchronous network I/O.

Related information

Boost C++ Libraries

Beast configuration options

Use this information to know which Beast configuration options can be passed to the embedded web server in the Ceph configuration file for the Ceph Object Gateway.

"Beast configuration options that can be passed to the embedded web server" on page 49 lists the Beast configuration options that can be passed to the embedded web server in the Ceph configuration file for the Ceph Object Gateway. Each option has a default value. If a value is not specified, the default value is empty.

Beast configuration options that can be passed to the embedded web server

Option	Description	Default
`endpoint` and `ssl_endpoint`	Sets the listening address in the form `address[:port]` where the address is an IPv4 address string in dotted decimal form, or an IPv6 address in hexadecimal notation surrounded by brackets, [ ]. The optional port defaults to `8080` for `endpoint` and `443` for `ssl_endpoint`. It can be specified multiple times as in `endpoint=[::1] endpoint=192.168.0.100:8000`.	EMPTY
`ssl_certificate`	Path to the SSL certificate file used for SSL-enabled endpoints.	EMPTY
`ssl_private_key`	Optional path to the private key file used for SSL-enabled endpoints. If one is not given the file that is specified by `ssl_certificate` is used as the private key.	EMPTY
`tcp_nodelay`	Performance optimization in some environments.	EMPTY

Note: By default, the Beast front end writes an access log line recording all requests that are processed by the server to the RADOS Gateway log file.

Example `/etc/ceph/ceph.conf` file with Beast options using SSL

```
...

[client.rgw.node1]
rgw frontends = beast ssl_endpoint=192.168.0.100:443 ssl_certificate=<path to SSL
certificate>
```

Configuring SSL for Beast

You can configure the Beast front-end web server to use the OpenSSL library to provide Transport Layer Security (TLS). To use Secure Socket Layer (SSL) with Beast, you need to obtain a certificate from a Certificate Authority (CA) that matches the hostname of the Ceph Object Gateway node. Beast also requires the secret key, server certificate, and any other CA in a single `.pem` file.

> **Important:**
>
> - Prevent unauthorized access to the `.pem` file, because it contains the secret key hash.
> - It is recommended to obtain certificate from a CA with the Subject Alternative Name (SAN) field, and a wildcard for use with S3-style sub-domains.

If the Ceph Object Gateway acts as a client and a custom certificate is used on the server, you can inject a custom CA by importing it on the node and then mapping the `etc/pki` directory into the container with the **`extra_container_args`** parameter in the Ceph Object Gateway specification file.

Before you begin

- A running, and healthy Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway software package.
- Installation of the OpenSSL software package.
- Root-level access to the Ceph Object Gateway node.

Procedure

1. Create a new file named `rgw.yml` in the current directory:
   Example

   ```
   [ceph: root@host01 /]# touch rgw.yml
   ```

2. Open the `rgw.yml` file for editing, and customize it for the environment:

   ```
   service_type: rgw
   service_id: SERVICE_ID
   service_name: SERVICE_NAME
   placement:
     hosts:
     - HOST_NAME
   spec:
     ssl: true
     rgw_frontend_ssl_certificate: CERT_HASH
   ```

   Example

   ```
   service_type: rgw
   service_id: foo
   service_name: rgw.foo
   placement:
     hosts:
     - host01
   ```

```
spec:
  ssl: true
  rgw_frontend_ssl_certificate: |
    -----BEGIN RSA PRIVATE KEY-----
    MIIEpAIBAAKCAQEA+Cf4l9OagD6x67HhdCy4Asqw89Zz9ZuGbH50/7ltIMQpJJU0
    gu9ObNtIoC0zabJ7n1jujueYgIpOqGnhRSvsGJiEkgN81NLQ9rqAVaGpadjrNLcM
    bpgqJCZj0vzzmtFBCtenpb5l/EccMFcAydGtGeLP33SaWiZ4Rne56GBInk6SATI/
    JSKweGD1y5GiAWipBR4C74HiAW9q6hCOuSdp/2WQxWT3T1j2sjlqxkHdtInUtwOm
    j5Ism276IndeQ9hR3reFR8PJnKIPx73oTBQ7p9CMR1J4ucq9Ny0J12wQYT00fmJp
    -----END RSA PRIVATE KEY-----
    -----BEGIN CERTIFICATE-----
    MIIEBTCCAu2gAwIBAgIUGfYFsj8HyA9Zv2l600hxzT8+gG4wDQYJKoZIhvcNAQEL
    BQAwgYkxCzAJBgNVBAYTAklOMQwwCgYDVQQIDANLQVIxDDAKBgNVBAcMA0JMUjEM
    MAoGA1UECgwDUkhUMQswCQYDVQQLDAJCVTEkMCIGA1UEAwwbY2VwaC1zc2wtcmhj
    czUtOGRjeHY2LW5vZGU1MR0wGwYJKoZIhvcNAQkBFg5hYmNAcmVkaGF0LmNvbTAe
    -----END CERTIFICATE-----
```

3. Deploy the Ceph Object Gateway using the service specification file:
   Example

```
[ceph: root@host01 /]# ceph orch apply -i rgw.yml
```

Deploying self-signed Cephadm certificates with SAN configuration

Configuring self-signed SSL/TLS certificates with Subject Alternative Name (SAN) entries for the Ceph Object Gateway (RGW) is essential for securing communication across RGW nodes in multi-site or multi-domain Ceph environments. This process enables administrators to automate the generation and deployment of certificates that support multiple domain names or hostnames, providing greater flexibility and improved security for distributed deployments.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running, and healthy Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway software package.
- Installation of the OpenSSL software package.
- Root-level access to the Ceph Object Gateway node.

Deploying self-signed Cephadm certificates with SAN configuration allows you to:

- Automatically generate and deploy SSL/TLS certificates for RGW services.
- Include multiple SAN entries to support various domains or hostnames.
- Secure communication between RGW nodes across different Ceph clusters, even across geographic regions or domain boundaries.

This approach simplifies secure, scalable multi-site RGW deployments by ensuring encrypted communication between all participating nodes.

1. Create a new file named `rgw.yml` in the current directory.
   For example,

```
[ceph: root@host01 /]# touch rgw.yml
```

2. Open the `rgw.yml` file for editing, and customize it for the environment.

```
service_type: rgw
service_id: SERVICE_ID
service_name: SERVICE_NAME
placement:
  hosts:
  - HOST_NAME
spec:
  ssl: true
  rgw_frontend_ssl_certificate: CERT_HASH
```

   For example,

```
service_type: rgw
service_id: foo
service_name: rgw.test
placement:
  hosts:
  - host01
spec:
  ssl: true
  rgw_frontend_ssl_certificate: |
    -----BEGIN RSA PRIVATE KEY-----
    MIIEpAIBAAKCAQEA+Cf4l9OagD6x67HhdCy4Asqw89Zz9ZuGbH50/7ltIMQpJJU0
    gu9ObNtIoC0zabJ7n1jujueYgIpOqGnhRSvsGJiEkgN81NLQ9rqAVaGpadjrNLcM
    bpgqJCZj0vzzmtFBCtenpb5l/EccMFcAydGtGeLP33SaWiZ4Rne56GBInk6SATI/
    JSKweGD1y5GiAWipBR4C74HiAW9q6hCOuSdp/2WQxWT3T1j2sjlqxkHdtInUtwOm
    j5Ism276IndeQ9hR3reFR8PJnKIPx73oTBQ7p9CMR1J4ucq9Ny0J12wQYT00fmJp
    -----END RSA PRIVATE KEY-----
    -----BEGIN CERTIFICATE-----
    MIIEBTCCAu2gAwIBAgIUGfYFsj8HyA9Zv2l600hxzT8+gG4wDQYJKoZIhvcNAQEL
    BQAwgYkxCzAJBgNVBAYTAklOMQwwCgYDVQQIDANLQVIxDDAKBgNVBAcMA0JMUjEM
    MAoGA1UECgwDUkhUMQswCQYDVQQLDAJCVTEkMCIGA1UEAwwbY2VwaC1zc2wtcmhj
    czUtOGRjeHY2LW5vZGU1MR0wGwYJKoZIhvcNAQkBFg5hYmNAcmVkaGF0LmNvbTAe
    -----END CERTIFICATE-----
```

3. Deploy the Ceph Object Gateway using the service specification file that includes SSL configuration and SAN settings.

> **Note:** Be sure to create the **`rgw_realm`**, **`rgw_zone`**, and **`rgw_zonegroup`** before deploying the `rgw` service with SSL.

For example,

```
# cat rgw_spec.yaml

service_type: rgw
service_id: bar
service_name: rgw.bar
placement:
  hosts:
    - ceph-pri-hsm-squid-j9zu1f-node5
spec:
  rgw_frontend_port: 8085
  generate_cert: true
  rgw_realm: india
  rgw_zone: primary
  rgw_zonegroup: shared
  ssl: true
  zonegroup_hostnames:
  - s3.example.com
  - s3.zone1.example.com
```

AFTER COMPLETING THE TASK

After the Ceph Object Gateway (`rgw`) daemons are deployed, verify the certificate.

1. Retrieve the certificate.

```
ceph orch cert-store get cert rgw_frontend_ssl_cert --service-name RGW_SERVICE_NAME
```

2. Check the SANs field of the certificate that is using OpenSSL.

```
openssl x509 -in rgw_cert.txt -noout -text
```

For example,

```
[root@vm-00 ~]# openssl x509 -in rgw_cert.txt -noout -text
X509v3 extensions:
X509v3 Subject Alternative Name:
DNS:s3.zone1.example.com, DNS:s3.example.com
```

The X509v3 Subject Alternative Name section shows the SANs populated based on the `zonegroup_hostnames` that is defined in the `rgw` service specification.

The Ceph Object Gateway is now configured with SSL/TLS certificates, including the SAN modifications to support multiple hostnames or domains as required for multi-zone or multi-region deployments.

Configuring RGW multi-site with SSL using Cephadm

Configure RADOS Gateway (RGW) with SSL using Cephadm in a multisite Ceph cluster. This setup secures RGW endpoints and enables data replication across sites, ensuring high availability, disaster recovery, and geographic redundancy. SSL encrypts client-to-RGW communication for secure data access.

Prerequisites

- A running, and healthy Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway software package.
- Installation of the OpenSSL software package.
- Root-level access to the Ceph Object Gateway node.

Procedure

1. Get the Cephadm-Root CA Certificate from the primary Site:

```
ceph orch cert-store get cert cephadm_root_ca_cert
```

   Example

```
[root@ceph-pri-hsm-cephadm-h0a759-node6 ~]# ceph orch cert-store get cert
cephadm_root_ca_cert

-----BEGIN CERTIFICATE-----

MIIE4DCCAsigAwIBAgIUZJdD1pvtpWwIBdk9WgQltOMsssEwDQYJKoZIhvcNAQEL

BQAwFzEVMBMGA1UEAwwMY2VwaGFkbS1yb290MB4XDTI0MTAzMDE0MzcwNFoXDTM0

MTAzMTE0MzcwNFowFzEVMBMGA1UEAwwMY2VwaGFkbS1yb290MIICIjANBgkqhkiG

9w0BAQEFAAOCAg8AMIICCgKCAgEAmYkNVL3rXAblrK12IEXcem4xW8Ki3bySGDcx

ft9vSSDRNCTHUYt8LLeQ+EoOVmQGbDwF2/z3FmC8+Mdkg8dB/7c0zpjw1RAkegOy

gZCxHxWk/oT25wZcr4SogDzSFbT7dPX3P3H+PxMa/bzvH7vgZjkTgT94sWXu3+Ll

5YbiD/DMEmgty2+UO+ZQOWQB9u9s8d8+tqGuaeAgAdB70D8wi/z4k+u76s4/697o

VzyfF7Gj9uCacCJUIs8XXUkIkqlFGxWMJa/ENSHQdd/Un+nWDyiN0KCC5b98EMru

lvqrlbdmmOsfUQFDIGI+s4/S90PZXsu8kjCj+sDB+A1371uaeWbn6jC5wjb6+p1H

xcCnqwJUO3eJAk6HAvD0OGb6raaVdN5wmFxF8rQ/Ghmuk2UHyupjD5ieotyI8sad

ugctyGLwx8/5BFSD7SDwvP2wqbIhGDv9yqpTUHpZqZLik1TOqvrRytdzgtMj2OK9

E6YmOYTajcXWshH1WwOoMjhPohHynXDTZdkNHWvfoy6KcvHYX9AiIEYX95CbM5O7

jOa9dMjp+tCbWrejZjyVV5zwtVsi5K2TmKOJseWg1QljqXU66ft7gvP69rG12e64

ZChnPiGjYJZoG+lJHgbb84Khhfg1foD1e/B6+3gaWpCkzlpvajzf4IhRGDaSRsKL

NqQVraECAwEAAaMkMCIwDwYDVR0RBAgwBocECgBA8zAPBgNVHRMBAf8EBTADAQH/

MA0GCSqGSIb3DQEBCwUAA4ICAQApdrsNb4MjpuC81Dz7/9d3QkiafULbxvvEhIgZ

dZZdfQPTv68HK4N0Ubw/3st5HRJCyejAICpiqgMnvR6q7DLgJ4CdMtvFiIRYZvG4

SF48s5ri1F5mBuIChdtd02IsjVt/zUt7Z4Wbco2F/g6MF6v5vJ5s8ZoOokNhw+h6

lcD85x+Bvg1ytbtJGFitTu+Rhi3h3SZhz2lJVyLkAj4C7R1Te/uUvPP1AiAVvzL9

bMY/KPPjqxsBf6YDlgNncZyRaXsu6lh428lviwUlz/Yl9hOIyYXBv2UYu41H22f3

u2Ewc+bNH2GfWFbbicXUSqIg6fFdKUgrXU/uRC99OIxVHU/P0KWIHJzsCclAJpBA

J7KMGxOpRctMPl0aRtK4yWY92/E8rxNR5LAMBosE1V4iCQr9QkxFiu76Bp2Vz35f

Ly/0vwfwXwYt2E3DVKKegcnIr355SjRoBChaYAsVK8qvSmtzQQENbUECMUKVtdtZ

NbFRvfor+9uQemMC5ty9OfXCfTnLZ/wtL17NEMy5/7249L/3JrEwixOyooZR8zY0

wR7eN9iI5XHSW11F3qBdpqvpVubSrzxyoUPD2QpAWyraXn2QlLqsVTrCDj+heMda

DTL4YvDlOnKyb08etBCJKKg90sJxkqTfQKipogTtQSlQrMLwGLfpf67NsJ9izeYt

Mn1JbQ==

-----END CERTIFICATE-----

[root@ceph-pri-hsm-cephadm-h0a759-node6 ~]#
```

2. On the secondary site, place the certificate content in `/etc/pki/ca-trust/source/anchors` file.

```
[root@ceph-sec-hsm-cephadm-h0a759-node6 ~]# cd /etc/pki/ca-trust/source/anchors

[root@ ceph-sec-hsm-cephadm-h0a759-node6 anchors]#
```

Example

```
[root@ceph-sec-hsm-cephadm-h0a759-node6 ~]# cd /etc/pki/ca-trust/source/anchors

[root@ceph-sec-hsm-cephadm-h0a759-node6 anchors]# ls

RH-IT-Root-CA.crt  ceph-qe-ca.pem  cephqe-ca.pem  pri-site-cephadm-root-ca.crt
```

```
[root@ceph-sec-hsm-cephadm-h0a759-node6 anchors]# cat pri-site-cephadm-root-ca.crt
-----BEGIN CERTIFICATE-----
MIIE4DCCAsigAwIBAgIUZJdD1pvtpWwIBdk9WgQltOMsssEwDQYJKoZIhvcNAQEL
BQAwFzEVMBMGA1UEAwwMY2VwaGFkbS1yb290MB4XDTI0MTAzMDE0MzcwNFoXDTM0
MTAzMTE0MzcwNFowFzEVMBMGA1UEAwwMY2VwaGFkbS1yb290MIICIjANBgkqhkiG
9w0BAQEFAAOCAg8AMIICCgKCAgEAmYkNVL3rXAblrK12IEXcem4xW8Ki3bySGDcx
ft9vSSDRNCTHUYt8LLeQ+EoOVmQGbDwF2/z3FmC8+Mdkg8dB/7c0zpjw1RAkegOy
gZCxHxWk/oT25wZcr4SogDzSFbT7dPX3P3H+PxMa/bzvH7vgZjkTgT94sWXu3+Ll
5YbiD/DMEmgty2+UO+ZQOWQB9u9s8d8+tqGuaeAgAdB70D8wi/z4k+u76s4/697o
VzyfF7Gj9uCacCJUIs8XXUkIkqlFGxWMJa/ENSHQdd/Un+nWDyiN0KCC5b98EMru
lvqrlbdmmOsfUQFDIGI+s4/S90PZXsu8kjCj+sDB+A1371uaeWbn6jC5wjb6+p1H
xcCnqwJUO3eJAk6HAvD0OGb6raaVdN5wmFxF8rQ/Ghmuk2UHyupjD5ieotyI8sad
ugctyGLwx8/5BFSD7SDwvP2wqbIhGDv9yqpTUHpZqZLik1TOqvrRytdzgtMj2OK9
E6YmOYTajcXWshH1WwOoMjhPohHynXDTZdkNHWvfoy6KcvHYX9AiIEYX95CbM5O7
jOa9dMjp+tCbWrejZjyVV5zwtVsi5K2TmKOJseWg1QljqXU66ft7gvP69rG12e64
ZChnPiGjYJZoG+lJHgbb84Khhfg1foD1e/B6+3gaWpCkzlpvajzf4IhRGDaSRsKL
NqQVraECAwEAAaMkMCIwDwYDVR0RBAgwBocECgBA8zAPBgNVHRMBAf8EBTADAQH/
MA0GCSqGSIb3DQEBCwUAA4ICAQApdrsNb4MjpuC81Dz7/9d3QkiafULbxvvEhIgZ
dZZdfQPTv68HK4N0Ubw/3st5HRJCyejAICpiqgMnvR6q7DLgJ4CdMtvFiIRYZvG4
SF48s5ri1F5mBuIChdtd02IsjVt/zUt7Z4Wbco2F/g6MF6v5vJ5s8ZoOokNhw+h6
lcD85x+Bvg1ytbtJGFitTu+Rhi3h3SZhz2lJVyLkAj4C7R1Te/uUvPP1AiAVvzL9
bMY/KPPjqxsBf6YDlgNncZyRaXsu6lh428lviwUlz/Yl9hOIyYXBv2UYu41H22f3
u2Ewc+bNH2GfWFbbicXUSqIg6fFdKUgrXU/uRC99OIxVHU/P0KWIHJzsCclAJpBA
J7KMGxOpRctMPl0aRtK4yWY92/E8rxNR5LAMBosE1V4iCQr9QkxFiu76Bp2Vz35f
Ly/0vwfwXwYt2E3DVKKegcnIr355SjRoBChaYAsVK8qvSmtzQQENbUECMUKVtdtZ
NbFRvfor+9uQemMC5ty9OfXCfTnLZ/wtL17NEMy5/7249L/3JrEwixOyooZR8zY0
wR7eN9iI5XHSW11F3qBdpqvpVubSrzxyoUPD2QpAWyraXn2QlLqsVTrCDj+heMda
DTL4YvDlOnKyb08etBCJKKg90sJxkqTfQKipogTtQSlQrMLwGLfpf67NsJ9izeYt
Mn1JbQ==
-----END CERTIFICATE-----
[root@ceph-sec-hsm-cephadm-h0a759-node6 anchors]#
```

3. Run the **update-ca-trust** command:

```
[root@ceph-sec-hsm-cephadm-h0a759-node6 anchors]# update-ca-trust
[root@ceph-sec-hsm-cephadm-h0a759-node6 anchors]#
```

4. Verify Connectivity with Radosgw. Now that the SSL certificate is trusted, you can execute radosgw-admin commands such as **realm pull**or check the sync status without encountering SSL certificate issues.

   Example

```
[root@ceph-sec-hsm-cephadm-h0a759-node6 anchors]# date; radosgw-admin realm pull --
rgw-realm india --url https://10.0.65.88 --access-key 21e86bce636c3aa0 --secret
cf764951f1fdde5d --default

2024-10-30T17:23:54.450+0000 7fe300751800  1 Set the period's master zonegroup shared
as the default

{

    "id": "b32e147e-6276-4a70-952c-334cc11d998c",

    "name": "india",

    "current_period": "3013ead2-56a8-4d90-808b-800d88817ac7",

    "epoch": 2

}

[root@ceph-sec-hsm-cephadm-h0a759-node6 anchors]#

[root@ceph-sec-hsm-cephadm-h0a759-node6 anchors]# date; curl  https://10.0.65.88

<?xml version="1.0" encoding="UTF-8"?><ListAllMyBucketsResult xmlns="http://
s3.amazonaws.com/doc/2006-03-01/"><Owner><ID>anonymous</ID></Owner><Buckets></
Buckets></ListAllMyBucketsResult>[root@ceph-sec-hsm-cephadm-h0a759-node6 anchors]#

[root@ceph-sec-hsm-cephadm-h0a759-node6 anchors]#
```

5. Repeat the process on the primary site. To establish mutual trust, copy the secondary site's RGW SSL certificate to the primary site using the same steps. This ensures both sites trust each other's certificates and allows seamless multisite RGW communication.

D3N Data Cache

Datacenter-Data-Delivery Network (D3N) uses high-speed storage, such as `NVMe`, to cache datasets on the access side. Such caching allows big data jobs to use the compute and fast-storage resources available on each Ceph Object Gateway node at the edge. The Ceph Object Gateways act as cache servers for the back-end object store (OSDs), storing data locally for reuse.

> **Note:** Each time the Ceph Object Gateway is restarted the content of the cache directory is purged.

Adding D3N cache directory

To enable D3N cache on Ceph Object Gateway, you need to also include the D3N cache directory in the podman `unit.run` file.

PREREQUISITE

- A running Red Hat Ceph Storage cluster.
- Ceph Object Gateway installed.
- Root-level access to the admin node.
- A fast NVMe drive in each Ceph Object Gateway node to serve as the local cache storage.

1. Create a mount point for the NVMe drive.

```
mkfs.ext4 NVME_DRIVE_PATH
```

For example,

```
[ceph: root@host01 /]# mkfs.ext4 /dev/nvme0n1
mount /dev/nvme0n1 /mnt/nvme0n1/
```

2. Create a cache directory path.

```
mkdir NVME_MOUNT_PATH/CACHE_DIRECTORY_NAME
```

For example,

```
[ceph: root@host01 /]# mkdir /mnt/nvme0n1/rgw_datacache
```

3. Provide a+rwx permission to *NVME_MOUNT_PATH* and to the *rgw_d3n_l1_datacache_persistent_path*.

```
chmod a+rwx NVME_MOUNT_PATH ; chmod a+rwx rgw_d3n_l1_datacache_persistent_path
```

For example,

```
[ceph: root@host01 /]# chmod a+rwx /mnt/nvme0n1 ; chmod a+rwx /mnt/nvme0n1/
rgw_datacache/
```

4. Create or modify the Ceph Object Gateway specification file with extra_container_args to add *rgw_d3n_l1_datacache_persistent_path* into podman unit.run file.

```
"extra_container_args:
          "-v"
          "rgw_d3n_l1_datacache_persistent_path:rgw_d3n_l1_datacache_persistent_path"
            "
```

For example,

```
[ceph: root@host01 /]# cat rgw-spec.yml
    service_type: rgw
    service_id: rgw.test
    placement:
        hosts:
            host1
            host2
        extra_container_args:
            "-v"
            "/mnt/nvme0n1/rgw_datacache/:/mnt/nvme0n1/rgw_datacache/"
```

If there are multiple instances of Ceph Object Gateway in a single host, then a separate *rgw_d3n_l1_datacache_persistent_path* must be created for each instance and add each path in extra_container_args.
For two instances of Ceph Object Gateway in each host, create two separate instances of cache-directory under *rgw_d3n_l1_datacache_persistent_path*, /mnt/nvme0n1/rgw_datacache/rgw1 and /mnt/nvme0n1/rgw_datacache/rgw2.

Example for extra_container_args in a Ceph Object Gateway specification file

```
"extra_container_args:
    "-v"
    "/mnt/nvme0n1/rgw_datacache/rgw1/:/mnt/nvme0n1/rgw_datacache/rgw1/"
    "-v"
    "/mnt/nvme0n1/rgw_datacache/rgw2/:/mnt/nvme0n1/rgw_datacache/rgw2/"
"
```

Example for Ceph Object Gateway specification YAML file

```
[ceph: root@host01 /]# cat rgw-spec.yml
    service_type: rgw
        service_id: rgw.test
        placement:
            hosts:
                host1
                host2
            count-per-host: 2
            extra_container_args:
                "-v"
                "/mnt/nvme0n1/rgw_datacache/rgw1/:/mnt/nvme0n1/rgw_datacache/
rgw1/"
                "-v"
                "/mnt/nvme0n1/rgw_datacache/rgw2/:/mnt/nvme0n1/rgw_datacache/
rgw2/"
```

5. Redeploy the Ceph Object Gateway service by using the **ceph orch apply** command.

For example,

```
[ceph: root@host01 /]# ceph orch apply -i rgw-spec.yml
```

Configuring D3N

You can configure the D3N data cache on an existing Ceph Object Gateway to improve the performance of big-data jobs running in Red Hat Ceph Storage clusters.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Ceph Object Gateway installed.
- Root-level access to the admin node.
- A fast NVMe drive in each RGW node to serve as the local cache storage.

For more information about using high-availability, see Ceph subsystems default logging level values and Understanding Ceph logs .
To enable D3N on an existing Ceph Object Gateway, use the following command to set the D3N-related configuration:

```
ceph config set CLIENT.RGW CONFIGURATION_OPTION VALUE
```

rgw_d3n_l1_local_datacache_enabled=true

```
ceph config set CLIENT.RGW rgw_d3n_l1_local_datacache_enabled TRUE/FALSE
```

For example,

```
[ceph: root@host01 /]# ceph config set client.rgw.host01
rgw_d3n_l1_local_datacache_enabled true
```

rgw_d3n_l1_datacache_persistent_path=*PATH_TO_CACHE_DIRECTORY*

```
ceph config set CLIENT.RGW rgw_d3n_l1_datacache_persistent_path PATH_TO_CACHE_DIRECTORY
```

For example,

```
[ceph: root@host01 /]# ceph config set client.rgw.host01
rgw_d3n_l1_datacache_persistent_path /mnt/nvme/rgw_datacache/
```

rgw_d3n_l1_datacache_size=*max_size_of_cache_in_bytes*

```
ceph config set CLIENT.RGW rgw_d3n_l1_datacache_size VALUE
```

For example,

```
[ceph: root@host01 /]# ceph config set client.rgw.host01 rgw_d3n_l1_datacache_size
10737418240
```

1. Create a test object.

   Note: The test object needs to be larger than 4 MB to cache.

   ```
   [ceph: root@host01 /]# fallocate -l 1G ./1G.dat
   [ceph: root@host01 /]# s3cmd mb s3://bkt
   [ceph: root@host01 /]# s3cmd put ./1G.dat s3://bkt
   ```

2. Run GET of an object.

```
[ceph: root@host01 /]# s3cmd get s3://bkt/1G.dat /dev/shm/1G_get.dat
download: 's3://bkt/1G.dat' -> './1G_get.dat'  [1 of 1]
1073741824 of 1073741824   100% in    13s    73.94 MB/s  done
```

3. Verify cache creation. Cache is created with the name consisting of object `key-name` within a configured *`rgw_d3n_l1_datacache_persistent_path`*.

```
[ceph: root@host01 /]# ls -lh /mnt/nvme/rgw_datacache

rw-rr. 1 ceph ceph 1.0M Jun  2 06:18
cc7f967c-0021-43b2-9fdf-23858e868663.615391.1_shadow.ZCiCtMWeu_19wb100JIEZ-o4tv2IyA_1
```

After the cache is created for an object, the next `GET` operation for that object will access from cache resulting in faster access.
After the cache is created, you can see the cache acceleration by writing to the RAM drive, in this example `/dev/shm`.

```
[ceph: root@host01 /]# s3cmd get s3://bkt/1G.dat /dev/shm/1G_get.dat
download: 's3://bkt/1G.dat' -> './1G_get.dat'  [1 of 1]
1073741824 of 1073741824   100% in    6s   155.07 MB/s  done
```

Multi-site Ceph Object Gateway command line usage

As a storage administrator, you can have a good understanding of how to use the Ceph Object Gateway in a multi-site environment. You can learn how to better manage the realms, zone groups, and zones in a multi-site environment.

Realms

A realm represents a globally unique namespace consisting of one or more zonegroups containing one or more zones, and zones containing buckets, which in turn contain objects. A realm enables the Ceph Object Gateway to support multiple namespaces and their configuration on the same hardware.

A realm contains the notion of periods. Each period represents the state of the zone group and zone configuration in time. Each time you make a change to a zonegroup or zone, update the period and commit it.

> **Important:** Create realms for new clusters.

Creating a realm

Use the **`realm create`** command to create a realm.

Make one realm in the list of realms be the default realm. If there is only one realm and it wasn't specified as the default realm when it was created, make it the default realm.

- Create a realm.

```
radosgw-admin realm create --rgw-realm=REALM_NAME
```

For example,

```
[ceph: root@host01 /]# radosgw-admin realm create --rgw-realm=test_realm
```

In this example, `test_realm` becomes the default realm.

> **Important:** Do not use the realm with the **`--default`** flag if the data and metadata are stored in the `default.rgw.data` and `default.rgw.index` pools. If a new realm is set as the default and these pools contain important data, the `radosgw-admin` utility can fail to manage this data properly.
> Only use the **`--default`** flag if necessary to specify the realm as the default and if you do not need the existing data or metadata in the `default.rgw` pools. If the existing data or metadata is needed, either migrate the default configuration to a multi-site or realm setup or avoid setting the new realm

as the default. For more information about migrating to a multi-site, see "Migrating a single site system to multi-site" on page 75. By specifying **--default**, the realm is called implicitly with each `radosgw-admin` call unless `--rgw-realm` and the realm name are explicitly provided.

- Optional: Change the default realm.

```
radosgw-admin realm default --rgw-realm=REALM_NAME
```

For example,

```
[ceph: root@host01 /]# radosgw-admin realm default --rgw-realm=test_realm1
```

Deleting a realm

Use this information to delete a Ceph Object Gateway realm.

1. Delete a realm, by using the **realm rm** command.

```
radosgw-admin realm rm --rgw-realm=REALM_NAME
```

For example,

```
[ceph: root@host01 /]# radosgw-admin realm rm --rgw-realm=test_realm
```

Important: If the deleted realm was configured as the default realm, continue to step "2" on page 60. If the deleted realm was not configured as default, no further steps are required.

2. Remove the default realm ID from the `default_info` in the realm list.
This step is required for realms that were configured as the default.

```
radosgw-admin realm default rm
```

For example,

```
[ceph: root@host01 /]# radosgw-admin realm default rm
```

AFTER COMPLETING THE TASK

Verify that the realm and the realm ID are no longer listed.

```
radosgw-admin realm list
```

Getting a realm

Use the **realm get** command to get a realm.

Syntax

```
radosgw-admin realm get --rgw-realm=REALM_NAME
```

Example

```
[ceph: root@host01 /]# radosgw-admin realm get --rgw-realm=test_realm >filename.json
```

The CLI will echo a JSON object with the realm properties.

```
{
    "id": "0a68d52e-a19c-4e8e-b012-a8f831cb3ebc",
    "name": "test_realm",
    "current_period": "b0c5bbef-4337-4edd-8184-5aeab2ec413b",
    "epoch": 1
}
```

Use > and an output file name to output the JSON object to a file.

Listing realms

Use the **`realm list`** command to list realms.

Example

```
[ceph: root@host01 /]# radosgw-admin realm list
```

Setting a realm

Set a specific realm with an input file name.

To set a realm, run the **`realm set`** command, specify the realm name, and `--infile=` with an input file name.

Syntax

```
radosgw-admin realm set --rgw-realm=REALM_NAME --infile=IN_FILENAME
```

Example

```
[ceph: root@host01 /]# radosgw-admin realm set --rgw-realm=test_realm --
infile=filename.json
```

Listing realm periods

Use the **`realm list-periods`** command to list realm periods.

Example

```
[ceph: root@host01 /]# radosgw-admin realm list-periods
```

Pulling a realm

Pull a realm from the node containing the master zone group and master zone to a node containing a secondary zone group or zone.

To pull a realm from the node containing the master zone group and master zone to a node containing a secondary zone group or zone, run the **`realm pull`** command on the node that will receive the realm configuration.

Syntax

```
radosgw-admin realm pull --url=URL_TO_MASTER_ZONE_GATEWAY--access-key=ACCESS_KEY --
secret=SECRET_KEY
```

Renaming a realm

Renaming a realm is applied locally, as a realm is not part of the period. When renaming a realm with multiple zones, run the command on each zone.

Renaming a realm does not get pulled with the **`realm pull`** command.

1. Rename the realm.

   ```
   radosgw-admin realm rename --rgw-realm=REALM_NAME --realm-new-name=NEW_REALM_NAME
   ```

> **Note:** Do NOT use **`realm set`** command to change the **name** parameter. That changes the internal name only. Specifying **`--rgw-realm`** would still use the old realm name.

For example,

```
[ceph: root@host01 /]# radosgw-admin realm rename --rgw-realm=test_realm --realm-new-name=test_realm2
```

2. Commit the changes.

```
radosgw-admin period update --commit
```

For example,

```
[ceph: root@host01 /]# radosgw-admin period update --commit
```

Zone groups

Zone groups define the geographic location of one or more Ceph Object Gateway instances within one or more zones.

The Ceph Object Gateway supports multi-site deployments and a global namespace by using the notion of zone groups.

> **Note:** Zone groups were previously referred to as regions.

Configuring zone groups differs from typical configuration procedures, because not all of the settings end up in a Ceph configuration file. You can list zone groups, get a zone group configuration, and set a zone group configuration.

> **Note:** The `radosgw-admin zonegroup` operations can be performed on any node within the realm, because the step of updating the period propagates the changes throughout the cluster. However, `radosgw-admin zone` operations **MUST** be performed on a host within the zone.

Creating a zone group

Use the **`zonegroup create`** command to create a zone group. A default zone group must always exist.

Make one zone group in the list of zone groups be the default zone group. If there is only one zone group and it wasn't specified as the default zone group when it was created, make it the default zone group.

> **Note:** Only one default realm can exist at a time.

Creating a zone group consists of specifying the zone group name. Creating a zone assumes that it is created in the default realm unless `--rgw-realm=`*`REALM_NAME`* is specified.

Specify whether the zone group is the primary zone group, by using the `--master` flag.

> **Important:** Do not use the zone group with the **`--default`** flag if the data and metadata are stored in the `default.rgw.data` and `default.rgw.index` pools. If a new zone group is set as the default and these pools contain important data, the `radosgw-admin` utility can fail to manage this data properly.
> Only use the **`--default`** flag if necessary to specify the zone group as the default and if you do not need the existing data or metadata in the `default.rgw` pools. If the existing data or metadata is needed, either migrate the default configuration to a multi-site or zone group setup or avoid setting the new zone group

as the default. For more information about migrating to a multi-site, see "Migrating a single site system to multi-site" on page 75. By specifying **--default**, the zone group is called implicitly with each `radosgw-admin` call unless `--rgw-zonegroup` and the zone group name are explicitly provided.

1. Create a zone group.

   ```
   radosgw-admin zonegroup create --rgw-zonegroup=ZONE_GROUP_NAME [--rgw-realm=REALM_NAME] [--master]
   ```

   For example,

   ```
   [ceph: root@host01 /]# radosgw-admin zonegroup create --rgw-zonegroup=zonegroup1 --rgw-realm=test_realm --default
   ```

   In this example, `zonegroup1` becomes the default zone group.

   Note: When the zone group is default, the command line assumes `--rgw-zonegroup=`*ZONE_GROUP_NAME* as an argument.

2. Optional: Change a zone group setting.

   ```
   radosgw-admin zonegroup modify --rgw-zonegroup=ZONE_GROUP_NAME
   ```

   For example,

   ```
   [ceph: root@host01 /]# radosgw-admin zonegroup modify --rgw-zonegroup=zonegroup1
   ```

3. Optional: Change the default zone group.

   ```
   radosgw-admin zonegroup default --rgw-zonegroup=ZONE_GROUP_NAME
   ```

   For example,

   ```
   [ceph: root@host01 /]# radosgw-admin zonegroup default --rgw-zonegroup=zonegroup2
   ```

4. Commit the change.

   ```
   radosgw-admin period update --commit
   ```

   For example,

   ```
   [ceph: root@host01 /]# radosgw-admin period update --commit
   ```

Renaming a zone group

Use the **zonegroup rename** command to rename a zone group.

To rename a zone group, run the following command:

Syntax

```
radosgw-admin zonegroup rename --rgw-zonegroup=ZONE_GROUP_NAME --zonegroup-new-name=NEW_ZONE_GROUP_NAME
```

Then, update the period:

Example

```
[ceph: root@host01 /]# radosgw-admin period update --commit
```

Deleting a zone group

Use the **zonegroup delete** command to delete a zone group.

To delete a zone group, run the following command:

Syntax

```
radosgw-admin zonegroup delete --rgw-zonegroup=ZONE_GROUP_NAME
```

Then, update the period:

Example

```
[ceph: root@host01 /]# radosgw-admin period update --commit
```

Listing zone groups

Use the **zonegroup list** command to list zone groups within a Ceph cluster.

To list the zone groups, run the following command:

```
[ceph: root@host01 /]# radosgw-admin zonegroup list
```

The `radosgw-admin` returns a JSON formatted list of zone groups.

```
{
    "default_info": "90b28698-e7c3-462c-a42d-4aa780d24eda",
    "zonegroups": [
        "us"
    ]
}
```

Getting a zone group

Use the **zonegroup get** command to retrieve a zone group configuration.

To view the configuration of a zone group, run the following command:

Syntax

```
radosgw-admin zonegroup get [--rgw-zonegroup=ZONE_GROUP_NAME]
```

Example of a zone group configuration:

```
{
    "id": "90b28698-e7c3-462c-a42d-4aa780d24eda",
    "name": "us",
    "api_name": "us",
    "is_master": "true",
    "endpoints": [
        "http://rgw1:80"
    ],
    "hostnames": [],
    "hostnames_s3website": [],
    "master_zone": "9248cab2-afe7-43d8-a661-a40bf316665e",
    "zones": [
        {
            "id": "9248cab2-afe7-43d8-a661-a40bf316665e",
            "name": "us-east",
            "endpoints": [
                "http://rgw1"
            ],
            "log_meta": "true",
            "log_data": "true",
            "bucket_index_max_shards": 11,
            "read_only": "false"
        },
        {
            "id": "d1024e59-7d28-49d1-8222-af101965a939",
```

```
            "name": "us-west",
            "endpoints": [
                "http://rgw2:80"
            ],
            "log_meta": "false",
            "log_data": "true",
            "bucket_index_max_shards": 11,
            "read_only": "false"
        }
    ],
    "placement_targets": [
        {
            "name": "default-placement",
            "tags": []
        }
    ],
    "default_placement": "default-placement",
    "realm_id": "ae031368-8715-4e27-9a99-0c9468852cfe"
}
```

Setting a zone group map

Setting a zone group map consists of creating a JSON object consisting of one or more zone groups, and setting the `master_zonegroup` for the cluster. Each zone group in the zone group map consists of a key/value pair, where the `key` setting is equivalent to the `name` setting for an individual zone group configuration, and the `val` is a JSON object consisting of an individual zone group configuration.

You may only have one zone group with `is_master` equal to `true`, and it must be specified as the `master_zonegroup` at the end of the zone group map. The following JSON object is an example of a default zone group map.

```
{
    "zonegroups": [
        {
            "key": "90b28698-e7c3-462c-a42d-4aa780d24eda",
            "val": {
                "id": "90b28698-e7c3-462c-a42d-4aa780d24eda",
                "name": "us",
                "api_name": "us",
                "is_master": "true",
                "endpoints": [
                    "http://rgw1:80"
                ],
                "hostnames": [],
                "hostnames_s3website": [],
                "master_zone": "9248cab2-afe7-43d8-a661-a40bf316665e",
                "zones": [
                    {
                        "id": "9248cab2-afe7-43d8-a661-a40bf316665e",
                        "name": "us-east",
                        "endpoints": [
                            "http://rgw1"
                        ],
                        "log_meta": "true",
                        "log_data": "true",
                        "bucket_index_max_shards": 11,
                        "read_only": "false"
                    },
                    {
                        "id": "d1024e59-7d28-49d1-8222-af101965a939",
                        "name": "us-west",
                        "endpoints": [
                            "http://rgw2:80"
                        ],
                        "log_meta": "false",
                        "log_data": "true",
                        "bucket_index_max_shards": 11,
                        "read_only": "false"
                    }
                ],
                "placement_targets": [
                    {
                        "name": "default-placement",
                        "tags": []
                    }
                ],
```

```
                "default_placement": "default-placement",
                "realm_id": "ae031368-8715-4e27-9a99-0c9468852cfe"
            }
        }
    ],
    "master_zonegroup": "90b28698-e7c3-462c-a42d-4aa780d24eda",
    "bucket_quota": {
        "enabled": false,
        "max_size_kb": -1,
        "max_objects": -1
    },
    "user_quota": {
        "enabled": false,
        "max_size_kb": -1,
        "max_objects": -1
    }
}
```

To set a zone group map, run the following command:

Example

```
[ceph: root@host01 /]# radosgw-admin zonegroup-map set --infile zonegroupmap.json
```

Where `zonegroupmap.json` is the JSON file you created. Ensure that you have zones created for the ones specified in the zone group map. Finally, update the period.

Example

```
[ceph: root@host01 /]#  radosgw-admin period update --commit
```

Setting a zone group

Set a zone group by creating a JSON object and specifying minimally the required settings.

1. `name`: The name of the zone group. Required.
2. `api_name`: The API name for the zone group. Optional.
3. `is_master`: Determines if the zone group is the master zone group. Required.

   > **Note:** You can only have one master zone group.

4. `endpoints`: A list of all the endpoints in the zone group. For example, you may use multiple domain names to refer to the same zone group. Remember to escape the forward slashes (`/`). You may also specify a port (`fqdn:port`) for each endpoint. Optional.
5. `hostnames`: A list of all the hostnames in the zone group. For example, you may use multiple domain names to refer to the same zone group. Optional. The `rgw dns name` setting will automatically be included in this list. You should restart the gateway daemon(s) after changing this setting.
6. `master_zone`: The master zone for the zone group. Optional. Uses the default zone if not specified.

   > **Note:** You can only have one master zone per zone group.

7. `zones`: A list of all zones within the zone group. Each zone has a name (required), a list of endpoints (optional), and whether or not the gateway will log metadata and data operations (false by default).
8. `placement_targets`: A list of placement targets (optional). Each placement target contains a name (required) for the placement target and a list of tags (optional) so that only users with the tag can use the placement target (i.e., the user's `placement_tags` field in the user info).
9. `default_placement`: The default placement target for the object index and object data. Set to `default-placement` by default. You may also set a per-user default placement in the user info for each user.

To set a zone group, create a JSON object consisting of the required fields, save the object to a file, for example, `zonegroup.json`; then, run the following command:

Example

```
[ceph: root@host01 /]# radosgw-admin zonegroup set --infile zonegroup.json
```

Where `zonegroup.json` is the JSON file you created.

> **Important:** The `default` zone group `is_master` setting is `true` by default. If you create a new zone group and want to make it the master zone group, you must either set the `default` zone group `is_master` setting to `false`, or delete the `default` zone group.

Finally, update the period:

Example

```
[ceph: root@host01 /]# radosgw-admin period update --commit
```

Zones

Ceph Object Gateway supports the notion of zones. A zone defines a logical group consisting of one or more Ceph Object Gateway instances.

Configuring zones differs from typical configuration procedures, because not all of the settings end up in a Ceph configuration file. You can list zones, get a zone configuration, and set a zone configuration.

> **Important:** All `radosgw-admin zone` operations **MUST** be issued on a host that operates or will operate within the zone.

Creating a zone

Create a zone by specifying a zone name.

If it is a master zone, specify the `--master` option. Only one zone in a zone group can be a master zone. To add the zone to a zone group, specify the `--rgw-zonegroup` option with the zone group name.

> **Important:** Zones must be created on a Ceph Object Gateway node that will be within the zone.

> **Important:** Do not use the zone with the **`--default`** flag if the data and metadata are stored in the `default.rgw.data` and `default.rgw.index` pools. If a new zone is set as the default and these pools contain important data, the `radosgw-admin` utility can fail to manage this data properly.
> Only use the **`--default`** flag if necessary to specify the zone as the default and if you do not need the existing data or metadata in the `default.rgw` pools. If the existing data or metadata is needed, either migrate the default configuration to a multi-site or zone setup or avoid setting the new zone as the default. For more information about migrating to a multi-site, see "Migrating a single site system to multi-site" on page 75. By specifying **`--default`**, the zone is called implicitly with each `radosgw-admin` call unless `--rgw-zone` and the zone name are explicitly provided.

1. Create the zone.

```
radosgw-admin zone create --rgw-zone=ZONE_NAME
                [--zonegroup=ZONE_GROUP_NAME]
                [--endpoints=ENDPOINT_PORT [,<endpoint:port>]
```

```
                [--master] [--default]
                --access-key ACCESS_KEY --secret SECRET_KEY
```

2. Commit the change.

```
radosgw-admin period update --commit
```

For example,

```
[ceph: root@host01 /]# radosgw-admin period update --commit
```

Deleting a zone

Learn how to delete a zone.

To delete a zone, start by removing it from the zonegroup.

1. Remove the zone from the zonegroup.

> **Important:** Do not delete a zone without removing it from a zone group first. Otherwise, updating the period fails.

```
radosgw-admin zonegroup remove --rgw-zonegroup=ZONE_GROUP_NAME
                               --rgw-zone=ZONE_NAME
```

2. Update the period.

```
radosgw-admin period update --commit
```

For example,

```
[ceph: root@host01 /]# radosgw-admin period update --commit
```

3. Delete the zone.

> **Important:** This procedure must be used on a host within the zone.

```
radosgw-admin zone delete --rgw-zone=ZONE_NAME
```

4. Update the period.

```
radosgw-admin period update --commit
```

For example,

```
[ceph: root@host01 /]# radosgw-admin period update --commit
```

5. Delete the pools.
If the pools for the deleted zone will not be used anywhere else, consider deleting the pools. Replace *DELETED_ZONE_NAME* in the following example with the deleted zone's name.

> **Important:**
> - Once Ceph deletes the zone pools, it deletes all of the data within them in an unrecoverable manner. Only delete the zone pools if Ceph clients no longer need the pool contents.

- In a multi-realm cluster, deleting the `.rgw.root` pool along with the zone pools removes ALL the realm information for the cluster. Ensure that `.rgw.root` does not contain other active realms before deleting the `.rgw.root` pool.

```
ceph osd pool delete DELETED_ZONE_NAME.rgw.control DELETED_ZONE_NAME.rgw.control --
yes-i-really-really-mean-it
ceph osd pool delete DELETED_ZONE_NAME.rgw.data.root DELETED_ZONE_NAME.rgw.data.root
--yes-i-really-really-mean-it
ceph osd pool delete DELETED_ZONE_NAME.rgw.log DELETED_ZONE_NAME.rgw.log --yes-i-
really-really-mean-it
ceph osd pool delete DELETED_ZONE_NAME.rgw.users.uid DELETED_ZONE_NAME.rgw.users.uid
--yes-i-really-really-mean-it
```

6. Restart the Ceph Object Gateway process.

Modifying a zone

To modify a zone, specify the zone name and the parameters that you want to modify.

Important: Zones should be modified on a Ceph Object Gateway node that will be within the zone.

Syntax

```
radosgw-admin zone modify [options]

`--access-key=<key>`
`--secret/--secret-key=<key>`
`--master`
`--default`
`--endpoints=<list>`
```

Then, update the period:

Example

```
[ceph: root@host01 /]# radosgw-admin period update --commit
```

Listing zones

Use the **zone list** command to list zones in a cluster.

As `root`, to list the zones in a cluster, run the following command:

Example

```
[ceph: root@host01 /]# radosgw-admin zone list
```

Getting a zone

Use the **zone get** command to retrieve a zone configuration.

As `root`, to get the configuration of a zone, run the following command:

Syntax

```
radosgw-admin zone get [--rgw-zone=ZONE_NAME]
```

The `default` zone looks like this:

```
{ "domain_root": ".rgw",
  "control_pool": ".rgw.control",
  "gc_pool": ".rgw.gc",
  "log_pool": ".log",
  "intent_log_pool": ".intent-log",
  "usage_log_pool": ".usage",
  "user_keys_pool": ".users",
  "user_email_pool": ".users.email",
  "user_swift_pool": ".users.swift",
  "user_uid_pool": ".users.uid",
  "system_key": { "access_key": "", "secret_key": ""},
  "placement_pools": [
     {  "key": "default-placement",
        "val": { "index_pool": ".rgw.buckets.index",
                 "data_pool": ".rgw.buckets"}
     }
   ]
  }
```

Setting a zone

Configuring a zone involves specifying a series of Ceph Object Gateway pools.

For consistency, use a pool prefix that is the same as the zone name. See the Pools for details on configuring pools.

> **Important:** Zones should be set on a Ceph Object Gateway node that will be within the zone.

To set a zone, create a JSON object consisting of the pools, save the object to a file, for example, `zone.json`; then, run the following command, replacing `ZONE_NAME` with the name of the zone:

Example

```
[ceph: root@host01 /]# radosgw-admin zone set --rgw-zone=test-zone --infile zone.json
```

Where `zone.json` is the JSON file you created.

Then, as `root`, update the period:

Example

```
[ceph: root@host01 /]# radosgw-admin period update --commit
```

Renaming a zone

use the **zone reanme** command to rename a zone on a host.

To rename a zone, specify the zone name and the new zone name. Issue the following command on a host within the zone:

Syntax

```
radosgw-admin zone rename --rgw-zone=ZONE_NAME --zone-new-name=NEW_ZONE_NAME
```

Then, update the period:

Example

```
[ceph: root@host01 /]# radosgw-admin period update --commit
```

Adding a zone to a zone group

Use the **zonegroup add** command to add a zone to a zone group.

This command must be run on a host that will be in the zone. To add a zone to a zonegroup, run the following command:

Syntax

```
radosgw-admin zonegroup add --rgw-zonegroup=ZONE_GROUP_NAME --rgw-zone=ZONE_NAME
```

Then, update the period:

Example

```
[ceph: root@host01 /]# radosgw-admin period update --commit
```

Removing a zone from a zone group

Use the **`zonegroup remove`** command to remove a zone from a zone group.

To remove a zone from a zone group, run the following command:

Syntax

```
radosgw-admin zonegroup remove --rgw-zonegroup=ZONE_GROUP_NAME --rgw-zone=ZONE_NAME
```

Then, update the period:

Example

```
[ceph: root@host01 /]# radosgw-admin period update --commit
```

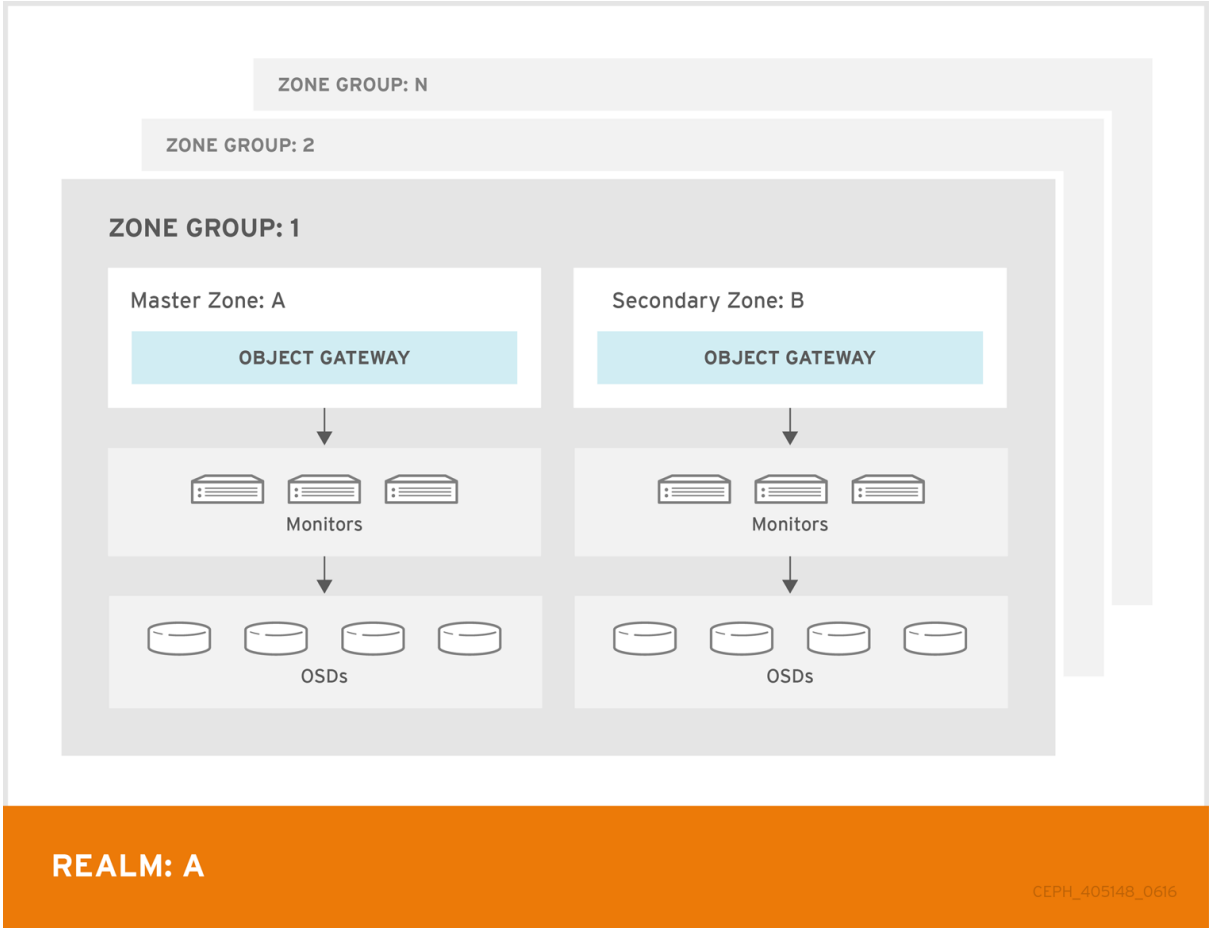
Multi-site configuration and administration

As a storage administrator, you can configure and administer multiple Ceph Object Gateways for a variety of use cases. You can learn what to do during a disaster recovery and failover events. Also, you can learn more about realms, zones, and syncing policies in multi-site Ceph Object Gateway environments.

A single zone configuration typically consists of one zone group containing one zone and one or more `ceph-radosgw` instances where you may load-balance gateway client requests between the instances. In a single zone configuration, typically multiple gateway instances point to a single Ceph storage cluster. However, Red Hat supports several multi-site configuration options for the Ceph Object Gateway:

- **Multi-zone:** A more advanced configuration consists of one zone group and multiple zones, each zone with one or more `ceph-radosgw` instances. Each zone is backed by its own Ceph Storage Cluster. Multiple zones in a zone group provides disaster recovery for the zone group should one of the zones experience a significant failure. Each zone is active and may receive write operations. In addition to disaster recovery, multiple active zones may also serve as a foundation for content delivery networks.
- **Multi-zone-group:** Formerly called 'regions', the Ceph Object Gateway can also support multiple zone groups, each zone group with one or more zones. Objects stored to zone groups within the same realm share a global namespace, ensuring unique object IDs across zone groups and zones.
- **Multiple Realms:** The Ceph Object Gateway supports the notion of realms, which can be a single zone group or multiple zone groups and a globally unique namespace for the realm. Multiple realms provides the ability to support numerous configurations and namespaces.

Figure: Ceph Object Gateway realm



Prerequisites

- A healthy running Red Hat Ceph Storage cluster.
- Deployment of the Ceph Object Gateway software (see “Deployment” on page 23).

Requirements and assumptions

A multi-site configuration requires at least two Ceph storage clusters, and at least two Ceph Object Gateway instances, one for each Ceph storage cluster. The information in this documentation assumes at least two Ceph storage clusters in geographically separate locations; however, the configuration can work on the same physical site. This information also assumes four Ceph Object Gateway servers named `rgw1`, `rgw2`, `rgw3`, and `rgw4` respectively.

A multi-site configuration requires a master zone group and a master zone. Also, each zone group requires a master zone. Zone groups might have one or more secondary or non-master zones.

Important: When planning network considerations for multi-site, it is important to understand the relation bandwidth and latency that is observed on the multi-site synchronization network and the clients ingest rate in direct correlation with the current sync state of the objects owed to the secondary site. Multi-site synchronization is asynchronous and one of the limitations is the rate at which the sync gateways can process data across the link. An example to look at in terms of network inter-connectivity speed might be 1 GbE or inter-datacenter connectivity, for every 8 TB or cumulative receive data, per client gateway. Thus, if you replicate to two other sites, and ingest 16 TB a day, you need 6 GbE of dedicated bandwidth for multi-site replication.

Red Hat also recommends private Ethernet or Dense wavelength-division multiplexing (DWDM) as a VPN over the internet is not ideal due to the additional overhead incurred.

Important: The master zone within the master zone group of a realm is responsible for storing the master copy of the realm's metadata, including users, quotas, and buckets (created by the `radosgw-admin` CLI). This metadata gets synchronized to secondary zones and secondary zone groups automatically. Metadata operations run with the `radosgw-admin` CLI **MUST be run** on a host within the master zone of the master zone group in order to ensure that they get synchronized to the secondary zone groups and zones. Currently, it is *possible* to run metadata operations on secondary zones and zone groups, but it is **NOT recommended** because they **WILL NOT** be synchronized, leading to fragmented metadata.

Note: For new Ceph Object Gateway deployment in multi-site, it takes around 20 minutes to sync metadata operations to the secondary site.

In the following examples, the `rgw1` host serves as the master zone of the master zone group; the `rgw2` host serves as the secondary zone of the master zone group; the `rgw3` host serves as the master zone of the secondary zone group; and the `rgw4` host serves as the secondary zone of the secondary zone group.

Important: Red Hat recommends provisioning 3 Ceph Object Gateway daemons that are dedicated to sync processing in a multi-site deployment. Prevent the other nonsyncing Ceph Object Gateway daemons in the multi-site configuration that are to be dedicated to client I/O operations through an ingress service or other load balancer from running sync operations. Prevent the sync operations, by running the following command:

```
ceph config set client.rgw.CLIENT_NODE rgw_run_sync_thread false
```

After running the command, restart those client-facing Ceph Object Gateway daemons. These **`ceph config`** commands must substitute your actual host or service names for *`CLIENT_NODE`*. This should be backported to previous releases as well, as this is purely improvement to language and clarity.

The following is a typical example configuration file for HAProxy for syncing gateways:

```
[root@host01 ~]# cat ./haproxy.cfg

global

  log         127.0.0.1 local2

  chroot      /var/lib/haproxy
  pidfile     /var/run/haproxy.pid
  maxconn     7000
  user        haproxy
  group       haproxy
  daemon

  stats socket /var/lib/haproxy/stats

defaults
  mode                    http
  log                     global
  option                  httplog
  option                  dontlognull
  option http-server-close
  option forwardfor       except 127.0.0.0/8
  option                  redispatch
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          30s
  timeout server          30s
  timeout http-keep-alive 10s
  timeout check           10s
        timeout client-fin 1s
        timeout server-fin 1s
  maxconn                 6000


listen  stats
  bind    0.0.0.0:1936
      mode            http
      log             global

      maxconn 256


      clitimeout      10m
      srvtimeout      10m
      contimeout      10m
      timeout queue   10m

# JTH start
      stats enable
      stats hide-version
      stats refresh 30s
      stats show-node
##        stats auth admin:password
      stats uri  /haproxy?stats
      stats admin if TRUE


frontend  main
  bind     *:5000
  acl url_static       path_beg       -i /static /images /javascript /stylesheets
  acl url_static       path_end       -i .jpg .gif .png .css .js

  use_backend static          if url_static
  default_backend             app
        maxconn 6000

backend static
  balance     roundrobin
    fullconn 6000
    server  app8 host01:8080 check maxconn 2000
    server  app9 host02:8080 check maxconn 2000
    server  app10 host03:8080 check maxconn 2000


backend app
  balance     roundrobin
    fullconn 6000
    server  app8 host01:8080 check maxconn 2000
    server  app9 host02:8080 check maxconn 2000
    server  app10 host03:8080 check maxconn 2000
```

Pools

Calculate a suitable number of placement groups for the pools the `radosgw` daemon will create. Set the calculated values as defaults in the Ceph configuration database.

Use the *Ceph Placement Group's per Pool Calculator* to calculate a suitable number of placement groups for the pools the `radosgw` daemon will create.

For example,

```
[ceph: root@host01 /]#  ceph config set osd osd_pool_default_pg_num = 32
[ceph: root@host01 /]#  ceph config set osd osd_pool_default_pgp_num = 32
```

Note: Making this change to the Ceph configuration will use those defaults when the Ceph Object Gateway instance creates the pools. Alternatively, you can create the pools manually.

Pool names particular to a zone follow the naming convention `ZONE_NAME.POOL_NAME`. For example, a zone named `us-east` will have the following pools:

- `.rgw.root`
- `us-east.rgw.control`
- `us-east.rgw.meta`
- `us-east.rgw.log`
- `us-east.rgw.buckets.index`
- `us-east.rgw.buckets.data`
- `us-east.rgw.buckets.non-ec`
- `us-east.rgw.meta:users.keys`
- `us-east.rgw.meta:users.email`
- `us-east.rgw.meta:users.swift`
- `us-east.rgw.meta:users.uid`

Reference

For more information, see the following:

- *Pools*

Migrating a single site system to multi-site

You can migrate from a single site system with a `default` zone group and zone to a multi-site system.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- A Ceph Object Gateway installed.

1. Create a realm.

   ```
   radosgw-admin realm create --rgw-realm REALM_NAME --default
   ```

   Replace *REALM_NAME* with the realm name.

2. Rename the default zone and zone group.

```
radosgw-admin zonegroup rename --rgw-zonegroup default --zonegroup-new-
name NEW_ZONE_GROUP_NAME
radosgw-admin zone rename --rgw-zone default --zone-new-name NEW_ZONE_NAME --rgw-
zonegroup NEW_ZONE_GROUP_NAME
```

Replace *NEW_ZONE_GROUP_NAME* and *NEW_ZONE_NAME* with the zone group and zone name, respectively.

3. Rename the default zone group's `api_name`.

```
radosgw-admin zonegroup modify --api-name NEW_ZONE_GROUP_NAME --rgw-
zonegroup NEW_ZONE_GROUP_NAME
```

Replace *NEW_ZONE_GROUP_NAME* with the zone group name.

The `api_name` field in the zone group map refers to the name of the RADOS API used for data replication across different zones. This field helps clients interact with the correct APIs for accessing and managing data within the Ceph storage cluster.

4. Configure the primary zone group.

```
radosgw-admin zonegroup modify --rgw-realm REALM_NAME --rgw-
zonegroup NEW_ZONE_GROUP_NAME --endpoints http://ENDPOINT --master --default
```

Replace the following:

- *REALM_NAME* with realm name.
- *NEW_ZONE_GROUP_NAME* with the zone group name.
- *ENDPOINT* with the fully qualified domain names in the zone group.

5. Configure the primary zone.

```
radosgw-admin zone modify --rgw-realm REALM_NAME --rgw-
zonegroup NEW_ZONE_GROUP_NAME --rgw-zone NEW_ZONE_NAME --endpoints http://ENDPOINT --
master --default
```

Replace the following:

- *REALM_NAME* with realm name.
- *NEW_ZONE_GROUP_NAME* with the zone group name.
- *NEW_ZONE_NAME* with the zone name.
- *ENDPOINT* with the fully qualified domain names in the zone group.

6. Create a system user.

```
radosgw-admin user create --uid USER_ID --display-name DISPLAY_NAME --access-
key ACCESS_KEY --secret SECRET_KEY --system
```

Replace the following:

- *USER_ID* with the username.
- *DISPLAY_NAME* with the display name. The display name can contain spaces.

7. Commit the updated configuration.

```
radosgw-admin period update --commit
```

For example,

```
[ceph: root@host01 /]# radosgw-admin period update --commit
```

8. Restart the Ceph Object Gateway.

```
systemctl restart ceph-radosgw@rgw.`hostname -s`
ceph orch restart RGW_SERVICE_NAME
```

For example,

```
[ceph: root@host01 /]# systemctl restart ceph-radosgw@rgw.`hostname -s`
[ceph: root@host01 /]# ceph orch restart rgw.rgwsvcid.mons-1.jwgwwp
```

Establishing a secondary zone

Zones within a zone group replicate all data to ensure that each zone has the same data. When creating the secondary zone, issue all of the `radosgw-admin zone` operations on a host identified to serve the secondary zone.

PREREQUISITE

- At least two Red Hat Ceph Storage clusters.
- At least two Ceph Object Gateway instances, one for each Red Hat Ceph Storage cluster.
- Root-level access to all the nodes.
- Nodes or containers are added to the storage cluster.
- All Ceph Manager, Monitor, and OSD daemons are deployed.

Learn how to establish a secondary zone.

Note: To add additional zones, follow the same procedures as for adding the secondary zone. Use a different zone name.

Important:

- Run the metadata operations, such as user creation and quotas, on a host within the master zone of the master zonegroup. The master zone and the secondary zone can receive bucket operations from the RESTful APIs, but the secondary zone redirects bucket operations to the master zone. If the master zone is down, bucket operations will fail. If you create a bucket using the `radosgw-admin` CLI, you must run it on a host within the master zone of the master zone group so that the buckets will synchronize with other zone groups and zones.
- Bucket creation for a particular user is not supported, even if you create a user in the secondary zone with **`--yes-i-really-mean-it`**.
- Creating a user from a non-primary site is successful when using the **`--yes-i-really-mean-it`** however the user does not sync to the primary site.

1. Log into the `cephadm` shell.
   For example,

   ```
   [root@host04 ~]# cephadm shell
   ```

2. Pull the primary realm configuration from the host.

   ```
   radosgw-admin realm pull --url=URL_TO_PRIMARY_ZONE_GATEWAY --access-key=ACCESS_KEY --
   secret-key=SECRET_KEY
   ```

   For example,

   ```
   [ceph: root@host04 /]# radosgw-admin realm pull --url=http://10.74.249.26:80 --access-
   key=LIPEYZJLTWXRKXS9LPJC --secret-key=IsAje0AVDNXNw48LjMAimpCpI7VaxJYSnfD0FFKQ
   ```

3. Pull the primary period configuration from the host.

   ```
   radosgw-admin period pull --url=URL_TO_PRIMARY_ZONE_GATEWAY --access-key=ACCESS_KEY --
   secret-key=SECRET_KEY
   ```

For exaple,

```
[ceph: root@host04 /]# radosgw-admin period pull --url=http://10.74.249.26:80 --
access-key=LIPEYZJLTWXRKXS9LPJC --secret-key=IsAje0AVDNXNw48LjMAimpCpI7VaxJYSnfD0FFKQ
```

4. Configure a secondary zone.

> **Note:** All zones run in an active-active configuration by default; that is, a gateway client might write data to any zone and the zone will replicate the data to all other zones within the zone group. If the secondary zone should not accept write operations, specify the `--read-only` flag to create an active-passive configuration between the master zone and the secondary zone. Additionally, provide the `access_key` and `secret_key` of the generated system user stored in the master zone of the master zone group.

```
radosgw-admin zone create --rgw-zonegroup=ZONE_GROUP_NAME \
             --rgw-zone=SECONDARY_ZONE_NAME --endpoints=http://
RGW_SECONDARY_HOSTNAME:RGW_PRIMARY_PORT_NUMBER_1 \
             --access-key=SYSTEM_ACCESS_KEY --secret=SYSTEM_SECRET_KEY \
             [--read-only]
```

For example,

```
[ceph: root@host04 /]# radosgw-admin zone create --rgw-zonegroup=us --rgw-zone=us-
east-2 --endpoints=http://rgw2:80 --access-key=LIPEYZJLTWXRKXS9LPJC --secret-
key=IsAje0AVDNXNw48LjMAimpCpI7VaxJYSnfD0FFKQ
```

5. Optional: Delete the default zone.

> **Important:** Do not delete the default zone and its pools if you are using the default zone and zone group to store data.

For example,

```
[ceph: root@host04 /]# radosgw-admin zone rm --rgw-zone=default
[ceph: root@host04 /]# ceph osd pool rm default.rgw.log default.rgw.log --yes-i-
really-really-mean-it
[ceph: root@host04 /]# ceph osd pool rm default.rgw.meta default.rgw.meta --yes-i-
really-really-mean-it
[ceph: root@host04 /]# ceph osd pool rm default.rgw.control default.rgw.control --yes-
i-really-really-mean-it
[ceph: root@host04 /]# ceph osd pool rm default.rgw.data.root default.rgw.data.root --
yes-i-really-really-mean-it
[ceph: root@host04 /]# ceph osd pool rm default.rgw.gc default.rgw.gc --yes-i-really-
really-mean-it
```

6. Optional: If you specified the realm and zone in the service specification during the deployment of the Ceph Object Gateway, update the `spec` section of the specification file.

```
spec:
rgw_realm: REALM_NAME
rgw_zone: ZONE_NAME
```

7. Update the Ceph configuration database.

```
ceph config set client.rgw.SERVICE_NAME rgw_realm REALM_NAME
ceph config set client.rgw.SERVICE_NAME rgw_zonegroup ZONE_GROUP_NAME
ceph config set client.rgw.SERVICE_NAME rgw_zone SECONDARY_ZONE_NAME
```

For example,

```
[ceph: root@host04 /]# ceph config set client.rgw.rgwsvcid.mons-1.jwgwwp rgw_realm
test_realm
[ceph: root@host04 /]# ceph config set client.rgw.rgwsvcid.mons-1.jwgwwp rgw_zonegroup
```

```
us
[ceph: root@host04 /]# ceph config set client.rgw.rgwsvcid.mons-1.jwgwwp rgw_zone us-
east-2
```

8. Commit the changes.

```
radosgw-admin period update --commit
```

For example,

```
[ceph: root@host04 /]# radosgw-admin period update --commit
```

9. Outside the cephadm shell, fetch the FSID of the storage cluster and the processes.
For example,

```
[root@host04 ~]#  systemctl list-units | grep ceph
```

10. Start the Ceph Object Gateway daemon.

```
systemctl start ceph-FSID@DAEMON_NAME
systemctl enable ceph-FSID@DAEMON_NAME
```

For example,

```
[root@host04 ~]# systemctl start ceph-62a081a6-88aa-11eb-
a367-001a4a000672@rgw.test_realm.us-east-2.host04.ahdtsw.service
[root@host04 ~]# systemctl enable ceph-62a081a6-88aa-11eb-
a367-001a4a000672@rgw.test_realm.us-east-2.host04.ahdtsw.service
```

Configuring the archive zone

Archive object data residing on Red Hat Ceph Storage using the Object Storage Archive Zone Feature.

PREREQUISITE

- A running Red Hat Ceph Storage cluster.
- Root-level access to a Ceph Monitor node.
- Ceph Object Gateway software is installed.

> **Note:** Ensure you have a realm before configuring a zone as an archive. Without a realm, you cannot archive data through an archive zone for default zone/zonegroups.

The archive zone uses multi-site replication and S3 object versioning feature in Ceph Object Gateway. The archive zone retains all version of all the objects available, even when deleted in the production file.
The archive zone has a history of versions of S3 objects that can be eliminated only through the gateways that are associated with the archive zone. It captures all the data updates and metadata to consolidate them as versions of S3 objects. Bucket granular replication to the archive zone can be used after creating an archive zone.

You can control the storage space usage of an archive zone through the bucket Lifecycle policies, where you can define the number of versions you would like to keep for an object.

An archive zone helps protect your data against logical or physical errors. It can save users from logical failures, such as accidentally deleting a bucket in the production zone. It can also save your data from massive hardware failures, like a complete production site failure. Additionally, it provides an immutable copy, which can help build a ransomware protection strategy.

To implement the bucket granular replication, use the sync policies commands for enabling and disabling policies. For more information, see “Creating a sync policy group” on page 96 and “Modifying a sync policy group” on page 97.

Note: Using the sync policy group procedures is optional and only necessary to use enabling and disabling with bucket granular replication. For using the archive zone without bucket granular replication, it is not necessary to use the sync policy procedures.

If you want to migrate the storage cluster from single site, see "Migrating a single site system to multi-site" on page 75.

- During new zone creation, use the `archive` tier to configure the archive zone.

  ```
  radosgw-admin zone create --rgw-zonegroup={ZONE_GROUP_NAME} --rgw-zone={ZONE_NAME} --
  endpoints={http://FQDN:PORT},{http://FQDN:PORT} --tier-type=archive
  ```

  For example,

  ```
  [ceph: root@host01 /]# radosgw-admin zone create --rgw-zonegroup=us --rgw-zone=us-east
  --endpoints={http://example.com:8080} --tier-type=archive
  ```

- From the archive zone, modify the archive zone to sync from only the primary zone and perform a period update commit.

  ```
  $ radosgw-admin zone modify --rgw-zone archive --sync_from primary --sync_from_all
  false --sync-from-rm secondary

  $ radosgw-admin period update --commit
  ```

  Note: The recommendation is to reduce the `max_objs_per_shard` to 50K to account for the omap olh entries in the archive zone. This helps in keeping the number of omap entries per bucket index shard object in check to prevent large omap warnings. For example,

  ```
  ceph config set client.rgw rgw_max_objs_per_shard 50000
  ```

Related information

Deploying a multi-site Ceph Object Gateway using the Ceph Orchestrator
S3 add an object to a bucket

Deleting objects in archive zone

Use the S3 lifecycle policy extension to delete objects within an **<ArchiveZone>** element.

Important: Archive zone objects can only be deleted using the **expiration** lifecycle policy rule.

- If any **<Rule>** section contains an **<ArchiveZone>** element, that rule runs in archive zone and is the only rule that runs in an archive zone.
- Rules marked **<ArchiveZone>** do NOT run in nonarchive zones.

The rules within the lifecycle policy determine when and what objects to delete. For more information about lifecycle creation and management, see "Configuring Bucket lifecycle" on page 135.

1. Set the **<ArchiveZone>** lifecycle policy rule.
   For more information about creating a lifecycle policy, see "Creating a lifecycle management policy" on page 135.
   For example,

```
<?xml version="1.0" ?>
<LifecycleConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
        <Rule>
                <ID>delete-1-days-az</ID>
                <Filter>
          <Prefix></Prefix>
          <ArchiveZone />
                </Filter>
                <Status>Enabled</Status>
                <Expiration>
                        <Days>1</Days>
                </Expiration>
        </Rule>
</LifecycleConfiguration>
```

2. Optional: Verify whether a specific lifecycle policy contains an archive zone rule.

```
radosgw-admin lc get --bucket BUCKET_NAME
```

The following example shows the output of a lifecycle policy with an archive zone rule.

```
[ceph: root@host01 /]# radosgw-admin lc get --bucket test-bkt

{
    "prefix_map": {
        "": {
            "status": true,
            "dm_expiration": true,
            "expiration": 0,
            "noncur_expiration": 2,
            "mp_expiration": 0,
            "transitions": {},
            "noncur_transitions": {}
        }
    },
    "rule_map": [
        {
            "id": "Rule 1",
            "rule": {
                "id": "Rule 1",
                "prefix": "",
                "status": "Enabled",
                "expiration": {
                    "days": "",
                    "date": ""
                },
                "noncur_expiration": {
                    "days": "2",
                    "date": ""
                },
                "mp_expiration": {
                    "days": "",
                    "date": ""
                },
                "filter": {
                    "prefix": "",
                    "obj_tags": {
                        "tagset": {}
                    },
                    "archivezone": ""
                },
                "transitions": {},
                "noncur_transitions": {},
                "dm_expiration": true
            }
        }
    ]
}
```

3. If the Ceph Object Gateway user is deleted, the buckets at the archive site owned by that user is inaccessible. Link those buckets to another Ceph Object Gateway user to access the data.

```
radosgw-admin bucket link --uid NEW_USER_ID --bucket BUCKET_NAME --yes-i-really-mean-
it
```

```
[ceph: root@host01 /]# radosgw-admin bucket link --uid arcuser1 --bucket arc1-deleted-
da473fbbaded232dc5d1e434675c1068 --yes-i-really-mean-it
```

Related information

S3 bucket lifecycle
Managing your storage lifecycle
Configuring Bucket lifecycle

Failover and disaster recovery

Recover your data from different failover and disaster scenarios.

PREREQUISITE

- A running Red Hat Ceph Storage cluster.
- Root-level access to a Ceph Monitor node.
- Installation of the Ceph Object Gateway software.

Use any of the following procedures, according to your needs:

- “Primary zone failure with a failover to the secondary zone for disaster recovery” on page 82
- “Switching archive zone sync from primary to secondary zone with I/Os in progress” on page 83
- “Archive zone syncs only from primary zone with primary zone failure” on page 84

Primary zone failure with a failover to the secondary zone for disaster recovery

1. Make the secondary zone the primary and default zone.

   ```
   radosgw-admin zone modify --rgw-zone=ZONE_NAME --master --default
   ```

   By default, Ceph Object Gateway runs in an active-active configuration. If the cluster was configured to run in an active-passive configuration, the secondary zone is a read-only zone. Remove the `--read-only` status to allow the zone to receive write operations.

   For example,

   ```
   [ceph: root@host01 /]# radosgw-admin zone modify --rgw-zone=ZONE_NAME --master --
   default --read-only=false
   ```

2. Update the period to make the changes take effect.

   ```
   radosgw-admin period update --commit
   ```

3. Restart the Ceph Object Gateway.

   Note: Use the output from the **`ceph orch ps`** command, under the `NAME` column, to get the *SERVICE_TYPE.ID* information.

   - Restart the Ceph Object Gateway on an individual node in the storage cluster.

     ```
     systemctl restart ceph-CLUSTER_ID@SERVICE_TYPE.ID.service
     ```

     For example,

     ```
     [root@host01 ~]# systemctl restart ceph-c4b34c6f-8365-11ba-
     dc31-529020a7702d@rgw.realm.zone.host01.gwasto.service
     ```

   - Restart the Ceph Object Gateways on all nodes in the storage cluster.

```
ceph orch restart SERVICE_TYPE
```

For example,

```
[ceph: root@host01 /]# ceph orch restart rgw
```

If the former primary zone recovers, revert the operation.

4. From the recovered zone, pull the realm from the current primary zone.

```
radosgw-admin realm pull --url=URL_TO_PRIMARY_ZONE_GATEWAY
                           --access-key=ACCESS_KEY --secret=SECRET_KEY
```

5. Make the recovered zone the primary and default zone.

```
adosgw-admin zone modify --rgw-zone=ZONE_NAME --master --default
```

6. Update the period to make the changes take effect.

```
radosgw-admin period update --commit
```

7. Restart the Ceph Object Gateway in the recovered zone.

```
ceph orch restart SERVICE_TYPE
```

For example,

```
[ceph: root@host01 /]# ceph orch restart rgw
```

8. If the secondary zone needs to be a read-only configuration, update the secondary zone.

```
radosgw-admin zone modify --rgw-zone=ZONE_NAME --read-only
radosgw-admin zone modify --rgw-zone=ZONE_NAME --read-only
```

a. Update the period to make the changes take effect.

```
radosgw-admin period update --commit
```

b. Restart the Ceph Object Gateway in the recovered zone.
For example,

```
[ceph: root@host01 /]# ceph orch restart rgw
```

Switching archive zone sync from primary to secondary zone with I/Os in progress

1. Switch the sync source to the secondary zone.

```
radosgw-admin zone modify --rgw-zone archive --sync_from secondary --sync_from_all
false --sync-from-rm primary radosgw-admin period update --commit
```

2. Verify the sync status and data consistency in the archive zone.
3. Switch the sync source back to the primary zone.

```
radosgw-admin zone modify --rgw-zone archive --sync_from primary --sync_from_all false
--sync-from-rm secondary radosgw-admin period update --commit
```

4. Verify the sync status and data consistency in the archive zone after the switch.
As a result:
   - The archive zone starts syncing from the secondary zone after the first modification.
   - Data consistency is maintained throughout the switch.

- Upon switching back to the primary zone, the archive zone resumes syncing from the primary zone without data loss or corruption.
- Sync remains consistent in all the zones.

Archive zone syncs only from primary zone with primary zone failure

1. Ensure that the archive zone is set to sync only from the primary zone.
2. Stop the primary zone gateways to simulate a primary zone failure.
3. Failover to the secondary zone and run the period update commit.
4. Observe the sync status of the archive zone.
5. Post a time interval of about 30 minutes, restart the primary zone gateways.
6. Verify that the archive zone resumes syncing from the primary zone.
   As a result:
   - The archive zone stops syncing when the primary zone fails.
   - After restoring the primary zone, the archive zone automatically resumes syncing from the primary zone.
   - Data integrity and sync status is maintained throughout the process.

3-site failover and disaster recovery

Recover your data from different failover and disaster scenarios.

PREREQUISITE

- A running Red Hat Ceph Storage cluster.
- Root-level access to a Ceph Monitor node.
- Installation of the Ceph Object Gateway software.

1. Make the either the secondary or tertiary zone the primary and default zone.

   ```
   radosgw-admin zone modify --rgw-zone=ZONE_NAME --master --default
   ```

   By default, Ceph Object Gateway runs in an active-active-active configuration. If the cluster was configured to run in an active-passive-passive configuration, the secondary and tertiary zones are read-only zones. Remove the `--read-only` status to allow the zone to receive write operations.

   For example,

   ```
   [ceph: root@host01 /]# radosgw-admin zone modify --rgw-zone=ZONE_NAME --master --
   default --read-only=false
   ```

2. Update the period to make the changes take effect.

   ```
   radosgw-admin period update --commit
   ```

3. Restart the Ceph Object Gateway.

   Note: Use the output from the **`ceph orch ps`** command, under the `NAME` column, to get the *`SERVICE_TYPE.ID`* information.

   - Restart the Ceph Object Gateway on an individual node in the storage cluster.

     ```
     systemctl restart ceph-CLUSTER_ID@SERVICE_TYPE.ID.service
     ```

     For example,

```
[root@host01 ~]# systemctl restart ceph-c4b34c6f-8365-11ba-
dc31-529020a7702d@rgw.realm.zone.host01.gwasto.service
```

- Restart the Ceph Object Gateways on all nodes in the storage cluster.

```
ceph orch restart SERVICE_TYPE
```

For example,

```
[ceph: root@host01 /]# ceph orch restart rgw
```

If the former primary zone recovers, revert the operation.

4. From the recovered zone, pull the realm from the current primary zone.

```
radosgw-admin realm pull --url=URL_TO_PRIMARY_ZONE_GATEWAY
                           --access-key=ACCESS_KEY --secret=SECRET_KEY
```

5. Make the recovered zone the primary and default zone.

```
adosgw-admin zone modify --rgw-zone=ZONE_NAME --master --default
```

6. Update the period to make the changes take effect.

```
radosgw-admin period update --commit
```

7. Restart the Ceph Object Gateway in the recovered zone.

```
ceph orch restart SERVICE_TYPE
```

For example,

```
[ceph: root@host01 /]# ceph orch restart rgw
```

8. If the secondary and tertiary zones need to be a read-only configuration, update each zone.

```
radosgw-admin zone modify --rgw-zone=ZONE_NAME --read-only
radosgw-admin zone modify --rgw-zone=ZONE_NAME --read-only
```

a. Update the period to make the changes take effect.

```
radosgw-admin period update --commit
```

b. Restart the Ceph Object Gateway in the recovered zones.
For example,

```
[ceph: root@host01 /]# ceph orch restart rgw
```

Configuring multiple realms in the same storage cluster

You can configure multiple realms in the same storage cluster. This is a more advanced use case for multi-site. Configuring multiple realms in the same storage cluster enables you to use a local realm to handle local Ceph Object Gateway client traffic, as well as a replicated realm for data that will be replicated to a secondary site.

> **Note:** Each realm should have its own Ceph Object Gateway.

Prerequisites

- Two running Red Hat Ceph Storage data centers in a storage cluster.
- The access key and secret key for each data center in the storage cluster.

- Root-level access to all the Ceph Object Gateway nodes.
- Each data center has its own local realm. They share a realm that replicates on both sites.

Procedure

1. Create one local realm on the first data center in the storage cluster:

   Syntax

   ```
   radosgw-admin realm create --rgw-realm=REALM_NAME --default
   ```

   Example

   ```
   [ceph: root@host01 /]# radosgw-admin realm create --rgw-realm=ldc1 --default
   ```

2. Create one local master zonegroup on the first data center:

   Syntax

   ```
   radosgw-admin zonegroup create --rgw-zonegroup=ZONE_GROUP_NAME --endpoints=http://
   RGW_NODE_NAME:80 --rgw-realm=REALM_NAME --master --default
   ```

   Example

   ```
   [ceph: root@host01 /]# radosgw-admin zonegroup create --rgw-zonegroup=ldc1zg --
   endpoints=http://rgw1:80 --rgw-realm=ldc1 --master --default
   ```

3. Create one local zone on the first data center:

   Syntax

   ```
   radosgw-admin zone create --rgw-zonegroup=ZONE_GROUP_NAME --rgw-zone=ZONE_NAME --
   master --default --endpoints=HTTP_FQDN[,HTTP_FQDN]
   ```

   Example

   ```
   [ceph: root@host01 /]# radosgw-admin zone create --rgw-zonegroup=ldc1zg --rgw-
   zone=ldc1z --master --default --endpoints=http://rgw.example.com
   ```

4. Commit the period:

   Example

   ```
   [ceph: root@host01 /]# radosgw-admin period update --commit
   ```

5. Optional: If you specified the realm and zone in the service specification during the deployment of the Ceph Object Gateway, update the `spec` section of the specification file:

   Syntax

   ```
   spec:
       rgw_realm: REALM_NAME
       rgw_zone: ZONE_NAME
   ```

6. You can either deploy the Ceph Object Gateway daemons with the appropriate realm and zone or update the configuration database:

   - Deploy the Ceph Object Gateway using placement specification:

     Syntax

     ```
     ceph orch apply rgw SERVICE_NAME --realm=REALM_NAME --zone=ZONE_NAME --
     placement="NUMBER_OF_DAEMONS HOST_NAME_1 HOST_NAME_2"
     ```

     Example

```
[ceph: root@host01 /]# ceph orch apply rgw rgw --realm=ldc1 --zone=ldc1z --
placement="1 host01"
```

- Update the Ceph configuration database:

  Syntax

  ```
  ceph config set client.rgw.SERVICE_NAME rgw_realm REALM_NAME
  ceph config set client.rgw.SERVICE_NAME rgw_zonegroup ZONE_GROUP_NAME
  ceph config set client.rgw.SERVICE_NAME rgw_zone ZONE_NAME
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph config set client.rgw.rgwsvcid.mons-1.jwgwwp
  rgw_realm ldc1
  [ceph: root@host01 /]# ceph config set client.rgw.rgwsvcid.mons-1.jwgwwp
  rgw_zonegroup ldc1zg
  [ceph: root@host01 /]# ceph config set client.rgw.rgwsvcid.mons-1.jwgwwp rgw_zone
  ldc1z
  ```

7. Restart the Ceph Object Gateway.

   Note: Use the output from the `ceph orch ps` command, under the `NAME` column, to get the *SERVICE_TYPE.ID* information.

   a. To restart the Ceph Object Gateway on an individual node in the storage cluster:

      Syntax

      ```
      systemctl restart ceph-CLUSTER_ID@SERVICE_TYPE.ID.service
      ```

      Example

      ```
      [root@host01 ~]# systemctl restart ceph-c4b34c6f-8365-11ba-
      dc31-529020a7702d@rgw.realm.zone.host01.gwasto.service
      ```

   b. To restart the Ceph Object Gateways on all nodes in the storage cluster:

      Syntax

      ```
      ceph orch restart SERVICE_TYPE
      ```

      Example

      ```
      [ceph: root@host01 /]# ceph orch restart rgw
      ```

8. Create one local realm on the second data center in the storage cluster:

   Syntax

   ```
   radosgw-admin realm create --rgw-realm=REALM_NAME --default
   ```

   Example

   ```
   [ceph: root@host04 /]# radosgw-admin realm create --rgw-realm=ldc2 --default
   ```

9. Create one local master zonegroup on the second data center:

   Syntax

   ```
   radosgw-admin zonegroup create --rgw-zonegroup=ZONE_GROUP_NAME --endpoints=http://
   RGW_NODE_NAME:80 --rgw-realm=REALM_NAME --master --default
   ```

   Example

```
[ceph: root@host04 /]# radosgw-admin zonegroup create --rgw-zonegroup=ldc2zg --
endpoints=http://rgw2:80 --rgw-realm=ldc2 --master --default
```

10. Create one local zone on the second data center:

    Syntax

    ```
    radosgw-admin zone create --rgw-zonegroup=ZONE_GROUP_NAME --rgw-zone=ZONE_NAME --
    master --default --endpoints=HTTP_FQDN[, HTTP_FQDN]
    ```

    Example

    ```
    [ceph: root@host04 /]# radosgw-admin zone create --rgw-zonegroup=ldc2zg --rgw-
    zone=ldc2z --master --default --endpoints=http://rgw.example.com
    ```

11. Commit the period:

    Example

    ```
    [ceph: root@host04 /]# radosgw-admin period update --commit
    ```

12. Optional: If you specified the realm and zone in the service specification during the deployment of the Ceph Object Gateway, update the `spec` section of the specification file:

    Syntax

    ```
    spec:
        rgw_realm: REALM_NAME
        rgw_zone: ZONE_NAME
    ```

13. You can either deploy the Ceph Object Gateway daemons with the appropriate realm and zone or update the configuration database:

    - Deploy the Ceph Object Gateway using placement specification:

      Syntax

      ```
      ceph orch apply rgw SERVICE_NAME --realm=REALM_NAME --zone=ZONE_NAME --
      placement="NUMBER_OF_DAEMONS HOST_NAME_1 HOST_NAME_2"
      ```

      Example

      ```
      [ceph: root@host01 /]# ceph orch apply rgw rgw --realm=ldc2 --zone=ldc2z --
      placement="1 host01"
      ```

    - Update the Ceph configuration database:

      Syntax

      ```
      ceph config set client.rgw.SERVICE_NAME rgw_realm REALM_NAME
      ceph config set client.rgw.SERVICE_NAME rgw_zonegroup ZONE_GROUP_NAME
      ceph config set client.rgw.SERVICE_NAME rgw_zone ZONE_NAME
      ```

      Example

      ```
      [ceph: root@host01 /]# ceph config set client.rgw.rgwsvcid.mons-1.jwgwwp
      rgw_realm ldc2
      [ceph: root@host01 /]# ceph config set client.rgw.rgwsvcid.mons-1.jwgwwp
      rgw_zonegroup ldc2zg
      [ceph: root@host01 /]# ceph config set client.rgw.rgwsvcid.mons-1.jwgwwp rgw_zone
      ldc2z
      ```

14. Restart the Ceph Object Gateway.

Note: Use the output from the `ceph orch ps` command, under the `NAME` column, to get the *SERVICE_TYPE.ID* information.

a. To restart the Ceph Object Gateway on individual node in the storage cluster:

Syntax

```
systemctl restart ceph-CLUSTER_ID@SERVICE_TYPE.ID.service
```

Example

```
[root@host04 ~]# systemctl restart ceph-c4b34c6f-8365-11ba-
dc31-529020a7702d@rgw.realm.zone.host01.gwasto.service
```

b. To restart the Ceph Object Gateways on all nodes in the storage cluster:

Syntax

```
ceph orch restart SERVICE_TYPE
```

Example

```
[ceph: root@host04 /]# ceph orch restart rgw
```

15. Create a replicated realm on the first data center in the storage cluster:

Syntax

```
radosgw-admin realm create --rgw-realm=REPLICATED_REALM_1 --default
```

Example

```
[ceph: root@host01 /]# radosgw-admin realm create --rgw-realm=rdc1 --default
```

Use the `--default` flag to make the replicated realm default on the primary site.

16. Create a master zonegroup for the first data center:

Syntax

```
radosgw-admin zonegroup create --rgw-zonegroup=RGW_ZONE_GROUP --endpoints=http://
_RGW_NODE_NAME:80 --rgw-realm=_RGW_REALM_NAME --master --default
```

Example

```
[ceph: root@host01 /]# radosgw-admin zonegroup create --rgw-zonegroup=rdc1zg --
endpoints=http://rgw1:80 --rgw-realm=rdc1 --master --default
```

17. Create a master zone on the first data center:

Syntax

```
radosgw-admin zone create --rgw-zonegroup=RGW_ZONE_GROUP --rgw-
zone=_MASTER_RGW_NODE_NAME --master --default --endpoints=HTTP_FQDN[,HTTP_FQDN]
```

Example

```
[ceph: root@host01 /]# radosgw-admin zone create --rgw-zonegroup=rdc1zg --rgw-
zone=rdc1z --master --default --endpoints=http://rgw.example.com
```

18. Create a synchronization user and add the system user to the master zone for multi-site:

Syntax

```
radosgw-admin user create --uid="SYNCHRONIZATION_USER" --display-name="Synchronization
User" --system
radosgw-admin zone modify --rgw-zone=RGW_ZONE --access-key=ACCESS_KEY --
secret=SECRET_KEY
```

Example

```
radosgw-admin user create --uid="synchronization-user" --display-name="Synchronization
User" --system
[ceph: root@host01 /]# radosgw-admin zone modify --rgw-zone=rdc1zg --access-
key=3QV0D6ZMMCJZMSCXJ2QJ --secret=VpvQWcsfI9OPzUCpR4kynDLAbqa1OIKqRB6WEnH8
```

19. Commit the period:

    Example

    ```
    [ceph: root@host01 /]# radosgw-admin period update --commit
    ```

20. Optional: If you specified the realm and zone in the service specification during the deployment of the Ceph Object Gateway, update the `spec` section of the specification file:

    Syntax

    ```
    spec:
        rgw_realm: REALM_NAME
        rgw_zone: ZONE_NAME
    ```

21. You can either deploy the Ceph Object Gateway daemons with the appropriate realm and zone or update the configuration database:

    - Deploy the Ceph Object Gateway using placement specification:

      Syntax

      ```
      ceph orch apply rgw SERVICE_NAME --realm=REALM_NAME --zone=ZONE_NAME --
      placement="NUMBER_OF_DAEMONS HOST_NAME_1 HOST_NAME_2"
      ```

      Example

      ```
      [ceph: root@host01 /]# ceph orch apply rgw rgw --realm=rdc1 --zone=rdc1z --
      placement="1 host01"
      ```

    - Update the Ceph configuration database:

      Syntax

      ```
      ceph config set client.rgw.SERVICE_NAME  rgw_realm REALM_NAME
      ceph config set client.rgw.SERVICE_NAME  rgw_zonegroup ZONE_GROUP_NAME
      ceph config set client.rgw.SERVICE_NAME  rgw_zone ZONE_NAME
      ```

      Example

      ```
      [ceph: root@host01 /]# ceph config set client.rgw.rgwsvcid.mons-1.jwgwwp
      rgw_realm rdc1
      [ceph: root@host01 /]# ceph config set client.rgw.rgwsvcid.mons-1.jwgwwp
      rgw_zonegroup rdc1zg
      [ceph: root@host01 /]# ceph config set client.rgw.rgwsvcid.mons-1.jwgwwp rgw_zone
      rdc1z
      ```

22. Restart the Ceph Object Gateway.

    Note: Use the output from the `ceph orch ps` command, under the `NAME` column, to get the *SERVICE_TYPE.ID* information.

    a. To restart the Ceph Object Gateway on individual node in the storage cluster:

Syntax

```
systemctl restart ceph-CLUSTER_ID@SERVICE_TYPE.ID.service
```

Example

```
[root@host01 ~]# systemctl restart ceph-c4b34c6f-8365-11ba-
dc31-529020a7702d@rgw.realm.zone.host01.gwasto.service
```

b. To restart the Ceph Object Gateways on all nodes in the storage cluster:

Syntax

```
ceph orch restart SERVICE_TYPE
```

Example

```
[ceph: root@host01 /]# ceph orch restart rgw
```

23. Pull the replicated realm on the second data center:

Syntax

```
radosgw-admin realm pull --url=https://tower-osd1.cephtips.com --access-key=ACCESS_KEY
--secret-key=SECRET_KEY
```

Example

```
[ceph: root@host01 /]# radosgw-admin realm pull --url=https://tower-osd1.cephtips.com
--access-key=3QV0D6ZMMCJZMSCXJ2QJ --secret-
key=VpvQWcsfI9OPzUCpR4kynDLAbqa1OIKqRB6WEnH8
```

24. Pull the period from the first data center:

Syntax

```
radosgw-admin period pull --url=https://tower-osd1.cephtips.com --access-
key=ACCESS_KEY --secret-key=SECRET_KEY
```

Example

```
[ceph: root@host01 /]# radosgw-admin period pull --url=https://tower-osd1.cephtips.com
--access-key=3QV0D6ZMMCJZMSCXJ2QJ --secret-
key=VpvQWcsfI9OPzUCpR4kynDLAbqa1OIKqRB6WEnH8
```

25. Create the secondary zone on the second data center:

Syntax

```
radosgw-admin zone create --rgw-zone=RGW_ZONE --rgw-zonegroup=RGW_ZONE_GROUP --
endpoints=https://tower-osd4.cephtips.com --access-key=_ACCESS_KEY --secret-
key=SECRET_KEY
```

Example

```
[ceph: root@host04 /]# radosgw-admin zone create --rgw-zone=rdc2z --rgw-
zonegroup=rdc1zg --endpoints=https://tower-osd4.cephtips.com --access-
key=3QV0D6ZMMCJZMSCXJ2QJ --secret-key=VpvQWcsfI9OPzUCpR4kynDLAbqa1OIKqRB6WEnH8
```

26. Commit the period:

Example

```
[ceph: root@host04 /]# radosgw-admin period update --commit
```

27. Optional: If you specified the realm and zone in the service specification during the deployment of the Ceph Object Gateway, update the `spec` section of the specification file:

    Syntax

    ```
    spec:
        rgw_realm: REALM_NAME
        rgw_zone: ZONE_NAME
    ```

28. You can either deploy the Ceph Object Gateway daemons with the appropriate realm and zone or update the configuration database:

    - Deploy the Ceph Object Gateway using placement specification:

      Syntax

      ```
      ceph orch apply rgw SERVICE_NAME --realm=REALM_NAME --zone=ZONE_NAME --
      placement="NUMBER_OF_DAEMONS HOST_NAME_1 HOST_NAME_2"
      ```

      Example

      ```
      [ceph: root@host04 /]# ceph orch apply rgw rgw --realm=rdc1 --zone=rdc2z --
      placement="1 host04"
      ```

    - Update the Ceph configuration database:

      Syntax

      ```
      ceph config set client.rgw.SERVICE_NAME rgw_realm REALM_NAME
      ceph config set client.rgw.SERVICE_NAME rgw_zonegroup ZONE_GROUP_NAME
      ceph config set client.rgw.SERVICE_NAME rgw_zone ZONE_NAME
      ```

      Example

      ```
      [ceph: root@host04 /]# ceph config set client.rgw.rgwsvcid.mons-1.jwgwwp
      rgw_realm rdc1
      [ceph: root@host04 /]# ceph config set client.rgw.rgwsvcid.mons-1.jwgwwp
      rgw_zonegroup rdc1zg
      [ceph: root@host04 /]# ceph config set client.rgw.rgwsvcid.mons-1.jwgwwp rgw_zone
      rdc2z
      ```

29. Restart the Ceph Object Gateway.

    > **Note:** Use the output from the `ceph orch ps` command, under the `NAME` column, to get the *SERVICE_TYPE.ID* information.

    a. To restart the Ceph Object Gateway on individual node in the storage cluster:

       Syntax

       ```
       systemctl restart ceph-CLUSTER_ID@SERVICE_TYPE.ID.service
       ```

       Example

       ```
       [root@host02 ~]# systemctl restart ceph-c4b34c6f-8365-11ba-
       dc31-529020a7702d@rgw.realm.zone.host01.gwasto.service
       ```

    b. To restart the Ceph Object Gateways on all nodes in the storage cluster:

       Syntax

       ```
       ceph orch restart SERVICE_TYPE
       ```

       Example

```
[ceph: root@host04 /]# ceph orch restart rgw
```

30. Log in as `root` on the endpoint for the second data center.

31. Verify the synchronization status on the master realm:

    Syntax

    ```
    radosgw-admin sync status
    ```

    Example

    ```
    [ceph: root@host04 /]# radosgw-admin sync status
              realm 59762f08-470c-46de-b2b1-d92c50986e67 (ldc2)
          zonegroup 7cf8daf8-d279-4d5c-b73e-c7fd2af65197 (ldc2zg)
               zone 034ae8d3-ae0c-4e35-8760-134782cb4196 (ldc2z)
      metadata sync no sync (zone is master)
    ```

32. Log in as `root` on the endpoint for the first data center.

33. Verify the synchronization status for the replication-synchronization realm:

    Syntax

    ```
    radosgw-admin sync status --rgw-realm RGW_REALM_NAME
    ```

    Example

    ```
    [ceph: root@host01 /]# radosgw-admin sync status --rgw-realm rdc1
              realm 73c7b801-3736-4a89-aaf8-e23c96e6e29d (rdc1)
          zonegroup d67cc9c9-690a-4076-89b8-e8127d868398 (rdc1zg)
               zone 67584789-375b-4d61-8f12-d1cf71998b38 (rdc2z)
      metadata sync syncing
                    full sync: 0/64 shards
                    incremental sync: 64/64 shards
                    metadata is caught up with master
          data sync source: 705ff9b0-68d5-4475-9017-452107cec9a0 (rdc1z)
                            syncing
                            full sync: 0/128 shards
                            incremental sync: 128/128 shards
                            data is caught up with source
              realm 73c7b801-3736-4a89-aaf8-e23c96e6e29d (rdc1)
          zonegroup d67cc9c9-690a-4076-89b8-e8127d868398 (rdc1zg)
               zone 67584789-375b-4d61-8f12-d1cf71998b38 (rdc2z)
      metadata sync syncing
                    full sync: 0/64 shards
                    incremental sync: 64/64 shards
                    metadata is caught up with master
          data sync source: 705ff9b0-68d5-4475-9017-452107cec9a0 (rdc1z)
                            syncing
                            full sync: 0/128 shards
                            incremental sync: 128/128 shards
                            data is caught up with source
    ```

34. To store and access data in the local site, create the user for local realm:

    Syntax

    ```
    radosgw-admin user create --uid="LOCAL_USER" --display-name="Local user" --rgw-
    realm=_REALM_NAME --rgw-zonegroup=ZONE_GROUP_NAME --rgw-zone=ZONE_NAME
    ```

    Example

    ```
    [ceph: root@host04 /]# radosgw-admin user create --uid="local-user" --display-
    name="Local user" --rgw-realm=ldc1 --rgw-zonegroup=ldc1zg --rgw-zone=ldc1z
    ```

    Important: By default, users are created under the default realm. For the users to access data in the local realm, the `radosgw-admin` command requires the `--rgw-realm` argument.

Using multi-site sync policies

As a storage administrator, you can use multi-site sync policies at the bucket level to control data movement between buckets in different zones. These policies are called bucket-granularity sync policies. Previously, all buckets within zones were treated symmetrically. This means that each zone contained a mirror copy of a given bucket, and the copies of buckets were identical in all of the zones. The sync process assumed that the bucket sync source and the bucket sync destination referred to the same bucket.

Important: Bucket sync policies apply to data only, and metadata is synced across all the zones in the multi-site irrespective of the presence of the bucket sync policies. Objects that were created, modified, or deleted when the bucket sync policy was in `allowed` or `forbidden` place, it does not automatically sync when policy takes effect. Run the `bucket sync run` command to sync these objects.

Important: If there are multiple sync policies defined at zonegroup level, only one policy can be in enabled state at any time. We can toggle between policies if needed

The sync policy supersedes the old zone group coarse configuration (`sync_from*`). The sync policy can be configured at the zone group level. If it is configured, it replaces the old-style configuration at the zone group level, but it can also be configured at the bucket level.

When the sync policy of a bucket or zonegroup moves from `disabled` to `enabled` state, the following behavioral changes are observed:

- **Greenfield deployment** - To set up bucket granular sync replication, a new zonegroup, or zone must be configured at a minimum.
- **Brownfield deployment** - Migrate or upgrade Ceph Object Gateway multi-site replication configurations to the newly featured bucket granular sync policy replication.
- **Data flow** - directional, symmetrical - Both unidirectional and bi-directional (symmetrical) replication can be configured.

Note: The bucket sync policies are applicable to the archive zones. The movement from an archive zone is not bidirectional wherein all the objects can be moved from active zone to archive zone. However, you cannot move objects from the archive zone to active zone since archive zone is read-only.

Important: In this release, the following features are not supported:

- Source filters
- Storage class
- Destination owner translation
- User mode

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to a Ceph Monitor node.
- Installation of the Ceph Object Gateway.

Multi-site sync policy group state

In the sync policy, multiple groups that can contain lists of data-flow configurations can be defined and listed. The data-flow defines the flow of data between the different zones. It can define symmetrical data flow, in which multiple zones sync data from each other. It can also define directional data flow, in which the data moves in one way from one zone to another.

A pipe defines the actual buckets that can use these data flows, and the properties that are associated with it, such as the source object prefix.

A sync policy group can be in three states:

- `enabled` - sync is allowed and enabled.
- `allowed` - sync is allowed.
- `forbidden` - sync, as defined by this group, is not allowed. Sync states in this group can override other groups.

When the zones replicate, you can disable replication for specific buckets using the sync policy. The following are the semantics that need to be followed to resolve the policy conflicts:

Semantics		
Zonegroup	**Bucket**	**Result**
enabled	enabled	enabled
enabled	allowed	enabled
enabled	forbidden	disabled
allowed	enabled	enabled
allowed	allowed	disabled
allowed	forbidden	disabled
forbidden	enabled	disabled
forbidden	allowed	disabled
forbidden	forbidden	disabled

For multiple group polices that are set to reflect for any sync pair (*SOURCE_ZONE*,*SOURCE_BUCKET*), (*DESTINATION_ZONE*,*DESTINATION_BUCKET*), the following rules are applied in the following order:

- Even if one sync policy is `forbidden`, the sync is `disabled`.
- At least one policy should be `enabled` for the sync to be `allowed`.

A policy can be defined at the bucket level. A bucket-level sync policy inherits the data flow of the zonegroup policy, and can define a subset of what the zonegroup allows only.

A wildcard zone, and a wildcard bucket parameter in the policy defines all relevant zones, or all relevant buckets. In the context of a bucket policy, it means the current bucket instance. A disaster recovery configuration where entire zones are mirrored does not require configuring anything on the buckets. However, for a fine grained bucket sync it would be better to configure the pipes to be synced by allowing (`status=allowed`) them at the zonegroup level (for example, by using wildcard). However, enable the specific sync at the bucket level (`status=enabled`) only. If needed, the policy at the bucket level can limit the data movement to specific relevant zones.

> **Important:** Any changes to the zonegroup policy need to be applied on the zonegroup master zone, and require period update and commit. Changes to the bucket policy need to be applied on the zonegroup master zone. Ceph Object Gateway handles these changes dynamically.

References

- See "S3 bucket replication API" on page 130 for more details.

Retrieving the current policy

You can use the **get** command to retrieve the current zonegroup sync policy, or a specific bucket policy.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root or sudo access.
- The Ceph Object Gateway is installed.

Procedure

- Retrieve the current zonegroup sync policy or bucket policy. To retrieve a specific bucket policy, use the `--bucket` option:

  Syntax

  ```
  radosgw-admin sync policy get --bucket=BUCKET_NAME
  ```

  Example

  ```
  [ceph: root@host01 /]# radosgw-admin sync policy get --bucket=mybucket
  ```

Creating a sync policy group

You can create a sync policy group for the current zone group, or for a specific bucket.

PREREQUISITE

- A running Red Hat Ceph Storage cluster.
- Root or sudo access.
- The Ceph Object Gateway is installed.
- When creating for an archive zone, be sure that the archive zone is created before the sync policy group.

When creating a sync policy for bucket granular replication for a sync policy group that has changed from `forbidden` to `enabled`, a manual update might be necessary to complete the sync process.

For example, if any data is written to `bucket1` when its policy is `forbidden`, the data might not sync properly across zones after the policy is changed to `enabled`. To properly sync the changes, run the **bucket sync run** command on the sync policy. This step is also necessary if the bucket is resharded when the policy is `forbidden`. In this case the **bucket sync run** command must also be used after enabling the policy.

For more information about configuring an archive zone and bucket granular replication, see "Configuring the archive zone" on page 79.

1. Create a sync policy group or a bucket policy, using the `.--bucket` option.

   ```
   radosgw-admin sync group create --bucket=BUCKET_NAME --group-id=GROUP_ID --
   status=enabled | allowed | forbidden
   ```

   For example,

   ```
   [ceph: root@host01 /]# radosgw-admin sync group create --group-id=mygroup1 --
   status=enabled
   ```

2. Optional: Manually complete the sync process for bucket granular replication.

> **Note:** This step is mandatory when using as part of an archive zone with bucket granular replication if the policy has data written or the bucket was resharded when the policy was `forbidden`.

```
radosgw-admin bucket sync run
```

For example,

```
[ceph: root@host01 /]# radosgw-admin bucket sync run
```

Modifying a sync policy group

You can modify an existing sync policy group for the current zone group, or for a specific bucket.

PREREQUISITE

- A running Red Hat Ceph Storage cluster.
- Root or `sudo` access.
- The Ceph Object Gateway is installed.
- When creating for an archive zone, be sure that the archive zone is created before the sync policy group.

When creating a sync policy for bucket granular replication for a sync policy group that has changed from `forbidden`to`enabled`, a manual update might be necessary to complete the sync process.

For example, if any data is written to `bucket1` when its policy is`forbidden`, the data might not sync properly across zones after the policy is changed to`enabled`. To properly sync the changes, run the **`bucket sync run`** command on the sync policy. This step is also necessary if the bucket is resharded when the policy is `forbidden`. In this case the **`bucket sync run`** command must also be used after enabling the policy.

For more information about configuring an archive zone and bucket granular replication, see “Configuring the archive zone” on page 79.

1. Modify the sync policy group or a bucket policy, using the`--bucket`option.

   ```
   radosgw-admin sync group modify --bucket=BUCKET_NAME --group-id=GROUP_ID --
   status=enabled | allowed | forbidden
   ```

   For example,

   ```
   [ceph: root@host01 /]# radosgw-admin sync group modify --group-id=mygroup1 --
   status=forbidden
   ```

2. Optional: Manually complete the sync process for bucket granular replication.

   > **Note:** This step is mandatory when using as part of an archive zone with bucket granular replication if the policy has data written or the bucket was resharded when the policy was `forbidden`.

   ```
   radosgw-admin bucket sync run
   ```

   For example,

   ```
   [ceph: root@host01 /]# radosgw-admin bucket sync run
   ```

Getting a sync policy group

You can use the **`group get`** command to show the current sync policy group by group ID, or to show a specific bucket policy.

If the `--bucket` option is not provided, the groups created at zonegroup level is retrieved and not the groups at the bucket-level.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root or `sudo` access.
- The Ceph Object Gateway is installed.

Procedure

- Show the current sync policy group or bucket policy. To show a specific bucket policy, use the `--bucket` option:

  Syntax

  ```
  radosgw-admin sync group get --bucket=BUCKET_NAME --group-id=GROUP_ID
  ```

  Example

  ```
  [ceph: root@host01 /]# radosgw-admin sync group get --group-id=mygroup
  ```

Removing a sync policy group

You can use the **`group remove`** command to remove the current sync policy group by group ID, or to remove a specific bucket policy.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root or `sudo` access.
- The Ceph Object Gateway is installed.

Procedure

- Remove the current sync policy group or bucket policy. To remove a specific bucket policy, use the `--bucket` option:

  Syntax

  ```
  radosgw-admin sync group remove --bucket=BUCKET_NAME --group-id=GROUP_ID
  ```

  Example

  ```
  [ceph: root@host01 /]# radosgw-admin sync group remove --group-id=mygroup
  ```

Creating a sync flow

Create a sync flow for a multi-site sync policy.

You can create two different types of flows for a sync policy group or for a specific bucket:

- Directional sync flow

- Symmetrical sync flow

The `group flow create` command creates a sync flow. For an existing sync policy group or bucket, this command overwrites the existing settings for the sync flow and applies the settings that you specify.

Option	Description	Required/Optional
`--bucket`	Name of the bucket to which sync policy needs to be configured. Used only in bucket-level sync policy.	Optional
`--group-id`	ID of the sync group.	Required
`--flow-id`	ID of the flow.	Required
`--flow-type`	Types of flows for a sync policy group or for a specific bucket - directional or symmetrical.	Required
`--source-zone`	To specify the source zone from which sync should happen. Zone that send data to the sync group. Required if flow type of sync group is directional.	Optional
`--dest-zone`	To specify the destination zone to which sync should happen. Zone that receives data from the sync group. Required if flow type of sync group is directional.	Optional
`--zones`	Zones that part of the sync group. Zones mention will be both sender and receiver zone. Specify zones separated by ",". Required if flow type of sync group is symmetrical.	Optional

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root or `sudo` access.
- The Ceph Object Gateway is installed.

Procedure

1. Create or update a directional sync flow. To create or update directional sync flow for a specific bucket, use the `--bucket` option.

   Syntax

   ```
   radosgw-admin sync group flow create --bucket=BUCKET_NAME --group-id=GROUP_ID --flow-id=FLOW_ID --flow-type=directional --source-zone=SOURCE_ZONE --dest-zone=DESTINATION_ZONE
   ```

2. Create or update a symmetrical sync flow. To specify multiple zones for a symmetrical flow type, use a comma-separated list for the `--zones` option.

   Syntax

   ```
   radosgw-admin sync group flow create --bucket=BUCKET_NAME --group-id=GROUP_ID --flow-id=FLOW_ID --flow-type=symmetrical --zones=ZONE_NAME_1,ZONE_NAME_2
   ```

`zones` are comma-separated lists of all zones that need to be added to the flow.

Removing sync flows and zones

Remove sync flows and zones from a sync policy or bucket.

The `group flow remove` command removes sync flows or zones from a sync policy group or bucket.

For sync policy groups or buckets using directional flows, `group flow remove` command removes the flow. For sync policy groups or buckets using symmetrical flows, you can use the `group flow remove` command to remove specified zones from the flow, or to remove the flow.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root or `sudo` access.
- The Ceph Object Gateway is installed.

Procedure

1. Remove a directional sync flow. To remove the directional sync flow for a specific bucket, use the `--bucket` option.

   Syntax

   ```
   radosgw-admin sync group flow remove --bucket=BUCKET_NAME --group-id=GROUP_ID --flow-id=FLOW_ID --flow-type=directional --source-zone=SOURCE_ZONE --dest-zone=DESTINATION_ZONE
   ```

2. Remove specific zones from a symmetrical sync flow. To remove multiple zones from a symmetrical flow, use a comma-separated list for the `--zones` option.

   Syntax

   ```
   radosgw-admin sync group flow remove --bucket=BUCKET_NAME --group-id=GROUP_ID --flow-id=FLOW_ID --flow-type=symmetrical --zones=ZONE_NAME_1,ZONE_NAME_2
   ```

3. Remove a symmetrical sync flow. To remove the sync flow at the zonegroup level, use the `--bucket` option.

   Syntax

   ```
   radosgw-admin sync group flow remove --group-id=GROUP_ID --flow-id=FLOW_ID --flow-type=symmetrical --zones=ZONE_NAME_1,ZONE_NAME_2
   ```

Creating and modifying a sync group pipe

Define pipes to specify which buckets can use your configured data flows and the properties that are associated with those data flows. Use the **`sync group pipe create`** command to create or modify the sync group pipe.

PREREQUISITE

- A running Red Hat Ceph Storage cluster.
- Root or `sudo` access.
- The Ceph Object Gateway is installed.

As a storage administrator, you can use the **`sync group pipe create`** command to create pipes. These pipes are custom sync group data flows between specific buckets or groups of buckets, or between specific zones or groups of zones. The **`sync group pipe create`** command uses the options that are listed in “sync group pipe create Command options” on page 101.

sync group pipe create *Command options*		
Option	**Description**	**Required/Optional**
--bucket	Name of the bucket to which sync policy needs to be configured. Used only in bucket-level sync policy.	Optional
--group-id	ID of the sync group	Required
--pipe-id	ID of the pipe	Required
--source-zones	Zones that send data to the sync group. Use single quotation marks (') for value. Use commas to separate multiple zones. Use the wildcard * for all zones that match the data flow rules.	Required
--source-bucket	Bucket or buckets that send data to the sync group. If bucket name is not mentioned, then * (wildcard) is taken as the default value. At bucket-level, source bucket will be the bucket for which the sync group created and at zonegroup-level, source bucket will be all buckets.	Optional
--source-bucket-id	ID of the source bucket. If this option is not specified, then * is the default value and is applied to all the buckets.	Optional
--dest-zones	Zone or zones that receive the sync data. Use single quotation marks (') for value. Use commas to separate multiple zones. Use the wildcard * for all zones that match the data flow rules.	Required
--dest-bucket	Bucket or buckets that receive the sync data. If bucket name is not mentioned, then * (wildcard) is taken as the default value. At bucket-level, destination bucket will be the bucket for which the sync group is created and at zonegroup-level, destination bucket will be all buckets.	Optional
--dest-bucket-id	ID of the destination bucket. If this option is not specified, then * is the default value and is applied to all the buckets.	Optional
--prefix	Bucket prefix. Use the wildcard * to filter for source objects.	Optional
--prefix-rm	Remove object prefix from filtering. Do not use bucket prefix for filtering.	Optional

Option	Description	Required/Optional
--tags-add	Comma-separated list of key=value pairs.	Optional
--tags-rm	Removes one or more key=value pairs of tags.	Optional
--dest-owner	Destination owner of the objects from source.	Optional
--storage-class	Destination storage class for the objects from source.	Optional
--mode	Use `system` for system mode or `user` for user mode.	Optional
--uid	Used for permissions validation in user mode. Specifies the user ID under which the sync operation will be issued.	Optional

Note: To enable or disable sync at zonegroup level for certain buckets, set zonegroup level sync policy to `enable` or `disable` state respectively, and create a pipe for each bucket with `--source-bucket` and `--dest-bucket` with its bucket name or with `bucket-id` i.e, `--source-bucket-id` and `--dest-bucket-id`.

- Create the sync group pipe.
  The `create` command is also used to modify a command by creating the sync group pipe with only the relevant options.

```
radosgw-admin sync group pipe create --bucket=BUCKET_NAME --group-id=GROUP_ID --pipe-id=PIPE_ID --source-zones='ZONE_NAME','ZONE_NAME2'... --source-bucket=SOURCE_BUCKET1 --source-bucket-id=SOURCE_BUCKET_ID --dest-zones='ZONE_NAME1_','ZONE_NAME2'... --dest-bucket=DESTINATION_BUCKET1 --dest-bucket-id=DESTINATION_BUCKET_ID --prefix=SOURCE_PREFIX --prefix-rm --tags-add=KEY1=VALUE1,KEY2=VALUE2, ... --tags-rm=KEY1=VALUE1,KEY2=VALUE2, ... --dest-owner=OWNER_ID --storage-class=STORAGE_CLASS --mode=USER --uid=USER_ID
```

Modifying or deleting a sync group pipe

Use the **sync group pipe remove** command to modify or delete the sync group pipe.

PREREQUISITE

- A running Red Hat Ceph Storage cluster.
- Root or `sudo` access.
- The Ceph Object Gateway is installed.

As a storage administrator, you can use the **sync group pipe modify** command or **sync group pipe remove** command to modify the sync group pipe by removing certain options. You can also use this command to remove zones, buckets, or the sync group pipe completely.

- Modify the sync group pipe options with the `modify` argument.

```
radosgw-admin sync group pipe modify --bucket=BUCKET_NAME --group-id=GROUP_ID --pipe-id=PIPE_ID --source-zones='ZONE_NAME_1','ZONE_NAME2'... --source-bucket=SOURCE_BUCKET --source-bucket-id=SOURCE_BUCKET_ID --dest-zones='ZONE_NAME_1','ZONE_NAME2'... --dest-bucket=DESTINATION_BUCKET1 --dest-bucket-id=DESTINATION_BUCKET-ID
```

Note: Ensure to put the zones in single quotes ('). The source bucket does not need the quotes.

```
[root@host01 ~]# radosgw-admin sync group pipe modify --group-id=zonegroup --pipe-
id=pipe --dest-zones='primary','secondary','tertiary' --source-
zones='primary','secondary','tertiary' --source-bucket=pri-bkt-1 --dest-bucket=pri-
bkt-1
```

- Modify the sync group pipe options with the `remove` argument.

```
radosgw-admin sync group pipe remove --bucket=BUCKET_NAME --group-id=GROUP_ID --pipe-
id=PIPE_ID --source-zones='ZONE_NAME1','ZONE_NAME2'... --source-bucket=SOURCE_BUCKET --
source-bucket-id=SOURCE_BUCKET_ID --dest-zones='ZONE_NAME_1','ZONE_NAME2'... --dest-
bucket=DESTINATION_BUCKET --dest-bucket-id=DESTINATION_BUCKET-ID
```

```
[root@host01 ~]# radosgw-admin sync group pipe remove --group-id=zonegroup --pipe-
id=pipe --dest-zones='primary','secondary','tertiary' --source-
zones='primary','secondary','tertiary' --source-bucket=pri-bkt-1 --dest-bucket=pri-
bkt-1
```

- Delete a sync group pipe.

```
radosgw-admin sync group pipe remove --bucket=BUCKET_NAME --group-id=GROUP_ID --pipe-
id=PIPE_ID
```

```
[root@host01 ~]# radosgw-admin sync group pipe remove -bucket-name=mybuck --group-
id=zonegroup --pipe-id=pipe
```

Obtaining information about sync operations

The **`sync info`** command enables you to get information about the expected sync sources and targets, as defined by the sync policy.

When you create a sync policy for a bucket, that policy defines how data moves from that bucket toward a different bucket in a different zone. Creating the policy also creates a list of bucket dependencies that are used as hints whenever that bucket syncs with another bucket. Note that a bucket can refer to another bucket without actually syncing to it, since syncing depends on whether the data flow allows the sync to take place.

Both the `--bucket` and `effective-zone-name` parameters are optional. If you invoke the `sync info` command without specifying any options, the Ceph Object Gateway returns all of the sync operations defined by the sync policy in all zones.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root or `sudo` access.
- The Ceph Object Gateway is installed.
- A group sync policy is defined.

Procedure

- Get information about sync operations for buckets.

  Syntax

```
radosgw-admin sync info --bucket=BUCKET_NAME --effective-zone-name=ZONE_NAME
```

- Get information about sync operations at the zonegroup level.

  Syntax

```
radosgw-admin sync info
```

Using destination parameters

Learn to use the destination parameters for multisite syn policy.

There are 3 types of destination parameters:

- Storage class
- Destination owner translation
- User mode

You can set the destination owner to force a specific destination owner for the objects. You can configure the destination storage class. If you select user mode, you can set only the destination bucket owner. You can set the user ID for the user mode to execute the sync operation for permissions validation.

Destination Params: Storage Class

Example for configuring storage class of the destination objects:

```
[ceph: root@host01 /]# radosgw-admin sync group create --bucket=buck10 \
 --group-id=buck10-default --status=enabled
[ceph: root@host01 /]# radosgw-admin sync group pipe create --bucket=buck10 \
 --group-id=buck10-default \
 --pipe-id=pipe-storage-class \
 --source-zones='*' --dest-zones=us-west-2 \
 --storage-class=CHEAP_AND_SLOW
```

Destination Params: Destination Owner Translation

Example for setting the destination objects owner as the destination bucket owner. This requires specifying the uid of the destination bucket:

```
[ceph: root@host01 /]# radosgw-admin sync group create --bucket=buck11 \
 --group-id=buck11-default --status=enabled
[ceph: root@host01 /]# radosgw-admin sync group pipe create --bucket=buck11 \
 --group-id=buck11-default --pipe-id=pipe-dest-owner \
 --source-zones='*' --dest-zones='*' \
 --dest-bucket=buck12 --dest-owner=joe
```

Destination Params: User Mode

User mode makes sure that the user has permissions to both read the objects, and write to the destination bucket. This requires that the uid of the user (which in its context the operation executes) is specified.

```
[ceph: root@host01 /]# radosgw-admin sync group pipe modify --bucket=buck11 \
                         --group-id=buck11-default --pipe-id=pipe-dest-owner \
                         --mode=user --uid=jenny
```

Bucket granular sync policies

You can set bucket granular sync policies for a Red Hat Ceph Storage cluster with Ceph Object Gateway configuration in multi-site.

When the sync policy of a bucket or zone group moves from `disabled` to `enabled` state, the following behavioral changes are observed.

Normal scenario

Zone group level

Data written when the sync policy is `disabled` catches up when it is `enabled` with no additional steps.

Bucket level

Data written when the sync policy is `disabled` does not catch up when the policy is `enabled`. In this case, either one of the following two workarounds can be applied:

- Writing new data to the bucket resynchronizes the old data.

- Running **bucket sync run** command syncs all the old data.

> **Note:** When you want to toggle from the sync policy to the legacy policy, you need to first run the **sync init** command followed by the **radosgw-admin bucket sync run** command to sync all the objects.

Reshard scenario

> **Note:** When the policy is set to `enabled` for the zone group and the policy is set to `enabled` or `allowed` for the bucket, the pipe configuration takes effect from zone group level and not at the bucket level. This is known issue BZ#2240719 (IBMCEPH-7545).

Zone group level

Any reshard that happens when the policy is `disabled` sync gets stuck after the policy is `enabled` again. New objects also do not sync at this point. Run the **bucket sync run** command as a workaround.

Bucket level

If any bucket is resharded when the policy is `disabled`, sync gets stuck after the policy is `enabled` again. New objects also do not sync. Run the **bucket sync run** command as a workaround.

Setting bi-directional policy for zonegroups

Set the policy for zone groups bidirectionally with the sync policy engine.

PREREQUISITE

Before you begin setting bi-directional policies for zone groups, be sure to have at least two running Red Hat Ceph Storage clusters with Ceph Object Gateway configured on the same version.
In the following example, create a group policy and define a data flow for the movement of data from one zonegroup to another. Configure a pipe for the zonegroups to define the buckets that can use this data flow. The system in the following examples includes 3 zones: `us-east` (the primary zone), `us-west`, and `us-west-2`. Zonegroup sync policies are created with the new sync policy engine. Any change to the zonegroup sync policy requires a period update and a commit.

1. Create a sync group with the status set to `allowed`.

```
radosgw-admin sync group create --group-id=GROUP_ID --status=allowed
```

For example,

```
ceph: root@host01 /]# radosgw-admin sync group create --group-id=group1 --
status=allowed
```

> **Note:** Until a fully configured zonegroup replication policy is created, set the `--status` to `allowed` to prevent the replication from starting.

2. Create a flow policy for the newly created group with the `--flow-type` set as `symmetrical` to enable bidirectional replication.

```
radosgw-admin sync group flow create --group-id=GROUP_ID --flow-id=FLOW_ID --flow-
type=symmetrical --zones=ZONE_NAMES
```

For example,

```
[ceph: root@host01 /]# radosgw-admin sync group flow create --group-id=group1 --flow-
id=flow-mirror --flow-type=symmetrical --zones=us-east,us-west
```

3. Create a pipe called `pipe`.

```
radosgw-admin sync group pipe create --group-id=GROUP_ID
                          --pipe-id=PIPE_ID --source-zones='*' \
                          --source-bucket='*' --dest-zones='*' \
                          --dest-bucket='*'
```

For example,

```
[ceph: root@host01 /]# radosgw-admin sync group pipe create --group-id=group1 \
                          --pipe-id=pipe1 --source-zones='*' \
                          --source-bucket='*' --dest-zones='*' \
                          --dest-bucket='*'
```

Note:
Use the * wildcard for zones to include all zones set in the previous flow policy, and *for buckets to replicate all existing buckets in the zones.

4. Set the group policy `--status` to `enabled`.

```
radosgw-admin sync group modify --group-id=GROUP_ID --status=enabled
```

For example,

```
[ceph: root@host01 /]# radosgw-admin sync group modify --group-id=group1 --
status=enabled
```

5. Update and commit the new period.

```
radosgw-admin period update --commit
```

For example,

```
[ceph: root@host01 /]# radosgw-admin period update --commit
```

Note: Updating and committing the period is mandatory for a zonegroup policy.

6. Optional: Check the sync source, and destination for a specific bucket. All buckets in `us-east` and `us-west` zones replicate bi-directionally.

```
radosgw-admin sync info --bucket=BUCKET_NAME
```

The `id` field in the output reflects the pipe rule that generated that entry. A single rule can generate multiple sync entries as seen in the following example.

7. Optional: Run the **sync info** command to retrieve information about the expected bucket sync sources and targets, as defined in the policy.

```
radosgw-admin sync info --bucket=BUCKET_NAME
```

For example,

```
[ceph: root@host01 /]# radosgw-admin sync info --bucket=buck
{
    "sources": [
        {
            "id": "pipe1",
            "source": {
                "zone": "us-east",
                "bucket": "buck:115b12b3-....4409.1"
            },
            "dest": {
                "zone": "us-west",
                "bucket": "buck:115b12b3-....4409.1"
            },
            ...
        }
    ],
    "dests": [
        {
            "id": "pipe1",
            "source": {
                "zone": "us-west",
                "bucket": "buck:115b12b3-....4409.1"
            },
            "dest": {
                "zone": "us-east",
                "bucket": "buck:115b12b3-....4409.1"
            },
           ...
        }
    ],
    ...
}
```

Setting directional policy for zonegroups

Set the policy for zone groups uni directionally with the sync policy engine.

PREREQUISITE

At least two running Red Hat Ceph Storage clusters with Ceph Object Gateway configured on the same version. In the following example, create a group policy and configure the data flow for the movement of data from one zonegroup to another. Configure a pipe for the zonegroups to define the buckets that can use this data flow. The system in the following examples includes 3 zones: `us-east` (the primary zone), `us-west` (secondary zone), and `us-west-2` (backup zone). Here, `us-west-2` is a replica of `us-west`, but data is not replicated back from it. Zonegroup sync policies are created with the new sync policy engine. Any change to the zonegroup sync policy requires a period update and a commit.

1. On the primary zone, create a sync group with the status set to `allowed`.

   ```
   radosgw-admin sync group create --group-id=GROUP_ID --status=allowed
   ```

   For example,

   ```
   [ceph: root@host01 /]# radosgw-admin sync group create --group-id=group1
          --status=allowed
   ```

   Note: Until a fully configured zonegroup replication policy is created, set the `--status` to `allowed` to prevent the replication from starting.

2. Create the flow.

   ```
   radosgw-admin sync group flow create --group-id=GROUP_ID --flow-id=FLOW_ID --flow-
   type=directional --source-zone=SOURCE_ZONE_NAME --dest-zone=DESTINATION_ZONE_NAME
   ```

   ```
   [ceph: root@host01 /]# radosgw-admin sync group flow create --group-id=group1 --flow-
   id=us-west-backup --flow-type=directional --source-zone=us-west --dest-zone=us-west-2
   ```

3. Create the pipe.

```
radosgw-admin sync group pipe create --group-id=GROUP_ID --pipe-id=PIPE_ID  --source-zones='SOURCE_ZONE_NAME' --dest-zones='DESTINATION_ZONE_NAME'
```

```
[ceph: root@host01 /]# radosgw-admin sync group pipe create --group-id=group1 --pipe-id=pipe1 --source-zones='us-west' --dest-zones='us-west-2'
```

4. Update and commit the new period.

```
radosgw-admin period update --commit
```

For example,

```
[ceph: root@host01 /]# radosgw-admin period update --commit
```

> **Note:** Updating and committing the period is mandatory for a zonegroup policy.

5. On the secondary zone, check the status of the buckets. The secondary zone has two destinations.

```
radosgw-admin sync info
```

For example,

```
[ceph: root@host01 /]# radosgw-admin sync info --bucket=buck
{
    "sources": [
        {
            "id": "pipe1",
            "source": {
                "zone": "us-east",
                "bucket": "buck:115b12b3-....4409.1"
            },
            "dest": {
                "zone": "us-west",
                "bucket": "buck:115b12b3-....4409.1"
            },
           ...
        }
    ],
    "dests": [
        {
            "id": "pipe1",
            "source": {
                "zone": "us-west",
                "bucket": "buck:115b12b3-....4409.1"
            },
            "dest": {
                "zone": "us-east",
                "bucket": "buck:115b12b3-....4409.1"
            },
           ...
        },
        {
            "id": "pipe1",
            "source": {
                "zone": "us-west",
                "bucket": "buck:115b12b3-....4409.1"
            },
            "dest": {
                "zone": "us-west-2",
                "bucket": "buck:115b12b3-....4409.1"
            },
           ...
        }
    ],
    ...
}
```

6. On the backup of the secondary zone, check the status of the buckets. The backup zone has only one source and no destinations.

```
radosgw-admin sync info --bucket=BUCKET_NAME
```

For example,

```
[ceph: root@host01 /]# radosgw-admin sync info --bucket=buck
{
    "sources": [
        {
            "id": "pipe1",
            "source": {
                "zone": "us-west",
                "bucket": "buck:115b12b3-....4409.1"
            },
            "dest": {
                "zone": "us-west-2",
                "bucket": "buck:115b12b3-....4409.1"
            },
           ...
        }
    ],
    "dests": [],
    ...
}
```

Setting bi-directional policy for buckets

Set the sync policy for buckets in both the directions symmetrically.

PREREQUISITE

At least two running Red Hat Ceph Storage 8.1 clusters with Ceph Object Gateway configured.
The data flow for the bucket-level policy is inherited from the zonegroup policy.
The data flow and pipes for bucket-level policy are a subset of the flow that is defined in the zonegroup policy. You need not change them.

Note:

- A bucket-level policy can `enable` pipes that are not enabled, except `forbidden`, at the zonegroup policy.
- Bucket-level policies do not require period updates.

1. Set the zonegroup policy `--status` to `allowed` to allow per-bucket replication.

   ```
   radosgw-admin sync group modify --group-id=GROUP_ID --status=allowed
   ```

   For example,

   ```
   [ceph: root@host01 /]# radosgw-admin sync group modify --group-id=group1 --
   status=allowed
   ```

2. Update the period.

   ```
   radosgw-admin period update --commit
   ```

   For example,

   ```
   [ceph: root@host01 /]# radosgw-admin period update --commit
   ```

3. Create a sync group for the bucket that you want to synchronize to and set `--status` to `enabled`.

   ```
   radosgw-admin sync group create --bucket=BUCKET_NAME --group-id=GROUP_ID --
   status=enabled
   ```

   For example,

```
[ceph: root@host01 /]# radosgw-admin sync group create --bucket=buck --group-id=buck-
default --status=enabled
```

4. Create a pipe for the group that was created in the previous step. The flow is inherited from the zonegroup level policy where data flow is symmetrical.

```
radosgw-admin sync group pipe create --bucket=BUCKET_NAME --group-id=GROUP_ID --pipe-
id=PIPE_ID --source-zones='*' --dest-zones='*'
```

For example,

```
[ceph: root@host01 /]# radosgw-admin sync group pipe create --bucket=buck --group-
id=buck-default --pipe-id=pipe1 --source-zones='*' --dest-zones='*'
```

> **Note:** Use wildcards * to specify the source and destination zones for the bucket replication.

5. Optional: To retrieve information about the expected bucket sync sources and targets, as defined by the sync policy, run the **radosgw-admin bucket sync info** command with the `--bucket` flag.

```
radosgw-admin bucket sync info --bucket BUCKET_NAME
```

For example,

```
[ceph: root@host01 /]# radosgw-admin bucket sync info --bucket buck
realm 33157555-f387-44fc-b4b4-3f9c0b32cd66 (india)
zonegroup 594f1f63-de6f-4e1e-90b6-105114d7ad55 (shared)
zone ffaa5ba4-c1bd-4c17-b176-2fe34004b4c5 (primary)
bucket :buck[ffaa5ba4-c1bd-4c17-b176-2fe34004b4c5.16191.1]

source zone e0e75beb-4e28-45ff-8d48-9710de06dcd0
bucket :buck[ffaa5ba4-c1bd-4c17-b176-2fe34004b4c5.16191.1]
```

6. Optional: To retrieve information about the expected sync sources and targets, as defined by the sync policy, run the **radosgw-admin sync info** command with the `--bucket` flag.

```
radosgw-admin sync info --bucket BUCKET_NAME
```

For example,

```
[ceph: root@host01 /]# radosgw-admin sync info --bucket buck
{
    "id": "pipe1",
    "source": {
        "zone": "secondary",
        "bucket": "buck:ffaa5ba4-c1bd-4c17-b176-2fe34004b4c5.16191.1"
    },
    "dest": {
        "zone": "primary",
        "bucket": "buck:ffaa5ba4-c1bd-4c17-b176-2fe34004b4c5.16191.1"
    },
    "params": {
        "source": {
            "filter": {
                "tags": []
            }
        },
        "dest": {},
        "priority": 0,
        "mode": "system",
        "user": ""
    }
},
    {
    "id": "pipe1",
    "source": {
        "zone": "primary",
        "bucket": "buck:ffaa5ba4-c1bd-4c17-b176-2fe34004b4c5.16191.1"
    },
    "dest": {
        "zone": "secondary",
        "bucket": "buck:ffaa5ba4-c1bd-4c17-b176-2fe34004b4c5.16191.1"
    },
    "params": {
        "source": {
            "filter": {
                "tags": []
            }
        },
        "dest": {},
        "priority": 0,
        "mode": "system",
        "user": ""
    }
}
```

Setting directional policy for buckets

Set the policy for buckets uni directionally with the sync policy engine.

PREREQUISITE

At least two running Red Hat Ceph Storage clusters with Ceph Object Gateway configured on the same version. In the following example, create a group policy and configure the data flow for the movement of data from one zone to another. Also, configure a pipe for the zonegroups to define the buckets that can use this data flow. The system in the following examples includes 3 zones: `us-east` (the primary zone), `us-west` (secondary zone), and `us-west-2` (backup zone). Here, `us-west-2` is a replica of `us-west`, but data is not replicated back from it.

The difference between setting the directional policy for zonegroups and buckets is that you need to specify the `--bucket` option.

1. On the primary zone, create a sync group with the status set to `allowed`.

   ```
   radosgw-admin sync group create --group-id=GROUP_ID --status=allowed
   ```

   For example,

   ```
   [ceph: root@host01 /]# radosgw-admin sync group create --group-id=group1 --
   status=allowed
   ```

> **Note:** Until a fully configured zonegroup replication policy is created, set the `--status` to `allowed` to prevent the replication from starting.

2. Create the flow.

```
radosgw-admin sync group flow create --bucket-name=BUCKET_NAME --group-id=GROUP_ID --
flow-id=FLOW_ID --flow-type=directional --source-zone=SOURCE_ZONE_NAME --dest-
zone=DESTINATION_ZONE_NAME
```

```
[ceph: root@host01 /]# radosgw-admin sync group flow create --bucket-name=buck --
group-id=group1 --flow-id=us-west-backup --flow-type=directional --source-zone=us-west
--dest-zone=us-west-2
```

3. Create the pipe.

```
radosgw-admin sync group pipe create --group-id=GROUP_ID --pipe-id=PIPE_ID  --source-
zones='SOURCE_ZONE_NAME' --dest-zones='DESTINATION_ZONE_NAME'
```

```
[ceph: root@host01 /]# radosgw-admin sync group pipe create --group-id=group1 --pipe-
id=pipe1 --source-zones='us-west' --dest-zones='us-west-2'
```

4. Verify source and destination of zonegroup using sync info in both the sites.

```
radosgw-admin sync info --bucket-name=BUCKET_NAME
```

Syncing between buckets

Sync data between source and destination buckets.

PREREQUISITE

At least two running Red Hat Ceph Storage 8.1 clusters with Ceph Object Gateway configured.
You can sync data between the source and destination buckets across zones, but not within the same zone.
Note that internally, data is still pulled from the source at the destination zone.

A wildcard bucket name means that the current bucket is in the context of the bucket sync policy.

There are two types of syncing between buckets:

- “Syncing from a different bucket” on page 112, where you need to specify the source bucket.
- “Syncing to a different bucket” on page 114, where you need to specify the destination bucket.

Syncing from a different bucket

1. Create a sync group to pull data from a bucket of another zone.

```
radosgw-admin sync group create  --bucket=BUCKET_NAME --group-id=GROUP_ID --
status=enabled
```

For example,

```
[ceph: root@host01 /]# radosgw-admin sync group create --bucket=buck4 --group-
id=buck4-default --status=enabled
```

2. Pull the data.

```
radosgw-admin sync group pipe create --bucket-name=BUCKET_NAME --group-id=GROUP_ID --
pipe-id=PIPE_ID --source-zones=SOURCE_ZONE_NAME --source-bucket=SOURCE_BUCKET_NAME --
dest-zones=DESTINATION_ZONE_NAME
```

In the following example the source bucket is `buck5`.

```
[ceph: root@host01 /]# radosgw-admin sync group pipe create --bucket=buck4 \
                          --group-id=buck4-default --pipe-id=pipe1 \
                          --source-zones='*' --source-bucket=buck5 \
                          --dest-zones='*'
```

3. Optional: Sync from buckets in specific zones.

```
radosgw-admin sync group pipe modify --bucket=BUCKET_NAME
```

For example,

```
[ceph: root@host01 /]# radosgw-admin sync group pipe modify --bucket=buck4 \
                          --group-id=buck4-default --pipe-id=pipe1 \
                          --source-zones=us-west --source-bucket=buck5 \
                          --dest-zones='*'
```

4. Check sync status.

```
radosgw-admin sync info --bucket-name=BUCKET_NAME
```

For example,

```
[ceph: root@host01 /]# radosgw-admin sync info --bucket=buck5
{
    "sources": [],
    "dests": [],
    "hints": {
        "sources": [],
        "dests": [
            "buck4:115b12b3-....14433.2"
        ]
    },
    "resolved-hints-1": {
        "sources": [],
        "dests": [
            {
                "id": "pipe1",
                "source": {
                    "zone": "us-west",
                    "bucket": "buck5"
                },
                "dest": {
                    "zone": "us-east",
                    "bucket": "buck4:115b12b3-....14433.2"
                },
                ...
            },
            {
                "id": "pipe1",
                "source": {
                    "zone": "us-west",
                    "bucket": "buck5"
                },
                "dest": {
                    "zone": "us-west-2",
                    "bucket": "buck4:115b12b3-....14433.2"
                },
                ...
            }
        ]
    },
    "resolved-hints": {
        "sources": [],
        "dests": []
    }
```

Note: Note that there are `resolved-hints`, which means that the bucket `buck5` found about `buck4` syncing from it indirectly, and not from its own policy. The policy for `buck5` itself is empty.

Syncing to a different bucket

1. Create a sync group.

```
radosgw-admin sync group create  --bucket=BUCKET_NAME --group-id=GROUP_ID --
status=enabled
```

   For example,

```
[ceph: root@host01 /]# radosgw-admin sync group create --bucket=buck6 --group-
id=buck6-default --status=enabled
```

2. Push the data.

```
radosgw-admin sync group pipe create --bucket-name=BUCKET_NAME --group-id=GROUP_ID --
pipe-id=PIPE_ID --source-zones=SOURCE_ZONE_NAME --dest-zones=DESTINATION_ZONE_NAME --
dest-bucket=DESTINATION_BUCKET_NAME
```

   In the following example the destination bucket is `buck5`.

```
[ceph: root@host01 /]# radosgw-admin sync group pipe create --bucket=buck6 \
                          --group-id=buck6-default --pipe-id=pipe1 \
                          --source-zones='*' --dest-zones='*' --dest-bucket=buck5
```

3. Optional: Sync to buckets in specific zones.

```
radosgw-admin sync group pipe modify --bucket=BUCKET_NAME
```

   For example,

```
[ceph: root@host01 /]# radosgw-admin sync group pipe modify --bucket=buck6 \
                          --group-id=buck6-default --pipe-id=pipe1 \
                          --source-zones='*' --dest-zones='us-west' --dest-
bucket=buck5
```

4. Check the sync status.

```
radosgw-admin sync info --bucket-name=BUCKET_NAME
```

   For example,

```
[ceph: root@host01 /]#  radosgw-admin sync info --bucket buck5
{
  "sources": [],
  "dests": [
    {
      "id": "pipe1",
      "source": {
        "zone": "us-west",
        "bucket": "buck6:c7887c5b-f6ff-4d5f-9736-aa5cdb4a15e8.20493.4"
      },
      "dest": {
        "zone": "us-east",
        "bucket": "buck5"
      },
      "params": {
        "source": {
          "filter": {
            "tags": []
          }
        },
        "dest": {},
        "priority": 0,
        "mode": "system",
        "user": "s3cmd"
      }
    },
  ],
  "hints": {
    "sources": [],
    "dests": [
      "buck5"
    ]
  },
  "resolved-hints-1": {
    "sources": [],
    "dests": []
  },
  "resolved-hints": {
    "sources": [],
    "dests": []
  }
}
```

Filtering objects

Filter objects within the bucket with prefixes and tags.

PREREQUISITE

At least two running Red Hat Ceph Storage 99 clusters with Ceph Object Gateway configured and buckets that are created.
You can set object filter at zonegroup level policy as well. If the `--bucket` option is used, then it is set at bucket level for a bucket. In the following example, sync from `buck1` bucket from one zone is synced to `buck1` bucket in another zone with the objects that start with the prefix `foo/`. Similarly, you can filter objects that have tags such as `color=blue`.
Prefixes and tags can be combined, in which objects need to have both in the order to be synced. The priority parameter can also be passed, and it is used to determine when multiple different rules are there that are matches. This configuration determines that rules to be used.

Note:

1. If the tag in sync policy has more than one tags, while syncing objects, it sync objects that match at least one tag, the key value pair. Objects might not match all the tag sets.
2. If the prefix and tag is set, then to sync the object to another zone, the objects must have prefix and any one tag should match. Only then, it syncs with each other.

1. On the primary zone, create a sync group. If you want to create at the bucket level, use the `--bucket` option.

```
radosgw-admin sync group create --bucket=BUCKET_NAME --group-id=GROUP_ID --
status=enabled
```

For example,

```
[ceph: root@host01 /]# radosgw-admin sync group create --bucket=buck1 --group-
id=buck8-default --status=enabled
```

2. Sync between buckets where the object matches the tags. The flow is inherited from the zonegroup level policy where data flow is symmetrical.

```
radosgw-admin sync group pipe create --bucket=BUCKET_NAME --group-id=GROUP_ID --pipe-
id=PIPE_ID --tags-add=KEY1=VALUE1,KEY2=VALUE2 --source-zones='ZONE_NAME1','ZONE_NAME2'
--dest-zones='ZONE_NAME1','ZONE_NAME2'
```

For example,

```
[ceph: root@host01 /]# radosgw-admin sync group pipe create --bucket=buck1 \
                         --group-id=buck1-default --pipe-id=pipe-prefix \
                         --tags-add=color=blue,color=red --source-zones='*' --dest-
zones='*'
```

3. Sync between buckets where the object matches the prefix. The flow is inherited from the zonegroup level policy where data flow is symmetrical.

```
radosgw-admin sync group pipe create --bucket=BUCKET_NAME --group-id=GROUP_ID --pipe-
id=PIPE_ID --prefix=PREFIX --source-zones='ZONE_NAME1','ZONE_NAME2' --dest-
zones='ZONE_NAME1','ZONE_NAME2'
```

For example,

```
[ceph: root@host01 /]# radosgw-admin sync group pipe create --bucket=buck1 \
                         --group-id=buck1-default --pipe-id=pipe-tags \
                         --prefix=foo/ --source-zones='*' \
                         --dest-zones='*' --dest-bucket=buck1
```

4. Check the updated sync.

```
radosgw-admin sync info --bucket=BUCKET_NAME
```

For example,

```
[ceph: root@host01 /]# radosgw-admin sync info --bucket=buck1
```

> **Note:** In the example, only two different destinations and no sources reflect, one each for configuration. When the sync process happens, it selects the relevant rule for each object it syncs.

Disabling policy between buckets

You can disable the policy between buckets along with sync information with the `forbidden` or `allowed` state.

PREREQUISITE

At least two running Red Hat Ceph Storage 99 clusters with Ceph Object Gateway and sync policy configured. See “Multi-site sync policy group state” on page 95for the different combinations that can used for zonegroup and bucket level sync policies. In certain cases, to interrupt the replication between two buckets, you can set the group policy for the bucket to be `forbidden`. You can also disable policy at zonegroup level, if the sync that is set does not happen for any of the buckets.

Note: You can also create a sync policy in disabled state with `allowed` or `forbidden` state using the `radosgw-admin sync group create` command.

1. Run the **`sync group modify`** command to change the status from `allowed` to `forbidden`.

```
radosgw-admin sync group modify --group-id  GROUP_ID --status forbidden --
bucket BUCKET_NAME
```

For example,

```
ceph: root@host01 /]# radosgw-admin sync group modify --group-id  buck-default --
status forbidden --bucket buck
{
    "groups": [
        {
            "id": "buck-default",
            "data_flow": {},
            "pipes": [
                {
                    "id": "pipe1",
                    "source": {
                        "bucket": "*",
                        "zones": [
                            "*"
                        ]
                    },
                    "dest": {
                        "bucket": "*",
                        "zones": [
                            "*"
                        ]
                    },
                    "params": {
                        "source": {
                            "filter": {
                                "tags": []
                            }
                        },
                        "dest": {},
                        "priority": 0,
                        "mode": "system",
                    }
                }
            ],
            "status": "forbidden"
        }
    ]
}
```

In this example, the replication of the bucket `buck` is interrupted between zones `us-east` and `us-west`.

Note: No update and commit for the period is required as this is a bucket sync policy.

2. Optional: Run **`sync info command`** command to check the status of the sync for bucket.

```
radosgw-admin sync info --bucket BUCKET_NAME
```

For example,

```
[ceph: root@host01 /]# radosgw-admin sync info --bucket buck
{
    "sources": [],
    "dests": [],
    "hints": {
        "sources": [],
        "dests": []
    },
    "resolved-hints-1": {
        "sources": [],
        "dests": []
    },
    "resolved-hints": {
        "sources": [],
        "dests": []
    }
}
```

Note: No source and destination targets are there as the replication is interrupted.

Feature specific configuration

As a storage administrator, you can configure some of the more advanced features of the Ceph Object Gateway. You can configure a multi-site Ceph Object Gateway and integrate it with directory services, such as Microsoft Active Directory and OpenStack Keystone service.

Bucket management

As a storage administrator, when using the Ceph Object Gateway you can manage buckets by moving them between users and renaming them. You can create bucket notifications to trigger on specific events. Also, you can find orphan or leaky objects within the Ceph Object Gateway that can occur over the lifetime of a storage cluster.

Note:

- When millions of objects are uploaded to a Ceph Object Gateway bucket with a high ingest rate, incorrect **num_objects** are reported with the **radosgw-admin bucket stats** command. With the **radosgw-admin bucket list** command you can correct the value of **num_objects** parameter.
- The **radosgw-admin bucket stats** command does not return the `Unknown error 2002` error and explicitly translates to POSIX error 2 such as the `No such file or directory` error.

Related information

Developer

Renaming buckets

Learn how to rename buckets.

PREREQUISITE

- A running Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway software.
- An existing bucket.

If you want to allow underscores in bucket names, then set the `rgw_relaxed_s3_bucket_names` option to `true`.

1. List the buckets.

```
radosgw-admin bucket list
```

For example,

```
[ceph: root@host01 /]# radosgw-admin bucket list
[
    "34150b2e9174475db8e191c188e920f6/swcontainer",
    "s3bucket1",
    "34150b2e9174475db8e191c188e920f6/swimpfalse",
    "c278edd68cfb4705bb3e07837c7ad1a8/ec2container",
    "c278edd68cfb4705bb3e07837c7ad1a8/demoten1",
    "c278edd68cfb4705bb3e07837c7ad1a8/demo-ct",
    "c278edd68cfb4705bb3e07837c7ad1a8/demopostup",
    "34150b2e9174475db8e191c188e920f6/postimpfalse",
    "c278edd68cfb4705bb3e07837c7ad1a8/demoten2",
    "c278edd68cfb4705bb3e07837c7ad1a8/postupsw"
]
```

2. Rename the bucket.

Note: If the bucket is inside a tenant also specify the tenant.

```
radosgw-admin bucket link --bucket=tenant/ORIGINAL_NAME --bucket-new-
name=NEW_NAME --uid=TENANT$USER_ID
```

```
radosgw-admin bucket link --bucket=ORIGINAL_NAME --bucket-new-name=NEW_NAME --
uid=USER_ID
```

For example,

```
[ceph: root@host01 /]# radosgw-admin bucket link --bucket=s3bucket1 --bucket-new-
name=s3newb --uid=testuser
```

3. Verify that the bucket was renamed.

```
radosgw-admin bucket list
```

For example,

```
[ceph: root@host01 /]# radosgw-admin bucket list
[
    "34150b2e9174475db8e191c188e920f6/swcontainer",
    "34150b2e9174475db8e191c188e920f6/swimpfalse",
    "c278edd68cfb4705bb3e07837c7ad1a8/ec2container",
    "s3newb",
    "c278edd68cfb4705bb3e07837c7ad1a8/demoten1",
    "c278edd68cfb4705bb3e07837c7ad1a8/demo-ct",
    "c278edd68cfb4705bb3e07837c7ad1a8/demopostup",
    "34150b2e9174475db8e191c188e920f6/postimpfalse",
    "c278edd68cfb4705bb3e07837c7ad1a8/demoten2",
    "c278edd68cfb4705bb3e07837c7ad1a8/postupsw"
]
```

Removing buckets

When the bucket does not have objects, you can run the `radosgw-admin bucket rm` command. For objects in the buckets, use the `--purge-objects` option to remove them.

For multi-site configuration, delete the buckets from the primary site.

For large-scale or ongoing data cleanup, implement an S3 Lifecycle (LC) policy instead of purging with `radosgw-admin`. LC policies apply good data-management practices and naturally throttle deletions because they run during the RGW lifecycle processing window. During each LC window:

- The lifecycle process marks objects for removal.

- Garbage collection later removes the deleted objects in smaller batches.
- The cycle repeats daily until the bucket is empty.

After all objects are removed, you can safely delete the bucket.

1. List the buckets.

   ```
   radosgw-admin bucket list
   ```

   For example,

   ```
   [ceph: root@host01 /]# radosgw-admin bucket list
   [
       "34150b2e9174475db8e191c188e920f6/swcontainer",
       "s3bucket1",
       "34150b2e9174475db8e191c188e920f6/swimpfalse",
       "c278edd68cfb4705bb3e07837c7ad1a8/ec2container",
       "c278edd68cfb4705bb3e07837c7ad1a8/demoten1",
       "c278edd68cfb4705bb3e07837c7ad1a8/demo-ct",
       "c278edd68cfb4705bb3e07837c7ad1a8/demopostup",
       "34150b2e9174475db8e191c188e920f6/postimpfalse",
       "c278edd68cfb4705bb3e07837c7ad1a8/demoten2",
       "c278edd68cfb4705bb3e07837c7ad1a8/postupsw"
   ]
   ```

2. Remove the bucket.
   Remove the bucket by using the **`radosgw-admin bucket rm`** command.
   When objects exist in the bucket, remove objects using an S3 Lifecycle policy and wait for garbage collection to complete, then delete the bucket (recommended for large buckets). When necessary, use the **`--purge-objects`** option.

   ```
   radosgw-admin bucket rm --bucket=BUCKET_NAME  [--purge-objects]
   ```

   > **Note:** Use **`--bypass-gc`** with caution.
   > You can optionally use the **`--bypass-gc`** flag, but use it **with caution**. This flag causes deletions to skip the garbage collection mechanism, which can negatively impact cluster health and overall performance. The garbage collection process exists to regulate deletion workloads and maintain cluster stability. Skipping it removes that protection and can lead to performance degradation, especially during large or rapid deletion operations.

   For example,

   ```
   [ceph: root@host01 /]# radosgw-admin bucket rm --bucket=s3bucket1
   ```

AFTER COMPLETING THE TASK

Verify that the bucket was removed.

```
[ceph: root@host01 /]# radosgw-admin bucket list
[
    "34150b2e9174475db8e191c188e920f6/swcontainer",
    "34150b2e9174475db8e191c188e920f6/swimpfalse",
    "c278edd68cfb4705bb3e07837c7ad1a8/ec2container",
    "c278edd68cfb4705bb3e07837c7ad1a8/demoten1",
    "c278edd68cfb4705bb3e07837c7ad1a8/demo-ct",
    "c278edd68cfb4705bb3e07837c7ad1a8/demopostup",
    "34150b2e9174475db8e191c188e920f6/postimpfalse",
    "c278edd68cfb4705bb3e07837c7ad1a8/demoten2",
    "c278edd68cfb4705bb3e07837c7ad1a8/postupsw"
]
```

Moving buckets

The `radosgw-admin bucket` utility provides the ability to move buckets between users. To do so, link the bucket to a new user and change the ownership of the bucket to the new user.

You can move buckets:

- Between two non-tenanted users
- Between two tenanted users
- Between a non-tenanted user to a tenanted user

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Ceph Object Gateway is installed.
- An S3 bucket.
- Various tenanted and non-tenanted users.

Moving buckets between non-tenanted users

The **`radosgw-admin bucket chown`** command provides the ability to change the ownership of buckets and all objects they contain from one user to another.

To move buckets between non-tenant users, unlink a bucket from the current user, link it to a new user, and change the ownership of the bucket to the new user.

Procedure

1. Link the bucket to a new user:

   Syntax

   ```
   radosgw-admin bucket link --uid=USER --bucket=BUCKET
   ```

   Example

   ```
   [ceph: root@host01 /]# radosgw-admin bucket link --uid=user2 --bucket=data
   ```

2. Verify that the bucket has been linked to `user2` successfully:

   Example

   ```
   [ceph: root@host01 /]# radosgw-admin bucket list --uid=user2
   [
       "data"
   ]
   ```

3. Change the ownership of the bucket to the new user:

   Syntax

   ```
   radosgw-admin bucket chown --uid=user --bucket=bucket
   ```

   Example

   ```
   [ceph: root@host01 /]# radosgw-admin bucket chown --uid=user2 --bucket=data
   ```

4. Verify that the ownership of the `data` bucket has been successfully changed by checking the `owner` line in the output of the following command:

   Example

   ```
   [ceph: root@host01 /]# radosgw-admin bucket list --bucket=data
   ```

Moving buckets between tenanted users

You can move buckets between one tenanted user and another.

Procedure

1. Link the bucket to a new user:

   Syntax

   ```
   radosgw-admin bucket link --bucket=CURRENT_TENANT/BUCKET --uid=NEW_TENANT$USER
   ```

   Example

   ```
   [ceph: root@host01 /]# radosgw-admin bucket link --bucket=test/data --uid=test2$user2
   ```

2. Verify that the bucket has been linked to `user2` successfully:

   ```
   [ceph: root@host01 /]# radosgw-admin bucket list --uid=test$user2
   [
       "data"
   ]
   ```

3. Change the ownership of the bucket to the new user:

   Syntax

   ```
   radosgw-admin bucket chown --bucket=NEW_TENANT/BUCKET --uid=NEW_TENANT$USER
   ```

   Example

   ```
   [ceph: root@host01 /]# radosgw-admin bucket chown --bucket='test2/data' --
   uid='test$tuser2'
   ```

4. Verify that the ownership of the `data` bucket has been successfully changed by checking the `owner` line in the output of the following command:

   ```
   [ceph: root@host01 /]# radosgw-admin bucket list --bucket=test2/data
   ```

Moving buckets from non-tenanted users to tenanted users

You can move buckets from a non-tenanted user to a tenanted user.

Procedure

1. Optional: If you do not already have multiple tenants, you can create them by enabling `rgw_keystone_implicit_tenants` and accessing the Ceph Object Gateway from an external tenant:

   Enable the `rgw_keystone_implicit_tenants` option:

   Example

   ```
   [ceph: root@host01 /]# ceph config set client.rgw rgw_keystone_implicit_tenants true
   ```

   Access the Ceph Object Gateway from an eternal tenant using either the `s3cmd` or `swift` command:

   Example

   ```
   [ceph: root@host01 /]# swift list
   ```

   Or use `s3cmd`:

   Example

```
[ceph: root@host01 /]# s3cmd ls
```

The first access from an external tenant creates an equivalent Ceph Object Gateway user.

2. Move a bucket to a tenanted user:

   Syntax

   ```
   radosgw-admin bucket link --bucket=/BUCKET --uid=TENANT$USER
   ```

   Example

   ```
   [ceph: root@host01 /]# radosgw-admin bucket link --bucket=/data --uid='test$tenanted-
   user'
   ```

3. Verify that the `data` bucket has been linked to `tenanted-user` successfully:

   Example

   ```
   [ceph: root@host01 /]# radosgw-admin bucket list --uid='test$tenanted-user'
   [
       "data"
   ]
   ```

4. Change the ownership of the bucket to the new user:

   Syntax

   ```
   radosgw-admin bucket chown --bucket=tenant/bucket name --uid=tenant$user
   ```

   Example

   ```
   [ceph: root@host01 /]# radosgw-admin bucket chown --bucket='test/data' --
   uid='test$tenanted-user'
   ```

5. Verify that the ownership of the `data` bucket has been successfully changed by checking the `owner` line in the output of the following command:

   Example

   ```
   [ceph: root@host01 /]# radosgw-admin bucket list --bucket=test/data
   ```

Finding orphan and leaky objects

A healthy storage cluster does not have any orphan or leaky objects, but in some cases orphan or leaky objects can occur. Use this information to learn how to find orphan and leaky objects.

An orphan object exists in a storage cluster and has an object ID associated with the RADOS object. However, there is no reference of the RADOS object with the S3 object in the bucket index reference.

For example, if the Ceph Object Gateway goes down in the middle of an operation, this can cause some objects to become orphans. Also, an undiscovered bug can cause orphan objects to occur.

You can see how the Ceph Object Gateway objects map to the RADOS objects. The `radosgw-admin` command provides a tool to search for and produce a list of these potential orphan or leaky objects. Using the `radoslist` subcommand displays objects stored within buckets, or all buckets in the storage cluster. The `rgw-orphan-list` script displays orphan objects within a pool.

> **Note:** The **`radoslist`** subcommand is replacing the deprecated **`orphans find`** and **`orphans finish`** subcommands.

Important: Do not use this command where `Indexless` buckets are in use as all the objects appear as `orphaned`.

Important: Another alternate way to identity orphaned objects is to run the **`rados -p <pool> ls | grep BUCKET_ID`** command.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- A running Ceph Object Gateway.

Procedure

1. Generate a list of objects that hold data within a bucket.

   Syntax

   ```
   radosgw-admin bucket radoslist --bucket BUCKET_NAME
   ```

   Example

   ```
   [root@host01 ~]# radosgw-admin bucket radoslist --bucket mybucket
   ```

   Note: If the *BUCKET_NAME* is omitted, then all objects in all buckets are displayed.

2. Check the version of `rgw-orphan-list`.

   Example

   ```
   [root@host01 ~]# head /usr/bin/rgw-orphan-list
   ```

   The version should be `2023-01-11` or newer.

3. Create a directory where you need to generate the list of orphans.

   Example

   ```
   [root@host01 ~]# mkdir orphans
   ```

4. Navigate to the directory created earlier.

   Example

   ```
   [root@host01 ~]# cd orphans
   ```

5. From the pool list, select the pool in which you want to find orphans. This script might run for a long time depending on the objects in the cluster.

   Example

   ```
   [root@host01 orphans]# rgw-orphan-list
   ```

   Example

   ```
   Available pools:
       .rgw.root
   ```

```
    default.rgw.control
    default.rgw.meta
    default.rgw.log
    default.rgw.buckets.index
    default.rgw.buckets.data
    rbd
    default.rgw.buckets.non-ec
    ma.rgw.control
    ma.rgw.meta
    ma.rgw.log
    ma.rgw.buckets.index
    ma.rgw.buckets.data
    ma.rgw.buckets.non-ec
Which pool do you want to search for orphans?
```

Enter the pool name to search for orphans.

> **Important:** A data pool must be specified when using the `rgw-orphan-list` command, and not a metadata pool.

6. View the details of the `rgw-orphan-list` tool usage.

   Syntax

   ```
   rgw-orphan-list -h
   rgw-orphan-list POOL_NAME /DIRECTORY
   ```

   Example

   ```
   [root@host01 orphans]# rgw-orphan-list default.rgw.buckets.data /orphans

   2023-09-12 08:41:14 ceph-host01 Computing delta...
   2023-09-12 08:41:14 ceph-host01 Computing results...
   10 potential orphans found out of a possible 2412 (0%).         <<<<<<< orphans
  detected
   The results can be found in './orphan-list-20230912124113.out'.
       Intermediate files are './rados-20230912124113.intermediate' and './radosgw-
  admin-20230912124113.intermediate'.
   ***
   *** WARNING: This is EXPERIMENTAL code and the results should be used
   ***          only with CAUTION!
   ***
   Done at 2023-09-12 08:41:14.
   ```

7. Run the `ls -l` command to verify the files ending with error should be zero length indicating the script ran without any issues.

   Example

   ```
   [root@host01 orphans]# ls -l

   -rw-r--r--. 1 root root    770 Sep 12 03:59 orphan-list-20230912075939.out
   -rw-r--r--. 1 root root      0 Sep 12 03:59 rados-20230912075939.error
   -rw-r--r--. 1 root root 248508 Sep 12 03:59 rados-20230912075939.intermediate
   -rw-r--r--. 1 root root      0 Sep 12 03:59 rados-20230912075939.issues
   -rw-r--r--. 1 root root      0 Sep 12 03:59 radosgw-admin-20230912075939.error
   -rw-r--r--. 1 root root 247738 Sep 12 03:59 radosgw-admin-20230912075939.intermediate
   ```

8. Review the orphan objects listed.

   Example

   ```
   [root@host01 orphans]# cat ./orphan-list-20230912124113.out

   a9c042bc-be24-412c-9052-dda6b2f01f55.16749.1_key1.cherylf.433-bucky-4865-0.0
   a9c042bc-be24-412c-9052-dda6b2f01f55.16749.1_key1.cherylf.433-bucky-4865-0.1
   a9c042bc-be24-412c-9052-dda6b2f01f55.16749.1_key1.cherylf.433-bucky-4865-0.2
   a9c042bc-be24-412c-9052-dda6b2f01f55.16749.1_key1.cherylf.433-bucky-4865-0.3
   a9c042bc-be24-412c-9052-dda6b2f01f55.16749.1_key1.cherylf.433-bucky-4865-0.4
   a9c042bc-be24-412c-9052-dda6b2f01f55.16749.1_key1.cherylf.433-bucky-4865-0.5
   ```

```
a9c042bc-be24-412c-9052-dda6b2f01f55.16749.1_key1.cherylf.433-bucky-4865-0.6
a9c042bc-be24-412c-9052-dda6b2f01f55.16749.1_key1.cherylf.433-bucky-4865-0.7
a9c042bc-be24-412c-9052-dda6b2f01f55.16749.1_key1.cherylf.433-bucky-4865-0.8
a9c042bc-be24-412c-9052-dda6b2f01f55.16749.1_key1.cherylf.433-bucky-4865-0.9
```

9. Remove orphan objects:

   Syntax

   ```
   rados -p POOL_NAME rm OBJECT_NAME
   ```

   Example

   ```
   [root@host01 orphans]# rados -p default.rgw.buckets.data rm myobject
   ```

   Warning: Verify you are removing the correct objects. Running the `rados rm` command removes data from the storage cluster.

Managing bucket index entries

Each bucket index entry related to a piece of a multipart upload object is matched against its corresponding `.meta` index entry. There should be one `.meta` entry for all the pieces of a given multipart upload. If it fails to find a corresponding `.meta` entry for a piece, it lists out the "orphaned" piece entries in a section of the output.

The stats for the bucket are stored in the bucket index headers. This phase loads those headers and also iterates through all the plain object entries in the bucket index and recalculates the stats. It then displays the actual and calculated stats in sections labeled "existing_header" and "calculated_header" respectively, so they can be compared.

If you use the `--fix` option with the `bucket check` sub-command, it removes the "orphaned" entries from the bucket index and also overwrites the existing stats in the header with those that it calculated. It causes all entries, including the multiple entries used in versioning, to be listed in the output.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- A running Ceph Object Gateway.
- A newly created bucket.

Procedure

1. Check the bucket index of a specific bucket:

   Syntax

   ```
   radosgw-admin bucket check --bucket=BUCKET_NAME
   ```

   Example

   ```
   [root@rgw ~]# radosgw-admin bucket check --bucket=mybucket
   ```

2. Fix the inconsistencies in the bucket index, including removal of orphaned objects:

   Syntax

   ```
   radosgw-admin bucket check --fix --bucket=BUCKET_NAME
   ```

   Example

   ```
   [root@rgw ~]# radosgw-admin bucket check --fix --bucket=mybucket
   ```

Bucket notifications

Bucket notifications provide a way to send information out of the Ceph Object Gateway when certain events happen in the bucket.

Bucket notifications can be sent to HTTP, AMQP0.9.1, and Kafka endpoints. A notification entry must be created to send bucket notifications for events on a specific bucket and to a specific topic. A bucket notification can be created on a subset of event types or by default for all event types. The bucket notification can filter out events based on key prefix or suffix, regular expression matching the keys, and on the metadata attributes attached to the object, or the object tags. Bucket notifications have a REST API to provide configuration and control interfaces for the bucket notification mechanism.

> **Note:** The bucket notifications API is enabled by default. If **`rgw_enable_apis`** configuration parameter is explicitly set, ensure that `s3`, and `notifications` are included. To verify, run the following command, replacing *NAME* with the Ceph Object Gateway instance name.
>
> ```
> ceph --admin-daemon /var/run/ceph/ceph-client.rgw.NAME.asok config get rgw_enable_apis
> ```

By default, bucket notifications and topics that saved in fresh installation deployments (Greenfield) have their information synced between zones, due to the new notification_v2 feature. This provides enhanced troubleshooting of persistent topic issues. When upgrading from release versions earlier than Red Hat Ceph Storage 8.0, this enhancement needs to be added by enabling the `notification_v2` feature. Trigger a migration of any existing bucket notifications and topic configurations to the new format by running the **`zonegroup modify`** command.

```
radosgw-admin zonegroup modify --enable-feature=notification_v2
radosgw-admin period update
radosgw-admin period commit
```

Topic management using CLI

You can manage list, get, and remove topics for the Ceph Object Gateway buckets.

Topic management for Ceph Object Gateway buckets

Action	Description	Command example
List topics	Run the command to list the configuration of all topics.	`[ceph: host01 /]# radosgw-admin topic list`
Get topics	Run the command to get the configuration of a specific topic.	`[ceph: host01 /]# radosgw-admin topic get --topic=topic1`
Remove topics	Run the command to remove the configuration of a specific topic. **Note:** The topic is removed even if the Ceph Object Gateway bucket is configured to that topic.	`[ceph: host01 /]# radosgw-admin topic rm --topic=topic1`

Creating bucket notifications

Bucket notifications are S3 operations that can be created at the bucket level. The notification configuration has the Red Hat Ceph Storage Object Gateway S3 events, `ObjectCreated`, `ObjectRemoved`, and `ObjectLifecycle:Expiration`. The configurations need to be published with the destination to send the bucket notifications.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster with Ceph Object Gateway.
- A running HTTP Server, a RabbitMQ server, or a Kafka server.
- Root-level access.
- User access key and secret key.
- Endpoint parameters.

> **Important:** Red Hat supports `ObjectCreate` events, such as `put`, `post`, `multipartUpload`, and `copy`. Red Hat also supports `ObjectRemove` events, such as `object_delete` and `s3_multi_object_delete`.

Create bucket notifications in one of two ways:

- “Creating bucket notifications by using the boto script” on page 128
- “Creating bucket notifications by using the AWS CLI” on page 129

Creating bucket notifications by using the boto script

1. Install the `python3-boto3` package.

   ```
   dnf install python3-boto3
   ```

   For example,

   ```
   [user@client ~]$  dnf install python3-boto3
   ```

2. Create an S3 bucket.
3. Create a python script `topic.py` to create an SNS topic for `http`,`amqp`, or `kafka` protocol.
   For example,

   ```
   import boto3
   from botocore.client import Config
   import sys

   # endpoint and keys from vstart
   endpoint = 'http://127.0.0.1:8000'
   access_key='0555b35654ad1656d804'
   secret_key='h7GhxuBLTrlhVUyxSPUKUV8r/2EI4ngqJxD7iBdBYLhwluN30JaT3Q=='

   client = boto3.client('sns',
           endpoint_url=endpoint,
           aws_access_key_id=access_key,
           aws_secret_access_key=secret_key,
           config=Config(signature_version='s3'))

   attributes = {"push-endpoint": "amqp://localhost:5672", "amqp-exchange": "ex1", "amqp-
   ack-level": "broker"}

   client.create_topic(topic_name="mytopic", Attributes=attributes)
   ```

4. Run the python script for creating topic.

   ```
   python3 topic.py
   ```

5. Create a python script `notification.py` to create S3 bucket notification for `s3:objectCreate`, `s3:objectRemove`, and `s3:ObjectLifecycle:Expiration` events.
   For example,

```
import boto3
import sys

# bucket name as first argument
bucketname = sys.argv[1]
# topic ARN as second argument
topic_arn = sys.argv[2]
# notification id as third argument
notification_id = sys.argv[3]

# endpoint and keys from vstart
endpoint = 'http://127.0.0.1:8000'
access_key='0555b35654ad1656d804'
secret_key='h7GhxuBLTrlhVUyxSPUKUV8r/2EI4ngqJxD7iBdBYLhwluN30JaT3Q=='

client = boto3.client('s3',
        endpoint_url=endpoint,
        aws_access_key_id=access_key,
        aws_secret_access_key=secret_key)

# regex filter on the object name and metadata based filtering are extension to AWS S3
API
# bucket and topic should be created beforehand

topic_conf_list = [{'Id': notification_id,
                    'TopicArn': topic_arn,
                    'Events': ['s3:ObjectCreated:*', 's3:ObjectRemoved:*'],
                    }]
    client.put_bucket_notification_configuration(
        Bucket=bucketname,
          NotificationConfiguration={
            'TopicConfigurations': [
              {
                'Id': notification_id,
                'TopicArn': topic_arn,
                'Events': ['s3:ObjectCreated:*', 's3:ObjectRemoved:*',
's3:ObjectLifecycle:Expiration:*']
             }]})
```

6. Run the python script for creating the bucket notification.

```
python3 notification.py
```

7. Create S3 objects in the bucket.
8. Fetch the notification configuration.

```
import boto3
endpoint = 'http://127.0.0.1:8000'
access_key='0555b35654ad1656d804'
secret_key='h7GhxuBLTrlhVUyxSPUKUV8r/2EI4ngqJxD7iBdBYLhwluN30JaT3Q=='

client = boto3.client('s3',
        endpoint_url=endpoint,
        aws_access_key_id=access_key,
        aws_secret_access_key=secret_key)

# getting a specific notification configuration is an extension to AWS S3 API

print(client.get_bucket_notification_configuration(Bucket=bucketname))
```

9. Optional: Delete the objects.
   Verify the object deletion events at the `http`, `rabbitmq`, or `kafka` receiver.

Creating bucket notifications by using the AWS CLI

1. Create a topic.

```
aws --endpoint=AWS_END_POINT sns create-topic --name NAME --attributes=ATTRIBUTES_FILE
```

   For example,

```
[user@client ~]$ aws --endpoint=http://localhost sns create-topic --name test-kafka --
attributes=file://topic.json

 sample topic.json:
 {"push-endpoint": "kafka://localhost","verify-ssl": "False", "kafka-ack-level":
"broker", "persistent":"true"}
 ref: https://docs.aws.amazon.com/cli/latest/reference/sns/create-topic.html
```

2. Create the bucket notification.

```
aws s3api put-bucket-notification-configuration --bucket BUCKET_NAME --notification-
configuration NOTIFICATION_FILE
```

For example,

```
[user@client ~]$ aws s3api put-bucket-notification-configuration --bucket my-bucket --
notification-configuration file://notification.json

 sample notification.json
 {
     "TopicConfigurations": [
         {
             "Id": "test_notification",
             "TopicArn": "arn:aws:sns:us-west-2:123456789012:test-kafka",
             "Events": [
                 "s3:ObjectCreated:*"
             ]
         }
     ]
 }
```

3. Fetch the notification configuration.

```
aws s3api --endpoint=AWS_ENDPOINT get-bucket-notification-configuration --
bucket BUCKET_NAME
```

For example,

```
[user@client ~]$ aws s3api --endpoint=http://localhost get-bucket-notification-
configuration --bucket my-bucket
 {
     "TopicConfigurations": [
         {
             "Id": "test_notification",
             "TopicArn": "arn:aws:sns:default::test-kafka",
             "Events": [
                 "s3:ObjectCreated:*"
             ]
         }
     ]
 }
```

S3 bucket replication API

Create replication between different buckets in a storage cluster with Ceph Object Gateway configuration.

The S3 bucket replication API is implemented, and allows users to create replication rules between different buckets. While the AWS replication feature allows bucket replication within the same zone, Ceph Object Gateway does not allow it at the moment. However, the Ceph Object gateway API also added a `Zone` array that allows users to select to what zones the specific bucket is synced.

Creating S3 bucket replication

Create a replication configuration for a bucket or replace an existing one.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage 9 cluster with Multi-site Ceph object Gateway configured. For more information on creating multi-site sync policies, see Creating a sync policy group.

- Zonegroup-level policy is created. For more information on creating zone group policies, see Bucket granular sync policies.

You can create a replication configuration for a bucket or replace an existing one. Specify the replication configuration in the request body. Provide the name of the destination bucket or buckets where you want to replicate objects.

A replication configuration must include at least one rule. Each rule identifies a subset of objects to replicate by filtering the objects in the source bucket.

1. Create a replication configuration file that contains the details of replication.

```
{
    "Role": "arn:aws:iam::account-id:role/role-name",
    "Rules": [
        {
            "ID": "String",
            "Status": "Enabled",
            "Priority": 1,
            "DeleteMarkerReplication": { "Status": "Enabled"|"Disabled" },
            "Destination": {
                "Bucket": "BUCKET_NAME"
            }
        }
    ]
}
```

   For example,

```
[root@host01 ~]# cat replication.json
{
    "Role": "arn:aws:iam::account-id:role/role-name",
    "Rules": [
        {
            "ID": "pipe-bkt",
            "Status": "Enabled",
            "Priority": 1,
            "DeleteMarkerReplication": { "Status": "Disabled" },
            "Destination": {
                "Bucket": "testbucket"
            }
        }
    ]
}
```

2. Create the S3 API put bucket replication.

```
aws --endpoint-url=RADOSGW_ENDPOINT_URL s3api put-bucket-replication --
bucket BUCKET_NAME --replication-configuration file://
REPLICATION_CONFIIRATION_FILE.json
```

   For example,

```
[root@host01 ~]# aws --endpoint-url=http://host01:80 s3api put-bucket-replication --
bucket testbucket --replication-configuration file://replication.json
```

AFTER COMPLETING THE TASK

- Verify the sync policy, by using the **`sync policy get`** command.
  Syntax

```
radosgw-admin sync policy get --bucket BUCKET_NAME
```

> **Note:** When applying replication policy, the rules are converted to sync-policy rules, that are known as *pipes*, and are categorized as`enabled`and`disabled`.
>
> **Enabled**
>
> These pipes are enabled and are active and the group status is set to `rgw_sync_policy_group:STATUS`.

Example,

s3-bucket-replication:enabled.

Disabled

The pipes under this set are not active and the group status is set to `rgw_sync_policy_group:STATUS`.
Example,

s3-bucket-replication:disabled.

Since there can be multiple configurable rules as part of replication policy, it has two separate groups (one with`enabled`and another with`allowed`state) for accurate mapping of each group.

Example

```
[host01 /]# radosgw-admin sync policy get --bucket testbucket

{
    "groups": [
        {
            "id": "s3-bucket-replication:disabled",
            "data_flow": {},
            "pipes": [],
            "status": "allowed"
        },
        {
            "id": "s3-bucket-replication:enabled",
            "data_flow": {},
            "pipes": [
                {
                    "id": "pipe-bkt",
                    "source": {
                        "bucket": "*",
                        "zones": [
                            "*"
                        ]
                    },
                    "dest": {
                        "bucket": "testbucket",
                        "zones": [
                            "*"
                        ]
                    },
                    "params": {
                        "source": {},
                        "dest": {},
                        "priority": 1,
                        "mode": "user",
                        "user": "s3cmd"
                    }
                }
            ],
            "status": "enabled"
        }
    ]
}
```

Related information

Using multi-site sync policies

Getting S3 bucket replication

Get a replication configuration for a bucket.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage 99 cluster with Multi-site Ceph object Gateway configured. For more information on creating multi-site sync policies, see Creating a sync policy group.

- Zonegroup-level policy is created. For more information on creating zone group policies, see Bucket granular sync policies.
- An S3 bucket replication created.

You can retrieve the replication configuration of the bucket.

- Get the S3 API put bucket replication.

```
aws s3api get-bucket-replication --bucket BUCKET_NAME --endpoint-
url=RADOSGW_ENDPOINT_URL
```

For example,

```
[root@host01 ~]# aws s3api get-bucket-replication --bucket testbucket --endpoint-
url=http://host01:80
{

"ReplicationConfiguration": {
    "Role": "",
    "Rules": [
        {
            "ID": "pipe-bkt",
            "Status": "Enabled",
            "Priority": 1,
            "Destination": {
                "Bucket": "testbucket"
            }
        }
    ]
  }

}
```

Deleting S3 bucket replication

Delete a replication configuration from a bucket.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage 99 cluster with Multi-site Ceph object Gateway configured. For more information on creating multi-site sync policies, see Creating a sync policy group.
- Zonegroup-level policy is created. For more information on creating zone group policies, see Bucket granular sync policies.

You can delete a replication configuration from a bucket. The bucket owner can grant permission to others to remove the replication configuration.

> **Note:** It can take a while to delete the replication configuration.

- Delete the S3 API put bucket replication.

```
aws s3api delete-bucket-replication --bucket BUCKET_NAME --endpoint-
url=RADOSGW_ENDPOINT_URL
```

For example,

```
[root@host01 ~]# aws s3api delete-bucket-replication --bucket testbucket --endpoint-
url=http://host01:80
```

AFTER COMPLETING THE TASK

- Verify that the existing replication rules are deleted.

```
radosgw-admin sync policy get --bucket=BUCKET_NAME
```

For example,

```
[ceph: root@host01 /]# radosgw-admin sync policy get --bucket=testbucket
```

Disabling S3 bucket replication for a user

As an administrator, you can set a user policy for other users to restrict them from performing any S3 replication API operations on buckets that reside under that particular user/users.

PREREQUISITE

- A running Red Hat Ceph Storage 99 cluster with Multi-site Ceph object Gateway configured. For more information on creating multi-site sync policies, see Creating a sync policy group.
- Zonegroup-level policy is created. For more information on creating zone group policies, see Bucket granular sync policies.

1. Create a user policy configuration file to deny access to the S3 bucket replication API.
   For example,

```
[root@host01 ~]# cat user_policy.json
{
    "Version":"2012-10-17",
    "Statement":
    {
        "Effect":"Deny",
        "Action":
        [
            "s3:PutReplicationConfiguration",
            "s3:GetReplicationConfiguration",
            "s3:DeleteReplicationConfiguration"
        ],
        "Resource": "arn:aws:s3:::*",
    }
}
```

2. As an admin user, set a user policy to user to disable user access to the S3 API.

```
aws --endpoint-url=ENDPOINT_URL iam put-user-policy --user-name USER_NAME --policy-
name USER_POLICY_NAME --policy-document POLICY_DOCUMENT_PATH
```

   For example,

```
[root@host01 ~]# aws --endpoint-url=http://host01:80 iam put-user-policy --user-name
newuser1 --policy-name userpolicy --policy-document file://user_policy.json
```

AFTER COMPLETING THE TASK

- As an admin, verify the user policy set.

```
aws --endpoint-url=ENDPOINT_URL iam get-user-policy --user-name USER_NAME --policy-
name USER_POLICY_NAME --region us
```

  For example,

```
[root@host01 ~]# aws --endpoint-url=http://host01:80 iam get-user-policy --user-name
newuser1 --policy-name userpolicy --region us
```

- As a user, perform the below S3 bucket replication API operations to verify whether the action is denied as expected.
  - Creating S3 bucket replication
  - “Getting S3 bucket replication” on page 132
  - “Deleting S3 bucket replication” on page 133

Configuring Bucket lifecycle

As a storage administrator, you can use a bucket lifecycle configuration to manage your objects so they are stored effectively throughout their lifetime. For example, you can transition objects to less expensive storage classes, archive, or even delete them based on your use case.

RADOS Gateway supports S3 API object expiration by using rules defined for a set of bucket objects. Each rule has a prefix, which selects the objects, and a number of days after which objects become unavailable.

Creating a lifecycle management policy

You can manage a bucket lifecycle policy configuration using standard S3 operations rather than using the **`radosgw-admin`** command. RADOS Gateway supports only a subset of the Amazon S3 API policy language applied to buckets. The lifecycle configuration contains one or more rules defined for a set of bucket objects.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway.
- Root-level access to a Ceph Object Gateway node.
- An S3 bucket created.
- An S3 user created with user access.
- Access to a Ceph Object Gateway client with the `AWS CLI` package installed.

Procedure

1. Create a JSON file for lifecycle configuration:

   Example

   ```
   [user@client ~]$ vi lifecycle.json
   ```

2. Add the specific lifecycle configuration rules in the file:

   Example

   ```
   {
           "Rules": [
           {
                       "Filter": {
                               "Prefix": "images/"
                       },
                       "Status": "Enabled",
                       "Expiration": {
                               "Days": 1
                       },
                       "ID": "ImageExpiration"
               }
       ]
   }
   ```

   The lifecycle configuration example expires objects in the images directory after 1 day.

3. Set the lifecycle configuration on the bucket:

   Syntax

   ```
   aws --endpoint-url=_RADOSGW_ENDPOINT_URL_:PORT s3api put-bucket-lifecycle-
   configuration --bucket _BUCKET_NAME_ --lifecycle-configuration file://
   _PATH_TO_LIFECYCLE_CONFIGURATION_FILE_/_LIFECYCLE_CONFIGURATION_FILE_.json
   ```

   Example

```
[user@client ~]$ aws --endpoint-url=http://host01:80 s3api put-bucket-lifecycle-
configuration --bucket testbucket --lifecycle-configuration file://lifecycle.json
```

In this example, the `lifecycle.json` file exists in the current directory.

- Retrieve the lifecycle configuration for the bucket:

  Syntax

```
aws --endpoint-url=_RADOSGW_ENDPOINT_URL_:PORT s3api get-bucket-lifecycle-configuration
--bucket _BUCKET_NAME_
```

  Example

```
[user@client ~]$ aws --endpoint-url=http://host01:80 s3api get-bucket-lifecycle-
configuration --bucket testbucket
{
        "Rules": [
        {
                    "Expiration": {
                            "Days": 1
                    },
                    "ID": "ImageExpiration",
                    "Filter": {
                            "Prefix": "images/"
                    },
                    "Status": "Enabled"
            }
    ]
}
```

- Optional: From the Ceph Object Gateway node, log into the Cephadm shell and retrieve the bucket lifecycle configuration:

  Syntax

```
radosgw-admin lc get --bucket=BUCKET_NAME
```

  Example

```
[ceph: root@host01 /]# radosgw-admin lc get --bucket=testbucket
{
        "prefix_map": {
                "images/": {
                        "status": true,
                        "dm_expiration": false,
                        "expiration": 1,
                        "noncur_expiration": 0,
                        "mp_expiration": 0,
                        "transitions": {},
                        "noncur_transitions": {}
                }
        },
        "rule_map": [
        {
                "id": "ImageExpiration",
                "rule": {
                        "id": "ImageExpiration",
                        "prefix": "",
                        "status": "Enabled",
                        "expiration": {
                                "days": "1",
                                "date": ""
                        },
                        "mp_expiration": {
                                "days": "",
                                "date": ""
                        },
                        "filter": {
                                "prefix": "images/",
                                "obj_tags": {
                                        "tagset": {}
                                }
                        },
```

```
                        "transitions": {},
                        "noncur_transitions": {},
                        "dm_expiration": false
                }
        }
  ]
}
```

Reference

For more information, see the following:

- S3 bucket lifecycle
- *Setting lifecycle configuration on a bucket*

Deleting a lifecycle management policy

You can delete the lifecycle management policy for a specified bucket by using the **s3api delete-bucket-lifecycle** command.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway.
- Root-level access to a Ceph Object Gateway node.
- An S3 bucket created.
- An S3 user created with user access.
- Access to a Ceph Object Gateway client with the `AWS CLI` package installed.

Procedure

- Delete a lifecycle configuration:

  Syntax

  ```
  aws --endpoint-url=_RADOSGW_ENDPOINT_URL_:PORT s3api delete-bucket-lifecycle --bucket
  _BUCKET_NAME_
  ```

  Example

  ```
  [user@client ~]$ aws --endpoint-url=http://host01:80 s3api delete-bucket-lifecycle --
  bucket testbucket
  ```

Verification

- Retrieve lifecycle configuration for the bucket:

  Syntax

  ```
  aws --endpoint-url=_RADOSGW_ENDPOINT_URL_:PORT s3api get-bucket-lifecycle-configuration
  --bucket _BUCKET_NAME_
  ```

  Example

  ```
  [user@client ~]# aws --endpoint-url=http://host01:80  s3api get-bucket-lifecycle-
  configuration --bucket testbucket
  ```

- Optional: From the Ceph Object Gateway node, retrieve the bucket lifecycle configuration:

  Syntax

```
radosgw-admin lc get --bucket=BUCKET_NAME
```

Example

```
[ceph: root@host01 /]# radosgw-admin lc get --bucket=testbucket
```

The command does not return any information if a bucket lifecycle policy is not present.

Updating a lifecycle management policy

You can update a lifecycle management policy by using the **`s3cmd put-bucket-lifecycle-configuration`** command.

> **Note:** The `put-bucket-lifecycle-configuration` overwrites an existing bucket lifecycle configuration. If you want to retain any of the current lifecycle policy settings, you must include them in the lifecycle configuration file.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway.
- Root-level access to a Ceph Object Gateway node.
- An S3 bucket created.
- An S3 user created with user access.
- Access to a Ceph Object Gateway client with the `AWS CLI` package installed.

Procedure

1. Create a JSON file for the lifecycle configuration:

   Example

   ```
   [user@client ~]$ vi lifecycle.json
   ```

2. Add the specific lifecycle configuration rules to the file:

   Example

   ```
   {
           "Rules": [
           {
                       "Filter": {
                               "Prefix": "images/"
                       },
                       "Status": "Enabled",
                       "Expiration": {
                               "Days": 1
                       },
                       "ID": "ImageExpiration"
               },
                   {
                       "Filter": {
                               "Prefix": "docs/"
                       },
                       "Status": "Enabled",
                       "Expiration": {
                               "Days": 30
                       },
                       "ID": "DocsExpiration"
                   }
   ```

```
        ]
}
```

3. Update the lifecycle configuration on the bucket:

   Syntax

   ```
   aws --endpoint-url=RADOSGW_ENDPOINT_URL:PORT s3api put-bucket-lifecycle-configuration
   --bucket BUCKET_NAME --lifecycle-configuration file://
   PATH_TO_LIFECYCLE_CONFIGURATION_FILE/LIFECYCLE_CONFIGURATION_FILE.json
   ```

   Example

   ```
   [user@client ~]$ aws --endpoint-url=http://host01:80 s3api put-bucket-lifecycle-
   configuration --bucket testbucket --lifecycle-configuration file://lifecycle.json
   ```

- Retrieve the lifecycle configuration for the bucket:

  Syntax

  ```
  aws --endpointurl=RADOSGW_ENDPOINT_URL:PORT s3api get-bucket-lifecycle-configuration --
  bucket BUCKET_NAME
  ```

  Example

  ```
  [user@client ~]$ aws -endpoint-url=http://host01:80 s3api get-bucket-lifecycle-
  configuration --bucket testbucket

  {
      "Rules": [
          {
              "Expiration": {
                  "Days": 30
              },
              "ID": "DocsExpiration",
              "Filter": {
                  "Prefix": "docs/"
              },
              "Status": "Enabled"
          },
          {
              "Expiration": {
                  "Days": 1
              },
              "ID": "ImageExpiration",
              "Filter": {
                  "Prefix": "images/"
              },
              "Status": "Enabled"
          }
      ]
  }
  ```

- Optional: From the Ceph Object Gateway node, log into the Cephadm shell and retrieve the bucket lifecycle configuration:

  Syntax

  ```
  radosgw-admin lc get --bucket=BUCKET_NAME
  ```

  Example

  ```
  [ceph: root@host01 /]# radosgw-admin lc get --bucket=testbucket
  {
          "prefix_map": {
          "docs/": {
                          "status": true,
                          "dm_expiration": false,
                          "expiration": 1,
                          "noncur_expiration": 0,
                          "mp_expiration": 0,
  ```

```
                        "transitions": {},
                        "noncur_transitions": {}
                },
                "images/": {
                        "status": true,
                        "dm_expiration": false,
                        "expiration": 1,
                        "noncur_expiration": 0,
                        "mp_expiration": 0,
                        "transitions": {},
                        "noncur_transitions": {}
                }
        },
        "rule_map": [
        {
        "id": "DocsExpiration",
        "rule": {
                "id": "DocsExpiration",
                "prefix": "",
                "status": "Enabled",
                "expiration": {
                        "days": "30",
                        "date": ""
                },
            "noncur_expiration": {
                "days": "",
                "date": ""
            },
                "mp_expiration": {
                        "days": "",
                        "date": ""
                },
                "filter": {
                        "prefix": "docs/",
                        "obj_tags": {
                                "tagset": {}
                        }
                },
                "transitions": {},
                "noncur_transitions": {},
                "dm_expiration": false
        }
    },
    {
                "id": "ImageExpiration",
                "rule": {
                        "id": "ImageExpiration",
                        "prefix": "",
                        "status": "Enabled",
                        "expiration": {
                                "days": "1",
                                "date": ""
                        },
                        "mp_expiration": {
                                "days": "",
                                "date": ""
                        },
                        "filter": {
                                "prefix": "images/",
                                "obj_tags": {
                                        "tagset": {}
                                }
                        },
                        "transitions": {},
                        "noncur_transitions": {},
                        "dm_expiration": false
                }
        }
  ]
}
```


Monitoring bucket lifecycles

You can monitor lifecycle processing and manually process the lifecycle of buckets with the **radosgw-admin lc list** and **radosgw-admin lc process** commands.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to a Ceph Object Gateway node.
- Creation of an S3 bucket with a lifecycle configuration policy applied.

Procedure

1. Log into the Cephadm shell:

   Example

   ```
   [root@host01 ~]# cephadm shell
   ```

2. List bucket lifecycle progress:

   Example

   ```
   [ceph: root@host01 /]# radosgw-admin lc list

   [
      {
            “bucket”: “:testbucket:8b63d584-9ea1-4cf3-8443-a6a15beca943.54187.1”,
            “started”: “Thu, 01 Jan 1970 00:00:00 GMT”,
            “status” : “UNINITIAL”
      },
      {
            “bucket”: “:testbucket1:8b635499-9e41-4cf3-8443-a6a15345943.54187.2”,
            “started”: “Thu, 01 Jan 1970 00:00:00 GMT”,
            “status” : “UNINITIAL”
      }
   ]
   ```

   The bucket lifecycle processing status can be one of the following:

   - UNINITIAL - The process has not run yet.
   - PROCESSING - The process is currently running.
   - COMPLETE - The process has completed.

3. Optional: You can manually process bucket lifecycle policies:

   a. Process the lifecycle policy for a single bucket:

      Syntax

      ```
      radosgw-admin lc process --bucket=BUCKET_NAME
      ```

      Example

      ```
      [ceph: root@host01 /]# radosgw-admin lc process --bucket=testbucket1
      ```

   b. Process all bucket lifecycle policies immediately:

      Example

      ```
      [ceph: root@host01 /]# radosgw-admin lc process
      ```

Verification

- List the bucket lifecycle policies:

  ```
  [ceph: root@host01 /]# radosgw-admin lc list
  [
      {
            “bucket”: “:testbucket:8b63d584-9ea1-4cf3-8443-a6a15beca943.54187.1”,
  ```

```
            "started": "Thu, 17 Mar 2022 21:48:50 GMT",
            "status" : "COMPLETE"
    }
    {
            "bucket": ":testbucket1:8b635499-9e41-4cf3-8443-a6a15345943.54187.2",
            "started": "Thu, 17 Mar 2022 20:38:50 GMT",
            "status" : "COMPLETE"
    }
]
```

Reference

For more information, see the following:

- S3 bucket lifecycle

Configuring lifecycle expiration window

You can set the time that the lifecycle management process runs each day by setting the **rgw_lifecycle_work_time** parameter. By default, lifecycle processing occurs once per day, at midnight.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- A Ceph Object Gateway node.
- Installation of the Ceph Object Gateway.
- Root-level access to a Ceph Object Gateway node.

Procedure

1. Log into the Cephadm shell:

   Example

   ```
   [root@host01 ~]# cephadm shell
   ```

2. Set the lifecycle expiration time:

   Syntax

   ```
   ceph config set client.rgw rgw_lifecycle_work_time %D:%D-%D:%D
   ```

   Replace *%d:%d-%d:%d* with `start_hour:start_minute-end_hour:end_minute`.

   Example

   ```
   [ceph: root@host01 /]# ceph config set client.rgw rgw_lifecycle_work_time 06:00-08:00
   ```

Verification

- Retrieve the lifecycle expiration work time:

  Example

  ```
  [ceph: root@host01 /]# ceph config get client.rgw rgw_lifecycle_work_time

  06:00-08:00
  ```

Reference

For more information, see the following:

- S3 bucket lifecycle

S3 bucket lifecycle transition within a storage cluster

You can use a bucket lifecycle configuration to manage objects so objects are stored effectively throughout the object's lifetime. The object lifecycle transition rule allows you to manage, and effectively store the objects throughout the object's lifetime. You can transition objects to less expensive storage classes, archive, or even delete them.

You can create storage classes for:

- Fast media, such as SSD or NVMe for I/O sensitive workloads.
- Slow magnetic media, such as SAS or SATA for archiving.

You can create a schedule for data movement between a hot storage class and a cold storage class. You can schedule this movement after a specified time so that the object expires and is deleted permanently for example you can transition objects to a storage class 30 days after you have created or even archived the objects to a storage class one year after creating them. You can do this through a transition rule. This rule applies to an object transitioning from one storage class to another. The lifecycle configuration contains one or more rules using the `<Rule>` element.

Transitioning an object from one storage class to another

The object lifecycle transition rule allows you to transition an object from one storage class to another class.

PREREQUISITE

- A running Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway software.
- Root-level access to the Ceph Object Gateway node.
- An S3 user created with user access.

You can migrate data between replicated pools, erasure-coded pools, replicated to erasure-coded pools, or erasure-coded to replicated pools with the Ceph Object Gateway lifecycle transition policy.

> **Note:** Starting with 7.1, the policy-based lifecycle transition and storage class model is extended to support two more S3-compatible cloud platforms for cloud archival - **IBM COS** and **Red Hat Ceph Storage**.

1. Create a new data pool.

   ```
   ceph osd pool create POOL_NAME
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph osd pool create test.hot.data
   ```

2. Add a new storage class.

   ```
   radosgw-admin zonegroup placement add  --rgw-zonegroup default --placement-
   id PLACEMENT_TARGET --storage-class STORAGE_CLASS
   ```

   For example,

```
[ceph: root@host01 /]# radosgw-admin zonegroup placement add  --rgw-zonegroup default
--placement-id default-placement --storage-class hot.test
{
        "key": "default-placement",
        "val": {
            "name": "default-placement",
            "tags": [],
            "storage_classes": [
                "STANDARD",
                "hot.test"
            ]
        }
    }
```

3. Provide the zone placement information for the new storage class.

```
radosgw-admin zone placement add --rgw-zone default --placement-id PLACEMENT_TARGET --
storage-class STORAGE_CLASS --data-pool DATA_POOL
```

For example,

```
[ceph: root@host01 /]# radosgw-admin zone placement add --rgw-zone default --
placement-id default-placement --storage-class hot.test --data-pool test.hot.data
{
           "key": "default-placement",
           "val": {
               "index_pool": "test_zone.rgw.buckets.index",
               "storage_classes": {
                   "STANDARD": {
                       "data_pool": "test.hot.data"
                   },
                   "hot.test": {
                       "data_pool": "test.hot.data",
                   }
               },
               "data_extra_pool": "",
               "index_type": 0
 }
```

> **Note:** Consider setting the `compression_type` when creating cold or archival data storage pools with write once.

4. Enable the `rgw` application on the data pool.

```
ceph osd pool application enable POOL_NAME rgw
```

For example,

```
[ceph: root@host01 /]# ceph osd pool application enable test.hot.data rgw
enabled application 'rgw' on pool 'test.hot.data'
```

5. Restart all the `rgw` daemons.
6. Create a bucket.

```
aws s3api create-bucket --bucket BUCKET_NAME --create-bucket-
configuration BUCKET_CONFIGURATION --endpoint-url ENDPOINT_URL
```

For example,

```
[ceph: root@host01 /]# aws s3api create-bucket --bucket testbucket10 --create-bucket-
configuration LocationConstraint=default:default-placement --endpoint-url http://
10.0.0.80:8080
```

7. Add the object.

```
aws --endpoint-url ENDPOINT_URL s3api put-object --bucket BUCKET_NAME
```

For example,

```
[ceph: root@host01 /]# aws --endpoint=http://10.0.0.80:8080 s3api put-object --bucket
testbucket10
```

8. Create a second data pool.

```
ceph osd pool create POOL_NAME
```

For example,

```
[ceph: root@host01 /]# ceph osd pool create test.cold.data
```

9. Add a new storage class.

```
radosgw-admin zonegroup placement add  --rgw-zonegroup default --placement-
id PLACEMENT_TARGET --storage-class STORAGE_CLASS
```

For example,

```
[ceph: root@host01 /]# radosgw-admin zonegroup placement add  --rgw-zonegroup default
--placement-id default-placement --storage-class cold.test
{
        "key": "default-placement",
        "val": {
            "name": "default-placement",
            "tags": [],
            "storage_classes": [
                "STANDARD",
                "cold.test"
            ]
        }
}
```

10. Provide the zone placement information for the new storage class.

```
radosgw-admin zone placement add --rgw-zone default --placement-id PLACEMENT_TARGET --
storage-class STORAGE_CLASS --data-pool DATA_POOL
```

For example,

```
[ceph: root@host01 /]# radosgw-admin zone placement add --rgw-zone default --
placement-id default-placement --storage-class cold.test --data-pool test.cold.data
```

11. Enable `rgw` application on the data pool.

```
ceph osd pool application enable POOL_NAME rgw
```

For example,

```
[ceph: root@host01 /]# ceph osd pool application enable test.cold.data rgw
enabled application 'rgw' on pool 'test.cold.data'
```

12. Restart all the `rgw` daemons.
13. View the zone group configuration.

```
radosgw-admin zonegroup get --rgw-zonegroup=ZONE_GROUP_NAME > JSON_FILE_NAME
```

For example,

```
radosgw-admin zonegroup get --rgw-zonegroup= us > rgw.json
{
    "id": "3019de59-ddde-4c5c-b532-7cdd29de09a1",
    "name": "default",
    "api_name": "default",
    "is_master": "true",
    "endpoints": [],
    "hostnames": [],
    "hostnames_s3website": [],
    "master_zone": "adacbe1b-02b4-41b8-b11d-0d505b442ed4",
    "zones": [
        {
            "id": "adacbe1b-02b4-41b8-b11d-0d505b442ed4",
            "name": "default",
            "endpoints": [],
            "log_meta": "false",
            "log_data": "false",
            "bucket_index_max_shards": 11,
            "read_only": "false",
            "tier_type": "",
            "sync_from_all": "true",
            "sync_from": [],
            "redirect_zone": ""
        }
    ],
    "placement_targets": [
        {
            "name": "default-placement",
            "tags": [],
            "storage_classes": [
                "hot.test",
                "cold.test",
                "STANDARD"
            ]
        }
    ],
    "default_placement": "default-placement",
    "realm_id": "",
    "sync_policy": {
        "groups": []
    }
}
```

14. View the zone configuration.

```
radosgw-admin zone get [--rgw-zone=ZONE_NAME]
```

For example,

```
radosgw-admin zone get
{
    "id": "adacbe1b-02b4-41b8-b11d-0d505b442ed4",
    "name": "default",
    "domain_root": "default.rgw.meta:root",
    "control_pool": "default.rgw.control",
    "gc_pool": "default.rgw.log:gc",
    "lc_pool": "default.rgw.log:lc",
    "log_pool": "default.rgw.log",
    "intent_log_pool": "default.rgw.log:intent",
    "usage_log_pool": "default.rgw.log:usage",
    "roles_pool": "default.rgw.meta:roles",
    "reshard_pool": "default.rgw.log:reshard",
    "user_keys_pool": "default.rgw.meta:users.keys",
    "user_email_pool": "default.rgw.meta:users.email",
    "user_swift_pool": "default.rgw.meta:users.swift",
    "user_uid_pool": "default.rgw.meta:users.uid",
    "otp_pool": "default.rgw.otp",
    "system_key": {
        "access_key": "",
        "secret_key": ""
    },
    "placement_pools": [
        {
            "key": "default-placement",
            "val": {
                "index_pool": "default.rgw.buckets.index",
                "storage_classes": {
                    "cold.test": {
                        "data_pool": "test.cold.data"
                    },
                    "hot.test": {
                        "data_pool": "test.hot.data"
                    },
                    "STANDARD": {
                        "data_pool": "default.rgw.buckets.data"
                    }
                },
                "data_extra_pool": "default.rgw.buckets.non-ec",
                "index_type": 0
            }
        }
    ],
    "realm_id": "",
    "notif_pool": "default.rgw.log:notif"
}
```

15. Create a bucket.

```
aws s3api create-bucket --bucket BUCKET_NAME --create-bucket-
configuration BUCKET_CONFIGURATION --endpoint-url ENDPOINT_URL
```

For example,

```
[ceph: root@host01 /]# aws s3api create-bucket --bucket testbucket10 --create-bucket-
configuration LocationConstraint=default:default-placement --endpoint-url http://
10.0.0.80:8080
```

16. List the objects prior to transition.

```
radosgw-admin bucket list --bucket BUCKET_NAME
```

```
[ceph: root@host01 /]# radosgw-admin bucket list --bucket testbucket10
{
            "ETag": "\"211599863395c832a3dfcba92c6a3b90\"",
            "Size": 540,
            "StorageClass": "STANDARD",
            "Key": "obj1",
            "VersionId": "W95teRsXPSJI4YWJwwSG30KxSCzSgk-",
            "IsLatest": true,
            "LastModified": "2023-11-23T10:38:07.214Z",
            "Owner": {
                "DisplayName": "test-user",
                "ID": "test-user"
            }
}
```

17. Create a JSON file for lifecycle configuration.

```
vi LICEYCLE_CONFGURATION_FILE.json
```

For example,

```
[ceph: root@host01 /]# vi lifecycle.json
```

18. Add the specific lifecycle configuration rule in the file.

```
{
    "Rules": [
        {
            "Filter": {
                "Prefix": ""
            },
            "Status": "Enabled",
            "Transitions": [
                {
                    "Days": 5,
                    "StorageClass": "hot.test"
                },
 {
                    "Days": 20,
                    "StorageClass": "cold.test"
                }
            ],
            "Expiration": {
                "Days": 365
            },
            "ID": "double transition and expiration"
        }
    ]
}
```

The lifecycle configuration example shows an object that transitions from the default `STANDARD` storage class to the `hot.test` storage class after 5 days, again transitions after 20 days to the `cold.test` storage class, and finally expires after 365 days in the `cold.test` storage class.

19. Set the lifecycle configuration on the bucket.

```
aws s3api put-bucket-lifecycle-configuration --bucket BUCKET_NAME --lifecycle-
configuration LICEYCLE_CONFGURATION_FILE.json
```

For example,

```
[ceph: root@host01 /]# aws s3api put-bucket-lifecycle-configuration --bucket
testbucket10 --lifecycle-configuration file://lifecycle.json
```

20. Retrieve the lifecycle configuration on the bucket.

```
aws s3api get-bucket-lifecycle-configuration --bucket BUCKET_NAME
```

For example,

```
[ceph: root@host01 /]# aws s3api get-bucket-lifecycle-configuration --bucket
testbucket10
{
   "Rules": [
       {
           "Expiration": {
               "Days": 365
           },
           "ID": "double transition and expiration",
           "Prefix": "",
           "Status": "Enabled",
           "Transitions": [
               {
                   "Days": 20,
                   "StorageClass": "cold.test"
               },
               {
                   "Days": 5,
                   "StorageClass": "hot.test"
               }
           ]
       }
   ]
}
```

21. Verify that the object is transitioned to the given storage class.

```
radosgw-admin bucket list --bucket BUCKET_NAME
```

For example,

```
[ceph: root@host01 /]# radosgw-admin bucket list --bucket testbucket10
{
            "ETag": "\"211599863395c832a3dfcba92c6a3b90\"",
            "Size": 540,
            "StorageClass": "cold.test",
            "Key": "obj1",
            "VersionId": "W95teRsXPSJI4YWJwwSG30KxSCzSgk-",
            "IsLatest": true,
            "LastModified": "2023-11-23T10:38:07.214Z",
            "Owner": {
                "DisplayName": "test-user",
                "ID": "test-user"
            }
}
```

Enabling object lock for S3

Using the S3 object lock mechanism, you can use object lock concepts like retention period, legal hold, and bucket configuration to implement Write-Once-Read_Many (WORM) functionality as part of the custom workflow overriding data deletion permissions.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- Root-level access to a Ceph Object Gateway node.
- S3 user with version-bucket creation access.

Important:

- The object version(s), not the object name, is the defining and required value for object lock to perform correctly to support the **GOVERNANCE** or **COMPLIANCE** mode. You need to know the version of the object when it is written so that you can retrieve it at a later time.
- Consulting company Cohasset concludes that Red Hat Ceph Storage, when properly configured and upon satisfying the additional considerations, meets the electronic record-keeping system requirements of **SEC Rules 17a-4(f)(2)**, **18a-6(e)(2)**, and **FINRA Rule 4511(c)**, as well as, supports the regulated entity in its compliance with the audit system requirements in **SEC Rules 17a-4(f)(3)**

(iii) and **18a-6(e)(3)(iii)**. In addition, the assessed capabilities meet the principles-based electronic records requirements of **CFTC Rule 1.31(c)-(d)**. See the Cohasset compliance certification for more information.
Red Hat Ceph Storage Compliance Assessment

1. Create a bucket with object lock enabled.

```
aws --endpoint=http://RGW_PORT:8080 s3api create-bucket --bucket BUCKET_NAME --object-lock-enabled-for-bucket
```

For example,

```
[root@rgw-2 ~]# aws --endpoint=http://rgw.ceph.com:8080 s3api create-bucket --bucket worm-bucket --object-lock-enabled-for-bucket
```

2. Set a retention period for the bucket.

```
aws --endpoint=http://RGW_PORT:8080 s3api put-object-lock-configuration --bucket BUCKET_NAME --object-lock-configuration { "ObjectLockEnabled": "Enabled", "Rule": { "DefaultRetention": { "Mode": "RETENTION_MODE", "Days": NUMBER_OF_DAYS }}}
```

You can choose either the GOVERNANCE or COMPLIANCE mode for the *RETENTION_MODE* in S3 object lock, to apply different levels of protection to any object version that is protected by object lock.

GOVERNANCE

In GOVERNANCE mode, users cannot overwrite or delete an object version or alter its lock settings unless they have special permissions.

COMPLIANCE

In COMPLIANCE mode, a protected object version cannot be overwritten or deleted by any user, including the root user in your AWS account. When an object is locked in COMPLIANCE mode, its *RETENTION_MODE* cannot be changed, and its retention period cannot be shortened. COMPLIANCE mode helps ensure that an object version cannot be overwritten or deleted for the duration of the period.

```
[root@rgw-2 ~]# aws --endpoint=http://rgw.ceph.com:8080 s3api put-object-lock-configuration --bucket worm-bucket --object-lock-configuration '{ "ObjectLockEnabled": "Enabled", "Rule": { "DefaultRetention": { "Mode": "COMPLIANCE", "Days": 10 }}}'
```

3. Put the object into the bucket with a retention time set.

```
aws --endpoint=http://RGW_PORT:8080 s3api put-object --bucket BUCKET_NAME --object-lock-mode RETENTION_MODE --object-lock-retain-until-date "DATE" --key compliance-upload --body TEST_FILE
```

For example,

```
[root@rgw-2 ~]# aws --endpoint=http://rgw.ceph.com:8080 s3api put-object --bucket worm-bucket --object-lock-mode COMPLIANCE --object-lock-retain-until-date "2022-05-31" --key compliance-upload --body test.dd
{
    "ETag": ""d560ea5652951637ba9c594d8e6ea8c1"",
    "VersionId": "Nhhk5kRS6Yp6dZXVWpZZdRcpSpBKToD"
}
```

4. Upload a new object using the same key.

```
aws --endpoint=http://RGW_PORT:8080 s3api put-object --bucket BUCKET_NAME --object-lock-mode RETENTION_MODE --object-lock-retain-until-date "DATE" --key compliance-upload --body PATH
```

For example,

```
[root@rgw-2 ~]# aws --endpoint=http://rgw.ceph.com:8080 s3api put-object --bucket
worm-bucket --object-lock-mode COMPLIANCE --object-lock-retain-until-date "2022-05-31"
--key compliance-upload --body /etc/fstab
{
    "ETag": ""d560ea5652951637ba9c594d8e6ea8c1"",
    "VersionId": "Nhhk5kRS6Yp6dZXVWpZZdRcpSpBKToD"
}
```

AFTER COMPLETING THE TASK

You can use the following command-line options with object lock.

- Set an object lock legal hold on an object version.
  For example,

```
[root@rgw-2 ~]# aws --endpoint=http://rgw.ceph.com:8080 s3api put-object-legal-hold --
bucket worm-bucket --key compliance-upload --legal-hold Status=ON
```

> **Note:** Using the object lock legal hold operation, you can place a legal hold on an object version, thereby preventing an object version from being overwritten or deleted. A legal hold doesn't have an associated retention period and hence, remains in effect until removed.

- List the objects from the bucket to retrieve only the latest version of the object.
  For example,

```
[root@rgw-2 ~]# aws --endpoint=http://rgw.ceph.com:8080 s3api list-objects --bucket
worm-bucket
```

- List the object versions from the bucket.
  For example,

```
[root@rgw-2 ~]# aws --endpoint=http://rgw.ceph.com:8080 s3api list-objects --bucket
worm-bucket
{
    "Versions": [
        {
            "ETag": ""d560ea5652951637ba9c594d8e6ea8c1"",
            "Size": 288,
            "StorageClass": "STANDARD",
            "Key": "hosts",
            "VersionId": "Nhhk5kRS6Yp6dZXVWpZZdRcpSpBKToD",
            "IsLatest": true,
            "LastModified": "2022-06-17T08:51:17.392000+00:00",
            "Owner": {
                "DisplayName": "Test User in Tenant test",
                "ID": "test$test.user"
            }
            }
        }
    ]
}
```

- Access objects using version IDs.
  For example,

```
[root@rgw-2 ~]# aws --endpoint=http://rgw.ceph.com:8080 s3api get-object --bucket worm-
bucket  --key compliance-upload --version-id 'IGOU.vdIs3SPduZglrB-RBaK.sfXpcd'
download.1
{
    "AcceptRanges": "bytes",
    "LastModified": "2022-06-17T08:51:17+00:00",
    "ContentLength": 288,
    "ETag": ""d560ea5652951637ba9c594d8e6ea8c1"",
    "VersionId": "Nhhk5kRS6Yp6dZXVWpZZdRcpSpBKToD",
    "ContentType": "binary/octet-stream",
    "Metadata": {},
    "ObjectLockMode": "COMPLIANCE",
    "ObjectLockRetainUntilDate": "2023-06-17T08:51:17+00:00"
}
```

Bucket logging

Bucket logging records all access requests to a bucket and stores them as log objects in a separate log bucket. The log data can be used to monitor bucket activity, detect unauthorized access, analyze usage patterns, and maintain an audit trail of bucket changes.

Logging is configured at the bucket level and can be enabled or disabled at any time. The log bucket can collect logs from multiple source buckets. To distinguish logs from different buckets, configure a unique prefix for each source bucket so that the logs are stored in separate objects within the log bucket.Before enabling logging, ensure that the log bucket is created. The log bucket cannot be the same as the source bucket, and it cannot have logging, compression, or server-side encryption (SSE-S3 or SSE-C) enabled. It must not have RequestPayer enabled. Source and log buckets must be in the same zonegroup, but they can belong to different accounts if the log bucket policy allows it.

> **Note:** The log bucket may have object lock enabled with a default retention period. The 16-byte unique ID in the log object name is a lexicographically ordered value made up of a 10-byte counter followed by a 6-byte random alphanumeric string. If the counter is unavailable, the ID is generated entirely from a random alphanumeric string.

The logs are stored as objects in a dedicated log bucket, which can collect events from multiple source buckets. Each source bucket should be configured with a unique prefix so that logs are written to separate objects within the log bucket.

Logging modes

Bucket logging supports two modes that differ in reliability guarantees and in the bucket operations they record.

Standard mode

In Standard mode, log records are written to the log bucket after the bucket operation completes. Failures in logging do not affect the bucket operation. Logs follow the AWS-compatible standard record format, which includes information such as request ID, client IP, HTTP status, and user agent.

Journal mode

In Journal mode, log records are written before the bucket operation completes. If logging fails, the corresponding bucket operation also fails, except for specific operations where failure is ignored. The following bucket operations are supported in Journal mode:

- `PutObject, CopyObject, CompleteMultipartUpload, PutObjectAcl, PutObjectLegalHold, PutObjectRetention, and PutObjectTagging` — if logging fails, the operation fails.
- `DeleteObject, DeleteObjects, and DeleteObjectTagging` — the operation continues even if logging fails.

Journal mode uses a smaller, Ceph-specific record format and is intended for journaling changes and ensuring strong consistency. Journal mode supports filtering out records based on matches of the prefixes and suffixes of the logged object keys. Regular expression matching can also be used on these to create filters. Note that it may happen that the log records were successfully written but the bucket operation failed, since the logs are written.

Reliability and flushing

Log objects do not appear in the log bucket immediately. By default, log objects are rolled every five minutes or when they reach 128 MB in size.Log objects can be flushed manually. Automatic flush occurs when logging is disabled, when the configuration changes, or when the source bucket is deleted. The time (in seconds) can be set per source bucket via a Ceph extension to the REST API , or globally via the `rgw_bucket_logging_obj_roll_time` configuration option. If not set, the default time is 5 minutes.

Multisite

In multisite deployments, each zone maintains temporary log objects before they are flushed to the log bucket. After flushing, log objects are then replicated across all zones similar to other objects sync in a multisite.

Policies and quotas

Only the bucket owner can enable or disable logging. To use a bucket as a log bucket, its policy must allow the `s3:PutObject` action for the `logging.s3.amazonaws.com` service principal. The policy should also specify the source bucket and account that are permitted to write logs.Bucket and user quotas apply to the log bucket:

- In Journal mode, exceeding quota causes both logging and the bucket operation to fail.
- In Standard mode, logging is skipped, but the bucket operation continues successfully.

For example,

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowLoggingFromSourceBucket",
      "Effect": "Allow",
      "Principal": {
        "Service": "logging.s3.amazonaws.com"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::log-bucket-name/prefix*",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "source-account-id"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:s3:::source-bucket-name"
        }
      }
    }
  ]
}
```

Log Objects Key Format

There are two types of log object key formats: Simple and Partitioned.

The Simple log object key follows this structure:

```
<prefix><year-month-day-hour-minute-second>-<16-byte-unique-id>
```

For example,

```
fish/2024-08-06-09-40-09-0000000002AGQ6W1
```

The Partitioned log object key uses a hierarchical structure that includes the bucket owner, zone group, and date components:

```
<prefix><source-bucket-owner>/<zone-group>/[tenant:]<source-bucket-name>/<year>/<month>/
<day>/<year-month-day-hour-minute-second>-<16-byte-unique-id>
```

For example,

```
fish/testid/default/fish-bucket/2024/08/06/2024-08-06-10-11-18-0000000011D1FGPA
```

Log record formats

Bucket logging supports two formats for log records, depending on the logging mode used.

Journal format

A minimal Ceph-specific record used for journaling bucket changes.

Standard format

AWS-compatible record format with extended metadata fields such as request ID, client IP, and HTTP status.

The Journal record format uses minimum amount of data for journaling bucket changes (this is a Ceph extension):

- bucket owner (or dash if empty)
- bucket name (or dash if empty), in the format: `[tenant:]<bucket name>`
- time in the following format: `[day/month/year:hour:minute:second timezone]`
- operation in the following format: `WEBSITE/REST.<HTTP method>.<resource>`
- object key (or dash if empty)
- object size (or dash if empty
- version id (or dash if empty)
- eTag (or dash if empty)

For example,

```
testid fish [06/Aug/2024:09:40:09 +0000] REST.PUT.OBJECT myfile - 512
4cfdfc1f58e762d3e116787cb92fac60
testid fish [06/Aug/2024:09:40:28 +0000] REST.DELETE.OBJECT myfile - -
4cfdfc1f58e762d3e116787cb92fac60
```

The Standard record format is based on AWS Logging Record Format.

- bucket owner (or dash if empty)
- bucket name (or dash if empty) in the format: `[tenant:]<bucket name>`
- time in the following format: `[day/month/year:hour:minute:second timezone]` where “timezone” is in UTC offset
- client IP address (or dash if empty)
- user or account (or dash if empty)
- request ID
- operation in the following format: `WEBSITE/REST.<HTTP method>.<resource>`
- object key (or dash if empty)
- request URI in the following format: `"<HTTP method> <URI> <HTTP version>"`
- HTTP status (or dash if zero). Note that in most cases log is written before the status is known
- error code (or dash if empty)
- bytes sent (or dash if zero)
- object size (or dash if zero)
- total time (not supported, always a dash)
- turnaround time in milliseconds
- referer (or dash if empty)
- user agent (or dash if empty) inside double quotes
- version id (or dash if empty)
- host id taken from `x-amz-id-2` (or dash if empty)
- signature version (or dash if empty)
- cipher suite (or dash if empty)
- authentication type (`AuthHeader` for regular auth, `QueryString` for presigned URL or dash if unauthenticated)
- host header (or dash if empty)
- TLS version (or dash if empty)

- access point ARN (not supported, always a dash)
- ACL flag (`Yes` if an ACL was required for authorization, otherwise dash)

For example,

```
testid fish [06/Aug/2024:09:30:25 +0000] - testid 9e369a15-5f43-4f07-b638-
de920b22f91b.4179.15085270386962380710 REST.PUT.OBJECT myfile "PUT /fish/myfile HTTP/1.1"
200 - 512 512 - - - - - - - - - localhost - -
testid fish [06/Aug/2024:09:30:51 +0000] - testid 9e369a15-5f43-4f07-b638-
de920b22f91b.4179.7046073853138417766 REST.GET.OBJECT myfile "GET /fish/myfile HTTP/1.1"
200 - - 512 - - - - - - - - - localhost - -
testid fish [06/Aug/2024:09:30:56 +0000] - testid 9e369a15-5f43-4f07-b638-
de920b22f91b.4179.10723158448701085570 REST.DELETE.OBJECT myfile "DELETE /fish/myfile1
HTTP/1.1" 200 - - 512 - - - - - - - - - localhost - -
```

For information about enabling S2 bucket logging, see “S3 enable bucket logging” on page 0.

Identity and Access Management (IAM)

The Ceph Object Gateway supports `user accounts` as an optional feature to enable the self-service management of users, groups and roles similar to those in AWS Identity and Access Management (IAM).

Account Root User

Each account is managed by an `account root user`. Like normal users and roles, accounts and account root users must be created by an administrator using radosgw-admin or the Admin Ops API. The account root user has default permissions on all resources owned by the account. The root user’s access and secret keys can be used with the Ceph Object Gateway IAM API to create and manage IAM users and roles for the Ceph Object Gateway S3 API, including their associated access keys and policies.

Account owners are encouraged to use this account root user for management only, and create users and roles with fine-grained permissions for specific applications.

> **Note:** The account root user does not require an IAM policy to access resources within the account. However, you can explicitly deny access by attaching a policy with Deny statements. Use Deny statements with caution, as they override all allow permissions.

Resource Ownership

When a normal (non-account) user creates buckets and uploads objects, those resources are owned by the user. The associated S3 ACLs name that user as both the owner and grantee, and those buckets are only visible to the owning user in a `s3:ListBuckets request`. In contrast, when users or roles belong to an account, the resources they create are instead owned by the account itself. The associated S3 ACLs name the account id as the owner and grantee, and those buckets are visible to `s3:ListBuckets` requests sent by any user or role in that account.

Because the resources are owned by the account rather than its users, all usage statistics and quota enforcement apply to the account as a whole rather than its individual users.

Account IDS

Account identifiers can be used in several places that otherwise accept User IDs or tenant names, so Account IDs use a special format to avoid ambiguity: the string RGW followed by 17 numeric digits like `RGW33567154695143645`. An Account ID in that format is randomly generated upon account creation if one is not specified.

Account IDs are commonly found in the Amazon Resource Names (ARNs) of IAM policy documents. For example, `arn:aws:iam::RGW33567154695143645:user/A` refers to an IAM user named A in that account. The Ceph Object Gateway also supports tenant names in that position. Accounts IDs can also be used in ACLs for a Grantee of type CanonicalUser. User IDs are also supported here.

IAM Policy

While non-account users are allowed to create buckets and upload objects by default, account users start with no permissions at all. Before an IAM user can perform API operations, some policy must be added to allow it. The account root user can add identity policies to its users in several ways.

- Add policy directly to the user with the iam:PutUserPolicy and iam:AttachUserPoliicy actions.
- Create an IAM group and add group policy with the `iam:PutGroupPolicy` and `iam:AttachGroupPolicy`actions. Users added to that group with the `iam:AddUserToGroup` action will inherit all of the group's policy.
- Create an IAM role and add role policy with the `iam:PutRolePolicy` and iam:AttachRolePoliicy actions. Users that assume this role with the`sts:AssumeRole` and `sts:AssumeRoleWithWebIdentity` actions will inherit all of the role's policy.
  These identity policies are evaluated according to the rules in Evaluating policies within a single account and Cross-account policy evaluation logic.

Principals

The "Principal" ARNs in policy documents refer to users differently when they belong to an account. Outside of an account, user principals are named by user id such as `arn:aws:iam:::user/uid` or `arn:aws:iam::tenantname:user/uid`, where `uid` corresponds to the `--uid` argument from `radosgw-admin`.

Within an account, user principals instead use the user name, such as `arn:aws:iam::RGW33567154695143645:user/name` where `name` corresponds to the `--display-name` argument from `radosgw-admin`. Account users continue to match the tenant form so that existing policy continues to work when users are migrated into accounts.

Tenant Isolation

Like users, accounts can optionally belong to a tenant for namespace isolation of buckets. For example, one account named "acct" can exist under a tenant "a", and a different account named "acct" can exist under tenant "b".

A tenanted account can only contain users with the same tenant name. Regardless of tenant, account IDs and email addresses must be globally unique.

Create an account

Use the `radosgw-admin account create` command to create an IAM user.

You MUST specify a user ID and a display name. You may also specify an email address.

```
radosgw-admin account create [--account-name={name}] [--account-id={id}] [--email={email}]
```

Example

```
radosgw-admin account create --account-name=user1 --account-id=12345 --
email=user1@example.com
```

Create an account root user

Use the `radosgw-admin user create` command to create an IAM user.

You MUST specify a user ID and a display name. You may also specify an email address. A **root user** is a privileged IAM user that has full access to the account, and can perform all operations, including managing other users.

```
radosgw-admin user create --uid={userid} --display-name={name} --account-id={accountid} --
account-root --gen-secret --gen-access-key
```

Example

```
radosgw-admin user create --uid=rootuser1 --display-name="Root User One" --account-
id=account123 --account-root --gen-secret --gen-access-key
```

Delete an account

Use the `radosgw-admin account rm` command to delete an IAM user.

Removing a user removes the user from the system..

```
radosgw-admin account rm --account-id={accountid}
```

Example

```
radosgw-admin account rm --account-id=account123
```

Enable and view account statistics and quota

You can retrieve detailed statistics for a specific account, such as storage usage and object counts. You can also define and enable a storage quota on an account, and set and enable a quota on a specific bucket within an account, limiting the maximum number of objects that can be stored in that bucket.

View account statistics

To view account statistics, use the following command:

```
radosgw-admin account stats --account-id=ACCOUNT_ID --sync-stats
```

For example,

```
{
  "account": "account123",
  "data_size": 3145728000,        # Total size in bytes (3 GB)
  "num_objects": 12000,           # Total number of objects
  "num_buckets": 5,               # Total number of buckets
  "usage": {
    "total_size": 3145728000,     # Total size in bytes (3 GB)
    "num_objects": 12000
  }
}
```

Enable an account quota

To enable an account quota, use the following commands:

```
radosgw-admin quota set --quota-scope=account --account-id=ACCOUNT_ID --max-size=10G
radosgw-admin quota enable --quota-scope=account --account-id=ACCOUNT_ID
```

For example,

```
{
  "status": "OK",
  "message": "Quota enabled for account account123"
}
```

Enable a bucket quota for the account

To enable a bucket quota for the account, use the following commands:

```
radosgw-admin quota set --quota-scope=bucket --account-id=ACCOUNT_ID --max-objects=1000000
radosgw-admin quota enable --quota-scope=bucket --account-id=ACCOUNT_ID
```

For example,

```
{
  "status": "OK",
  "message": "Quota enabled for bucket in account account123"
}
```

Modify the maximum account quota

To modify the account quota for the maximum number of buckets, users, or roles, use the **quota set** command.

```
radosgw-admin quota set --quota-scope=account --account-id ACCOUNT_ID --max-buckets VALUE
```

```
[root@magna045 ~]# radosgw-admin quota set --quota-scope=account --account-id
RGW12345678901234568 --max-buckets 10000
{
    "id": "RGW12345678901234568",
    "tenant": "tenant1",
    "name": "account1",
    "email": "tenataccount1",
    "quota": {
        "enabled": true,
        "check_on_raw": false,
        "max_size": 10737418240,
        "max_size_kb": 10485760,
        "max_objects": 100
    },
    "bucket_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
    "max_users": 1000,
    "max_roles": 1000,
    "max_groups": 1000,
    "max_buckets": 1000,
    "max_access_keys": 4
}
[root@magna045 ~]# radosgw-admin quota enable --quota-scope=account --account-id
RGW12345678901234568
{
    "id": "RGW12345678901234568",
    "tenant": "tenant1",
    "name": "account1",
    "email": "tenataccount1",
    "quota": {
        "enabled": true,
        "check_on_raw": false,
        "max_size": 10737418240,
        "max_size_kb": 10485760,
        "max_objects": 100
    },
    "bucket_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
    "max_users": 1000,
    "max_roles": 1000,
    "max_groups": 1000,
    "max_buckets": 1000,
    "max_access_keys": 4
}
[root@magna045 ~]# radosgw-admin account get --account-id RGW12345678901234568
{
    "id": "RGW12345678901234568",
    "tenant": "tenant1",
    "name": "account1",
    "email": "tenataccount1",
    "quota": {
        "enabled": true,
        "check_on_raw": false,
        "max_size": 10737418240,
        "max_size_kb": 10485760,
        "max_objects": 100
    },
    "bucket_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
```

```
        "max_objects": -1
    },
    "max_users": 1000,
    "max_roles": 1000,
    "max_groups": 1000,
    "max_buckets": 1000,
    "max_access_keys": 4
}
[root@magna045 ~]#

[root@magna045 ~]# ceph versions
{
    "mon": {
        "ceph version 19.1.1-63.el9cp (8fa7b56d5e9f208c4233b0a8273665087bded8ae) squid
(rc)": 3
    },
    "mgr": {
        "ceph version 19.1.1-63.el9cp (8fa7b56d5e9f208c4233b0a8273665087bded8ae) squid
(rc)": 3
    },
    "osd": {
        "ceph version 19.1.1-63.el9cp (8fa7b56d5e9f208c4233b0a8273665087bded8ae) squid
(rc)": 9
    },
    "rgw": {
        "ceph version 19.1.1-63.el9cp (8fa7b56d5e9f208c4233b0a8273665087bded8ae) squid
(rc)": 3
    },
    "overall": {
        "ceph version 19.1.1-63.el9cp (8fa7b56d5e9f208c4233b0a8273665087bded8ae) squid
(rc)": 18
    }
}
[root@magna045 ~]#
```

Migrate an existing user to an account

Use the `radosgw-admin user modify` command to modify an existing IAM user account.

When you create a role associated with an RGW account, you can list it using `radosgw-admin role list --account-id <RGWaccountID>` command.

To modify an existing user IAM account:

Syntax

```
radosgw-admin user modify --uid={userid} --account-id={accountid}
```

> **Note:** Ownership of all of the user's buckets will be transferred to the account.

> **Note:** Account membership is permanent. Once added, users cannot be removed from their account.

> **Note:** Ownership of the user's notification topics will not be transferred to the account. Notifications will continue to work, but the topics will no longer be visible to SNS Topic APIs.

Because account users have no permissions by default, some identity policy must be added to restore the user's original permissions. Alternatively, you may want to create a new account for each existing user. In that case, you may want to add the --account-root option to make each user the root user of their account.

Migrating notification topics

Account topics are supported only when the notification_v2 feature is enabled

Migration Impact

When a non-account user is migrated to an account, the the existing notification topics remain accessible through the RadosGW admin API, but the user loses access to them via the SNS Topic API. Despite this, the topics remain functional, and bucket notifications will continue to be delivered as expected.

Re-creation of Topics

The account user should re-create the topics using the same names. The old topics (now inaccessible) and the new account-owned topics will coexist without interference.

Updating Bucket Notification Configurations

Buckets that are subscribed to the old user-owned topics should be updated to use the new account-owned topics. To prevent duplicate notifications, maintain the same notification IDs.

For example, if a bucket's existing notification configuration is:

```
{"TopicConfigurations": [{ "Id": "ID1", "TopicArn": "arn:aws:sns:default::topic1",
"Events": ["s3:ObjectCreated:*"]}]}
```

The updated configuration would be:

```
{"TopicConfigurations": [{ "Id": "ID1", "TopicArn":
"arn:aws:sns:default:RGW00000000000000001:topic1", "Events": ["s3:ObjectCreated:*"]}]}
```

```
In this example, RGW00000000000000001 is the account ID, topic1 is the topic name and ID1
is the notification ID.
```

Removing Old Topics

Once no buckets are subscribed to the old user-owned topics, they can be removed by an admin:

```
$ radosgw-admin topic rm --topic topic1
```

Migrating users from non-account to RGW accounts

When migrating a user from a non-account to an RGW account, follow the steps below to ensure a seamless IO migration and appropriate IAM configuration:

1. Assign the account-root flag during user migration. Ensure that the `account-root` flag is added when migrating the user to an account. This grants the user the necessary privileges for migration.

   ```
   radosgw-admin user modify --uid <user_ID> --account-id <Account_ID> --account-root
   ```

2. Attach S3 access policy. After migrating the user, attach the appropriate IAM policy to grant the user S3 access. In this case, we're assigning the `AmazonS3FullAccess` policy.

   ```
   radosgw-admin user policy attach --uid <user_ID> --policy-arn arn:aws:iam::aws:policy/
   AmazonS3FullAccess
   ```

3. Remove the account-root privilege. After the migration and granting necessary access, remove the `account-root` privilege from the user to limit the scope of their permissions.

   ```
   radosgw-admin user modify --uid <user_ID> --account-root=0
   ```

User management

Ceph Object Storage user management refers to users that are client applications of the Ceph Object Storage service; not the Ceph Object Gateway as a client application of the Ceph Storage Cluster. You must create a user, access key, and secret to enable client applications to interact with the Ceph Object Gateway service.

There are two user types:

- *User:* The term 'user' reflects a user of the S3 interface.
- *Subuser:* The term 'subuser' reflects a user of the Swift interface. A subuser is associated to a user .

You can create, modify, view, suspend, and remove users and subusers.

Important: When managing users in a multi-site deployment, ALWAYS issue the `radosgw-admin` command on a Ceph Object Gateway node within the master zone of the master zone group to ensure that users synchronize throughout the multi-site cluster. DO NOT create, modify, or delete users on a multi-site cluster from a secondary zone or a secondary zone group.

In addition to creating user and subuser IDs, you may add a display name and an email address for a user. You can specify a key and secret, or generate a key and secret automatically. When generating or specifying keys, note that user IDs correspond to an S3 key type and subuser IDs correspond to a swift key type. Swift keys also have access levels of `read`, `write`, `readwrite` and `full`.

User management command line syntax generally follows the pattern `user COMMAND USER_ID` where `USER_ID` is either the `--uid=` option followed by the user's ID (S3) or the `--subuser=` option followed by the user name (Swift).

Syntax

```
radosgw-admin user <create|modify|info|rm|suspend|enable|check|stats> <--uid=USER_ID|--
subuser=SUB_USER_NAME> [other-options]
```

Additional options may be required depending on the command you issue.

Multi-tenant namespace

The Ceph Object Gateway supports multi-tenancy for both the S3 and Swift APIs, where each user and bucket lies under a "tenant." Multi tenancy prevents namespace clashing when multiple tenants are using common bucket names, such as "test" and "main".

Each user and bucket lies under a tenant. For backward compatibility, a "legacy" tenant with an empty name is added. Whenever referring to a bucket without specifically specifying a tenant, the Swift API will assume the "legacy" tenant. Existing users are also stored under the legacy tenant, so they will access buckets and objects the same way as earlier releases.

Tenants as such do not have any operations on them. They appear and disappear as needed, when users are administered. In order to create, modify, and remove users with explicit tenants, either an additional option `--tenant` is supplied, or a syntax `"_TENANT_$_USER_"` is used in the parameters of the `radosgw-admin` command.

To create a user `testx$tester` for S3, run the following command:

Example

```
[root@host01 ~]# radosgw-admin --tenant testx --uid tester
                    --display-name "Test User" --access_key TESTER
                    --secret test123 user create
```

To create a user `testx$tester` for Swift, run one of the following commands:

Example

```
[root@host01 ~]# radosgw-admin --tenant testx --uid tester
                    --display-name "Test User" --subuser tester:swift
                    --key-type swift --access full subuser create

[root@host01 ~]# radosgw-admin key create --subuser 'testx$tester:swift'
                    --key-type swift --secret test123
```

Note: The subuser with explicit tenant had to be quoted in the shell.

Create a user

Use the `user create` command to create an S3-interface user.

You MUST specify a user ID and a display name. You may also specify an email address. If you DO NOT specify a key or secret, `radosgw-admin` will generate them for you automatically. However, you may specify a key and/or a secret if you prefer not to use generated key/secret pairs.

Syntax

```
radosgw-admin user create --uid=USER_ID
[--key-type=KEY_TYPE] [--gen-access-key|--access-key=ACCESS_KEY]
[--gen-secret | --secret=SECRET_KEY]
[--email=EMAIL] --display-name=DISPLAY_NAME
```

Example

```
[root@host01 ~]# radosgw-admin user create --uid=janedoe --access-key=11BS02LGFB6AL6H1ADMW
--secret=vzCEkuryfn060dfee4fgQPqFrncKEIkh3ZcdOANY --email=jane@example.com --display-
name=Jane Doe

{ "user_id": "janedoe",
  "display_name": "Jane Doe",
  "email": "jane@example.com",
  "suspended": 0,
  "max_buckets": 1000,
  "auid": 0,
  "subusers": [],
  "keys": [
        { "user": "janedoe",
          "access_key": "11BS02LGFB6AL6H1ADMW",
          "secret_key": "vzCEkuryfn060dfee4fgQPqFrncKEIkh3ZcdOANY"}],
  "swift_keys": [],
  "caps": [],
  "op_mask": "read, write, delete",
  "default_placement": "",
  "placement_tags": [],
  "bucket_quota": { "enabled": false,
      "max_size_kb": -1,
      "max_objects": -1},
  "user_quota": { "enabled": false,
      "max_size_kb": -1,
      "max_objects": -1},
  "temp_url_keys": []}
```

Important: Check the key output. Sometimes `radosgw-admin` generates a JSON escape `()` character, and some clients do not know how to handle JSON escape characters. Remedies include removing the JSON escape character `()`, encapsulating the string in quotes, regenerating the key to ensure that it does not have a JSON escape character, or specifying the key and secret manually.

Create a subuser

To create a subuser (Swift interface), you must specify the user ID (`--uid=_USERNAME_`), a subuser ID and the access level for the subuser. If you DO NOT specify a key or secret, `radosgw-admin` generates them for you automatically. However, you can specify a key, a secret, or both if you prefer not to use generated key and secret pairs.

Note: `full` is not `readwrite`, as it also includes the access control policy.

Syntax

```
 radosgw-admin subuser create --uid=USER_ID --subuser=SUB_USER_ID --access=[ read | write |
readwrite | full ]
```

Example

```
[root@host01 ~]# radosgw-admin subuser create --uid=janedoe --subuser=janedoe:swift --
access=full
```

```
{ "user_id": "janedoe",
  "display_name": "Jane Doe",
  "email": "jane@example.com",
  "suspended": 0,
  "max_buckets": 1000,
  "auid": 0,
  "subusers": [
        { "id": "janedoe:swift",
          "permissions": "full-control"}],
  "keys": [
        { "user": "janedoe",
          "access_key": "11BS02LGFB6AL6H1ADMW",
          "secret_key": "vzCEkuryfn060dfee4fgQPqFrncKEIkh3ZcdOANY"}],
  "swift_keys": [],
  "caps": [],
  "op_mask": "read, write, delete",
  "default_placement": "",
  "placement_tags": [],
  "bucket_quota": { "enabled": false,
      "max_size_kb": -1,
      "max_objects": -1},
  "user_quota": { "enabled": false,
      "max_size_kb": -1,
      "max_objects": -1},
  "temp_url_keys": []}
```

Get user information

Use the **`user info`** command and the user ID to retrieve user information.

To get information about a user, specify `user info` and the user ID (`--uid=_USERNAME_`).

Example

```
[root@host01 ~]# radosgw-admin user info --uid=janedoe
```

To get information about a tenanted user, specify both the user ID and the name of the tenant.

```
[root@host01 ~]# radosgw-admin user info --uid=janedoe --tenant=test
```

Modify user information

Modify user information, by using the **`user modify`** command. Typically, modifications are made to keys and secrets, email addresses, display names, and access levels.

To modify information about a user, you must specify the user ID (`--uid=_USERNAME_`) and the attributes you want to modify.

Example

```
[root@host01 ~]# radosgw-admin user modify --uid=janedoe --display-name="Jane E. Doe"
```

To modify subuser values, specify `subuser modify` and the subuser ID.

Example

```
[root@host01 ~]# radosgw-admin subuser modify --subuser=janedoe:swift --access=full
```

Enable and suspend users

When you create a user, the user is enabled by default. However, you may suspend user privileges and re-enable them at a later time.

To suspend a user, specify `user suspend` and the user ID.

```
[root@host01 ~]# radosgw-admin user suspend --uid=johndoe
```

To re-enable a suspended user, specify `user enable` and the user ID:

```
[root@host01 ~]# radosgw-admin user enable --uid=johndoe
```

> **Note:** Disabling the user disables the subuser.

Removing a user and subuser

Removing a user removes both the user and subuser from the system. A subuser can also be removed separately.

Remove a user only when wanting to remove both a subuser and subuser from the system.

When you remove a subuser, you are removing access to the Swift interface but the user remains in the system.

Removing a user

- Remove a user and subuser from the system, by using the **`user rm`** command with the user ID.

  ```
  radosgw-admin user rm --uid=USER_ID [--purge-keys] [--purge-data]
  ```

  For example,

  ```
  [ceph: root@host01 /]# radosgw-admin user rm --uid=johndoe --purge-data
  ```

  The following command options are available:

  `--purge-data`

  The `--purge-data` option purges all data associated with the user ID.

  `--purge-keys`

  The `--purge-keys` option purges all keys associated with the user ID.

Removing a subuser

- Remove a subuser from the system, by using the **`subuser rm`** command with the subuser name.

  ```
  radosgw-admin subuser rm --subuser=SUBUSER_ID:swift --purge-keys
  ```

  > **Note:** Use the `--purge-keys` option to purge all subuser keys.

  For example,

  ```
  [root@host01 /]# radosgw-admin subuser rm --subuser=johndoe:swift --purge-keys
  ```

Rename a user

To change the name of a user, use the **`radosgw-admin user rename`** command.

The time that this command takes depends on the number of buckets and objects that the user has. If the number is large, Red Hat recommends using the command in the `Screen` utility provided by the `screen` package.

Prerequisites

- A working Red Hat Ceph Storage cluster.
- `root` or `sudo` access to the host running the Ceph Object Gateway.

- Installed Ceph Object Gateway.

Procedure

1. Rename a user:

   Syntax

   ```
   radosgw-admin user rename --uid=CURRENT_USER_NAME --new-uid=NEW_USER_NAME
   ```

   Example

   ```
   [ceph: root@host01 /]# radosgw-admin user rename --uid=user1 --new-uid=user2

   {
       "user_id": "user2",
       "display_name": "user 2",
       "email": "",
       "suspended": 0,
       "max_buckets": 1000,
       "auid": 0,
       "subusers": [],
       "keys": [
           {
               "user": "user2",
               "access_key": "59EKHI6AI9F8WOW8JQZJ",
               "secret_key": "XH0uY3rKCUcuL73X0ftjXbZqUbk0cavD11rD8MsA"
           }
       ],
       "swift_keys": [],
       "caps": [],
       "op_mask": "read, write, delete",
       "default_placement": "",
       "placement_tags": [],
       "bucket_quota": {
           "enabled": false,
           "check_on_raw": false,
           "max_size": -1,
           "max_size_kb": 0,
           "max_objects": -1
       },
       "user_quota": {
           "enabled": false,
           "check_on_raw": false,
           "max_size": -1,
           "max_size_kb": 0,
           "max_objects": -1
       },
       "temp_url_keys": [],
       "type": "rgw"
   }
   ```

   If a user is inside a tenant, specify both the user name and the tenant:

   Syntax

   ```
   radosgw-admin user rename --uid USER_NAME --new-uid NEW_USER_NAME --tenant TENANT
   ```

   Example

   ```
   [ceph: root@host01 /]# radosgw-admin user rename --uid=test$user1 --new-uid=test$user2
   --tenant test

   1000 objects processed in tvtester1. Next marker 80_tVtester1_99
   2000 objects processed in tvtester1. Next marker 64_tVtester1_44
   3000 objects processed in tvtester1. Next marker 48_tVtester1_28
   4000 objects processed in tvtester1. Next marker 2_tVtester1_74
   5000 objects processed in tvtester1. Next marker 14_tVtester1_53
   6000 objects processed in tvtester1. Next marker 87_tVtester1_61
   7000 objects processed in tvtester1. Next marker 6_tVtester1_57
   8000 objects processed in tvtester1. Next marker 52_tVtester1_91
   9000 objects processed in tvtester1. Next marker 34_tVtester1_74
   9900 objects processed in tvtester1. Next marker 9_tVtester1_95
   1000 objects processed in tvtester2. Next marker 82_tVtester2_93
   ```

```
2000 objects processed in tvtester2. Next marker 64_tVtester2_9
3000 objects processed in tvtester2. Next marker 48_tVtester2_22
4000 objects processed in tvtester2. Next marker 32_tVtester2_42
5000 objects processed in tvtester2. Next marker 16_tVtester2_36
6000 objects processed in tvtester2. Next marker 89_tVtester2_46
7000 objects processed in tvtester2. Next marker 70_tVtester2_78
8000 objects processed in tvtester2. Next marker 51_tVtester2_41
9000 objects processed in tvtester2. Next marker 33_tVtester2_32
9900 objects processed in tvtester2. Next marker 9_tVtester2_83
{
    "user_id": "test$user2",
    "display_name": "User 2",
    "email": "",
    "suspended": 0,
    "max_buckets": 1000,
    "auid": 0,
    "subusers": [],
    "keys": [
        {
            "user": "test$user2",
            "access_key": "user2",
            "secret_key": "123456789"
        }
    ],
    "swift_keys": [],
    "caps": [],
    "op_mask": "read, write, delete",
    "default_placement": "",
    "placement_tags": [],
    "bucket_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
    "user_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
    "temp_url_keys": [],
    "type": "rgw"
}
```

2. Verify that the user has been renamed successfully:

   Syntax

   ```
   radosgw-admin user info --uid=NEW_USER_NAME
   ```

   Example

   ```
   [ceph: root@host01 /]# radosgw-admin user info --uid=user2
   ```

   If a user is inside a tenant, use the *TENANT*$*USER_NAME* format:

   Syntax

   ```
   radosgw-admin user info --uid= TENANT$USER_NAME
   ```

   Example

   ```
   [ceph: root@host01 /]# radosgw-admin user info --uid=test$user2
   ```

- The `screen(1)` manual page

Create a key

To create a key for a user, you must specify `key create`. For a user, specify the user ID and the `s3` key type.

To create a key for a subuser, you must specify the subuser ID and the `swift` keytype.

Example

```
[ceph: root@host01 /]# radosgw-admin key create --subuser=johndoe:swift --key-type=swift --
gen-secret

{ "user_id": "johndoe",
  "rados_uid": 0,
  "display_name": "John Doe",
  "email": "john@example.com",
  "suspended": 0,
  "subusers": [
     { "id": "johndoe:swift",
       "permissions": "full-control"}],
  "keys": [
    { "user": "johndoe",
      "access_key": "QFAMEDSJP5DEKJO0DDXY",
      "secret_key": "iaSFLDVvDdQt6lkNzHyW4fPLZugBAI1g17LO0+87"}],
  "swift_keys": [
    { "user": "johndoe:swift",
      "secret_key": "E9T2rUZNu2gxUjcwUBO8n/Ev4KX6/GprEuH4qhu1"}]}
```

Add and remove access keys

Users and subusers must have access keys to use the S3 and Swift interfaces. When you create a user or subuser and you do not specify an access key and secret, the key and secret get generated automatically. You may create a key and either specify or generate the access key and/or secret. You may also remove an access key and secret.

Options include:

- `--secret=SECRET_KEY` specifies a secret key, for example, manually generated.
- `--gen-access-key` generates a random access key (for S3 users by default).
- `--gen-secret` generates a random secret key.
- `--key-type=KEY_TYPE` specifies a key type. The options are: swift and s3.

To add a key, specify the user:

Example

```
[root@host01 ~]# radosgw-admin key create --uid=johndoe --key-type=s3 --gen-access-key --
gen-secret
```

You might also specify a key and a secret.

To remove an access key, you need to specify the user and the key:

1. Find the access key for the specific user:

   Example

   ```
   [root@host01 ~]# radosgw-admin user info --uid=johndoe
   ```

   The access key is the `"access_key"` value in the output:

   Example

   ```
   [root@host01 ~]# radosgw-admin user info --uid=johndoe
   {
       "user_id": "johndoe",
       ...
       "keys": [
           {
               "user": "johndoe",
               "access_key": "0555b35654ad1656d804",
               "secret_key": "h7GhxuBLTrlhVUyxSPUKUV8r/2EI4ngqJxD7iBdBYLhwluN30JaT3Q=="
           }
       ],
   ```

```
    ...
}
```

2. Specify the user ID and the access key from the previous step to remove the access key:

   Syntax

   ```
   radosgw-admin key rm --uid=USER_ID --access-key ACCESS_KEY
   ```

   Example

   ```
   [root@host01 ~]# radosgw-admin key rm --uid=johndoe --access-key 0555b35654ad1656d804
   ```

Add and remove admin capabilities

The Ceph Storage Cluster provides an administrative API that enables users to run administrative functions via the REST API. By default, users DO NOT have access to this API. To enable a user to exercise administrative functionality, provide the user with administrative capabilities.

To add administrative capabilities to a user, run the following command:

Syntax

```
radosgw-admin caps add --uid=USER_ID--caps=CAPS
```

You can add read, write, or all capabilities to users, buckets, metadata, and usage (utilization).

Syntax

```
--caps="[users|buckets|metadata|usage|zone]=[*|read|write|read, write]"
```

Example

```
[root@host01 ~]# radosgw-admin caps add --uid=johndoe --caps="users=*"
```

To remove administrative capabilities from a user, run the following command:

Example

```
[root@host01 ~]# radosgw-admin caps remove --uid=johndoe --caps={caps}
```

Role management

As a storage administrator, you can create, delete, or update a role and the permissions associated with that role with the **radosgw-admin** commands.

A role is similar to a user and has permission policies attached to it. It can be assumed by any identity. If a user assumes a role, a set of dynamically created temporary credentials are returned to the user. A role can be used to delegate access to users, applications and services that do not have permissions to access some S3 resources.

Creating a role

Create a role for the user with the **radosgw-admin role create** command. You need to create a user with the **assume-role-policy-doc** parameter in the command, which is the trust relationship policy document that grants an entity the permission to assume the role.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway.

- Root-level access to a Ceph Object Gateway node.
- An S3 bucket created.
- An S3 user created with user access.

Procedure

- Create the role:

  Syntax

  ```
  radosgw-admin role create --role-name=ROLE_NAME [--path=="PATH_TO_FILE"] [--assume-
  role-policy-doc=TRUST_RELATIONSHIP_POLICY_DOCUMENT]
  ```

  Example

  ```
  [root@host01 ~]# radosgw-admin role create --role-name=S3Access1 --path=/
  application_abc/component_xyz/ --assume-role-policy-
  doc={"Version":"2012-10-17","Statement":[{"Effect":"Allow","Principal":{"AWS":
  ["arn:aws:iam:::user/TESTER"]},"Action":["sts:AssumeRole"]}]}

  {
    "RoleId": "ca43045c-082c-491a-8af1-2eebca13deec",
    "RoleName": "S3Access1",
    "Path": "/application_abc/component_xyz/",
    "Arn": "arn:aws:iam:::role/application_abc/component_xyz/S3Access1",
    "CreateDate": "2022-06-17T10:18:29.116Z",
    "MaxSessionDuration": 3600,
    "AssumeRolePolicyDocument": "{"Version":"2012-10-17","Statement":
  [{"Effect":"Allow","Principal":{"AWS":["arn:aws:iam:::user/TESTER"]},"Action":
  ["sts:AssumeRole"]}]}"
  }
  ```

  The value for `--path` is `/` by default.

Getting a role

Get the information about a role with the **`role get`** command.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway.
- Root-level access to a Ceph Object Gateway node.
- An S3 bucket created.
- A role created.
- An S3 user created with user access.

Procedure

- Getting the information about the role:

  Syntax

  ```
  radosgw-admin role get --role-name=ROLE_NAME
  ```

  Example

  ```
  [root@host01 ~]# radosgw-admin role get --role-name=S3Access1

  {
  ```

```
  "RoleId": "ca43045c-082c-491a-8af1-2eebca13deec",
  "RoleName": "S3Access1",
  "Path": "/application_abc/component_xyz/",
  "Arn": "arn:aws:iam:::role/application_abc/component_xyz/S3Access1",
  "CreateDate": "2022-06-17T10:18:29.116Z",
  "MaxSessionDuration": 3600,
  "AssumeRolePolicyDocument": "{"Version":"2012-10-17","Statement":
[{"Effect":"Allow","Principal":{"AWS":["arn:aws:iam:::user/TESTER"]},"Action":
["sts:AssumeRole"]}]}"
}
```

Reference

For more information, see "Creating a role" on page 168.

Listing a role

You can list the roles in the specific path with the **`role list`** command.

With Ceph 8.0 release, the behavior of radosgw-admin role list has changed in relation to RGW (Rados Gateway) accounts. A simple radosgw-admin role list will **not** list the roles associated with RGW accounts. To list the roles for a specific RGW account, you must specify the account ID by running the following command:

```
radosgw-admin role list --account-id <RGWaccountID>
```

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway.
- Root-level access to a Ceph Object Gateway node.
- An S3 bucket created.
- A role created.
- An S3 user created with user access.

Procedure

- List the roles:

  Syntax

  ```
  radosgw-admin role list --role-name=ROLE_NAME [--path-prefix =PATH_PREFIX]
  ```

  Example

  ```
  [root@host01 ~]# radosgw-admin role list --role-name=S3Access1 --path-prefix="/
  application"

  [
      {
          "RoleId": "85fb46dd-a88a-4233-96f5-4fb54f4353f7",
          "RoleName": "kvm-sts",
          "Path": "/application_abc/component_xyz/",
          "Arn": "arn:aws:iam:::role/application_abc/component_xyz/kvm-sts",
          "CreateDate": "2022-09-13T11:55:09.39Z",
          "MaxSessionDuration": 7200,
          "AssumeRolePolicyDocument": "{"Version":"2012-10-17","Statement":
  [{"Effect":"Allow","Principal":{"AWS":["arn:aws:iam:::user/kvm"]},"Action":
  ["sts:AssumeRole"]}]}"
      },
      {
          "RoleId": "9116218d-4e85-4413-b28d-cdfafba24794",
          "RoleName": "kvm-sts-1",
          "Path": "/application_abc/component_xyz/",
  ```

```
        "Arn": "arn:aws:iam:::role/application_abc/component_xyz/kvm-sts-1",
        "CreateDate": "2022-09-16T00:05:57.483Z",
        "MaxSessionDuration": 3600,
        "AssumeRolePolicyDocument": "{"Version":"2012-10-17","Statement":
[{"Effect":"Allow","Principal":{"AWS":["arn:aws:iam:::user/kvm"]},"Action":
["sts:AssumeRole"]}]}"
    }
]
```

Updating assume role policy document of a role

You can update the assume role policy document that grants an entity permission to assume the role with the **`role-trust-policy modify`** command.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway.
- Root-level access to a Ceph Object Gateway node.
- An S3 bucket created.
- A role created.
- An S3 user created with user access.

Procedure

- Modify the assume role policy document of a role:

  Syntax

  ```
  radosgw-admin role-trust-policy modify --role-name=ROLE_NAME --assume-role-policy-
  doc=TRUST_RELATIONSHIP_POLICY_DOCUMENT
  ```

  Example

  ```
  [root@host01 ~]# radosgw-admin role-trust-policy modify --role-name=S3Access1 --assume-
  role-policy-doc={"Version":"2012-10-17","Statement":[{"Effect":"Allow","Principal":
  {"AWS":["arn:aws:iam:::user/TESTER"]},"Action":["sts:AssumeRole"]}]}

  {
    "RoleId": "ca43045c-082c-491a-8af1-2eebca13deec",
    "RoleName": "S3Access1",
    "Path": "/application_abc/component_xyz/",
    "Arn": "arn:aws:iam:::role/application_abc/component_xyz/S3Access1",
    "CreateDate": "2022-06-17T10:18:29.116Z",
    "MaxSessionDuration": 3600,
    "AssumeRolePolicyDocument": "{"Version":"2012-10-17","Statement":
  [{"Effect":"Allow","Principal":{"AWS":["arn:aws:iam:::user/TESTER"]},"Action":
  ["sts:AssumeRole"]}]}"
  }
  ```

Getting permission policy attached to a role

You can get the specific permission policy attached to a role with the **`role-policy get`** command.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway.
- Root-level access to a Ceph Object Gateway node.

- An S3 bucket created.
- A role created.
- An S3 user created with user access.

Procedure

- Get the permission policy:

  Syntax

  ```
  radosgw-admin role-policy get --role-name=ROLE_NAME --policy-name=POLICY_NAME
  ```

  Example

  ```
  [root@host01 ~]# radosgw-admin role-policy get --role-name=S3Access1 --policy-
  name=Policy1

  {
    "Permission policy": "{"Version":"2012-10-17","Statement":
  [{"Effect":"Allow","Action":["s3:*"],"Resource":"arn:aws:s3:::example_bucket"}]}"
  }
  ```

Listing permission policy attached to a role

You can list the names of the permission policies attached to a role with the **`role-policy list`** command.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway.
- Root-level access to a Ceph Object Gateway node.
- An S3 bucket created.
- A role created.
- An S3 user created with user access.

Procedure

- List the names of the permission policies:

  Syntax

  ```
  radosgw-admin role-policy list --role-name=ROLE_NAME
  ```

  Example

  ```
  [root@host01 ~]# radosgw-admin role-policy list --role-name=S3Access1

  [
    "Policy1"
  ]
  ```

Deleting policy attached to a role

You can delete the permission policy attached to a role with the **`role-policy delete`** command.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway.
- Root-level access to a Ceph Object Gateway node.
- An S3 bucket created.
- A role created.
- An S3 user created with user access.

Procedure

- Delete the permission policy:

  Syntax

  ```
  radosgw-admin role-policy delete --role-name=ROLE_NAME --policy-name=POLICY_NAME
  ```

  Example

  ```
  [root@host01 ~]# radosgw-admin role-policy delete --role-name=S3Access1 --policy-
  name=Policy1
  ```

Deleting a role

You can delete the role only after removing the permission policy attached to it.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway.
- Root-level access to a Ceph Object Gateway node.
- A role created.
- An S3 bucket created.
- An S3 user created with user access.

Procedure

1. Delete the policy attached to the role:

   Syntax

   ```
   radosgw-admin role policy delete --role-name=ROLE_NAME --policy-name=POLICY_NAME
   ```

   Example

   ```
   [root@host01 ~]# radosgw-admin role policy delete --role-name=S3Access1 --policy-
   name=Policy1
   ```

2. Delete the role:

   Syntax

   ```
   radosgw-admin role delete --role-name=ROLE_NAME
   ```

   Example

```
[root@host01 ~]# radosgw-admin role delete --role-name=S3Access1
```

Reference

For more information, see “Deleting policy attached to a role” on page 172.

Updating the session duration of a role

You can update the session duration of a role with the **`role update`** command to control the length of time that a user can be signed into the account with the provided credentials.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway.
- Root-level access to a Ceph Object Gateway node.
- An S3 bucket created.
- A role created.
- An S3 user created with user access.

Procedure

- Update the max-session-duration using the `update` command:

  Syntax

  ```
  [root@node1 ~]# radosgw-admin role update --role-name=ROLE_NAME --max-session-
  duration=7200
  ```

  Example

  ```
  [root@node1 ~]# radosgw-admin role update --role-name=test-sts-role --max-session-
  duration=7200
  ```

Verification

- List the roles to verify the updates:

  Example

  ```
  [root@node1 ~]#radosgw-admin role list
  [
      {
          "RoleId": "d4caf33f-caba-42f3-8bd4-48c84b4ea4d3",
          "RoleName": "test-sts-role",
          "Path": "/",
          "Arn": "arn:aws:iam:::role/test-role",
          "CreateDate": "2022-09-07T20:01:15.563Z",
          "MaxSessionDuration": 7200,                                    <<<<<<
          "AssumeRolePolicyDocument": "{"Version":"2012-10-17","Statement":
  [{"Effect":"Allow","Principal":{"AWS":["arn:aws:iam:::user/kvm"]},"Action":
  ["sts:AssumeRole"]}]}"
      }
  ]
  ```

Static web hosting

As a storage administrator, you can configure the Ceph Object Gateway to host static websites in S3 buckets. Traditional website hosting involves configuring a web server for each website, which can use resources inefficiently when content does not change dynamically. For example, sites that do not use server-side services

like PHP, servlets, databases, nodejs, and the like. This approach is substantially more economical than setting up virtual machines with web servers for each site.

Static web hosting requirements and assumptions

Static web hosting requires at least one running Red Hat Ceph Storage cluster, and at least two Ceph Object Gateway instances for the static web sites.

Red Hat assumes that each zone will have multiple gateway instances using a load balancer, such as high-availability (HA) Proxy and `keepalived`.

> **Important:** Red Hat **DOES NOT** support using a Ceph Object Gateway instance to deploy both standard S3/Swift APIs and static web hosting simultaneously.

Requirements

Use this information to understand static web hosting requirements.

Static web hosting functionality uses its own API, so configuring a gateway to use static web sites in S3 buckets requires the following:

1. S3 static web hosting uses Ceph Object Gateway instances that are separate and distinct from instances used for standard S3/Swift API use cases.
2. Gateway instances hosting S3 static web sites should have separate, non-overlapping domain names from the standard S3/Swift API gateway instances.
3. Gateway instances hosting S3 static web sites should use separate public-facing IP addresses from the standard S3/Swift API gateway instances.
4. Gateway instances hosting S3 static web sites load balance, and if necessary terminate SSL, using HAProxy/keepalive.

Reference

For more information, see the following:

- “High availability service” on page 41

Static web hosting gateway setup

Use this information to help enable Ceph Object GAteway for static web hosting.

To enable a Ceph Object Gateway for static web hosting, set the following options:

Syntax

```
ceph config set client.rgw OPTION VALUE
```

Example

```
[ceph: root@host01 /]# ceph config set client.rgw rgw_enable_static_website true
[ceph: root@host01 /]# ceph config set client.rgw rgw_enable_apis s3,s3website
[ceph: root@host01 /]# ceph config set client.rgw rgw_dns_name objects-
zonegroup.example.com
[ceph: root@host01 /]# ceph config set client.rgw rgw_dns_s3website_name objects-website-
zonegroup.example.com
[ceph: root@host01 /]# ceph config set client.rgw rgw_resolve_cname true
```

The `rgw_enable_static_website` setting MUST be `true`. The `rgw_enable_apis` setting MUST enable the `s3website` API. The `rgw_dns_name` and `rgw_dns_s3website_name` settings must provide their fully qualified domains. If the site uses canonical name extensions, then set the `rgw_resolve_cname` option to `true`.

Important: The FQDNs of `rgw_dns_name` and `rgw_dns_s3website_name` **MUST NOT** overlap.

Static web hosting DNS configuration

Example of assumed DNS settings for a static web hosting.

This is an example of assumed DNS settings, where the first two lines specify the domains of the gateway instance using a standard S3 interface and point to the IPv4 and IPv6 addresses. The third line provides a wildcard CNAME setting for S3 buckets using canonical name extensions. The fourth and fifth lines specify the domains for the gateway instance using the S3 website interface and point to their IPv4 and IPv6 addresses.

```
objects-zonegroup.domain.com. IN    A 192.0.2.10
objects-zonegroup.domain.com. IN AAAA 2001:DB8::192:0:2:10
*.objects-zonegroup.domain.com. IN CNAME objects-zonegroup.domain.com.
objects-website-zonegroup.domain.com. IN    A 192.0.2.20
objects-website-zonegroup.domain.com. IN AAAA 2001:DB8::192:0:2:20
```

Note: The IP addresses in the first two lines differ from the IP addresses in the fourth and fifth lines.

If using Ceph Object Gateway in a multi-site configuration, consider using a routing solution to route traffic to the gateway closest to the client.

The Amazon Web Service (AWS) requires static web host buckets to match the host name. Ceph provides a few different ways to configure the DNS, and HTTPS will work if the proxy has a matching certificate.

Hostname to a bucket on a subdomain

To use AWS-style S3 sub-domains, use a wildcard in the DNS entry which can redirect requests to any bucket. A DNS entry might look like the following:

```
*.objects-website-zonegroup.domain.com. IN CNAME objects-website-zonegroup.domain.com.
```

Access the bucket name, where the bucket name is `bucket1`, in the following manner:

```
http://bucket1.objects-website-zonegroup.domain.com
```

Hostname to non-matching bucket

Ceph supports mapping domain names to buckets without including the bucket name in the request, which is unique to Ceph Object Gateway. To use a domain name to access a bucket, map the domain name to the bucket name. A DNS entry might look like the following, where the bucket name is `bucket2`:

```
www.example.com. IN CNAME bucket2.objects-website-zonegroup.domain.com.
```

Access the bucket in the following manner:

```
http://www.example.com
```

Hostname to long bucket with CNAME

AWS typically requires the bucket name to match the domain name. To configure the DNS for static web hosting using CNAME, the DNS entry might look like the following:

```
www.example.com. IN CNAME www.example.com.objects-website-zonegroup.domain.com.
```

Access the bucket in the following manner:

```
http://www.example.com
```

Hostname to long bucket without CNAME

If the DNS name contains other non-CNAME records, such as `SOA`, `NS`, `MX` or `TXT`, the DNS record must map the domain name directly to the IP address. For example:

```
www.example.com. IN A 192.0.2.20
www.example.com. IN AAAA 2001:DB8::192:0:2:20
```

Access the bucket in the following manner:

```
http://www.example.com
```

Creating a static web hosting site

Use this information for creating a static web hosting site.

To create a static website, perform the following steps:

1. Create an S3 bucket. The bucket name MIGHT be the same as the website's domain name. For example, `mysite.com` may have a bucket name of `mysite.com`. This is required for AWS, but it is NOT required for Ceph.

   For more information, see "Static web hosting DNS configuration" on page 176.

2. Upload the static website content to the bucket. Contents may include HTML, CSS, client-side JavaScript, images, audio/video content, and other downloadable files. A website MUST have an `index.html` file and might have an `error.html` file.

3. Verify the website's contents. At this point, only the creator of the bucket has access to the contents.

4. Set permissions on the files so that they are publicly readable.

Storage classes

Storage classes define how object data is placed and managed in Ceph Object Gateway (RGW). They map objects to specific placement targets and support cost- and performance-optimized data tiering, especially when used with S3 Bucket Lifecycle transitions.

All placement targets include a STANDARD storage class, which is applied to new objects by default. Users can override this default by setting the **`default_storage_class`** value. To store an object in a non-default storage class, include the storage class name in the request header.

S3 protocol

```
X-Amz-Storage-Class
```

Swift protocol

```
X-Object-Storage-Class
```

S3 Object Lifecycle Management can then be used to move object data between storage classes using Transition actions. When using AWS S3 SDKs, such as boto3, it is important that storage class names match those provided by AWS S3, or else the SDK will drop the request and raise an exception. Moreover, some S3 clients and libraries expect AWS-specific behavior when a storage class named or prefixed with `GLACIER` is used and thus will fail when accessing Ceph Object Gateway services. For this reason we advise that other storage class names be used with Ceph, including `INTELLIGENT-TIERING`, `STANDARD_IA`, `REDUCED_REDUNDANCY`, and `ONEZONE_IA`. Custom storage class names like `CHEAPNDEEP` are accepted by Ceph but might not be by some clients and libraries.

Use cases

Storage classes are commonly used in the following scenarios to optimize data placement, cost, and performance:

- Moving infrequently accessed objects to low-cost pools using automated lifecycle transitions.
- Assigning latency-sensitive or frequently accessed workloads to faster pools.
- Creating custom storage classes for compliance, isolation, or application-specific data placement (for example, APP_LOGS, ML_DATA).

- Automating multi-tier lifecycle transitions such as: STANDARD → STANDARD_IA → archival pool based on age or access patterns.
- Applying different durability or resiliency profiles by mapping storage classes to pools with varied replication or erasure coding.
- Separating workloads such as analytics, logging, or backup data into pools with optimized compression or cost models.

Adding a storage class

Add a new storage class to a placement target in Red Hat Ceph Storage Object Gateway and associate it with a data pool and compression settings.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- You have administrator privileges to run **`radosgw-admin`** commands.
- The zonegroup and zone are already configured in the Ceph Object Gateway.
- The required data pool (for example, `default.rgw.glacier.data`) exists in the Ceph cluster.
- The `radosgw-admin` tool is installed and available on the system where you will run the commands.

1. Run the following command to add the new storage class to the zonegroup placement target.

```
radosgw-admin zonegroup placement add \
  --rgw-zonegroup default \

  --placement-id default-placement \

  --storage-class STANDARD_IA
```

   For example,

```
radosgw-admin zonegroup placement add \
  --rgw-zonegroup prod-zonegroup \
  --placement-id app-placement \
  --storage-class APP_LOGS
```

   This command updates the zonegroup's placement configuration to include the new STANDARD_IA storage class.

2. Run the following command to define the zone-specific placement configuration for the storage class.

```
radosgw-admin zone placement add \
  --rgw-zone DEFAULT \
  --placement-id DEFAULT_PLACEMENT \
  --storage-class STANDARD_IA \
  --data-pool DEFAULT.rgw.GLACIER.data \
  --compression lz4
```

   For example,

```
radosgw-admin zone placement add \
  --rgw-zone prod-zone \
  --placement-id app-placement \
  --storage-class APP_LOGS \
  --data-pool prod.rgw.logs.data \
  --compression lz4
```

   This command maps the STANDARD_IA storage class to the specified data pool with LZ4 compression.

Result

The new storage class STANDARD_IA is now available for use in the specified placement target. You can use it when uploading objects via S3 headers or configure transitions using S3 Bucket Lifecycle rules.

Ceph Object Gateway data layout

Ceph Object Gateway organizes its data into metadata, bucket index, and data.

While RADOS recognizes pools and objects by their Extended Attributes (`xattrs`) and object map (OMAP), conceptually Ceph Object Gateway organizes its data into three different kinds:

Metadata

There are three sections of metadata:

`user`

Holds user information.

`bucket`

Holds a mapping between bucket name and bucket instance ID.

`bucket.instance`

Holds bucket instance information.

Use the following commands to view metadata entries:

```
radosgw-admin metadata get bucket:BUCKET_NAME
radosgw-admin metadata get bucket.instance:BUCKET:BUCKET_ID
radosgw-admin metadata get user:USER
radosgw-admin metadata set user:USER
```

For example,

```
[ceph: root@host01 /]# radosgw-admin metadata list
[ceph: root@host01 /]# radosgw-admin metadata list bucket
[ceph: root@host01 /]# radosgw-admin metadata list bucket.instance
[ceph: root@host01 /]# radosgw-admin metadata list user
```

Every metadata entry is kept on a single RADOS object.

Important: When using the `radosgw-admin` tool, ensure that the tool and the Ceph Cluster are of the same version. The use of mismatched versions is *not* supported.

Note: A Ceph Object Gateway object might consist of several RADOS objects. The first type of object is the head that contains the metadata, such as manifest, Access Control List (ACL), content type, ETag, and user-defined metadata. The metadata is stored in `xattrs`. The head might also contain up to 512 KB of object data, for efficiency and atomicity. The manifest describes how each object is laid out in RADOS objects.

Bucket index

Bucket index is a different kind of metadata, and kept separately. The bucket index holds a key-value map in RADOS objects. By default, it is a single RADOS object per bucket, but it is possible to shard the map over multiple RADOS objects.

The map itself is kept in OMAP associated with each RADOS object. The key of each OMAP is the name of the objects, and the value holds some basic metadata of that object, the metadata that appears when listing the bucket. Each OMAP holds a header, and some bucket accounting metadata is kept in that header such as number of objects, total size, and the like.

Note: OMAP is a key-value store, which is associated with an object, in a way similar to how extended attributes associate with a POSIX file. An object's OMAP is not physically located in the object's storage, but its precise implementation is invisible and immaterial to the Ceph Object Gateway.

Data

Objects data is kept in one or more RADOS objects for each Ceph Object Gateway object.

Object lookup path

When accessing objects, REST APIs come to Ceph Object Gateway with account information, bucket name or key, and object names. The Ceph Object Gateway uses the account information to find out the user ID and for access control. The bucket name and object key are used to address the object in a pool.

Account information

The account information has the access key in S3 or the account name in Swift.
The user ID in Ceph Object Gateway is a string, typically the actual username from the user credentials and not a hashed or mapped identifier.

When accessing a user's data, the user record is loaded from an object `USER_ID` in the `default.rgw.meta` pool with `users.uid` namespace.

Bucket or container name

The bucket or container names are represented in the `default.rgw.meta` pool with `root` namespace. Bucket record is loaded to obtain a marker, which serves as a bucket ID.

Object name or key

The object is located in the `default.rgw.buckets.data` pool. Object name is `MARKER_KEY`, for example `default.7593.4_image.png`, where the marker is `default.7593.4` and the key is `image.png`. These concatenated names are not parsed and are passed down to RADOS only. Therefore, the choice of the separator is not important and causes no ambiguity. For the same reason, you can use slashes in object names, such as keys.

Multiple data pools

It is possible to create multiple data pools so that different users' buckets are created in different RADOS pools by default, thus providing the necessary scaling. The layout and naming of these pools is controlled by a `policy` setting.

Optimize the Ceph Object Gateway's garbage collection

When new data objects are written into the storage cluster, the Ceph Object Gateway immediately allocates the storage for these new objects. After you delete or overwrite data objects in the storage cluster, the Ceph Object Gateway deletes those objects from the bucket index. Some time afterward, the Ceph Object Gateway then purges the space that was used to store the objects in the storage cluster. The process of purging the deleted object data from the storage cluster is known as Garbage Collection, or GC.

Garbage collection operations typically run in the background. You can configure these operations to either run continuously, or to run only during intervals of low activity and light workloads. By default, the Ceph Object Gateway conducts GC operations continuously. Because GC operations are a normal part of Ceph Object Gateway operations, deleted objects that are eligible for garbage collection exist most of the time.

Viewing the garbage collection queue

Before you purge deleted and overwritten objects from the storage cluster, use the **`radosgw-admin`** command to view the objects awaiting garbage collection.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the Ceph Object Gateway.

Procedure

- To view the queue of objects awaiting garbage collection:

  Example

  ```
  [ceph: root@host01 /]# radosgw-admin gc list
  ```

> **Note:** To list all entries in the queue, including unexpired entries, use the `--include-all` option.

Adjusting garbage collection settings

Adjust garbage collection settings to increase priority or tune the garbage collection queue.

The Ceph Object Gateway allocates storage for new and overwritten objects immediately. Also, the parts of a multi-part upload also use some storage.

The Ceph Object Gateway purges the storage space that is used for deleted objects after deleting the objects from the bucket index. Similarly, the Ceph Object Gateway will delete data that is associated with a multi-part upload after the multi-part upload completes or when the upload has gone inactive or failed to complete for a configurable amount of time.

Viewing the objects awaiting garbage collection can be done with the following command:

```
radosgw-admin gc list
```

Garbage collection is a background activity that runs continuously or during times of low loads, depending upon how the storage administrator configures the Ceph Object Gateway. By default, the Ceph Object Gateway conducts garbage collection operations continuously. Since garbage collection operations are a normal function of the Ceph Object Gateway, especially with object delete operations, objects eligible for garbage collection exist most of the time.

Some workloads can temporarily or permanently outpace the rate of garbage collection activity. This is especially true of delete-heavy workloads, where many objects get stored for a short period of time and then deleted. For these types of workloads, storage administrators can increase the priority of garbage collection operations relative to other operations with the following configuration parameters:

- The `rgw_gc_obj_min_wait` configuration option waits a minimum length of time, in seconds, before purging a deleted object's data. The default value is two hours, or 7200 seconds. The object is not purged immediately because a client might be reading the object. Under heavy workloads, this setting can use too much storage or have many deleted objects to purge. Do not set this value below 30 minutes, or 1800 seconds.
- The `rgw_gc_processor_period` configuration option is the garbage collection cycle run time. That is, the amount of time between the start of consecutive runs of garbage collection threads. If garbage collection runs longer than this period, the Ceph Object Gateway will not wait before running a garbage collection cycle again.
- The `rgw_gc_max_concurrent_io` configuration option specifies the maximum number of concurrent IO operations that the gateway garbage collection thread uses when purging deleted data. Under delete heavy workloads, consider increasing this setting to a larger number of concurrent IO operations.
- The `rgw_gc_max_trim_chunk` configuration option specifies the maximum number of keys to remove from the garbage collector log in a single operation. Under delete heavy operations, consider increasing the maximum number of keys so that more objects are purged during each garbage collection operation.

In addition, the following configuration parameters tune the garbage collection queue, as follows:

- The `rgw_gc_max_deferred_entries_size` configuration option sets the maximum size of deferred entries in the garbage collection queue.
- The `rgw_gc_max_queue_size` configuration option sets the maximum queue size that is used for garbage collection. Do not set this value greater than `osd_max_object_size` minus `rgw_gc_max_deferred_entries_size` minus 1 KB.
- The `rgw_gc_max_deferred` configuration option sets the maximum number of deferred entries that are stored in the garbage collection queue.

Note: In testing, with an evenly balanced delete-write workload, such as 50% delete and 50% write operations, the storage cluster fills completely in 11 hours. This is because Ceph Object Gateway garbage collection fails to keep pace with the delete operations. The cluster status switches to the `HEALTH_ERR` state if this happens. Aggressive settings for parallel garbage collection tunables significantly delayed the onset of storage cluster fill in testing and can be helpful for many workloads. Typical real-world storage cluster workloads are not likely to cause a storage cluster fill primarily due to garbage collection.

Adjusting garbage collection for delete-heavy workloads

Delete-heavy workloads can outpace garbage collection when large numbers of short-lived objects are created and removed in a short time.

In some environments, garbage collection activity may not keep up with the rate at which objects are deleted, either temporarily or on an ongoing basis. This situation commonly occurs in delete-heavy workloads, where objects are stored briefly and then removed. In these cases, increasing the priority of garbage collection operations relative to other operations can help maintain system stability. For additional questions about Ceph Object Gateway garbage collection, contact Red Hat Support.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to all nodes in the storage cluster.

Procedure

1. Set the value of `rgw_gc_max_concurrent_io` to `20`, and the value of `rgw_gc_max_trim_chunk` to `64`:

   Example

   ```
   [ceph: root@host01 /]# ceph config set client.rgw rgw_gc_max_concurrent_io 20
   [ceph: root@host01 /]# ceph config set client.rgw rgw_gc_max_trim_chunk 64
   ```

2. Restart the Ceph Object Gateway to allow the changed settings to take effect.
3. Monitor the storage cluster during GC activity to verify that the increased values do not adversely affect performance.

Important: Never modify the value for the `rgw_gc_max_objs` option in a running cluster. You should only change this value before deploying the RGW nodes.

Reference

For more information, see the following:

- *Red Hat Ceph RGW - Garbage Collection (GC) Tuning Options*
- “General settings” on page 298

Bucket and object listing

Buckets that belong to a given user are listed in an OMAP of an object named `USER_ID.buckets`, for example, foo.buckets, in the `default.rgw.meta` pool with `users.uid` namespace. These objects are accessed when listing buckets, when updating bucket contents, and updating and retrieving bucket statistics such as quota. These listings are kept consistent with buckets in the `.rgw` pool.

> **Note:** See the user-visible, encoded class `cls_user_bucket_entry` and its nested class `cls_user_bucket` for the values of these OMAP entries.

Objects that belong to a given bucket are listed in a bucket index. The default naming for index objects is `.dir.MARKER` in the `default.rgw.buckets.index` pool.

Reference

For more information, see "Configure bucket index resharding" on page 247.

Object Gateway data layout parameters

This information provides a list of data layout parameters for Ceph Object Gateway.

Known pools:

`.rgw.root` Unspecified region, zone, and global information records, one per object.

`ZONE.rgw.control` notify.*N*

`ZONE.rgw.meta` Multiple namespaces with different kinds of metadata

namespace: root BUCKET.bucket.meta.BUCKET:MARKER # see put_bucket_instance_info() The tenant is used to disambiguate buckets, but not bucket instances.

Example

```
.bucket.meta.prodtx:test%25star:default.84099.6
.bucket.meta.testcont:default.4126.1
.bucket.meta.prodtx:testcont:default.84099.4
prodtx/testcont
prodtx/test%25star
testcont
```

namespace: users.uid Contains per-user information (RGWUserInfo) in `USER` objects and per-user lists of buckets in omaps of `USER.buckets` objects. The `USER` might contain the tenant if non-empty.

Example

```
prodtx$prodt
test2.buckets
prodtx$prodt.buckets
test2
```

namespace: users.email Unimportant

namespace: users.keys 47UA98JSTJZ9YAN3OS3O

This allows Ceph Object Gateway to look up users by their access keys during authentication.

namespace: users.swift test:tester

`ZONE.rgw.buckets.index` Objects are named `.dir.MARKER`, each contains a bucket index. If the index is sharded, each shard appends the shard index after the marker. dow_].488urDFerTYXavx4yAd-Op8mxehnvTI_1 MARKER_pass:KEY

An example of a marker would be `default.16004.1` or `default.7593.4`. The current format is `ZONE.INSTANCE_ID.BUCKET`, but once generated, a marker is not parsed again, so its format might change freely in the future.

ZONE.rgw.buckets.data default.7593.4__shadow_.488urDFerTYXavx4yAd-Op8mxehnvTI_1 MARKER_KEY

An example of a marker would be `default.16004.1` or `default.7593.4`. The current format is `ZONE.INSTANCE_ID.BUCKET_ID`, but once generated, a marker is not parsed again, so its format might change freely in the future.

Reference

For more information, see the following:

- "Ceph Object Gateway data layout" on page 179

Rate limits for ingesting data

As a storage administrator, you can set rate limits on users and buckets based on the operations and bandwidth when saving an object in a Red Hat Ceph Storage cluster with a Ceph Object Gateway configuration. The rate limit includes the maximum number of read operations, write operations per minute, and how many bytes per minute can be written or read per user or per bucket.

Requests that use GET or HEAD method in the REST are either read or write requests.

Important: Ensure to have the following permissions in place for ratelimit SET and GET operations through REST API:

- To get a rate limit, the user must have ratelimit capability set with read permission.

```
radosgw-admin caps add --uid=UID --caps='ratelimit=read'
```

- To set a rate limit, the user must have ratelimit capability set with write permission.

```
radosgw-admin caps add --uid=UID --caps='ratelimit=write'
```

The Ceph Object Gateway tracks the user and bucket requests separately and does not share with other gateways, which means that the desired limits configured should be divided by the number of active Object Gateways.

For example, if user A should be limited by ten ops per minute and there are two Ceph Object Gateways in the cluster, the limit over user A should be five, that is, ten ops per minute for two Ceph Object Gateways. If the requests are not balanced between Ceph Object Gateways, the rate limit may be underutilized. For example, if the ops limit is five and there are two Ceph Object Gateways, but the load balancer sends load only to one of those Ceph Object Gateways, the effective limit would be five ops, because this limit is enforced per Ceph Object Gateway.

If there is a limit reached for the bucket, but not for the user, or vice versa the request would be canceled as well.

The bandwidth counting happens after the request is accepted.As a result, this request proceeds even if the bucket or the user has reached its bandwidth limit in the middle of the request.

The Ceph Object Gateway keeps a debt of used bytes more than the configured value and prevents this user or bucket from sending more requests until their debt is paid. The debt maximum size is twice the max-read/write-bytes per minute. If user A has 1 byte read limit per minute and this user tries to GET 1 GB object, the user can do it.

After user A completes this 1 GB operation, the Ceph Object Gateway blocks the user request for up to two minutes until user A is able to send the GET request again.

The following are different options for limiting rates:

Bucket

The **`--bucket`** option allows you to specify a rate limit for a bucket.

User

The **`--uid`** option allows you to specify a rate limit for a user.

Maximum read ops

The **--max-read-ops** setting allows you to specify the maximum number of read ops per minute per Ceph Object Gateway.
A value of `0` disables this setting, which means unlimited access.

Maximum read bytes

The **--max-read-bytes** setting allows you to specify the maximum number of read bytes per minute per Ceph Object Gateway.
A value of `0` disables this setting, which means unlimited access.

Maximum write ops

The **--max-write-ops** setting allows you to specify the maximum number of write ops per minute per Ceph Object Gateway.
A value of `0` disables this setting, which means unlimited access.

Maximum write bytes

The **--max-write-bytes** setting allows you to specify the maximum number of write bytes per minute per Ceph Object Gateway.
A value of `0` disables this setting, which means unlimited access.

Rate limit scope

The **--rate-limit-scope** option sets the scope for the rate limit. The options are`bucket`,`user`, and `anonymous`. Bucket rate limit applies to buckets, user rate limit applies to a user, and anonymous applies to an unauthenticated user.
Anonymous scope is only available for global rate limit.

Managing user rate limits

You can set rate limits on users in a Ceph Object Gateway configuration. The rate limit on users include the maximum number of read operations, write operations per minute, and how many bytes per minute can be written or read per user.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- A Ceph Object Gateway installed.

Enabling user rate limits

You can enable the rate limit on users after setting the value of rate limits by using the **radosgw-admin ratelimit set** command with the **ratelimit-scope** set as `user`.

1. Set the rate limit for the user.

```
radosgw-admin ratelimit set --ratelimit-scope=user --uid=USER_ID [--max-read-
ops=NUMBER_OF_OPERATIONS] [--max-read-bytes=NUMBER_OF_BYTES][--max-write-
ops=NUMBER_OF_OPERATIONS] [--max-write-bytes=NUMBER_OF_BYTES]
```

   Inputting a value of `0` for *NUMBER_OF_OPERATIONS* or *NUMBER_OF_BYTES* disables that specific rate limit attribute check.

   For example,

```
[ceph: root@host01 /]# radosgw-admin ratelimit set --ratelimit-scope=user --
uid=testing --max-read-ops=1024 --max-write-bytes=10240
```

2. Get the user rate limit.

```
radosgw-admin ratelimit get --ratelimit-scope=user --uid=USER_ID
```

   For example,

```
[ceph: root@host01 /]# radosgw-admin ratelimit get --ratelimit-scope=user --
uid=testing

{
    "user_ratelimit": {
        "max_read_ops": 1024,
        "max_write_ops": 0,
        "max_read_bytes": 0,
        "max_write_bytes": 10240,
        "enabled": false
    }
}
```

3. Enable the user rate limit.

```
radosgw-admin ratelimit enable --ratelimit-scope=user --uid=_USER_ID
```

For example,

```
[ceph: root@host01 /]# radosgw-admin ratelimit enable --ratelimit-scope=user --
uid=testing

{
    "user_ratelimit": {
        "max_read_ops": 1024,
        "max_write_ops": 0,
        "max_read_bytes": 0,
        "max_write_bytes": 10240,
        "enabled": true
    }
}
```

Disabling user rate limits

- Disable user rate limits by running the **`ratelimit disable`** command.

```
radosgw-admin ratelimit disable --ratelimit-scope=user --uid=USER_ID
```

For example,

```
[ceph: root@host01 /]# radosgw-admin ratelimit disable --ratelimit-scope=user --
uid=testing
```

Managing bucket rate limits

You can set rate limits on buckets in a Ceph Object Gateway configuration. The rate limit on buckets include the maximum number of read operations, write operations per minute, and how many bytes per minute can be written or read per bucket.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- A Ceph Object Gateway installed.

Enabling user rate limits

You can enable the rate limit on buckets after setting the value of rate limits by using the **`radosgw-admin ratelimit set`** command with the **`ratelimit-scope`** set as `bucket`.

1. Set the rate limit for the bucket.

```
radosgw-admin ratelimit set --ratelimit-scope=bucket --bucket=BUCKET_NAME [--max-read-
ops=NUMBER_OF_OPERATIONS] [--max-read-bytes=NUMBER_OF_BYTES][--max-write-
ops=NUMBER_OF_OPERATIONS] [--max-write-bytes=NUMBER_OF_BYTES]
```

Inputting a value of `0` for *NUMBER_OF_OPERATIONS* or *NUMBER_OF_BYTES* disables that specific rate limit attribute check.

For example,

```
[ceph: root@host01 /]# radosgw-admin ratelimit set --ratelimit-scope=bucket --
bucket=mybucket --max-read-ops=1024 --max-write-bytes=10240
```

2. Get the bucket rate limit.

```
radosgw-admin ratelimit get --ratelimit-scope=bucket --bucket=BUCKET_NAME
```

For example,

```
[ceph: root@host01 /]# radosgw-admin ratelimit get --ratelimit-scope=bucket --
bucket=mybucket

{
    "bucket_ratelimit": {
        "max_read_ops": 1024,
        "max_write_ops": 0,
        "max_read_bytes": 0,
        "max_write_bytes": 10240,
        "enabled": false
    }
}
```

3. Enable the bucket rate limit.

```
radosgw-admin ratelimit enable --ratelimit-scope=bucket --bucket=BUCKET_NAME
```

For example,

```
[ceph: root@host01 /]# radosgw-admin ratelimit enable --ratelimit-scope=bucket --
bucket=mybucket

{
    "bucket_ratelimit": {
        "max_read_ops": 1024,
        "max_write_ops": 0,
        "max_read_bytes": 0,
        "max_write_bytes": 10240,
        "enabled": true
    }
}
```

Disabling the bucket rate limits

- Disable bucket rate limits by running the **`ratelimit disable`** command.

```
radosgw-admin ratelimit disable --ratelimit-scope=bucket --bucket=BUCKET_NAME
```

For example,

```
[ceph: root@host01 /]# radosgw-admin ratelimit disable --ratelimit-scope=bucket --
bucket=mybucket
```

Managing global rate limits

You can read or write global rate limit settings in period configuration.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- A Ceph Object Gateway installed.

You can override the user or bucket rate limit configuration by manipulating the global rate limit settings with the **`global ratelimit`** parameter, which is the counterpart of **`ratelimit set`**, **`ratelimit enable`**, and **`ratelimit disable`** commands.

> **Note:** In a multi-site configuration, where there is a realm and period present, changes to the global rate limit must be committed using **`period update --commit`** command. If there is no period present, the Ceph Object Gateways must be restarted for the changes to take effect.

Enabling user rate limits

You can enable the rate limit on buckets after setting the value of rate limits by using the **`radosgw-admin ratelimit set`** command with the **`ratelimit-scope`** set as `bucket`.

1. Get the global rate limit.

```
radosgw-admin global ratelimit get
```

   For example,

```
[ceph: root@host01 /]# radosgw-admin global ratelimit get

{
    "bucket_ratelimit": {
        "max_read_ops": 1024,
        "max_write_ops": 0,
        "max_read_bytes": 0,
        "max_write_bytes": 0,
        "enabled": false
    },
    "user_ratelimit": {
        "max_read_ops": 0,
        "max_write_ops": 0,
        "max_read_bytes": 0,
        "max_write_bytes": 0,
        "enabled": false
    },
    "anonymous_ratelimit": {
        "max_read_ops": 0,
        "max_write_ops": 0,
        "max_read_bytes": 0,
        "max_write_bytes": 0,
        "enabled": false
    }
}
```

2. Configure and enable rate limit scope for the buckets.
   a. Set the global rate limits for bucket.

```
radosgw-admin global ratelimit set --ratelimit-scope=bucket [--max-read-
ops=NUMBER_OF_OPERATIONS] [--max-read-bytes=NUMBER_OF_BYTES][--max-write-
ops=NUMBER_OF_OPERATIONS] [--max-write-bytes=NUMBER_OF_BYTES]
```

      For example,

```
[ceph: root@host01 /]# radosgw-admin global ratelimit set --ratelimit-scope
bucket --max-read-ops=1024
```

   b. Enable the bucket rate limit.

```
radosgw-admin global ratelimit enable --ratelimit-scope=bucket
```

      For example,

```
[ceph: root@host01 /]# radosgw-admin global ratelimit enable --ratelimit-scope
bucket
```

3. Configure and enable rate limit scope for authenticated users.
   a. Set the global rate limits for users.

```
radosgw-admin global ratelimit set --ratelimit-scope=user [--max-read-
ops=NUMBER_OF_OPERATIONS] [--max-read-bytes=NUMBER_OF_BYTES][--max-write-
ops=NUMBER_OF_OPERATIONS] [--max-write-bytes=NUMBER_OF_BYTES]
```

For example,

```
[ceph: root@host01 /]# radosgw-admin global ratelimit set --ratelimit-scope=user
--max-read-ops=1024
```

b. Enable the user rate limit.

```
radosgw-admin global ratelimit enable --ratelimit-scope=user
```

For example,

```
[ceph: root@host01 /]# radosgw-admin global ratelimit enable --ratelimit-
scope=user
```

4. Configure and enable rate limit scope for unauthenticated users.

a. Set the global rate limits for unauthenticated users.

```
radosgw-admin global ratelimit set --ratelimit-scope=anonymous [--max-read-
ops=NUMBER_OF_OPERATIONS] [--max-read-bytes=NUMBER_OF_BYTES][--max-write-
ops=NUMBER_OF_OPERATIONS] [--max-write-bytes=NUMBER_OF_BYTES]
```

For example,

```
[ceph: root@host01 /]# radosgw-admin global ratelimit set --ratelimit-
scope=anonymous --max-read-ops=1024
```

b. Enable the user rate limit.

```
radosgw-admin global ratelimit enable --ratelimit-scope=anonymous
```

For example,

```
[ceph: root@host01 /]# radosgw-admin global ratelimit enable --ratelimit-
scope=anonymous
```

Authenticating with Ceph Object Gateway

Configure the Red Hat Directory Server to authenticate Ceph Object Gateway users.

To configure the Red Hat Directory Server to authenticate Ceph Object Gateway users, perform the following steps:

1. “Installing the Red Hat Directory Server” on page 190
2. “Configuring the Directory Server firewall” on page 191
3. “Labeling ports for SELinux” on page 191
4. “Configuring LDAPS” on page 191
5. “Adding a gateway user” on page 192
6. “Configuring the gateway to use LDAP” on page 192
7. “Using a custom search filter” on page 193
8. “Adding an S3 user to the LDAP server” on page 194
9. “Exporting an LDAP token” on page 194
10. “Testing the configuration with an S3 client” on page 194

LDAP integration

To integrate LDAP with the Ceph Object Gateway, you must perform several steps.

You must install the Red Hat Directory Server on a Red Hat Enterprise Linux 9 with a graphical user interface (GUI) to use the Java Swing GUI Directory and Administration consoles. Alternatively, you can still service the Directory Server exclusively from the command line interface (CLI). Next, you must configure the Directory Server firewall to allow access to the Directory Server's secure (636) port. After that, you need to label the ports for SELinux to ensure that they do not block requests.

After the LDAP is working, you need to configure the Ceph Object Gateway servers to trust the Directory Server's certificate. You can extract/download a PEM-formatted certificate for the Certificate Authority (CA) that signed the LDAP server's SSL certificate and add it to the store at `/etc/openldap/certs` using the certutil command.

After that, you must create an LDAP user for the Ceph Object Gateway and configure the gateway to use LDAP. You can use a custom search filter to limit user access by using the rgw_ldap_searchfilter setting.

Finally, you need to create at least one S3 user so that an S3 client can use the LDAP user credentials and export an LDAP token when running Ceph Object Gateway with LDAP. The access token is created from the access key and secret key.

Following are the steps to configure LDAP and Ceph Object Gateway::

Installing the Red Hat Directory Server

Red Hat Directory Server should be installed on a Red Hat Enterprise Linux 9 with a graphical user interface (GUI) in order to use the Java Swing GUI Directory and Administration consoles. However, Red Hat Directory Server can still be serviced exclusively from the command line interface (CLI).

Prerequisites

- Red Hat Enterprise Linux (RHEL) is installed on the server.
- The Directory Server node's FQDN is resolvable using DNS or the `/etc/hosts` file.
- Register the Directory Server node to the Red Hat subscription management service.
- A valid Red Hat Directory Server subscription is available in your Red Hat account.

Procedure

- Follow the instructions in *Installing the Directory Server packages* and *Setting up a new Directory Server instance* of the Red Hat Directory Server Installation Guide.

Reference

For more information, see the following:

- *Red Hat Director Server Installation Guide*

Configuring LDAPS

The Ceph Object Gateway uses a simple ID and password to authenticate with the LDAP server, so the connection requires an SSL certificate for LDAP. Once the LDAP is working, configure the Ceph Object Gateway servers to trust the Directory Server's certificate.

1. Extract/Download a PEM-formatted certificate for the Certificate Authority (CA) that signed the LDAP server's SSL certificate.
2. Confirm that `/etc/openldap/ldap.conf` does not have `TLS_REQCERT` set.
3. Confirm that `/etc/openldap/ldap.conf` contains a `TLS_CACERTDIR /etc/openldap/certs` setting.
4. Use the `certutil` command to add the AD CA to the store at `/etc/openldap/certs`. For example, if the CA is "msad-frog-MSAD-FROG-CA", and the PEM-formatted CA file is `ldap.pem`, use the following command:

   Example

   ```
   # certutil -d /etc/openldap/certs -A -t "TC,," -n "msad-frog-MSAD-FROG-CA" -i /path/
   to/ldap.pem
   ```

5. Update SELinux on all remote LDAP sites:

   Example

   ```
   # setsebool -P httpd_can_network_connect on
   ```

   > **Note:** This still has to be set even if SELinux is in permissive mode.

6. Make the `certs` database world-readable:

   Example

   ```
   # chmod 644 /etc/openldap/certs/*
   ```

7. Connect to the server using the "ldapwhoami" command as a non-root user.

   Example

   ```
   $ ldapwhoami -H ldaps://redhat-directory-server.example.com -d 9
   ```

   The `-d 9` option will provide debugging information in case something went wrong with the SSL negotiation.

Configuring the Directory Server firewall

On the LDAP host, make sure that the firewall allows access to the Directory Server's secure (636) port, so that LDAP clients can access the Directory Server.

Leave the default unsecure port (389) closed.

```
# firewall-cmd --zone=public --add-port=636/tcp
# firewall-cmd --zone=public --add-port=636/tcp --permanent
```

Labeling ports for SELinux

To ensure SELinux does not block requests, label the ports for SELinux.

For details see the *Changing the LDAP and LDAPS Port Numbers*.

Adding a gateway user

Create an LDAP user for the Ceph Object Gateway.

PREREQUISITE

Before creating the gateway user, ensure that the Ceph Object Gateway does not already have the user.

```
radosgw-admin metadata list user
```

For example,

```
[ceph: root@host01 /]# radosgw-admin metadata list user
```

Look in the output to be sure that the username is not in this list of users.

1. Create an LDAP user for the Ceph Object Gateway, and make note of the `binddn`.
   Since the Ceph Object Gateway uses the `ceph` user, consider using `ceph` as the username. The user needs to have permission to search the directory. The Ceph Object Gateway binds to this user as specified in `rgw_ldap_binddn`.
2. Test to verify that the user creation worked.
   You can run a search for the user, when the user ID (`uid`) is `ceph` for `People` with the `example.com` domain.

   ```
   # ldapsearch -x -D "uid=ceph,ou=People,dc=example,dc=com" -W -H ldaps://example.com -b
   "ou=People,dc=example,dc=com" -s sub 'uid=ceph'
   ```

3. On each gateway node, create a file for the user's secret.
   For example, the secret might get stored in a file entitled `/etc/bindpass`.
   For security, change the owner of this file to the `ceph` user and group to ensure that it is not globally readable.
4. Add the `rgw_ldap_secret` option.

   ```
   ceph config set client.rgw OPTION VALUE
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph config set client.rgw rgw_ldap_secret /etc/bindpass
   ```

5. Patch the bind password file to the Ceph Object Gateway container and reapply the Ceph Object Gateway specification.
   For example,

   ```
   service_type: rgw
   service_id: rgw.1
   service_name: rgw.rgw.1
   placement:
     label: rgw
     extra_container_args:
     - -v
     - /etc/bindpass:/etc/bindpass
   ```

   > **Note:** `/etc/bindpass` is not included automatically with Red Hat Ceph Storage. Check that the content is available on all the possible Ceph Object Gateway instance nodes.

Configuring the gateway to use LDAP

Configure the Ceoh Object Gateway to be able to use LDAP.

1. Change the Ceph configuration with the following commands on all the Ceph nodes.
   Syntax

```
ceph config set client.rgw OPTION VALUE
```

```
[ceph: root@host01 /]# ceph config set client.rgw rgw_ldap_uri ldaps://:636
[ceph: root@host01 /]# ceph config set client.rgw rgw_ldap_binddn
"ou=poc,dc=example,dc=local"
[ceph: root@host01 /]# ceph config set client.rgw rgw_ldap_searchdn
"ou=poc,dc=example,dc=local"
[ceph: root@host01 /]# ceph config set client.rgw rgw_ldap_dnattr "uid"
[ceph: root@host01 /]# ceph config set client.rgw rgw_s3_auth_use_ldap true
```

2. Restart the Ceph Object Gateway.

> **Note:** Use the output from the `ceph orch ps` command, under the `NAME` column, to get the `SERVICE_TYPE.ID` information.

a. To restart the Ceph Object Gateway on an individual node in the storage cluster.

Syntax

```
systemctl restart ceph-CLUSTER_ID@SERVICE_TYPE.ID.service
```

Example

```
[root@host01 ~]# systemctl restart ceph-c4b34c6f-8365-11ba-
dc31-529020a7702d@rgw.realm.zone.host01.gwasto.service
```

b. To restart the Ceph Object Gateways on all nodes in the storage cluster.

Syntax

```
ceph orch restart SERVICE_TYPE
```

Example

```
[ceph: root@host01 /]# ceph orch restart rgw
```

Using a custom search filter

You can create a custom search filter to limit user access by using the `rgw_ldap_searchfilter` setting.

There are two ways to use the `rgw_ldap_searchfilter` setting:

1. Specifying a partial filter:

Example

```
"objectclass=inetorgperson"
```

The Ceph Object Gateway generates the search filter with the user name from the token and the value of `rgw_ldap_dnattr`. The constructed filter is then combined with the partial filter from the `rgw_ldap_searchfilter` value. For example, the user name and the settings generate the final search filter:

Example

```
"(&(uid=joe)(objectclass=inetorgperson))"
```

User `joe` is only granted access if he is found in the LDAP directory, has an object class of `inetorgperson`, and specifies a valid password.

2. Specifying a complete filter:

A complete filter must contain a `USERNAME` token which is substituted with the user name during the authentication attempt. The `rgw_ldap_dnattr` setting is not used in this case. For example, to limit valid users to a specific group, use the following filter:

Example

```
"(&(uid=@USERNAME@)(memberOf=cn=ceph-users,ou=groups,dc=mycompany,dc=com))"
```

Adding an S3 user to the LDAP server

Create at least one S3 user so that an S3 client can use the LDAP user credentials.

In the administrative console on the LDAP server, create at least one S3 user so that an S3 client can use the LDAP user credentials. Make a note of the user name and secret for use when passing the credentials to the S3 client.

Exporting an LDAP token

When running Ceph Object Gateway with LDAP, the access token is all that is required. The access token is created from the access key and secret key.

Export the access key and secret key as an LDAP token.

1. Export the access key:

   Syntax

   ```
   export RGW_ACCESS_KEY_ID="USERNAME"
   ```

2. Export the secret key:

   Syntax

   ```
   export RGW_SECRET_ACCESS_KEY="PASSWORD"
   ```

3. Export the token. For LDAP, use `ldap` as the token type (`ttype`).

   Example

   ```
   radosgw-token --encode --ttype=ldap
   ```

   For Active Directory, use `ad` as the token type.

   Example

   ```
   radosgw-token --encode --ttype=ad
   ```

   The result is a base-64 encoded string, which is the access token. Provide this access token to S3 clients in lieu of the access key. The secret key is no longer required.

4. Optional: For added convenience, export the base-64 encoded string to the `RGW_ACCESS_KEY_ID` environment variable if the S3 client uses the environment variable.

   Example

   ```
   export
   RGW_ACCESS_KEY_ID="ewogICAgIlJHV19UT0tFTiI6IHsKICAgICAgICAidmVyc2lvbiI6IDEsCiAgICAgICAgInR5cGUiOiAibGRhcCIsCiAgICAgICAgImlkIjogImNlcGgiLAogICAgICAgICJrZXkiOiAiODAwI0dvcmlsbGEiCiAgICB9Cn0K"
   ```

Testing the configuration with an S3 client

Test the configuration with a Ceph Object Gateway client, using a script such as Python Boto.

1. Use the `RGW_ACCESS_KEY_ID` environment variable to configure the Ceph Object Gateway client. Alternatively, you can copy the base-64 encoded string and specify it as the access key. Following is an example of the configured S3 client:

```
cat .aws/credentials

[default]
aws_access_key_id =
ewogICaGbnjlwe9UT0tFTiI6IHsKICAgICAgICAidmVyc2lvbiI6IDEsCiAgICAgICAgInR5cGUiOiAiYWQiLA
ogICAgICAgICJpZCI6ICJjZXBoIiwKICAgICAgICAia2V5IjogInBhc3M0Q2VwaCIKICAgIH0KfQo=
aws_secret_access_key =
```

> **Note:** The secret key is no longer required.

2. Run the **`aws s3 ls`** command to verify the user.

```
[root@host01 ~]# aws s3 ls --endpoint http://host03

2023-12-11 17:08:50 mybucket
2023-12-24 14:55:44 mybucket2
```

3. Optional: You can also run the **`radosgw-admin user`** command to verify the user in the directory.

```
[root@host01 ~]# radosgw-admin user info --uid dir1
{
    "user_id": "dir1",
    "display_name": "dir1",
    "email": "",
    "suspended": 0,
    "max_buckets": 1000,
    "subusers": [],
    "keys": [],
    "swift_keys": [],
    "caps": [],
    "op_mask": "read, write, delete",
    "default_placement": "",
    "default_storage_class": "",
    "placement_tags": [],
    "bucket_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
    "user_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
    "temp_url_keys": [],
    "type": "ldap",
    "mfa_ids": []
}
```

Active Directory integration

Configuring Active Directory (AD) for LDAPS is important to enable secure LDAP communication between the Ceph Object Gateway (RGW) and the LDAP server

To configure an Active Directory server to authenticate Ceph Object Gateway users, perform the following steps:

1. “Using Microsoft Active Directory” on page 196
2. “Configuring Active Directory for LDAPS” on page 196
3. “Adding a gateway user” on page 196
4. “Configuring the gateway to use Active Directory” on page 197
5. Add an S3 user to the LDAP server

6. Export an LDAP token
7. Test the configuration with an S3 client

Using Microsoft Active Directory

Ceph Object Gateway LDAP authentication is compatible with any LDAP-compliant directory service that can be configured for simple bind, including Microsoft Active Directory. By using Active Directory, the Ceph Object Gateway binds as the user configured in the `rgw_ldap_binddn` setting, and uses LDAPs to ensure security.

The process for configuring Active Directory is essentially identical to "Authenticating with Ceph Object Gateway" on page 189, but may have some Windows-specific usage.

Configuring Active Directory for LDAPS

Active Directory LDAP servers are configured to use LDAPs by default. Windows Server 2012 and higher can use Active Directory Certificate Services.

Instructions for generating and installing SSL certificates for use with Active Directory LDAP are available in the following MS TechNet article:LDAP over SSL (LDAPS) Certificate.

> **Note:** Ensure that port `636` is open on the Active Directory host.

Adding a gateway user

Create an LDAP user for the Ceph Object Gateway.

PREREQUISITE

Before creating the gateway user, ensure that the Ceph Object Gateway does not already have the user.

```
radosgw-admin metadata list user
```

For example,

```
[ceph: root@host01 /]# radosgw-admin metadata list user
```

Look in the output to be sure that the username is not in this list of users.

1. Create an LDAP user for the Ceph Object Gateway, and make note of the `binddn`.
   Since the Ceph Object Gateway uses the `ceph` user, consider using `ceph` as the username. The user needs to have permission to search the directory. The Ceph Object Gateway binds to this user as specified in `rgw_ldap_binddn`.
2. Test to verify that the user creation worked.
   You can run a search for the user, when the user ID (`uid`) is `ceph` for `People` with the `example.com` domain.

   ```
   # ldapsearch -x -D "uid=ceph,ou=People,dc=example,dc=com" -W -H ldaps://example.com -b
   "ou=People,dc=example,dc=com" -s sub 'uid=ceph'
   ```

3. On each gateway node, create a file for the user's secret.
   For example, the secret might get stored in a file entitled `/etc/bindpass`.
   For security, change the owner of this file to the `ceph` user and group to ensure that it is not globally readable.
4. Add the `rgw_ldap_secret` option.

   ```
   ceph config set client.rgw OPTION VALUE
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph config set client.rgw rgw_ldap_secret /etc/bindpass
   ```

5. Copy the updated configuration file to each Ceph node.

```
scp /etc/ceph/ceph.conf NODE:/etc/ceph
```

Configuring the gateway to use Active Directory

Configure the Ceph Object Gateway to use the Active Directory.

1. Change the Ceph configuration with the following commands on all the Ceph nodes.

   Syntax

   ```
   ceph config set client.rgw OPTION VALUE
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph config set client.rgw rgw_ldap_uri ldaps://FQDN:636
   [ceph: root@host01 /]# ceph config set client.rgw rgw_ldap_binddn "BINDDN"
   [ceph: root@host01 /]# ceph config set client.rgw rgw_ldap_searchdn "SEARCHDN"
   [ceph: root@host01 /]# ceph config set client.rgw rgw_ldap_dnattr "cn"
   [ceph: root@host01 /]# ceph config set client.rgw rgw_s3_auth_use_ldap true
   ```

   For the `rgw_ldap_uri` setting, substitute `FQDN` with the fully qualified domain name of the LDAP server. If there is more than one LDAP server, specify each domain.

   For the `rgw_ldap_binddn` setting, substitute `BINDDN` with the bind domain. With a domain of `example.com` and a `ceph` user under `users` and `accounts`, it should look something like this:

   Example

   ```
   rgw_ldap_binddn "uid=ceph,cn=users,cn=accounts,dc=example,dc=com"
   ```

   For the `rgw_ldap_searchdn` setting, substitute `SEARCHDN` with the search domain. With a domain of `example.com` and users under `users` and `accounts`, it should look something like this:

   ```
   rgw_ldap_searchdn "cn=users,cn=accounts,dc=example,dc=com"
   ```

2. Restart the Ceph Object Gateway:

   > **Note:** Use the output from the `ceph orch ps` command, under the `NAME` column, to get the `SERVICE_TYPE.ID` information.

   a. To restart the Ceph Object Gateway on an individual node in the storage cluster:

      Syntax

      ```
      systemctl restart ceph-CLUSTER_ID@SERVICE_TYPE.ID.service
      ```

      Example

      ```
      [root@host01 ~]# systemctl restart ceph-c4b34c6f-8365-11ba-
      dc31-529020a7702d@rgw.realm.zone.host01.gwasto.service
      ```

   b. To restart the Ceph Object Gateways on all nodes in the storage cluster:

      Syntax

      ```
      ceph orch restart SERVICE_TYPE
      ```

      Example

      ```
      [ceph: root@host01 /]# ceph orch restart rgw
      ```

Adding an S3 user to the LDAP server

Create at least one S3 user so that an S3 client can use the LDAP user credentials.

In the administrative console on the LDAP server, create at least one S3 user so that an S3 client can use the LDAP user credentials. Make a note of the user name and secret for use when passing the credentials to the S3 client.

Exporting an LDAP token

When running Ceph Object Gateway with LDAP, the access token is all that is required. The access token is created from the access key and secret key.

Export the access key and secret key as an LDAP token.

1. Export the access key:

   Syntax

   ```
   export RGW_ACCESS_KEY_ID="USERNAME"
   ```

2. Export the secret key:

   Syntax

   ```
   export RGW_SECRET_ACCESS_KEY="PASSWORD"
   ```

3. Export the token. For LDAP, use `ldap` as the token type (`ttype`).

   Example

   ```
   radosgw-token --encode --ttype=ldap
   ```

   For Active Directory, use `ad` as the token type.

   Example

   ```
   radosgw-token --encode --ttype=ad
   ```

   The result is a base-64 encoded string, which is the access token. Provide this access token to S3 clients in lieu of the access key. The secret key is no longer required.

4. Optional: For added convenience, export the base-64 encoded string to the `RGW_ACCESS_KEY_ID` environment variable if the S3 client uses the environment variable.

   Example

   ```
   export
   RGW_ACCESS_KEY_ID="ewogICAgIlJHV19UT0tFTiI6IHsKICAgICAgICAidmVyc2lvbiI6IDEsCiAgICAgICA
   gInR5cGUiOiAibGRhcCIsCiAgICAgICAgImlkIjogImNlcGgiLAogICAgICAgICJrZXkiOiAiODAwI0dvcmlsb
   GEiCiAgICB9Cn0K"
   ```

Testing the configuration with an S3 client

Test the configuration with a Ceph Object Gateway client, using a script such as Python Boto.

1. Use the `RGW_ACCESS_KEY_ID` environment variable to configure the Ceph Object Gateway client. Alternatively, you can copy the base-64 encoded string and specify it as the access key. Following is an example of the configured S3 client:

   ```
   cat .aws/credentials

   [default]
   aws_access_key_id =
   ewogICaGbnjlwe9UT0tFTiI6IHsKICAgICAgICAidmVyc2lvbiI6IDEsCiAgICAgICAgInR5cGUiOiAiYWQiLA
   ogICAgICAgICJpZCI6ICJjZXBoIiwKICAgICAgICAia2V5IjogInBhc3M0Q2VwaCIKICAgIH0KfQo=
   aws_secret_access_key =
   ```

> **Note:** The secret key is no longer required.

2. Run the **aws s3 ls** command to verify the user.

```
[root@host01 ~]# aws s3 ls --endpoint http://host03

2023-12-11 17:08:50 mybucket
2023-12-24 14:55:44 mybucket2
```

3. Optional: You can also run the **radosgw-admin user** command to verify the user in the directory.

```
[root@host01 ~]# radosgw-admin user info --uid dir1
{
    "user_id": "dir1",
    "display_name": "dir1",
    "email": "",
    "suspended": 0,
    "max_buckets": 1000,
    "subusers": [],
    "keys": [],
    "swift_keys": [],
    "caps": [],
    "op_mask": "read, write, delete",
    "default_placement": "",
    "default_storage_class": "",
    "placement_tags": [],
    "bucket_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
    "user_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
    "temp_url_keys": [],
    "type": "ldap",
    "mfa_ids": []
}
```

OpenStack Keystone integration

As a storage administrator, you can use OpenStack's Keystone authentication service to authenticate users through the Ceph Object Gateway. Before you can configure the Ceph Object Gateway, you need to configure Keystone first. This enables the Swift service, and points the Keystone service to the Ceph Object Gateway. Next, you need to configure the Ceph Object Gateway to accept authentication requests from the Keystone service.

When using the OpenStack Keystone authentication service, be sure to configure the Ceph Object Gateway in the following order:

1. Understand Keystone authentication roles and usage. See “Keystone authentication and the Ceph Object Gateway” on page 200.
2. “Creating the Swift service” on page 200.
3. “Setting the Ceph Object Gateway endpoints” on page 201.
4. “Verifying Openstack is using the Ceph Object Gateway endpoints” on page 202.
5. Configure the Ceph Object Gateway in one of the following ways:
   - “Configuring the Ceph Object Gateway to use Keystone SSL” on page 203
   - “Configuring the Ceph Object Gateway to use Keystone authentication” on page 203
6. “Restarting the Ceph Object Gateway daemon” on page 204.

Before you begin

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat OpenStack Platform environment.
- A running Red Hat Ceph Storage environment.
- A running Ceph Object Gateway environment.

Keystone authentication and the Ceph Object Gateway

Organizations that use OpenStack Keystone to authenticate users can integrate Keystone with the Ceph Object Gateway. The Ceph Object Gateway enables the gateway to accept a Keystone token, authenticate the user, and create a corresponding Ceph Object Gateway user. When Keystone validates a token, the gateway considers the user authenticated.

The following are the benefits of using Keystone authentication:

- Assigning `admin`, `member`, and `reader` roles to users with Keystone.
- Automatic user creation in the Ceph Object Gateway.
- Managing users with Keystone
- The Ceph Object Gateway queries Keystone periodically for a list of revoked tokens.

Roles for Keystone authentication

The OpenStack Keystone service provides three roles: `admin`, `member`, and `reader`. These roles are hierarchical. users with the `admin` role inherit the capabilities of the `member` role and users with the `member` role inherit the capabilities of the `reader` role.

> **Note:** The `member` role's read permissions only apply to objects of the project that it belongs to.

admin

The admin role is reserved for the highest level of authorization within a particular scope. This usually includes all the create, read, update, or delete operations on a resource or API.

member

The `member` role is not used directly by default. It provides flexibility during deployments and helps reduce responsibility for administrators.

For example, you can override a policy for a deployment by using the default `member` role and a simple policy override to allow system members to update services and endpoints. This provides a layer of authorization between `admin` and `reader` roles.

reader

The `reader` role is reserved for read-only operations regardless of the scope.

> **Important:** If you use a `reader` to access sensitive information such as image license keys, administrative image data, administrative volume metadata, application credentials, and secrets, you might unintentionally expose sensitive information. Hence, APIs that expose these resources should carefully consider the impact of the `reader` role and appropriately defer access to the `member` and `admin` roles.

Creating the Swift service

Before configuring the Ceph Object Gateway, configure Keystone so that the Swift service is enabled and pointing to the Ceph Object Gateway.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Access to the Ceph software repository.
- Root-level access to OpenStack controller node.

Procedure

- Create the Swift service:

```
[root@swift~]# openstack service create --name=swift --description="Swift Service"
object-store
```

  Creating the service will echo the service settings.

Example

Field	Value
description	Swift Service
enabled	True
id	37c4c0e79571404cb4644201a4a6e5ee
name	swift
type	object-store

Setting the Ceph Object Gateway endpoints

After creating the Swift service, point the service to a Ceph Object Gateway.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Access to the Ceph software repository.
- A running Swift service on a RHOSP 17.1 environment.

Procedure

- Create the OpenStack endpoints pointing to the Ceph Object Gateway:

  Syntax

```
openstack endpoint create --region REGION_NAME swift admin "URL"
openstack endpoint create --region REGION_NAME swift public "URL"
openstack endpoint create --region REGION_NAME swift internal "URL"
```

  Replace *REGION_NAME* with the name of the gateway's zone group name or region name. Replace *URL* with URLs appropriate for the Ceph Object Gateway.

  Example

```
[root@osp ~]# openstack endpoint create --region us-west swift admin "http://
radosgw.example.com:8080/swift/v1"
[root@osp ~]# openstack endpoint create --region us-west swift public "http://
radosgw.example.com:8080/swift/v1"
[root@osp ~]# openstack endpoint create --region us-west swift internal "http://
radosgw.example.com:8080/swift/v1"
```

Field	Value
adminurl	`http://radosgw.example.com:8080/swift/v1`
id	`e4249d2b60e44743a67b5e5b38c18dd3`
internalurl	`http://radosgw.example.com:8080/swift/v1`
publicurl	`http://radosgw.example.com:8080/swift/v1`
region	`us-west`
service_id	`37c4c0e79571404cb4644201a4a6e5ee`
service_name	`swift`
service_type	`object-store`

Setting the endpoints will output the service endpoint settings.

Verifying Openstack is using the Ceph Object Gateway endpoints

After creating the Swift service and setting the endpoints, show the endpoints to ensure that all settings are correct.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Access to the Ceph software repository.
- Set Ceph Object Gateway endpoints.

Procedure

1. List the endpoints under the Swift service:

```
[root@swift~]# openstack endpoint list --service=swift
```

2. Verify settings for the endpoints listed in the previous command:

```
[root@swift~] #openstack endpoint show ENDPOINT_ID
```

Showing the endpoints will echo the endpoints settings, and the service settings.

Example	
Field	**Value**
adminurl	http://radosgw.example.com:8080/swift/v1
enabled	True
id	e4249d2b60e44743a67b5e5b38c18dd3
internalurl	http://radosgw.example.com:8080/swift/v1
publicurl	http://radosgw.example.com:8080/swift/v1
region	us-west
service_id	37c4c0e79571404cb4644201a4a6e5ee

Field	Value
service_name	swift
service_type	object-store

Configuring the Ceph Object Gateway to use Keystone SSL

Converting the OpenSSL certificates that Keystone uses configures the Ceph Object Gateway to work with Keystone. When the Ceph Object Gateway interacts with OpenStack's Keystone authentication, Keystone will terminate with a self-signed SSL certificate.

Prerequisites

- A running, and healthy Red Hat Ceph Storage cluster.
- Access to the Ceph software repository.

Procedure

1. Convert the OpenSSL certificate to the `nss db` format:

   Example

   ```
   [root@osp ~]# mkdir /var/ceph/nss

   [root@osp ~]# openssl x509 -in /etc/keystone/ssl/certs/ca.pem -pubkey |
       certutil -d /var/ceph/nss -A -n ca -t "TCu,Cu,Tuw"

   [root@osp ~]# openssl x509 -in /etc/keystone/ssl/certs/signing_cert.pem -pubkey |
       certutil -A -d /var/ceph/nss -n signing_cert -t "P,P,P"
   ```

2. Install Keystone's SSL certificate in the node running the Ceph Object Gateway. Alternatively set the value of the configurable `rgw_keystone_verify_ssl` setting to `false`.

   Setting `rgw_keystone_verify_ssl` to `false` means that the gateway will not attempt to verify the certificate.

Configuring the Ceph Object Gateway to use Keystone authentication

Configure the Red Hat Ceph Storage to use OpenStack's Keystone authentication.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running, and healthy Red Hat Ceph Storage cluster.
- Access to the Ceph software repository.
- Have `admin` privileges to the production environment.

1. For each gateway instance, set `nss_db_path` to the path where the NSS database is stored.
   For example,

   ```
   [ceph: root@host01 /]# ceph config set client.rgw nss_db_path "/var/lib/ceph/radosgw/
   ceph-rgw.rgw01/nss"
   ```

2. Provide authentication credentials.
   It is possible to configure a Keystone service tenant, user, and password for keystone for the OpenStack Identity API, similar to the way system administrators tend to configure OpenStack services. Providing a username and password avoids providing the shared secret to the `rgw_keystone_admin_token` setting.

> **Important:** Disable authentication by admin token in production environments. The service tenant credentials should have `admin` privileges.

Set the following necessary configuration options:

```
ceph config set client.rgw rgw_keystone_verify_ssl TRUE/FALSE
ceph config set client.rgw rgw_s3_auth_use_keystone TRUE/FALSE
ceph config set client.rgw rgw_keystone_url KEYSTONE_URL:ADMIN_PORT
ceph config set client.rgw rgw_keystone_accepted_roles ACCEPTED_ROLES_
ceph config set client.rgw rgw_keystone_accepted_admin_roles ACCEPTED_ADMIN_ROLES
ceph config set client.rgw rgw_keystone_admin_domain default
ceph config set client.rgw rgw_keystone_admin_project SERVICE_NAME
ceph config set client.rgw rgw_keystone_admin_user KEYSTONE_TENANT_USER_NAME
ceph config set client.rgw rgw_keystone_admin_password KEYSTONE_TENANT_USER_PASSWORD
ceph config set client.rgw rgw_keystone_implicit_tenants KEYSTONE_IMPLICIT_TENANT_NAME
ceph config set client.rgw rgw_swift_versioning_enabled TRUE/FALSE
ceph config set client.rgw rgw_swift_enforce_content_length TRUE/FALSE
ceph config set client.rgw rgw_swift_account_in_url TRUE/FALSE
ceph config set client.rgw rgw_trust_forwarded_https TRUE/FALSE
ceph config set client.rgw rgw_max_attr_name_len MAXIMUM_LENGTH_OF_METADATA_NAMES
ceph config set client.rgw rgw_max_attrs_num_in_req MAXIMUM_NUMBER_OF_METADATA_ITEMS
ceph config set client.rgw rgw_max_attr_size MAXIMUM_LENGTH_OF_METADATA_VALUE
ceph config set client.rgw rgw_keystone_accepted_reader_roles SwiftSystemReader
```

For example,

```
[ceph: root@host01 /]# ceph config set client.rgw rgw_keystone_verify_ssl false
[ceph: root@host01 /]# ceph config set client.rgw rgw_s3_auth_use_keystone true
[ceph: root@host01 /]# ceph config set client.rgw rgw_keystone_url http://<public
Keystone endpoint>:5000/
[ceph: root@host01 /]# ceph config set client.rgw rgw_keystone_accepted_roles 'member,
Member, admin'
[ceph: root@host01 /]# ceph config set client.rgw rgw_keystone_accepted_admin_roles
'ResellerAdmin, swiftoperator'
[ceph: root@host01 /]# ceph config set client.rgw rgw_keystone_admin_domain default
[ceph: root@host01 /]# ceph config set client.rgw rgw_keystone_admin_project service
[ceph: root@host01 /]# ceph config set client.rgw rgw_keystone_admin_user swift
[ceph: root@host01 /]# ceph config set client.rgw rgw_keystone_admin_password password
[ceph: root@host01 /]# ceph config set client.rgw rgw_keystone_implicit_tenants true
[ceph: root@host01 /]# ceph config set client.rgw rgw_swift_versioning_enabled  true
[ceph: root@host01 /]# ceph config set client.rgw rgw_swift_enforce_content_length
true
[ceph: root@host01 /]# ceph config set client.rgw rgw_swift_account_in_url true
[ceph: root@host01 /]# ceph config set client.rgw rgw_trust_forwarded_https true
[ceph: root@host01 /]# ceph config set client.rgw rgw_max_attr_name_len 128
[ceph: root@host01 /]# ceph config set client.rgw rgw_max_attrs_num_in_req 90
[ceph: root@host01 /]# ceph config set client.rgw rgw_max_attr_size  1024
[ceph: root@host01 /]# ceph config set client.rgw rgw_keystone_accepted_reader_roles
SwiftSystemReader
```

A Ceph Object Gateway user is mapped into a Keystone `tenant`. A Keystone user has different roles assigned to it on possibly more than a single tenant. When the Ceph Object Gateway gets the ticket, it looks at the tenant, and the user roles that are assigned to that ticket, and accepts or rejects the request according to the `rgw_keystone_accepted_roles` configurable.

Related information

Users and Identity Management Guide

Restarting the Ceph Object Gateway daemon

Restarting the Ceph Object Gateway must be done to active configuration changes.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Access to the Ceph software repository.
- `admin` privileges to the production environment.

Procedure

- Once you have saved the Ceph configuration file and distributed it to each Ceph node, restart the Ceph Object Gateway instances:

  > **Note:** Use the output from the `ceph orch ps` command, under the `NAME` column, to get the *SERVICE_TYPE.ID* information.

  a. To restart the Ceph Object Gateway on an individual node in the storage cluster:

  Syntax

  ```
  systemctl restart ceph-CLUSTER_ID@SERVICE_TYPE.ID.service
  ```

  Example

  ```
  [root@host01 ~]# systemctl restart ceph-c4b34c6f-8365-11ba-
  dc31-529020a7702d@rgw.realm.zone.host01.gwasto.service
  ```

  b. To restart the Ceph Object Gateways on all nodes in the storage cluster:

  Syntax

  ```
  ceph orch restart SERVICE_TYPE
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph orch restart rgw
  ```

Secure Token Service

The Amazon Web Services Secure Token Service (STS) is a set of APIs that return a set of temporary set of S3 access and secret keys to authenticate users.

Red Hat Ceph Storage Object Gateway supports a subset of Amazon STS application programming interfaces (APIs) for identity and access management (IAM).

Users first authenticate against STS and receive a short-lived S3 access key and secret key that can be used in subsequent requests.

Red Hat Ceph Storage Object Storage can authenticate S3 users by integrating with a Single Sign-On by configuring an OIDC provider. This feature enables Object Storage users to authenticate against an enterprise identity provider rather than the local Ceph Object Gateway database. For instance, if the SSO is connected to an enterprise IDP in the backend, Object Storage users can use their enterprise credentials to authenticate and get access to the Ceph Object Gateway S3 endpoint.

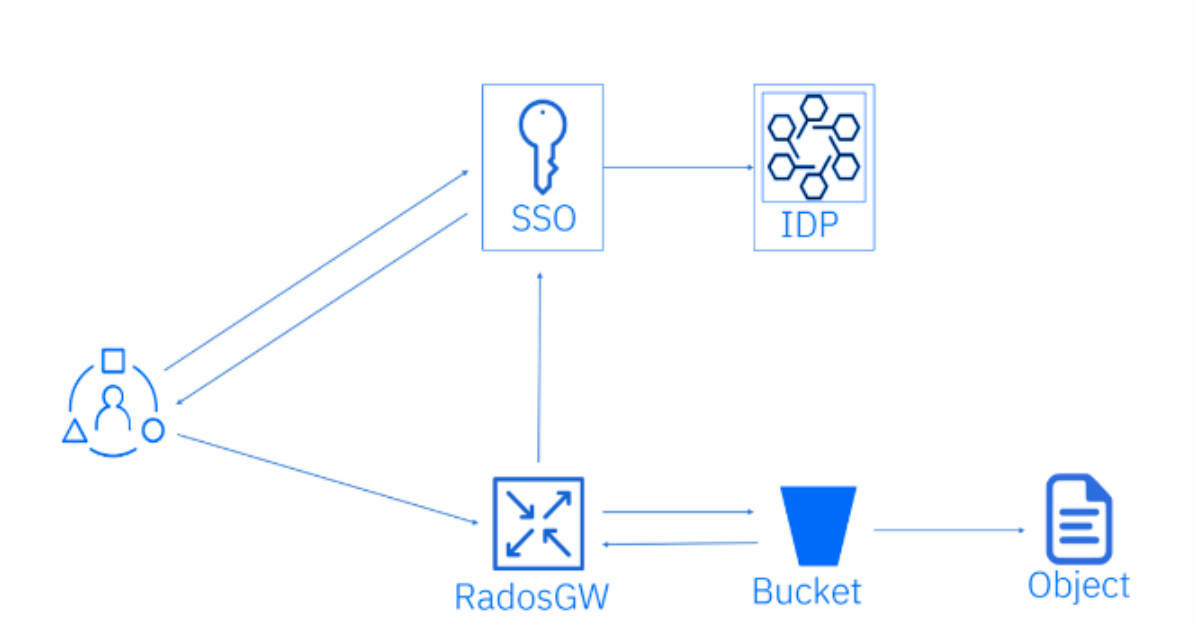
By using STS along with the IAM role policy feature, you can create finely tuned authorization policies to control access to your data. These policies allow you to implement either a Role-Based Access Control (RBAC) or Attribute-Based Access Control (ABAC) authorization model for your object storage data, giving you complete control over who can access the data.

Following is a simplified workflow to access S3 resources with STS:

1. The user wants access S3 resources in Red Hat Ceph Storage.
2. The user needs to authenticate against the SSO provider.
3. The SSO provider is federated with an IDP and checks if the user credentials are valid, the user gets authenticated and the SSO provides a Token to the user.
4. Using the Token provided by the SSO, the user accesses the Ceph Object Gateway STS endpoint, asking to assume an IAM role that provides the user with access to S3 resources.
5. The Red Hat Ceph Storage gateway receives the user token and asks the SSO to validate the token.

6. Once the SSO validates the token, the user is allowed to assume the role. Through STS, the user is provided with temporary access and secret keys that give the user access to the S3 resources.
7. Depending on the policies attached to the IAM role the user has assumed, the user can access a set of S3 resources.
8. For example, read for bucket A and write to bucket B.

Figure: Secure Token Service



- For more information about STS Lite and Keystone, see "Configuring and using STS Lite with Keystone (Technology Preview)" on page 216.
- For more information about STS Lite and Keystone limitations, see "Working around the limitations of using STS Lite with Keystone (Technology Preview)" on page 219.

Related information

Welcome to the AWS Security Token Service API Reference

Secure Token Service application programming interfaces

The Ceph Object Gateway implements the AssumeRole and AssumeRoleWithWebIdentity Secure Token Service (STS) application programming interfaces (APIs).

AssumeRole

This API returns a set of temporary credentials for cross-account access. These temporary credentials allow for both, permission policies attached with Role and policies attached with AssumeRole API. The `RoleArn` and the `RoleSessionName` request parameters are required, but the other request parameters are optional.

`RoleArn`

Description

The role to assume for the Amazon Resource Name (ARN) with a length of 20 to 2048 characters.

Type

String

Required

Yes

`RoleSessionName`

Description

Identifying the role session name to assume. The role session name can uniquely identify a session when different principals or different reasons assume a role. This parameter's value has a length of 2 to 64 characters. The `=`, `,`, `.`, `@`, and `-` characters are allowed, but no spaces allowed.

Type

String

Required

Yes

`Policy`

Description

An identity and access management policy (IAM) in a JSON format for use in an inline session. This parameter's value has a length of 1 to 2048 characters.

Type

String

Required

No

`DurationSeconds`

Description

The duration of the session in seconds, with a minimum value of `900` seconds to a maximum value of `43200` seconds. The default value is `3600` seconds.

Type

Integer

Required

No

`ExternalId`

Description

When assuming a role for another account, provide the unique external identifier if available. This parameter's value has a length of 2 to 1224 characters.

Type

String

Required

No

`SerialNumber`

Description

A user's identification number from their associated multi-factor authentication (MFA) device. The parameter's value can be the serial number of a hardware device or a virtual device, with a length of 9 to 256 characters.

Type

String

Required

No

`TokenCode`

Description

The value generated from the multi-factor authentication (MFA) device, if the trust policy requires MFA. If an MFA device is required, and if this parameter's value is empty or expired, then AssumeRole call returns an "access denied" error message. This parameter's value has a fixed length of 6 characters.

Type

String

Required

No

AssumeRoleWithWebIdentity

This API returns a set of temporary credentials for users who have been authenticated by an application, such as OpenID Connect or OAuth 2.0 Identity Provider. The `RoleArn` and the `RoleSessionName` request parameters are required, but the other request parameters are optional.

`RoleArn`

Description

The role to assume for the Amazon Resource Name (ARN) with a length of 20 to 2048 characters.

Type

String

Required

Yes

`RoleSessionName`

Description

Identifying the role session name to assume. The role session name can uniquely identify a session when different principals or different reasons assume a role. This parameter's value has a length of 2 to 64 characters. The `=`, `,`, `.`, `@`, and `-` characters are allowed, but no spaces are allowed.

Type

String

Required

Yes

`Policy`

Description

An identity and access management policy (IAM) in a JSON format for use in an inline session. This parameter's value has a length of 1 to 2048 characters.

Type

String

Required

No

`DurationSeconds`

Description

The duration of the session in seconds, with a minimum value of `900` seconds to a maximum value of `43200` seconds. The default value is `3600` seconds.

Type

Integer

Required

No

`ProviderId`

Description

The fully qualified host component of the domain name from the identity provider. This parameter's value is only valid for OAuth 2.0 access tokens, with a length of 4 to 2048 characters.

Type

String

Required

No

`WebIdentityToken`

Description

The OpenID Connect identity token or OAuth 2.0 access token provided from an identity provider. This parameter's value has a length of 4 to 2048 characters.

Type

String

Required

No

Reference

For more information, see Examples using the Secure Token Service APIs.

Related information

Amazon Web Services Security Token Service: AssumeRole action

Amazon Web Services Security Token Service: AssumeRoleWithWebIdentity action

Configuring the Secure Token Service

Configure the Secure Token Service (STS) for use with the Ceph Object Gateway.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- A running Ceph Object Gateway.
- Root-level access to a Ceph Manager node.

Configure the Secure Token Service (STS) for use with the Ceph Object Gateway by setting the `rgw_sts_key`, and `rgw_s3_auth_use_sts` options.

Important: Do not use the STS **`backward_compatibility`** configuration option in new Red Hat Ceph Storage 99 deployments. If the option is present in an existing configuration, remove it. This option is planned for deprecation along with tenant IAM/Roles. For more information, see Deprecated functionality .

Note: The S3 and STS APIs co-exist in the same namespace, and both can be accessed from the same endpoint in the Ceph Object Gateway.

1. Set the following configuration options for the Ceph Object Gateway client.

```
ceph config set RGW_CLIENT_NAME rgw_sts_key STS_KEY
ceph config set RGW_CLIENT_NAME rgw_s3_auth_use_sts true
```

The `rgw_sts_key` is the STS key for encrypting or decrypting the session token and is exactly 16 hex characters.

> **Important:** The STS key needs to be alphanumeric.

For example,

```
[root@mgr ~]# ceph config set client.rgw rgw_sts_key 7f8fd8dd4700mnop
[root@mgr ~]# ceph config set client.rgw rgw_s3_auth_use_sts true
```

2. Restart the Ceph Object Gateway for the added key to take effect.

> **Note:** Use the output from the **ceph orch ps** command, under the *NAME* column, to get the *SERVICE_TYPE.ID* information.

- Restart the Ceph Object Gateway on an individual node in the storage cluster.

  ```
  systemctl restart ceph-CLUSTER_ID@SERVICE_TYPE.ID.service
  ```

  For example,

  ```
  [root@host01 ~]# systemctl restart ceph-c4b34c6f-8365-11ba-
  dc31-529020a7702d@rgw.realm.zone.host01.gwasto.service
  ```

- Restart the Ceph Object Gateways on all nodes in the storage cluster.

  ```
  ceph orch restart SERVICE_TYPE
  ```

  For example,

  ```
  [ceph: root@host01 /]# ceph orch restart rgw
  ```

AFTER COMPLETING THE TASK

- For more information about STS APIs, see “Secure Token Service application programming interfaces” on page 206.
- For more information about using the Ceph configuration database, see “Configuration database” on page 0.

Creating a user for an OpenID Connect provider

To establish trust between the Ceph Object Gateway and the OpenID Connect Provider create a user entity and a role trust policy.

- For more information about STS APIs, see “Secure Token Service application programming interfaces” on page 206.
- For more information, see “Examples using the Secure Token Service APIs” on page 0.

Prerequisites

- User-level access to the Ceph Object Gateway node.
- Secure Token Service configured.

Procedure

1. Create a Ceph user.

   Syntax

```
radosgw-admin --uid USER_NAME --display-name "DISPLAY_NAME" --access_key USER_NAME --
secret SECRET user create
```

Example

```
[user@rgw ~]$ radosgw-admin --uid TESTER --display-name "TestUser" --access_key TESTER
--secret test123 user create
```

2. Configure the Ceph user capabilities.

   Syntax

   ```
   radosgw-admin caps add --uid="USER_NAME" --caps="oidc-provider=*"
   ```

   Example

   ```
   [user@rgw ~]$ radosgw-admin caps add --uid="TESTER" --caps="oidc-provider=*"
   ```

Related information

Obtaining the Root CA Thumbprint for an OpenID Connect Identity Provider

Obtaining a thumbprint of an OpenID Connect provider

Configure and get the OpenID Connect provider's (IDP) configuration document.

Any SSO that follows the OIDC protocol standards can be used but the following SSO providers have been tested:

- Red Hat Single Sign-on (RH-SSO) - An open-source identity and access management solution based on Keycloak, which provides enterprise-grade security and scalability for modern applications. For more information, see Red Hat Single Sign-On.
- Key cloak - A standalone tool for identity and access management, which allows users to create a user database with custom roles and groups. For more information, see Keycloak.
- IBM Security Verify - Provides end-to-end protection of resources over geographically dispersed intranets and extranet with deep, AI-powered context for both consumer and workforce identity and access management (IAM). A free 90-day trial is available with IBM Security Verify. Start your IBM Security Verify Free trial.
- For more information about STS APIs, see "Secure Token Service application programming interfaces" on page 206.
- For more information, see "Examples using the Secure Token Service APIs" on page 0.

Prerequisites

- Installation of the `openssl` and `curl` packages.
- Secure Token Service configured.
- User created for an OIDC provider.

Procedure

1. Get the configuration document from the IDP's URL:

   Syntax

   ```
   curl -k -v \
        -X GET \
        -H "Content-Type: application/x-www-form-urlencoded" \
        "IDP_URL:8000/CONTEXT/realms/REALM/.well-known/openid-configuration" \
      | jq .
   ```

   Example

```
[user@client ~]$ curl -k -v \
     -X GET \
     -H "Content-Type: application/x-www-form-urlencoded" \
     "http://www.example.com:8000/auth/realms/quickstart/.well-known/openid-
configuration" \
   | jq .
```

2. Get the IDP certificate:

   Syntax

```
curl -k -v \
     -X GET \
     -H "Content-Type: application/x-www-form-urlencoded" \
     "IDP_URL/CONTEXT/realms/REALM/protocol/openid-connect/certs" \
     | jq .
```

   Example

```
[user@client ~]$ curl -k -v \
     -X GET \
     -H "Content-Type: application/x-www-form-urlencoded" \
     "http://www.example.com/auth/realms/quickstart/protocol/openid-connect/certs" \
     | jq .
```

   Note: The `x5c` cert can be available on the `/certs` path or in the `/jwks` path depending on the SSO provider.

3. Copy the result of the `x5c` response from the previous command and paste it into the `certificate.crt` file. Include `––BEGIN CERTIFICATE––` at the beginning and `––END CERTIFICATE––` at the end.

```
-----BEGIN CERTIFICATE-----
MIIDYjCCAkqgAwIBAgIEEEd2CDANBgkqhkiG9w0BAQsFADBzMQkwBwYDVQQGEwAxCTAHBgNVBAgTADEJMAcGA1
UEBxMAMQkwBwYDVQQKEwAxCTAHBgNVBAsTADE6MDgGA1UEAxMxYXV0aHN2Yy1pbmxpbmVtZmEuZGV2LnZlcmlm
eS5pYm1jbG91ZHNlY3VyaXR5LmNvbTAeFw0yMTA3MDUxMzU2MzZaFw0zMTA3MDMxMzU2MzZaMHMxCTAHBgNVBA
YTADEJMAcGA1UECBMAMQkwBwYDVQQHEwAxCTAHBgNVBAoTADEJMAcGA1UECxMAMTowOAYDVQQDEzFhdXRoc3Zj
LWlubGluZW1mYS5kZXYudmVyaWZ5LmlibWNsb3Vkc2VjdXJpdHkuY29tMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ
8AMIIBCgKCAQEAphyu3HaAZ14JH/
EXetZxtNnerNuqcnfxcmLhBz9SsTlFD59ta+BOVlRnK5SdYEqO3ws2iGEzTvC55rczF+hDVHFZEBJLVLQe8ABm
i22RAtG1P0dA/
Bq8ReFxpOFVWJUBc31QM+ummW0T4yw44wQJI51LZTMz7PznB0ScpObxKe+frFKd1TCMXPlWOSzmTeFYKzR83Fg
9hsnz7Y8SKGxi+RoBbTLT+ektfWpR7O+oWZIf4INe1VYJRxZvn+qWcwI5uMRCtQkiMknc3Rj6Eupiqq6FlAjDs
0p//
EzsHAlW244jMYnHCGq0UP3oE7vViLJyiOmZw7J3rvs3m9mOQiPLoQIDAQABMA0GCSqGSIb3DQEBCwUAA4IBAQC
eVqAzSh7Tp8LgaTIFUuRbdjBAKXC9Nw3+pRBHoiUTdhqO3ualyGih9m/js/clb8Vq/
39zl0VPeaslWl2NNX9zaK7xo+ckVIOY3ucCaTC04ZUn1KzZu/7azlN0C5XSWg/
CfXgU2P3BeMNzc1UNY1BASGyWn2lEplIVWKLaDZpNdSyyGyaoQAIBdzxeNCyzDfPCa2oSO8WH1czmFiNPqR5kd
knHI96CmsQdi+DT4jwzVsYgrLfcHXmiWyIAb883hR3Pobp+Bsw7LUnxebQ5ewccjYmrJzOk5Wb5FpXBhaJH1B3
AEd6RGalRUyc/zUKdvEy0nIRMDS9x2BP3NVvZSADD
-----END CERTIFICATE-----
```

4. Get the certificate thumbprint.

   Syntax

```
openssl x509 -in CERT_FILE -fingerprint -noout
```

   Example

```
[user@client ~]$ openssl x509 -in certificate.crt -fingerprint -noout
SHA1 Fingerprint=F7:D7:B3:51:5D:D0:D3:19:DD:21:9A:43:A9:EA:72:7A:D6:06:52:87
```

   Note: If the IDP/SSO uses https instead of http with self-signed certificates, for the Ceph Object Gateway to connect to the SSO/OIDC, it needs to have the CA to trust the self-signed certificates. Bind the CA certs into the Ceph Object Gateway container:

Example

```
[root@host01 ~]# cat /tmp/rgw.yaml
---
service_type: rgw
service_id: objectgw
service_name: rgw.objectgw
placement:
count: 1
label: rgws
extra_container_args:
- -v
- /etc/pki/tls/certs:/etc/pki/tls/certs:z
- -v
- /etc/pki/ca-trust:/etc/pki/ca-trust:z
spec:
rgw_frontend_port: 8443
rgw_frontend_type: beast
rgw_realm: multisite
rgw_zone: zone1
ssl: true
```

5. Remove all the colons from the SHA1 fingerprint and use SHA1 as the input for creating the IDP entity in the IAM request.

Related information

Obtaining the Root CA Thumbprint for an OpenID Connect Identity Provider

Registering the OpenID Connect provider

Configure and get the OpenID Connect provider's (IDP) configuration document.

- For more information about STS APIs, see "Secure Token Service application programming interfaces" on page 206.
- For more information, see "Examples using the Secure Token Service APIs" on page 0.

Prerequisites

- Installation of the `openssl` and `curl` packages.
- Secure Token Service configured.
- User created for an OIDC provider.
- Thumbprint of an OIDC obtained.

Procedure

1. Extract URL from the token.

```
[root@host01 ~]# bash check_token_isv.sh | jq .iss

"https://keycloak-sso.apps.ocp.example.com/auth/realms/ceph"
```

2. Register the OIDC provider with Ceph Object Gateway.

```
[root@host01 ~]# aws --endpoint https://cephproxy1.example.com:8443 iam create-open-
id-connect-provider --url https://keycloak-sso.apps.ocp.example.com/auth/realms/ceph
--thumbprint-list 00E9CFD697E0B16DD13C86B0FFDC29957E5D24DF
```

3. Verify that the OIDC provider is added to the Ceph Object Gateway.

```
[root@host01 ~]# aws --endpoint https://cephproxy1.example.com:8443 iam
list-open-id-connect-providers

{
 "OpenIDConnectProviderList": [
 {
 "Arn":
"arn:aws:iam:::oidc-provider/keycloak-sso.apps.ocp.example.com/auth/realms/ceph"
 }
 ]
}
```

Related information

Obtaining the Root CA Thumbprint for an OpenID Connect Identity Provider

Creating IAM roles and policies

Learn how to create IAM roles and policies.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- Secure Token Service configured.
- User created for an OIDC provider.
- Thumbprint of an OIDC obtained.
- The OIDC provider in Ceph Object Gateway registered.

> **Note:** Configuring OIDC federation and IAM roles at the tenant level is deprecated.
> Tenant-based OIDC federation users should migrate their configurations to the new Ceph Object Gateway per-account model, before feature removal.
>
> For migration options, see “Identity and Access Management (IAM)” on page 155 .
>
> For more information, see Deprecated functionality .

1. Retrieve and validate JWT token.

```
[root@host01 ~]# curl -k -q -L -X POST
"https://keycloak-sso.apps.example.com/auth/realms/ceph/protocol/openid-connect/
token" \
-H 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'client_id=ceph' \
--data-urlencode 'grant_type=password' \
--data-urlencode 'client_secret=XXXXXXXXXXXXXXXXXXXXXXX' \
--data-urlencode 'scope=openid' \
--data-urlencode "username=SSOUSERNAME" \
--data-urlencode "password=SSOPASSWORD"
```

2. Verify the token.

```
[root@host01 ~]# cat check_token.sh
USERNAME=$1
PASSWORD=$2
KC_CLIENT="ceph"
KC_CLIENT_SECRET="7sQXqyMSzHIeMcSALoKaljB6sNIBDRjU"
KC_ACCESS_TOKEN="$(./get_web_token.sh $USERNAME $PASSWORD | jq -r '.access_token')"
KC_SERVER="https://keycloak-sso.apps.ocp.stg.local"
KC_CONTEXT="auth"
KC_REALM="ceph"
curl -k -s -q \
-X POST \
-u "$KC_CLIENT:$KC_CLIENT_SECRET" \
-d "token=$KC_ACCESS_TOKEN" \
```

```
"$KC_SERVER/$KC_CONTEXT/realms/$KC_REALM/protocol/openid-connect/token/introspect" |
jq .

[root@host01 ~]# ./check_token.sh s3admin passw0rd | jq .sub
"ceph"
```

In this example, the `jq` filter is used by the subfield in the token and is set to `ceph`.

3. Create a JSON file with role properties.
   Set **`Statement`** to `Allow` and the **`Action`** as `AssumeRoleWithWebIdentity`. Allow access to any user with the JWT token that matches the condition with `sub:ceph`.

```
[root@host01 ~]# cat role-rgwadmins.json
{
 "Version": "2012-10-17",
 "Statement": [
   {
     "Effect": "Allow",
     "Principal": {
       "Federated": [
         "arn:aws:iam:::oidc-provider/keycloak-sso.apps.example.com/auth/realms/ceph"
       ]
     },
     "Action": [
       "sts:AssumeRoleWithWebIdentity"
     ],
     "Condition": {
       "StringLike": { "keycloak-sso.apps.example.com/auth/realms/ceph:sub":"ceph" }
     }
   }
 ]
}
```

4. Create a Ceph Object Gateway role using the JSON file.

```
[root@host01 ~]# radosgw-admin role create --role-name rgwadmins \
--assume-role-policy-doc=$(jq -rc . /root/role-rgwadmins.json)
```

5. Once the role is successfully created, specify the S3 resources you need access to after assuming the `rgwadmins` role.

```
[root@host01 ~]# cat policy-rgwadmin.json
{
"Version": "2012-10-17",
"Statement": [
  {
   "Effect": "Allow",
   "Action": [ "s3:*" ],
   "Resource": "arn:aws:s3:::*"
  }
 ]
}
```

6. Apply the role policy for the `rgwadmins` role.

```
[root@host01 ~]# radosgw-admin role policy put --role-name=rgwadmins --policy-
name=admin --policy-doc=$(jq -rc . /root/policy-rgwadmin.json)
Permission policy attached successfully
```

Related information

Obtaining the Root CA Thumbprint for an OpenID Connect Identity Provider
Role management

Accessing S3 resources

Verify the AssumeRole with STS credentials to access S3 resources.

Prerequisites

- Secure Token Service configured.
- User created for an OIDC provider.
- Thumbprint of an OIDC obtained.
- The OIDC provider in Ceph Object Gateway registered.
- IAM roles and policies created.

Procedure

1. Following is an example of AssumeRole with STS to get temporary access and secret key to access S3 resources.

```
[roo@host01 ~]# cat test-assume-role.sh
#!/bin/bash
export AWS_CA_BUNDLE="/etc/pki/ca-trust/source/anchors/cert.pem"
unset AWS_ACCESS_KEY_ID
unset AWS_SECRET_ACCESS_KEY
unset AWS_SESSION_TOKEN
KC_ACCESS_TOKEN=$(curl -k -q -L -X POST
"https://keycloak-sso.apps.ocp.example.com/auth/realms/ceph/protocol/openid-connect/
token" \
-H 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'client_id=ceph' \
--data-urlencode 'grant_type=password' \
--data-urlencode 'client_secret=XXXXXXXXXXXXXXXXXXXXXXXX' \
--data-urlencode 'scope=openid' \
--data-urlencode "<varname>SSOUSERNAME</varname>" \
--data-urlencode "<varname>SSOPASSWORD</varname>" | jq -r .access_token)
echo ${KC_ACCESS_TOKEN}
IDM_ASSUME_ROLE_CREDS=$(aws sts assume-role-with-web-identity --role-arn
"arn:aws:iam:::role/$3" --role-session-name testbr
--endpoint=https://cephproxy1.example.com:8443
--web-identity-token="$KC_ACCESS_TOKEN")
echo "aws sts assume-role-with-web-identity --role-arn "arn:aws:iam:::role/$3"
--role-session-name testb --endpoint=https://cephproxy1.example.com:8443
--web-identity-token="$KC_ACCESS_TOKEN""
echo $IDM_ASSUME_ROLE_CREDS
export AWS_ACCESS_KEY_ID=$(echo $IDM_ASSUME_ROLE_CREDS | jq -r
.Credentials.AccessKeyId)
export AWS_SECRET_ACCESS_KEY=$(echo $IDM_ASSUME_ROLE_CREDS | jq -r
.Credentials.SecretAccessKey)
export AWS_SESSION_TOKEN=$(echo $IDM_ASSUME_ROLE_CREDS | jq -r
.Credentials.SessionToken)
```

2. Run the script.

```
[root@host01 ~]# source ./test-assume-role.sh s3admin SSOPASSWORD rgwadmins
[root@host01 ~]# aws s3 mb s3://testbucket
[root@host01 ~]# aws s3 ls
```

For more information, see “Examples using the Secure Token Service APIs” on page 0.

Related information

Obtaining the Root CA Thumbprint for an OpenID Connect Identity Provider

Configuring and using STS Lite with Keystone (Technology Preview)

The Amazon Secure Token Service (STS) and S3 APIs co-exist in the same namespace. The STS options can be configured in conjunction with the Keystone options.

Important: Technology Preview features are not supported with production service level agreements (SLAs), might not be functionally complete, and does not recommend using them for production. These

features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

Note: Both S3 and STS APIs can be accessed using the same endpoint in Ceph Object Gateway.

- For more information about installing the Boto Python module, see Test S3 Access.
- For more information, see Create a User.

Prerequisites

- Red Hat Ceph Storage 5.0 or later.
- A running Ceph Object Gateway.
- Installation of the Boto Python module, version 3 or higher.
- Root-level access to a Ceph Manager node.
- User-level access to an OpenStack node.

Procedure

1. Set the following configuration options for the Ceph Object Gateway client:

   Syntax

   ```
   ceph config set RGW_CLIENT_NAME rgw_sts_key STS_KEY
   ceph config set RGW_CLIENT_NAME rgw_s3_auth_use_sts true
   ```

   The `rgw_sts_key` is the STS key for encrypting or decrypting the session token and is exactly 16 hex characters.

   Important: The STS key needs to be alphanumeric.

   Example

   ```
   [root@mgr ~]# ceph config set client.rgw rgw_sts_key 7f8fd8dd4700mnop
   [root@mgr ~]# ceph config set client.rgw rgw_s3_auth_use_sts true
   ```

2. Generate the EC2 credentials on the OpenStack node:

   Example

   ```
   [user@osp ~]$ openstack ec2 credentials create

   +------------+---------------------------------------------------------+
   | Field      | Value                                                   |
   +------------+---------------------------------------------------------+
   | access     | b924dfc87d454d15896691182fdeb0ef                        |
   | links      | {u'self': u'http://192.168.0.15/identity/v3/users/      |
   |            | 40a7140e424f493d8165abc652dc731c/credentials/           |
   |            | OS-EC2/b924dfc87d454d15896691182fdeb0ef'}               |
   | project_id | c703801dccaf4a0aaa39bec8c481e25a                        |
   | secret     | 6a2142613c504c42a94ba2b82147dc28                        |
   | trust_id   | None                                                    |
   | user_id    | 40a7140e424f493d8165abc652dc731c                        |
   +------------+---------------------------------------------------------+
   ```

3. Use the generated credentials to get back a set of temporary security credentials using *GetSessionToken* API:

   Example

```
import boto3

access_key = b924dfc87d454d15896691182fdeb0ef
secret_key = 6a2142613c504c42a94ba2b82147dc28

client = boto3.client('sts',
aws_access_key_id=access_key,
aws_secret_access_key=secret_key,
endpoint_url=https://www.example.com/rgw,
region_name='',
)

response = client.get_session_token(
    DurationSeconds=43200
)
```

4. Obtaining the temporary credentials can be used for making S3 calls:

   Example

```
s3client = boto3.client('s3',
  aws_access_key_id = response['Credentials']['AccessKeyId'],
  aws_secret_access_key = response['Credentials']['SecretAccessKey'],
  aws_session_token = response['Credentials']['SessionToken'],
  endpoint_url=https://www.example.com/s3,
  region_name='')

bucket = s3client.create_bucket(Bucket='my-new-shiny-bucket')
response = s3client.list_buckets()
for bucket in response["Buckets"]:
  print "{name}\t{created}".format(
    name = bucket['Name'],
    created = bucket['CreationDate'],
  )
```

5. Create a new *S3Access* role and configure a policy.

   a. Assign a user with administrative CAPS:

      Syntax

```
radosgw-admin caps add --uid="USER" --caps="roles=*"
```

      Example

```
[root@mgr ~]# radosgw-admin caps add --uid="gwadmin" --caps="roles=*"
```

   b. Create the *S3Access* role:

      Syntax

```
radosgw-admin role create --role-name=ROLE_NAME --path=PATH --assume-role-policy-
doc=TRUST_POLICY_DOC
```

      Example

```
[root@mgr ~]# radosgw-admin role create --role-name=S3Access --path=/
application_abc/component_xyz/ --assume-role-policy-doc=\{\"Version\":
\"2012-10-17\",\"Statement\":\[\{\"Effect\":\"Allow\",\"Principal\":\{\"AWS\":\
[\"arn:aws:iam:::user/TESTER\"\]\},\"Action\":\[\"sts:AssumeRole\"\]\}\]\}
```

   c. Attach a permission policy to the *S3Access* role:

      Syntax

```
radosgw-admin role-policy put --role-name=ROLE_NAME --policy-name=POLICY_NAME --
policy-doc=PERMISSION_POLICY_DOC
```

      Example

```
[root@mgr ~]# radosgw-admin role-policy put --role-name=S3Access --policy-
name=Policy --policy-doc=\{\"Version\":\"2012-10-17\",\"Statement\":\[\
{\"Effect\":\"Allow\",\"Action\":\[\"s3:*\"\],\"Resource\":
\"arn:aws:s3:::example_bucket\"\}\]\}
```

d. Now another user can assume the role of the `gwadmin` user. For example, the `gwuser` user can assume the permissions of the `gwadmin` user.

e. Make a note of the assuming user's `access_key` and `secret_key` values.

Example

```
[root@mgr ~]# radosgw-admin user info --uid=gwuser | grep -A1 access_key
```

6. Use the *AssumeRole* API call, providing the `access_key` and `secret_key` values from the assuming user:

Example

```
import boto3

access_key = 11BS02LGFB6AL6H1ADMW
secret_key = vzCEkuryfn060dfee4fgQPqFrncKEIkh3ZcdOANY

client = boto3.client('sts',
aws_access_key_id=access_key,
aws_secret_access_key=secret_key,
endpoint_url=https://www.example.com/rgw,
region_name='',
)

response = client.assume_role(
RoleArn='arn:aws:iam:::role/application_abc/component_xyz/S3Access',
RoleSessionName='Bob',
DurationSeconds=3600
)
```

> **Important:** The *AssumeRole* API requires the *S3Access* role.

Working around the limitations of using STS Lite with Keystone (Technology Preview)

Working with Keystone has some limitations. Keystone does not support Secure Token Service (STS) requests and the payload hash is not included with the request. To work around these limitations the Boto authentication code must be modified.

> **Important:** Technology Preview features are not supported with production service level agreements (SLAs), might not be functionally complete, and does not recommend using them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

Prerequisites

- A running Red Hat Ceph Storage cluster, version 5.0 or later.
- A running Ceph Object Gateway.
- Installation of Boto Python module, version 3 or later.

For more information about installing the Boto Python module, see Test S3 Access.

Procedure

Open and edit Boto's `auth.py` file.

1. Add the following four lines to the code block:

```
f service_name == 'sts':
          self._service_name = 's3'
      else:
          self._service_name = service_name
```

   For example:

```
class SigV4Auth(BaseSigner):
  """
  Sign a request with Signature V4.
  """
  REQUIRES_REGION = True

  def __init__(self, credentials, service_name, region_name):
      self.credentials = credentials
      # We initialize these value here so the unit tests can have
      # valid values.  But these will get overriden in ``add_auth``
      # later for real requests.
      self._region_name = region_name
      if service_name == 'sts':
          self._service_name = 's3'
      else:
          self._service_name = service_name
```

2. Add the following two lines to the code block:

```
request.headers['X-Amz-Content-SHA256'] = UNSIGNED_PAYLOAD
        else:
```

   For example:

```
def _modify_request_before_signing(self, request):
        if 'Authorization' in request.headers:
            del request.headers['Authorization']
        self._set_necessary_date_headers(request)
        if self.credentials.token:
            if 'X-Amz-Security-Token' in request.headers:
                del request.headers['X-Amz-Security-Token']
            request.headers['X-Amz-Security-Token'] = self.credentials.token

        if not request.context.get('payload_signing_enabled', True):
            if 'X-Amz-Content-SHA256' in request.headers:
                del request.headers['X-Amz-Content-SHA256']
            request.headers['X-Amz-Content-SHA256'] = UNSIGNED_PAYLOAD
        else:
            request.headers['X-Amz-Content-SHA256'] = self.payload(request)
```

Security

As a storage administrator, securing the storage cluster environment is important. Red Hat Ceph Storage provides encryption and key management to secure the Ceph Object Gateway access point.

Prerequisites

- A healthy running Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway software.

QAT acceleration for encryption and compression

Intel QuickAssist Technology (QAT) can provide extended accelerated encryption and compression services by offloading the actual encryption and compression requests to the hardware QuickAssist accelerators, which are

more efficient in terms of cost and power than general purpose CPUs for those specific compute-intensive workloads.

PREREQUISITE

- A running Red Hat Ceph Storage cluster.
- Ceph Object gateway installed.
- *Grub* is configured to pass the **`intel_iommu`** parameter.

```
grubby --update-kernel=ALL --args="intel_iommu=on"
```

Important: Hardware accelerated compression in Ceph Object Gateway requires RHEL 9.4 on a Sapphire or Emerald Rapids Xeon CPU (or newer) with QAT devices. For more information, see Intel Ark.

Setting up the QAT service

You can set up the QAT service to encrypt and compress the Ceph Object Gateway objects.

1. Install `qatlib-service`, `qatlib`, `qatzip`, and `qatengine` packages.

```
dnf install -y qatlib-service qatlib qatzip qatengine
```

2. Add `root` to the `QAT` group on the host.

```
usermod -aG qat root
```

3. Check that the `limits.conf` file exists with one of the following **`ServicesEnabled`** parameters in the configuration file.
   - For using data encryption, check that **`ServicesEnabled`** is set to `asym`.

```
cat /etc/sysconfig/qat

ServicesEnabled=sym
POLICY=8
```

   - For using data compression, check that **`ServicesEnabled`** is set to `dc`.

```
cat /etc/sysconfig/qat

ServicesEnabled=dc
POLICY=8
```

   - For using both data encryption and compression, check that **`ServicesEnabled`** is set to `asym,dc`.

```
cat /etc/sysconfig/qat

ServicesEnabled=sym,dc
POLICY=8
```

4. Configure the `limits.conf` file with `memlock` information.

```
sudo vim /etc/security/limits.conf

...
root - memlock 500000
ceph - memlock 500000
...
```

5. Enable the configurations in the `limits.conf` file.

```
sudo su -l $USER
```

6. Enable the QAT service.

```
systemctl enable qat
```

7. Restart the node.

```
systemctl reboot
```

8. Create the specification file and provide additional arguments to Podman for Ceph Object Gateway.

> **Note:** You can use the following command to generate the device list:
>
> ```
> --device /dev/vfio --device /dev/qat_adf_ctl $(for i in `ls /dev/vfio/* | grep
> 'dev' | grep -v ':'` ; do echo --device $i;
> ```

For example,

```
[root@host01 ~]# cat rgw-test-qat.yaml

service_type: rgw
service_id: rgw_qat_auto
placement:
  label: rgw
spec:
  rgw_frontend_port: 8000
  rgw_realm: REALM
  rgw_zone: ZONE
  rgw_zonegroup: ZONE_GROUP
  qat:
    compression: sw | hw
  ssl: true
  generate_cert: true
```

`cephadm` automatically sets **`qat_compressor_enabled`** to `true` when `qat:compression` is configured.

AFTER COMPLETING THE TASK

When using hardware compression (`compression: hw`), verify that the QAT compressor is enabled in the `/etc/sysconfig/qat` file.
For example,

```
[root@host01 sysconfig]# cat qat

ServicesEnabled=dc

POLICY=8

QAT_USER=ceph
```

Enabling QAT-based encryption

Encrypt objects in Ceph Object Gateway by using the QAT-based encryption for OpenSSL.

- To enable QAT-based encryption, edit the Ceph configuration file to make use of QAT-based crypto plug-in.

```
plugin crypto accelerator = crypto_qat
```

Enabling QAT-based compression

Compress objects in Ceph Object Gateway by using the tool class for QAT acceleration.

When using Red Hat Ceph Storage 9.0z1 or later, compression is enabled when creating the QATspecification file. Compression is enabled when creating the QAT specification file. For more information, see “Setting up the QAT service” on page 221.

For Red Hat Ceph Storage 9.0, QAT-based compression has to be enabled manually by using this procedure.

If needed, use this procedure to enable compression manually.

- To enable QAT-based compression, edit the Ceph configuration file to enable QAT support for compression.

```
qat compressor enabled=true
```

Server side encryption

The Ceph Object Gateway supports server-side encryption of uploaded objects for the S3 application programming interface (API). Server-side encryption means that the S3 client sends data over HTTP in its unencrypted form, and the Ceph Object Gateway stores that data in the Red Hat Ceph Storage cluster in encrypted form.

Important: To use encryption, client requests MUST send requests over an SSL connection. Red Hat does not support S3 encryption from a client unless the Ceph Object Gateway uses SSL. However, for testing purposes, administrators can disable SSL during testing by setting the `rgw_crypt_require_ssl` configuration setting to `false` at runtime, using the **`ceph config set client.rgw`** command, and then restarting the Ceph Object Gateway instance. In a production environment, it might not be possible to send encrypted requests over SSL. In such a case, send requests by using HTTP with server-side encryption.

Note:

- Red Hat does not support S3 object encryption of Static Large Object (SLO) or Dynamic Large Object (DLO).
- Currently, none of the Server-Side Encryption (SSE) modes have implemented support for CopyObject.

The following options are supported for the management of encryption keys.

Customer-provided keys

When using customer-provided keys, the S3 client passes an encryption key along with each request to read or write encrypted data. It is the customer's responsibility to manage those keys. Customers must remember which key the Ceph Object Gateway used to encrypt each object.

Ceph Object Gateway implements the customer-provided key behavior in the S3 API according to the Amazon SSE-C specification.

Since the customer handles the key management and the S3 client passes keys to the Ceph Object Gateway, the Ceph Object Gateway requires no special configuration to support this encryption mode.

Key management service

When using a key management service, the secure key management service stores the keys and the Ceph Object Gateway retrieves them on demand to serve requests to encrypt or decrypt data.

Ceph Object Gateway implements the key management service behavior in the S3 API according to the Amazon SSE-KMS specification.

Important: Currently, the only tested key management implementations are HashiCorp Vault, OpenStack Barbican, and IBM Guardium Key Lifecycle Manager (GKLM). OpenStack Barbican is a Technology Preview and is not supported for use in production systems.

SSE-S3

When using SSE-S3, the keys are stored in a vault, but they are automatically created and deleted by Ceph and retrieved as required to serve requests to encrypt or decrypt data.

Ceph Object Gateway implements the SSE-S3 behavior in the S3 API according to the Amazon SSE-S3 specification.

For more information, see the following:

- "Configuring server-side encryption" on page 227
- "Using HashiCorp Vault" on page 229
- "Using IBM GKLM" on page 239

Related information

Amazon SSE-C
Amazon SSE-KMS

Encryption at rest and in transit

Understand how Red Hat Ceph Storage Object Gateway uses encryption at rest and in transit to protect sensitive data.

Data security is a critical priority in today's digital landscape. Red Hat Ceph Storage Object Gateway provides options for implementing encryption at rest and in transit, helping safeguard data whether stored or transferred. These mechanisms often rely on key management solutions such as IBM Guardium Key Lifecycle Manager (GKLM) to manage and store encryption keys securely. This topic explains how these encryption methods work and the benefits of using them together.

Encryption at rest with Ceph Object Gateway

Ceph Object Gateway offers robust encryption for data that is stored on disk, supporting granular object-level encryption. It is fully compatible with S3 encryption semantics and integrates with key management systems such as IBM Guardium Key Lifecycle Manager, Hashicorp Vault, and Thales CipherTrust.

Encryption at rest is essential for safeguarding stored data from unauthorized access by encrypting it while it resides on disk. Ceph facilitates this process through seamless integration with external Key Management Systems (KMS) like IBM Guardium Key Lifecycle Manager. By doing so, Ceph can automatically retrieve encryption keys from the key manager, helping ensure that objects are encrypted as they are uploaded. This integration provides data security throughout its lifecycle while centralizing key management, simplifying key rotation, auditing, and overall compliance efforts.

Encryption in transit with Ceph Object Gateway

Data can travel through various unknown, untrusted, or both unknown and untrusted networks as it moves between the object storage and the application. Encrypting the data as it travels over the network can effectively prevent tampering and helps ensure the security of the data, regardless of its path.

In Ceph Object, encryption in transit is typically achieved by using TLS/SSL encryption, which secures communication between clients and the Ceph Object Gateway. After the Object Gateways are configured, they provide an S3 RESTful API over HTTPS. The RESTful API encrypts and protects all data that is exchanged between clients and the object storage system.

From an architectural standpoint, SSL encryption can be applied at multiple layers: at the load balancer (ingress service), Ceph Object Gateway level, or both. For full client-to-Ceph Object Gateway HTTPS encryption, the two approaches for load balancer are *pass-through* and *reencrypt*.

Pass-through

Encrypted traffic is forwarded at level 3 directly to the Object Gateway service.

Reencrypt

The load balancer decrypts and reencrypts traffic before sending it to the Ceph Object Gateway.

Server-side encryption (SSE) options with Ceph Object Gateway

Ceph Object Gateway supports various server-side encryption (SSE) methods to secure data stored in its object storage. These methods allow Ceph to automatically encrypt data as it is written, helping ensure that data at rest is protected without requiring the user to manage encryption independently. See the following for the supported SSE methods and their differences.

SSE-C (server-side encryption with customer-provided keys)

SSE-C allows clients to manage their own encryption keys. The client provides Ceph with a key for each object request in this mode. Ceph then uses this key to encrypt the data before storing it and must decrypt it using the same key when data is requested.
This method suits users who require complete control over their encryption keys but are willing to manage them securely.

SSE-S3 (server-side encryption with S3 managed keys)

Red Hat Ceph Storage manages encryption and decryption in this configuration. The encryption and decryption are done by using external KMS to handle keys securely. SSE-S3 provides transparent encryption, automatically encrypting data when it is stored and decrypted when retrieved without requiring extra effort from the client. SSE-S3 is only supported with the HashiCorp Vault Integration Program.

SSE-KMS (server-side encryption with Key Management Service)

SSE-KMS provides server-side encryption where an external Key Management Service (KMS) keys, such as IBM Guardium Key Lifecycle Manager, manage keys. When using this method, Ceph communicates with a KMS to retrieve the encryption keys for encrypting and decrypting data. The keys are stored and managed externally, which can simplify compliance with encryption standards.

"SSE options advantages and disadvantages" on page 225 compares and contrasts the different options.

SSE options advantages and disadvantages

	SSE-C	SSE-KMS	SSE-S3
Key management	Handled by the client. The key is provided on every read-and-write request.	Handled by the KMS. For example, IBM GKLM.	SSE-S3.
Advantages	Full control over the keys remains with the user, allowing for more custom security policies.	Centralized and secure key management. The KMS takes care of the entire lifecycle of the encryption keys, including auditing, rotation, and deletion.	Provides transparent encryption, automatically encrypting data when it is stored and decrypted when retrieved without requiring extra effort from the client
Disadvantages	The user bears the burden of key management. Losing the key results in the permanent loss of data.	Requires integration with an external KMS, which can add complexity to the setup.	SSE-S3 is only supported by the Hashicorp Vault Integration Program.

Setting the default encryption for an existing S3 bucket

As a storage administrator, you can set the default encryption for an existing Amazon S3 bucket so that all objects are encrypted when they are stored in a bucket. You can use Bucket Encryption APIs to support server-side encryption with Amazon S3-managed keys (SSE-S3) or Amazon KMS customer master keys (SSE-KMS).

You can manage default encryption for an existing Amazon S3 bucket using the `PutBucketEncryption` API. All files uploaded to this bucket will have this encryption by defining the default encryption at the bucket level.

Prerequisites

- A running Red Hat Ceph Storage
- Installation of the Ceph Object Gateway.
- An S3 bucket created.
- An S3 user created with user access.
- Access to a Ceph Object Gateway client with the AWS CLI package installed.

Procedure

1. Create a JSON file for the encryption configuration:

Example

```
[user@client ~]$ vi bucket-encryption.json
```

2. Add the encryption configuration rules to the file:

   Example

```
{
        "Rules": [
        {
            "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
            }
      }
    ]
}
```

3. Set the default encryption for the bucket:

   Syntax

```
aws --endpoint-url=RADOSGW_ENDPOINT_URL:PORT s3api put-bucket-encryption --bucket
BUCKET_NAME --server-side-encryption-configuration file://
PATH_TO_BUCKET_ENCRYPTION_CONFIGURATION_FILE/BUCKET_ENCRYPTION_CONFIGURATION_FILE.json
```

   Example

```
[user@client ~]$ aws --endpoint-url=http://host01:80 s3api put-bucket-encryption --
bucket testbucket --server-side-encryption-configuration file://bucket-encryption.json
```

Verification

- Retrieve the bucket encryption configuration for the bucket:

  Syntax

```
aws --endpoint-url=RADOSGW_ENDPOINT_URL:PORT s3api get-bucket-encryption --bucket
BUCKET_NAME
```

  Example

```
[user@client ~]$  aws --profile ceph --endpoint=http://host01:80 s3api get-bucket-
encryption --bucket testbucket

{
    "ServerSideEncryptionConfiguration": {
        "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
                }
            }
        ]
    }
}
```

Note: If the bucket does not have a default encryption configuration, the `get-bucket-encryption` command returns `ServerSideEncryptionConfigurationNotFoundError`.

Deleting the default bucket encryption

You can delete the default bucket encryption for a specified bucket using the **`s3api delete-bucket-encryption`** command.

Prerequisites

- A running Red Hat Ceph Storage
- Installation of the Ceph Object Gateway.
- An S3 bucket created.
- An S3 user created with user access.
- Access to a Ceph Object Gateway client with the AWS CLI package installed.

Procedure

- Delete a bucket encryption configuration:

  Syntax

```
aws --endpoint-url=RADOSGW_ENDPOINT_URL:PORT s3api delete-bucket-encryption --bucket
BUCKET_NAME
```

  Example

```
[user@client ~]$ aws --endpoint-url=http://host01:80 s3api delete-bucket-encryption --
bucket testbucket
```

Verification

- Retrieve the bucket encryption configuration for the bucket:

  Syntax

```
aws --endpoint-url=RADOSGW_ENDPOINT_URL:PORT s3api get-bucket-encryption --bucket
BUCKET_NAME
```

  Example

```
[user@client ~]$  aws --endpoint=http://host01:80 s3api get-bucket-encryption --bucket
testbucket

An error occurred (ServerSideEncryptionConfigurationNotFoundError) when calling the
GetBucketEncryption operation:
The server side encryption configuration was not found
```

Server-side encryption requests

In a production environment, clients often contact the Ceph Object Gateway through a proxy. This proxy is referred to as a load balancer because it connects to multiple Ceph Object Gateways. When the client sends requests to the Ceph Object Gateway, the load balancer routes those requests to the multiple Ceph Object Gateways, thus distributing the workload.

In this type of configuration, it is possible that SSL terminations occur both at a load balancer and between the load balancer and the multiple Ceph Object Gateways. Communication occurs using HTTP only.

Configuring server-side encryption

You can set up server-side encryption to send requests to the Ceph Object Gateway using HTTP, in cases where it might not be possible to send encrypted requests over SSL.

This procedure uses HAProxy as proxy and load balancer.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to all nodes in the storage cluster.

- Installation of the Ceph Object Gateway software.
- Installation of the HAProxy software.

Procedure

1. Edit the `haproxy.cfg` file:

   Example

   ```
   frontend http_web *:80
       mode http
       default_backend rgw

   frontend rgw-https
     bind *:443 ssl crt /etc/ssl/private/example.com.pem
     default_backend rgw

   backend rgw
       balance roundrobin
       mode http
       server  rgw1 10.0.0.71:8080 check
       server  rgw2 10.0.0.80:8080 check
   ```

2. Comment out the lines that allow access to the `http` front end and add instructions to direct HAProxy to use the `https` front end instead:

   Example

   ```
   #     frontend http_web *:80
   #     mode http
   #     default_backend rgw

   frontend rgw-https
     bind *:443 ssl crt /etc/ssl/private/example.com.pem
     http-request set-header X-Forwarded-Proto https if { ssl_fc }
     http-request set-header X-Forwarded-Proto https
   # here we set the incoming HTTPS port on the load balancer (eg : 443)
     http-request set-header X-Forwarded-Port 443
     default_backend rgw

   backend rgw
       balance roundrobin
       mode http
       server  rgw1 10.0.0.71:8080 check
       server  rgw2 10.0.0.80:8080 check
   ```

3. Set the `rgw_trust_forwarded_https` option to `true`:

   Example

   ```
   [ceph: root@host01 /]# ceph config set client.rgw rgw_trust_forwarded_https true
   ```

4. Enable and start HAProxy:

   ```
   [root@host01 ~]# systemctl enable haproxy
   [root@host01 ~]# systemctl start haproxy
   ```

Reference

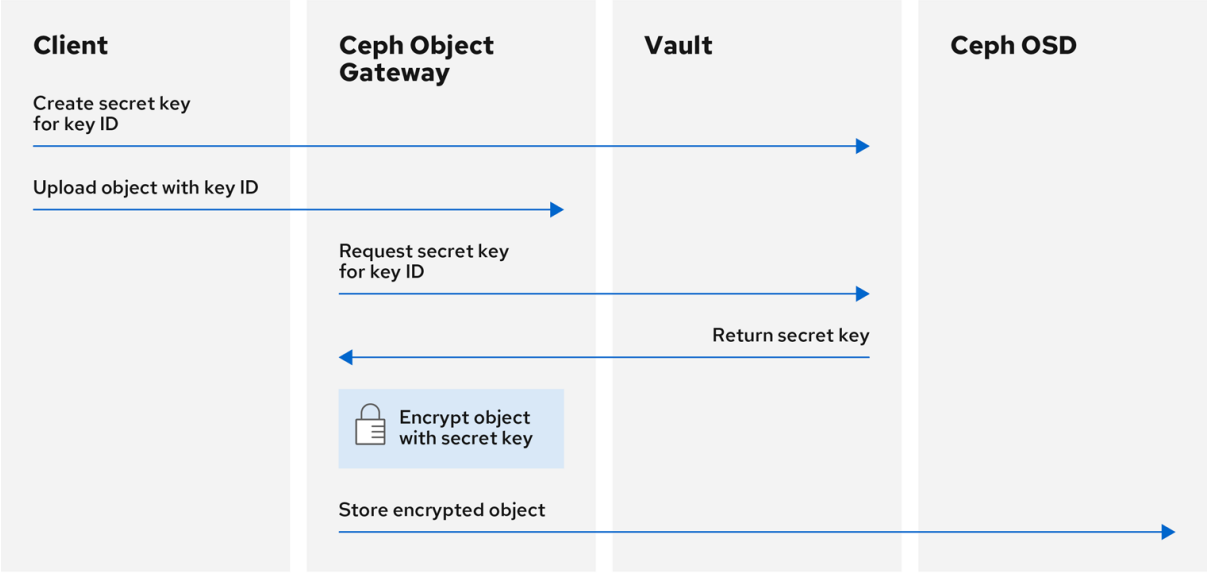
For more information, see the following:

- “High availability service” on page 41
- Installing

Using HashiCorp Vault

As a storage administrator, you can securely store keys, passwords, and certificates in the HashiCorp Vault for use with the Ceph Object Gateway. The HashiCorp Vault provides a secure key management service for server-side encryption used by the Ceph Object Gateway.

Figure: Ceph Vault integration



The basic workflow:

1. The client requests the creation of a secret key from the Vault based on an object's key ID.
2. The client uploads an object with the object's key ID to the Ceph Object Gateway.
3. The Ceph Object Gateway then requests the newly created secret key from the Vault.
4. The Vault replies to the request by returning the secret key to the Ceph Object Gateway.
5. Now the Ceph Object Gateway can encrypt the object using the new secret key.
6. After encryption is done the object is then stored on the Ceph OSD.

Important: Red Hat works with our technology partners to provide this documentation as a service to our customers. However, Red Hat does not provide support for this product. If you need technical assistance for this product, then contact HashiCorp for support.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway software.
- Installation of the HashiCorp Vault software.

Reference

For more information, see *Install Vault*.

Secret engines for Vault

The HashiCorp Vault provides several secret engines to generate, store, or encrypt data. The application programming interface (API) sends data calls to the secret engine requesting an action on that data, and the secret engine returns a result of that action request.

The Ceph Object Gateway supports two of the HashiCorp Vault secret engines:

- "Key/Value version 2" on page 230
- "Transit" on page 230

> **Important:** The secret engines can be configured at any time but the engines are not supported simultaneously.

Key/Value version 2

The Key/Value secret engine stores random secrets within the Vault, on disk. With version 2 of the `kv` engine, a key can have a configurable number of versions. The default number of versions is 10. Deleting a version does not delete the underlying data, but marks the data as deleted, allowing deleted versions to be undeleted. You can use the API endpoint or the **`destroy`** command to permanently remove a version's data. To delete all versions and metadata for a key, you can use the **`metadata`** command or the API endpoint. The key names must be strings, and the engine converts non-string values into strings when using the command line interface. To preserve non-string values, provide a JSON file or use the HTTP application programming interface (API).

> **Note:** For access control list (ACL) policies, the Key/Value secret engine recognizes the distinctions between the `create` and `update` capabilities.

For more information about the Vault's project site, see the *KV Secrets Engine* documentation.

Transit

The Transit secret engine performs cryptographic functions on in-transit data. The Transit secret engine can generate hashes, can be a source of random bytes, and can also sign and verify data. The Vault does not store data when using the Transit secret engine. The Transit secret engine supports key derivation, by allowing the same key to be used for multiple purposes. Also, the transit secret engine supports key versioning.

"Supported key types" on page 230 lists the supported key types for the transit secret engine.

Supported key types

Key type	Description
`aes128-gcm96`	• AES-GCM with a 128-bit AES key and a 96-bit nonce • Supports encryption, decryption, key derivation, and convergent encryption
`aes256-gcm96`	• AES-GCM with a 256-bit AES key and a 96-bit nonce • Supports encryption, decryption, key derivation, and convergent encryption (default)
`chacha20-poly1305`	• ChaCha20-Poly1305 with a 256-bit key • Supports encryption, decryption, key derivation, and convergent encryption
`ed25519`	• Ed25519 • Supports signing, signature verification, and key derivation

Key type	Description
`ecdsa-p256`	• ECDSA using curve P-256 • Supports signing and signature verification
`ecdsa-p384`	• ECDSA using curve P-384 • Supports signing and signature verification
`ecdsa-p521`	• ECDSA using curve P-521 • Supports signing and signature verification
`rsa-2048`	• 2048-bit RSA key • Supports encryption, decryption, signing, and signature verification
`rsa-3072`	• 3072-bit RSA key • Supports encryption, decryption, signing, and signature verification
`rsa-4096`	• 4096-bit RSA key • Supports encryption, decryption, signing, and signature verification

For more information about the Vault's project site, see the *Transit Secrets Engine* documentation.

Authentication for Vault

The HashiCorp Vault supports several types of authentication mechanisms. The Ceph Object Gateway currently supports the Vault Agent method.

The Ceph Object Gateway uses the `rgw_crypt_vault_auth`, and `rgw_crypt_vault_addr` options to configure the use of the HashiCorp Vault.

> **Important:** Red Hat Ceph Storage supports the usage of Vault Agent as the authentication method for containers and the usage of token authentication is not supported on containers.

Vault Agent

The Vault Agent is a daemon that runs on a client node and provides client-side caching, along with token renewal. The Vault Agent typically runs on the Ceph Object Gateway node. Run the Vault Agent and refresh the token file. When the Vault Agent is used in this mode, you can use file system permissions to restrict who has access to the usage of tokens. Also, the Vault Agent can act as a proxy server, that is, Vault will add a token when required and add it to the requests passed to it before forwarding them to the actual server. The Vault Agent can still handle token renewal just as it would when storing a token in the Filesystem. It is required to secure the network that Ceph Object Gateways uses to connect with the Vault Agent, for example, the Vault Agent listens to only the localhost.

For more information, see the HashiCorp Developer Vault Agent documentation.

Namespaces for Vault

Using HashiCorp Vault as an enterprise service provides centralized management for isolated namespaces that teams within an organization can use.

These isolated namespace environments are known as *tenants*, and teams within an organization can utilize these tenants to isolate their policies, secrets, and identities from other teams. The namespace features of Vault help support secure multi-tenancy from within a single infrastructure.

Reference

For more information, see the following:

- *Vault Enterprise Namespaces*

Transit engine compatibility support

There is compatibility support for the previous versions of Ceph which used the Transit engine as a simple key store.

You can use the `compat` option in the Transit engine to configure the compatibility support. You can disable previous support with the following command:

Example

```
[ceph: root@host03 /]# ceph config set client.rgw rgw_crypt_vault_secret_engine transit
compat=0
```

Note: This is the default for future versions and you can use the current version for new installations.

The normal default with the current version is:

Example

```
[ceph: root@host03 /]# ceph config set client.rgw rgw_crypt_vault_secret_engine transit
compat=1
```

This enables the new engine for newly created objects and still allows the old engine to be used for the old objects. To access old and new objects, the Vault token must have both the old and new transit policies.

You can force use only the old engine with the following command:

Example

```
[ceph: root@host03 /]# ceph config set client.rgw rgw_crypt_vault_secret_engine transit
compat=2
```

This mode is selected by default if the Vault ends in `export/encryption-key`.

Important: After configuring the `client.rgw` options, you need to restart the Ceph Object Gateway daemons for the new values to take effect.

Reference

For more information, see the following:

- *Vault Agent*

Creating token policies for Vault

A token policy specifies the powers that all Vault tokens have. One token can have multiple policies. Use the required policy for the configuration.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Installation of the HashiCorp Vault software.
- Root-level access to the HashiCorp Vault node.

Procedure

1. Create a token policy:
   a. For the Key/Value secret engine:

      Example

```
[root@vault ~]# vault policy write rgw-kv-policy -<<EOF
 path "secret/data/*" {
   capabilities = ["read"]
 }
EOF
```

b. For the Transit engine:

Example

```
[root@vault ~]# vault policy write rgw-transit-policy -<<EOF
  path "transit/keys/*" {
    capabilities = [ "create", "update" ]
    denied_parameters = {"exportable" = [], "allow_plaintext_backup" = [] }
  }

  path "transit/keys/*" {
    capabilities = ["read", "delete"]
  }

  path "transit/keys/" {
    capabilities = ["list"]
  }

  path "transit/keys/+/rotate" {
    capabilities = [ "update" ]
  }

  path "transit/*" {
    capabilities = [ "update" ]
  }
EOF
```

Note: If you have used the Transit secret engine on an older version of Ceph, the token policy is:

Example

```
[root@vault ~]# vault policy write old-rgw-transit-policy -<<EOF
  path "transit/export/encryption-key/*" {
    capabilities = ["read"]
  }
EOF
```

If you are using both SSE-KMS and SSE-S3, you should point each to separate containers. You could either use separate Vault instances or separately mount transit instances or different branches under a common transit point. If you are not using separate Vault instances, you can point SSE-KMS or SSE-S3 to serparate containers using `rgw_crypt_vault_prefix` and `rgw_crypt_sse_s3_vault_prefix`. When granting Vault permissions to SSE-KMS bucket owners, you should not give them permission to SSE-S3 keys; only Ceph should have access to the SSE-S3 keys.

Configuring the Ceph Object Gateway to use SSE-S3 with Vault

To configure the Ceph Object Gateway to use the HashiCorp Vault with SSE-S3 for key management, it must be set as the encryption key store.

Currently, the Ceph Object Gateway two secret engines, and two different authentication methods.

Prerequisites

- A running, and healthy Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway software.
- Root-level access to a Ceph Object Gateway node.

Procedure

1. Log into the Cephadm shell.
   Example

```
[root@host01 ~]# cephadm shell
```

2. Enable Vault as the secrets engine to retrieve SSE-S3 encryption keys.

```
ceph config set client.rgw rgw_crypt_sse_s3_backend vault
```

3. Set the authentication method to use with SSE-S3 and Vault.
   - **Method 1** If using **agent** authentication, configure the following settings:

```
ceph config set client.rgw rgw_crypt_sse_s3_vault_auth agent
ceph config set client.rgw rgw_crypt_sse_s3_vault_addr http://
_VAULT_AGENT_:_VAULT_AGENT_PORT_
```

     Example

```
[ceph: root@host01 ~]# ceph config set client.rgw rgw_crypt_sse_s3_vault_auth
agent
[ceph: root@host01 ~]# ceph config set client.rgw rgw_crypt_sse_s3_vault_addr
http://vaultagent:8100
```

     a. Customize the policy as per your use case to set up a Vault agent.

     b. Get the role-id.

```
vault read auth/approle/role/rgw-ap/role-id -format=json | \ jq -r .rgw-ap-
role-id > _PATH_TO_FILE_
```

     c. Get the secret-id.

```
vault read auth/approle/role/rgw-ap/role-id -format=json | \ jq -r .rgw-ap-
secret-id > _PATH_TO_FILE_
```

     d. Create the configuration for the Vault agent.
     Example

```
pid_file = "/run/rgw-vault-agent-pid"
auto_auth {
  method "AppRole" {
    mount_path = "auth/approle"
    config = {
      role_id_file_path ="/usr/local/etc/vault/.rgw-ap-role-id"
      secret_id_file_path ="/usr/local/etc/vault/.rgw-ap-secret-id"
      remove_secret_id_file_after_reading ="false"
    }
  }
}
cache {
  use_auto_auth_token = true
}
listener "tcp" {
  address = "127.0.0.1:8100"
  tls_disable = true
}
vault {
  address = "https://vaultserver:8200"
}
```

     e. Use systemctl to run the persistent daemon.
     Example

```
[root@host01 ~]# /usr/local/bin/vault agent -config=/usr/local/etc/vault/
rgw-agent.hcl
```

     f. A token file is populated with a valid token when the Vault agent runs.

   - **Method 2** If using **token** authentication, configure the following settings:

```
ceph config set client.rgw rgw_crypt_sse_s3_vault_auth token
ceph config set client.rgw rgw_crypt_sse_s3_vault_token_file _PATH_TO_TOKEN_FILE_
ceph config set client.rgw rgw_crypt_sse_s3_vault_addr http://
_VAULT_SERVER_:_VAULT_PORT_
```

Example

```
[ceph: root@host01 ~]# ceph config set client.rgw rgw_crypt_sse_s3_vault_auth
token
[ceph: root@host01 ~]# ceph config set client.rgw
rgw_crypt_sse_s3_vault_token_file /run/.rgw-vault-token
[ceph: root@host01 ~]# ceph config set client.rgw rgw_crypt_sse_s3_vault_addr
http://vaultserver:8200
```

Note: For security reasons, the path to the token file should only be readable by the RADOS Gateway.

4. Set the Vault secret engine to use to retrieve encryption keys, either Key/Value or Transit.
   a. If using **Key/Value**, set the following:
   Example

   ```
   [ceph: root@host01 /]# ceph config set client.rgw
   rgw_crypt_sse_s3_vault_secret_engine kv
   ```

   b. If using **Transit**, set the following:
   Example

   ```
   [ceph: root@host01 /]# ceph config set client.rgw
   rgw_crypt_sse_s3_vault_secret_engine transit
   ```

5. Optional: Configure the Ceph Object Gateway to access Vault within a particular namespace to retrieve the encryption keys.
   Example

   ```
   [ceph: root@host01 /]# ceph config set client.rgw rgw_crypt_sse_s3_vault_namespace
   company/testnamespace1
   ```

   Note: Vault namespaces allow teams to operate within isolated environments known as tenants. The Vault namespaces feature is only available in the Vault Enterprise version.

6. Optional: Restrict access to a particular subset of the Vault secret space by setting a URL path prefix, where the Ceph Object Gateway retrieves the encryption keys from.
   a. If using **Key/Value**, set the following:
   Example

   ```
   [ceph: root@host01 /]# ceph config set client.rgw rgw_crypt_sse_s3_vault_prefix /
   v1/secret/data
   ```

   b. If using **Transit**, set the following:
   Example

   ```
   [ceph: root@host01 /]# ceph config set client.rgw rgw_crypt_sse_s3_vault_prefix /
   v1/transit
   ```

   Assuming the domain name of the Vault server is `vaultserver`, the Ceph Object Gateway will fetch encrypted transit keys from the following URL:

   Example

   ```
   http://vaultserver:8200/v1/transit
   ```

7. Optional: To use custom SSL certification to authenticate with Vault, configure the following settings:

```
ceph config set client.rgw rgw_crypt_sse_s3_vault_verify_ssl true
ceph config set client.rgw rgw_crypt_sse_s3_vault_ssl_cacert PATH_TO_CA_CERTIFICATE
ceph config set client.rgw rgw_crypt_sse_s3_vault_ssl_clientcert
PATH_TO_CLIENT_CERTIFICATE
ceph config set client.rgw rgw_crypt_sse_s3_vault_ssl_clientkey PATH_TO_PRIVATE_KEY
```

Example

```
[ceph: root@host01 /]# ceph config set client.rgw rgw_crypt_sse_s3_vault_verify_ssl
true
[ceph: root@host01 /]# ceph config set client.rgw rgw_crypt_sse_s3_vault_ssl_cacert /
etc/ceph/vault.ca
[ceph: root@host01 /]# ceph config set client.rgw
rgw_crypt_sse_s3_vault_ssl_clientcert /etc/ceph/vault.crt
[ceph: root@host03 /]# ceph config set client.rgw
rgw_crypt_sse_s3_vault_ssl_clientkey /etc/ceph/vault.key
```

8. Restart the Ceph Object Gateway daemons.
   a. To restart the Ceph Object Gateway on an individual node in the storage cluster.

```
systemctl restart ceph-CLUSTER_ID@SERVICE_TYPE.ID.service
```

Example

```
[root@host01 ~]# systemctl restart ceph-c4b34c6f-8365-11ba-
dc31-529020a7702d@rgw.realm.zone.host01.gwasto.service
```

   b. To restart the Ceph Object Gateways on all nodes in the storage cluster.

```
ceph orch restart SERVICE_TYPE
```

Example

```
[ceph: root@host01 /]# ceph orch restart rgw
```

Reference

For more information, see the following:

- “Secret engines for Vault” on page 230
- “Authentication for Vault” on page 231

Creating a key using the kv engine

Configure the HashiCorp Vault Key/Value secret engine (kv) so you can create a key for use with the Ceph Object Gateway. Secrets are stored as key-value pairs in the kv secret engine.

> **Important:** Keys for server-side encryption must be 256-bits long and encoded using base64.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- root or sudo access
- Ceph Object Gateway installed
- Installation of the HashiCorp Vault software.
- Root-level access to the HashiCorp Vault node.

Procedure

1. Enable the Key/Value version 2 secret engine:

   Example

```
vault secrets enable -path secret kv-v2
```

2. Create a new key:

   Syntax

   ```
   vault kv put secret/PROJECT_NAME/BUCKET_NAME key=$(openssl rand -base64 32)
   ```

   Example

   ```
   [root@vault ~]# vault kv put secret/myproject/mybucketkey key=$(openssl rand -base64
   32)

   ====== Metadata ======
   Key              Value
   ---              -----
   created_time     2020-02-21T17:01:09.095824999Z
   deletion_time    n/a
   destroyed        false
   version          1
   ```

Creating a key using the transit engine

Configure the HashiCorp Vault Transit secret engine (`transit`) so you can create a key for use with the Ceph Object Gateway. Creating keys with the Transit secret engine must be exportable to be used for server-side encryption with the Ceph Object Gateway.

PREREQUISITE

- A running Red Hat Ceph Storage cluster.
- Installation of the HashiCorp Vault software.
- Root-level access to the HashiCorp Vault node.

1. Enable the Transit secret engine.

   ```
   vault secrets enable transit
   ```

   For example,

   ```
   [root@vault ~]# vault secrets enable transit
   ```

2. Create a new exportable key

   ```
   vault write -f transit/keys/BUCKET_NAME exportable=true
   ```

   > **Note:** By default this command creates a `aes256-gcm96` type key.

   For example,

   ```
   [root@vault ~]# vault write -f transit/keys/mybucketkey exportable=true
   ```

   a. Enable key rotation.

      ```
      vault write -f transit/keys/BUCKET_NAME/rotate exportable=true
      ```

      For example,

      ```
      [root@vault ~]# vault write -f transit/keys/mybucketkey/rotate exportable=true
      ```

   b. Specify the duration for key rotation.

      ```
      vault write -f transit/keys/BUCKET_NAME/config auto_rotate_period=DURATION
      ```

For example,

```
[root@vault ~]# vault write -f transit/keys/mybucketkey/config
auto_rotate_period=30d
```

In this example, **30d** specifies that the key is rotated after 30 days. To specify the key rotation duration in hours, use **auto_rotate_period=1h**. **1h** specifies that the key rotates every 1 hour.

c. Verify that the key rotation is successful by checking the value of **latest_version** parameter. The value must be incremented.

```
vault read transit/export/encryption-key/BUCKET_NAME
```

For example,

```
[root@vault ~]# vault read transit/export/encryption-key/mybucketkey
```

3. Verify the creation of the key.

> **Note:** Providing the full key path, including the key version, is required.

```
vault read transit/export/encryption-key/BUCKET_NAME/VERSION_NUMBER
```

For example,

```
[root@vault ~]# vault read transit/export/encryption-key/mybucketkey/1

Key     Value
---     -----
keys    map[1:-gbTI9lNpqv/V/2lDcmH2Nq1xKn6FPDWarCmFM2aNsQ=]
name    mybucketkey
type    aes256-gcm96
```

Uploading an object using AWS and the Vault

When uploading an object to the Ceph Object Gateway, the Ceph Object Gateway fetches the key from the Vault, and then encrypts and stores the object in a bucket. When a request is made to download the object, the Ceph Object Gateway automatically retrieves the corresponding key from the Vault and decrypts the object. To upload an object, the Ceph Object Gateway fetches the key from the Vault and then encrypts the object and stores it in the bucket. The Ceph Object Gateway retrieves the corresponding key from the Vault and decrypts the object when there is a request to download the object.

> **Note:** The URL is constructed using the base address, set by the `rgw_crypt_vault_addr` option, and the path prefix, set by the `rgw_crypt_vault_prefix` option.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway software.
- Installation of the HashiCorp Vault software.
- Access to a Ceph Object Gateway client node.
- Access to Amazon Web Services (AWS).

Procedure

1. Upload an object using the AWS command line client and provide the Secure Side Encryption(SSE) key ID in the request:

   a. For the Key/Value secret engine:

      Example

```
[user@client ~]$ aws --endpoint=http://radosgw:8000 s3 cp plaintext.txt s3://
mybucket/encrypted.txt --sse=aws:kms --sse-kms-key-id myproject/mybucketkey
```

Note: In the example, the Ceph Object Gateway would fetch the secret from `http://vault-server:8200/v1/secret/data/myproject/mybucketkey`

b. For the Transit engine:

Example

```
[user@client ~]$ aws --endpoint=http://radosgw:8000 s3 cp plaintext.txt s3://
mybucket/encrypted.txt --sse=aws:kms --sse-kms-key-id mybucketkey
```

Note: In the example, the Ceph Object Gateway would fetch the secret from `http://vaultserver:8200/v1/transit/mybucketkey`

Using IBM GKLM

IBM Security Guardium Key Lifecycle Manager (GKLM) is an enterprise-grade key management system that is designed to help organizations manage and safeguard encryption keys throughout their lifecycle.

As data security demands increase and compliance requirements become stricter, centralized and secure key management becomes paramount. GKLM provides this capability while seamlessly integrating with multiple platforms, including Red Hat Ceph Storage.

IBM GKLM provides centralized key management and automated key rotation, ensuring that encryption keys are updated regularly without manual intervention. It also offers detailed auditing and logging to track key usage and lifecycle events. GKLM adheres to the Key Management Interoperability Protocol (KMIP), among other protocols, and provides high availability through clustering.

Note: Try IBM GKLM for 90-days, For more information, see IBM Guardium Key Lifecycle Manager on ibm.com

Benefits of Using IBM GKLM with Ceph Object Gateway

When integrated with Red Hat Ceph Storage Object through KMIP, IBM GKLM provides secure and compliant key management specifically for encryption at rest (SSE-KMS). This integration helps ensure that data is securely encrypted while stored in Ceph by using keys managed centrally by IBM GKLM. By using IBM GKLM, organizations can enforce strict security policies for data at rest, including key rotation, auditing, and lifecycle management. The centralized management of encryption keys also simplifies compliance with regulatory requirements.

For information about IBM GKLM supported devices, see IBM Security Guardium Key Lifecycle Manager Supported Storage and Non-Storage Devices on IBM Support.

For information about the latest compatible IBM GKLM version for Red Hat Ceph Storage, see “Compatibility matrix for Red Hat Ceph Storage 9.1” on page 0.

IBM GKLM resources

Use the following resources to know more about installing and using IBM GKLM with Ceph Object Gateway.

- **Installing › Installing IBM Security Guardium Key Lifecycle Manager traditional** in the IBM Security Guardium Key Lifecycle Manager documentation on IBM Documentation.
- IBM Security Guardium Key Lifecycle Manager Support Matrix on IBM Support.
- Installing GKLM Traditional on a Windows system in GUI mode on IBM Training.
- Installing GKLM in silent mode on IBM Training.
- IBM Security Guardium Key Lifecycle Manager Redbooks publication.

Multi-factor authentication

When a bucket is configured for object versioning, a developer can optionally configure the bucket to require multi-factor authentication (MFA) for delete requests. Using MFA, a time-based one time password (TOTP) token is passed as a key to the `x-amz-mfa` header. The tokens are generated with virtual MFA devices like Google Authenticator, or a hardware MFA device like those provided by Gemalto.

Use `radosgw-admin` to assign time-based one time password tokens to a user. You must set a secret seed and a serial ID. You can also use `radosgw-admin` to list, remove, and resynchronize tokens.

> **Important:** In a multi-site environment it is advisable to use different tokens for different zones, because, while MFA IDs are set on the user's metadata, the actual MFA one time password configuration resides on the local zone's OSDs.

Term	Description
TOTP	Time-based One Time Password.
Token serial	A string that represents the ID of a TOTP token.
Token seed	The secret seed that is used to calculate the TOTP. It can be hexadecimal or base32.
TOTP seconds	The time resolution used for TOTP generation.
TOTP window	The number of TOTP tokens that are checked before and after the current token when validating tokens.
TOTP pin	The valid value of a TOTP token at a certain time.

Table: Terminology

Creating a seed for multi-factor authentication

To set up multi-factor authentication (MFA), you must create a secret seed for use by the one-time password generator and the back-end MFA system.

Prerequisites

- A Linux system.
- Access to the command line shell.

Procedure

1. Generate a 30 character seed from the `urandom` Linux device file and store it in the shell variable `SEED`:

   Example

   ```
   [user@host01 ~]$ SEED=$(head -10 /dev/urandom | sha512sum | cut -b 1-30)
   ```

2. Print the seed by running echo on the `SEED` variable:

   Example

   ```
   [user@host01 ~]$ echo $SEED
   492dedb20cf51d1405ef6a1316017e
   ```

   Configure the one-time password generator and the back-end MFA system to use the same seed.

Reference

For more information, see the following:

- "Multi-factor authentication" on page 240

Creating a new multi-factor authentication TOTP token

Create a new multi-factor authentication (MFA) time-based one time password (TOTP) token.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Ceph Object Gateway is installed.
- You have root access on a Ceph Monitor node.
- A secret seed for the one-time password generator and Ceph Object Gateway MFA was generated.

Procedure

- Create a new MFA TOTP token:

  Syntax

  ```
  radosgw-admin mfa create --uid=USERID --totp-serial=SERIAL --totp-seed=SEED --totp-
  seed-type=SEED_TYPE --totp-seconds=TOTP_SECONDS --totp-window=TOTP_WINDOW
  ```

  Set *USERID* to the user name to set up MFA on, set *SERIAL* to a string that represents the ID for the TOTP token, and set *SEED* to a hexadecimal or base32 value that is used to calculate the TOTP. The following settings are optional: Set the *SEED_TYPE* to `hex` or `base32`, set *TOTP_SECONDS* to the timeout in seconds, or set *TOTP_WINDOW* to the number of TOTP tokens to check before and after the current token when validating tokens.

  Example

  ```
  [root@host01 ~]# radosgw-admin mfa create --uid=johndoe --totp-serial=MFAtest --totp-
  seed=492dedb20cf51d1405ef6a1316017e
  ```

Reference

For more information, see the following:

- "Creating a seed for multi-factor authentication" on page 240
- "Resynchronizing a multi-factor authentication TOTP token" on page 242

Test a multi-factor authentication TOTP token

Test a multi-factor authentication (MFA) time-based one time password (TOTP) token.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the nodes.
- Ceph Object Gateway is installed.
- You have root access on a Ceph Monitor node.
- An MFA TOTP token was created using `radosgw-admin mfa create`.

Procedure

- Test the TOTP token PIN to verify that TOTP functions correctly:

  Syntax

  ```
  radosgw-admin mfa check --uid=USERID --totp-serial=SERIAL --totp-pin=PIN
  ```

  Set *USERID* to the user name MFA is set up on, set *SERIAL* to the string that represents the ID for the TOTP token, and set *PIN* to the latest PIN from the one-time password generator.

  Example

  ```
  [root@host01 ~]# radosgw-admin mfa check  --uid=johndoe --totp-serial=MFAtest --totp-
  pin=870305
  ok
  ```

  If this is the first time you have tested the PIN, it may fail. If it fails, resynchronize the token.

Reference

For more information, see the following:

- “Creating a seed for multi-factor authentication” on page 240
- “Resynchronizing a multi-factor authentication TOTP token” on page 242

Resynchronizing a multi-factor authentication TOTP token

Resynchronize a multi-factor authentication (MFA) time-based one time password token.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Ceph Object Gateway is installed.
- You have root access on a Ceph Monitor node.
- An MFA TOTP token was created using `radosgw-admin mfa create`.

Procedure

1. Resynchronize a multi-factor authentication TOTP token in case of time skew or failed checks.

   This requires passing in two consecutive pins: the previous pin, and the current pin.

   Syntax

   ```
   radosgw-admin mfa resync --uid=USERID --totp-serial=SERIAL --totp-pin=PREVIOUS_PIN --
   totp=pin=CURRENT_PIN
   ```

   Set *USERID* to the user name MFA is set up on, set *SERIAL* to the string that represents the ID for the TOTP token, set *PREVIOUS_PIN* to the user’s previous PIN, and set *CURRENT_PIN* to the user’s current PIN.

   Example

   ```
   [root@host01 ~]# radosgw-admin mfa resync --uid=johndoe --totp-serial=MFAtest --totp-
   pin=802021 --totp-pin=439996
   ```

2. Verify the token was successfully resynchronized by testing a new PIN:

   Syntax

   ```
   radosgw-admin mfa check --uid=USERID --totp-serial=SERIAL --totp-pin=PIN
   ```

Set *USERID* to the user name MFA is set up on, set *SERIAL* to the string that represents the ID for the TOTP token, and set *PIN* to the user's PIN.

Example

```
[root@host01 ~]# radosgw-admin mfa check  --uid=johndoe --totp-serial=MFAtest --totp-
pin=870305
ok
```

Reference

For more information, see "Creating a seed for multi-factor authentication" on page 240.

Listing multi-factor authentication TOTP tokens

List all multi-factor authentication (MFA) time-based one time password (TOTP) tokens that a particular user has.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Ceph Object Gateway is installed.
- You have root access on a Ceph Monitor node.
- An MFA TOTP token was created using `radosgw-admin mfa create`.

Procedure

- List MFA TOTP tokens:

  Syntax

  ```
  radosgw-admin mfa list --uid=USERID
  ```

  Set *USERID* to the user name MFA is set up on.

  Example

  ```
  [root@host01 ~]# radosgw-admin mfa list --uid=johndoe
  {
      "entries": [
          {
              "type": 2,
              "id": "MFAtest",
              "seed": "492dedb20cf51d1405ef6a1316017e",
              "seed_type": "hex",
              "time_ofs": 0,
              "step_size": 30,
              "window": 2
          }
      ]
  }
  ```

Reference

For more information, see the following:

- *Creating a new multi-factor authentication TOTP token*

Display a multi-factor authentication TOTP token

Display a specific multi-factor authentication (MFA) time-based one time password (TOTP) token by specifying its serial.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Ceph Object Gateway is installed.
- You have root access on a Ceph Monitor node.
- An MFA TOTP token was created using `radosgw-admin mfa create`.

Procedure

- Show the MFA TOTP token:

  Syntax

  ```
  radosgw-admin mfa get --uid=USERID --totp-serial=SERIAL
  ```

  Set *USERID* to the user name MFA is set up on and set *SERIAL* to the string that represents the ID for the TOTP token.

Reference

For more information, see “Creating a new multi-factor authentication TOTP token” on page 241.

Deleting a multi-factor authentication TOTP token or versioned-object

Delete a multi-factor authentication (MFA) time-based one time password (TOTP) token or versioned-object.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Ceph Object Gateway is installed.
- You have root access on a Ceph Monitor node.
- An MFA TOTP token that was created by using the **`radosgw-admin mfa create`** command.

Deleting an MFA TOTP token

1. Delete the MFA TOTP token.

   ```
   radosgw-admin mfa remove --uid=USERID --totp-serial=SERIAL
   ```

   Set *USERID* to the user name MFA is set up on and set *SERIAL* to the string that represents the ID for the TOTP token.
   For example,

   ```
   [root@host01 ~]# radosgw-admin mfa remove --uid=johndoe --totp-serial=MFAtest
   ```

2. Verify that the MFA TOTP token was deleted.

   ```
   radosgw-admin mfa get --uid=USERID --totp-serial=SERIAL
   ```

   Set *USERID* to the user name MFA is set up on and set *SERIAL* to the string that represents the ID for the TOTP token.
   For example,

   ```
   [root@host01 ~]# radosgw-admin mfa get --uid=johndoe --totp-serial=MFAtest
   MFA serial id not found
   ```

Deleting an MFA-enabled versioned object

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- Root user must be authenticated with an MFA to delete a bucket that is versioning enabled.
- MFA-Object you want to delete is in a bucket with versioning enabled.

1. Delete the MFA-enabled versioned object.

```
radosgw-admin object rm --bucket BUCKET_NAME --object OBJECT_NAME --object-version
OBJECT_VERSION --totp-pin TOTP_TOKEN
```

Set *BUCKET_NAME* to the name of the bucket, *OBJECT_NAME* to the name of the object, and *VERSION_ID* to the version ID of the object to delete, along with the TOTP token.
For example,

```
[root@host01 ~]# radosgw-admin object rm --bucket test-mfa --object obj1 --object-
version 4SdKdSmpVAarLLdsFukjUVEwr-2oWfC --totp-pin 562813
```

2. Verify that the MFA-enabled versioned object was deleted.

```
radosgw-admin bucket list --bucket BUCKET_NAME
```

Set *USERID* to the user name MFA is set up on and set *SERIAL* to the string that represents the ID for the TOTP token.
For example,

```
[root@host01 ~]# radosgw-admin bucket list --bucket test-mfa
```

Administrating Ceph Object Gateway

As a storage administrator, you can manage the Ceph Object Gateway using the `radosgw-admin` command line interface (CLI) or using the Red Hat Ceph Storage Dashboard.

> **Note:** Not all of the Ceph Object Gateway features are available to the Red Hat Ceph Storage Dashboard. For details about administrating the Ceph Object Gateway through the dashboard, see Managing Ceph Object Gateway.

Prerequisites

- A healthy running Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway software.

Creating storage policies

The Ceph Object Gateway stores the client bucket and object data by identifying placement targets, and storing buckets and objects in the pools associated with a placement target. If you don't configure placement targets and map them to pools in the instance's zone configuration, the Ceph Object Gateway will use default targets and pools, for example, `default_placement`.

Storage policies give Ceph Object Gateway clients a way of accessing a storage strategy, that is, the ability to target a particular type of storage, such as SSDs, SAS drives, and SATA drives, as a way of ensuring, for example, durability, replication, and erasure coding. For more information, see Storage strategies

To create a storage policy, use the following procedure:

1. Create a new pool `.rgw.buckets.special` with the desired storage strategy. For example, a pool customized with erasure-coding, a particular CRUSH ruleset, the number of replicas, and the `pg_num` and `pgp_num` count.
2. Get the zone group configuration and store it in a file:

   Syntax

   ```
   radosgw-admin zonegroup --rgw-zonegroup=ZONE_GROUP_NAME get > FILE_NAME.json
   ```

   Example

   ```
   [root@host01 ~]# radosgw-admin zonegroup --rgw-zonegroup=default get > zonegroup.json
   ```

3. Add a `special-placement` entry under `placement_target` in the `zonegroup.json` file:

   Example

   ```
   {
           "name": "default",
           "api_name": "",
           "is_master": "true",
           "endpoints": [],
           "hostnames": [],
           "master_zone": "",
           "zones": [{
                   "name": "default",
                   "endpoints": [],
                   "log_meta": "false",
                   "log_data": "false",
                   "bucket_index_max_shards": 5
           }],
           "placement_targets": [{
                   "name": "default-placement",
                   "tags": []
           }, {
                   "name": "special-placement",
                   "tags": []
           }],
           "default_placement": "default-placement"
   }
   ```

4. Set the zone group with the modified `zonegroup.json` file:

   Example

   ```
   [root@host01 ~]# radosgw-admin zonegroup set < zonegroup.json
   ```

5. Get the zone configuration and store it in a file, for example, `zone.json`:

   Example

   ```
   [root@host01 ~]# radosgw-admin zone get > zone.json
   ```

6. Edit the zone file and add the new placement policy key under `placement_pool`:

   Example

   ```
   {
           "domain_root": ".rgw",
           "control_pool": ".rgw.control",
           "gc_pool": ".rgw.gc",
           "log_pool": ".log",
           "intent_log_pool": ".intent-log",
           "usage_log_pool": ".usage",
           "user_keys_pool": ".users",
           "user_email_pool": ".users.email",
           "user_swift_pool": ".users.swift",
           "user_uid_pool": ".users.uid",
           "system_key": {
                   "access_key": "",
                   "secret_key": ""
   ```

```
        },
        "placement_pools": [{
                "key": "default-placement",
                "val": {
                        "index_pool": ".rgw.buckets.index",
                        "data_pool": ".rgw.buckets",
                        "data_extra_pool": ".rgw.buckets.extra"
                }
        }, {
                "key": "special-placement",
                "val": {
                        "index_pool": ".rgw.buckets.index",
                        "data_pool": ".rgw.buckets.special",
                        "data_extra_pool": ".rgw.buckets.extra"
                }
        }]
}
```

7. Set the new zone configuration:

   Example

   ```
   [root@host01 ~]# radosgw-admin zone set < zone.json
   ```

8. Update the zone group map:

   Example

   ```
   [root@host01 ~]# radosgw-admin period update --commit
   ```

   The `special-placement` entry is listed as a `placement_target`.

9. To specify the storage policy when making a request:

   Example

   ```
   $ curl -i http://10.0.0.1/swift/v1/TestContainer/file.txt -X PUT -H "X-Storage-Policy:
   special-placement" -H "X-Auth-Token: AUTH_rgwtxxxxxx"
   ```

Configure bucket index resharding

As a storage administrator, you can configure bucket index resharding in single-site and multi-site deployments to improve performance. Reshard a bucket index either manually offline or dynamically online.

The Ceph Object Gateway stores bucket index data in the index pool, which defaults to the **`.rgw.buckets.index`** parameter. When the client puts many objects in a single bucket without setting quotas for the maximum number of objects per bucket, the index pool can result in significant performance degradation.

- Bucket index resharding prevents performance bottlenecks when you add a high number of objects per bucket.
- You can configure bucket index resharding for new buckets or change the bucket index on the existing ones.
- You need to have the shard count as the nearest prime number to the calculated shard count. The bucket index shards that are prime numbers tend to work better in an evenly distributed bucket index entries across shards.
- Bucket index can be resharded manually or dynamically.
  During the process of resharding bucket index dynamically, there is a periodic check of all the Ceph Object Gateway buckets and it detects buckets that require resharding. If a bucket has grown larger than the value specified in the **`rgw_max_objs_per_shard`** parameter, the Ceph Object Gateway reshards the bucket dynamically in the background. The default value for **`rgw_max_objs_per_shard`** is 100,000 objects per shard. Resharding bucket index dynamically works as expected on the upgraded single-site configuration without any modification to the zone or the zone group. A single site-configuration can be any of the following:
  - A default zone configuration with no realm.

- A non-default configuration with at least one realm.
- A multi-realm single-site configuration.

> **Note:** Versioned buckets may exhibit imbalanced indexes, especially if a small subset of objects are written and re-written. This issue may lead to large omaps on the versioned bucket when a large number of object uploads happen (around a million objects).

Enabling non-blocking bucket resharding

Enable bucket index resharding in both single-site and multi-site deployments to improve RGW client performance.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- Red Hat Ceph Storage deployment uses the version that supports non-blocking resharding.
- All RGW, rgw-admin, and OSD nodes in the cluster are upgraded to the same supported version.
- Any ongoing bucket resharding operations are completed before performing the upgrade.

By default, bucket resharding temporarily blocks write and delete operations, especially on large buckets. Enabling non-blocking resharding allows write and delete operations to continue with minimal disruption while resharding happens in the background.

> **Note:** Dynamic resharding is enabled by default in this release. Verify that it remains enabled.

1. Upgrade the Red Hat Ceph Storage cluster to a version that supports non-blocking resharding.

   > **Note:** Do not perform a partial upgrade. All relevant components must be upgraded. Mismatched versions between components may cause resharding to fail or revert to blocking mode.

2. Verify that all RGW, rgw-admin, and OSD nodes are running the same version.

Result

After the upgrade, bucket resharding occurs in the background without significantly disrupting write or delete operations. This improves performance during resharding events.

Recovering bucket index

Resharding a bucket that was created with **`bucket_index_max_shards = 0`**, removes the bucket's metadata. However, you can restore the bucket indexes by recovering the affected buckets.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- A Ceph Object Gateway is installed at a minimum of two sites.
- The package for `jq` installed.

The `/usr/bin/rgw-restore-bucket-index` tool creates temporary files in the `/tmp` directory. These temporary files consume space based on the bucket objects count from the previous buckets. The previous buckets with more than 10 M objects need more than 4 GB of space in `/tmp` directory. If the storage space in `/tmp` is exhausted, the tool fails with the following message:

```
ln: failed to access '/tmp/rgwrbi-object-list.4053207': No such file or directory
```

The temporary objects are removed.

Run either of the following steps to recover the bucket index.

- Run **radosgw-admin object reindex** command.

  ```
  radosgw-admin object reindex --bucket BUCKET_NAME --object OBJECT_NAME
  ```

- Run the `rgw-restore-bucket-index` tool.

  ```
  /usr/bin/rgw-restore-bucket-index -b BUCKET_NAME -p DATA_POOL_NAME
  ```

  For example,

  ```
  [root@host01 ceph]# /usr/bin/rgw-restore-bucket-index -b bucket-large-1 -p local-
  zone.rgw.buckets.data

  marker is d8a347a4-99b6-4312-a5c1-75b83904b3d4.41610.2
  bucket_id is d8a347a4-99b6-4312-a5c1-75b83904b3d4.41610.2
  number of bucket index shards is 5
  data pool is local-zone.rgw.buckets.data
  NOTICE: This tool is currently considered EXPERIMENTAL.
  The list of objects that we will attempt to restore can be found in "/tmp/rgwrbi-
  object-list.49946".
  Please review the object names in that file (either below or in another window/
  terminal) before proceeding.
  Type "proceed!" to proceed, "view" to view object list, or "q" to quit: view
  Viewing...
  Type "proceed!" to proceed, "view" to view object list, or "q" to quit: proceed!
  Proceeding...
  NOTICE: Bucket stats are currently incorrect. They can be restored with the following
  command after 2 minutes:
      radosgw-admin bucket list --bucket=bucket-large-1 --allow-unordered --max-
  entries=1073741824
  Would you like to take the time to recalculate bucket stats now? [yes/no] yes
  Done

  real    2m16.530s
  user    0m1.082s
  sys    0m0.870s
  ```

  Note:

  - The tool does not work for versioned buckets.

    ```
    [root@host01 ~]# time rgw-restore-bucket-index  --proceed serp-bu-ver-1
    default.rgw.buckets.data
    NOTICE: This tool is currently considered EXPERIMENTAL.
    marker is e871fb65-b87f-4c16-a7c3-064b66feb1c4.25076.5
    bucket_id is e871fb65-b87f-4c16-a7c3-064b66feb1c4.25076.5
    Error: this bucket appears to be versioned, and this tool cannot work with
    versioned buckets.
    ```

  - The tool's scope is limited to a single site only and not multi-site, that is, if you run the **rgw-restore-bucket-index** command at Site A, it does not recover objects in Site B, and vice versa. On a multi-site, the recovery tool and the **object re-index** command must be run at both sites for a bucket.

Creating indexless buckets

You can configure a placement target where created buckets do not use the bucket index to store objects index; that is, indexless buckets. Placement targets that do not use data replication or listing might implement indexless buckets. Indexless buckets provide a mechanism in which the placement target does not track objects in specific buckets. Indexless buckets removes a resource contention that happens whenever an object write happens and reduces the number of round-trips that Ceph Object Gateway needs to make to the Ceph storage cluster. This can have a positive effect on concurrent operations and small object write performance.

PREREQUISITE

- A running Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway software.
- Root-level access to a Ceph Object Gateway node.

Important: The bucket index does not reflect the correct state of the bucket, and listing these buckets does not correctly return their list of objects, which affects multiple features. Specifically, these buckets are not synced in a multi-zone environment because the bucket index is not used to store change information. Do not to use S3 object versioning on indexless buckets because the bucket index is necessary for this feature.

Note: Using indexless buckets removes the limit of the max number of objects in a single bucket.

1. Add a new placement target to the zonegroup by using the **`radosgw-admin zonegroup placement add`** command.
   For example,

   ```
   [ceph: root@host03 /]# radosgw-admin zonegroup placement add --rgw-
   zonegroup="default"
     --placement-id="indexless-placement"
   ```

2. Add a new placement target to the zone by using the **`radosgw-admin zone placement add`** command.
   For example,

   ```
   [ceph: root@host03 /]# radosgw-admin zone placement add --rgw-zone="default"
      --placement-id="indexless-placement"
      --data-pool="default.rgw.buckets.data"
      --index-pool="default.rgw.buckets.index"
      --data_extra_pool="default.rgw.buckets.non-ec"
      --placement-index-type="indexless"
   ```

3. Set the default placement of the zonegroup to `indexless-placement`.
   All buckets that are created in the `indexless-placement` target will be indexless buckets.

   ```
   radosgw-admin zonegroup placement default --placement-id "indexless-placement"
   ```

   For example,

   ```
   [ceph: root@host03 /]# radosgw-admin zonegroup placement default --placement-id
   "indexless-placement"
   ```

4. When the cluster is in a multi-site configuration, update and commit the period.

   ```
   radosgw-admin period update --commit
   ```

   For example,

   ```
   [ceph: root@host03 /]# radosgw-admin period update --commit
   ```

5. Restart the Ceph Object Gateways on all nodes in the storage cluster for the change to take effect.

   ```
   ceph orch restart SERVICE_TYPE
   ```

   For example,

   ```
   [ceph: root@host03 /]# ceph orch restart rgw
   ```

Limitations of bucket index resharding

Understand the limitations of bucket index shardings.

> **Important:** Use the following limitations with caution. There are implications related to your hardware choices and configuration, so you should always discuss these requirements with your Red Hat account team.

- **Maximum number of objects in one bucket before it needs resharding:** Use a maximum of 102,400 objects per bucket index shard. To take full advantage of resharding and maximize parallelism, provide a sufficient number of OSDs in the Ceph Object Gateway bucket index pool. This parallelization scales with the number of Ceph Object Gateway instances, and replaces the in-order index shard enumeration with a number sequence. The default locking timeout is extended from 60 seconds to 90 seconds.
- **Maximum number of objects imposed by bucket index sharding:** Based on prior testing, the number of bucket index shards currently supported is 65521.
- **You can reshard a bucket three times before the other zones catch-up:** Resharding is not recommended until the older generations synchronize. Around four generations of the buckets from previous reshards are supported. Once the limit is reached, dynamic resharding does not reshard the bucket again until at least one of the old log generations are fully trimmed. Using the command `radosgw-admin bucket reshard` throws the following error:

  ```
  Bucket BUCKET_NAME already has too many log generations (4) from previous reshards that peer zones haven't finished syncing.

  Resharding is not recommended until the old generations sync, but you can force a reshard with `--yes-i-really-mean-it`.
  ```

Scalability considerations for dynamic resharding
Ceph Object Gateway automatically reshares a bucket when it exceeds approximately 400 million objects, scaling up to a maximum of 1,999 shards. After reaching this limit, it requires manual resharding to support additional object uploads. If you do not manually reshard beyond this point, the object map may grow too large and degrade bucket performance.

> **Note:** Monitor the shard count proactively and plan for manual resharding before the bucket reaches the 1,999 shard limit to avoid performance issues.

Configuring bucket index resharding in simple deployments

To enable and configure bucket index resharding on all new buckets, use the **`rgw_override_bucket_index_max_shards`** parameter.

You can set the parameter to one of the following values:

- `0` to disable bucket index sharding, which is the default value.
- A value greater than `0` to enable bucket sharding and to set the maximum number of shards.

Prerequisites

- Two running Red Hat Ceph Storage clusters.
- A Ceph Object Gateway installed.

Procedure

1. Calculate the recommended number of shards:

   ```
   number of objects expected in a bucket / 100,000
   ```

> **Note:** The maximum number of bucket index shards currently supported is 65,521.

2. Set the `rgw_override_bucket_index_max_shards` option accordingly:

   Syntax

   ```
   ceph config set client.rgw rgw_override_bucket_index_max_shards VALUE
   ```

   Replace *VALUE* with the recommended number of shards calculated:

   Example

   ```
   [ceph: root@host01 /]# ceph config set client.rgw rgw_override_bucket_index_max_shards
   12
   ```

   - To configure bucket index resharding for all instances of the Ceph Object Gateway, set the `rgw_override_bucket_index_max_shards` parameter with the `global` option.
   - To configure bucket index resharding only for a particular instance of the Ceph Object Gateway, add `rgw_override_bucket_index_max_shards` parameter under the instance.

3. Restart the Ceph Object Gateways on all nodes in the cluster to take effect:

   Syntax

   ```
   ceph orch restart SERVICE_TYPE
   ```

   Example

   ```
   [ceph: root#host01 /]# ceph orch restart rgw
   ```

Reference

For more information, see the following:

- “Resharding bucket index dynamically” on page 254
- “Resharding bucket index manually” on page 259

Configuring bucket index resharding in multi-site deployments

In multi-site deployments, each zone can have a different `index_pool` setting to manage failover. To configure a consistent shard count for zones in one zone group, set the `bucket_index_max_shards` parameter in the configuration for that zone group.

The default value of `bucket_index_max_shards` parameter is 11.

You can set the parameter to one of the following values:

- `0` to disable bucket index sharding.
- A value greater than `0` to enable bucket sharding and to set the maximum number of shards.

> **Note:** Mapping the index pool, for each zone, if applicable, to a CRUSH ruleset of SSD-based OSDs might also help with bucket index performance. See *CRUSH performance domains*

> **Important:** To prevent sync issues in multi-site deployments, a bucket should not have more than three generation gaps.

Prerequisites

- Two running Red Hat Ceph Storage clusters.
- A Ceph Object Gateway installed at a minimum of two sites.

Procedure

1. Calculate the recommended number of shards:

```
number of objects expected in a bucket / 100,000
```

> **Note:** The maximum number of bucket index shards currently supported is 65,521.

2. Extract the zone group configuration to the `zonegroup.json` file:

   Example

```
[ceph: root@host01 /]# radosgw-admin zonegroup get > zonegroup.json
```

3. In the `zonegroup.json` file, set the `bucket_index_max_shards` parameter for each named zone:

   Syntax

```
bucket_index_max_shards = VALUE
```

   Replace *VALUE* with the recommended number of shards calculated:

   Example

```
bucket_index_max_shards = 12
```

4. Reset the zone group:

   Example

```
[ceph: root@host01 /]# radosgw-admin zonegroup set < zonegroup.json
```

5. Update the period:

   Example

```
[ceph: root@host01 /]# radosgw-admin period update --commit
```

6. Check if resharding is complete:

   Syntax

```
radosgw-admin reshard status --bucket BUCKET_NAME
```

   Example

```
[ceph: root@host01 /]# radosgw-admin reshard status --bucket data
```

Verification

- Check the sync status of the storage cluster:

  Example

```
[ceph: root@host01 /]# radosgw-admin sync status
```

Resharding bucket index dynamically

Dynamic resharding is a process where the system automatically reshards the bucket index without requiring manual intervention. The system monitors factors like data distribution and triggers resharding when necessary to optimize storage and performance.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- A Ceph Object Gateway installed.

When dynamic resharding is needed, the system automatically adds the bucket to the resharding queue, and background threads perform the resharding one bucket at a time. This ensures the system adapts to changing demands efficiently.

This process is seamless and happens in the background, allowing the system to adjust dynamically to varying workloads without admin intervention.

> **Note:**
> For more information, see the following:
>
> - “Cleaning stale instances of bucket entries after resharding” on page 260
> - “Resharding bucket index manually” on page 259
> - “Configuring bucket index resharding in simple deployments” on page 251

1. In the Ceph configuration file, set the **`rgw_dynamic_resharding`** parameter to `true`.

   ```
   ceph config set client.rgw OPTION VALUE
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph config set client.rgw rgw_dynamic_resharding true
   ```

2. Optional: Customize the Ceph configuration with any of the optional parameters.

   ```
   ceph config set client.rgw OPTION VALUE
   ```

Ceph configuration options for dynamic resharding

Parameter option	Description	Default value
`rgw_reshard_num_logs`	The number of shards for the resharding log.	`16`
`rgw_reshard_bucket_lock_duration`	The duration in seconds of the lock on a bucket during resharding.	`360`
`rgw_dynamic_resharding`	A boolean value that enables or disables dynamic resharding.	`true`
`rgw_max_objs_per_shard`	The maximum number of objects per shard.	`100000`
`rgw_reshard_thread_interval`	The maximum time in seconds between rounds of reshard thread processing.	`600`

Parameter option	Description	Default value
rgw_dynamic_resharding_may_reduce	Set to`true`to allow shard reduction during dynamic resharding. Must be used with **rgw_dynamic_reshard** set to `true`for dynamic resharding.	`true`
rgw_dynamic_resharding_reduction_wait	Defines the wait time in hours before performing a reduction after identifying a bucket for reduction. This option provides a protection mechanism to prevent constant resharding due to fluctuating object counts. **Note:** A low value can affect cluster performance for buckets with frequent object count fluctuations.	`120`

The following example increases the number of shards for the resharding log to 23.

```
[ceph: root@host01 /]# ceph config set client.rgw rgw_reshard_num_logs 23
```

3. Add a bucket to the resharding queue.

```
radosgw-admin reshard add --bucket BUCKET --num-shards NUMBER
```

For example,

```
[ceph: root@host01 /]# radosgw-admin reshard add --bucket data --num-shards 10
```

4. List the resharding queue.

```
radosgw-admin reshard list
```

For example,

```
[ceph: root@host01 /]# radosgw-admin reshard list
```

5. Check the bucket log generations and shards.

```
radosgw-admin bucket layout --bucket data
```

For example,

```
[ceph: root@host01 /]# radosgw-admin bucket layout --bucket data
{
    "layout": {
        "resharding": "None",
        "current_index": {
            "gen": 1,
            "layout": {
                "type": "Normal",
                "normal": {
                    "num_shards": 23,
                    "hash_type": "Mod"
                }
            }
        },
        "logs": [
            {
                "gen": 0,
                "layout": {
                    "type": "InIndex",
                    "in_index": {
                        "gen": 0,
                        "layout": {
                            "num_shards": 11,
                            "hash_type": "Mod"
                        }
                    }
                }
            },
            {
                "gen": 1,
                "layout": {
                    "type": "InIndex",
                    "in_index": {
                        "gen": 1,
                        "layout": {
                            "num_shards": 23,
                            "hash_type": "Mod"
                        }
                    }
                }
            }
        ]
    }
}
```

6. Check bucket resharding status.

```
radosgw-admin reshard status --bucket BUCKET
```

For example,

```
[ceph: root@host01 /]# radosgw-admin reshard status --bucket data
```

7. Immediately process task entries on the resharding queue.

```
radosgw-admin reshard process
```

For example,

```
[ceph: root@host01 /]# radosgw-admin reshard process
```

8. Optional: Cancel pending bucket resharding tasks.

> **Important:** You can only cancel `pending` resharding operations. Do not cancel `ongoing` resharding operations.

```
radosgw-admin reshard cancel --bucket BUCKET
```

For example,

```
[ceph: root@host01 /]# radosgw-admin reshard cancel --bucket data
```

AFTER COMPLETING THE TASK

Verify the bucket resharding status.

```
radosgw-admin reshard status --bucket BUCKET
```

For example,

```
[ceph: root@host01 /]# radosgw-admin reshard status --bucket data
```

Resharding bucket index dynamically in multi-site configuration

The feature allows buckets to be resharded in a multi-site configuration without interrupting the replication of their objects.

PREREQUISITE

- At least two running Red Hat Ceph Storage clusters.
- All the Ceph Object Gateway daemons enabled at both the sites are upgraded to the latest version.
- Root-level access to all the nodes.

When `rgw_dynamic_resharding` is enabled, it runs on each zone independently, and the zones might choose different shard counts for the same bucket.
You need to enable the `resharding` feature manually on the existing zones and the zone groups after upgrading the storage cluster.

> **Note:**
>
> - You can reshard a bucket three times before the other zones catch-up. For more information, see “Limitations of bucket index resharding” on page 251.
> - Zones and zone groups are supported and enabled by default.
> - If a bucket is created and uploaded with more than the threshold number of objects for resharding dynamically, you need to continue to write I/Os to old buckets to begin the resharding process.

For more information, see “Resharding bucket index dynamically” on page 254.

1. Check if `resharding` is enabled on the zonegroup.

   ```
   radosgw-admin sync status
   ```

   For example,

   ```
   [ceph: root@host01 /]# radosgw-admin sync status
   ```

   If `zonegroup features enabled` is not enabled for resharding on the zonegroup, then continue with the procedure.

2. Enable the `resharding` feature on all the zonegroups in the multi-site configuration where Ceph Object Gateway is installed.

   ```
   radosgw-admin zonegroup modify --rgw-zonegroup=ZONEGROUP_NAME --enable-
   feature=resharding
   ```

   For example,

   ```
   [ceph: root@host01 /]# radosgw-admin zonegroup modify --rgw-zonegroup=us --enable-
   feature=resharding
   ```

3. Update the period and commit.

```
radosgw-admin period update --commit
```

For example,

```
[ceph: root@host01 /]# radosgw-admin period update --commit
```

4. Enable the `resharding` feature on all the zones in the multi-site configuration where Ceph Object Gateway is installed.

```
radosgw-admin zone modify --rgw-zone=ZONE_NAME --enable-feature=resharding
```

For example,

```
[ceph: root@host01 /]# radosgw-admin zone modify --rgw-zone=us-east --enable-
feature=resharding
```

5. Update the period and commit.

```
radosgw-admin period update --commit
```

6. Verify the `resharding` feature is enabled on the zones and zonegroups.
You can see that each zone lists its `supported_features` and the zonegroups lists its `enabled_features`.

```
radosgw-admin period get
```

For example,

```
[ceph: root@host01 /]# radosgw-admin period get

"zones": [
                    {
                        "id": "505b48db-6de0-45d5-8208-8c98f7b1278d",
                        "name": "us_east",
                        "endpoints": [
                            "http://10.0.208.11:8080"
                        ],
                        "log_meta": "false",
                        "log_data": "true",
                        "bucket_index_max_shards": 11,
                        "read_only": "false",
                        "tier_type": "",
                        "sync_from_all": "true",
                        "sync_from": [],
                        "redirect_zone": "",
                        "supported_features": [
                            "resharding"
                        ]
                "default_placement": "default-placement",
                "realm_id": "26cf6f23-c3a0-4d57-aae4-9b0010ee55cc",
                "sync_policy": {
                    "groups": []
                },
                "enabled_features": [
                    "resharding"
                ]
```

7. Check the sync status.

```
radosgw-admin sync status
```

For example,

```
[ceph: root@host01 /]# radosgw-admin sync status
          realm 26cf6f23-c3a0-4d57-aae4-9b0010ee55cc (usa)
      zonegroup 33a17718-6c77-493e-99fe-048d3110a06e (us)
           zone 505b48db-6de0-45d5-8208-8c98f7b1278d (us_east)
zonegroup features enabled: resharding
```

In this example, the `resharding` feature is enabled for the `us` zonegroup.

8. Optional: Disable the `resharding` feature for the zonegroups.

Important: To disable resharding on any singular zone, set the `rgw_dynamic_resharding` configuration option to `false` on that specific zone.

a. Disable the feature on all the zonegroups in the multi-site where Ceph Object Gateway is installed.

```
radosgw-admin zonegroup modify --rgw-zonegroup=ZONEGROUP_NAME --disable-
feature=resharding
```

For example,

```
[ceph: root@host01 /]# radosgw-admin zonegroup modify --rgw-zonegroup=us --
disable-feature=resharding
```

b. Update the period and commit.

```
radosgw-admin period update --commit
```

Resharding bucket index manually

If a bucket has grown larger than the initial configuration for which it was optimized, reshard the bucket index pool by using the **`radosgw-admin bucket reshard`** command.

This command performs the following tasks:

- Creates a new set of bucket index objects for the specified bucket.
- Distributes object entries across these bucket index objects.
- Creates a new bucket instance.
- Links the new bucket instance with the bucket so that all new index operations go through the new bucket indexes.
- Prints the old and the new bucket ID to the command output.

Prerequisites

- At least two running Red Hat Ceph Storage clusters.
- A Ceph Object Gateway installed at a minimum of two sites.

Procedure

1. Back up the original bucket index:

   Syntax

   ```
   radosgw-admin bi list --bucket=BUCKET > BUCKET.list.backup
   ```

   Example

   ```
   [ceph: root@host01 /]# radosgw-admin bi list --bucket=data > data.list.backup
   ```

2. Reshard the bucket index:

   Syntax

   ```
   radosgw-admin bucket reshard --bucket=BUCKET --num-shards=NUMBER
   ```

   Example

   ```
   [ceph: root@host01 /]# radosgw-admin bucket reshard --bucket=data --num-shards=100
   ```

Verification

- Check bucket resharding status:

  Syntax

  ```
  radosgw-admin reshard status --bucket bucket
  ```

  Example

  ```
  [ceph: root@host01 /]# radosgw-admin reshard status --bucket data
  ```

Cleaning stale instances of bucket entries after resharding

The resharding process might not clean stale instances of bucket entries automatically and these instances can impact performance of the storage cluster.

Clean them manually to prevent the stale instances from negatively impacting the performance of the storage cluster.

Important: Before cleaning stale instances, contact Red Hat Support.

Important: Use this procedure only in simple deployments, not in multi-site clusters.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- A Ceph Object Gateway node.

Procedure

1. List stale instances:

   ```
   [ceph: root@host01 /]# radosgw-admin reshard stale-instances list
   ```

2. Clean the stale instances of the bucket entries:

   ```
   [ceph: root@host01 /]# radosgw-admin reshard stale-instances rm
   ```

Verification

- Check bucket resharding status:

  Syntax

  ```
  radosgw-admin reshard status --bucket BUCKET
  ```

  Example

  ```
  [ceph: root@host01 /]# radosgw-admin reshard status --bucket data
  ```

Fixing lifecycle policies after resharding

For storage clusters with resharded instances, the old lifecycle processes would have flagged and deleted the lifecycle processing as the bucket instance changed during a reshard.

However, for older buckets that had lifecycle policies and have undergone resharding, you can fix such buckets with the `reshard fix` option.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- A Ceph Object Gateway installed.

Procedure

- Fix the lifecycle policies of the older bucket:

  Syntax

  ```
  radosgw-admin lc reshard fix --bucket BUCKET_NAME
  ```

  Important: If you do not use the `--bucket` argument, then the command fixes lifecycle policies for all the buckets in the storage cluster.

  Example

  ```
  [ceph: root@host01 /]# radosgw-admin lc reshard fix --bucket mybucket
  ```

Creating indexless buckets

You can configure a placement target where created buckets do not use the bucket index to store objects index; that is, indexless buckets. Placement targets that do not use data replication or listing might implement indexless buckets. Indexless buckets provide a mechanism in which the placement target does not track objects in specific buckets. Indexless buckets removes a resource contention that happens whenever an object write happens and reduces the number of round-trips that Ceph Object Gateway needs to make to the Ceph storage cluster. This can have a positive effect on concurrent operations and small object write performance.

PREREQUISITE

- A running Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway software.
- Root-level access to a Ceph Object Gateway node.

Important: The bucket index does not reflect the correct state of the bucket, and listing these buckets does not correctly return their list of objects, which affects multiple features. Specifically, these buckets are not synced in a multi-zone environment because the bucket index is not used to store change information. Do not to use S3 object versioning on indexless buckets because the bucket index is necessary for this feature.

Note: Using indexless buckets removes the limit of the max number of objects in a single bucket.

1. Add a new placement target to the zonegroup by using the **`radosgw-admin zonegroup placement add`** command.
   For example,

   ```
   [ceph: root@host03 /]# radosgw-admin zonegroup placement add --rgw-
   zonegroup="default"
     --placement-id="indexless-placement"
   ```

2. Add a new placement target to the zone by using the **radosgw-admin zone placement add** command.
   For example,

```
[ceph: root@host03 /]# radosgw-admin zone placement add --rgw-zone="default"
   --placement-id="indexless-placement"
   --data-pool="default.rgw.buckets.data"
   --index-pool="default.rgw.buckets.index"
   --data_extra_pool="default.rgw.buckets.non-ec"
   --placement-index-type="indexless"
```

3. Set the default placement of the zonegroup to`indexless-placement`.
   All buckets that are created in the `indexless-placement` target will be indexless buckets.

```
radosgw-admin zonegroup placement default --placement-id "indexless-placement"
```

   For example,

```
[ceph: root@host03 /]# radosgw-admin zonegroup placement default --placement-id
"indexless-placement"
```

4. When the cluster is in a multi-site configuration, update and commit the period.

```
radosgw-admin period update --commit
```

   For example,

```
[ceph: root@host03 /]# radosgw-admin period update --commit
```

5. Restart the Ceph Object Gateways on all nodes in the storage cluster for the change to take effect.

```
ceph orch restart SERVICE_TYPE
```

   For example,

```
[ceph: root@host03 /]# ceph orch restart rgw
```

Enabling compression

The Ceph Object Gateway supports server-side compression of uploaded objects using any of Ceph's compression plugins.

The supported compression plugins include the following:

- `zlib`
- `snappy`
- `zstd`

Configuring compression

To enable compression on a zone's placement target, provide the `--compression=TYPE` option to the **radosgw-admin zone placement modify** command. The compression `TYPE` refers to the name of the compression plugin to use when writing new object data.

Each compressed object stores the compression type. Changing the setting does not hinder the ability to decompress existing compressed objects, nor does it force the Ceph Object Gateway to re-compress existing objects.

This compression setting applies to all new objects uploaded to buckets using this placement target.

To disable compression on a zone's placement target, provide the `--compression=TYPE` option to the **radosgw-admin zone placement modify** command and specify an empty string or `none`.

Example

```
[root@host01 ~] radosgw-admin zone placement modify --rgw-zone=default --placement-
id=default-placement --compression=zlib
{
...
    "placement_pools": [
        {
            "key": "default-placement",
            "val": {
                "index_pool": "default.rgw.buckets.index",
                "data_pool": "default.rgw.buckets.data",
                "data_extra_pool": "default.rgw.buckets.non-ec",
                "index_type": 0,
                "compression": "zlib"
            }
        }
    ],
...
}
```

After enabling or disabling compression, restart the Ceph Object Gateway instance so the change will take effect.

> **Note:** Ceph Object Gateway creates a `default` zone and a set of pools. For production deployments, see "Creating a realm" on page 59.

Compression statistics

While all existing commands and APIs continue to report object and bucket sizes based on their uncompressed data, the **`radosgw-admin bucket stats`** command includes compression statistics for all buckets.

The usage types for the **`radosgw-admin bucket stats`** command are as follows:

- `rgw.main`, for regular entries or objects.
- `rgw.multimeta`, for the metadata of incomplete multipart uploads.
- `rgw.cloudtiered`, for objects that a lifecycle policy has transitioned to a cloud tier. When configured with `retain_head_object=true`, a head object that no longer contains data remains but can still serve the object's metadata with HeadObject requests. These stub head objects use the `rgw.cloudtiered` category. For more information, see Transitioning data to Amazon S3 cloud service.

Syntax

```
radosgw-admin bucket stats --bucket=BUCKET_NAME
{
...
    "usage": {
        "rgw.main": {
            "size": 1075028,
            "size_actual": 1331200,
            "size_utilized": 592035,
            "size_kb": 1050,
            "size_kb_actual": 1300,
            "size_kb_utilized": 579,
            "num_objects": 104
        }
    },
...
}
```

The `size` is the accumulated size of the objects in the bucket, uncompressed and unencrypted. The `size_kb` is the accumulated size in kilobytes and is calculated as `size/1024`. In this example, it is `1075028/1024 = 1050`.

The `size_actual` is the actual size of all the objects after each object is rounded up with the ceiling function to the nearest 4096 bytes since the object is stored in 4096-sized chunks. If a bucket has two objects, one of size 4100 bytes and the other of 8500 bytes, the first object is rounded up to 8192 bytes, and the second one rounded 12288 bytes, and their total for the bucket is 20480 bytes. The `size_kb_actual` is the actual size in kilobytes and is calculated as `size_actual/1024`. In this example, it is `1331200/1024 = 1300`.

The `size_utilized` is the total size of the compressed data in bytes. This considers both compression and encryption. Encryption could increase the size of the object while compression could decrease it. The

`size_kb_utilized` is the total size in kilobytes and is calculated as `size_utilized/1024`. In this example, it is `592035/1024= 579`.

Here, all the sizes in kilobytes is rounded up with the ceiling function.

Quota management

The Ceph Object Gateway enables you to set quotas on users and buckets owned by users. Quotas include the maximum number of objects in a bucket and the maximum storage size in megabytes.

- *Bucket:* The `--bucket` option allows you to specify a quota for buckets the user owns.
- *Maximum Objects:* The `--max-objects` setting allows you to specify the maximum number of objects. A negative value disables this setting.
- *Maximum Size:* The `--max-size` option allows you to specify a quota for the maximum number of bytes. A negative value disables this setting.
- *Quota Scope:* The `--quota-scope` option sets the scope for the quota. The options are `bucket` and `user`. Bucket quotas apply to buckets a user owns. User quotas apply to a user.

> **Important:** Buckets with a large number of objects can cause serious performance issues. The recommended maximum number of objects in a one bucket is 100,000. To increase this number, configure bucket index sharding. For more information, see “Limitations of bucket index resharding” on page 251.

To start managing quotas, follow this procedure:

1. “Set user quotas” on page 264.
2. “Enable and disable user quotas” on page 264.
3. “Set bucket quotas” on page 265.
4. “Enable and disable bucket quotas” on page 265.

Set user quotas

Before you enable a quota, you must first set the quota parameters.

Syntax

```
radosgw-admin quota set --quota-scope=user --uid=USER_ID [--max-objects=NUMBER_OF_OBJECTS]
[--max-size=MAXIMUM_SIZE_IN_BYTES]
```

Example

```
[root@host01 ~]# radosgw-admin quota set --quota-scope=user --uid=johndoe --max-
objects=1024 --max-size=1024
```

A negative value for num objects and / or max size means that the specific quota attribute check is disabled.

Enable and disable user quotas

Once you set a user quota, you can enable it.

Syntax

```
radosgw-admin quota enable --quota-scope=user --uid=USER_ID
```

You may disable an enabled user quota.

Syntax

```
radosgw-admin quota disable --quota-scope=user --uid=USER_ID
```

Set bucket quotas

Bucket quotas apply to the buckets owned by the specified `uid` and are independent of the user.

Syntax

```
radosgw-admin quota set --uid=USER_ID --quota-scope=bucket --bucket=BUCKET_NAME [--max-
objects=NUMBER_OF_OBJECTS] [--max-size=MAXIMUM_SIZE_IN_BYTES]
```

A negative value for *NUMBER_OF_OBJECTS*, *MAXIMUM_SIZE_IN_BYTES*, or both means that the specific quota attribute check is disabled.

Enable and disable bucket quotas

Enable a bucket quota after it is set. Disable the quota, by using the **`diable`** command.

Syntax

```
radosgw-admin quota enable --quota-scope=bucket --uid=USER_ID
```

You may disable an enabled bucket quota.

Syntax

```
radosgw-admin quota disable --quota-scope=bucket --uid=USER_ID
```

Get quota settings

You can access each user's quota settings via the user information API.

To read user quota setting information with the CLI interface, run the following command:

Syntax

```
radosgw-admin user info --uid=USER_ID
```

To get quota settings for a tenanted user, specify the user ID and the name of the tenant:

Syntax

```
radosgw-admin user info --uid=USER_ID --tenant=TENANT
```

Update quota stats

Quota stats get updated asynchronously. You can update quota statistics for all users and all buckets manually to retrieve the latest quota stats.

Syntax

```
radosgw-admin user stats --uid=USER_ID --sync-stats
```

Get user quota usage stats

Get the latest user quota usage stats to see how much of the quota a user has used.

To see how much of the quota a user has consumed, run the following command:

Syntax

```
radosgw-admin user stats --uid=USER_ID
```

Note: You should run the `radosgw-admin user stats` command with the `--sync-stats` option to receive the latest data.

Quota cache

Quota statistics are cached for each Ceph Gateway instance. If there are multiple instances, then the cache can keep quotas from being perfectly enforced, as each instance will have a different view of the quotas.

The options that control how the cache keeps the quotas are `rgw bucket quota ttl`, `rgw user quota bucket sync interval`, and `rgw user quota sync interval`. The higher these values are, the more efficient quota operations are, but the more out-of-sync multiple instances will be. The lower these values are, the closer to perfect enforcement multiple instances will achieve. If all three are 0, then quota caching is effectively disabled, and multiple instances will have perfect quota enforcement.

For more details on these options, see "Configuration reference" on page 297.

Reading and writing global quotas

You can read and write quota settings in a zonegroup map.

To get a zonegroup map:

```
[root@host01 ~]# radosgw-admin global quota get
```

The global quota settings can be manipulated with the `global quota` counterparts of the `quota set`, `quota enable`, and `quota disable` commands, for example:

```
[root@host01 ~]# radosgw-admin global quota set --quota-scope bucket --max-objects 1024
[root@host01 ~]# radosgw-admin global quota enable --quota-scope bucket
```

Note: In a multi-site configuration, where there is a realm and period present, changes to the global quotas must be committed using `period update --commit`. If there is no period present, the Ceph Object Gateways must be restarted for the changes to take effect.

Usage

The Ceph Object Gateway logs usage for each user. You can also track user usage within date ranges.

Options include:

- *Start Date:* The `--start-date` option allows you to filter usage stats from a particular start date (*format:* `yyyy-mm-dd[HH:MM:SS]`).
- *End Date:* The `--end-date` option allows you to filter usage up to a particular date (*format:* `yyyy-mm-dd[HH:MM:SS]`).
- *Log Entries:* The `--show-log-entries` option allows you to specify whether or not to include log entries with the usage stats (options: `true` | `false`).

Note: You can specify time with minutes and seconds, but it is stored with 1 hour resolution.

Show usage

To show usage statistics, specify the `usage show`.

To show usage for a particular user, you must specify a user ID. You may also specify a start date, end date, and whether or not to show log entries.

Example

```
[ceph: root@host01 /]# radosgw-admin usage show
                --uid=johndoe --start-date=2022-06-01
                --end-date=2022-07-01
```

You may also show a summary of usage information for all users by omitting a user ID.

Example

```
[ceph: root@host01 /]# radosgw-admin usage show --show-log-entries=false
```

Trim usage

With heavy use, usage logs can begin to take up storage space. You can trim usage logs for all users and for specific users. You may also specify date ranges for trim operations.

Example

```
[ceph: root@host01 /]# radosgw-admin usage trim --start-date=2022-06-01
                     --end-date=2022-07-31

[ceph: root@host01 /]# radosgw-admin usage trim --uid=johndoe
[ceph: root@host01 /]# radosgw-admin usage trim --uid=johndoe --end-date=2021-04-31
```

Deduplication statistics

Use the Ceph Object Gateway deduplication statistics feature to estimate potential storage savings. This command does not perform deduplication or modify data.

The Ceph Object Gateway deduplication statistics utility helps administrators identify redundant data across Ceph Object Gateway buckets. It uses radosgw-admin subcommands to start and manage estimation tasks. The utility reports duplication estimates but does not modify data.

The utility scans bucket index metadata instead of object data. This design reduces I/O load and enables fast, scalable analysis.

Use this utility to assess whether deduplication can reduce storage consumption in your Ceph Object Gateway environment. It supports capacity planning, storage optimization, and analysis of object storage patterns.

How estimation works

The estimation process runs in parallel across multiple Ceph Object Gateway daemons. It reads each bucket index shard once, which allows the workload to scale efficiently. The process does not read object data or metadata, so it runs independently of storage media performance, whether SSD or HDD.

Skipped objects

The utility excludes these objects from estimation:

- Objects smaller than 4 MB, unless Multipart.
- Objects stored with different placement rules, pools, or storage classes.
- Deduplication skips data storage pools that use a replicated rule.

Note: Do not use RAID. Red Hat Ceph Storage uses object copies and erasure coding, which eliminate the need for RAID solutions. A degraded RAID negatively impacts performance, and data recovery through RAID is significantly slower compared to recovery with object replicas or erasure-coded chunks.

Note: The deduplication process skips compressed and user-encrypted objects. The estimation process includes them because it cannot detect compression or encryption.

Memory usage

The utility uses a small, predictable amount of memory based on the object count. The following table lists the reference values.

Ceph Object Gateway object count	Approximate memory usage
1 million	8 MB
4 million	16 MB
16 million	32 MB
64 million	64 MB
256 million	128 MB
1 billion	256 MB
4 billion	512 MB
16 billion	1024 MB (1 GB)

Running a deduplication estimate

Run the **`radosgw-admin dedup estimate`** command to start deduplication estimation. The command identifies duplicate objects but does not modify data.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place.

- Administrative access to the Ceph cluster.
- RGW service is running.
- radosgw-admin tool is installed and available in your system path.

1. Open a terminal on a node that has access to the RGW service.
2. Run the **`radosgw-admin dedup estimate`** command to start the estimate. Confirm that the system displays a message indicating the estimate has started.

```
radosgw-admin dedup estimate
```

> **Note:** Starting a new estimate automatically aborts the current estimation process

Result

To check progress or view results, run the **`radosgw-admin dedup stats`** command.

Monitoring the estimate

- Run the following command to check progress or view results.

```
radosgw-admin dedup stats
```

Managing the estimate

- Use the following commands to manage the deduplication estimate:
  - Pause the estimate:

```
radosgw-admin dedup pause
```

  - Resume a paused estimate:

```
radosgw-admin dedup resume
```

- Cancel the estimate and release resources:

```
radosgw-admin dedup abort
```

Optimize the Ceph Object Gateway's data object storage

Bucket lifecycle configuration optimizes data object storage to increase its efficiency and to provide effective storage throughout the lifetime of the data.

The S3 API in the Ceph Object Gateway currently supports a subset of the AWS bucket lifecycle configuration actions:

- Expiration
- NoncurrentVersionExpiration
- AbortIncompleteMultipartUpload

Parallel thread processing for bucket life cycles

The Ceph Object Gateway now allows for parallel thread processing of bucket life cycles across multiple Ceph Object Gateway instances. Increasing the number of threads that run in parallel enables the Ceph Object Gateway to process large workloads more efficiently. In addition, the Ceph Object Gateway now uses a numbered sequence for index shard enumeration instead of using in-order numbering.

Optimizing the bucket lifecycle

Optimize the bucket lifecycle for different types of workloads.

PREREQUISITE

- A running Red Hat Ceph Storage cluster.
- Root-level access to all nodes in the storage cluster.

Two options in the Ceph configuration file affect the efficiency of bucket lifecycle processing:

`rgw_lc_max_worker`

Specifies the number of lifecycle worker threads to run in parallel. This enables the simultaneous processing of both bucket and index shards. The default value for this option is 3.

`rgw_lc_max_wp_worker`

Specifies the number of threads in each lifecycle worker thread's work pool. This option helps to accelerate processing for each bucket. The default value for this option is 3.

For a workload with many buckets, for example a workload with thousands of buckets, consider increasing the value of the `rgw_lc_max_worker` option.

For a workload with a smaller number of buckets but with a higher number of objects in each bucket, such as in the hundreds of thousands, consider increasing the value of the `rgw_lc_max_wp_worker` option.

> **Note:** Before increasing the value of either of these options, validate current storage cluster performance and Ceph Object Gateway use. Do not assign a value of 10 or more for either of these options.

1. Increase the number of threads.
   - To increase the number of threads to run in parallel, set the value of `rgw_lc_max_worker` to a value between `3` and `9`.

```
ceph config set client.rgw rgw_lc_max_worker 7
```

- To increase the number of threads in each thread's work pool, set the value of `rgw_lc_max_wp_worker` to a value between `3` and `9`.

```
ceph config set client.rgw rgw_lc_max_wp_worker 7
```

2. Restart the Ceph Object Gateway.
   Restarting allows the changed settings to take effect.
3. Monitor the storage cluster to verify that the increased values do not adversely affect performance.

Policy based data archival and retrieval to S3 compatible platforms

The object lifecycle transition rule allows you to transition an object from one storage class to another S3-compatible platform.

You can migrate data to S3-compatible platforms with the Ceph Object Gateway lifecycle transition policy.

> **Note:** In a multi-site configuration, when the lifecycle transition rule is applied on the first site, to transition objects from one data pool to another in the same storage cluster, then the same rule is valid for the second site, if the second site has the respective data pool created and enabled with `rgw` application.

Transitioning data to Amazon S3 cloud service

You can transition data to a remote cloud service as part of the lifecycle configuration using storage classes to reduce cost and improve manageability. The transition is unidirectional and data cannot be transitioned back from the remote zone. This feature is to enable data transition to multiple cloud providers such as Amazon (S3).

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- An Red Hat Ceph Storage with Ceph Object Gateway installed.
- User credentials for the remote cloud service, Amazon S3.
- Target path created on Amazon S3.
- `s3cmd` installed on the bootstrapped node.
- Amazon AWS configured locally to download data.

Use `cloud-s3` as `tier-type` to configure the remote cloud S3 object store service to which the data needs to be transitioned. These do not need a data pool and are defined in terms of the zonegroup placement targets.

1. From the Cephadm shell, create a user with access key and secret key.

```
radosgw-admin user create --uid=USER_NAME --display-name="DISPLAY_NAME" [--access-
key ACCESS_KEY --secret-key SECRET_KEY]
```

   For example,

```
[ceph: root@host01 /]# radosgw-admin user create --uid=test-user --display-name="test-
user" --access-key a21e86bce636c3aa1 --secret-key cf764951f1fdde5e
{
    "user_id": "test-user",
    "display_name": "test-user",
    "email": "",
    "suspended": 0,
    "max_buckets": 1000,
    "subusers": [],
    "keys": [
        {
            "user": "test-user",
            "access_key": "a21e86bce636c3aa1",
            "secret_key": "cf764951f1fdde5e"
        }
    ],
    "swift_keys": [],
    "caps": [],
    "op_mask": "read, write, delete",
    "default_placement": "",
    "default_storage_class": "",
    "placement_tags": [],
    "bucket_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
    "user_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
    "temp_url_keys": [],
    "type": "rgw",
    "mfa_ids": []
}
```

2. On the bootstrapped node, add a storage class. with the tier type as `cloud-s3`.
   Add one of the following tier types. The tier type options are `cloud-s3` and `cloud-s3-glacier`.

   - Adding the `cloud-s3` tier type.

     > **Important:** After a storage class is created with the **`--tier-type=cloud-s3`** option , it cannot be later modified to any other storage class type.

     ```
     radosgw-admin zonegroup placement add --rgw-zonegroup =ZONE_GROUP_NAME \
                                --placement-id=PLACEMENT_ID \
                                --storage-class =STORAGE_CLASS_NAME \
                                --tier-type=cloud-s3
     ```

     For example,

```
[ceph: root@host01 /]# radosgw-admin zonegroup placement add --rgw-
zonegroup=default \
                                                    --placement-id=default-placement
\
                                                    --storage-class=CLOUDTIER \
                                                    --tier-type=cloud-s3
[
    {
        "key": "default-placement",
        "val": {
            "name": "default-placement",
            "tags": [],
            "storage_classes": [
                "CLOUDTIER",
                "STANDARD"
            ],
            "tier_targets": [
                {
                    "key": "CLOUDTIER",
                    "val": {
                        "tier_type": "cloud-s3",
                        "storage_class": "CLOUDTIER",
                        "retain_head_object": "false",
                        "s3": {
                            "endpoint": "",
                            "access_key": "",
                            "secret": "",
                            "host_style": "path",
                            "target_storage_class": "",
                            "target_path": "",
                            "acl_mappings": [],
                            "multipart_sync_threshold": 33554432,
                            "multipart_min_part_size": 33554432
                        }
                    }
                }
            ]
        }
    }
]
```

- Adding the `cloud-s3-glacier` tier type.

```
radosgw-admin zonegroup placement add --rgw-zonegroup =ZONE_GROUP_NAME \
                            --placement-id=PLACEMENT_ID \
                            --storage-class =CLOUDTIER-GLACIER \
                            --tier-type=cloud-s3-glacier
```

For example,

```
[ceph: root@host01 /]# radosgw-admin zonegroup placement add --rgw-
zonegroup=default \
                                                --placement-id=default-placement
\
                                                --storage-class=CLOUDTIER-
GLACIER \
                                                --tier-type=cloud-s3-glacier

[
    {
                "key": "CLOUDTIER-GLACIER",
                "val": {
                    "tier_type": "cloud-s3-glacier",
                    "storage_class": "CLOUDTIER-GLACIER",
                    "retain_head_object": true,
                    "s3": {
                        "endpoint": "http://s3.us-east-1.amazonaws.com",
                        "access_key": "XXXXXXXXXX",
                        "secret": "YYYYYYYYYY",
                        "region": "us-east-1",
                        "host_style": "path",
                        "target_storage_class": "GLACIER",
                        "target_path": "rgwbucket",
                        "acl_mappings": [],
                        "multipart_sync_threshold": 44432,
                        "multipart_min_part_size": 44432
                    },
                    "allow_read_through": false,
                    "read_through_restore_days": 10,
                    "restore_storage_class": "COLDTIER",
                    "s3-glacier": {
                        "glacier_restore_days": 2,
                        "glacier_restore_tier_type": "Expedited"
                    }
                }
            }

]
```

3. Update the **storage_class**.

> **Note:**
>
> - If the cluster is part of a multi-site setup, run the **period update --commit** command so that the zone group changes are propagated to all the zones in the multi-site.
> - On a multisite environment, the object restore gets replicated when the object is permanently restored.
> - Make sure access_key and secret do not start with a digit.

```
radosgw-admin zonegroup placement modify --rgw-zonegroup ZONE_GROUP_NAME \
                                         --placement-id PLACEMENT_ID \
                                         --storage-class STORAGE_CLASS_NAME  \
                                         --tier-config=endpoint=AWS_ENDPOINT_URL,\

 access_key=AWS_ACCESS_KEY,secret=AWS_SECRET_KEY,\
                                         target_path="TARGET_BUCKET_ON_AWS",\
                                         multipart_sync_threshold=44432,\
                                         multipart_min_part_size=44432,\
                                         retain_head_object=true
                                         region=REGION_NAME
```

The following are mandatory parameters:

access_key

The remote cloud S3 access key used for a specific connection.

secret

The secret key for the remote cloud S3 service.

endpoint

The URL of the remote cloud S3 service endpoint.

region

The remote cloud S3 service region name for AWS.

The following are optional parameters:

target_path

Defines how the target path is created. The target path specifies a prefix to which the source `bucket-name/object-name` is appended. If not specified, the **target_path** created is `rgwx-ZONE_GROUP_NAME-STORAGE_CLASS_NAME-cloud-bucket`.

target_storage_class

Defines the target storage class to which the object transitions. If not specified, the object is transitioned toSTANDARDstorage class.

retain_head_object

Set to`true`to retain the metadata of the object transitioned to cloud. Set to`false`to delete the object after transition. The default setting is`false`.

> **Note:** This option is ignored for current versioned objects.

multipart_sync_threshold

Specifies that objects this size or larger are transitioned to the cloud using multipart upload.

multipart_min_part_size

Specifies the minimum part size to use when transitioning objects using multipart upload.

glacier_restore_days

For use with `cloud-s3-glacier`. Specifies the number of days that the object can be restored on the Glacier or Tape endpoint. The default setting is1(1 day).

glacier_restore_tier_type

For use with `cloud-s3-glacier`. Specifies the restore retrieval type. The options are`Standard`or `Expedited`. The default setting is`Standard`.

For example,

```
[ceph: root@host01 /]# radosgw-admin zonegroup placement modify --rgw-zonegroup
default
                                                                --placement-id
default-placement \
                                                                --storage-class
CLOUDTIER \
                                                                --tier-
config=endpoint=http://10.0.210.010:8080,\

access_key=a21e86bce636c3aa2,secret=cf764951f1fdde5f,\
                                                                target_path="dfqe-
bucket-01",\

multipart_sync_threshold=44432,\

multipart_min_part_size=44432,\

retain_head_object=true
                                                                region= us-east-1

[
    {
        "key": "default-placement",
        "val": {
            "name": "default-placement",
            "tags": [],
            "storage_classes": [
                "CLOUDTIER",
                "STANDARD",
                "cold.test",
                "hot.test"
            ],
            "tier_targets": [
                {
                    "key": "CLOUDTIER",
                    "val": {
                        "tier_type": "cloud-s3",
                        "storage_class": "CLOUDTIER",
                        "retain_head_object": "true",
                        "s3": {
                            "endpoint": "http://10.0.210.010:8080",
                            "access_key": "a21e86bce636c3aa2",
                            "secret": "cf764951f1fdde5f",
                            "region": "",
                            "host_style": "path",
                            "target_storage_class": "",
                            "target_path": "dfqe-bucket-01",
                            "acl_mappings": [],
                            "multipart_sync_threshold": 44432,
                            "multipart_min_part_size": 44432
                        }
                    }
                }
            ]
        }
    }
]
```

4. Restart the Ceph Object Gateway.

```
ceph orch restart CEPH_OBJECT_GATEWAY_SERVICE_NAME
```

For example,

```
[ceph: root@host 01 /]# ceph orch restart rgw.rgw.1

Scheduled to restart rgw.rgw.1.host03.vkfldf on host 'host03'
```

5. Exit the Cephadm shell and configure Amazon S3 on your bootstrapped node, as a root user.

```
s3cmd --configure
```

For example,

```
[root@host01 ~]# s3cmd --configure

Enter new values or accept defaults in brackets with Enter.
Refer to user manual for detailed description of all options.

Access key and Secret key are your identifiers for Amazon S3. Leave them empty for
using the env variables.
Access Key: a21e86bce636c3aa2
Secret Key: cf764951f1fdde5f
Default Region [US]:

Use "s3.amazonaws.com" for S3 Endpoint and not modify it to the target Amazon S3.
S3 Endpoint [s3.amazonaws.com]: 10.0.210.78:80

Use "%(bucket)s.s3.amazonaws.com" to the target Amazon S3. "%(bucket)s" and "%
(location)s" vars can be used
if the target S3 system supports dns based buckets.
DNS-style bucket+hostname:port template for accessing a bucket [%
(bucket)s.s3.amazonaws.com]: 10.0.210.78:80

Encryption password is used to protect your files from reading
by unauthorized persons while in transfer to S3
Encryption password:
Path to GPG program [/usr/bin/gpg]:

When using secure HTTPS protocol all communication with Amazon S3
servers is protected from 3rd party eavesdropping. This method is
slower than plain HTTP, and can only be proxied with Python 2.7 or newer
Use HTTPS protocol [Yes]: No

On some networks all internet access must go through a HTTP proxy.
Try setting it here if you can't connect to S3 directly
HTTP Proxy server name:

New settings:
  Access Key: a21e86bce636c3aa2
  Secret Key: cf764951f1fdde5f
  Default Region: US
  S3 Endpoint: 10.0.210.78:80
  DNS-style bucket+hostname:port template for accessing a bucket: 10.0.210.78:80
  Encryption password:
  Path to GPG program: /usr/bin/gpg
  Use HTTPS protocol: False
  HTTP Proxy server name:
  HTTP Proxy server port: 0

Test access with supplied credentials? [Y/n] Y
Please wait, attempting to list all buckets...
Success. Your access key and secret key worked fine :-)

Now verifying that encryption works...
Not configured. Never mind.

Save settings? [y/N] y
Configuration saved to '/root/.s3cfg'
```

6. Create the S3 bucket.

```
s3cmd mb s3://NAME_OF_THE_BUCKET_FOR_S3
```

For example,

```
[root@host01 ~]# s3cmd mb s3://awstestbucket

Bucket 's3://awstestbucket/' created
```

7. Create your file, input the needed data, and move it to S3 service.

```
s3cmd put FILE_NAME  s3://NAME_OF_THE_BUCKET_ON_S3
```

For example,

```
[root@host01 ~]# s3cmd put test.txt s3://awstestbucket

upload: 'test.txt' -> 's3://awstestbucket/test.txt'  [1 of 1]
 21 of 21   100% in    1s    16.75 B/s  done
```

8. Create the lifecycle configuration transition policy.

```
<LifecycleConfiguration>
  <Rule>
    <ID>RULE_NAME</ID>
    <Filter>
      <Prefix></Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>DAYS</Days>
      <StorageClass>STORAGE_CLASS_NAME</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

> **Note:** If an object that is temporarily or permanently restored has an existing lifecycle Transition or Expiration rule that is set to expire or transition the object before the restore duration ends, the object will be expired or transitioned according to the defined lifecycle rule.

For example,

```
[root@host01 ~]# cat lc_cloud.xml
<LifecycleConfiguration>
  <Rule>
    <ID>Archive all objects</ID>
    <Filter>
      <Prefix></Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>2</Days>
      <StorageClass>CLOUDTIER</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

9. Set the lifecycle configuration transition policy.

```
s3cmd setlifecycle FILE_NAME s3://NAME_OF_THE_BUCKET_FOR_S3
```

For example,

```
[root@host01 ~]#  s3cmd setlifecycle lc_config.xml s3://awstestbucket

s3://awstestbucket/: Lifecycle Policy updated
```

10. Log in to the Cephadm shell.

```
cephadm shell
```

11. Restart the Ceph Object Gateway.

```
ceph orch restart CEPH_OBJECT_GATEWAY_SERVICE_NAME
```

For example,

```
[ceph: root@host 01 /]# ceph orch restart rgw.rgw.1

Scheduled to restart rgw.rgw.1.host03.vkfldf on host `host03
```

AFTER COMPLETING THE TASK

1. On the source cluster, verify if the data has moved to S3 with **`radosgw-admin lc list`** command.

```
[ceph: root@host01 /]# radosgw-admin lc list
[
    {
```

```
        "bucket": ":awstestbucket:552a3adb-39e0-40f6-8c84-00590ed70097.54639.1",
        "started": "Mon, 26 Sep 2022 18:32:07 GMT",
        "status": "COMPLETE"
    }
]
```

2. Verify object transition at cloud endpoint.

```
radosgw-admin bucket list
```

For example,

```
[root@client ~]$ radosgw-admin bucket list
[
    "awstestbucket"
]
```

3. List the objects in the bucket.

```
aws s3api list-objects --bucket awstestbucket --endpoint=AWS_ENDPOINT_URL
```

For example,

```
[root@client ~]$ aws s3api list-objects --bucket awstestbucket --endpoint=http://
10.0.209.002:8080
{
    "Contents": [
        {
            "Key": "awstestbucket/test",
            "LastModified": "2022-08-25T16:14:23.118Z",
            "ETag": "\"378c905939cc4459d249662dfae9fd6f\"",
            "Size": 29,
            "StorageClass": "STANDARD",
            "Owner": {
                "DisplayName": "test-user",
                "ID": "test-user"
            }
        }
    ]
}
```

4. List the contents of the S3 bucket.

```
s3cmd ls s3://awstestbucket
```

For example,

```
[root@host01 ~]# s3cmd ls s3://awstestbucket
2022-08-25 09:57            0  s3://awstestbucket/test.txt
```

5. Verify the file information.

```
s3cmd info s3://awstestbucket/test.txt
```

For example,

```
[root@host01 ~]# s3cmd info s3://awstestbucket/test.txt
s3://awstestbucket/test.txt (object):
   File size: 0
   Last mod:  Mon, 03 Aug 2022 09:57:49 GMT
   MIME type: text/plain
   Storage:   CLOUDTIER
   MD5 sum:   991d2528bb41bb839d1a9ed74b710794
   SSE:       none
   Policy:    none
   CORS:      none
   ACL:       test-user: FULL_CONTROL
   x-amz-meta-s3cmd-attrs: atime:1664790668/ctime:1664790668/gid:0/gname:root/
md5:991d2528bb41bb839d1a9ed74b710794/mode:33188/mtime:1664790668/uid:0/uname:root
```

6. Download data locally from Amazon S3.

a. Configure AWS, using the **`aws configure`** command.
For example,

```
[client@client01 ~]$ aws configure

AWS Access Key ID [****************6VVP]:
AWS Secret Access Key [****************pXqy]:
Default region name [us-east-1]:
Default output format [json]:
```

b. List the contents of the AWS bucket.
For example,

```
[client@client01 ~]$ aws s3 ls s3://dfqe-bucket-01/awstest
PRE awstestbucket/
```

c. Download the data from S3.
For example,

```
[client@client01 ~]$ aws s3 cp s3://dfqe-bucket-01/awstestbucket/test.txt

download: s3://dfqe-bucket-01/awstestbucket/test.txt to ./test.txt
```

Transitioning data to IBM Cloud Object Store (COS) service

You can transition data to an IBM Cloud Object Store (COS) service as part of the lifecycle configuration using storage classes to reduce cost and improve manageability. The transition is unidirectional and data cannot be transitioned back from the remote zone. This feature is to enable data transition to multiple cloud providers such as IBM Cloud Object Store (COS) service.

- The procedure for setting up cloud transition to IBM COS is similar to Transitioning data to Amazon S3 cloud service.

  Important:
  - The following parameters are different in IBM COS:
    - **`target_path`**
    - **`tier-config`**
    - **`access_key`**
    - **`secret_key`**
    - **`region`**
  - The only supported option for `glacier_restore_tier_type` is **`NoTier`**.

Transitioning data to Azure cloud service

You can transition data to a remote cloud service as part of the lifecycle configuration using storage classes to reduce cost and improve manageability. The transition is unidirectional and data cannot be transitioned back from the remote zone. This feature is to enable data transition to multiple cloud providers such as Azure. One of the key differences with the AWS configuration is that you need to configure the multi-cloud gateway (MCG) and use MCG to translate from the S3 protocol to Azure Blob.

Use `cloud-s3` as `tier-type` to configure the remote cloud S3 object store service to which the data needs to be transitioned. These do not need a data pool and are defined in terms of the zonegroup placement targets.

Prerequisites

- A running Red Hat Ceph Storage cluster with Ceph Object Gateway installed.
- User credentials for the remote cloud service, Azure.

- Azure configured locally to download data.
- `s3cmd` installed on the bootstrapped node.
- Azure container for the for MCG namespace created. In this example, it is `mcgnamespace`.

Procedure

1. Create a user with access key and secret key:

   Syntax

   ```
   radosgw-admin user create --uid=USER_NAME --display-name="DISPLAY_NAME" [--access-
   key ACCESS_KEY --secret-key SECRET_KEY]
   ```

   Example

   ```
   [ceph: root@host01 /]# radosgw-admin user create --uid=test-user --display-name="test-
   user" --access-key a21e86bce636c3aa1 --secret-key cf764951f1fdde5e
   {
       "user_id": "test-user",
       "display_name": "test-user",
       "email": "",
       "suspended": 0,
       "max_buckets": 1000,
       "subusers": [],
       "keys": [
           {
               "user": "test-user",
               "access_key": "a21e86bce636c3aa1",
               "secret_key": "cf764951f1fdde5e"
           }
       ],
       "swift_keys": [],
       "caps": [],
       "op_mask": "read, write, delete",
       "default_placement": "",
       "default_storage_class": "",
       "placement_tags": [],
       "bucket_quota": {
           "enabled": false,
           "check_on_raw": false,
           "max_size": -1,
           "max_size_kb": 0,
           "max_objects": -1
       },
       "user_quota": {
           "enabled": false,
           "check_on_raw": false,
           "max_size": -1,
           "max_size_kb": 0,
           "max_objects": -1
       },
       "temp_url_keys": [],
       "type": "rgw",
       "mfa_ids": []
   }
   ```

2. As a root user, configure AWS CLI with the user credentials and create a bucket with default placement:

   Syntax

   ```
   aws s3 --ca-bundle CA_PERMISSION --profile rgw  --endpoint ENDPOINT_URL --region
   default mb s3://BUCKET_NAME
   ```

   Example

   ```
   [root@host01 ~]$ aws s3   --ca-bundle  /etc/pki/ca-trust/source/anchors/myCA.pem --
   profile rgw  --endpoint https://host02.example.com:8043 --region default mb s3://
   transition
   ```

3. Verify that the bucket is using `default-placement` with the placement rule:

   Example

```
[root@host01 ~]# radosgw-admin bucket stats --bucket transition
{
    "bucket": "transition",
    "num_shards": 11,
    "tenant": "",
    "zonegroup": "b29b0e50-1301-4330-99fc-5cdcfc349acf",
    "placement_rule": "default-placement",
    "explicit_placement": {
        "data_pool": "",
        "data_extra_pool": "",
        "index_pool": ""
    },
```

4. Log into the OpenShift Container Platform (OCP) cluster with OpenShift Data Foundation (ODF) deployed:

   Example

```
[root@host01 ~]$ oc project openshift-storage
[root@host01 ~]$ oc get clusterversion
NAME      VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version   4.11.6    True        False         4d1h    Cluster version is 4.11.6

[root@host01 ~]$ oc get storagecluster
NAME                 AGE   PHASE   EXTERNAL   CREATED AT             VERSION
ocs-storagecluster   4d    Ready              2023-06-27T15:23:01Z   4.11.0
```

5. Configure the multi-cloud gateway (MCG) namespace Azure bucket running on an OCP cluster in Azure:

   Syntax

```
noobaa namespacestore create azure-blob az --account-key='ACCOUNT_KEY' --account-
name='ACCOUNT_NAME' --target-blob-container='AZURE_CONTAINER_NAME'
```

   Example

```
[root@host01 ~]$ noobaa namespacestore create azure-blob  az --account-
key='iq3+6hRtt9bQ46QfHKQ0nSm2aP+tyMzdn8dBSRW4XWrFhY+1nwfqEj4hk2q66nmD85E/
o5OrrUqo+AStkKwm9w==' --account-name='transitionrgw' --target-blob-
container='mcgnamespace'
```

6. Create an MCG bucket class pointing to the `namespacestore`:

   Example

```
[root@host01 ~]$ noobaa bucketclass create namespace-bucketclass single aznamespace-
bucket-class --resource az -n openshift-storage
```

7. Create an object bucket claim (OBC) for the transition to cloud:

   Syntax

```
noobaa obc create OBC_NAME --bucketclass aznamespace-bucket-class -n openshift-storage
```

   Example

```
[root@host01 ~]$ noobaa obc create rgwobc --bucketclass aznamespace-bucket-class -n
openshift-storage
```

> **Note:** Use the credentials provided by OBC to configure zonegroup placement on the Ceph Object Gateway.

8. On the bootstrapped node, create a storage class with the tier type as `cloud-s3` on the default placement within the default zonegroup on the previously configured MCG in Azure:

> **Note:** Once a storage class is created with the `--tier-type=cloud-s3` option , it cannot be later modified to any other storage class type.

Syntax

```
radosgw-admin zonegroup placement add --rgw-zonegroup =ZONE_GROUP_NAME \
                            --placement-id=PLACEMENT_ID \
                            --storage-class =STORAGE_CLASS_NAME \
                            --tier-type=cloud-s3
```

Example

```
[ceph: root@host01 /]# radosgw-admin zonegroup placement add --rgw-zonegroup=default \
                                                    --placement-id=default-placement \
                                                    --storage-class=AZURE \
                                                    --tier-type=cloud-s3
[
    {
        "key": "default-placement",
        "val": {
            "name": "default-placement",
            "tags": [],
            "storage_classes": [
                "AZURE",
                "STANDARD"
            ],
            "tier_targets": [
                {
                    "key": "AZURE",
                    "val": {
                        "tier_type": "cloud-s3",
                        "storage_class": "AZURE",
                        "retain_head_object": "false",
                        "s3": {
                            "endpoint": "",
                            "access_key": "",
                            "secret": "",
                            "host_style": "path",
                            "target_storage_class": "",
                            "target_path": "",
                            "acl_mappings": [],
                            "multipart_sync_threshold": 33554432,
                            "multipart_min_part_size": 33554432
                        }
                    }
                }
            ]
        }
    }
]
```

9. Configure the cloud S3 cloud storage class:

   Syntax

```
radosgw-admin zonegroup placement modify --rgw-zonegroup ZONE_GROUP_NAME \
                                         --placement-id PLACEMENT_ID \
                                         --storage-class STORAGE_CLASS_NAME  \
                                         --tier-config=endpoint=ENDPOINT_URL,\
                                         access_key=ACCESS_KEY,secret=SECRET_KEY,\
                                         target_path="TARGET_BUCKET_ON",\
                                         multipart_sync_threshold=44432,\
                                         multipart_min_part_size=44432,\
                                         retain_head_object=true
                                         region=REGION_NAME
```

   Important: Setting the `retain_head_object` parameter to `true` retains the metadata or the head of the object to list the objects that are transitioned.

10. Example

```
[ceph: root@host01 /]# radosgw-admin zonegroup placement modify --rgw-zonegroup
default
                                                                --placement-id
default-placement \
                                                                --storage-class AZURE
```

```
\
                                                                --tier-
config=endpoint="https://s3-openshift-storage.apps.ocp410.0e73azopenshift.com",\

access_key=a21e86bce636c3aa2,secret=cf764951f1fdde5f,\
                                                                target_path="dfqe-
bucket-01",\

multipart_sync_threshold=44432,\

multipart_min_part_size=44432,\

retain_head_object=true
                                                                region= us-east-1

[
    {
        "key": "default-placement",
        "val": {
            "name": "default-placement",
            "tags": [],
            "storage_classes": [
                "AZURE",
                "STANDARD",
                "cold.test",
                "hot.test"
            ],
            "tier_targets": [
                {
                    "key": "AZURE",
                    "val": {
                        "tier_type": "cloud-s3",
                        "storage_class": "AZURE",
                        "retain_head_object": "true",
                        "s3": {
                            "endpoint": "https://s3-openshift-
storage.apps.ocp410.0e73azopenshift.com",
                            "access_key": "a21e86bce636c3aa2",
                            "secret": "cf764951f1fdde5f",
                            "region": "",
                            "host_style": "path",
                            "target_storage_class": "",
                            "target_path": "dfqe-bucket-01",
                            "acl_mappings": [],
                            "multipart_sync_threshold": 44432,
                            "multipart_min_part_size": 44432
                        }
                    }
                }
            ]
        }
    }
]
```

11. Restart the Ceph Object Gateway:

    Syntax

    ```
    ceph orch restart CEPH_OBJECT_GATEWAY_SERVICE_NAME
    ```

    Example

    ```
    [ceph: root@host 01 /]# ceph orch restart client.rgw.objectgwhttps.host02.udyllp

    Scheduled to restart client.rgw.objectgwhttps.host02.udyllp on host 'host02
    ```

12. Create the lifecycle configuration transition policy for the bucket created previously. In this example, the bucket is `transition`:

    Syntax

    ```
    cat transition.json
    {
        "Rules": [
            {
                "Filter": {
    ```

```
                "Prefix": ""
            },
            "Status": "Enabled",
            "Transitions": [
                {
                    "Days": 30,
                    "StorageClass": "STORAGE_CLASS"
                }
            ],
            "ID": "TRANSITION_ID"
        }
    ]
        }
```

Note: All the objects in the bucket older than 30 days are transferred to the cloud storage class called `AZURE`.

Example

```
[root@host01 ~]$ cat transition.json
{
    "Rules": [
        {
            "Filter": {
                "Prefix": ""
            },
            "Status": "Enabled",
            "Transitions": [
                {
                    "Days": 30,
                    "StorageClass": "AZURE"
                }
            ],
            "ID": "Transition Objects in bucket to AZURE Blob after 30 days"
        }
    ]
        }
```

13. Apply the bucket lifecycle configuration using AWS CLI:

    Syntax

    ```
     aws s3api --ca-bundle CA_PERMISSION --profile rgw --endpoint ENDPOINT_URL --region
    default put-bucket-lifecycle-configuration --lifecycle-configuration  file://
    BUCKET.json --bucket BUCKET_NAME
    ```

    Example

    ```
    [root@host01 ~]$ aws s3api --ca-bundle  /etc/pki/ca-trust/source/anchors/myCA.pem --
    profile rgw  --endpoint https://host02.example.com:8043 --region default put-bucket-
    lifecycle-configuration --lifecycle-configuration  file://transition.json --bucket
    transition
    ```

14. Optional: Get the lifecycle configuration:

    Syntax

    ```
     aws s3api --ca-bundle CA_PERMISSION --profile rgw --endpoint ENDPOINT_URL--region
    default get-bucket-lifecycle-configuration --lifecycle-configuration  file://
    BUCKET.json --bucket BUCKET_NAME
    ```

    Example

    ```
    [root@host01 ~]$ aws s3api   --ca-bundle  /etc/pki/ca-trust/source/anchors/myCA.pem --
    profile rgw  --endpoint https://host02.example.com:8043 --region default get-bucket-
    lifecycle-configuration --bucket transition
    {
        "Rules": [
            {
                "ID": "Transition Objects in bucket to AZURE Blob after 30 days",
    ```

```
            "Prefix": "",
            "Status": "Enabled",
            "Transitions": [
                {
                    "Days": 30,
                    "StorageClass": "AZURE"
                }
            ]
        }
    ]
}
```

15. Optional: Get the lifecycle configuration with the **`radosgw-admin lc list`** command:

```
[root@host 01 ~]# radosgw-admin lc list
[
    {
        "bucket": ":transition:d9c4f708-5598-4c44-9d36-849552a08c4d.169377.1",
        "started": "Thu, 01 Jan 1970 00:00:00 GMT",
        "status": "UNINITIAL"
    }
]
```

> **Note:** The `UNINITAL` status implies that the lifecycle configuration is not processed. It moves to `COMPLETED` state after the transition process is complete.

16. Log in to `cephadm shell`:

```
[root@host 01 ~]# cephadm shell
```

17. Restart the Ceph Object Gateway daemon:

    Syntax

```
ceph orch daemon CEPH_OBJECT_GATEWAY_DAEMON_NAME
```

    Example

```
[ceph: root@host 01 /]# ceph orch daemon restart rgw.objectgwhttps.host02.udyllp
[ceph: root@host 01 /]# ceph orch daemon restart rgw.objectgw.host02.afwvyq
[ceph: root@host 01 /]# ceph orch daemon restart rgw.objectgw.host05.ucpsrr
```

18. Migrate data from the source cluster to Azure:

    Example

```
[root@host 01 ~]# for i in 1 2 3 4 5
do
aws s3 --ca-bundle  /etc/pki/ca-trust/source/anchors/myCA.pem --profile rgw --endpoint
https://host02.example.com:8043 --region default cp /etc/hosts s3://transition/
transition$i
done
```

19. Verify transition of data:

    Example

```
[root@host 01 ~]# aws s3 --ca-bundle  /etc/pki/ca-trust/source/anchors/myCA.pem --
profile rgw  --endpoint https://host02.example.com:8043 --region default ls s3://
transition
2023-06-30 10:24:01       3847 transition1
2023-06-30 10:24:04       3847 transition2
2023-06-30 10:24:07       3847 transition3
2023-06-30 10:24:09       3847 transition4
2023-06-30 10:24:13       3847 transition5
```

20. Verify if the data has moved to Azure with `rados ls` command:

    Example

```
[root@host 01 ~]# rados ls -p default.rgw.buckets.data | grep transition
d9c4f708-5598-4c44-9d36-849552a08c4d.169377.1_transition1
d9c4f708-5598-4c44-9d36-849552a08c4d.169377.1_transition4
d9c4f708-5598-4c44-9d36-849552a08c4d.169377.1_transition2
d9c4f708-5598-4c44-9d36-849552a08c4d.169377.1_transition3
d9c4f708-5598-4c44-9d36-849552a08c4d.169377.1_transition5
```

21. If the data is not transitioned, you can run the `lc process` command:

    Example

    ```
    [root@host 01 ~]# radosgw-admin lc process
    ```

    This will force the lifecycle process to start and evaluates all the bucket lifecycle policies configured. It then starts the transition of data wherever needed.

Verification

1. Run the `radosgw-admin lc list` command to verify the completion of the transition:

   Example

   ```
   [root@host 01 ~]# radosgw-admin lc list
   [
       {
           "bucket": ":transition:d9c4f708-5598-4c44-9d36-849552a08c4d.170017.5",
           "started": "Mon, 30 Jun 2023-06-30 16:52:56 GMT",
           "status": "COMPLETE"
       }
   ]
   ```

2. List the objects in the bucket:

   Example

   ```
   [root@host01 ~]$ aws s3api list-objects --bucket awstestbucket --endpoint=http://
   10.0.209.002:8080
   {
       "Contents": [
           {
               "Key": "awstestbucket/test",
               "LastModified": "2023-06-25T16:14:23.118Z",
               "ETag": "\"378c905939cc4459d249662dfae9fd6f\"",
               "Size": 29,
               "StorageClass": "STANDARD",
               "Owner": {
                   "DisplayName": "test-user",
                   "ID": "test-user"
               }
           }
       ]
   }
   ```

3. List the objects on the cluster:

   Example

   ```
   [root@host01 ~]$ aws s3 --ca-bundle  /etc/pki/ca-trust/source/anchors/myCA.pem --
   profile rgw  --endpoint https://host02.example.com:8043 --region default ls s3://
   transition
   2023-06-30 17:52:56          0 transition1
   2023-06-30 17:51:59          0 transition2
   2023-06-30 17:51:59          0 transition3
   2023-06-30 17:51:58          0 transition4
   2023-06-30 17:51:59          0 transition5
   ```

   The objects are `0` in size. You can list the objects, but cannot copy them since they are transitioned to Azure.

4. Check the head of the object using the S3 API:

   Example

```
[root@host01 ~]$ aws s3api --ca-bundle  /etc/pki/ca-trust/source/anchors/myCA.pem --
profile rgw  --endpoint https://host02.example.com:8043 --region default head-object
--key transition1 --bucket transition
{
    "AcceptRanges": "bytes",
    "LastModified": "2023-06-31T16:52:56+00:00",
    "ContentLength": 0,
    "ETag": "\"46ecb42fd0def0e42f85922d62d06766\"",
    "ContentType": "binary/octet-stream",
    "Metadata": {},
    "StorageClass": "CLOUDTIER"
}
```

You can see that the storage class has changed from `STANDARD` to `CLOUDTIER`.

Restoring objects from S3 cloud-tier storage

You can restore the objects from the cloud back into Ceph Object Gateway cluster using S3 Restore API on the transitioned object.

PREREQUISITE

Ensure the following prerequisites are met before restoring objects from S3 cloud-tier storage.

- Target path created on Amazon S3.
- **`retain_head_object`** should be set to`true`while configuring cloudtier storage-class to be able to restore from cloud service.
- `s3cmd` installed on the bootstrapped node.
- Ensure user credentials provided for that cloud-tier storage class remain valid for the `restore` function to work properly.
- For `cloud-s3-glacier` tier types, be sure that the **`glacier_restore_days`** and **`glacier_restore_tier_type`** options are correctly set. For more information, see “Transitioning data to Amazon S3 cloud service” on page 270.

By default, objects are restored to STANDARD storage-class. However, you can configure the storage class to which the objects need to be restored to by setting the `restore_storage_class` option.

> **Note:**
> If the null version of an object is not the latest version, do not specify `version-id null` when issuing a `restore-object` request.
>
> For example, suppose an object named `object1` is uploaded to a bucket called `testbucket1`, transitioned to the cloud, and then restored. Later, versioning is enabled on `testbucket1`, and a new version of `object1` is uploaded. At this point, the original (null) version is no longer the latest.
>
> In this case, when restoring `object1`, you should omit the `--version-id null` parameter from the `restore-object`command.

Restore an object, using one of the following procedures.

- Restore cloud transitioned objects by using the S3 **`restore-object`** command.

```
aws s3api restore-object
          --bucket <value>
          --key <value>
          [--version-id <value>]
          --restore-request (structure)
{
            // for temporary restore
            "Days": integer, // "Days" is optional and if not provided, the object will
be restored permanently.
          }
```

> **Note:** The restoration period of the temporary copies can be updated by reissuing the request with a new period.

The following example demonstrates how to restore the object `doc1.rtf` from the specified version for a period of 10 days.

```
aws s3api restore-object
--bucket bucket1
--key doc1.rtf [--version-id
3sL4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY+MTRCxf3vjVBH40Nr8X8gdRQBpUMLUo]
--restore-request "Days": 10
```

The following example demonstrates how to restore the object `doc2.rtf` permanently, where it is treated as a regular object.

```
aws s3api restore-object  --bucket bucket1 --key doc2.rtf --restore-request {}
```

> **Note:** If a restore request for an object is already issued or in progress, the `s3api` restore request with the `--debug` flag displays a `RestoreAlreadyInProgress` message.

- Restore and read the transitioned objects by using the `read-through` feature.

  > **Note:** The object copy restored via `read-through` is temporary and is retained only for the duration of `read_through_restore_days`.

  Ensure that the following parameters are set:

  - `"Allow_read_through": "enable"`
  - `"read_through_restore_days": 10`

```
--storage-class CLOUDTIER
--tier-config=endpoint=http://XX.XX.XX.XX:YY,
access_key=<access_key>,secret=<secret>,
retain_head_object=true,
restore_storage_class=COLDTIER,
allow_read_through=true,
read_through_restore_days=10
```

Following is an example to restore a object using the S3 Get Object when read-through is enabled.

```
aws s3api get-object  --bucket bucket1 --key doc3.rtf
```

Following are the examples of tier configuration.

Example for restoring an object from `cloud-s3`tier-type

```
tier-type = cloud-s3
tier-config =
{
"access_key": xxx,
"secret": xxx,
"endpoint": http://10.0.210.010:8080
“,
"region": “”,
"host_style": “path”,
"acls": [ { "type": “id”,
"source_id": “”,
"dest_id": “” } ... ],
"target_path": dfqe-bucket-01,
"target_storage_class": “”,
"multipart_sync_threshold": 44432,
"multipart_min_part_size": 44432,
"retain_head_object": “true”,
“Allow_read_through”: “enable”,
“read_through_restore_days”: 10
}
```

Note: The transition and restore logic works if the target storage class matches the actual storage class configured in the cloud endpoint that can archive objects.

Example for restoring an object from `cloud-s3-glacier` tier-type

```
tier-type = cloud-s3-glacier
tier-config =
{
"access_key": xxx,
"secret": xxx,
"endpoint": http://s3.us-east-1.amazonaws.com
“,
"region": “us-east-1”,
"host_style": “path”,
"acls": [ { "type": “id”,
"source_id": “”,
"dest_id": “” } ... ],
"target_path": rgwbucket,
"target_storage_class": “GLACIER”,
"multipart_sync_threshold": 44432,
"multipart_min_part_size": 44432,
"retain_head_object": “true”,
“Allow_read_through”: “enable”,
“read_through_restore_days”: 10
"restore_storage_class": "COLDTIER",
"s3-glacier": {
    "glacier_restore_days": 2,
    "glacier_restore_tier_type": "Expedited"
}
```

AFTER COMPLETING THE TASK

Verify the status of the restore by running an S3 head-object request with the specified parameters. Check the status of the restore using the S3 `head-object` request.

For example,

```
[root@host01 ~]$ aws s3api --ca-bundle
/etc/pki/ca-trust/source/anchors/myCA.pem --profile rgw
--endpoint https://host02.example.com:8043 --region default head-object
--key transition1 --bucket transition
```

Using the radosgw-admin cli for cloud restore operations

Use the **`radosgw-admin`** cli to check the cloud-restore status for a specific object or to list all objects in a bucket that are restored or in progress.

PREREQUISITE

- Administrator privileges to run radosgw-admin commands.
- radosgw-admin tool is installed and available in your $PATH.
- Target bucket exists in the cluster.
- Restore request is initiated for the object (for status checks).

.

1. To display detailed restore status for a specific object in the bucket.

```
radosgw-admin restore status --bucket <bucket-name> --object <object-name>
```

```
{
    "name": "new",
    "RestoreExpiryDate": "2025-08-21T13:58:46.970540Z",
    "RestoreStatus": "CloudRestored",
    "RestoreType": "Temporary",
    "RestoreTime": "2025-08-21T13:36:39.618810Z"
}
```

Following are the parameter descriptions:

--bucket

Name of the bucket.

--object

Name of the object whose restore status is queried.

2. To list objects in the bucket that are restored or are in the process of being restored.

```
radosgw-admin restore list --bucket <bucket-name>
```

Following is a sample output (no filter):

```
{
    "file": "CloudRestored",
    "new": "CloudRestored"
}
```

Following is a sample output (with filter):

```
radosgw-admin  restore list --bucket <bucket name> --restore-status=CloudRestored
 {
  "file": "CloudRestored"
  }
```

Following are the parameter descriptions:

--bucket

Name of the bucket.

--restore-status=<status>

Filter objects by restore status: `RestoreAlreadyInProgress`, `CloudRestored`, or `RestoreFailed`.

> **Note:** `--restore-status=<status>` is an optional parameter.

3. To display the count of restore operations in the **`bucket stats`** command.

```
radosgw-admin bucket stats --bucket <bucket-name> --show-restore-stats
```

Following is the sample output:

```
{
  "restore_stats": {
    "restore_completed_count": 0,
    "restore_in_progress_count": 0,
    "restore_failed_count": 0
  }
}
```

Following are the parameter descriptions:

--bucket

Name of the bucket to query restore statistics.

--show-restore-stats

Displays restore statistics. Must be set explicitly, statistics are not shown by default.

Viewing Ceph Object Gateway per-user and per-bucket usage statistics (Technology Preview)

Use the Ceph Object Gateway performance counters to view per-user and per-bucket usage statistics, such as total bytes used and number of objects.Performance counters are available through the Ceph admin socket and use a cache to provide accurate, low-latency metrics with minimal overhead.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Ceph Object Gateway is installed and running.
- Object gateway login credentials are configured.
- At least one user and one bucket exist and are active (not suspended).
- Configure the following parameters to the Ceph Object Gateway service and then restart the `rgw` service to apply the changes, by using the **`ceph orch restart rgw`** command.

> **Note:** This feature is disabled by default. Configuring these parameters enables the per-user and per-bucket view feature.

```
ceph config set client.rgw rgw_enable_usage_perf_counters true
ceph config set client.rgw rgw_usage_cache_path /tmp/usage_cache.mdb
ceph config set client.rgw rgw_usage_cache_max_size 1073741824
```

> **Important:** Technology Preview features are not supported with production service level agreements (SLAs), might not be functionally complete, and does not recommend using them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

1. Create a user.

   ```
   radosgw-admin user create
   ```

   For example,

```
radosgw-admin user create \
  --uid=testuser \
  --display-name="Test User" \
  --access-key=test \
  --secret-key=test
```

2. Configure a S3 client.
For example,

```
[default]
access_key = test
secret_key = test
host_base = localhost:8000
host_bucket = localhost:8000
use_https = False
signature_v2 = True
```

3. Create buckets and upload objects.
For example,

```
[default]
access_key = test
secret_key = test
host_base = localhost:8000
host_bucket = localhost:8000
use_https = False
signature_v2 = True
```

4. View usage counters through the admin socket.

> **Note:** Counter statistics are not real time. After uploading objects, counters may take up to 40 minutes to reflect changes.

```
SOCKET=out/radosgw.8000.asok
# ceph --admin-daemon $SOCKET perf dump | grep -A 20 rgw_usage
```

Example counters include:

- `rgw_usage` – Global counters
- `rgw_usage_cache` – Cache statistics
- `rgw_user_testuser` – User-specific statistics
- `rgw_bucket_bucket1` – Bucket-specific statistics

Example output for **`rgw_user_testuser`**

```
"used_bytes": 15728640,
"num_objects": 3
```

Example output for **`rgw_bucket_bucket1`**

```
"used_bytes": 5242880,
"num_objects": 2
```

5. Verify cache behavior.
   a. List a bucket multiple times.

```
s3cmd ls s3://bucket1
s3cmd ls s3://bucket1
s3cmd ls s3://bucket1
```

   b. Check the cache hit counter.

```
ceph --admin-daemon $SOCKET perf dump | grep cache_hit
```

Result

You can now view `per-user` and `per-bucket usage` statistics for Ceph Object Gateway, confirm that usage counters are active, and verify cache behavior.

Testing and validation

As a storage administrator, you can do basic functionality testing to verify that the Ceph Object Gateway environment is working as expected. You can use the REST interfaces by creating an initial Ceph Object Gateway user for the S3 interface, and then create a subuser for the Swift interface.

Prerequisites

- A healthy running Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway software.

Create an S3 user

To test the gateway, create an S3 user and grant the user access. The **`man radosgw-admin`** command provides information on additional command options.

> **Note:** In a multi-site deployment, always create a user on a host in the master zone of the master zone group.

Prerequisites

- `root` or `sudo` access
- Ceph Object Gateway installed

Procedure

1. Create an S3 user:

   Syntax

   ```
   radosgw-admin user create --uid=name --display-name="USER_NAME"
   ```

   Replace *name* with the name of the S3 user:

   Example

   ```
   [root@host01 ~]# radosgw-admin user create --uid="testuser" --display-name="Jane Doe"
   {
       "user_id": "testuser",
       "display_name": "Jane Doe",
       "email": "",
       "suspended": 0,
       "max_buckets": 1000,
       "auid": 0,
       "subusers": [],
       "keys": [
           {
               "user": "testuser",
               "access_key": "CEP28KDIQXBKU4M15PDC",
               "secret_key": "MARoio8HFc8JxhEilES3dKFVj8tV3NOOYymihTLO"
           }
       ],
       "swift_keys": [],
       "caps": [],
       "op_mask": "read, write, delete",
       "default_placement": "",
   ```

```
    "placement_tags": [],
    "bucket_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
    "user_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
    "temp_url_keys": [],
    "type": "rgw"
}
```

2. Verify the output to ensure that the values of `access_key` and `secret_key` do not include a JSON escape character (\). These values are needed for access validation, but certain clients cannot handle if the values include JSON escape characters. To fix this problem, perform one of the following actions:
   - Remove the JSON escape character.
   - Encapsulate the string in quotes.
   - Regenerate the key and ensure that it does not include a JSON escape character.
   - Specify the key and secret manually.

   Do not remove the forward slash / because it is a valid character.

Create a Swift user

To test the Swift interface, create a Swift subuser. Creating a Swift user is a two-step process. The first step is to create the user. The second step is to create the secret key.

> **Note:** In a multi-site deployment, always create a user on a host in the master zone of the master zone group.

Prerequisites

- A running, and healthy Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway.
- Root-level access to the Ceph Object Gateway node.

Procedure

1. Create the Swift user:

   Syntax

   ```
   radosgw-admin subuser create --uid=NAME --subuser=NAME:swift --access=full
   ```

   Replace `NAME` with the Swift user name, for example:

   Example

   ```
   [root@host01 ~]# radosgw-admin subuser create --uid=testuser --subuser=testuser:swift
   --access=full
   {
       "user_id": "testuser",
       "display_name": "First User",
   ```

```
    "email": "",
    "suspended": 0,
    "max_buckets": 1000,
    "auid": 0,
    "subusers": [
        {
            "id": "testuser:swift",
            "permissions": "full-control"
        }
    ],
    "keys": [
        {
            "user": "testuser",
            "access_key": "O8JDE41XMI74O185EHKD",
            "secret_key": "i4Au2yxG5wtr1JK01mI8kjJPM93HNAoVWOSTdJd6"
        }
    ],
    "swift_keys": [
        {
            "user": "testuser:swift",
            "secret_key": "13TLtdEW7bCqgttQgPzxFxziu0AgabtOc6vM8DLA"
        }
    ],
    "caps": [],
    "op_mask": "read, write, delete",
    "default_placement": "",
    "placement_tags": [],
    "bucket_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
    "user_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
    "temp_url_keys": [],
    "type": "rgw"
}
```

2. Create the secret key:

   Syntax

   ```
   radosgw-admin key create --subuser=NAME:swift --key-type=swift --gen-secret
   ```

   Replace `NAME` with the Swift user name, for example:

   Example

   ```
   [root@host01 ~]# radosgw-admin key create --subuser=testuser:swift --key-type=swift --
   gen-secret
   {
       "user_id": "testuser",
       "display_name": "First User",
       "email": "",
       "suspended": 0,
       "max_buckets": 1000,
       "auid": 0,
       "subusers": [
           {
               "id": "testuser:swift",
               "permissions": "full-control"
           }
       ],
       "keys": [
           {
               "user": "testuser",
               "access_key": "O8JDE41XMI74O185EHKD",
               "secret_key": "i4Au2yxG5wtr1JK01mI8kjJPM93HNAoVWOSTdJd6"
           }
       ],
   ```

```
    "swift_keys": [
        {
            "user": "testuser:swift",
            "secret_key": "a4ioT4jEP653CDcdU8p4OuhruwABBRZmyNUbnSSt"
        }
    ],
    "caps": [],
    "op_mask": "read, write, delete",
    "default_placement": "",
    "placement_tags": [],
    "bucket_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
    "user_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
    "temp_url_keys": [],
    "type": "rgw"
}
```

Test S3 access

You need to write and run a Python test script for verifying S3 access. The S3 access test script will connect to the `radosgw`, create a new bucket, and list all buckets. The values for `aws_access_key_id` and `aws_secret_access_key` are taken from the values of `access_key` and `secret_key` returned by the **`radosgw_admin`** command.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the nodes.

Procedure

1. Enable high availability repository.
2. Install the `python3-boto3` package:

```
dnf install python3-boto3
```

3. Create the Python script:

```
vi s3test.py
```

4. Add the following contents to the file:

   Syntax

```
import boto3

endpoint = "" # enter the endpoint _URL_ along with the port "http://URL:PORT"

access_key = 'ACCESS'
secret_key = 'SECRET'

s3 = boto3.client(
        's3',
        endpoint_url=endpoint,
        aws_access_key_id=access_key,
        aws_secret_access_key=secret_key
```

```
        )

s3.create_bucket(Bucket='my-new-bucket')

response = s3.list_buckets()
for bucket in response['Buckets']:
    print("{name}\t{created}".format(
                name = bucket['Name'],
                created = bucket['CreationDate']
))
```

- Replace `endpoint` with the URL of the host where you have configured the gateway service. That is, the `gateway host`. Ensure that the `host` setting resolves with DNS. Replace `PORT` with the port number of the gateway.
- Replace `ACCESS` and `SECRET` with the `access_key` and `secret_key` values

5. Run the script:

```
python3 s3test.py
```

The output will be something like the following:

```
my-new-bucket 2022-05-31T17:09:10.000Z
```

Test Swift access

Swift access can be verified via the `swift` command line client. The **`man swift`** command provides more information on available command line options.

To install the `swift` client, run the following command:

```
sudo yum install python-setuptools
sudo easy_install pip
sudo pip install --upgrade setuptools
sudo pip install --upgrade python-swiftclient
```

To test swift access, run the following command:

Syntax

```
# swift -A http://IP_ADDRESS:PORT/auth/1.0 -U testuser:swift -K 'SWIFT_SECRET_KEY' list
```

Replace `IP_ADDRESS` with the public IP address of the gateway server and `SWIFT_SECRET_KEY` with its value from the output of the `radosgw-admin key create` command issued for the `swift` user. Replace *PORT* with the port number you are using with Beast. If you do not replace the port, it will default to port `80`.

For example:

```
swift -A http://10.10.143.116:80/auth/1.0 -U testuser:swift -K
'244+fz2gSqoHwR3lYtSbIyomyPHf3i7rgSJrF/IA' list
```

The output should be:

```
my-new-bucket
```

Configuration reference

As a storage administrator, you can set various options for the Ceph Object Gateway. These options contain default values. If you do not specify each option, then the default value is set automatically.

To set specific values for these options, update the configuration database by using the **`ceph config set client.rgw OPTION VALUE`** command.

General settings

Use this information to get general settings for configuring Ceph Object Gateway.

Ceph Object Gateway general configuration settings

Name	Description	Type	Default
`rgw_data`	Sets the location of the data files for Ceph Object Gateway.	String	`/var/lib/ceph/radosgw/$cluster-$id`
`rgw_enable_apis`	Enables the specified APIs.	String	`s3, s3website, swift, swift_auth, admin, sts, iam, notifications`
`rgw_cache_enabled`	Whether the Ceph Object Gateway cache is enabled.	Boolean	`true`
`rgw_cache_lru_size`	The number of entries in the Ceph Object Gateway cache.	Integer	`10000`
`rgw_socket_path`	The socket path for the domain socket. `FastCgiExternalServer` uses this socket. If you do not specify a socket path, Ceph Object Gateway will not run as an external server. The path you specify here must be the same as the path specified in the `rgw.conf` file.	String	N/A
`rgw_host`	The host for the Ceph Object Gateway instance. Can be an IP address or a hostname.	String	`0.0.0.0`
`rgw_port`	Port the instance listens for requests. If not specified, Ceph Object Gateway runs external FastCGI.	String	None
`rgw_dns_name`	The DNS name of the served domain. See also the `hostnames` setting within zone groups.	String	None
`rgw_script_uri`	The alternative value for the `SCRIPT_URI` if not set in the request.	String	None
`rgw_request_uri`	The alternative value for the `REQUEST_URI` if not set in the request.	String	None
`rgw_print_continue`	Enable `100-continue` if it is operational.	Boolean	`true`

Name	Description	Type	Default
rgw_remote_addr_param	The remote address parameter. For example, the HTTP field containing the remote address, or the X-Forwarded-For address if a reverse proxy is operational.	String	REMOTE_ADDR
rgw_op_thread_timeout	The timeout in seconds for open threads.	Integer	600
rgw_op_thread_suicide_timeout	The timeout in seconds before a Ceph Object Gateway process dies. Disabled if set to 0.	Integer	0
rgw_thread_pool_size	The size of the thread pool.	Integer	512 threads.
rgw_num_control_oids	The number of notification objects used for cache synchronization between different rgw instances.	Integer	8
rgw_init_timeout	The number of seconds before Ceph Object Gateway gives up on initialization.	Integer	30
rgw_mime_types_file	The path and location of the MIME types. Used for Swift auto-detection of object types.	String	/etc/mime.types
rgw_gc_max_objs	The maximum number of objects that may be handled by garbage collection in one garbage collection processing cycle.	Integer	32
rgw_gc_obj_min_wait	The minimum wait time before the object may be removed and handled by garbage collection processing.	Integer	2 * 3600
rgw_gc_processor_max_time	The maximum time between the beginning of two consecutive garbage collection processing cycles.	Integer	3600
rgw_gc_processor_period	The cycle time for garbage collection processing.	Integer	3600
rgw_s3 success_create_obj_status	The alternate success status response for create-obj.	Integer	0
rgw_resolve_cname	Whether rgw should use the DNS CNAME record of the request hostname field (if hostname is not equal to rgw_dns name).	Boolean	false

Name	Description	Type	Default
rgw_object_stripe_size	The size of an object stripe for Ceph Object Gateway objects.	Integer	4 << 20
rgw_extended_http_attrs	Add a new set of attributes that could be set on an object. These extra attributes can be set through HTTP header fields when putting the objects. If set, these attributes will return as HTTP fields when doing GET/HEAD on the object.	String	None. For example: "content_foo, content_bar"
rgw_exit_timeout_secs	Number of seconds to wait for a process before exiting unconditionally.	Integer	120
rgw_get_obj_window_size	The window size in bytes for a single object request.	Integer	16 << 20
rgw_get_obj_max_req_size	The maximum request size of a single get operation sent to the Ceph Storage Cluster.	Integer	4 << 20
rgw_relaxed_s3_bucket_names	Enables relaxed S3 bucket names rules for zone group buckets.	Boolean	false
rgw_list buckets_max_chunk	The maximum number of buckets to retrieve in a single operation when listing user buckets.	Integer	1000
rgw_override_bucket_index_max_shards	The number of shards for the bucket index object. A value of 0 indicates there is no sharding. Red Hat does not recommend setting a value too large (for example, 1000) as it increases the cost for bucket listing. This variable should be set in the [client] or the [global] section so it is automatically applied to radosgw-admin commands.	Integer	0
rgw_curl_wait_timeout_ms	The timeout in milliseconds for certain curl calls.	Integer	1000
rgw_copy_obj_progress	Enables output of object progress during long copy operations.	Boolean	true
rgw_copy_obj_progress_every_bytes	The minimum bytes between copy progress output.	Integer	1024 * 1024
rgw_admin_entry	The entry point for an admin request URL.	String	admin

Name	Description	Type	Default
`rgw_content_length_compat`	Enable compatibility handling of FCGI requests with both CONTENT_LENGTH AND HTTP_CONTENT_LENGTH set.	Boolean	`false`
`rgw_bucket_default_quota_max_objects`	The default maximum number of objects per bucket. This value is set on new users if no other quota is specified. It has no effect on existing users. This variable should be set in the `[client]` or the `[global]` section so it is automatically applied to `radosgw-admin` commands.	Integer	`-1`
`rgw_bucket_quota_ttl`	The amount of time in seconds cached quota information is trusted. After this timeout, the quota information will be re-fetched from the cluster.	Integer	600
`rgw_user_quota_bucket_sync_interval`	The amount of time in seconds bucket quota information is accumulated before syncing to the cluster. During this time, other RGW instances will not see the changes in bucket quota stats from operations on this instance.	Integer	180
`rgw_user_quota_sync_interval`	The amount of time in seconds user quota information is accumulated before syncing to the cluster. During this time, other RGW instances will not see the changes in user quota stats from operations on this instance.	Integer	3600 * 24
`log_meta`	A zone parameter to determine whether or not the gateway logs the metadata operations.	Boolean	`false`
`log_data`	A zone parameter to determine whether or not the gateway logs the data operations.	Boolean	`false`

Name	Description	Type	Default
`sync_from_all`	A `radosgw-admin` command to set or unset whether zone syncs from all zonegroup peers.	Boolean	`false`
`rgw_curl_tcp_keepalive`	Enables TCP keepalive on the HTTP client sockets managed by libcurl. This does not apply to connections received by the HTTP frontend, but only to HTTP requests sent by radosgw. Examples include requests to Keystone for authentication, sync requests from multisite, and requests to key management servers for SSE.	Boolean	`false`

About pools

Red Hat Ceph Storage zones map to a series of Ceph Storage Cluster pools.

If the user key for the Ceph Object Gateway contains write capabilities, the gateway has the ability to create pools automatically. This is convenient for getting started. However, the Ceph Object Storage Cluster uses the placement group default values unless they were set in the Ceph configuration file. Additionally, Ceph will use the default CRUSH hierarchy. These settings are **NOT** ideal for production systems.

The default pools for the Ceph Object Gateway's default zone include:

- `.rgw.root`
- `.default.rgw.control`
- `.default.rgw.meta`
- `.default.rgw.log`
- `.default.rgw.buckets.index`
- `.default.rgw.buckets.data`
- `.default.rgw.buckets.non-ec`

The Ceph Object Gateway creates pools on a per zone basis. If you create the pools manually, prepend the zone name. The system pools store objects related to, for example, system control, logging, and user information. By convention, these pool names have the zone name prepended to the pool name.

- `.<zone-name>.rgw.control`: The control pool.
- `.<zone-name>.log`: The log pool contains logs of all bucket/container and object actions, such as create, read, update, and delete.
- `.<zone-name>.rgw.buckets.index`: This pool stores the index of the buckets.
- `.<zone-name>.rgw.buckets.data`: This pool stores the data of the buckets.
- `.<zone-name>.rgw.meta`: The metadata pool stores `user_keys` and other critical metadata.
- `.<zone-name>.meta:users.uid`: The user ID pool contains a map of unique user IDs.
- `.<zone-name>.meta:users.keys`: The keys pool contains access keys and secret keys for each user ID.
- `.<zone-name>.meta:users.email`: The email pool contains email addresses associated with a user ID.
- `.<zone-name>.meta:users.swift`: The Swift pool contains the Swift subuser information for a user ID.

Ceph Object Gateways store data for the bucket index (`index_pool`) and bucket data (`data_pool`) in placement pools. These may overlap; that is, you may use the same pool for the index and the data. The index pool for default placement is `{zone-name}.rgw.buckets.index` and for the data pool for default placement is `{zone-name}.rgw.buckets`.

Name	Description	Type	Default
`rgw_zonegroup_root_pool`	The pool for storing all zone group-specific information.	String	`.rgw.root`
`rgw_zone_root_pool`	The pool for storing zone-specific information.	String	`.rgw.root`

Lifecycle settings

As a storage administrator, you can set various bucket lifecycle options for a Ceph Object Gateway. These options contain default values. If you do not specify each option, then the default value is set automatically.

To set specific values for these options, update the configuration database by using the **`ceph config set client.rgw OPTION VALUE`** command.

Ceph Object Gateway bucket lifecycle settings

Name	Description	Type	Default
`rgw_lc_debug_interval`	For developer use only to debug lifecycle rules by scaling expiration rules from days into an interval in seconds. Red Hat recommends that this option not be used in a production cluster.	Integer	`-1`
`rgw_lc_lock_max_time`	The timeout value used internally by the Ceph Object Gateway.	Integer	`90`
`rgw_lc_max_objs`	Controls the sharding of the RADOS Gateway internal lifecycle work queues, and should only be set as part of a deliberate resharding workflow. Red Hat recommends not changing this setting after the setup of your cluster, without first contacting Red Hat Support.	Integer	`32`
`rgw_lc_max_rules`	The number of lifecycle rules to include in one, per bucket, lifecycle configuration document. The Amazon Web Service (AWS) limit is 1000 rules.	Integer	`1000`

Name	Description	Type	Default
`rgw_lc_max_worker`	The number of lifecycle worker threads to run in parallel, processing bucket and index shards simultaneously. Red Hat does not recommend setting a value larger than 10 without contacting Red Hat Support.	Integer	`3`
`rgw_lc_max_wp_worker`	The number of buckets that each lifecycle worker thread can process in parallel. Red Hat does not recommend setting a value larger than 10 without contacting Red Hat Support.	Integer	`3`
`rgw_lc_thread_delay`	A delay, in milliseconds, that can be injected into shard processing at several points. The default value is 0. Setting a value from 10 to 100 ms would reduce CPU utilization on RADOS Gateway instances and reduce the proportion of workload capacity of lifecycle threads relative to ingest if saturation is being observed.	Integer	`0`

Swift settings

Use this information to get Ceph Object Gateway Swift settings.

Ceph Object Gateway Swift settings			
Name	**Description**	**Type**	**Default**
`rgw_enforce_swift_acls`	Enforces the Swift Access Control List (ACL) settings.	Boolean	`true`
`rgw_swift_token_expiration`	The time in seconds for expiring a Swift token.	Integer	`24 * 3600`
`rgw_swift_url`	The URL for the Ceph Object Gateway Swift API.	String	None
`rgw_swift_url_prefix`	The URL prefix for the Swift API, for example, `http://fqdn.com/swift`.	`swift`	N/A

Name	Description	Type	Default
rgw_swift_auth_url	Default URL for verifying v1 auth tokens (if not using internal Swift auth).	String	None
rgw_swift_auth_entry	The entry point for a Swift auth URL.	String	auth

Logging settings

Use this information to get Ceph Object Gateway logging settings.

Ceph Object Gateway logging settings			
Name	**Description**	**Type**	**Default**
debug_rgw_datacache	Low level D3N logs can be enabled by the debug_rgw_datacache subsystem (up to debug_rgw_datacache=30)	Integer	1/5
rgw_log_nonexistent_bucket	Enables Ceph Object Gateway to log a request for a non-existent bucket.	Boolean	false
rgw_log_object_name	The logging format for an object name. See manpage date for details about format specifiers.	Date	%Y-%m-%d-%H-%i-%n
rgw_log_object_name_utc	Whether a logged object name includes a UTC time. If false, it uses the local time.	Boolean	false
rgw_usage_max_shards	The maximum number of shards for usage logging.	Integer	32
rgw_usage_max_user_shards	The maximum number of shards used for a single user's usage logging.	Integer	1
rgw_enable_ops_log	Enable logging for each successful Ceph Object Gateway operation.	Boolean	false
rgw_enable_usage_log	Enable the usage log.	Boolean	false
rgw_ops_log_rados	Whether the operations log should be written to the Ceph Storage Cluster backend.	Boolean	true
rgw_ops_log_socket_path	The Unix domain socket for writing operations logs.	String	None

Name	Description	Type	Default
`rgw_ops_log_data-backlog`	The maximum data backlog data size for operations logs written to a Unix domain socket.	Integer	`5 << 20`
`rgw_usage_log_flush_threshold`	The number of dirty merged entries in the usage log before flushing synchronously.	Integer	1024
`rgw_usage_log_tick_interval`	Flush pending usage log data every `n` seconds.	Integer	`30`
`rgw_intent_log_object_name`	The logging format for the intent log object name. See manpage date for details about format specifiers.	Date	`%Y-%m-%d-%i-%n`
`rgw_intent_log_object_name_utc`	Whether the intent log object name includes a UTC time. If `false`, it uses the local time.	Boolean	`false`
`rgw_data_log_window`	The data log entries window in seconds.	Integer	`30`
`rgw_data_log_changes_size`	The number of in-memory entries to hold for the data changes log.	Integer	`1000`
`rgw_data_log_num_shards`	The number of shards (objects) on which to keep the data changes log. **Note:** Changing the value is not supported.	Integer	`128`
`rgw_data_log_obj_prefix`	The object name prefix for the data log.	String	`data_log`
`rgw_replica_log_obj_prefix`	The object name prefix for the replica log.	String	`replica log`
`rgw_md_log_max_shards`	The maximum number of shards for the metadata log.	Integer	`64`
`rgw_log_http_headers`	Comma-delimited list of HTTP headers to include with ops log entries. Header names are case insensitive, and use the full header name with words separated by underscores.	String	None

Keystone settings

Use this information to get Ceph Object Gateway Keystone settings.

Ceph Object Gateway Keystone settings			
Name	**Description**	**Type**	**Default**
`rgw_keystone_url`	The URL for the Keystone server.	String	None
`rgw_keystone_admin_token`	The Keystone admin token (shared secret).	String	None
`rgw_keystone_accepted_roles`	The roles required to serve requests.	String	`Member, admin`
`rgw_keystone_token_cache_size`	The maximum number of entries in each Keystone token cache.	Integer	`10000`

Keystone integration configuration options

Integrate your configuration options into Keystone.

See "Keystone integration configuration options" on page 307 for details of the available Keystone integration configuration options.

> **Important:** After updating the Ceph configuration file, you must copy the new Ceph configuration file to all Ceph nodes in the storage cluster.

Keystone integration configuration options			
Name	**Description**	**Type**	**Default**
`rgw_s3_auth_use_keystone`	If set to `true`, the Ceph Object Gateway authenticates users by using Keystone.	Boolean	`false`
`nss_db_path`	The path to the NSS database.	String	`""`
`rgw_keystone_url`	The URL for the administrative RESTful API on the Keystone server.	String	`""`
`rgw_keystone_admin_token`	The token or shared secret that is configured internally in Keystone for administrative requests.	String	`""`
`rgw_keystone_admin_user`	The keystone admin username.	String	`""`
`rgw_keystone_admin_password`	The keystone admin user password.	String	`""`
`rgw_keystone_admin_tenant`	The Keystone admin user tenant for keystone v2.0.	String	`""`
`rgw_keystone_admin_project`	The Keystone admin user project for keystone v3.	String	`""`

Name	Description	Type	Default
`rgw_trust_forwarded_https`	When a proxy in front of the Ceph Object Gateway is used for SSL termination, it does not whether incoming HTTP connections are secure. Enable this option to trust the forwarded and X-forwarded headers sent by the proxy when determining when the connection is secure. This is mainly required for server-side encryption.	Boolean	`false`
`rgw_swift_account_in_url`	Whether the Swift account is encoded in the URL path. You *must* set this option to `true` and update the Keystone service catalog if you want the Ceph Object Gateway to support publicly-readable containers and temporary URLs.	Boolean	`false`
`rgw_keystone_admin_domain`	The Keystone admin user domain.	String	`""`
`rgw_keystone_api_version`	The version of the Keystone API to use. Valid options are `2` or `3`.	Integer	`2`
`rgw_keystone_accepted_roles`	The roles required to serve requests.	String	`"member, Member, admin"`
`rgw_keystone_accepted_admin_roles`	The list of roles allowing a user to gain administrative privileges.	String	`ResellerAdmin, swiftoperator`
`rgw_keystone_token_cache_size`	The maximum number of entries in the Keystone token cache.	Integer	`10000`
`rgw_max_attr_name_len`	The maximum length of metadata name. 0 skips the check.	Size	`0`
`rgw_max_attrs_num_in_req`	The maximum number of metadata items that can be put with a single request.	Unit	`0`
`rgw_max_attr_size`	The maximum length of metadata value. 0 skips the check.	Size	`0`
`rgw_swift_versioning_enabled`	Enabling Swift versioning.	Boolean	`0` or `1`
`rgw_keystone_accepted_reader_roles`	List of roles that can only be used for reads.	String	`""`
`rgw_swift_enforce_content_length`	Send content length when listing containers.	String	`false`

Name	Description	Type	Default
`rgw_keystone_verify_ssl`	If `true` Ceph tries to verify Keystone's SSL certificate.	Boolean	`true`
`rgw_keystone_implicit_tenants`	Create new users in their own tenants of the same name. Set this to `true` or `false` under most circumstances. This has the effect of splitting the identity space such that only the indicated protocol uses implicit tenants.	String	`false`

LDAP settings

Use this information to get Ceph Object Gateway LDAP settings.

Ceph Object Gateway LDAP settings			
Name	**Description**	**Type**	**Example**
`rgw_ldap_uri`	A space-separated list of LDAP servers in URI format.	String	`ldaps://<ldap.your.domain>`
`rgw_ldap_searchdn`	The LDAP search domain name, also known as base domain.	String	`cn=users,cn=accounts,dc=example,dc=com`
`rgw_ldap_binddn`	The gateway will bind with this LDAP entry (user match).	String	`uid=admin,cn=users,dc=example,dc=com`
`rgw_ldap_secret`	A file containing credentials for `rgw_ldap_binddn`.	String	`/etc/openldap/secret`
`rgw_ldap_dnattr`	LDAP attribute containing Ceph Object Gateway user names (to form binddns).	String	`uid`