Red Hat Ceph Storage 4

Object Gateway Configuration and Administration Guide

Configuring and administering the Ceph Storage Object Gateway

Last Updated: 2022-12-30

Red Hat Ceph Storage 4 Object Gateway Configuration and Administration Guide

Configuring and administering the Ceph Storage Object Gateway

Legal Notice

Abstract

This document provides instructions for configuring and administering the Ceph Storage Object Gateway. Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message .

Table of Contents

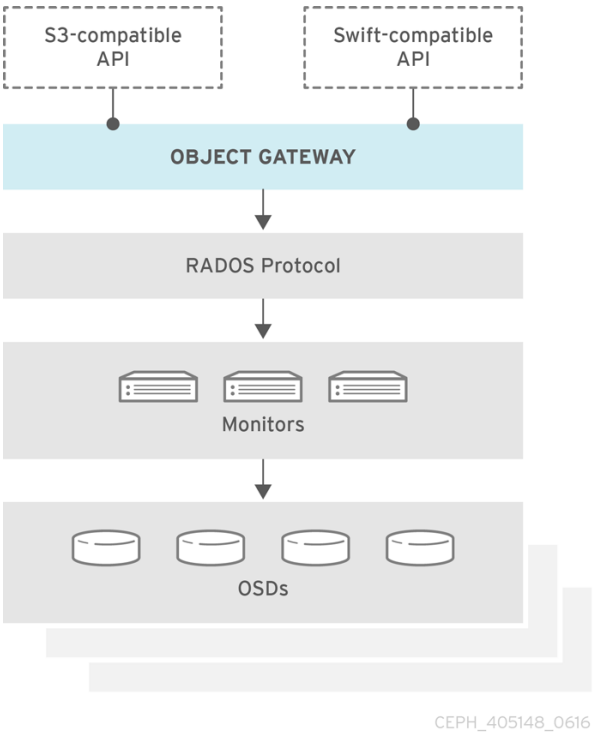
CHAPTER 1. OVERVIEW

Ceph Object Gateway, also known as RADOS Gateway (RGW) is an object storage interface built on top of **librados** to provide applications with a RESTful gateway to Ceph storage clusters. Ceph object gateway supports two interfaces:

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

2. **Swift-compatible:** 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 server for 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. 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 may write data with one API and retrieve it with the other.



WARNING

Do not use RADOS snapshots on pools used by RGW. Doing so can introduce undesirable data inconsistencies.

CHAPTER 2. CONFIGURATION

2.1. THE BEAST AND CIVETWEB FRONT END WEB SERVERS

The Ceph Object Gateway provides Beast and CivetWeb as front ends, both are C/C++ embedded web servers.

Beast

Starting with Red Hat Ceph Storage 4, Beast is the default front-end web server. When upgrading from Red Hat Ceph Storage 3, the **rgw_frontends** parameter automatically changes to Beast. Beast uses the **Boost.Beast** C++ library to parse HTTP, and **Boost.Asio** to do asynchronous network I/O.

CivetWeb

In Red Hat Ceph Storage 3, CivetWeb is the default front end, but Beast can also be used by setting the **rgw_frontends** option accordingly. CivetWeb is an HTTP library, which is a fork of the Mongoose project.

Additional Resources

- Boost C++ Libraries
- CivetWeb on GitHub

2.2. USING THE BEAST FRONT END

The Ceph Object Gateway provides CivetWeb and Beast embedded HTTP servers as front ends. The Beast front end uses the **Boost.Beast** library for HTTP parsing and the **Boost.Asio** library for asynchronous network I/O. In Red Hat Ceph Storage version 3.x, CivetWeb was the default front end, and to use the Beast front end it needed to be specified with **rgw_frontends** in the Red Hat Ceph Storage configuration file. As of Red Hat Ceph Storage version 4.0, the Beast front end is default, and upgrading from Red Hat Ceph Storage 3.x automatically changes the **rgw_frontends** parameter to Beast.

Additional Resources

- *Beast configuration options*

2.3. BEAST CONFIGURATION OPTIONS

The following Beast configuration options can be passed to the embedded web server in the Ceph configuration file for the RADOS Gateway. Each option has a default value. If a value is not specified, the default value is empty.

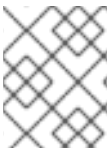
Option	Description	Default
endpoint and **ssl_endpoint**	Sets the listening address in the form **address[:port]** where the address is an IPv4 address string in dotted decimal form, or an IPv6 address in hexadecimal notation surrounded by square brackets. The optional port defaults to **8080** for **endpoint** and **443** for **ssl_endpoint**. It can be specified multiple times as in **endpoint=[::1] endpoint=192.168.0.100:8000**.	EMPTY

Option	Description	Default
ssl_certificate	Path to the SSL certificate file used for SSL-enabled endpoints. If the file is a PEM file containing more than one item the order is important. The file must begin with the RGW server key, then any intermediate certificate, and finally the CA certificate.	EMPTY
ssl_private_key	Optional path to the private key file used for SSL-enabled endpoints. If one is not given the file specified by **ssl_certificate** is used as the private key.	EMPTY
tcp_nodelay	Performance optimization in some environments.	EMPTY
request_timeout_ms	Set an explicit request timeout for the Beast front end. Setting a larger request timeout can make the gateway more tolerant of slow clients (for example, clients connected over high-latency networks).	65

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>
```



NOTE

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

Additional Resources

- See Using the Beast front end for more information.

2.4. CHANGING THE CIVETWEB PORT

When the Ceph Object Gateway is installed using Ansible it configures CivetWeb to run on port **8080**. Ansible does this by adding a line similar to the following in the Ceph configuration file:

```
rgw frontends = civetweb port=192.168.122.199:8080 num_threads=100
```

IMPORTANT

If the Ceph configuration file does not include the **rgw frontends = civetweb** line, the Ceph Object Gateway listens on port **7480**. If it includes an **rgw_frontends = civetweb** line but there is no port specified, the Ceph Object Gateway listens on port **80**.

IMPORTANT

Because Ansible configures the Ceph Object Gateway to listen on port **8080** and the supported way to install Red Hat Ceph Storage 4 is using **ceph-ansible**, port **8080** is considered the default port in the Red Hat Ceph Storage 4 documentation.

Prerequisites

- A running Red Hat Ceph Storage 4.1 cluster.
- A Ceph Object Gateway node.

Procedure

1. On the gateway node, open the Ceph configuration file in the **/etc/ceph/** directory.
2. Find an Ceph Object Gateway (RGW) client section similar to the example:

```
[client.rgw.gateway-node1]
host = gateway-node1
keyring = /var/lib/ceph/radosgw/ceph-rgw.gateway-node1/keyring
log file = /var/log/ceph/ceph-rgw-gateway-node1.log
rgw frontends = civetweb port=192.168.122.199:8080 num_threads=100
```

   The **[client.rgw.gateway-node1]** heading identifies this portion of the Ceph configuration file as configuring a Ceph Storage Cluster client where the client type is a Ceph Object Gateway as identified by **rgw**, and the name of the node is **gateway-node1**.

3. To change the default Ansible configured port of **8080** to **80** edit the **rgw frontends** line:

```
rgw frontends = civetweb port=192.168.122.199:80 num_threads=100
```

   Ensure there is no whitespace between **port=*port-number*** in the **rgw_frontends** key/value pair.

   Repeat this step on any other gateway nodes you want to change the port on.

4. Restart the Ceph Object Gateway service from each gateway node to make the new port setting take effect:

```
# systemctl restart ceph-radosgw.target
```

5. Ensure the configured port is open on each gateway node's firewall:

```
# firewall-cmd --list-all
```

6. If the port is not open, add the port and reload the firewall configuration:

```
# firewall-cmd --zone=public --add-port 80/tcp --permanent
# firewall-cmd --reload
```

Additional Resources

- *Using SSL with CivetWeb*

- *Civetweb Configuration Options*

2.5. USING SSL WITH CIVETWEB

In Red Hat Ceph Storage 1, Civetweb SSL support for the Ceph Object Gateway relied on HAProxy and keepalived. In Red Hat Ceph Storage 2 and later releases, Civetweb can use the OpenSSL library to provide Transport Layer Security (TLS).

IMPORTANT

Production deployments **MUST** use HAProxy and keepalived to terminate the SSL connection at HAProxy. Using SSL with Civetweb is recommended **ONLY** for small-to-medium sized **test and pre-production** deployments.

To use SSL with Civetweb, obtain a certificate from a Certificate Authority (CA) that matches the hostname of the gateway node. Red Hat recommends obtaining a certificate from a CA that has **subject alternate name** fields and a wildcard for use with S3-style subdomains.

Civetweb requires the key, server certificate and any other certificate authority or intermediate certificate in a single **.pem** file.

IMPORTANT

A **.pem** file contains the secret key. Protect the **.pem** file from unauthorized access.

To configure a port for SSL, add the port number to **rgw_frontends** and append an **s** to the port number to indicate that it is a secure port. Additionally, add **ssl_certificate** with a path to the **.pem** file. For example:

```
[client.rgw.{hostname}]
rgw_frontends = "civetweb port=443s ssl_certificate=/etc/ceph/private/server.pem"
```

2.6. CIVETWEB CONFIGURATION OPTIONS

The following Civetweb configuration options can be passed to the embedded web server in the Ceph configuration file for the RADOS Gateway. Each option has a default value and if a value is not specified, then the default value is empty.

Option	Description	Default
access_log_file	Path to a file for access logs. Either full path, or relative to the current working directory. If absent (default), then accesses are not logged.	EMPTY
error_log_file	Path to a file for error logs. Either full path, or relative to the current working directory. If absent (default), then errors are not logged.	EMPTY

Option	Description	Default
num_threads	Number of worker threads. Civetweb handles each incoming connection in a separate thread. Therefore, the value of this option is effectively the number of concurrent HTTP connections Civetweb can handle.	50
request_timeout_ms	Timeout for network read and network write operations, in milliseconds. If a client intends to keep long-running connection, either increase this value or (better) use keep-alive messages.	30000

The following is an example of the **/etc/ceph/ceph.conf** file with some of these options set:

```
...

[client.rgw.node1]
rgw frontends = civetweb request_timeout_ms=30000
error_log_file=/var/log/radosgw/civetweb.error.log
access_log_file=/var/log/radosgw/civetweb.access.log
```

Both the CivetWeb and Beast frontends write an access log line recording of all requests processed by the server to the RADOS gateway log file.

2.7. ADD A WILDCARD TO THE DNS

To use Ceph with S3-style subdomains, for example **bucket-name.domain-name.com**, add a wildcard to the DNS record of the DNS server the **ceph-radosgw** daemon uses to resolve domain names.

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

```
address=/.{hostname-or-fqdn}/{host-ip-address}
```

For example:

```
address=/.gateway-node1/192.168.122.75
```

For **bind**, add a wildcard to the DNS record. For example:

```
$TTL    604800
@       IN      SOA     gateway-node1. root.gateway-node1. (
                              2         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      gateway-node1.
@       IN      A       192.168.122.113
*       IN      CNAME   @
```

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

```
ping mybucket.{hostname}
```

For example:

```
ping mybucket.gateway-node1
```

If the DNS server is on the local machine, you may need to modify **/etc/resolv.conf** by adding a nameserver entry for the local machine.

Finally, specify the host name or address of the DNS server in the appropriate **[client.rgw.{instance}]** section of the Ceph configuration file using the **rgw_dns_name = {hostname}** setting. For example:

```
[client.rgw.rgw1.rgw0]
...
rgw_dns_name = {hostname}
```

NOTE

As a best practice, make changes to the Ceph configuration file at a centralized location such as an admin node or **ceph-ansible** and redistribute the configuration file as necessary to ensure consistency across the cluster.

Finally, restart the Ceph object gateway so that DNS setting takes effect.

2.8. ADJUSTING LOGGING AND DEBUGGING OUTPUT

Once you finish the setup procedure, check your logging output to ensure it meets your needs. If you encounter issues with your configuration, you can increase logging and debugging messages in the **[global]** section of your Ceph configuration file and restart the gateway(s) to help troubleshoot any configuration issues. For example:

```
[global]
#append the following in the global section.
debug ms = 1
debug civetweb = 20
```

For RGW debug logs, add the following parameter in the **[client.rgw.{instance}]** section of your Ceph configuration file:

```
[client.rgw.rgw1.rgw0]
...
debug rgw = 20
```

You may also modify these settings at runtime. For example:

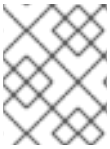
```
# ceph tell osd.0 injectargs --debug_civetweb 10/20
```

The Ceph log files reside in **/var/log/ceph** by default.

For general details on logging and debugging, see the *Ceph debugging and logging configuration* section of the *Red Hat Ceph Storage Configuration Guide*.

2.9. S3 SERVER-SIDE ENCRYPTION

The Ceph Object Gateway supports server-side encryption of uploaded objects for the S3 application programing 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.



NOTE

Red Hat does NOT support S3 object encryption of Static Large Object (SLO) or Dynamic Large Object (DLO).

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 may disable SSL during testing by setting the **rgw_crypt_require_ssl** configuration setting to **false** at runtime, setting it to **false** in the Ceph configuration file and restarting the gateway instance, or setting it to **false** in the Ansible configuration files and replaying the Ansible playbooks for the Ceph Object Gateway.

In a production environment, it might not be possible to send encrypted requests over SSL. In such a case, send requests using HTTP with server-side encryption.

For information about how to configure HTTP with server-side encryption, see the *Additional Resources* section below.

There are two options 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.

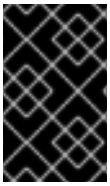
Ceph Object Gateway implements the customer-provided key behavior in the S3 API according to the Amazon SSE-C specification.

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, and OpenStack Barbican. However, OpenStack Barbican is a Technology Preview and is not supported for use in production systems.

Additional Resources

- Amazon SSE-C
- Amazon SSE-KMS
- *Configuring server-side encryption*
- *The HashiCorp Vault*

2.10. 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. To set up the Ceph Object Gateways to accept the server-side encryption requests, see Configuring server-side encryption.

2.11. CONFIGURING SERVER-SIDE ENCRYPTION

As a storage administrator, 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

- Root-level access to all nodes in the storage cluster.
- A running Red Hat Ceph Storage cluster.
- Ceph Object Gateway is installed.
- HAProxy is installed.

Procedure

1. Edit the **haproxy.cfg** file:

   Example

```
frontend http_web
    bind *: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
#     bind *: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. On all nodes in the cluster, add the following parameter to the **[global]** section of the Ceph configuration file:

```
  rgw_trust_forwarded_https=true
```

4. Enable and start HAProxy:

```
[root@haproxy]# systemctl enable haproxy
[root@haproxy]# systemctl start haproxy
```

5. To ensure that **rgw_trust_forwarded_https=true** is not removed from the Ceph configuration file when Ansible is run, edit the ceph-ansible **all.yml** file and set **rgw_trust_forwarded_https** in the **ceph_conf_overrides** / **global** section to **true**.

```
ceph_conf_overrides:
  global:
    rgw_trust_forwarded_https: true
```

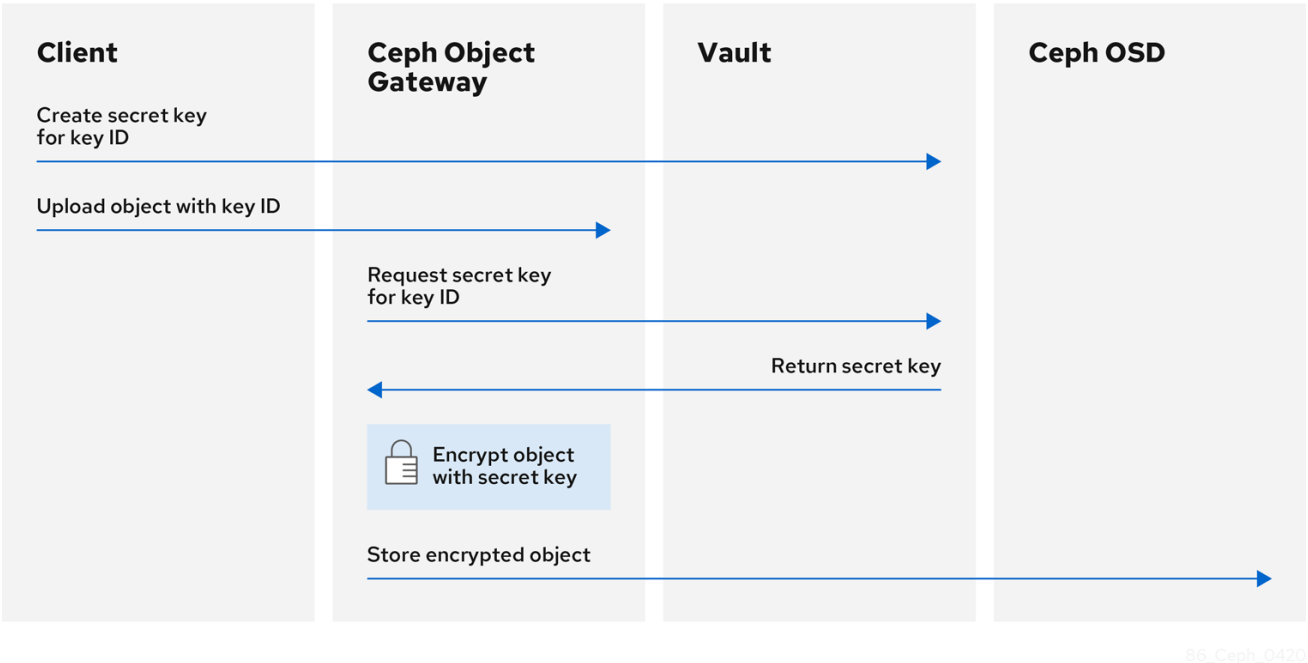
6. When you have finished making changes, run the ceph-ansible playbook to update the configuration on all Ceph nodes.

Additional Resources

- Installing and configuring HAProxy
- Installing the Ceph client role
- Installing a Red Hat storage cluster

2.12. THE 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.



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.

2.12.1. Prerequisites

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

2.12.2. Secret engines for Vault

The HashiCorp Vault provides several secret engines to generate, store, or encrypt data. The application programming interface (API) send data calls to the secret engine asking for 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
- Transit

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. The key names must be strings, and the engine will convert 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.

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. The Transit secret engine supports these key types:

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

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

Additional Resources

- See the *KV Secrets Engine* documentation on Vault's project site for more information.
- See the *Transit Secrets Engine* documentation on Vault's project site for more information.

2.12.3. Authentication for Vault

The HashiCorp Vault supports several types of authentication mechanisms. The Ceph Object Gateway currently supports the Vault agent and the token authentication 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.

Token

The token authentication method allows users to authenticate using a token. You can create new tokens, revoke secrets by token, and many other token operations. You can bypass other authentication methods, by using the token store. When using the token authentication method, the **rgw_crypt_vault_token_file** option must also be used. The token file can only be readable by the Ceph Object Gateway. Also, a Vault token with a restricted policy that allows fetching of keyrings from a specific path must be used.

WARNING

Red Hat recommends not using token authentication for production environments.

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.

Additional Resources

- See the *Token Auth Method* documentation on Vault's project site for more information.
- See the *Vault Agent* documentation on Vault's project site for more information.

2.12.4. 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.

Additional Resources

- See the *Vault Enterprise Namespaces* documentation on Vault's project site for more information.

2.12.5. Configuring the Ceph Object Gateway to use the Vault

To configure the Ceph Object Gateway to use the HashiCorp Vault it must be set as the encryption key store. Currently, the Ceph Object Gateway supports two different secret engines, and two different authentication methods.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Installation of the Ceph Object Gateway software.
- Root-level access to a Ceph Object Gateway node.

Procedure

1. Open for editing the Ceph configuration file, by default **/etc/ceph/ceph.conf**, and enable the Vault as the encryption key store:

```
rgw_crypt_s3_kms_backend = vault
```

2. Under the **[client.radosgw.*INSTANCE_NAME*]** section, choose a Vault authentication method, either Token or the Vault agent.

   a. If using **Token**, then add the following lines:

```
rgw_crypt_vault_auth = token
rgw_crypt_vault_token_file = /etc/ceph/vault.token
rgw_crypt_vault_addr = http://VAULT_SERVER:8200
```

   b. If using the **Vault agent**, then add the following lines:

```
rgw_crypt_vault_auth = agent
rgw_crypt_vault_addr = http://VAULT_SERVER:8100
```

3. Under the **[client.radosgw.*INSTANCE_NAME*]** section, choose a Vault secret engine, either Key/Value or Transit.

   a. If using **Key/Value**, then add the following line:

   ```
   rgw_crypt_vault_secret_engine = kv
   ```

   b. If using **Transit**, then add the following line:

   ```
   rgw_crypt_vault_secret_engine = transit
   ```

4. Optionally, Under the **[client.radosgw.*INSTANCE_NAME*]** section, you can set the Vault namespace where the encryption keys will be retrieved:

   ```
   rgw_crypt_vault_namespace = NAME_OF_THE_NAMESPACE
   ```

5. Restrict where the Ceph Object Gateway retrieves the encryption keys from the Vault by setting a path prefix:

   Example

   ```
   rgw_crypt_vault_prefix = /v1/secret/data
   ```

   a. For exportable Transit keys, set the prefix path as follows:

   ```
   rgw_crypt_vault_prefix = /v1/transit/export/encryption-key
   ```

   Assuming the domain name of the Vault server is **vault-server**, the Ceph Object Gateway will fetch encrypted transit keys from the following URL:

   Example

   ```
   http://vault-server:8200/v1/transit/export/encryption-key
   ```

6. Save the changes to the Ceph configuration file.

Additional Resources

- See the *Secret engines for Vault* section of the *Red Hat Ceph Storage Object Gateway Configuration and Administration Guide* for more details.
- See the *Authentication for Vault* section of the *Red Hat Ceph Storage Object Gateway Configuration and Administration Guide* for more details.

2.12.6. 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 encyption must be 256-bits long and encoded using **base64**.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Installation of the HashiCorp Vault software.
- Root-level access to the HashiCorp Vault node.

Procedure

1. Enable the Key/Value version 2 secret engine:

```
[root@vault ~]# vault secrets enable 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
```

2.12.7. 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 in order to be used for server-side encryption with the Ceph Object Gateway.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Installation of the HashiCorp Vault software.
- Root-level access to the HashiCorp Vault node.

Procedure

1. Enable the Transit secret engine:

```
[root@vault ~]# vault secrets enable transit
```

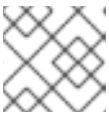
2. Create a new exportable key:

Syntax

```
vault write -f transit/keys/BUCKET_NAME exportable=true
```

Example

```
[root@vault ~]# vault write -f transit/keys/mybucketkey exportable=true
```



NOTE

By default the above command creates a **aes256-gcm96** type key.

3. Verify the creation of the key:

Syntax

```
vault read transit/export/encryption-key/BUCKET_NAME/VERSION_NUMBER
```

Example

```
[root@vault ~]# vault read transit/export/encryption-key/mybucketkey/1

Key     Value
---     -----
keys    map[1:-gbTI9lNpqv/V/2lDcmH2Nq1xKn6FPDWarCmFM2aNsQ=]
name    mybucketkey
type    aes256-gcm96
```

NOTE

Providing the full key path, including the key version is required.

2.12.8. Uploading an object using AWS and the Vault

When uploading an object to the Ceph Object Gateway, the Ceph Object Gateway will fetch the key from the Vault, and then encrypt and store the object in a bucket. When a request is made to download the object, the Ceph Object Gateway will automatically retrieve the corresponding key from the Vault and decrypt 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.

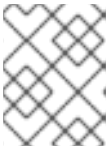
- Installation of the Ceph Object Gateway software.
- Installation of the HashiCorp Vault software.
- Access to a Ceph Object Gateway client node.
- Access to Amazon Web Services (AWS).

Procedure

1. Upload an object using the AWS command-line client:

   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

The key fetching URL used in the example is: **http://vault-server:8200/v1/secret/data/myproject/mybucketkey**

2.12.9. Additional Resources

- See the *Install Vault* documentation on Vault's project site for more information.

2.13. TESTING THE GATEWAY

To use the REST interfaces, first create an initial Ceph Object Gateway user for the S3 interface. Then, create a subuser for the Swift interface. You then need to verify if the created users are able to access the gateway.

2.13.1. 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:

   ```
   radosgw-admin user create --uid=name --display-name="First User"
   ```

Replace *name* with the name of the S3 user, for example:

```
[root@master-zone]# radosgw-admin user create --uid="testuser" --display-name="First
User"
{
    "user_id": "testuser",
    "display_name": "First User",
    "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 character. 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 is 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.

2.13.2. 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

- 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@rgw]# 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:

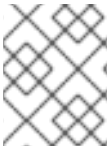
Example

```
[root@rgw]# 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"
}
```


2.13.3. 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.



NOTE

System users must have root privileges over the entire zone, as the output would contain additional json fields for maintaining metadata.

Prerequisites

- **root** or **sudo** access.
- Ceph Object Gateway installed.
- S3 user created.

Procedure

1. Enable the common repository for Red Hat Enterprise Linux 7 and the High Availability repository for Red Hat Enterprise Linux 8:

   Red Hat Enterprise Linux 7

```
# subscription-manager repos --enable=rhel-7-server-rh-common-rpms
```

Red Hat Enterprise Linux 8

```
# subscription-manager repos --enable=rhel-8-for-x86_64-highavailability-rpms
```

2. Install the **python-boto** package.

Red Hat Enterprise Linux 7

```
# yum install python-boto
```

Red Hat Enterprise Linux 8

```
# dnf install python3-boto3
```

3. Create the Python script:

```
vi s3test.py
```

4. Add the following contents to the file:

Red Hat Enterprise Linux 7

```
import boto
import boto.s3.connection

access_key = 'ACCESS'
secret_key = 'SECRET'

boto.config.add_section('s3')

conn = boto.connect_s3(
        aws_access_key_id = access_key,
        aws_secret_access_key = secret_key,
        host = 's3.ZONE.hostname',
        port = PORT,
        is_secure=False,
        calling_format = boto.s3.connection.OrdinaryCallingFormat(),
        )

bucket = conn.create_bucket('my-new-bucket')
for bucket in conn.get_all_buckets():
 print "{name}\t{created}".format(
  name = bucket.name,
  created = bucket.creation_date,
)
```

Red Hat Enterprise Linux 8

```
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']
))
```

a. Replace ***ZONE*** with the zone name 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.

b. Replace ***ACCESS*** and ***SECRET*** with the **access_key** and **secret_key** values from the *Create an S3 User* section in the *Red Hat Ceph Storage Object Gateway Configuration and Administration Guide*.

5. Run the script:

Red Hat Enterprise Linux 7

```
python s3test.py
```

Red Hat Enterprise Linux 8

```
python3 s3test.py
```

Example output:

```
my-new-bucket 2021-08-16T17:09:10.000Z
```

2.13.4. Test Swift Access

Swift access can be verified via the **swift** command line client. The command **man swift** will provide more information on available command line options.

To install **swift** client, execute the following:

```
sudo yum install python-setuptools
sudo easy_install pip
sudo pip install --upgrade setuptools
sudo pip install --upgrade python-swiftclient
```

To test swift access, execute the following:

```
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 **radosgw-admin key create** command executed for the **swift** user. Replace {port} with the port number you are using with Civetweb (e.g., **8080** is the default). If you don't replace the port, it will default to port **80**.

For example:

```
swift -A http://10.19.143.116:8080/auth/1.0 -U testuser:swift -K
'244+fz2gSqoHwR3lYtSbIyomyPHf3i7rgSJrF/IA' list
```

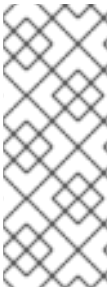
The output should be:

```
my-new-bucket
```

2.14. CONFIGURING HAPROXY/KEEPALIVED

The Ceph Object Gateway allows you to assign many instances of the object gateway to a single zone so that you can scale out as load increases, that is, the same zone group and zone; however, you do not need a federated architecture to use HAProxy/**keepalived**. Since each Ceph Object Gateway instance has its own IP address, you can use HAProxy and **keepalived** to balance the load across Ceph Object Gateway servers.

Another use case for HAProxy and **keepalived** is to terminate HTTPS at the HAProxy server. You can use an HAProxy server to terminate HTTPS at the HAProxy server and use HTTP between the HAProxy server and the Civetweb gateway instances.



NOTE

This section describes configuration of HAProxy and **keepalived** for Red Hat Enterprise Linux 7.

For Red Hat Enterprise Linux 8, install **keepalived** and **haproxy** packages to install the Load Balancer. See the *Do we need any additional subscription for Load Balancing on Red Hat Enterprise Linux 8?* Knowledgebase article for details.

2.14.1. HAProxy/keepalived Prerequisites

To set up an HA Proxy with the Ceph Object Gateway, you must have:

- A running Ceph cluster
- At least two Ceph Object Gateway servers within the same zone configured to run on port **80**. If you follow the simple installation procedure, the gateway instances are in the same zone group and zone by default. If you are using a federated architecture, ensure that the instances are in the same zone group and zone; and,
- At least two servers for HAProxy and **keepalived**.

NOTE

This section assumes that you have at least two Ceph Object Gateway servers running, and that you get a valid response from each of them when running test scripts over port **80**.

For a detailed discussion of HAProxy and **keepalived**, see Load Balancer Administration.

2.14.2. Preparing HAProxy Nodes

The following setup assumes two HAProxy nodes named **haproxy** and **haproxy2** and two Ceph Object Gateway servers named **rgw1** and **rgw2**. You may use any naming convention you prefer. Perform the following procedure on your at least two HAProxy nodes:

1. Install Red Hat Enterprise Linux 7.
2. Register the nodes.

```
[root@haproxy]# subscription-manager register
```

3. Enable the RHEL server repository.

```
[root@haproxy]# subscription-manager repos --enable=rhel-7-server-rpms
```

4. Update the server.

```
[root@haproxy]# yum update -y
```

5. Install admin tools (e.g., **wget**, **vim**, etc.) as needed.
6. Open port **80**.

```
[root@haproxy]# firewall-cmd --zone=public --add-port 80/tcp --permanent
[root@haproxy]# firewall-cmd --reload
```

7. For HTTPS, open port **443**.

```
[root@haproxy]# firewall-cmd --zone=public --add-port 443/tcp --permanent
[root@haproxy]# firewall-cmd --reload
```

8. Connect to the required port.

```
[root@haproxy]# semanage port -m -t http_cache_port_t -p tcp 8081
```

2.14.3. Installing and Configuring keepalived

Perform the following procedure on your at least two HAProxy nodes:

Prerequisites

- A minimum of two HAProxy nodes.
- A minimum of two Object Gateway nodes.

Procedure

1. Install **keepalived**:

```
[root@haproxy]# yum install -y keepalived
```

2. Configure **keepalived** on both HAProxy nodes:

```
[root@haproxy]# vim /etc/keepalived/keepalived.conf
```

In the configuration file, there is a script to check the **haproxy** processes:

```
vrrp_script chk_haproxy {
  script "killall -0 haproxy" # check the haproxy process
  interval 2 # every 2 seconds
  weight 2 # add 2 points if OK
}
```

Next, the instance on the master and backup load balancers uses **eno1** as the network interface. It also assigns a virtual IP address, that is, **192.168.1.20**.

Master load balancer node

```
vrrp_instance RGW {
    state MASTER # might not be necessary. This is on the Master LB node.
    @main interface eno1
    priority 100
    advert_int 1
    interface eno1
    virtual_router_id 50
    @main unicast_src_ip 10.8.128.43 80
    unicast_peer {
          10.8.128.53
          }
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.1.20
    }
    track_script {
      chk_haproxy
    }
}
virtual_server 192.168.1.20 80 eno1 { #populate correct interface
    delay_loop 6
    lb_algo wlc
    lb_kind dr
    persistence_timeout 600
    protocol TCP
    real_server 10.8.128.43 80 { # ip address of rgw2 on physical interface, haproxy listens
here, rgw listens to localhost:8080 or similar
        weight 100
        TCP_CHECK { # perhaps change these to a HTTP/SSL GET?
```

```
            connect_timeout 3
        }
    }
    real_server 10.8.128.53 80 { # ip address of rgw3 on physical interface, haproxy listens
here, rgw listens to localhost:8080 or similar
        weight 100
        TCP_CHECK { # perhaps change these to a HTTP/SSL GET?
            connect_timeout 3
        }
    }
}
```

Backup load balancer node

```
vrrp_instance RGW {
    state BACKUP # might not be necessary?
    priority 99
    advert_int 1
    interface eno1
    virtual_router_id 50
    unicast_src_ip 10.8.128.53 80
    unicast_peer {
          10.8.128.43
          }
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.1.20
    }
    track_script {
      chk_haproxy
    }
}
virtual_server 192.168.1.20 80 eno1 { #populate correct interface
    delay_loop 6
    lb_algo wlc
    lb_kind dr
    persistence_timeout 600
    protocol TCP
    real_server 10.8.128.43 80 { # ip address of rgw2 on physical interface, haproxy listens
here, rgw listens to localhost:8080 or similar
        weight 100
        TCP_CHECK { # perhaps change these to a HTTP/SSL GET?
            connect_timeout 3
        }
    }
    real_server 10.8.128.53 80 { # ip address of rgw3 on physical interface, haproxy listens
here, rgw listens to localhost:8080 or similar
        weight 100
        TCP_CHECK { # perhaps change these to a HTTP/SSL GET?
            connect_timeout 3
        }
    }
}
```

3. Enable and start the **keepalived** service:

```
[root@haproxy]# systemctl enable keepalived
[root@haproxy]# systemctl start keepalived
```

Additional Resources

- For a detailed discussion of configuring **keepalived**, refer to Initial Load Balancer Configuration with Keepalived.

2.14.4. Installing and Configuring HAProxy

Perform the following procedure on your at least two HAProxy nodes:

1. Install **haproxy**.

```
[root@haproxy]# yum install haproxy
```

2. Configure **haproxy** for SELinux and HTTP.

```
[root@haproxy]# vim /etc/firewalld/services/haproxy-http.xml
```

Add the following lines:

```
<?xml version="1.0" encoding="utf-8"?>
<service>
<short>HAProxy-HTTP</short>
<description>HAProxy load-balancer</description>
<port protocol="tcp" port="80"/>
</service>
```

As **root**, assign the correct SELinux context and file permissions to the **haproxy-http.xml** file.

```
[root@haproxy]# cd /etc/firewalld/services
[root@haproxy]# restorecon haproxy-http.xml
[root@haproxy]# chmod 640 haproxy-http.xml
```

3. If you intend to use HTTPS, configure **haproxy** for SELinux and HTTPS.

```
[root@haproxy]# vim /etc/firewalld/services/haproxy-https.xml
```

Add the following lines:

```
<?xml version="1.0" encoding="utf-8"?>
<service>
<short>HAProxy-HTTPS</short>
<description>HAProxy load-balancer</description>
<port protocol="tcp" port="443"/>
</service>
```

As **root**, assign the correct SELinux context and file permissions to the **haproxy-https.xml** file.

```
# cd /etc/firewalld/services
# restorecon haproxy-https.xml
# chmod 640 haproxy-https.xml
```

4. If you intend to use HTTPS, generate keys for SSL. If you do not have a certificate, you may use a self-signed certificate. To generate a key, see to *Generating a New Key and Certificate* section in the *System Administrator's Guide* for Red Hat Enterprise Linux 7.
Finally, put the certificate and key into a PEM file.

```
[root@haproxy]# cat example.com.crt example.com.key > example.com.pem
[root@haproxy]# cp example.com.pem /etc/ssl/private/
```

5. Configure **haproxy**.

```
[root@haproxy]# vim /etc/haproxy/haproxy.cfg
```

The **global** and **defaults** may remain unchanged. After the **defaults** section, you will need to configure **frontend** and **backend** sections. For example:

```
frontend http_web
    bind *: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 10.0.0.71:80 check
    server  rgw2 10.0.0.80:80 check
```

For a detailed discussion of HAProxy configuration, refer to HAProxy Configuration.

6. Enable/start **haproxy**

```
[root@haproxy]# systemctl enable haproxy
[root@haproxy]# systemctl start haproxy
```

2.14.5. Testing the HAProxy Configuration

On your HAProxy nodes, check to ensure the virtual IP address from your **keepalived** configuration appears.

```
[root@haproxy]# ip addr show
```

On your calamari node, see if you can reach the gateway nodes via the load balancer configuration. For example:

```
[root@haproxy]# wget haproxy
```

This should return the same result as:

```
[root@haproxy]# wget rgw1
```

If it returns an **index.html** file with the following contents:

```
<?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>
```

Then, your configuration is working properly.

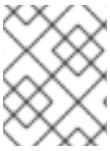
2.15. CONFIGURING GATEWAYS FOR STATIC WEB HOSTING

Traditional web hosting involves setting up a web server for each website, which can use resources inefficiently when content does not change dynamically. Ceph Object Gateway can host static web sites in S3 buckets—that is, 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 VMs with web servers for each site.

2.15.1. Static Web Hosting Assumptions

Static web hosting requires at least one running Ceph Storage Cluster, and at least two Ceph Object Gateway instances for static web sites. Red Hat assumes that each zone will have multiple gateway instances load balanced by HAProxy/keepalived.

See Configuring HAProxy/keepalived for additional details on HAProxy/keepalived.



NOTE

Red Hat **DOES NOT** support using a Ceph Object Gateway instance to deploy both standard S3/Swift APIs and static web hosting simultaneously.

2.15.2. 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/keepalived.

2.15.3. Static Web Hosting Gateway Setup

To enable a gateway for static web hosting, edit the Ceph configuration file and add the following settings:

```
[client.rgw.<STATIC-SITE-HOSTNAME>]
...
rgw_enable_static_website = true
rgw_enable_apis = s3, s3website
rgw_dns_name = objects-zonegroup.domain.com
rgw_dns_s3website_name = objects-website-zonegroup.domain.com
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 will use canonical name extensions, set **rgw_resolve_cname** to **true**.

IMPORTANT

The FQDNs of **rgw_dns_name** and **rgw_dns_s3website_name** **MUST NOT** overlap.

2.15.4. Static Web Hosting DNS Configuration

The following 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 respectively. 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 respectively.

```
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 subdomains, use a wildcard in the DNS entry and 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 in the following manner:

```
http://bucket1.objects-website-zonegroup.domain.com
```

Where the bucket name is **bucket1**.

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:

```
www.example.com. IN CNAME bucket2.objects-website-zonegroup.domain.com.
```

Where the bucket name is **bucket2**.

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
```

2.15.5. Creating a Static Web Hosting Site

To create a static website perform the following steps:

1. Create an S3 bucket. The bucket name MAY 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. See DNS Settings for details.

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 MAY have **error.html** file.

3. Verify the website's contents. At this point, only the creator of the bucket will have access to the contents.

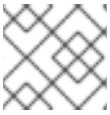
4. Set permissions on the files so that they are publicly readable.

2.16. EXPORTING THE NAMESPACE TO NFS-GANESHA

In Red Hat Ceph Storage 3 and later, the Ceph Object Gateway provides the ability to export S3 object namespaces by using NFS version 3 and NFS version 4.1 for production systems.

NOTE

The NFS Ganesha feature is not for general use, but rather for migration to an S3 cloud only.



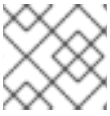
NOTE

Red Hat Ceph Storage does not support NFS-export of versioned buckets.

The implementation conforms to Amazon Web Services (AWS) hierarchical namespace conventions which map UNIX-style path names onto S3 buckets and objects. The top level of the attached namespace, which is subordinate to the NFSv4 pseudo root if present, consists of the Ceph Object Gateway S3 buckets, where buckets are represented as NFS directories. Objects within a bucket are presented as NFS file and directory hierarchies, following S3 conventions. Operations to create files and directories are supported.

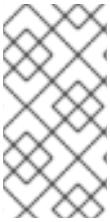
NOTE

Creating or deleting hard or soft links IS NOT supported. Performing rename operations on buckets or directories IS NOT supported via NFS, but rename on files IS supported within and between directories, and between a file system and an NFS mount. File rename operations are more expensive when conducted over NFS, as they change the target directory and typically forces a full **readdir** to refresh it.



NOTE

Editing files via the NFS mount IS NOT supported.



NOTE

The Ceph Object Gateway requires applications to write sequentially from offset 0 to the end of a file. Attempting to write out of order causes the upload operation to fail. To work around this issue, use utilities like **cp**, **cat**, or **rsync** when copying files into NFS space. Always mount with the **sync** option.

The Ceph Object Gateway with NFS is based on an in-process library packaging of the Gateway server and a File System Abstraction Layer (FSAL) namespace driver for the NFS-Ganesha NFS server. At runtime, an instance of the Ceph Object Gateway daemon with NFS combines a full Ceph Object Gateway daemon, albeit without the Civetweb HTTP service, with an NFS-Ganesha instance in a single process. To make use of this feature, deploy NFS-Ganesha version 2.3.2 or later.

Perform the steps in the Before you Start and Configuring an NFS-Ganesha Instance procedures on the host that will contain the NFS-Ganesha (**nfs-ganesha-rgw**) instance.

Running Multiple NFS Gateways

Each NFS-Ganesha instance acts as a full gateway endpoint, with the current limitation that an NFS-Ganesha instance cannot be configured to export HTTP services. As with ordinary gateway instances, any number of NFS-Ganesha instances can be started, exporting the same or different resources from the cluster. This enables the clustering of NFS-Ganesha instances. However, this does not imply high availability.

When regular gateway instances and NFS-Ganesha instances overlap the same data resources, they will be accessible from both the standard S3 API and through the NFS-Ganesha instance as exported. You can co-locate the NFS-Ganesha instance with a Ceph Object Gateway instance on the same host.

Before you Start

1. Disable any running kernel NFS service instances on any host that will run NFS-Ganesha before attempting to run NFS-Ganesha. NFS-Ganesha will not start if another NFS instance is running.

2. As **root**, enable the Red Hat Ceph Storage Tools repository:

   Red Hat Enterprise Linux 7

   ```
   # subscription-manager repos --enable=rhel-7-server-rhceph-4-tools-rpms
   ```

   Red Hat Enterprise Linux 8

   ```
   # subscription-manager repos --enable=rhceph-4-tools-for-rhel-8-x86_64-rpms
   ```

3. Make sure that the **rpcbind** service is running:

   ```
   # systemctl start rpcbind
   ```

   NOTE

   The **rpcbind** package that provides **rpcbind** is usually installed by default. If that is not the case, install the package first.

   For details on how NFS uses **rpcbind**, see the Required Services section in the Storage Administration Guide for Red Hat Enterprise Linux 7.

4. If the **nfs-service** service is running, stop and disable it:

   ```
   # systemctl stop nfs-server.service
   # systemctl disable nfs-server.service
   ```

Configuring an NFS-Ganesha Instance

1. Install the **nfs-ganesha-rgw** package:

```
# yum install nfs-ganesha-rgw
```

2. Copy the Ceph configuration file from a Ceph Monitor node to the **/etc/ceph/** directory of the NFS-Ganesha host, and edit it as necessary:

```
# scp <mon-host>:/etc/ceph/ceph.conf <nfs-ganesha-rgw-host>:/etc/ceph
```

NOTE

The Ceph configuration file must contain a valid **[client.rgw.{instance-name}]** section and corresponding parameters for the various required Gateway configuration variables such as **rgw_data**, **keyring**, or **rgw_frontends**. If exporting Swift containers that do not conform to valid S3 bucket naming requirements, set **rgw_relaxed_s3_bucket_names** to **true** in the **[client.rgw]** section of the Ceph configuration file. For example, if a Swift container name contains underscores, it is not a valid S3 bucket name and will not get synchronized unless **rgw_relaxed_s3_bucket_names** is set to **true**. When adding objects and buckets outside of NFS, those objects will appear in the NFS namespace in the time set by **rgw_nfs_namespace_expire_secs**, which is about 5 minutes by default. Override the default value for **rgw_nfs_namespace_expire_secs** in the Ceph configuration file to change the refresh rate.

3. Open the NFS-Ganesha configuration file:

```
# vim /etc/ganesha/ganesha.conf
```

4. Configure the **EXPORT** section with an **FSAL** (File System Abstraction Layer) block. Provide an ID, S3 user ID, S3 access key, and secret. For NFSv4, it should look something like this:

```
EXPORT
{
        Export_ID={numeric-id};
        Path = "/";
        Pseudo = "/";
        Access_Type = RW;
        SecType = "sys";
        NFS_Protocols = 4;
        Transport_Protocols = TCP;
        Squash = No_Root_Squash;

        FSAL {
                Name = RGW;
                User_Id = {s3-user-id};
                Access_Key_Id ="{s3-access-key}";
                Secret_Access_Key = "{s3-secret}";
        }
}
```

The **Path** option instructs Ganesha where to find the export. For the VFS FSAL, this is the location within the server's namespace. For other FSALs, it may be the location within the

filesystem managed by that FSAL's namespace. For example, if the Ceph FSAL is used to export an entire CephFS volume, **Path** would be /.

The **Pseudo** option instructs Ganesha where to place the export within NFS v4's pseudo file system namespace. NFS v4 specifies the server may construct a pseudo namespace that may not correspond to any actual locations of exports, and portions of that pseudo filesystem may exist only within the realm of the NFS server and not correspond to any physical directories. Further, an NFS v4 server places all its exports within a single namespace. It is possible to have a single export exported as the pseudo filesystem root, but it is much more common to have multiple exports placed in the pseudo filesystem. With a traditional VFS, often the **Pseudo** location is the same as the **Path** location. Returning to the example CephFS export with / as the **Path**, if multiple exports are desired, the export would likely have something else as the **Pseudo** option. For example, /**ceph**.

Any **EXPORT** block which should support NFSv3 should include version 3 in the **NFS_Protocols** setting. Additionally, NFSv3 is the last major version to support the UDP transport. Early versions of the standard included UDP, but RFC 7530 forbids its use. To enable UDP, include it in the **Transport_Protocols** setting. For example:

```
EXPORT {
...
   NFS_Protocols = 3,4;
   Transport_Protocols = UDP,TCP;
...
}
```

Setting **SecType = sys;** allows clients to attach without Kerberos authentication.

Setting **Squash = No_Root_Squash;** enables a user to change directory ownership in the NFS mount.

NFS clients using a conventional OS-native NFS 4.1 client typically see a federated namespace of exported file systems defined by the destination server's **pseudofs** root. Any number of these can be Ceph Object Gateway exports.

Each export has its own tuple of **name**, **User_Id**, **Access_Key**, and **Secret_Access_Key** and creates a proxy of the object namespace visible to the specified user.

An export in **ganesha.conf** can also contain an **NFSV4** block. Red Hat Ceph Storage supports the **Allow_Numeric_Owners** and **Only_Numberic_Owners** parameters as an alternative to setting up the **idmapper** program.

```
NFSV4 {
   Allow_Numeric_Owners = true;
   Only_Numeric_Owners = true;
}
```

5. Configure an **NFS_CORE_PARAM** block.

```
NFS_CORE_PARAM{
   mount_path_pseudo = true;
}
```

When the **mount_path_pseudo** configuration setting is set to **true**, it will make the NFS v3 and NFS v4.x mounts use the same server side path to reach an export, for example:

```
mount -o vers=3 <IP ADDRESS>:/export /mnt
mount -o vers=4 <IP ADDRESS>:/export /mnt
```

```
Path          Pseudo        Tag    Mechanism   Mount
/export/test1  /export/test1  test1  v3 Pseudo   mount -o vers=3 server:/export/test1
/export/test1  /export/test1  test1  v3 Tag      mount -o vers=3 server:test1
/export/test1  /export/test1  test1  v4 Pseudo   mount -o vers=4 server:/export/test1
/             /export/ceph1  ceph1  v3 Pseudo   mount -o vers=3 server:/export/ceph1
/             /export/ceph1  ceph1  v3 Tag      mount -o vers=3 server:ceph1
/             /export/ceph1  ceph1  v4 Pseudo   mount -o vers=4 server:/export/ceph1
/             /export/ceph2  ceph2  v3 Pseudo   mount -o vers=3 server:/export/ceph2
/             /export/ceph2  ceph2  v3 Tag      mount -o vers=3 server:ceph2
/             /export/ceph2  ceph2  v4 Pseudo   mount -o vers=4
```

When the **mount_path_pseudo** configuration setting is set to **false**, NFS v3 mounts use the **Path** option and NFS v4.x mounts use the **Pseudo** option.

```
Path          Pseudo        Tag    Mechanism   Mount
/export/test1  /export/test1  test1  v3 Path     mount -o vers=3 server:/export/test1
/export/test1  /export/test1  test1  v3 Tag      mount -o vers=3 server:test1
/export/test1  /export/test1  test1  v4 Pseudo   mount -o vers=4 server:/export/test1
/             /export/ceph1  ceph1  v3 Path     mount -o vers=3 server:/
/             /export/ceph1  ceph1  v3 Tag      mount -o vers=3 server:ceph1
/             /export/ceph1  ceph1  v4 Pseudo   mount -o vers=4 server:/export/ceph1
/             /export/ceph2  ceph2  v3 Path     not accessible
/             /export/ceph2  ceph2  v3 Tag      mount -o vers=3 server:ceph2
/             /export/ceph2  ceph2  v4 Pseudo   mount -o vers=4 server:/export/ceph2
```

6. Configure the **RGW** section. Specify the name of the instance, provide a path to the Ceph configuration file, and specify any initialization arguments:

```
RGW {
    name = "client.rgw.{instance-name}";
    ceph_conf = "/etc/ceph/ceph.conf";
    init_args = "--{arg}={arg-value}";
}
```

7. Save the **/etc/ganesha/ganesha.conf** configuration file.

8. Enable and start the **nfs-ganesha** service.

```
# systemctl enable nfs-ganesha
# systemctl start nfs-ganesha
```

9. For very large pseudo directories, set the configurable parameter **rgw_nfs_s3_fast_attrs** to **true** in the **ceph.conf** file to make the namespace immutable and accelerated:

```
rgw_nfs_s3_fast_attrs= true
```

10. Restart the Ceph Object Gateway service from each gateway node:

```
# systemctl restart ceph-radosgw.target
```

Configuring NFSv4 clients

To access the namespace, mount the configured NFS-Ganesha export(s) into desired locations in the local POSIX namespace. As noted, this implementation has a few unique restrictions:

- Only the NFS 4.1 and higher protocol flavors are supported.
- To enforce write ordering, use the **sync** mount option.

To mount the NFS-Ganesha exports, add the following entry to the **/etc/fstab** file on the client host:

```
<ganesha-host-name>:/ <mount-point> nfs noauto,soft,nfsvers=4.1,sync,proto=tcp 0 0
```

Specify the NFS-Ganesha host name and the path to the mount point on the client.

NOTE

To successfully mount the NFS-Ganesha exports, the **/sbin/mount.nfs** file must exist on the client. The **nfs-tools** package provides this file. In most cases, the package is installed by default. However, verify that the **nfs-tools** package is installed on the client and if not, install it.

For additional details on NFS, see the Network File System (NFS) chapter in the Storage Administration Guide for Red Hat Enterprise Linux 7.

Configuring NFSv3 clients

Linux clients can be configured to mount with NFSv3 by supplying **nfsvers=3** and **noacl** as mount options. To use UDP as the transport, add **proto=udp** to the mount options. However, TCP is the preferred protocol.

```
<ganesha-host-name>:/ <mount-point> nfs noauto,noacl,soft,nfsvers=3,sync,proto=tcp 0 0
```

NOTE

Configure the NFS Ganesha **EXPORT** block **Protocols** setting with version 3 and the **Transports** setting with UDP if the mount will use version 3 with UDP.

Since NFSv3 does not communicate client OPEN and CLOSE operations to file servers, RGW NFS cannot use these operations to mark the beginning and ending of file upload transactions. Instead, RGW NFS attempts to start a new upload when the first write is sent to a file at offset 0, and finalizes the upload when no new writes to the file have been seen for a period of time—by default, 10 seconds. To change this value, set a value for **rgw_nfs_write_completion_interval_s** in the RGW section(s) of the Ceph configuration file.

CHAPTER 3. ADMINISTRATION

Administrators can manage the Ceph Object Gateway using the **radosgw-admin** command-line interface.

- Administrative Data Storage
- Storage Policies
- Indexless Buckets
- Bucket Sharding
- Compression
- User Management
- Quota Management
- Usage
- Bucket management
- Bucket lifecycle
- Ceph Object Gateway data layout
- Object Gateway data layout parameters
- Session tags for Attribute-based access control (ABAC) in STS
- Optimize the Ceph Object Gateway's garbage collection
- Optimize the Ceph Object Gateway's data object storage
- The Ceph Object Gateway and multi-factor authentication
- Removing Ceph Object Gateway using Ansible

3.1. ADMINISTRATIVE DATA STORAGE

A Ceph Object Gateway stores administrative data in a series of pools defined in an instance's zone configuration. For example, the buckets, users, user quotas and usage statistics discussed in the subsequent sections are stored in pools in the Ceph Storage Cluster. By default, Ceph Object Gateway will create the following pools and map 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**

You should consider creating these pools manually so that 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 will often use the same CRUSH ruleset and use fewer placement groups, because there are 10 pools for the administrative data. See Pools and the *Storage Strategies* guide for Red Hat Ceph Storage 4 for additional details.

Also see Ceph Placement Groups (PGs) per Pool Calculator for placement group calculation details. The **mon_pg_warn_max_per_osd** setting warns you if assign too many placement groups to a pool (i.e., 300 by default). You may adjust the value to suit your needs and the capabilities of your hardware where **n** is the maximum number of PGs per OSD.

```
mon_pg_warn_max_per_osd = n
```

3.2. 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, for example, SSDs, SAS drives, SATA drives. A particular way of ensuring durability, replication, erasure coding, and so on. For details, see the *Storage Strategies* guide for Red Hat Ceph Storage 4.

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, for example, **zonegroup.json**:

   Syntax

   ```
   [root@master-zone]# radosgw-admin zonegroup --rgw-zonegroup=<zonegroup_name> get > zonegroup.json
   ```

   Example

   ```
   [root@master-zone]# radosgw-admin zonegroup --rgw-zonegroup=default get > zonegroup.json
   ```

3. Add a **special-placement** entry under **placement_target** in the **zonegroup.json** file.

   ```
   {
    "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": 11
 }],
 "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:

```
[root@master-zone]# radosgw-admin zonegroup set < zonegroup.json
```

5. Get the zone configuration and store it in a file, for example, **zone.json**:

```
[root@master-zone]# radosgw-admin zone get > zone.json
```

6. Edit the zone file and add the new placement policy key under **placement_pool**:

```
{
 "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.

```
[root@master-zone]# radosgw-admin zone set < zone.json
```

8. Update the zone group map.

```
[root@master-zone]# radosgw-admin period update --commit
```

The **special-placement** entry is listed as a **placement_target**.

To specify the storage policy when making a request:

Example:

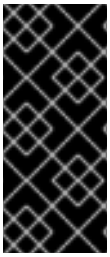
```
$ curl -i http://10.0.0.1/swift/v1/TestContainer/file.txt -X PUT -H "X-Storage-Policy: special-placement"
-H "X-Auth-Token: AUTH_rgwtxxxxxx"
```

3.3. 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 provides a mechanism in which the placement target does not track objects in specific buckets. This 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.

WARNING

It has been observed that the Ceph Object Gateway daemon crashes when performing operations on a bucket having indexless placement policy. Hence, Red Hat does not recommend having this placement policy.

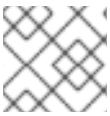


IMPORTANT

The bucket index will not reflect the correct state of the bucket, and listing these buckets will not correctly return their list of objects. This affects multiple features. Specifically, these buckets will not be synced in a multi-zone environment because the bucket index is not used to store change information. Red Hat recommends 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.



NOTE

Objects in indexless buckets cannot be listed from NFS.

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. Add a new placement target to the zonegroup:

   Example

   ```
   [root@rgw ~]# radosgw-admin zonegroup placement add --rgw-zonegroup="default" \
     --placement-id="indexless-placement"
   ```

2. Add a new placement target to the zone:

   Example

   ```
   [root@rgw ~]# 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 zonegroup's default placement to **indexless-placement**:

   Example

   ```
   [root@rgw ~]# radosgw-admin zonegroup placement default --placement-id "indexless-placement"
   ```

   In this example, the buckets created in the **indexless-placement** target will be indexless buckets.

4. Update and commit the period if the cluster is in a multi-site configuration:

   Example

   ```
   [root@rgw ~]# radosgw-admin period update --commit
   ```

5. Restart the Ceph Object Gateway daemon for the change to take effect:

   Example

   ```
   [root@rgw ~]# systemctl restart ceph-radosgw.target
   ```

3.4. CONFIGURING BUCKET SHARDING

The Ceph Object Gateway stores bucket index data in the index pool (**index_pool**), which defaults to **.rgw.buckets.index**. When the client puts many objects—hundreds of thousands to millions of objects—in a single bucket without having set quotas for the maximum number of objects per bucket, the index pool can suffer significant performance degradation.

Bucket index sharding helps prevent performance bottlenecks when allowing a high number of objects per bucket. Starting with Red Hat Ceph Storage 4.1, default number of bucket index shards, **bucket_index_max_shards**, has been changed from 1 to 11. This change increases the amount of write throughput for small buckets, and delays the onset of dynamic resharding. This change affects only the new buckets and deployments.

Red Hat recommends 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 evenly distributing bucket index entries across the shards. For example, 7001 bucket index shards is better than 7000 since the former is prime.

To configure bucket index sharding:

- For new buckets in simple configurations, use the **rgw_override_bucket_index_max_shards** option. See Section 3.4.3, “Configuring Bucket Index Sharding in Simple Configurations”
- For new buckets in multi-site configurations, use the **bucket_index_max_shards** option. See Section 3.4.4, “Configuring Bucket Index sharding in Multisite Configurations”

To reshard a bucket:

- Dynamically, see Section 3.4.5, “Dynamic Bucket Index Resharding”
- Manually, see Section 3.4.6, “Manual Bucket Index Resharding”
- Manually, in multi-site configurations, see *Manually Resharding Buckets with Multi-site*

3.4.1. Bucket sharding limitations

IMPORTANT

Use the following limitations with caution. There are implications related to your hardware selections, so you should always discuss these requirements with your Red Hat account team.

Maximum number of objects in one bucket before it needs sharding

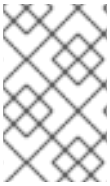
Red Hat recommends a maximum of 102,400 objects per bucket index shard. To take full advantage of sharding, provide a sufficient number of OSDs in the Ceph Object Gateway bucket index pool to get maximum parallelism.

NOTE

Ceph OSDs currently warn when any key range in indexed storage exceeds 200,000. As a consequence, if you approach the number of 200,000 objects per shard, you will get such warnings. In some setups, the value might be larger, and is adjustable.

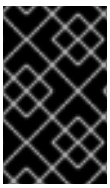
Maximum number of objects when using sharding

The default number of bucket index shards for dynamic bucket resharding is 1999. You can change this value up to 65521 shards. A value of 1999 bucket index shards gives 204697600 total objects in the bucket, a value of 65521 shards gives 6709350400 objects.



NOTE

Based on prior testing, the maximum number of bucket index shards currently supported is 65521. Red Hat quality assurance has NOT performed full scalability testing on bucket sharding.



IMPORTANT

If the number of bucket index shards exceeds 1999, ordinary S3 clients might not be able to list bucket contents. The custom clients can ask for unordered listing, which scales to any number of shards.

3.4.2. Bucket lifecycle parallel thread processing

A new feature in Red Hat Ceph Storage 4.1 allows for parallel thread processing of bucket lifecycles. This parallelization scales with the number of Ceph Object Gateway instance, and replaces the in-order index shard enumeration with a number sequence. The default locking timeout has been extended from 60 seconds to 90 seconds. New tunable options have been added to tune lifecycle worker threads to run in parallel for each Ceph Object Gateway instance.

rgw_lc_max_worker

This option specifies the number of lifecycle worker thread to run in parallel, thereby processing bucket and index shards simultaneously. The default value for the **rgw_lc_max_worker** option is **3**.

rgw_lc_max_wp_worker

This option specifies the number of threads in each lifecycle worker's work pool. This option can help accelerate processing each bucket. The default value for the **rgw_lc_max_wp_worker** option is **3**.

Additional Resources

- See the *The Ceph configuration file* section in the *Red Hat Ceph Storage Configuration Guide* for more details.

3.4.3. Configuring Bucket Index Sharding in Simple Configurations

To enable and configure bucket index sharding on all new buckets, use the **rgw_override_bucket_index_max_shards** parameter. Set the parameter to:

- **0** to disable bucket index sharding. This is the default value.
- A value greater than **0** to enable bucket sharding and to set the maximum number of shards.

Prerequisites

- Read the bucket sharding limitations.

Procedure

1. Calculate the recommended number of shards. To do so, use the following formula:

```
number of objects expected in a bucket / 100,000
```

   Note that maximum number of shards is 65521.

2. Add **rgw_override_bucket_index_max_shards** to the Ceph configuration file:

```
rgw_override_bucket_index_max_shards = value
```

   Replace *value* with the recommended number of shards calculated in the previous step, for example:

```
rgw_override_bucket_index_max_shards = 12
```

   - To configure bucket index sharding for all instances of the Ceph Object Gateway, add **rgw_override_bucket_index_max_shards** under the **[global]** section.
   - To configure bucket index sharding only for a particular instance of the Ceph Object Gateway, add **rgw_override_bucket_index_max_shards** under the instance.

3. Restart the Ceph Object Gateway:

```
# systemctl restart ceph-radosgw.target
```

Additional resources

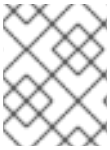
- *Dynamic Bucket Index Resharding*
- *Manual Bucket Index Resharding*

3.4.4. Configuring Bucket Index sharding in Multisite Configurations

In multisite configurations, 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** setting in the configuration for that zone group. Set the parameter to:

Set the parameter to:

- **0** to disable bucket index sharding, the default value of **bucket_index_max_shards** is **11**.
- 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.

Prerequisites

- Read the bucket sharding limitations.

Procedure

1. Calculate the recommended number of shards. To do so, use the following formula:

```
number of objects expected in a bucket / 100,000
```

   Note that maximum number of shards is 65521.

2. Extract the zone group configuration to the **zonegroup.json** file:

```
$ radosgw-admin zonegroup get > zonegroup.json
```

3. In the **zonegroup.json** file, set the **bucket_index_max_shards** setting for each named zone.

```
bucket_index_max_shards = VALUE
```

   Replace *value* with the recommended number of shards calculated in the previous step, for example:

```
bucket_index_max_shards = 12
```

4. Reset the zone group:

```
$ radosgw-admin zonegroup set < zonegroup.json
```

5. Update the period:

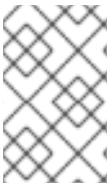
```
$ radosgw-admin period update --commit
```

Additional resources

- *Manually Resharding Buckets with Multisite*

3.4.5. Dynamic Bucket Index Resharding

The process for dynamic bucket resharding periodically checks all the Ceph Object Gateway buckets and 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 100k objects per shard.



NOTE

The default value is based on experience with bucket indexes stored on spinning disk. In more modern setups with bucket indexes on flash media, the value for maximum objects per bucket index shard might be higher.

NOTE

Dynamic bucket index resharding 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.

Prerequisites

- Read the bucket sharding limitations.

Procedure

- To enable dynamic bucket index resharding:
  1. Set the **rgw_dynamic_resharding** setting in the Ceph configuration file to **true**, which is the default value.
  2. *Optional*. Change the following parameters in the Ceph configuration file if needed:
     - **rgw_reshard_num_logs**: The number of shards for the resharding log. The default value is **16**.
     - **rgw_reshard_bucket_lock_duration**: The duration of the lock on a bucket during resharding. The default value is **360** seconds.
     - **rgw_dynamic_resharding**: Enables or disables dynamic resharding. The default value is **true**.
     - **rgw_max_objs_per_shard**: The maximum number of objects per shard. The default value is **100000** objects per shard.
     - **rgw_reshard_thread_interval**: The maximum time between rounds of reshard thread processing. The default value is **600** seconds.
- To add a bucket to the resharding queue:

  ```
  radosgw-admin reshard add --bucket bucket --num-shards number
  ```

  Replace:
  - *bucket* with the name of the bucket to reshard
  - *number* with the new number of shards

  For example:

  ```
  $ radosgw-admin reshard add --bucket data --num-shards 10
  ```

- To list the resharding queue:

```
$ radosgw-admin reshard list
```

- To check bucket resharding status:

```
radosgw-admin reshard status --bucket bucket
```

Replace:

  - *bucket* with the name of the bucket to reshard

For example:

```
$ radosgw-admin reshard status --bucket data
```

- To process entries on the resharding queue immediately:

```
$ radosgw-admin reshard process
```

- To cancel pending bucket resharding:

```
radosgw-admin reshard cancel --bucket bucket
```

Replace:

  - *bucket* with the name of the pending bucket

For example:

```
$ radosgw-admin reshard cancel --bucket data
```

IMPORTANT

You can only cancel **pending** resharding operations. Do not cancel **ongoing** resharding operations.

- If you use Red Hat Ceph Storage 3.1 and previous versions, remove stale bucket entries as described in the *Cleaning stale instances after resharding* section.

Additional resources

- *Manual Bucket Index Resharding*
- *Configuring Bucket Index Sharding in Simple Configurations*

3.4.6. Manual Bucket Index Resharding

If a bucket has grown larger than the initial configuration was optimized for, reshard the bucket index pool by using the **radosgw-admin bucket reshard** command. This command:

- 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.

IMPORTANT

Use this procedure only in simple configurations. To reshard buckets in multi-site configurations, see *Manually Resharding Buckets with Multi-site* .

Prerequisites

- Read the bucket sharding limitations .

Procedure

1. Back the original bucket index up:

```
radosgw-admin bi list --bucket=bucket > bucket.list.backup
```

   Replace:

   - *bucket* with the name of the bucket to reshard

   For example, for a bucket named **data**, enter:

```
$ radosgw-admin bi list --bucket=data > data.list.backup
```

2. Reshard the bucket index:

```
radosgw-admin bucket reshard --bucket=bucket
--num-shards=number
```

   Replace:

   - *bucket* with the name of the bucket to reshard
   - *number* with the new number of shards

   For example, for a bucket named **data** and the required number of shards being 100, enter:

```
$ radosgw-admin bucket reshard --bucket=data
--num-shards=100
```

3. If you use Red Hat Ceph Storage 3.1 and previous versions, remove stale bucket entries as described in the *Cleaning stale instances after resharding* section.

Additional Resources

- *Dynamic Bucket Index Resharding*
- *Configuring Bucket Index Sharding in Simple Configurations*

- *Configuring Bucket Index Sharding in Multi-site Configurations*
- *Manually Resharding Buckets with Multi-site*

3.4.7. Cleaning stale instances after resharding

In Red Hat Ceph Storage 3.1 and previous versions, the resharding process does not clean stale instances of bucket entries automatically. These stale instances can impact performance of the cluster if they are not cleaned manually.

IMPORTANT

Use this procedure only in simple configurations not in multi-site clusters.

Prerequisites

- Ceph Object Gateway installed.

Procedure

1. List stale instances:

```
$ radosgw-admin reshard stale-instances list
```

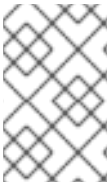
2. Clean the stale instances:

```
$ radosgw-admin reshard stale-instances rm
```

3.5. ENABLING COMPRESSION

The Ceph Object Gateway supports server-side compression of uploaded objects using any of Ceph's compression plugins. These include:

- **zlib**: Supported.
- **snappy**: Technology Preview.
- **zstd**: Technology Preview.



NOTE

The **snappy** and **zstd** compression plugins are Technology Preview features and as such they are not fully supported, as Red Hat has not completed quality assurance testing on them yet.

Configuration

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 recompress 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**.

For example:

```
$ 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 the *Ceph Object Gateway for Production* guide, more specifically, the *Creating a Realm* section first. See also Multisite.

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 a given bucket.

```
$ radosgw-admin bucket stats --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_utilized** and **size_kb_utilized** fields represent the total size of compressed data in bytes and kilobytes respectively.

3.6. 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 execute 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. This document uses **[root@master-zone]#** as a command line convention for a host in the master zone of the master 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). For example:

```
[root@master-zone]# radosgw-admin user <create|modify|info|rm|suspend|enable|check|stats> <--
uid={id}|--subuser={name}> [other-options]
```

Additional options may be required depending on the command you execute.

3.6.1. Multi Tenancy

In Red Hat Ceph Storage 2 and later, 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", "main" and so forth.

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 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, execute the following:

```
[root@master-zone]# 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, execute one of the following:

```
[root@master-zone]# radosgw-admin --tenant testx --uid tester \
                --display-name "Test User" --subuser tester:swift \
                --key-type swift --access full subuser create

[root@master-zone]# 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.

3.6.2. 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.

```
[root@master-zone]# radosgw-admin user create --uid=<id> \
[--key-type=<type>] [--gen-access-key|--access-key=<key>]\
[--gen-secret | --secret=<key>] \
[--email=<email>] --display-name=<name>
```

For example:

```
[root@master-zone]# radosgw-admin user create --uid=janedoe --display-name="Jane Doe" --
email=jane@example.com
```

```
{ "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 and ensuring that it does not have a JSON escape character or specify the key and secret manually.

3.6.3. 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** 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.

NOTE

full is not **readwrite**, as it also includes the access control policy.

```
[root@master-zone]# radosgw-admin subuser create --uid={uid} --subuser={uid} --access=[ read |
write | readwrite | full ]
```

For example:

```
[root@master-zone]# 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": []}
```

3.6.4. Get User Information

To get information about a user, specify **user info** and the user ID (**--uid={username}**).

```
[root@master-zone]# 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@master-zone]# radosgw-admin user info --uid=janedoe --tenant=test
```

3.6.5. Modify User Information

To modify information about a user, you must specify the user ID (**--uid={username}**) and the attributes you want to modify. Typical modifications are to keys and secrets, email addresses, display names and access levels. For example:

```
[root@master-zone]# radosgw-admin user modify --uid=janedoe --display-name="Jane E. Doe"
```

To modify subuser values, specify **subuser modify** and the subuser ID. For example:

```
[root@master-zone]# radosgw-admin subuser modify --subuser=janedoe:swift --access=full
```

3.6.6. 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@master-zone]# radosgw-admin user suspend --uid=johndoe
```

To re-enable a suspended user, specify **user enable** and the user ID. :

```
[root@master-zone]# radosgw-admin user enable --uid=johndoe
```

NOTE

Disabling the user disables the subuser.

3.6.7. Remove a User

When you remove a user, the user and subuser are removed from the system. However, you may remove just the subuser if you wish. To remove a user (and subuser), specify **user rm** and the user ID.

```
[root@master-zone]# radosgw-admin user rm --uid=<uid> [--purge-keys] [--purge-data]
```

For example:

```
[root@master-zone]# radosgw-admin user rm --uid=johndoe --purge-data
```

To remove the subuser only, specify **subuser rm** and the subuser name.

```
[root@master-zone]# radosgw-admin subuser rm --subuser=johndoe:swift --purge-keys
```

Options include:

- **Purge Data:** The **--purge-data** option purges all data associated to the UID.
- **Purge Keys:** The **--purge-keys** option purges all keys associated to the UID.

3.6.8. Remove a Subuser

When you remove a sub user, you are removing access to the Swift interface. The user will remain in the system. The Ceph Object Gateway To remove the subuser, specify **subuser rm** and the subuser ID.

```
[root@master-zone]# radosgw-admin subuser rm --subuser=johndoe:test
```

Options include:

- **Purge Keys:** The **--purge-keys** option purges all keys associated to the UID.

3.6.9. Rename a User

To change a 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 to use the command in the **Screen** utility provided by the **screen** package.

Prerequisites

- A working Ceph cluster
- **root** or **sudo** access
- Installed Ceph Object Gateway

Procedure

1. Rename a user:

```
radosgw-admin user rename --uid=current-user-name --new-uid=new-user-name
```

For example, to rename **user1** to **user2**:

```
# 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
```

For example, to rename **user1** to **user2** inside a **test** tenant:

Example

```
# 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
   ```

   For example:

   Example

   ```
   # radosgw-admin user info --uid=user2
   ```

   If a user is inside a tenant, use the *tenant*$*user-name* format:

   ```
   radosgw-admin user info --uid=tenant$new-user-name
   ```

   ```
   # radosgw-admin user info --uid=test$user2
   ```

Additional Resources

- The **screen(1)** manual page

3.6.10. 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 subuser, you must specify the subuser ID and the **swift** keytype. For example:

```
[root@master-zone]# 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"}]}
```

3.6.11. 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=<key>** specifies a secret key (e.g,. manually generated).
- **--gen-access-key** generates random access key (for S3 user by default).
- **--gen-secret** generates a random secret key.
- **--key-type=<type>** specifies a key type. The options are: swift, s3

To add a key, specify the user:

```
[root@master-zone]# radosgw-admin key create --uid=johndoe --key-type=s3 --gen-access-key --
gen-secret
```

You may 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:

   ```
   [root@master-zone]# radosgw-admin user info --uid=<testid>
   ```

   The access key is the **"access_key"** value in the output, for example:

   ```
   $ 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:

   ```
   [root@master-zone]# radosgw-admin key rm --uid=<user_id> --access-key <access_key>
   ```

   For example:

   ```
   [root@master-zone]# radosgw-admin key rm --uid=johndoe --access-key
   0555b35654ad1656d804
   ```

3.6.12. Add and Remove Admin Capabilities

The Ceph Storage Cluster provides an administrative API that enables users to execute 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, execute the following:

```
[root@master-zone]# radosgw-admin caps add --uid={uid} --caps={caps}
```

You can add read, write or all capabilities to users, buckets, metadata and usage (utilization). For example:

```
--caps="[users|buckets|metadata|usage|zone]=[*|read|write|read, write]"
```

For example:

```
[root@master-zone]# radosgw-admin caps add --uid=johndoe --caps="users=*"
```

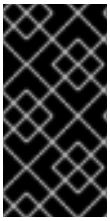
To remove administrative capabilities from a user, execute the following:

```
[root@master-zone]# radosgw-admin caps remove --uid=johndoe --caps={caps}
```

3.7. 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. See Section 3.4, “Configuring Bucket sharding” for details.

3.7.1. Set User Quotas

Before you enable a quota, you must first set the quota parameters. For example:

```
[root@master-zone]# radosgw-admin quota set --quota-scope=user --uid=<uid> [--max-objects=
<num objects>] [--max-size=<max size>]
```

For example:

```
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.

3.7.2. Enable and Disable User Quotas

Once you set a user quota, you may enable it. For example:

```
[root@master-zone]# radosgw-admin quota enable --quota-scope=user --uid=<uid>
```

You may disable an enabled user quota. For example:

```
[root@master-zone]# radosgw-admin quota disable --quota-scope=user --uid=<uid>
```

3.7.3. Set bucket quotas

Bucket quotas apply to the buckets owned by the specified **uid**. They 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.

3.7.4. Enable and Disable Bucket Quotas

Once you set a bucket quota, you may enable it. For example:

```
[root@master-zone]# radosgw-admin quota enable --quota-scope=bucket --uid=<uid>
```

You may disable an enabled bucket quota. For example:

```
[root@master-zone]# radosgw-admin quota disable --quota-scope=bucket --uid=<uid>
```

3.7.5. Get Quota Settings

You may access each user's quota settings via the user information API. To read user quota setting information with the CLI interface, execute the following:

```
# radosgw-admin user info --uid=<uid>
```

To get quota settings for a tenanted user, specify the user ID and the name of the tenant:

```
+ radosgw-admin user info --uid=_user-id_ --tenant=_tenant_
```

3.7.6. 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.

```
[root@master-zone]# radosgw-admin user stats --uid=<uid> --sync-stats
```

3.7.7. Get User Quota Usage Stats

To see how much of the quota a user has consumed, execute the following:

```
# radosgw-admin user stats --uid=<uid>
```

NOTE

You should execute **radosgw-admin user stats** with the **--sync-stats** option to receive the latest data.

3.7.8. 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 this 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. See Chapter 4, *Configuration Reference* for more details on these options.

3.7.9. Reading and Writing Global Quotas

You can read and write quota settings in a zonegroup map. To get a zonegroup map:

```
[root@master-zone]# 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@master-zone]# radosgw-admin global quota set --quota-scope bucket --max-objects 1024
[root@master-zone]# 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.

3.8. USAGE

The Ceph Object Gateway logs usage for each user. You can track user usage within date ranges too.

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.

3.8.1. 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.

```
# radosgw-admin usage show \
        --uid=johndoe --start-date=2012-03-01 \
        --end-date=2012-04-01
```

You may also show a summary of usage information for all users by omitting a user ID.

```
# radosgw-admin usage show --show-log-entries=false
```

3.8.2. 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.

```
[root@master-zone]# radosgw-admin usage trim --start-date=2010-01-01 \
          --end-date=2010-12-31

[root@master-zone]# radosgw-admin usage trim --uid=johndoe
[root@master-zone]# radosgw-admin usage trim --uid=johndoe --end-date=2013-12-31
```

3.9. BUCKET MANAGEMENT

As a storage administrator, when using the Ceph Object Gateway you can manage buckets by moving them between users and renaming them. Also, you can find orphan or leaky objects within the Ceph Object Gateway that can occur over the lifetime of a storage cluster.

3.9.1. 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

3.9.1.1. Prerequisites

- A running Red Hat Ceph Storage cluster
- Ceph Object Gateway is installed
- A bucket
- Various tenanted and non-tenanted users

3.9.1.2. 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 do so, 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:

```
radosgw-admin bucket link --uid=user --bucket=bucket
```

Replace:

- *user* with the user name of the user to link the bucket to
- *bucket* with the name of the bucket

For example, to link the **data** bucket to the user named **user2**:

```
# radosgw-admin bucket link --uid=user2 --bucket=data
```

2. Verify that the bucket has been linked to **user2** successfully:

```
# radosgw-admin bucket list --uid=user2
[
    "data"
]
```

3. Change the ownership of the bucket to the new user:

```
radosgw-admin bucket chown --uid=user --bucket=bucket
```

Replace:

- *user* with the user name of the user to change the bucket ownership to
- *bucket* with the name of the bucket

For example, to change the ownership of the **data** bucket to **user2**:

```
# 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:

```
# radosgw-admin bucket list --bucket=data
```

3.9.1.3. Moving buckets between tenanted users

You can move buckets between one tenanted user to another.

Procedure

1. Link the bucket to a new user:

```
radosgw-admin bucket link --bucket=current-tenant/bucket --uid=new-tenant$user
```

Replace:

- *current-tenant* with the name of the tenant the bucket is
- *bucket* with the name of the bucket to link
- *new-tenant* with the name of the tenant where the new user is
- *user* with the user name of the new user

For example, to link the **data** bucket from the **test** tenant to the user named **user2** in the **test2** tenant:

```
# radosgw-admin bucket link --bucket=test/data --uid=test2$user2
```

2. Verify that the bucket has been linked to **user2** successfully:

```
# radosgw-admin bucket list --uid=test$user2
[
    "data"
]
```

3. Change the ownership of the bucket to the new user:

```
radosgw-admin bucket chown --bucket=new-tenant/bucket --uid=new-tenant$user
```

Replace:

- *bucket* with the name of the bucket to link
- *new-tenant* with the name of the tenant where the new user is
- *user* with the user name of the new user

For example, to change the ownership of the **data** bucket to the **user2** inside the **test2** tenant:

```
# 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:

```
# radosgw-admin bucket list --bucket=test2/data
```

3.9.1.4. 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:
   Open and edit the Ceph configuration file, by default **/etc/ceph/ceph.conf**. Enable the **rgw_keystone_implicit_tenants** option:

   ```
   rgw_keystone_implicit_tenants = true
   ```

   Access the Ceph Object Gateway from an eternal tenant using either the **s3cmd** or **swift** command:

   ```
   # swift list
   ```

   Or use **s3cmd**:

   ```
   # s3cmd ls
   ```

   The first access from an external tenant creates an equivalent Ceph Object Gateway user.

2. Move a bucket to a tenanted user:

   ```
   radosgw-admin bucket link --bucket=/bucket --uid='tenant$user'
   ```

   Replace:

   - *bucket* with the name of the bucket
   - *tenant* with the name of the tenant where the new user is
   - *user* with the user name of the new user

   For example, to move the **data** bucket to the **tenanted-user** inside the **test** tenant:

   ```
   # radosgw-admin bucket link --bucket=/data --uid='test$tenanted-user'
   ```

3. Verify that the **data** bucket has been linked to **tenanted-user** successfully:

   ```
   # radosgw-admin bucket list --uid='test$tenanted-user'
   [
       "data"
   ]
   ```

4. Change the ownership of the bucket to the new user:

```
radosgw-admin bucket chown --bucket='tenant/bucket name' --uid='tenant$user'
```

Replace:

- *bucket* with the name of the bucket
- *tenant* with the name of the tenant where the new user is
- *user* with the user name of the new user

For example, to change the ownership of the **data** bucket to **tenanted-user** that is inside the **test** tenant:

```
# 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:

```
# radosgw-admin bucket list --bucket=test/data
```

3.9.2. Renaming buckets

You can rename buckets.

Prerequisites

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

Procedure

1. List the buckets:

```
radosgw-admin bucket list
```

For example, note a bucket from the output:

```
# 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:

```
radosgw-admin bucket link --bucket=original-name --bucket-new-name=new-name --
uid=user-ID
```

For example, to rename the **s3bucket1** bucket to **s3newb**:

```
# radosgw-admin bucket link --bucket=s3bucket1 --bucket-new-name=s3newb --uid=testuser
```

If the bucket is inside a tenant, specify the tenant as well:

```
radosgw-admin bucket link --bucket=tenant/original-name --bucket-new-name=new-name --
uid=tenant$user-ID
```

For example:

```
# radosgw-admin bucket link --bucket=test/s3bucket1 --bucket-new-name=s3newb --
uid=test$testuser
```

3. Verify the bucket was renamed:

```
radosgw-admin bucket list
```

For example, a bucket named **s3newb** exists now:

```
# radosgw-admin bucket list
[
    "34150b2e9174475db8e191c188e920f6/swcontainer",
    "34150b2e9174475db8e191c188e920f6/swimpfalse",
    "c278edd68cfb4705bb3e07837c7ad1a8/ec2container",
    "s3newb",
    "c278edd68cfb4705bb3e07837c7ad1a8/demoten1",
    "c278edd68cfb4705bb3e07837c7ad1a8/demo-ct",
    "c278edd68cfb4705bb3e07837c7ad1a8/demopostup",
    "34150b2e9174475db8e191c188e920f6/postimpfalse",
    "c278edd68cfb4705bb3e07837c7ad1a8/demoten2",
    "c278edd68cfb4705bb3e07837c7ad1a8/postupsw"
]
```

3.9.3. Finding orphan and leaky objects

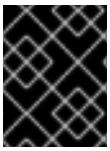
A healthy storage cluster does not have any orphan or leaky objects, but in some cases orphan or leaky objects can occur. For example, if the Ceph Object Gateway goes down in the middle of an operation, this can cause some objects to become orphans. Also, an undiscovered bug can cause orphan objects to occur.

Starting with Red Hat Ceph Storage 4.1, storage administrators can see how the Ceph Object Gateway objects map to the RADOS objects. The **radosgw-admin** command provides you a new tool to search for and produce a list of these potential orphan or leaky objects. Using the **radoslist** subcommand will

display objects stored within buckets, or all buckets in the storage cluster. The **rgw-orphan-list** script will display orphan objects within a pool.

WARNING

The **rgw-orphan-list** command is still experimental. Cautiously and carefully evaluate the objects listed by it before removing any using the **rados rm** command.



IMPORTANT

The **radoslist** subcommand is replacing the deprecated **orphans find** and **orphans finish** subcommands.

Prerequisites

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

Procedure

1. To generate a list of objects that hold data within a bucket:

   Syntax

   ```
   radosgw-admin bucket radoslist --bucket BUCKET_NAME
   ```

   Example

   ```
   [root@rgw ~]# radosgw-admin bucket radoslist --bucket mybucket
   ```

   NOTE

   If the *BUCKET_NAME* is omitted, then all objects in all buckets are displayed.

2. To generate a list of orphans for a pool:

   ```
   [root@rgw ~]# 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.

3. Review the orphan objects in the list.
4. To remove orphan objects:

Syntax

```
rados -p POOL_NAME rm OBJECT_NAME
```

Example

```
[root@rgw ~]# rados -p default.rgw.buckets.data rm myobject
```

WARNING

Verify you are removing the correct objects. Executing the **rados rm** command will remove data from the storage cluster.

Additional Resources

- See the *Finding Orphan Objects* section in the *Red Hat Ceph Storage 3 Object Gateway Administration Guide* for more details on the legacy **radosgw-admin orphans find** subcommand.

3.9.4. Managing bucket index entries

You can manage the bucket index entries of the Ceph Object Gateway in a Red Hat Ceph Storage cluster using the **radosgw-admin bucket check** subcommand.

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** subcommand, 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 a section of the output.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- A running Ceph Object Gateway.
- A bucket created.

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
   ```

3.9.5. 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 are enabled by default. If **rgw_enable_apis** configuration parameter is explicitly set, ensure that **s3** and **pubsub** are included. To verify this, run **ceph config get mon.* rgw_enable_apis** command.

Additional Resources

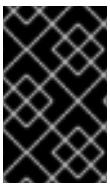
- See the *Red Hat Ceph Storage Developer Guide* for more details on the bucket notification REST API.

3.9.6. Creating bucket notifications

Create bucket notifications at the bucket level. The notification configuration has the Red Hat Ceph Storage Object Gateway S3 events, **ObjectCreated** and **ObjectRemoved**. These need to be published and the destination to send the bucket notifications. Bucket notifications are S3 operations.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- A running HTTP server, RabbitMQ server, or a Kafka server.
- Root-level access.
- Installation of the Red Hat Ceph Storage Object Gateway.
- 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**.

Procedure

1. Create a s3 bucket.
2. Create a SNS topic for **http**,**amqp** or **kafka** protocol.
3. Create a s3 bucket notification for **s3:objectCreate** and **s3:objectRemove** events:

Example

```
client.put_bucket_notification_configuration(
  Bucket=bucket_name,
  NotificationConfiguration={
    'TopicConfigurations': [
      {
        'Id': notification_name,
```

```
        'TopicArn': topic_arn,
        'Events': ['s3:ObjectCreated:*', 's3:ObjectRemoved:*']
    }]})
```

4. Create s3 objects in the bucket.
5. Verify the object creation events at the **http** or **rabbitmq** or **kafka** receiver.
6. Delete the objects.
7. Verify the object deletion events at the **http** or **rabbitmq** or **kafka** receiver.

3.9.7. Additional Resources

- See the *Using Keystone to Authenticate Ceph Object Gateway Users* for more information.
- See the *Red Hat Ceph Storage Developer Guide* for more information.

3.10. 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.

NOTE

The **radosgw-admin lc reshard** command is deprecated in Red Hat Ceph Storage 3.3 and not supported in Red Hat Ceph Storage 4 and later releases.

3.10.1. Creating a lifecycle management policy

You can manage 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 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 that are more than 1 day old.

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.

Verification

- 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

  ```
  [root@rgw ~]# 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
  }
 }
  ]
}
```

Additional Resources

- See the *S3 bucket lifecycle* section in the *Red Hat Ceph Storage Developer Guide* for details.
- For a more information on using the **AWS CLI** to manage lifecycle configurations, consult the *Setting lifecycle configuration on a bucket* section of the *Amazon Simple Storage Service* documentation.

3.10.2. 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 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

  ```
  [root@rgw ~]# radosgw-admin lc get --bucket=testbucket
  ```

NOTE

The command does not return any information if a bucket lifecycle policy is not present.

Additional Resources

- See the *S3 bucket lifecycle* section in the *Red Hat Ceph Storage Developer Guide* for details.

3.10.3. 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 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
   ```

Verification

- 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

  ```
  [root@rgw ~]# 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
   }
  }
   ]
}
```

Additional Resources

- See the *Red Hat Ceph Storage Developer Guide* for details on Amazon S3 bucket lifecycles.

3.10.4. 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. List bucket lifecycle progress:

   Example

```
[root@rgw ~]# 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.

2. 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

   ```
   [root@rgw ~]# radosgw-admin lc process --bucket=testbucket1
   ```

   b. Process all bucket lifecycle policies immediately:

   Example

   ```
   [root@rgw ~]# radosgw-admin lc process
   ```

Verification

- List the bucket lifecycle policies:

```
[root@rgw ~]# 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”
   }
]
```

Additional Resources

- See the *S3 bucket lifecycle* section in the *Red Hat Ceph Storage Developer Guide* for details.

3.10.5. 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.
- Installation of the Ceph Object Gateway.
- Root-level access to a Ceph Object Gateway node.

Procedure

1. 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

   ```
   [root@rgw ~]# ceph config set client.rgw rgw_lifecycle_work_time 06:00-08:00
   ```

Verification

- Retrieve the lifecycle expiration work time:

  Example

  ```
  [root@rgw ~]# ceph config get client.rgw rgw_lifecycle_work_time

  06:00-08:00
  ```

Additional Resources

- See the *S3 bucket lifecycle* section in the *Red Hat Ceph Storage Developer Guide* for details.

3.10.6. 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.

Additional Resources

- See the *Red Hat Ceph Storage Developer Guide* for details on bucket lifecycle.

3.10.7. 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.

Prerequisites

- Installation of the Ceph Object Gateway software.
- Root-level access to the Ceph Object Gateway node.
- An S3 user created with user access.

Procedure

1. Create a new data pool:

   Syntax

   ```
   ceph osd pool create POOL_NAME
   ```

   Example

   ```
   [root@rgw ~]# ceph osd pool create test.hot.data
   ```

2. Add a new storage class:

   Syntax

   ```
   radosgw-admin zonegroup placement add  --rgw-zonegroup default --placement-id
   PLACEMENT_TARGET --storage-class STORAGE_CLASS
   ```

   Example

   ```
   [root@rgw ~]# 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:

Syntax

```
radosgw-admin zone placement add --rgw-zone default --placement-id
PLACEMENT_TARGET --storage-class STORAGE_CLASS --data-pool DATA_POOL
```

Example

```
[root@rgw ~]# 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 **rgw** application on the data pool:

Syntax

```
ceph osd pool application enable POOL_NAME rgw
```

Example

```
[root@rgw ~] 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:

Example

```
[root@rgw ~]# aws s3api create-bucket --bucket testbucket10 --create-bucket-configuration
LocationConstraint=default:default-placement --endpoint-url http://1x.7x.2xx.1xx:80
```

7. Add the object:

Example

```
[root@rgw ~]# aws --endpoint=http://1x.7x.2xx.1xx:80 s3api put-object --bucket testbucket10
--key compliance-upload --body /root/test2.txt
```

8. Create a second data pool:

Syntax

```
ceph osd pool create POOL_NAME
```

Example

```
[root@rgw ~]# ceph osd pool create test.cold.data
```

9. Add a new storage class:

Syntax

```
radosgw-admin zonegroup placement add  --rgw-zonegroup default --placement-id
PLACEMENT_TARGET --storage-class STORAGE_CLASS
```

Example

```
[root@rgw ~]# 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:

Syntax

```
radosgw-admin zone placement add --rgw-zone default --placement-id
PLACEMENT_TARGET --storage-class STORAGE_CLASS --data-pool DATA_POOL
```

Example

```
[root@rgw ~]# 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:

    Syntax

    ```
    ceph osd pool application enable POOL_NAME rgw
    ```

    Example

    ```
    [root@rgw ~] ceph osd pool application enable test.cold.data rgw
    enabled application 'rgw' on pool 'test.cold.data'
    ```

12. Restart all the **rgw** daemons.

13. To view zone group configuration, execute:

    Syntax

    ```
    radosgw-admin zonegroup get
    {
        "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. To view the zone configuration,execute:

Syntax

```
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:

    Example

```
[root@rgw ~]# aws s3api create-bucket --bucket testbucket10 --create-bucket-configuration
LocationConstraint=default:default-placement --endpoint-url http://1x.7x.2xx.1xx:80
```

16. Create a JSON file for lifecycle configuration:

    Example

```
[root@rgw ~]# vi lifecycle.json
```

17. Add the specific lifecyle configuration rule in the file:

    Example

```
{
    "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 will transition 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.

18. Set the lifecycle configuration on the bucket:

    Example

```
[root@rgw ~] aws s3api put-bucket-lifecycle-configuration --bucket testbucket20 --lifecycle-configuration file://lifecycle.json
```

19. Retrieve the lifecycle configuration on the bucket:

Example

```
[root@rgw ~]aws s3api get-bucket-lifecycle-configuration --bucket testbucket20
{
    "Rules": [
        {
            "Expiration": {
                "Days": 365
            },
            "ID": "double transition and expiration",
            "Prefix": "",
            "Status": "Enabled",
            "Transitions": [
                {
                    "Days": 20,
                    "StorageClass": "cold.test"
                },
                {
                    "Days": 5,
                    "StorageClass": "hot.test"
                }
            ]
        }
    ]
}
```

Additional Resources

- See the *Red Hat Ceph Storage Developer Guide* for details on bucket lifecycle.

3.11. CEPH OBJECT GATEWAY DATA LAYOUT

Although RADOS only knows about pools and objects with their Extended Attributes (**xattrs**) and object map (OMAP), conceptually Ceph Object Gateway organizes its data into three different kinds:

- metadata
- bucket index
- data

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.

You can use the following commands to view metadata entries:

Syntax

```
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
```

Example

```
[root@host01 ~]# radosgw-admin metadata list
[root@host01 ~]# radosgw-admin metadata list bucket
[root@host01 ~]# radosgw-admin metadata list bucket.instance
[root@host01 ~]# radosgw-admin metadata list user
```

Every metadata entry is kept on a single RADOS object.

NOTE

A Ceph Object Gateway object might consist of several RADOS objects, the first of which 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

It 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 we keep some bucket accounting metadata in that header such as number of objects, total size, and the like.

NOTE

OMAP is a key-value store, 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.

3.11.1. Object lookup path

When accessing objects, REST APIs come to Ceph Object Gateway with three parameters:

- Account information, which has the access key in S3 or account name in Swift

- Bucket or container name
- Object name or key

At present, Ceph Object Gateway only uses account information to find out the user ID and for access control. It uses only the bucket name and object key to address the object in a pool.

Account information

The user ID in Ceph Object Gateway is a string, typically the actual user name 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 names

They are represented in the **default.rgw.meta** pool with **root** namespace. Bucket record is loaded in order to obtain a marker, which serves as a bucket ID.

Object names

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, slashes are permitted in object names, such as keys.

3.11.1.1. 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.

3.11.2. 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.

Additional Resources

- See the *Configuring bucket sharding* section in the *Red Hat Ceph Storage Object Gateway Guide* for more details.

3.12. OBJECT GATEWAY DATA LAYOUT PARAMETERS

This is 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 _both_ 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.

***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.

Additional Resources

- See the *Ceph Object Gateway data layout* section in the *Red Hat Ceph Storage Object Gateway Guide* for more details.

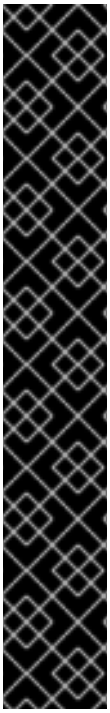
3.13. SESSION TAGS FOR ATTRIBUTE-BASED ACCESS CONTROL (ABAC) IN STS

Session tags are key-value pairs that can be passed while federating a user. They are passed as **aws:PrincipalTag** in the session or temporary credentials that are returned back by secure token service (STS). These principal tags consist of session tags that come in as part of the web token and tags that are attached to the role being assumed.

NOTE

Currently, the session tags are only supported as part of the web token passed to **AssumeRoleWithWebIdentity**.

The tags have to be always specified in the following namespace: **https://aws.amazon.com/tags**.



IMPORTANT

The trust policy must have **sts:TagSession** permission if the web token passed in by the federated user contains session tags. Otherwise, the **AssumeRoleWithWebIdentity** action fails.

Example of the trust policy with sts:TagSession:

```
{
      "Version":"2012-10-17",
      "Statement":[
      {
         "Effect":"Allow",
         "Action":["sts:AssumeRoleWithWebIdentity","sts:TagSession"],
         "Principal":{"Federated":["arn:aws:iam:::oidc-
provider/localhost:8080/auth/realms/quickstart"]},
         "Condition":{"StringEquals":{"localhost:8080/auth/realms/quickstart:sub":"test"}}
      }]
   }
```

Properties

The following are the properties of session tags:

- Session tags can be multi-valued.

NOTE

Multi-valued session tags are not supported in Amazon Web Service (AWS).

- Keycloak can be set up as an OpenID Connect Identity Provider (IDP) with a maximum of 50 session tags.
- The maximum size of a key allowed is 128 characters.
- The maximum size of a value allowed is 256 characters.
- The tag or the value cannot start with **aws:**.

Additional Resources

- See Secure Token Service for more information on secure token service.
- See Examples using session tags for Attribute-based access control in STS to see examples of the usage of session tags.
- See Sample code demonstrating usage of session tags to see a sample code demonstrating the usage of session tags.

3.13.1. Tag keys

The following are the tag keys that can be used in the role trust policy or the role permission policy.

aws:RequestTag

Description

Compares the key-value pair passed in the request with the key-value pair in the role's trust policy.
In the case of **AssumeRoleWithWebIdentity**, session tags can be used as **aws:RequestTag** in the role trust policy. Those session tags are passed by Keycloak in the web token. As a result, a federated user can assume a role.

aws:PrincipalTag

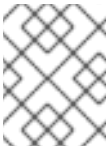
Description

Compares the key-value pair attached to the principal with the key-value pair in the policy.
In the case of **AssumeRoleWithWebIdentity**, session tags appear as principal tags in the temporary credentials once a user is authenticated. Those session tags are passed by Keycloak in the web token. They can be used as **aws:PrincipalTag** in the role permission policy.

iam:ResourceTag

Description

Compares the key-value pair attached to the resource with the key-value pair in the policy.
In the case of **AssumeRoleWithWebIdentity**, tags attached to the role are compared with those in the trust policy to allow a user to assume a role.



NOTE

The Ceph Object Gateway now supports RESTful APIs for tagging, listing tags, and untagging actions on a role.

aws:TagKeys

Description

Compares tags in the request with the tags in the policy.
In the case of **AssumeRoleWithWebIdentity**, tags are used to check the tag keys in a role trust policy or permission policy before a user is allowed to assume a role.

s3:ResourceTag

Description

Compares tags present on the S3 resource, that is bucket or object, with the tags in the role's permission policy.
It can be used for authorizing an S3 operation in the Ceph Object Gateway. However, this is not allowed in AWS.

It is a key used to refer to tags that have been attached to an object or a bucket. Tags can be attached to an object or a bucket using RESTful APIs available for the same.

3.13.2. S3 resource tags

The following list shows which S3 resource tag type is supported for authorizing a particular operation.

Tag type: Object tags

Operations

GetObject, **GetObjectTags**, **DeleteObjectTags**, **DeleteObject**, **PutACLs**, **InitMultipart**, **AbortMultipart** , `**ListMultipart**, **GetAttrs**, **PutObjectRetention**, **GetObjectRetention**, **PutObjectLegalHold**, **GetObjectLegalHold**

Tag type: Bucket tags

Operations

PutObjectTags, **GetBucketTags**, **PutBucketTags**, **DeleteBucketTags**, **GetBucketReplication**, **DeleteBucketReplication**, **GetBucketVersioning**, **SetBucketVersioning**, **GetBucketWebsite**, **SetBucketWebsite**, **DeleteBucketWebsite**, **StatBucket**, **ListBucket**, **GetBucketLogging**, **GetBucketLocation**, **DeleteBucket**, **GetLC**, **PutLC**, **DeleteLC**, **GetCORS**, **PutCORS**, **GetRequestPayment**, **SetRequestPayment**. **PutBucketPolicy**, **GetBucketPolicy**, **DeleteBucketPolicy**, **PutBucketObjectLock**, **GetBucketObjectLock**, **GetBucketPolicyStatus**, **PutBucketPublicAccessBlock**, **GetBucketPublicAccessBlock**, **DeleteBucketPublicAccessBlock**

Tag type: Bucket tags for bucket ACLs, Object tags for object ACLs

Operations

GetACLs, **PutACLs**

Tag type: Object tags of source object, Bucket tags of destination bucket

Operations

PutObject, **CopyObject**

3.14. 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 execute 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.

3.14.1. Viewing the garbage collection queue

Before you purge deleted and overwritten objects from the storage cluster, use **radosgw-admin** to view the objects awaiting garbage collection.

Prerequisites

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

Procedure

1. To view the queue of objects awaiting garbage collection:

   Example

   ```
   [root@rgw ~] radosgw-admin gc list
   ```

NOTE

To list all entries in the queue, including unexpired entries, use the **--include-all** option.

3.14.2. Adjusting garbage collection for delete-heavy workloads

Some workloads may 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 are then deleted. For these types of workloads, consider increasing the priority of garbage collection operations relative to other operations. Contact Red Hat Support with any additional questions about Ceph Object Gateway Garbage Collection.

Prerequisites

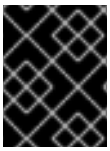
- A running Red Hat Ceph Storage cluster.
- Root-level access to all nodes in the storage cluster.

Procedure

1. Open **/etc/ceph/ceph.conf** for editing.
2. Set the value of **rgw_gc_max_concurrent_io** to 20, and the value of **rgw_gc_max_trim_chunk** to 64.

```
rgw_gc_max_concurrent_io = 20
rgw_gc_max_trim_chunk = 64
```

3. Restart the Ceph Object Gateway to allow the changed settings to take effect.
4. 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.

Additional Resources

- Ceph RGW - GC Tuning Options
- RGW General Settings
- Configuration Reference

3.14.3. Viewing the number of objects garbage collected

You can view the number of objects garbage collected since the restart of the Ceph Object Gateway using **gc_retire_object** parameter in the performance dump.

This counter can be monitored to determine the delta between the number of objects at a specific time frame for that Ceph Object Gateway, for example, the average number of objects garbage collected weekly, daily, or hourly. This can be used to determine how efficiently garbage collection progresses per Ceph Object Gateway.

The socket file is located in the **/var/run/ceph** directory.

Prerequisites

- A running Red Hat Ceph Storage cluster with Ceph Object Gateway installed.
- Root-level access to all nodes in the storage cluster.
- The **ceph-common** packages installed on the Ceph Monitor.

Procedure

- Access the performance counter data using **grep** for the **gc_retire_object** counter:

Syntax

```
ceph --admin-daemon PATH_TO_SOCKET_FILE perf dump | grep gc_retire_object
```

Example

```
[root@mon ~]# ceph --admin-daemon /var/run/ceph/ceph-client.rgw.f27-h25-000-6048r.rgw0.104.93991732704816.asok perf dump | grep gc_retire_object
```

Additional Resources

- See the Red Hat Knowledgebase article *Ceph RGW - GC Tuning Options* for more information.
- See the *General Settings* section in the *Red Hat Ceph Storage Object Gateway Configuration and Administration Guide* for more information.
- See the *Configuration Reference* section in the *Red Hat Ceph Storage Object Gateway Configuration and Administration Guide* for more information.
- See the *Ceph Object Gateway metrics* section in the *Red Hat Ceph Storage Administration Guide* for more information.

3.15. OPTIMIZE THE CEPH OBJECT GATEWAY'S DATA OBJECT STORAGE

Bucket life cycle 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 life cycle configuration actions:

- Expiration
- NoncurrentVersionExpiration
- AbortIncompleteMultipartUpload

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to all of the nodes in the storage cluster.

3.15.1. 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.

3.15.2. Optimizing the bucket life cycle

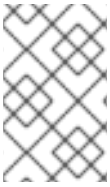
Two options in the Ceph configuration file affect the efficiency of bucket life cycle processing:

- **rgw_lc_max_worker** specifies the number of life cycle 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 life cycle 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 a large number of 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, please validate current storage cluster performance and Ceph Object Gateway utilization. Red Hat does not recommend that you assign a value of 10 or above for either of these options.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to all of the nodes in the storage cluster.

Procedure

1. Open **/etc/ceph/ceph.conf** for editing.
2. 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:

   Syntax

   ```
   rgw_lc_max_worker = VALUE
   ```

   Example

   ```
   rgw_lc_max_worker = 7
   ```

3. 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:

   Syntax

   ```
   rgw_lc_max_wp_worker = VALUE
   ```

   Example

   ```
   rgw_lc_max_wp_worker = 7
   ```

4. Restart the Ceph Object Gateway to allow the changed settings to take effect.

5. Monitor the storage cluster to verify that the increased values do not adversely affect performance.

Additional Resources

- For more information about the S3 API and bucket life cycle operations, refer to S3 API bucket life cycle.
- For more information about the bucket life cycle and parallel thread processing, see Bucket life cycle parallel processing
- For more information about Ceph Object Gateway life cycles, contact Red Hat Support.

3.15.3. Additional Resources

- For more information about bucket life cycle configuration actions, refer to S3 API bucket life cycle.

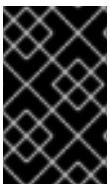
3.16. THE CEPH OBJECT GATEWAY AND MULTI-FACTOR AUTHENTICATION

As a storage administrator, you can manage time-based one time password (TOTP) tokens for Ceph Object Gateway users.

3.16.1. Multi-factor authentication

When a bucket is configured for object versioning, you 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 multisite 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.

Table 3.1. Terminology

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.

Term	Description
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.

- For more information, see *The Ceph Object Gateway and multi-factor authentication* .

3.16.2. 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@host ~]$ SEED=$(head -10 /dev/urandom | sha512sum | cut -b 1-30)
   ```

2. Print the seed by running echo on the **SEED** variable:

   Example

   ```
   [user@host ~]$ echo $SEED
   492dedb20cf51d1405ef6a1316017e
   ```

   Configure the one-time password generator and the back-end MFA system to use the same seed.

Additional Resources

- For more information, see the solution *Unable to create RGW MFA token for bucket* .
- For more information, see *The Ceph Object Gateway and multi-factor authentication* .

3.16.3. 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

1. 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@mon ~]# radosgw-admin mfa create --uid=johndoe --totp-serial=MFAtest --totp-seed=492dedb20cf51d1405ef6a1316017e
   ```

Additional Resources

- For more information, see *Creating a seed for multi-factor authentication* .
- For more information, See *Resynchronizing a multi-factor authenitcation token* .

3.16.4. 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.
- 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. 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@mon ~] # 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. See *Resynchronizing a multi-factor authentication token* in the *Red Hat Ceph Storage Object Gateway Configuration and Administration Guide*.

Additional Resources

- For more information, see *Creating a seed for multi-factor authentication* .
- For more information, see *Resynchronizing a multi-factor authenitcation token* .

3.16.5. 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

```
radosgw-admin mfa resync --uid=johndoe --totp-serial=MFAtest --totp-pin=802017 --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@mon ~]# radosgw-admin mfa check  --uid=johndoe --totp-serial=MFAtest --totp-pin=870305
ok
```

Additional Resources

- For more information, see *Creating a new multi-factor authentication TOTP token* .

3.16.6. 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

1. List MFA TOTP tokens:

Syntax

```
radosgw-admin mfa list --uid=USERID
```

Set *USERID* to the user name MFA is set up on.

Example

```
[root@mon ~]# radosgw-admin mfa list --uid=johndoe
{
    "entries": [
        {
```

```
            "type": 2,
            "id": "MFAtest",
            "seed": "492dedb20cf51d1405ef6a1316017e",
            "seed_type": "hex",
            "time_ofs": 0,
            "step_size": 30,
            "window": 2
        }
    ]
}
```

Additional Resources

- For more information, see *Creating a new multi-factor authentication TOTP token* .

3.16.7. 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

1. 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.

Additional Resources

- For more information, see *Creating a new multi-factor authentication TOTP token* .

3.16.8. Deleting a multi-factor authentication TOTP token

Delete a 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.
- An MFA TOTP token was created using **radosgw-admin mfa create**.

Procedure

1. Delete an MFA TOTP token:

   Syntax

   ```
   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.

   Example

   ```
   [root@mon ~]# radosgw-admin mfa remove --uid=johndoe --totp-serial=MFAtest
   ```

2. Verify the MFA TOTP token was deleted:

   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.

   Example

   ```
   [root@mon ~]# radosgw-admin mfa get --uid=johndoe --totp-serial=MFAtest
   MFA serial id not found
   ```

Additional Resources

- For more information, see *The Ceph Object Gateway and multi-factor authentication* .

3.17. REMOVING CEPH OBJECT GATEWAY USING ANSIBLE

To remove a Ceph Object Gateway with Ansible, use the **shrink-rgw.yml** playbook.

Prerequisites

- An Ansible administration node.
- A running Red Hat Ceph Storage cluster deployed by Ansible.

Procedure

1. Change to the **/usr/share/ceph-ansible/** directory.

   ```
   [user@admin ~]$ cd /usr/share/ceph-ansible
   ```

2. For **bare-metal** and **containers** deployments, run the **shrink-rgw.yml** Ansible playbook:

   Syntax

   ```
   ansible-playbook infrastructure-playbooks/shrink-rgw.yml -e
   rgw_to_kill=HOSTNAME.rgw_INSTANCE_NAME_ -u ANSIBLE_USER_NAME -i hosts
   ```

   Replace:

   - ***HOSTNAME*** with the short host name of the Ceph Object Gateway node. You can remove only one Ceph Object Gateway each time the playbook runs.
   - ***ANSIBLE_USER_NAME*** with the name of the Ansible user

   Example

   ```
   [user@admin ceph-ansible]$ ansible-playbook infrastructure-playbooks/shrink-rgw.yml -e
   rgw_to_kill=node03.rgw0 -u admin -i hosts
   ```

3. Remove the Ceph Object Gateway entry from all Ceph configuration files in the storage cluster.

4. Ensure that the Ceph Object Gateway has been successfully removed:

   ```
   [root@mon ~]# ceph -s
   ```

Additional Resources

- For more information on installing Red Hat Ceph Storage, see the *Red Hat Ceph Storage Installation Guide*.
- See the *Configuring Ansible's inventory location* section in the *{storage_product} Installation Guide* for more details on the Ansible inventory configuration.

CHAPTER 4. CONFIGURATION REFERENCE

The following settings may be added to the Ceph configuration file, that is, usually **ceph.conf**, under the **[client.rgw.<instance_name>]** section. The settings may contain default values. If you do not specify each setting in the Ceph configuration file, the default value will be set automatically.

Configuration variables set under the **[client.rgw.<instance_name>]** section will not apply to **rgw** or **radosgw-admin** commands without an **instance_name** specified in the command. Therefore, variables meant to be applied to all Ceph Object Gateway instances or all **radosgw-admin** commands can be put into the **[global]** or the **[client]** section to avoid specifying **instance_name**.

4.1. GENERAL 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, pubsub**
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

Name	Description	Type	Default
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**
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 time **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**

Name	Description	Type	Default
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 DNS CNAME record of the request hostname field (if hostname is not equal to **rgw_dns name**).	Boolean	**false**
rgw_object_stripe_size	The size of an object stripe for Ceph Object Gateway objects.	Integer	**4 << 20**
rgw_extended_http_attrs	Add 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**

Name	Description	Type	Default
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 to set 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**
rgw_content_length_compat	Enable compatability 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

Name	Description	Type	Default
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**
sync_from_all	A **radosgw-admin** command to set or unset whether zone syncs from all zonegroup peers.	Boolean	**false**

4.2. ABOUT POOLS

Ceph zones map to a series of Ceph Storage Cluster pools.

Manually Created Pools vs. Generated 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.

To set up production systems, see the *Ceph Object Gateway for Production* guide for Red Hat Ceph Storage 4. For storage strategies, see the *Developing Storage Strategies* section in the *Ceph Object Gateway for Production* guide.

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 system control, logging, user information, etc. 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 index of the buckes.
- **.<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>.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 to 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**

4.3. 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.

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**
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**

4.4. 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**

Name	Description	Type	Default
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 (e.g., **http://fqdn.com/swift**).	**swift**	N/A
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**

4.5. LOGGING SETTINGS

Name	Description	Type	Default
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.	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**

4.6. 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 requires to serve requests.	String	**Member, admin**

Name	Description	Type	Default
rgw_keystone_token_cache_size	The maximum number of entries in each Keystone token cache.	Integer	**10000**
rgw_keystone_revocation_interval	The number of seconds between token revocation checks.	Integer	**15 * 60**

4.7. 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**

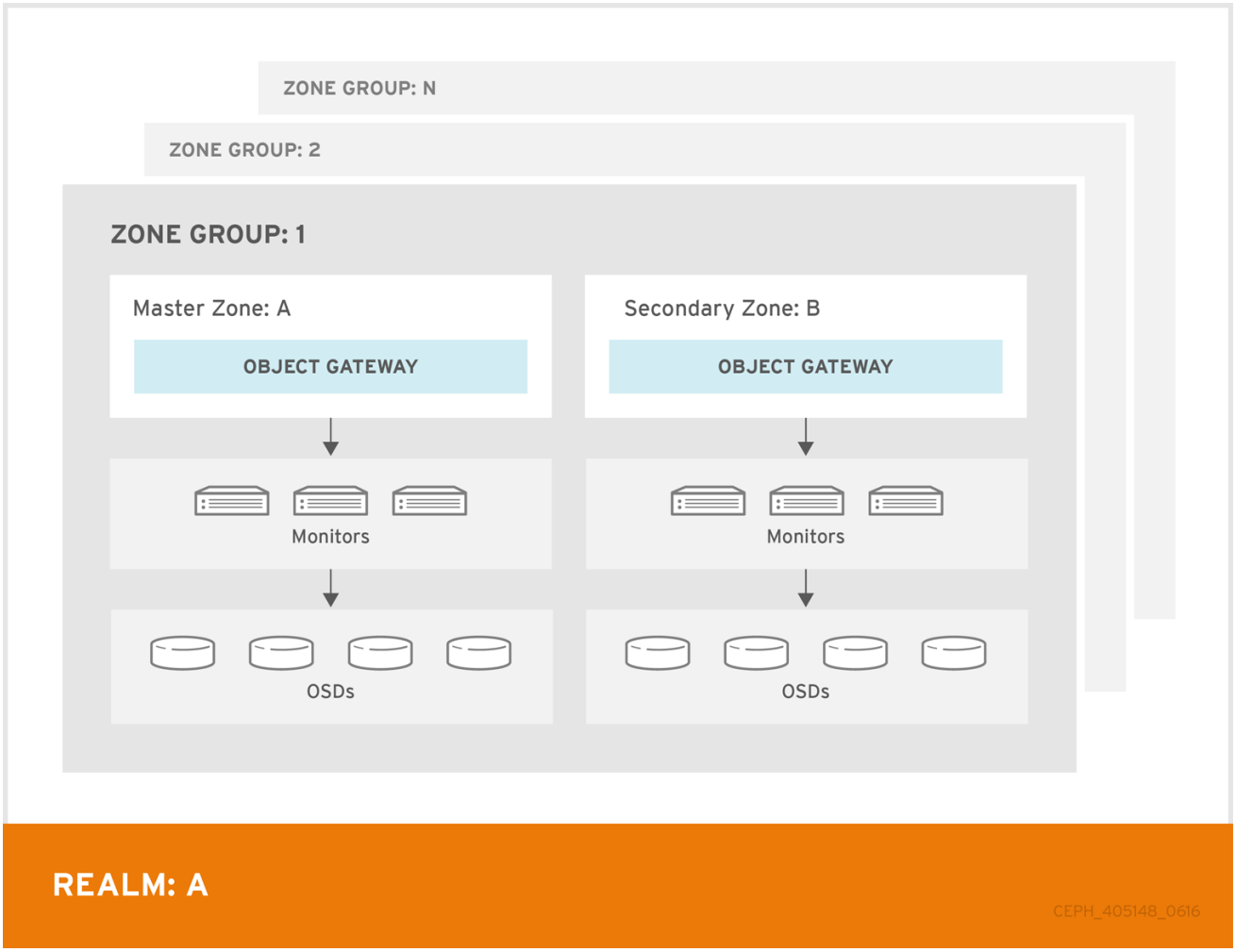
CHAPTER 5. MULTISITE

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. To configure multiple zones without replication, see Section 5.12, “Configuring Multiple Zones without Replication” .

- **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.

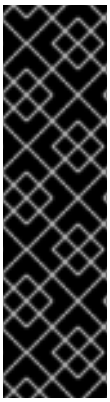


5.1. REQUIREMENTS AND ASSUMPTIONS

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.

This guide assumes at least two Ceph storage clusters in geographically separate locations; however, the configuration can work on the same physical site. This guide 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 Additionally, **each zone group requires a master zone.** Zone groups may have one or more secondary or non-master zones.



IMPORTANT

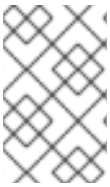
The master zone within the master zone group of a realm is responsible for storing the master copy of the realm's metadata, including users, quotas and buckets (created by the **radosgw-admin** CLI). This metadata gets synchronized to secondary zones and secondary zone groups automatically. Metadata operations executed with the **radosgw-admin** CLI **MUST be executed** 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 execute metadata operations on secondary zones and zone groups, but it is **NOT recommended** because they **WILL NOT** be syncronized, leading to fragmented metadata.

In the following examples, the **rgw1** host will serve as the master zone of the master zone group; the **rgw2** host will serve as the secondary zone of the master zone group; the **rgw3** host will serve as the master zone of the secondary zone group; and the **rgw4** host will serve as the secondary zone of the secondary zone group.

5.2. POOLS

Red Hat recommends using the Ceph Placement Group's per Pool Calculator to calculate a suitable number of placement groups for the pools the **ceph-radosgw** daemon will create. Set the calculated values as defaults in your Ceph configuration file. For example:

```
osd pool default pg num = 50
osd pool default pgp num = 50
```



NOTE

Make this change to the Ceph configuration file on your storage cluster; then, either make a runtime change to the configuration so that it will use those defaults when the gateway instance creates the pools.

Alternatively, create the pools manually. See *Pools* chapter in the *Storage Strategies* guide for details on creating pools.

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**

5.3. INSTALLING AN OBJECT GATEWAY

To install the Ceph Object Gateway, see the *Red Hat Ceph Storage Installation Guide* for details.

All Ceph Object Gateway nodes must follow the tasks listed in the *Requirements for Installing Red Hat Ceph Storage* section.

Ansible can install and configure Ceph Object Gateways for use with a Ceph Storage cluster. For multi-site and multi-site group deployments, you should have an Ansible configuration for each zone.

If you install Ceph Object Gateway with Ansible, the Ansible playbooks will handle the initial configuration for you. To install the Ceph Object Gateway with Ansible, add your hosts to the **/etc/ansible/hosts** file. Add the Ceph Object Gateway hosts under an **[rgws]** section to identify their roles to Ansible. If your hosts have sequential naming, you may use a range. For example:

```
[rgws]
<rgw-host-name-1>
<rgw-host-name-2>
<rgw-host-name[3..10]>
```

Once you have added the hosts, you may rerun your Ansible playbooks.

NOTE

Ansible will ensure your gateway is running, so the default zones and pools may need to be deleted manually. This guide provides those steps.

When updating an existing multi-site cluster with an asynchronous update, follow the installation instruction for the update. Then, restart the gateway instances.

NOTE

There is no required order for restarting the instances. Red Hat recommends restarting the master zone group and master zone first, followed by the secondary zone groups and secondary zones.

5.4. ESTABLISH A MULTISITE REALM

All gateways in a cluster have a configuration. In a multi-site realm, these gateways may reside in different zone groups and zones. Yet, they must work together within the realm. In a multi-site realm, all gateway instances **MUST** retrieve their configuration from a **ceph-radosgw** daemon on a host within the master zone group and master zone.

Consequently, the first step in creating a multi-site cluster involves establishing the realm, master zone group and master zone. To configure your gateways in a multi-site configuration, choose a **ceph-radosgw** instance that will hold the realm configuration, master zone group and master zone.

5.4.1. Create a Realm

A realm contains the multi-site configuration of zone groups and zones and also serves to enforce a globally unique namespace within the realm.

Create a new realm for the multi-site configuration by opening a command line interface on a host identified to serve in the master zone group and zone. Then, execute the following:

```
[root@master-zone]# radosgw-admin realm create --rgw-realm={realm-name} [--default]
```

For example:

```
[root@master-zone]# radosgw-admin realm create --rgw-realm=movies --default
```

If the cluster will have a single realm, specify the **--default** flag. If **--default** is specified, **radosgw-admin** will use this realm by default. If **--default** is not specified, adding zone-groups and zones requires specifying either the **--rgw-realm** flag or the **--realm-id** flag to identify the realm when adding zone groups and zones.

After creating the realm, **radosgw-admin** will echo back the realm configuration. For example:

```
{
    "id": "0956b174-fe14-4f97-8b50-bb7ec5e1cf62",
    "name": "movies",
    "current_period": "1950b710-3e63-4c41-a19e-46a715000980",
    "epoch": 1
}
```

NOTE

Ceph generates a unique ID for the realm, which allows the renaming of a realm if the need arises.

5.4.2. Create a Master Zone Group

A realm must have at least one zone group, which will serve as the master zone group for the realm.

Create a new master zone group for the multi-site configuration by opening a command line interface on a host identified to serve in the master zone group and zone. Then, execute the following:

```
[root@master-zone]# radosgw-admin zonegroup create --rgw-zonegroup={name} --endpoints={url} [--rgw-realm={realm-name}|--realm-id={realm-id}] --master --default
```

For example:

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

If the realm will only have a single zone group, specify the **--default** flag. If **--default** is specified, **radosgw-admin** will use this zone group by default when adding new zones. If **--default** is not specified, adding zones will require either the **--rgw-zonegroup** flag or the **--zonegroup-id** flag to identify the zone group when adding or modifying zones.

After creating the master zone group, **radosgw-admin** will echo back the zone group configuration. For example:

```
{
    "id": "f1a233f5-c354-4107-b36c-df66126475a6",
    "name": "us",
    "api_name": "us",
    "is_master": "true",
    "endpoints": [
        "http:\/\/rgw1:80"
    ],
    "hostnames": [],
    "hostnames_s3webzone": [],
    "master_zone": "",
    "zones": [],
    "placement_targets": [],
    "default_placement": "",
    "realm_id": "0956b174-fe14-4f97-8b50-bb7ec5e1cf62"
}
```

5.4.3. Create a Master Zone

IMPORTANT

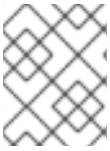
Zones must be created on a Ceph Object Gateway node that will be within the zone.

Create a master zone for the multi-site configuration by opening a command line interface on a host identified to serve in the master zone group and zone. Then, execute the following:

```
[root@master-zone]# radosgw-admin zone create
                        --rgw-zonegroup={zone-group-name} \
                        --rgw-zone={zone-name} \
                        --master --default \
                        --endpoints={http://fqdn:port}[,{http://fqdn:port}]
```

For example:

```
[root@master-zone]# radosgw-admin zone create --rgw-zonegroup=us \
                        --rgw-zone=us-east \
                        --master --default \
                        --endpoints={http://fqdn:port}[,{http://fqdn:port}]
```



NOTE

The **--access-key** and **--secret** aren't specified. These settings will be added to the zone once the user is created in the next section.

5.4.4. Delete the Default Zone Group and Zone

Delete the **default** zone if it exists. Make sure to remove it from the default zone group first.

IMPORTANT

The following steps assume a multi-site configuration using newly installed systems that aren't storing data yet. DO NOT DELETE the **default** zonegroup, zone, and its pools if you are already using it to store data, or the data will be deleted and unrecoverable.

In order to access old data in the **default** zone and zonegroup, use **--rgw-zone default** and **--rgw-zonegroup default** in **radosgw-admin** commands.

1. Remove the zonegroup and the zone:

   Example

   ```
   [root@master-zone]# radosgw-admin zonegroup remove --rgw-zonegroup=default --rgw-zone=default
   [root@master-zone]# radosgw-admin zone delete --rgw-zone=default
   [root@master-zone]# radosgw-admin zonegroup delete --rgw-zonegroup=default
   ```

2. Update and commit the period if the cluster is in a multi-site configuration:

   Example

   ```
   [root@master-zone]# radosgw-admin period update --commit
   ```

3. Delete the **default** pools in your Ceph storage cluster if they exist.

   Example

   ```
   [root@master-zone]# ceph osd pool delete default.rgw.control default.rgw.control --yes-i-really-really-mean-it
   [root@master-zone]# ceph osd pool delete default.rgw.data.root default.rgw.data.root --yes-i-really-really-mean-it
   [root@master-zone]# ceph osd pool delete default.rgw.log default.rgw.log --yes-i-really-really-mean-it
   [root@master-zone]# ceph osd pool delete default.rgw.users.uid default.rgw.users.uid --yes-i-really-really-mean-it
   ```

IMPORTANT

After deleting the pools, restart the Ceph Object Gateway process.

5.4.5. Create a System User

The **ceph-radosgw** daemons must authenticate before pulling realm and period information. In the master zone, create a system user to facilitate authentication between daemons.

```
[root@master-zone]# radosgw-admin user create --uid="{user-name}" --display-name="{Display Name}" --system
```

For example:

```
[root@master-zone]# radosgw-admin user create --uid="synchronization-user" --display-name="Synchronization User" --system
```

Make a note of the **access_key** and **secret_key**, as the secondary zones will require them to authenticate with the master zone.

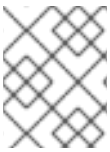
Finally, add the system user to the master zone.

```
[root@master-zone]# radosgw-admin zone modify --rgw-zone=us-east --access-key={access-key} --secret={secret}
[root@master-zone]# radosgw-admin period update --commit
```

5.4.6. Update the Period

After updating the master zone configuration, update the period.

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



NOTE

Updating the period changes the epoch, and ensures that other zones will receive the updated configuration.

5.4.7. Update the Ceph Configuration File

Update the Ceph configuration file on master zone hosts by adding the **rgw_zone** configuration option and the name of the master zone to the instance entry.

```
[client.rgw.{instance-name}]
...
rgw_zone={zone-name}
```

For example:

```
[client.rgw.rgw1.rgw0]
host = rgw1
rgw frontends = "civetweb port=80"
rgw_zone=us-east
```

5.4.8. Start the Gateway

On the object gateway host, start and enable the Ceph Object Gateway service:

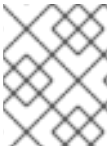
```
# systemctl start ceph-radosgw@rgw.`hostname -s`.rgw0
# systemctl enable ceph-radosgw@rgw.`hostname -s`.rgw0
```

If the service is already running, restart the service instead of starting and enabling it:

```
# systemctl restart ceph-radosgw@rgw.`hostname -s`.rgw0
```

5.5. ESTABLISH 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, execute **ALL** of the **radosgw-admin zone** operations on a host identified to serve the secondary zone.



NOTE

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

IMPORTANT

You must execute 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 execute it on a host within the master zone of the master zone group, or the buckets will not synchronize to other zone groups and zones.

5.5.1. Pull the Realm

Using the URL path, access key and secret of the master zone in the master zone group, pull the realm to the host. To pull a non-default realm, specify the realm using the **--rgw-realm** or **--realm-id** configuration options.

```
# radosgw-admin realm pull --url={url-to-master-zone-gateway} --access-key={access-key} --secret=
{secret}
```

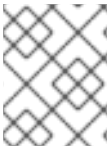
If this realm is the default realm or the only realm, make the realm the default realm.

```
# radosgw-admin realm default --rgw-realm={realm-name}
```

5.5.2. Pull the Period

Using the URL path, access key and secret of the master zone in the master zone group, pull the period to the host. To pull a period from a non-default realm, specify the realm using the **--rgw-realm** or **--realm-id** configuration options.

```
# radosgw-admin period pull --url={url-to-master-zone-gateway} --access-key={access-key} --secret=
{secret}
```



NOTE

Pulling the period retrieves the latest version of the zone group and zone configurations for the realm.

5.5.3. Create a Secondary Zone

IMPORTANT

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

Create a secondary zone for the multi-site configuration by opening a command line interface on a host identified to serve the secondary zone. Specify the zone group ID, the new zone name and an endpoint for the zone. **DO NOT** use the **--master** or **--default** flags. All zones run in an active-active configuration by default; that is, a gateway client may 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. Execute the following:

Syntax

```
[root@second-zone]# radosgw-admin zone create \
                    --rgw-zonegroup={zone-group-name}\
                    --rgw-zone={zone-name} \
                    --access-key={system-key} --secret={secret}\
                    --endpoints=http://{fqdn}:80 \
                    [--read-only]
```

Example

```
[root@second-zone]# radosgw-admin zone create
                     --rgw-zonegroup=us \
                     --rgw-zone=us-west \
                     --access-key={system-key} --secret={secret} \
                     --endpoints=http://rgw2:80
```

IMPORTANT

The following steps assume a multi-site configuration using newly installed systems that aren't storing data. **DO NOT DELETE** the **default** zone and its pools if you are already using them to store data, or the data will be lost and unrecoverable.

Delete the default zone if needed.

```
[root@second-zone]# radosgw-admin zone delete --rgw-zone=default
```

Finally, delete the default pools in your Ceph storage cluster if needed.

```
# ceph osd pool delete default.rgw.control default.rgw.control --yes-i-really-really-mean-it
# ceph osd pool delete default.rgw.data.root default.rgw.data.root --yes-i-really-really-mean-it
# ceph osd pool delete default.rgw.log default.rgw.log --yes-i-really-really-mean-it
```

```
# ceph osd pool delete default.rgw.users.uid default.rgw.users.uid --yes-i-really-really-mean-it
```

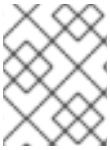
IMPORTANT

After deleting the pools, restart the RGW process.

5.5.4. Update the Period

After updating the master zone configuration, update the period.

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



NOTE

Updating the period changes the epoch, and ensures that other zones will receive the updated configuration.

5.5.5. Update the Ceph Configuration File

Update the Ceph configuration file on the secondary zone hosts by adding the **rgw_zone** configuration option and the name of the secondary zone to the instance entry.

```
[client.rgw.{instance-name}]
...
rgw_zone={zone-name}
```

For example:

```
[client.rgw.rgw2.rgw0]
host = rgw2
rgw frontends = "civetweb port=80"
rgw_zone=us-west
```

5.5.6. Start the Gateway

On the object gateway host, start and enable the Ceph Object Gateway service:

```
# systemctl start ceph-radosgw@rgw.`hostname -s`.rgw0
# systemctl enable ceph-radosgw@rgw.`hostname -s`.rgw0
```

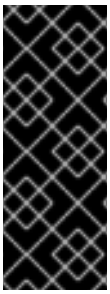
If the service is already running, restart the service instead of starting and enabling it:

```
# systemctl restart ceph-radosgw@rgw.`hostname -s`.rgw0
```

5.6. CONFIGURING THE ARCHIVE SYNC MODULE (TECHNOLOGY PREVIEW)

The archive sync module leverages the versioning feature of S3 objects in Ceph object gateway to have an archive zone. The archive zone has a history of versions of S3 objects that can only be eliminated through the gateways associated with the archive zone. It captures all the data updates and metadata to

consolidate them as versions of S3 objects.



IMPORTANT

The archive sync module is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process. See the support scope for Red Hat Technology Preview features for more details.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- **root** or **sudo** access.
- Installation of the Ceph Object Gateway.

Procedure

1. Configure the archive sync module when creating a new zone by using the **archive** tier:

Syntax

```
radosgw-admin zone create --rgw-zonegroup={ZONE_GROUP_NAME} --rgw-zone={ZONE_NAME}
--endpoints={http://fqdn:port}[,{http://fqdn:port] --tier-type=archive
```

Example

```
[root@master-zone]# radosgw-admin zone create --rgw-zonegroup=us --rgw-zone=us-east --
endpoints={http://fqdn:port}[,{http://fqdn:port}] --tier-type=archive
```

Additional resources

- See the *Establish a Multisite Realm* section in the *Red Hat Ceph Storage Object Gateway Guide* for more details.

5.7. FAILOVER AND DISASTER RECOVERY

If the master zone would fail, failover to the secondary zone for disaster recovery.

1. Make the secondary zone the master and default zone. For example:

   ```
   # 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:

   ```
   # 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.

```
# systemctl restart ceph-radosgw@rgw.`hostname -s`.rgw0
```

If the former master zone recovers, revert the operation.

1. From the recovered zone, pull the realm from the current master zone.

```
# radosgw-admin realm pull --url={url-to-master-zone-gateway} \
                         --access-key={access-key} --secret={secret}
```

2. Make the recovered zone the master and default zone.

```
# radosgw-admin zone modify --rgw-zone={zone-name} --master --default
```

3. Update the period to make the changes take effect.

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

4. Restart the Ceph Object Gateway in the recovered zone.

```
# systemctl restart ceph-radosgw@rgw.`hostname -s`.rgw0
```

5. 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
```

6. Update the period to make the changes take effect.

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

7. Restart the Ceph Object Gateway in the secondary zone.

```
# systemctl restart ceph-radosgw@rgw.`hostname -s`.rgw0
```

5.8. MIGRATING A SINGLE SITE SYSTEM TO MULTI-SITE

To migrate from a single site system with a **default** zone group and zone to a multi site system, use the following steps:

1. Create a realm. Replace **<name>** with the realm name.

```
[root@master-zone]# radosgw-admin realm create --rgw-realm=<name> --default
```

2. Rename the default zone and zonegroup. Replace **<name>** with the zonegroup or zone name.

```
[root@master-zone]# radosgw-admin zonegroup rename --rgw-zonegroup default --
zonegroup-new-name=<name>
[root@master-zone]# radosgw-admin zone rename --rgw-zone default --zone-new-name us-
```

```
east-1 --rgw-zonegroup=<name>
```

3. Configure the master zonegroup. Replace **<name>** with the realm or zonegroup name. Replace **<fqdn>** with the fully qualified domain name(s) in the zonegroup.

```
[root@master-zone]# radosgw-admin zonegroup modify --rgw-realm=<name> --rgw-
zonegroup=<name> --endpoints http://<fqdn>:80 --master --default
```

4. Configure the master zone. Replace **<name>** with the realm, zonegroup or zone name. Replace **<fqdn>** with the fully qualified domain name(s) in the zonegroup.

```
[root@master-zone]# radosgw-admin zone modify --rgw-realm=<name> --rgw-zonegroup=
<name> \
                    --rgw-zone=<name> --endpoints http://<fqdn>:80 \
                    --access-key=<access-key> --secret=<secret-key> \
                    --master --default
```

5. Create a system user. Replace **<user-id>** with the username. Replace **<display-name>** with a display name. It may contain spaces.

```
[root@master-zone]# radosgw-admin user create --uid=<user-id> \
                    --display-name="<display-name>" \
                    --access-key=<access-key> --secret=<secret-key> \ --system
```

6. Commit the updated configuration.

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

7. Restart the Ceph Object Gateway.

```
# systemctl restart ceph-radosgw@rgw.`hostname -s`.rgw0
```

After completing this procedure, proceed to Establish a Secondary Zone to create a secondary zone in the master zone group.

5.9. MULTISITE COMMAND LINE USAGE

5.9.1. 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.

By default, the Ceph Object Gateway version 2 does not create a realm for backward compatibility with version 1.3 and earlier releases. However, as a best practice, Red Hat recommends creating realms for new clusters.

5.9.1.1. Creating a Realm

To create a realm, execute **realm create** and specify the realm name. If the realm is the default, specify **--default**.

```
[root@master-zone]# radosgw-admin realm create --rgw-realm={realm-name} [--default]
```

For example:

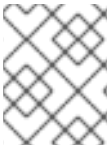
```
[root@master-zone]# radosgw-admin realm create --rgw-realm=movies --default
```

By specifying **--default**, the realm will be called implicitly with each **radosgw-admin** call unless **--rgw-realm** and the realm name are explicitly provided.

5.9.1.2. Making a Realm the Default

One realm in the list of realms should be the default realm. There may be only one 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. Alternatively, to change which realm is the default, execute:

```
[root@master-zone]# radosgw-admin realm default --rgw-realm=movies
```



NOTE

When the realm is default, the command line assumes **--rgw-realm=<realm-name>** as an argument.

5.9.1.3. Deleting a Realm

To delete a realm, execute **realm delete** and specify the realm name.

```
[root@master-zone]# radosgw-admin realm delete --rgw-realm={realm-name}
```

For example:

```
[root@master-zone]# radosgw-admin realm delete --rgw-realm=movies
```

5.9.1.4. Getting a Realm

To get a realm, execute **realm get** and specify the realm name.

```
# radosgw-admin realm get --rgw-realm=<name>
```

For example:

```
# radosgw-admin realm get --rgw-realm=movies [> filename.json]
```

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

```
{
    "id": "0a68d52e-a19c-4e8e-b012-a8f831cb3ebc",
    "name": "movies",
```

```
    "current_period": "b0c5bbef-4337-4edd-8184-5aeab2ec413b",
    "epoch": 1
}
```

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

5.9.1.5. Setting a Realm

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

```
[root@master-zone]# radosgw-admin realm set --rgw-realm=<name> --infile=<infilename>
```

For example:

```
[root@master-zone]# radosgw-admin realm set --rgw-realm=movies --infile=filename.json
```

5.9.1.6. Listing Realms

To list realms, execute **realm list**.

```
# radosgw-admin realm list
```

5.9.1.7. Listing Realm Periods

To list realm periods, execute **realm list-periods**.

```
# radosgw-admin realm list-periods
```

5.9.1.8. Pulling a Realm

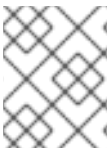
To pull a realm from the node containing the master zone group and master zone to a node containing a secondary zone group or zone, execute **realm pull** on the node that will receive the realm configuration.

```
# radosgw-admin realm pull --url={url-to-master-zone-gateway} --access-key={access-key} --secret=
{secret}
```

5.9.1.9. Renaming a Realm

A realm is not part of the period. Consequently, renaming the realm is only applied locally, and will not get pulled with **realm pull**. When renaming a realm with multiple zones, **run the command on each zone.** To rename a realm, execute the following:

```
# radosgw-admin realm rename --rgw-realm=<current-name> --realm-new-name=<new-realm-
name>
```



NOTE

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

5.9.2. Zone Groups

The Ceph Object Gateway supports multi-site deployments and a global namespace by using the notion of zone groups. Formerly called a region, a zone group defines the geographic location of one or more Ceph Object Gateway instances within one or more zones.

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.

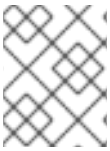
NOTE

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.

5.9.2.1. Creating a Zone Group

Creating a zone group consists of specifying the zone group name. Creating a zone assumes it will live in the default realm unless **--rgw-realm=<realm-name>** is specified. If the zonegroup is the default zonegroup, specify the **--default** flag. If the zonegroup is the master zonegroup, specify the **--master** flag. For example:

```
# radosgw-admin zonegroup create --rgw-zonegroup=<name> [--rgw-realm=<name>][--master] [--default]
```



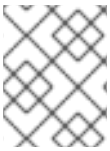
NOTE

Use **zonegroup modify --rgw-zonegroup=<zonegroup-name>** to modify an existing zone group's settings.

5.9.2.2. Making a Zone Group the Default

One zonegroup in the list of zonegroups should be the default zonegroup. There may be only one default zonegroup. If there is only one zonegroup and it wasn't specified as the default zonegroup when it was created, make it the default zonegroup. Alternatively, to change which zonegroup is the default, execute:

```
# radosgw-admin zonegroup default --rgw-zonegroup=comedy
```



NOTE

When the zonegroup is default, the command line assumes **--rgw-zonegroup=<zonegroup-name>** as an argument.

Then, update the period:

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

5.9.2.3. Adding a Zone to a Zone Group

To add a zone to a zonegroup, you **MUST** execute this step on a host that will be in the zone. To add a zone to a zonegroup, execute the following:

```
# radosgw-admin zonegroup add --rgw-zonegroup=<name> --rgw-zone=<name>
```

Then, update the period:

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

5.9.2.4. Removing a Zone from a Zone Group

To remove a zone from a zonegroup, execute the following:

```
# radosgw-admin zonegroup remove --rgw-zonegroup=<name> --rgw-zone=<name>
```

Then, update the period:

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

5.9.2.5. Renaming a Zone Group

To rename a zonegroup, execute the following:

```
# radosgw-admin zonegroup rename --rgw-zonegroup=<name> --zonegroup-new-name=<name>
```

Then, update the period:

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

5.9.2.6. Deleting a Zone Group

To delete a zonegroup, execute the following:

```
# radosgw-admin zonegroup delete --rgw-zonegroup=<name>
```

Then, update the period:

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

5.9.2.7. Listing Zone Groups

A Ceph cluster contains a list of zone groups. To list the zone groups, execute:

```
# radosgw-admin zonegroup list
```

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

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

```
        "us"
    ]
}
```

5.9.2.8. Getting a Zone Group

To view the configuration of a zone group, execute:

```
# radosgw-admin zonegroup get [--rgw-zonegroup=<zonegroup>]
```

The zone group configuration looks like this:

```
{
    "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"
}
```

5.9.2.9. Setting a Zone Group

Defining a zone group consists of creating a JSON object, specifying at least 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 (e.g., **zonegroup.json**); then, execute the following command:

```
# 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:

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

-

5.9.2.10. 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, execute the following:

```
# 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.

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

5.9.3. 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 executed on a host that operates or will operate within the zone.

5.9.3.1. Creating a Zone

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

IMPORTANT

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

```
[root@zone] radosgw-admin zone create --rgw-zone=<name> \
            [--zonegroup=<zonegroup-name]\
```

```
[--endpoints=<endpoint:port>[,<endpoint:port>] \
[--master] [--default] \
--access-key $SYSTEM_ACCESS_KEY --secret $SYSTEM_SECRET_KEY
```

Then, update the period:

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

5.9.3.2. Deleting a Zone

To delete zone, first remove it from the zonegroup.

```
# radosgw-admin zonegroup remove --rgw-zonegroup=<name>\
                               --rgw-zone=<name>
```

Then, update the period:

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

Next, delete the zone.

IMPORTANT

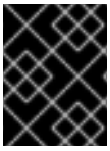
This procedure **MUST** be executed on a host within the zone.

Execute the following:

```
[root@zone]# radosgw-admin zone delete --rgw-zone<name>
```

Finally, update the period:

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



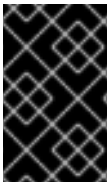
IMPORTANT

Do not delete a zone without removing it from a zone group first. Otherwise, updating the period will fail.

If the pools for the deleted zone will not be used anywhere else, consider deleting the pools. Replace **<del-zone>** in the example below 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.



IMPORTANT

In a multi-realm cluster, deleting the **.rgw.root** pool along with the zone pools will remove 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 <del-zone>.rgw.control <del-zone>.rgw.control --yes-i-really-really-mean-it
# ceph osd pool delete <del-zone>.rgw.data.root <del-zone>.rgw.data.root --yes-i-really-really-mean-it
# ceph osd pool delete <del-zone>.rgw.log <del-zone>.rgw.log --yes-i-really-really-mean-it
# ceph osd pool delete <del-zone>.rgw.users.uid <del-zone>.rgw.users.uid --yes-i-really-really-mean-it
```

IMPORTANT

After deleting the pools, restart the RGW process.

5.9.3.3. Modifying a Zone

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

IMPORTANT

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

```
[root@zone]# radosgw-admin zone modify [options]
```

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

Then, update the period:

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

5.9.3.4. Listing Zones

As **root**, to list the zones in a cluster, execute:

```
# radosgw-admin zone list
```

5.9.3.5. Getting a Zone

As **root**, to get the configuration of a zone, execute:

```
# radosgw-admin zone get [--rgw-zone=<zone>]
```

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"}
     }
   ]
  }
```

5.9.3.6. Setting a Zone

Configuring a zone involves specifying a series of Ceph Object Gateway pools. For consistency, we recommend using a pool prefix that is the same as the zone name. See Pools_ for details of 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 (e.g., **zone.json**); then, execute the following command, replacing **{zone-name}** with the name of the zone:

```
[root@zone]# radosgw-admin zone set --rgw-zone={zone-name} --infile zone.json
```

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

Then, as **root**, update the period:

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

5.9.3.7. Renaming a Zone

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

```
[root@zone]# radosgw-admin zone rename --rgw-zone=<name> --zone-new-name=<name>
```

Then, update the period:

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

5.10. ZONE GROUP AND ZONE CONFIGURATION SETTINGS

When configuring a default zone group and zone, the pool name includes the zone name. For example:

- **default.rgw.control**

To change the defaults, include the following settings in your Ceph configuration file under each **[client.rgw.{instance-name}]** instance.

Name	Description	Type	Default
rgw_zone	The name of the zone for the gateway instance.	String	None
rgw_zonegroup	The name of the zone group for the gateway instance.	String	None
rgw_zonegroup_root_pool	The root pool for the zone group.	String	**.rgw.root**
rgw_zone_root_pool	The root pool for the zone.	String	**.rgw.root**
rgw_default_zone_group_info_oid	The OID for storing the default zone group. We do not recommend changing this setting.	String	**default.zonegroup**

5.11. MANUALLY RESHARDING BUCKETS WITH MULTISITE

To manually reshard buckets in a multisite cluster, use the following procedure.

NOTE

Manual resharding is a very expensive process, especially for huge buckets that warrant manual resharding. Every secondary zone deletes all of the objects, and then resynchronizes them from the master zone.

Prerequisites

- Stop all Ceph Object Gateway instances.

Procedure

1. On a node within the master zone of the master zone group, execute the following command:

   Syntax

   ```
   # radosgw-admin bucket sync disable --bucket=BUCKET_NAME
   ```

   Wait for **sync status** on *all zones* to report that data synchronization is up to date.

2. Stop **ALL ceph-radosgw** daemons in **ALL** zones.

3. On a node within the master zone of the master zone group, reshard the bucket.

   Syntax

   ```
   # radosgw-admin bucket reshard --bucket=BUCKET_NAME --num-shards=NEW_SHARDS_NUMBER
   ```

4. On **EACH** secondary zone, execute the following:

   Syntax

   ```
   # radosgw-admin bucket rm --purge-objects --bucket=BUCKET_NAME
   ```

5. Restart **ALL ceph-radosgw** daemons in **ALL** zones.

6. On a node within the master zone of the master zone group, execute the following command:

   Syntax

   ```
   # radosgw-admin bucket sync enable --bucket=BUCKET_NAME
   ```

The metadata synchronization process will fetch the updated bucket entry point and bucket instance metadata. The data synchronization process will perform a full synchronization.

Additional resources

- See the *Configuring Bucket Index Sharding in Multi-site Configurations* in the *Red Hat Ceph Storage Object Gateway Configuration and Administration Guide* for more details.

5.12. CONFIGURING MULTIPLE ZONES WITHOUT REPLICATION

You can configure multiple zones that will not replicate each other. For example you can create a dedicated zone for each team in a company.

Prerequisites

- A Ceph Storage Cluster with the Ceph Object Gateway installed.

Procedure

1. Create a realm.

   ```
   radosgw-admin realm create --rgw-realm=realm-name [--default]
   ```

   For example:

   ```
   [root@master-zone]# radosgw-admin realm create --rgw-realm=movies --default
   {
       "id": "0956b174-fe14-4f97-8b50-bb7ec5e1cf62",
       "name": "movies",
       "current_period": "1950b710-3e63-4c41-a19e-46a715000980",
       "epoch": 1
   }
   ```

2. Create a zone group.

   ```
   radosgw-admin zonegroup create --rgw-zonegroup=zone-group-name --endpoints=url [--rgw-realm=realm-name|--realm-id=realm-id] --master --default
   ```

   For example:

```
[root@master-zone]# radosgw-admin zonegroup create --rgw-zonegroup=us --
endpoints=http://rgw1:80 --rgw-realm=movies --master --default
{
    "id": "f1a233f5-c354-4107-b36c-df66126475a6",
    "name": "us",
    "api_name": "us",
    "is_master": "true",
    "endpoints": [
        "http:\/\/rgw1:80"
    ],
    "hostnames": [],
    "hostnames_s3webzone": [],
    "master_zone": "",
    "zones": [],
    "placement_targets": [],
    "default_placement": "",
    "realm_id": "0956b174-fe14-4f97-8b50-bb7ec5e1cf62"
}
```

3. Create one or more zones depending on your use case.

```
radosgw-admin zone create
            --rgw-zonegroup=zone-group-name \
            --rgw-zone=zone-name \
            --master --default \
            --endpoints=http://fqdn:port[,http://fqdn:port]
```

For example:

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

4. Get the JSON file with the configuration of the zone group.

```
radosgw-admin zonegroup get --rgw-zonegroup=zone-group-name > zonegroup.json
```

For example:

```
[root@master-zone]# radosgw-admin zonegroup get --rgw-zonegroup=us > zonegroup.json
```

5. In the file, set the **log_meta**, **log_data**, and **sync_from_all** parameters to **false**.

```
    {
        "id": "72f3a886-4c70-420b-bc39-7687f072997d",
        "name": "default",
        "api_name": "",
        "is_master": "true",
        "endpoints": [],
        "hostnames": [],
        "hostnames_s3website": [],
        "master_zone": "a5e44ecd-7aae-4e39-b743-3a709acb60c5",
        "zones": [
```

```
            {
                "id": "975558e0-44d8-4866-a435-96d3e71041db",
                "name": "testzone",
                "endpoints": [],
                "log_meta": "false",
                "log_data": "false",
                "bucket_index_max_shards": 11,
                "read_only": "false",
                "tier_type": "",
                "sync_from_all": "false",
                "sync_from": []
            },
            {
                "id": "a5e44ecd-7aae-4e39-b743-3a709acb60c5",
                "name": "default",
                "endpoints": [],
                "log_meta": "false",
                "log_data": "false",
                "bucket_index_max_shards": 11,
                "read_only": "false",
                "tier_type": "",
                "sync_from_all": "false",
                "sync_from": []
            }
        ],
        "placement_targets": [
            {
                "name": "default-placement",
                "tags": []
            }
        ],
        "default_placement": "default-placement",
        "realm_id": "2d988e7d-917e-46e7-bb18-79350f6a5155"
    }
```

6. Use the updated JSON file.

```
radosgw-admin zonegroup set --rgw-zonegroup=zone-group-name --infile=zonegroup.json
```

For example:

```
[root@master-zone]# radosgw-admin zonegroup set --rgw-zonegroup=us --
infile=zonegroup.json
```

7. Update the period.

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

Additional Resources

- *Realms*
- *Zone Groups*

- *Zones*
- *Installation Guide*

5.13. CONFIGURING MULTIPLE REALMS IN THE SAME STORAGE CLUSTER

This section discusses how to 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

Red Hat recommends that each realm has its own Ceph Object Gateway.

Prerequisites

- The access key and secret key for each data center in the storage cluster.
- Two running Red Hat Ceph Storage data centers in a storage cluster.
- Root-level or sudo access to all the nodes.
- Each data center has its own local realm. They share a realm that replicates on both sites.
- On the Ceph Object Gateway nodes, perform the tasks listed in the *Requirements for Installing Red Hat Ceph Storage* found in the *Red Hat Ceph Storage Installation Guide*.
- For each Ceph Object Gateway node, perform steps 1-7 in the *Installing the Ceph Object Gateway* section of the *Red Hat Ceph Storage Installation Guide*.

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

   ```
   [root@rgw1 ~]# 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

```
[root@rgw1 ~]# 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

   ```
   [root@rgw1 ~]# radosgw-admin zone create --rgw-zonegroup=ldc1zg --rgw-zone=ldc1z --
   master --default --endpoints=http://rgw.example.com
   ```

4. Commit the period:

   Example

   ```
   [root@rgw1 ~]# radosgw-admin period update --commit
   ```

5. Update **ceph.conf** with the **rgw_realm**, **rgw_zonegroup** and **rgw_zone** names:

   Syntax

   ```
   rgw_realm = REALM_NAME
   rgw_zonegroup = ZONE_GROUP_NAME
   rgw_zone = ZONE_NAME
   ```

   Example

   ```
   rgw_realm = ldc1
   rgw_zonegroup = ldc1zg
   rgw_zone = ldc1z
   ```

6. Restart the RGW daemon:

   Syntax

   ```
   systemctl restart ceph-radosgw@rgw.$(hostname -s).rgw0.service
   ```

7. Create one local realm on the second data center in the storage cluster:

   Syntax

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

   Example

   ```
   [root@rgw2 ~]# radosgw-admin realm create --rgw-realm=ldc2 --default
   ```

8. 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

   ```
   [root@rgw2 ~]# radosgw-admin zonegroup create --rgw-zonegroup=ldc2zg --
   endpoints=http://rgw2:80 --rgw-realm=ldc2 --master --default
   ```

9. 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

   ```
   [root@rgw2 ~]# radosgw-admin zone create --rgw-zonegroup=ldc2zg --rgw-zone=ldc2z --
   master --default --endpoints=http://rgw.example.com
   ```

10. Commit the period:

    Example

    ```
    [root@rgw2 ~]# radosgw-admin period update --commit
    ```

11. Update **ceph.conf** with the **rgw_realm**, **rgw_zonegroup** and **rgw_zone** names:

    Syntax

    ```
    rgw_realm = REALM_NAME
    rgw_zonegroup = ZONE_GROUP_NAME
    rgw_zone = ZONE_NAME
    ```

    Example

    ```
    rgw_realm = ldc2
    rgw_zonegroup = ldc2zg
    rgw_zone = ldc2z
    ```

12. Restart the RGW daemon:

    Syntax

    ```
    systemctl restart ceph-radosgw@rgw.$(hostname -s).rgw0.service
    ```

13. 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

```
[user@rgw1 ~] radosgw-admin realm create --rgw-realm=rdc1 --default
```

Use the **--default** flag to make the replicated realm default on the primary site.

14. 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

```
[root@rgw1 ~]# radosgw-admin zonegroup create --rgw-zonegroup=rdc1zg --
endpoints=http://rgw1:80 --rgw-realm=rdc1 --master --default
```

15. 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

```
[root@rgw1 ~]# radosgw-admin zone create --rgw-zonegroup=rdc1zg --rgw-zone=rdc1z --
master --default --endpoints=http://rgw.example.com
```

16. Create a replication/synchronization user and add the system user to the master zone for multi-site:

Syntax

```
radosgw-admin user create --uid="r_REPLICATION_SYNCHRONIZATION_USER_" --
display-name="Replication-Synchronization User" --system
radosgw-admin zone modify --rgw-zone=RGW_ZONE --access-key=ACCESS_KEY --
secret=SECRET_KEY
```

Example

```
[root@rgw1 ~]# radosgw-admin zone modify --rgw-zone=rdc1zg --access-
key=3QV0D6ZMMCJZMSCXJ2QJ --
secret=VpvQWcsfI9OPzUCpR4kynDLAbqa1OIKqRB6WEnH8
```

17. Commit the period:

    Syntax

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

18. Update **ceph.conf** with the **rgw_realm**, **rgw_zonegroup** and **rgw_zone** names for the first data center:

    Syntax

    ```
    rgw_realm = REALM_NAME
    rgw_zonegroup = ZONE_GROUP_NAME
    rgw_zone = ZONE_NAME
    ```

    Example

    ```
    rgw_realm = rdc1
    rgw_zonegroup = rdc1zg
    rgw_zone = rdc1z
    ```

19. Restart the RGW daemon:

    Syntax

    ```
    systemctl restart ceph-radosgw@rgw.$(hostname -s).rgw0.service
    ```

20. 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

    ```
    radosgw-admin realm pull --url=https://tower-osd1.cephtips.com --access-
    key=3QV0D6ZMMCJZMSCXJ2QJ --secret-
    key=VpvQWcsfI9OPzUCpR4kynDLAbqa1OIKqRB6WEnH8
    ```

21. 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

    ```
    radosgw-admin period pull --url=https://tower-osd1.cephtips.com --access-
    key=3QV0D6ZMMCJZMSCXJ2QJ --secret-
    key=VpvQWcsfI9OPzUCpR4kynDLAbqa1OIKqRB6WEnH8
    ```

22. 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

    ```
    [root@rgw2 ~]# radosgw-admin zone create --rgw-zone=rdc2z --rgw-zonegroup=rdc1zg --endpoints=https://tower-osd4.cephtips.com --access-key=3QV0D6ZMMCJZMSCXJ2QJ --secret-key=VpvQWcsfI9OPzUCpR4kynDLAbqa1OIKqRB6WEnH8
    ```

23. Commit the period:

    Syntax

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

24. Update **ceph.conf** with the **rgw_realm**, **rgw_zonegroup** and **rgw_zone** names for the second data center:

    Syntax

    ```
    rgw_realm = REALM_NAME
    rgw_zonegroup = ZONE_GROUP_NAME
    rgw_zone = ZONE_NAME
    ```

    Example

    ```
    rgw realm = rdc1
    rgw zonegroup = rdc1zg
    rgw zone = rdc2z
    ```

25. Restart the Ceph Object Gateway daemon:

    Syntax

    ```
    systemctl restart ceph-radosgw@rgw.$(hostname -s).rgw0.service
    ```

26. Log in to the second data center and verify the synchronization status on the master realm:

    Syntax

    ```
    radosgw-admin sync status
    ```

    Example

    ```
    [root@rgw2 ~]# 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)
```

27. Log in to the first data center and verify the synchronization status for the replication-synchronization realm:

Syntax

```
radosgw-admin sync status --rgw-realm RGW_REALM_NAME
```

Example

```
[root@rgw1 ~]# 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
```

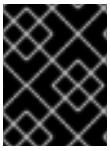
28. To store and access data in the local site, create the user for local realm:

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

```
[root@rgw2 ~]# 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.