



# **Red Hat Ceph Storage 2**

## **Object Gateway Guide for Red Hat Enterprise Linux**

Configuring and administering the Ceph Storage Object Gateway on Red Hat Enterprise Linux



# Red Hat Ceph Storage 2 Object Gateway Guide for Red Hat Enterprise Linux

---

Configuring and administering the Ceph Storage Object Gateway on Red Hat Enterprise Linux

## Legal Notice

Copyright © 2018 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

## Abstract

This document provides instructions for configuring and administering the Ceph Storage Object Gateway on Red Hat Enterprise Linux 7 running on AMD64 and Intel 64 architectures.

## Table of Contents

<b>CHAPTER 1. OVERVIEW .....</b>	<b>5</b>
<b>CHAPTER 2. CONFIGURATION .....</b>	<b>6</b>
2.1. CHANGING YOUR DEFAULT PORT	6
2.2. MIGRATING FROM APACHE TO CIVETWEB	7
2.3. USING SSL WITH CIVETWEB	7
2.4. CIVETWEB CONFIGURATION OPTIONS	8
2.5. ADDING A WILDCARD TO DNS	9
2.6. ADJUSTING LOGGING AND DEBUGGING OUTPUT	10
2.7. TESTING THE OBJECT GATEWAY	10
2.7.1. Creating a radosgw User for S3 Access	10
2.7.2. Creating a Swift User	11
2.7.3. Testing S3 Access	13
2.7.4. Testing Swift Access	14
2.8. HAPROXY/KEEPALIVED CONFIGURATION	15
2.8.1. Prerequisites	15
2.8.2. Preparing HAProxy Nodes	16
2.8.3. Installing and Configuring keepalived	16
2.8.4. Installing and Configuring HAProxy	17
2.8.5. Testing the HAProxy Configuration	19
2.9. CONFIGURING GATEWAYS FOR STATIC WEB HOSTING	19
2.9.1. Assumptions	19
2.9.2. Requirements	20
2.9.3. Setting Up the Gateway	20
2.9.4. Configuring the DNS	20
2.9.5. Creating a Site	22
2.10. EXPORTING THE NAMESPACE TO NFS-GANESHA	22
Running Multiple NFS Gateways	23
Before you Start	23
Configuring an NFS-Ganesha Instance	24
Configuring NFSv4 clients	26
Configuring NFSv3 clients (Technology Preview)	26
<b>CHAPTER 3. ADMINISTRATION (CLI) .....</b>	<b>28</b>
3.1. ADMINISTRATIVE DATA STORAGE	28
3.2. STORAGE POLICIES	29
3.3. INDEXLESS BUCKETS	31
3.4. BUCKET SHARDING	32
Configuring Bucket Index Sharding	32
Bucket Index Resharding	34
Bucket Sharding Limitations	35
3.5. RADOS GATEWAY USER MANAGEMENT	35
3.5.1. Multi Tenancy	36
3.5.2. Create a User	36
3.5.3. Create a Subuser	37
3.5.4. Get User Information	38
3.5.5. Modify User Information	38
3.5.6. Enable and Suspend Users	38
3.5.7. Remove a User	39
3.5.8. Remove a Subuser	39
3.5.9. Create a Key	39
3.5.10. Add and Remove Access Keys	40

3.5.11. Add and Remove Admin Capabilities	40
3.6. QUOTA MANAGEMENT	41
3.6.1. Set User Quotas	41
3.6.2. Enable and Disable User Quotas	42
3.6.3. Set Bucket Quotas	42
3.6.4. Enable and Disable Bucket Quotas	42
3.6.5. Get Quota Settings	42
3.6.6. Update Quota Stats	42
3.6.7. Get User Quota Usage Stats	42
3.6.8. Quota Cache	43
3.6.9. Reading and Writing Global Quotas	43
3.7. USAGE	43
3.7.1. Show Usage	44
3.7.2. Trim Usage	44
3.7.3. Finding Orphan Objects	44
<b>CHAPTER 4. OBJECT GATEWAY CONFIGURATION REFERENCE .....</b>	<b>46</b>
4.1. REALMS	49
4.1.1. Create a Realm	50
4.1.2. Make a Realm the Default	50
4.1.3. Delete a Realm	50
4.1.4. Get a Realm	50
4.1.5. Set a Realm	51
4.1.6. List Realms	51
4.1.7. List Realm Periods	51
4.1.8. Pull a Realm	51
4.1.9. Rename a Realm	52
4.2. ZONE GROUPS	52
4.2.1. Create a Zone Group	52
4.2.2. Make a Zone Group the Default	52
4.2.3. Add a Zone to a Zone Group	53
4.2.4. Remove a Zone from a Zone Group	53
4.2.5. Rename a Zone Group	53
4.2.6. Delete a Zone Group	53
4.2.7. List Zone Groups	54
4.2.8. Get a Zone Group Map	54
4.2.9. Get a Zone Group	54
4.2.10. Set a Zone Group	55
4.2.11. Set a Zone Group Map	56
4.3. ZONES	58
4.3.1. Create a Zone	58
4.3.2. Delete a Zone	58
4.3.3. Modify a Zone	59
4.3.4. List Zones	60
4.3.5. Get a Zone	60
4.3.6. Set a Zone	60
4.3.7. Rename a Zone	61
4.4. ZONE GROUP AND ZONE SETTINGS	61
4.5. POOLS	61
4.6. SWIFT SETTINGS	63
4.7. LOGGING SETTINGS	63
4.8. KEYSTONE SETTINGS	65
4.9. LDAP SETTINGS	65

---

<b>CHAPTER 5. MULTI-SITE .....</b>	<b>67</b>
5.1. FUNCTIONAL CHANGES FROM RED HAT CEPH STORAGE 1.X	67
5.2. REQUIREMENTS AND ASSUMPTIONS	68
5.3. POOLS	68
5.4. INSTALLING AN OBJECT GATEWAY	69
5.4.1. Installing using the Command Line	69
5.4.2. Installing an Object Gateway using Ansible	69
5.4.3. Updating an Object Gateway Installation	70
5.5. CONFIGURING A MASTER ZONE	70
5.5.1. Create a Realm	70
5.5.2. Create a Master Zone Group	71
5.5.3. Create a Master Zone	72
5.5.4. Delete Default Zone Group and Zone	72
5.5.5. Create a System User	73
5.5.6. Update the Period	73
5.5.7. Update the Ceph Configuration File	73
5.5.8. Start the Gateway	74
5.6. CONFIGURE SECONDARY ZONES	74
5.6.1. Pull the Realm	74
5.6.2. Pull the Period	75
5.6.3. Create a Secondary Zone	75
5.6.4. Update the Ceph Configuration File	76
5.6.5. Update the Period	76
5.6.6. Start the Gateway	76
5.6.7. Check Synchronization Status	77
5.7. FAILOVER AND DISASTER RECOVERY	77
5.8. MIGRATING A SINGLE SITE SYSTEM TO MULTI-SITE	78
5.9. MIGRATING A 1.3 MULTI-SITE SYSTEM TO VERSION 2	79



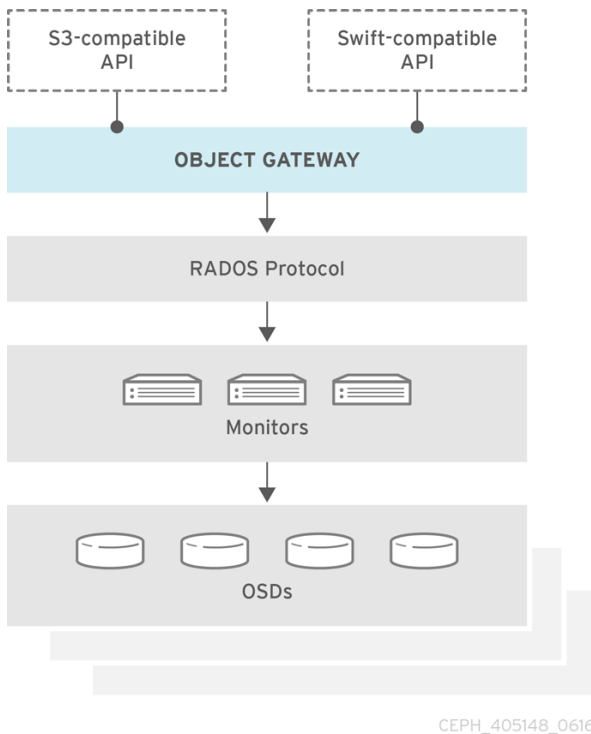


# CHAPTER 1. OVERVIEW

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.



## CHAPTER 2. CONFIGURATION

### 2.1. CHANGING YOUR DEFAULT PORT

Civetweb runs on port 7480 by default. To change the default port (e.g., to port 80), modify your Ceph configuration file in `/etc/ceph` directory of your administration server. Add a section entitled `[client.rgw.<gateway-node>]`, replacing `<gateway-node>` with the short node name of your Ceph Object Gateway node (i.e., `hostname -s`).



#### NOTE

In version 1.3, the Ceph Object Gateway does not support SSL. You may setup a reverse proxy web server with SSL to dispatch HTTPS requests as HTTP requests to CivetWeb.

For example, if your node name is `gateway-node1`, add a section like this after the `[global]` section in `/etc/ceph/ceph.conf` file:

```
[client.rgw.gateway-node1]
rgw_frontends = "civetweb port=80"
```



#### NOTE

Ensure that you leave no whitespace between `port=<port-number>` in the `rgw_frontends` key/value pair. 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 (i.e., `rgw`), and the name of the instance is `gateway-node1`.

Copy the updated configuration file from `/etc/ceph` directory to the working directory of your administration server.

```
# scp /etc/ceph/ceph.conf <admin-server>:/etc/ceph
```

Then, copy the updated configuration file to your Ceph Object Gateway node and other Ceph nodes. From the working directory of your administration server, execute:

```
# ssh <admin-server>
# scp /etc/ceph/ceph.conf <ceph-node>:/etc/ceph
```

To make the new port setting take effect, from your Ceph Object Gateway node, restart the Ceph Object Gateway.

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

Finally, check to ensure that the port you selected is open on the node's firewall (e.g., port 80). If it is not open, add the port and reload the firewall configuration. For example, on your Ceph Object Gateway node, execute:

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

## 2.2. MIGRATING FROM APACHE TO CIVETWEB

If you're running the Ceph Object Gateway on Apache and FastCGI with Red Hat Ceph Storage v1.2.x or above, you're already running Civetweb—it starts with the `ceph-radosgw` daemon and it's running on port 7480 by default so that it doesn't conflict with your Apache and FastCGI installation and other commonly used web service ports. Migrating to use Civetweb basically involves removing your Apache installation. Then, you must remove Apache and FastCGI settings from your Ceph configuration file and reset `rgw_frontends` to Civetweb.

Referring to the documentation for installing a Ceph Object Gateway notice that the configuration file has an `rgw_frontends` setting, which enables you to specify civetweb as a front end and change its port. Since you already have keys and a data directory, you will want to maintain those paths in your Ceph configuration file if you used something other than default paths.

A typical Ceph Object Gateway configuration file for an Apache-based deployment looks something like this:

```
[client.rgw.gateway-node1]
host = {hostname}
keyring = /etc/ceph/ceph.client.radosgw.keyring
rgw socket path = ""
log file = /var/log/radosgw/client.radosgw.gateway-node1.log
rgw frontends = fastcgi socket_port=9000 socket_host=0.0.0.0
rgw print continue = false
```

To modify it for use with Civetweb, simply remove the Apache-specific settings such as `rgw_socket_path` and `rgw_print_continue`. Then, change the `rgw_frontends` setting to reflect Civetweb rather than the Apache FastCGI front end and specify the port number you intend to use. For example:

```
[client.rgw.gateway-node1]
host = {hostname}
keyring = /etc/ceph/ceph.client.radosgw.keyring
log file = /var/log/radosgw/client.radosgw.gateway-node1.log
rgw_frontends = civetweb port=80
```

Finally, on your Ceph Object Gateway execute the following to restart the Ceph Object Gateway:

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

If you used a port number that is not open, you will also need to open that port on your firewall.

## 2.3. USING SSL WITH CIVETWEB

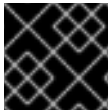
In previous versions, Civetweb SSL support for the Ceph Object Gateway relied on HAProxy and keepalived. In Red Hat Ceph Storage 2, Civetweb can use the OpenSSL library to provide Transport Layer Security (TLS).

**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. 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"
```

## 2.4. 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
<code>access_log_file</code>	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
<code>error_log_file</code>	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
<code>num_threads</code>	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
<code>request_timeout_ms</code>	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
```

## 2.5. ADDING A WILDCARD TO 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 your DNS server and ping your 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]
...
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.

## 2.6. ADJUSTING LOGGING AND DEBUGGING OUTPUT

Once you finish the setup procedure, check your logging output to ensure it meets your needs. Log files are located in `/var/log/radosgw` by default. 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 rgw = 20
debug civetweb = 20
```

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

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

For general details on logging and debugging, see [Logging and Debugging](#). For Ceph Object Gateway-specific details on logging settings, see [Logging Settings](#) in this guide.

## 2.7. TESTING THE OBJECT 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.

### 2.7.1. Creating a radosgw User for S3 Access

A `radosgw` user needs to be created and granted access. The command `man radosgw-admin` will provide information on additional command options.

To create the user, execute the following on the `gateway` host:

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

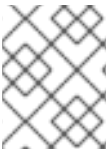
The output of the command will be something like the following:

```
{
```

```

"user_id": "testuser",
"display_name": "First User",
"email": "",
"suspended": 0,
"max_buckets": 1000,
"aud": 0,
"subusers": [],
"keys": [{
  "user": "testuser",
  "access_key": "I0PJDPICIYZ665MW88W9R",
  "secret_key": "dxaXZ8U90SXydYzyS5ivamEP20hkLSUViiaR+ZDA"
}],
"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": []
}

```



#### NOTE

The values of **keys→access\_key** and **keys→secret\_key** are needed for access validation.



#### IMPORTANT

Check the key output. Sometimes **radosgw-admin** generates a JSON escape character `\` in **access\_key** or **secret\_key** 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. Also, if **radosgw-admin** generates a JSON escape character `\` and a forward slash `/` together in a key, like `\/`, only remove the JSON escape character `\`. Do not remove the forward slash `/` as it is a valid character in the key.

### 2.7.2. Creating a Swift User

A Swift subuser needs to be created if this kind of access is needed. Creating a Swift user is a two step process. The first step is to create the user. The second is to create the secret key.

Execute the following steps on the **gateway** host:

Create the Swift user:

```
sudo radosgw-admin subuser create --uid=testuser --subuser=testuser:swift
--access=full
```

The output will be something like the following:

```
{
  "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:swift",
    "access_key": "3Y1LNW4Q6X0Y53A52DET",
    "secret_key": ""
  }, {
    "user": "testuser",
    "access_key": "I0PJDPICIYZ665MW88W9R",
    "secret_key": "dxaXZ8U90SXydYzyS5ivamEP20hkLSUViiaR+ZDA"
  }],
  "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": []
}
```

Create the secret key:

```
sudo radosgw-admin key create --subuser=testuser:swift --key-type=swift --
gen-secret
```

The output will be something like the following:

```
{
  "user_id": "testuser",
  "display_name": "First User",
  "email": "",
  "suspended": 0,
```



```

    "max_buckets": 1000,
    "aud": 0,
    "subusers": [{
        "id": "testuser:swift",
        "permissions": "full-control"
    }],
    "keys": [{
        "user": "testuser:swift",
        "access_key": "3Y1LNW4Q6X0Y53A52DET",
        "secret_key": ""
    }, {
        "user": "testuser",
        "access_key": "I0PJDPICIYZ665MW88W9R",
        "secret_key": "dxaXZ8U90SXydYzyS5ivamEP20hkLSUViiaR+ZDA"
    }],
    "swift_keys": [{
        "user": "testuser:swift",
        "secret_key": "244+fz2gSsqoHwR3lYtSbIyomyPHf3i7rgSJrF\//IA"
    }],
    "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": []
}

```

### 2.7.3. Testing 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.

Execute the following steps:

1. Enable the common repository.

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

2. Install the `python-boto` package.

```
sudo yum install python-boto
```

3. Create the Python script:

■

```
vi s3test.py
```

4. Add the following contents to the file:

```
import boto
import boto.s3.connection

access_key = $access
secret_key = $secret

boto.config.add_section('s3')
boto.config.set('s3', 'use-sigv4', 'True')

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,
    )
```

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.

5. Run the script:

```
python s3test.py
```

The output will be something like the following:

```
my-new-bucket 2015-02-16T17:09:10.000Z
```

## 2.7.4. Testing 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., **7480** is the default). If you don't replace the port, it will default to port **80**.

For example:

```
swift -A http://10.19.143.116:7480/auth/1.0 -U testuser:swift -K
'244+fz2gSqoHwR3lYtSbIyomyPHf3i7rgSJrF/IA' list
```

The output should be:

```
my-new-bucket
```



### WARNING

During uploads of large objects to versioned swift containers, please use the option **--leave-segments** in the upload using **python-swiftclient**. Not using this option will lead to an overwrite of the manifest file in which case an existing object is overwritten, leading to data loss.

## 2.8. HAProxy/KEEPALIVED CONFIGURATION

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 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. Red Hat Ceph Storage (RHCS) 1.3.x uses Civetweb, and the implementation in RHCS 1.3.x doesn't support HTTPS. 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.

### 2.8.1. 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 document 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](#).

### 2.8.2. 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 RHEL 7.x.

2. Register the nodes.

```
sudo subscription-manager register
```

3. Enable the RHEL server repository.

```
sudo subscription-manager repos --enable=rhel-7-server-rpms
```

4. Update the server.

```
sudo yum update -y
```

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

6. Open port 80.

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

7. For HTTPS, open port 443.

```
sudo firewall-cmd --zone=public --add-port 443/tcp --permanent  
sudo firewall-cmd --reload
```

### 2.8.3. Installing and Configuring keepalived

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

1. Install **keepalived**.

```
sudo yum install -y keepalived
```

2. Configure **keepalived**.

```
sudo vim /etc/keepalived/keepalived.conf
```

In the following configuration, there is a script to check the haproxy processes. The instance uses **eth0** as the network interface and configures **haproxy** as the master server and **haproxy2** as the backup server. It also assigns a virtual IP address (i.e., **192.168.0.100**).

```

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

vrp_instance VI_1 {
    interface eth0 # interface to monitor
    state MASTER # MASTER on haproxy, BACKUP on haproxy2
    virtual_router_id 51
    priority 101 # 101 on haproxy, 100 on haproxy2
    virtual_ipaddress {
        192.168.0.100 # virtual ip address
    }
    track_script {
        chk_haproxy
    }
}

```

For a detailed discussion of configuring **keepalived**, refer to [Initial Load Balancer Configuration with Keepalived](#).

### 3. Enable/start **keepalived**.

```

sudo systemctl enable keepalived
sudo systemctl start keepalived

```

## 2.8.4. Installing and Configuring HAProxy

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

### 1. Install **haproxy**.

```

sudo yum install haproxy

```

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

```

sudo 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.

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

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

```
sudo 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, refer to [generating a key](#). Finally, put the certificate and key into a PEM file.

```
cat example.com.crt example.com.key > example.com.pem
sudo cp example.com.pem /etc/ssl/private/
```

5. Configure **haproxy**.

```
sudo 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 *: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

```
sudo systemctl enable haproxy
sudo systemctl start haproxy
```

### 2.8.5. Testing the HAProxy Configuration

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

```
ip addr show
```

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

```
wget haproxy
```

This should return the same result as:

```
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.

## 2.9. CONFIGURING GATEWAYS FOR STATIC WEB HOSTING

Traditional web hosting sometimes involves setting up a web server for each website, which can use resources inefficiently when content doesn't 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.

### 2.9.1. 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 [HAProxy/keepalived Configuration](#) for additional details on HAProxy/keepalived.



#### NOTE

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

### 2.9.2. Requirements

Static web hosting functionality uses its own API, so configuring a gateway to use static web sites in S3 buckets requires the following:

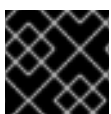
1. S3 static web hosting uses Ceph Object Gateway instances that are separate and distinct from instances used for standard S3/Swift API use cases.
2. Gateway instances hosting S3 static web sites should have separate, non-overlapping domain names from the standard S3/Swift API gateway instances.
3. 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.

### 2.9.3. Setting Up the Gateway

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 = 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.

### 2.9.4. Configuring the DNS

The following is an example of assumed DNS settings, where the first two lines specify the domains of the 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
```

## 2.9.5. Creating a 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 the contents.
4. Set permissions on the files so that they are publicly readable.

## 2.10. EXPORTING THE NAMESPACE TO NFS-GANESHA

In Red Hat Ceph Storage 2, the Ceph Object Gateway provides the ability to export S3 object namespaces by using NFS version 4.1 for production systems, and NFS version 3 as a Technology Preview only.



### NOTE

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

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.

The Ceph Object Gateway with NFS is based on a new, in-process library packaging of the Gateway server and a new File System Abstraction Layer (FSAL) namespace driver for the NFS-Ganesha NFSv4 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 limitation that currently 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 2 Tools repository:

```
# subscription-manager repos --enable=rhel-7-server-rhceph-2-tools-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



### IMPORTANT

NFSv4 is supported for production systems. NFSv3 is a Technology Preview and is not supported for production systems. Use NFSv3 with caution.

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. Copy the Object Gateway keyring from the Ceph Object Gateway host to the NFS Ganesha host.

- a. Create a directory to store the keyring:

```
# mkdir -p /var/lib/ceph/radosgw/ceph-rgw.<instance-name>/
```

- b. Set the correct ownership for the directory:

```
chown ceph.ceph /var/lib/ceph/radosgw/ceph-rgw.<instance-name>
```

- c. Copy the keyring:

```
# scp <rgw-instance-host>:/var/lib/ceph/radosgw/ceph-rgw.
```

```
<instance-name>/keyring <nfs-ganesha-rgw-
host>:/var/lib/ceph/radosgw/ceph-rgw-<instance-name>/.
```

4. Open the NFS-Ganesha configuration file:

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

5. 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}";
    }
}
```

NFSv3 is a Technology Preview and is **NOT supported** for production systems. 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_Numeric_Owners` parameters as an alternative to setting up the `idmapper` program.

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

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. Start the **nfs-ganesha** service.

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

### Configuring NFSv4 clients

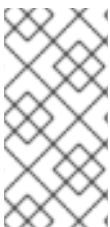
To access the namespace, mount the configured NFS-Ganesha export(s) into desired locations in the local POSIX namespace. As noted, this implementation has a few unique restriction:

- 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 (Technology Preview)

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 (CLI)

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

- [Administrative Data Storage](#)
- [Storage Policies](#)
- [Bucket Sharding](#)
- [User Management](#)
- [Quota Management](#)
- [Usage Tracking](#)

### 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`
- `.rgw.control`
- `.rgw.gc`
- `.log`
- `.intent-log`
- `.usage`
- `.users`
- `.users.email`
- `.users.swift`
- `.users.uid`

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 [Storage Strategies](#) 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. 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. 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

```
$ radosgw-admin zonegroup --rgw-zonegroup=<zonegroup_name> [--cluster <cluster_name>] get > zonegroup.json
```

### Example

```
$ 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": 5
  }],
  "placement_targets": [{
    "name": "default-placement",
    "tags": []
  }, {
    "name": "special-placement",
    "tags": []
  }],
  "default_placement": "default-placement"
}
```

4. Set the zone group with the modified `zonegroup.json` file:

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

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

```
$ 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.

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

8. Update the zone group map.

```
$ 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. INDEXLESS BUCKETS**

It is possible to configure a placement target where created buckets do not use the bucket index to store objects index—i.e., indexless buckets. If you are not using data replication or listing, you may want to implement indexless buckets in your environment. Indexless buckets provides a mechanism in which RGW 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 RGW needs to make to the RADOS backend. This can have a positive effect on concurrent operations and small object writes performance.

To specify a placement target as indexless, use the following procedure:

```
$ radosgw-admin zone get --rgw-zone=<zone> > zone.json
```

Modify `zone.json` by adding a new placement target or by modifying an existing one to have `"index_type": 1`, for example:

```
"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
    }
  },
  {
    "key": "indexeless",
    "val": {
      "index_pool": "default.rgw.buckets.index",
      "data_pool": "default.rgw.buckets.data",
      "data_extra_pool": "default.rgw.buckets.non-ec",
      "index_type": 1
    }
  }
],
```

```
$ radosgw-admin zone set --rgw-zone=<zone> --infile zone.json
```

Make sure the `zonegroup` refers to the new placement target (if you created a new one):

```
$ radosgw-admin zonegroup get --rgw-zonegroup=<zonegroup> > zonegroup.json
```

Modify the `zonegroup.json` as needed, for example:

```
"placement_targets": [
  {
```

```

    "name": "default-placement",
    "tags": []
  },
  {
    "name": "indexless",
    "tags": []
  }
],
"default_placement": "default-placement",

```

```
$ radosgw-admin zonegroup set --rgw-zonegroup=<zonegroup> < zonegroup.json
```

Update and commit period (if in a multi-site configuration):

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

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



### 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. It is not recommended 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

Indexless buckets cannot be viewed from NFS.

## 3.4. BUCKET SHARDING

The Ceph Object Gateway stores bucket index data in the index pool (**index\_pool1**), which defaults to **.rgw.buckets.index**. If you put 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.

See [Configuring Bucket Index Sharding](#) for details on configuring bucket index sharding for new buckets.

See [Bucket Index Resharding](#) for details on changing the bucket index sharding on already existing buckets.

### Configuring Bucket Index Sharding

To enable and configure bucket index sharding on all new buckets, use

- the `rgw_override_bucket_index_max_shards` setting for simple configurations,
- the `bucket_index_max_shards` setting for multi-site configurations.

Set the settings 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.

Use the following formula to calculate the recommended number of shards:

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

Note that maximum number of shards is 7877.

### Simple configurations

1. Add `rgw_override_bucket_index_max_shards` to the Ceph configuration file:

```
rgw_override_bucket_index_max_shards = 10
```

- 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.

2. Restart the Ceph Object Gateway:

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

### Multi-site configurations

In multi-site 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. To do so:

1. Extract the zone group configuration to the `zonegroup.json` file:

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

2. In the `zonegroup.json` file, set the `bucket_index_max_shards` setting for each named zone.

3. Reset the zone group:

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

4. Update the period:

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

**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.

**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 object.
- Spreads all objects entries of these 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.

To reshard the bucket index pool:

1. Make sure that all operations to the bucket are stopped.
2. Back the original bucket index up:

```
# radosgw-admin bi list --bucket=<bucket_name> >
<bucket_name>.list.backup
```

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

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

3. Reshard the bucket index:

```
# radosgw-admin reshard --bucket=<bucket_name> --num-shards=
<new_shards_number>
```

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

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

As part of its output, this command also prints the new and the old bucket ID. Note the old bucket ID down; you will need it to purge the old bucket index objects.

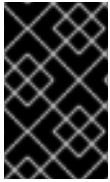
4. Verify that the objects are listed correctly by comparing the old bucket index listing with the new one.
5. Purge the old bucket index objects:

```
# radosgw-admin bi purge --bucket=<bucket_name> --bucket-id=
<old_bucket_id>
```

For example, for a bucket named **data** and the old bucket ID being **123456**, enter:

```
# radosgw-admin bi purge --bucket=data --bucket-id=123456
```

## Bucket Sharding Limitations



### IMPORTANT

The following limitations should be used 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 128,000 objects per bucket index shard. To take full advantage of sharding, you need to provide sufficient OSDs in the Ceph Object Gateway bucket index pool to get maximum parallelism.
- **Maximum number of objects when using sharding:** The number of bucket index shards currently supported is 7877.

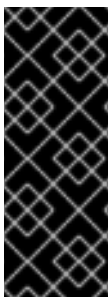
## 3.5. RADOS GATEWAY USER MANAGEMENT

Ceph Object Storage user management refers to users that are client applications of the Ceph Object Storage service (i.e., 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 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.

### 3.5.1. Multi Tenancy

In Red Hat Ceph Storage 2, 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 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:test --key-type swift --access full user
create
```

```
[root@master-zone]# radosgw-admin key create --subuser 'testx$tester:test'
--key-type swift --secret test123
```



#### NOTE

The subuser with explicit tenant had to be quoted in the shell.

### 3.5.2. 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.

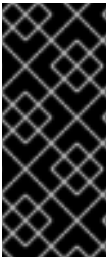
```
[root@master-zone]# radosgw-admin user create --uid=<id> \
[--key-type=<type>] [--gen-access-key|--access-key=<key>]\
[--gen-secret|--secret=<key>] \
[--email=<email>] --display-name=<name>
```

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": "vzCEkuryfn060dfee4fgQPqFrncKEIkh3Zcd0ANY"}],
  "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.

### 3.5.3. 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": "vzCEkuryfn060dfee4fgQPqFrncKEIkh3Zcd0ANY"}],
  "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": []}
```

### 3.5.4. Get User Information

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

```
radosgw-admin user info --uid=janedoe
```

### 3.5.5. 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
```

### 3.5.6. 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.

### 3.5.7. 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.

### 3.5.8. 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.

### 3.5.9. 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": "QFAMEDSJP5DEKJ00DDXY",
      "secret_key": "iaSFLDVvDdQt6lkNzHyW4fPLZugBAI1g17L00+87"}],
  "swift_keys": [
    { "user": "johndoe:swift",
      "secret_key": "E9T2rUZNu2gxUjcwUB08n\Ev4KX6\GprEuH4qhu1"}]]}
```

### 3.5.10. 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, specify the user.:

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

### 3.5.11. 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}
```

## 3.6. 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, “Bucket Sharding”](#) for details.

### 3.6.1. 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:

```
[root@master-zone]# 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.

### 3.6.2. 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>
```

### 3.6.3. Set Bucket Quotas

Bucket quotas apply to the buckets owned by the specified `uid`. They are independent of the user. :

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

A negative value for num objects and / or max size means that the specific quota attribute check is disabled.

### 3.6.4. 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>
```

### 3.6.5. 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:

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

### 3.6.6. 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
```

### 3.6.7. Get User Quota Usage Stats

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

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



#### NOTE

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

### 3.6.8. 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, Object Gateway Configuration Reference](#) for more details on these options.

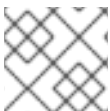
### 3.6.9. Reading and Writing Global Quotas

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

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

To set quota settings for the entire region, modify the quota settings in the region map. Then, use the `regionmap set` command to update the region map:

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



#### NOTE

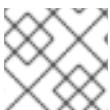
After updating the region map, you must restart the gateway.

## 3.7. 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 may specify time with minutes and seconds, but it is stored with 1 hour resolution.

### 3.7.1. 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
```

### 3.7.2. 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
```

### 3.7.3. Finding Orphan Objects

Normally, in a healthy storage cluster you should not have any leaking objects, but in some cases leaky objects can occur. For example, if the RADOS Gateway goes down in the middle of an operation, this may cause some RADOS objects to become orphans. Also, unknown bugs may cause these orphan objects to occur. The **radosgw-admin** command provides you a tool to search for these orphan objects and clean them up. With the **--pool** option, you can specify which pool to scan for leaky RADOS objects. With the **--num-shards** option, you may specify the number of shards to use for keeping temporary scan data.

1. Create a new log pool:

#### Example

```
rados mkpool .log
```

2. Search for orphan objects:

#### Syntax

```
[root@master-zone]# radosgw-admin orphans find --pool=<data_pool> --job-id=<job_name> [--num-shards=<num_shards>] [--orphan-stale-secs=<seconds>]
```

#### Example

```
[root@master-zone]# radosgw-admin orphans find --pool=.rgw.buckets --job-id=abc123
```



### 3. Clean up the search data:

#### **Syntax**

```
[root@master-zone]# radosgw-admin orphans finish --job-id=<job_name>
```

#### **Example**

```
[root@master-zone]# radosgw-admin orphans finish --job-id=abc123
```

## CHAPTER 4. OBJECT GATEWAY 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`.

Name	Description	Type	Default
<code>rgw_data</code>	Sets the location of the data files for Ceph Object Gateway.	String	<code>/var/lib/ceph/radosgw/\$cluster-\$id</code>
<code>rgw_enable_apis</code>	Enables the specified APIs.	String	<code>s3, swift, swift_auth, admin</code> All APIs.
<code>rgw_cache_enabled</code>	Whether the Ceph Object Gateway cache is enabled.	Boolean	<code>true</code>
<code>rgw_cache_lru_size</code>	The number of entries in the Ceph Object Gateway cache.	Integer	<code>10000</code>
<code>rgw_socket_path</code>	The socket path for the domain socket. <b>FastCGIExternalServer</b> 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 <code>rgw.conf</code> file.	String	N/A
<code>rgw_host</code>	The host for the Ceph Object Gateway instance. Can be an IP address or a hostname.	String	<code>0.0.0.0</code>
<code>rgw_port</code>	Port the instance listens for requests. If not specified, Ceph Object Gateway runs external FastCGI.	String	None
<code>rgw_dns_name</code>	The DNS name of the served domain. See also the <code>hostnames</code> setting within zone groups.	String	None
<code>rgw_script_uri</code>	The alternative value for the <code>SCRIPT_URI</code> if not set in the request.	String	None

Name	Description	Type	Default
<code>rgw_request_uri</code>	The alternative value for the <b>REQUEST_URI</b> if not set in the request.	String	None
<code>rgw_print_continue</code>	Enable <b>100-continue</b> if it is operational.	Boolean	<b>true</b>
<code>rgw_remote_addr_param</code>	The remote address parameter. For example, the HTTP field containing the remote address, or the <b>X-Forwarded-For</b> address if a reverse proxy is operational.	String	<b>REMOTE_ADDR</b>
<code>rgw_op_thread_timeout</code>	The timeout in seconds for open threads.	Integer	600
<code>rgw_op_thread_suicide_timeout</code>	The time <b>timeout</b> in seconds before a Ceph Object Gateway process dies. Disabled if set to <b>0</b> .	Integer	<b>0</b>
<code>rgw_thread_pool_size</code>	The size of the thread pool.	Integer	100 threads.
<code>rgw_num_control_oids</code>	The number of notification objects used for cache synchronization between different <b>rgw</b> instances.	Integer	<b>8</b>
<code>rgw_init_timeout</code>	The number of seconds before Ceph Object Gateway gives up on initialization.	Integer	<b>30</b>
<code>rgw_mime_types_file</code>	The path and location of the MIME types. Used for Swift auto-detection of object types.	String	<b>/etc/mime.types</b>
<code>rgw_gc_max_objs</code>	The maximum number of objects that may be handled by garbage collection in one garbage collection processing cycle.	Integer	<b>32</b>
<code>rgw_gc_obj_min_wait</code>	The minimum wait time before the object may be removed and handled by garbage collection processing.	Integer	<b>2 * 3600</b>
<code>rgw_gc_processor_max_time</code>	The maximum time between the beginning of two consecutive garbage collection processing cycles.	Integer	<b>3600</b>
<code>rgw_gc_processor_period</code>	The cycle time for garbage collection processing.	Integer	<b>3600</b>

Name	Description	Type	Default
<b>rgw_s3_success_create_obj_status</b>	The alternate success status response for <b>create-obj</b> .	Integer	<b>0</b>
<b>rgw_resolve_cname</b>	Whether <b>rgw</b> should use DNS CNAME record of the request hostname field (if hostname is not equal to <b>rgw_dns_name</b> ).	Boolean	<b>false</b>
<b>rgw_object_stripe_size</b>	The size of an object stripe for Ceph Object Gateway objects.	Integer	<b>4 &lt;= 20</b>
<b>rgw_extended_http_attrs</b>	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"
<b>rgw_exit_timeout_secs</b>	Number of seconds to wait for a process before exiting unconditionally.	Integer	<b>120</b>
<b>rgw_get_obj_window_size</b>	The window size in bytes for a single object request.	Integer	<b>16 &lt;= 20</b>
<b>rgw_get_obj_max_req_size</b>	The maximum request size of a single get operation sent to the Ceph Storage Cluster.	Integer	<b>4 &lt;= 20</b>
<b>rgw_relaxed_s3_bucket_names</b>	Enables relaxed S3 bucket names rules for zone group buckets.	Boolean	<b>false</b>
<b>rgw_list_buckets_max_chunk</b>	The maximum number of buckets to retrieve in a single operation when listing user buckets.	Integer	<b>1000</b>
<b>rgw_override_bucket_index_max_shards</b>	<p>The number of shards for the bucket index object. A value of <b>0</b> indicates there is no sharding. Red Hat does not recommend to set a value too large (for example, <b>1000</b>) as it increases the cost for bucket listing.</p> <p>This variable should be set in the <b>[client]</b> or the <b>[global]</b> section so it is automatically applied to <b>radosgw-admin</b> commands.</p>	Integer	<b>0</b>
<b>rgw_num_zone_opstate_shards</b>	The maximum number of shards for keeping inter-zonegroup copy progress information.	Integer	<b>128</b>
<b>rgw_opstate_rate_limit_sec</b>	The minimum time between opstate updates on a single upload. <b>0</b> disables the ratelimit.	Integer	<b>30</b>

Name	Description	Type	Default
<code>rgw_curl_wait_timeout_ms</code>	The timeout in milliseconds for certain <code>curl</code> calls.	Integer	<b>1000</b>
<code>rgw_copy_obj_progress</code>	Enables output of object progress during long copy operations.	Boolean	<b>true</b>
<code>rgw_copy_obj_progress_every_bytes</code>	The minimum bytes between copy progress output.	Integer	<b>1024 * 1024</b>
<code>rgw_admin_entry</code>	The entry point for an admin request URL.	String	<b>admin</b>
<code>rgw_content_length_compat</code>	Enable compatibility handling of FCGI requests with both <code>CONTENT_LENGTH</code> AND <code>HTTP_CONTENT_LENGTH</code> set.	Boolean	<b>false</b>
<code>rgw_bucket_default_quota_max_objects</code>	<p>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.</p> <p>This variable should be set in the <code>[client]</code> or the <code>[global]</code> section so it is automatically applied to <code>radosgw-admin</code> commands.</p>	Integer	<b>-1</b>
<code>rgw_bucket_quota_ttl</code>	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	<b>600</b>
<code>rgw_user_quota_bucket_sync_interval</code>	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	<b>180</b>
<code>rgw_user_quota_sync_interval</code>	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	<b>3600 * 24</b>

## 4.1. 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.

### 4.1.1. Create 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.

### 4.1.2. Make 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.

### 4.1.3. Delete 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
```

### 4.1.4. Get 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.

#### 4.1.5. Set 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
```

#### 4.1.6. List Realms

To list realms, execute **realm list**.

```
# radosgw-admin realm list
```

#### 4.1.7. List Realm Periods

To list realm periods, execute **realm list-periods**.

```
# radosgw-admin realm list-periods
```

#### 4.1.8. Pull 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}
```

### 4.1.9. Rename 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.

## 4.2. 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 in Red Hat Ceph Storage 1.3, 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 **MAY** be performed on any host 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.

### 4.2.1. Create 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

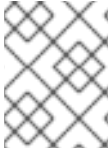
Use `zonegroup modify --rgw-zonegroup=<zonegroup-name>` to modify an existing zone group's settings.

### 4.2.2. Make 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
```

### 4.2.3. Add a Zone to a Zone Group

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
```

### 4.2.4. Remove 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
```

### 4.2.5. Rename 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
```

### 4.2.6. Delete 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
```

### 4.2.7. List 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"
  ]
}
```

### 4.2.8. Get a Zone Group Map

To list the details of each zone group, execute:

```
# radosgw-admin zonegroup-map get
```



#### NOTE

If you receive a **failed to read zonegroup map** error, run `radosgw-admin zonegroup-map update` as root first.

### 4.2.9. Get 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": 0,
    "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": 0,
    "read_only": "false"
  }
],
"placement_targets": [
  {
    "name": "default-placement",
    "tags": []
  }
],
"default_placement": "default-placement",
"realm_id": "ae031368-8715-4e27-9a99-0c9468852cfe"
}

```

#### 4.2.10. Set 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
```

## 4.2.11. Set 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": 0,
        "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": 0,
        "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
```

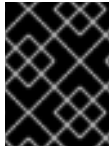
Where **zonegroupmap.json** is the JSON file you created. Ensure that you have zones created for the ones specified in the zone group map. Finally, update the period.

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

## 4.3. 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.

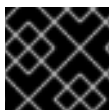


### IMPORTANT

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

### 4.3.1. Create 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

This operation **MUST** be performed on a node that will be in the zone.

```
[root@new-zone]# radosgw-admin zone create --rgw-zone=<name> \
    [--zonegroup=<zonegroup-name>] \
    [--endpoints=<endpoint>[,<endpoint>] \
    [--master] [--default] \
    --access-key $SYSTEM_ACCESS_KEY --secret
$SYSTEM_SECRET_KEY
```

Then, update the period:

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

### 4.3.2. Delete a Zone

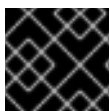
To delete zone, first remove it from the zonegroup.

```
# 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 operation **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:

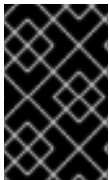
```
# radosgw-admin period update --commit
```



### IMPORTANT

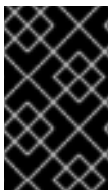
Do not delete a zone without removing it from a zone group first. Otherwise, updating the period will fail.

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.



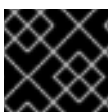
### IMPORTANT

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.

```
# rados rmppool <del-zone>.rgw.control <del-zone>.rgw.control --yes-i-  
really-really-mean-it  
# rados rmppool <del-zone>.rgw.data.root <del-zone>.rgw.data.root --yes-i-  
really-really-mean-it  
# rados rmppool <del-zone>.rgw.gc <del-zone>.rgw.gc --yes-i-really-really-  
mean-it  
# rados rmppool <del-zone>.rgw.log <del-zone>.rgw.log --yes-i-really-  
really-mean-it  
# rados rmppool <del-zone>.rgw.users.uid <del-zone>.rgw.users.uid --yes-i-  
really-really-mean-it
```

### 4.3.3. Modify a Zone

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



### IMPORTANT

This operation **MUST** be performed on a host 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
```

#### 4.3.4. List Zones

As **root**, to list the zones in a cluster, execute:

```
# radosgw-admin zone list
```

#### 4.3.5. Get 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"}
    }
  ]
}
```

#### 4.3.6. Set 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.

To set a zone, create a JSON object consisting of the pools, save the object to a file; then, execute the following command **on a node within the zone**, 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
```

### 4.3.7. Rename a Zone

To rename a zone, specify the zone name and the new zone name.



#### IMPORTANT

This operation **MUST** be performed on a node 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
```

## 4.4. ZONE GROUP AND ZONE 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
<code>rgw_zone</code>	The name of the zone for the gateway instance.	String	None
<code>rgw_zonegroup</code>	The name of the zone group for the gateway instance.	String	None
<code>rgw_zonegroup_root_pool</code>	The root pool for the zone group.	String	<code>.rgw.root</code>
<code>rgw_zone_root_pool</code>	The root pool for the zone.	String	<code>.rgw.root</code>
<code>rgw_default_zone_group_info_oid</code>	The OID for storing the default zone group. We do not recommend changing this setting.	String	<code>default.zone group</code>
<code>rgw_num_zone_operation_shards</code>	The maximum number of shards for keeping inter-zone group synchronization progress.	Integer	<b>128</b>

## 4.5. POOLS

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

## Manually Created Pools vs. Generated Pools

If you provide write capabilities to the user key for your Ceph Object Gateway, the gateway has the ability to create pools automatically. This is convenient, but the Ceph Object Storage Cluster uses the default values for the number of placement groups, which may not be ideal or the values you specified in your Ceph configuration file. If you allow the Ceph Object Gateway to create pools automatically, ensure that you have reasonable defaults for the number of placement groups.

The default pools for the Ceph Object Gateway's default zone include:

- `.rgw.root`
- `.default.rgw.control`
- `.default.rgw.gc`
- `.default.log`
- `.default.intent-log`
- `.default.usage`
- `.default.users`
- `.default.users.email`
- `.default.users.swift`
- `.default.users.uid`

The Ceph Object Gateway creates pools on a per zone basis. If you create the pools manually, prepend the zone name. For example:

- `.zone-name.domain.rgw`
- `.zone-name.rgw.control`
- `.zone-name.rgw.gc`
- `.zone-name.log`
- `.zone-name.intent-log`
- `.zone-name.usage`
- `.zone-name.users`
- `.zone-name.users.email`
- `.zone-name.users.swift`
- `.zone-name.users.uid`

Ceph Object Gateways store data for the bucket index (`index_pool`) and bucket data (`data_pool`) in placement pools. These may overlap—i.e., 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
<code>rgw_zonegroup_root_pool</code>	The pool for storing all zone group-specific information.	String	<code>.rgw.root</code>
<code>rgw_zone_root_pool</code>	The pool for storing zone-specific information.	String	<code>.rgw.root</code>

## 4.6. SWIFT SETTINGS

Name	Description	Type	Default
<code>rgw_enforce_swift_acls</code>	Enforces the Swift Access Control List (ACL) settings.	Boolean	<code>true</code>
<code>rgw_swift_token_expiration</code>	The time in seconds for expiring a Swift token.	Integer	<code>24 * 3600</code>
<code>rgw_swift_url</code>	The URL for the Ceph Object Gateway Swift API.	String	None
<code>rgw_swift_url_prefix</code>	The URL prefix for the Swift API (e.g., <a href="http://fqdn.com/swift">http://fqdn.com/swift</a> ).	<code>swift</code>	N/A
<code>rgw_swift_auth_url</code>	Default URL for verifying v1 auth tokens (if not using internal Swift auth).	String	None
<code>rgw_swift_auth_entry</code>	The entry point for a Swift auth URL.	String	<code>auth</code>

## 4.7. LOGGING SETTINGS

Name	Description	Type	Default
<code>rgw_log_nonexistent_bucket</code>	Enables Ceph Object Gateway to log a request for a non-existent bucket.	Boolean	<code>false</code>
<code>rgw_log_object_name</code>	The logging format for an object name. See manpage date for details about format specifiers.	Date	<code>%Y-%m-%d-%H-%i-%n</code>
<code>rgw_log_object_name_utc</code>	Whether a logged object name includes a UTC time. If <code>false</code> , it uses the local time.	Boolean	<code>false</code>

Name	Description	Type	Default
<code>rgw_usage_max_shards</code>	The maximum number of shards for usage logging.	Integer	<b>32</b>
<code>rgw_usage_max_user_shards</code>	The maximum number of shards used for a single user's usage logging.	Integer	<b>1</b>
<code>rgw_enable_ops_log</code>	Enable logging for each successful Ceph Object Gateway operation.	Boolean	<b>false</b>
<code>rgw_enable_usage_log</code>	Enable the usage log.	Boolean	<b>false</b>
<code>rgw_ops_log_rados</code>	Whether the operations log should be written to the Ceph Storage Cluster backend.	Boolean	<b>true</b>
<code>rgw_ops_log_socket_path</code>	The Unix domain socket for writing operations logs.	String	None
<code>rgw_ops_log_data-backlog</code>	The maximum data backlog data size for operations logs written to a Unix domain socket.	Integer	<b>5 &lt;= 20</b>
<code>rgw_usage_log_flush_threshold</code>	The number of dirty merged entries in the usage log before flushing synchronously.	Integer	1024
<code>rgw_usage_log_tick_interval</code>	Flush pending usage log data every <b>n</b> seconds.	Integer	<b>30</b>
<code>rgw_intent_log_object_name</code>	The logging format for the intent log object name. See manpage date for details about format specifiers.	Date	<b>%Y-%m-%d-%i-%n</b>
<code>rgw_intent_log_object_name_utc</code>	Whether the intent log object name includes a UTC time. If <b>false</b> , it uses the local time.	Boolean	<b>false</b>
<code>rgw_data_log_window</code>	The data log entries window in seconds.	Integer	<b>30</b>
<code>rgw_data_log_changes_size</code>	The number of in-memory entries to hold for the data changes log.	Integer	<b>1000</b>
<code>rgw_data_log_num_shards</code>	The number of shards (objects) on which to keep the data changes log.	Integer	<b>128</b>
<code>rgw_data_log_object_prefix</code>	The object name prefix for the data log.	String	<b>data_log</b>

Name	Description	Type	Default
<b>rgw_replica_log_obj_prefix</b>	The object name prefix for the replica log.	String	<b>replica log</b>
<b>rgw_md_log_max_shards</b>	The maximum number of shards for the metadata log.	Integer	<b>64</b>

## 4.8. KEYSTONE SETTINGS

Name	Description	Type	Default
<b>rgw_keystone_url</b>	The URL for the Keystone server.	String	None
<b>rgw_keystone_admin_token</b>	The Keystone admin token (shared secret).	String	None
<b>rgw_keystone_accepted_roles</b>	The roles requires to serve requests.	String	<b>Member, admin</b>
<b>rgw_keystone_token_cache_size</b>	The maximum number of entries in each Keystone token cache.	Integer	<b>10000</b>
<b>rgw_keystone_revocation_interval</b>	The number of seconds between token revocation checks.	Integer	<b>15 * 60</b>

## 4.9. LDAP SETTINGS

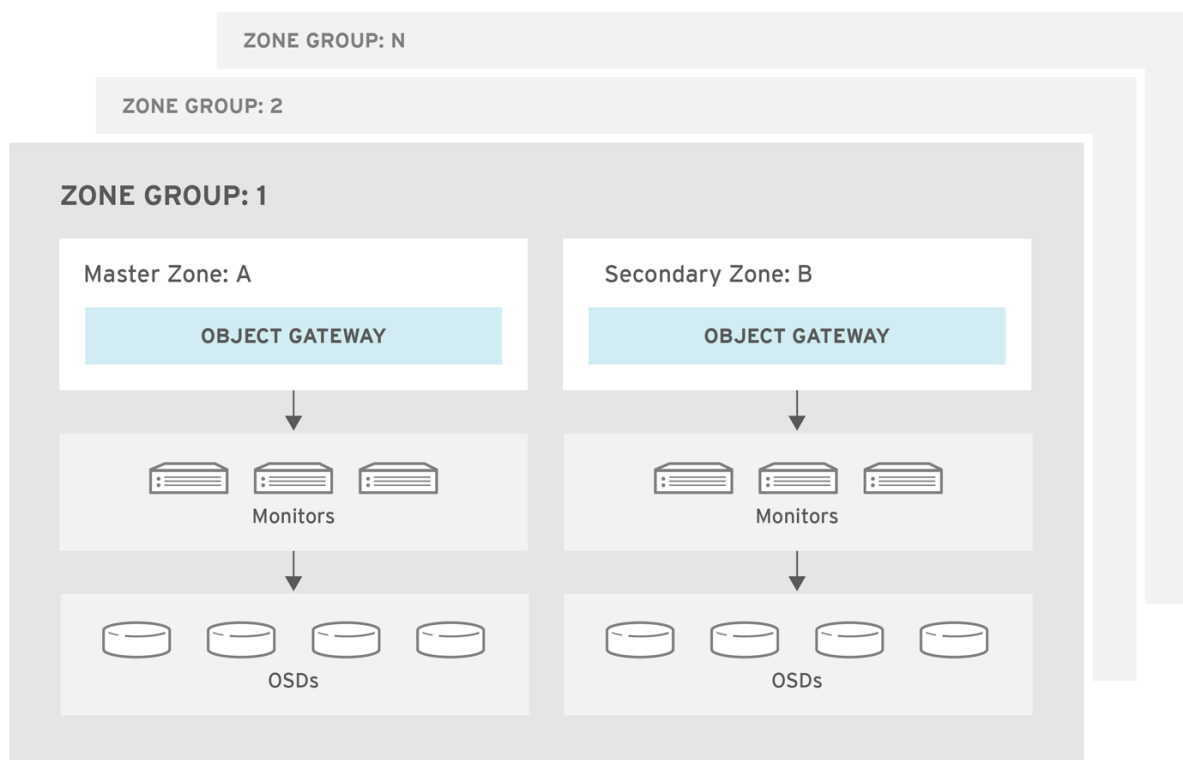
Name	Description	Type	Example
<b>rgw_ldap_uri</b>	A space-separated list of LDAP servers in URI format.	String	<b>ldaps://&lt;ldap.your.domain&gt;</b>
<b>rgw_ldap_searchdn</b>	The LDAP search domain name, also known as base domain.	String	<b>cn=users,cn=accounts,dc=example,dc=com</b>
<b>rgw_ldap_binddn</b>	The gateway will bind with this LDAP entry (user match).	String	<b>uid=admin,cn=users,dc=example,dc=com</b>
<b>rgw_ldap_secret</b>	A file containing credentials for <b>rgw_ldap_binddn</b>	String	<b>/etc/openldap/secret</b>

Name	Description	Type	Example
<b>rgw_ldap_dnattr</b>	LDAP attribute containing Ceph object gateway user names (to form binddns).	String	<b>uid</b>

## CHAPTER 5. MULTI-SITE

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. In Red Hat Ceph Storage 2, each zone is active and may receive write operations. In addition to disaster recovery, multiple active zones may also serve as a foundation for content delivery networks.
- **Multi-zone-group:** Formerly called 'regions', 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:** In Red Hat Ceph Storage 2, 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.



**REALM: A**

CEPH\_405148\_0616

### 5.1. FUNCTIONAL CHANGES FROM RED HAT CEPH STORAGE 1.X

In Red Hat Ceph Storage 2, you can configure each Ceph Object Gateway to work in an active-active zone configuration, allowing for writes to non-master zones.

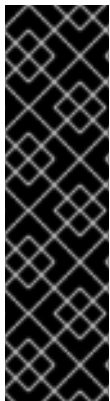
The multi-site configuration is stored within a container called a "realm." The realm stores zone groups, zones, and a time "period" with multiple epochs for tracking changes to the configuration. In Red Hat Ceph Storage 2, the `ceph-radosgw` daemons handle the synchronization, eliminating the need for a separate synchronization agent. Additionally, the new approach to synchronization allows the Ceph Object Gateway to operate with an "active-active" configuration instead of "active-passive".

## 5.2. REQUIREMENTS AND ASSUMPTIONS

A multi-site configuration requires at least two Ceph storage clusters, preferably given a distinct cluster name. 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 synchronized, leading to fragmented metadata.

In this guide, 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.

## 5.3. 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](#) 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.data.root`
- `us-east.rgw.gc`
- `us-east.rgw.log`
- `us-east.rgw.intent-log`
- `us-east.rgw.usage`
- `us-east.rgw.users.keys`
- `us-east.rgw.users.email`
- `us-east.rgw.users.swift`
- `us-east.rgw.users.uid`
- `us-east.rgw.buckets.index`
- `us-east.rgw.buckets.data`

## 5.4. INSTALLING AN OBJECT GATEWAY

To install the Ceph Object Gateway, see the [Red Hat Ceph Storage Installation Guide for Red Hat Enterprise Linux](#).

All Ceph Object Gateway nodes must follow the tasks listed in the Prerequisites section. Red Hat supports two methods for installing Ceph Object Gateway

- by using the command line
- by using the Ansible automation application

### 5.4.1. Installing using the Command Line

The procedure for installing Ceph Object Gateway is identical to the procedure described in the [Red Hat Ceph Storage Installation Guide for Red Hat Enterprise Linux](#) but with this difference: proceed to configure the zones BEFORE starting the Ceph Object Gateway daemons to avoid the need to remove default zones and pools.

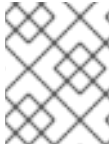
### 5.4.2. Installing an Object Gateway using Ansible

Ansible can install and configure Ceph Object Gateways for use with a Ceph Storage cluster. So for a simple configuration, add Ceph Object Gateways to your Ansible configuration. 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.

### 5.4.3. Updating an Object Gateway Installation

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.

## 5.5. CONFIGURING A MASTER ZONE

All gateways in a multi-site configuration will retrieve their configuration from a `ceph-radosgw` daemon on a host within the master zone group and master zone. To configure your gateways in a multi-site configuration, choose a `ceph-radosgw` instance to configure the master zone group and master zone.

### 5.5.1. 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.

### 5.5.2. 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"
  }

```

### 5.5.3. Create a Master Zone



#### IMPORTANT

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

Create a new 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}[,{http://fqdn}]

```

For example:

```

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

```



#### 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.



#### IMPORTANT

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

### 5.5.4. Delete Default Zone Group and Zone

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

```

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

```

Finally, delete the `default` pools in your Ceph storage cluster if they exist.



## IMPORTANT

The following step assumes a multi-site configuration using newly installed systems that aren't currently storing data. **DO NOT DELETE** the **default** zone group if you are already using it to store data.

```
# rados rmpool default.rgw.control default.rgw.control --yes-i-really-
really-mean-it
# rados rmpool default.rgw.data.root default.rgw.data.root --yes-i-really-
really-mean-it
# rados rmpool default.rgw.gc default.rgw.gc --yes-i-really-really-mean-it
# rados rmpool default.rgw.log default.rgw.log --yes-i-really-really-mean-
it
# rados rmpool default.rgw.users.uid default.rgw.users.uid --yes-i-really-
really-mean-it
```

### 5.5.5. 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
```

### 5.5.6. 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.

### 5.5.7. 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]
host = rgw1
rgw frontends = "civetweb port=80"
rgw_zone=us-east
```

### 5.5.8. Start the Gateway

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

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

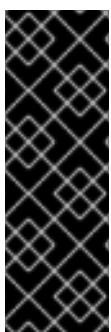
## 5.6. CONFIGURE SECONDARY ZONES

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 zone operations on a host identified to serve the secondary zone.



### NOTE

To add a third zone, follow the same procedures as for adding the secondary zone. Use 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 zone group. The master zone and the secondary zone can receive bucket operations executed via the RESTful API, and the secondary zone redirects bucket operations to the master zone. If the master zone is down, bucket operations will fail. If you create buckets using the `radosgw-admin` CLI, you **MUST** do so on a host within the master zone of the master zone group, or the buckets **WILL NOT** synchronize with secondary zones and zone groups.

### 5.6.1. 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}
```

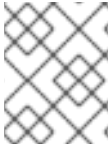
If this realm is the default realm or the only realm, make the realm the default realm.

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

### 5.6.2. 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.

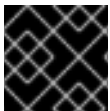
```
# radosgw-admin period pull --url={url-to-master-zone-gateway} --access-key={access-key} --secret={secret}
```



#### NOTE

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

### 5.6.3. 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. In Red Hat Ceph Storage 2, 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:

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

For 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.

```
# rados rmpool default.rgw.control default.rgw.control --yes-i-really-  
really-mean-it  
# rados rmpool default.rgw.data.root default.rgw.data.root --yes-i-really-  
really-mean-it  
# rados rmpool default.rgw.gc default.rgw.gc --yes-i-really-really-mean-it  
# rados rmpool default.rgw.log default.rgw.log --yes-i-really-really-mean-  
it  
# rados rmpool default.rgw.users.uid default.rgw.users.uid --yes-i-really-  
really-mean-it
```

#### 5.6.4. 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]  
host = rgw2  
rgw frontends = "civetweb port=80"  
rgw_zone=us-west
```

#### 5.6.5. 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.

#### 5.6.6. Start the Gateway

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

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



### 5.6.7. Check Synchronization Status

Once the secondary zone is up and running, check the synchronization status. Synchronization copies users and buckets created in the master zone to the secondary zone.

```
# radosgw-admin sync status
```

The output will provide the status of synchronization operations. For example:

```
realm f3239bc5-e1a8-4206-a81d-e1576480804d (earth)
  zonegroup c50dbb7e-d9ce-47cc-a8bb-97d9b399d388 (us)
    zone 4c453b70-4a16-4ce8-8185-1893b05d346e (us-west)
metadata sync syncing
  full sync: 0/64 shards
  metadata is caught up with master
  incremental sync: 64/64 shards
data sync source: 1ee9da3e-114d-4ae3-a8a4-056e8a17f532 (us-east)
  syncing
  full sync: 0/128 shards
  incremental sync: 128/128 shards
  data is caught up with source
```



#### NOTE

Secondary zones accept bucket operations; however, secondary zones redirect bucket operations to the master zone and then synchronize with the master zone to receive the result of the bucket operations. If the master zone is down, bucket operations executed on the secondary zone will fail, but object operations should succeed.

## 5.7. FAILOVER AND DISASTER RECOVERY

If the master zone should 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 will run 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
```

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

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

3. Finally, restart the Ceph Object Gateway.

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

If the former master zone recovers, revert the operation.

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

```
# radosgw-admin period 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. Then, restart the Ceph Object Gateway in the recovered zone.

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

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. Finally, restart the Ceph Object Gateway in the secondary zone.

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

## 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.

```
# radosgw-admin realm create --rgw-realm=<name> --default
```

2. Rename the default zone and zonegroup. Replace `<name>` with the zonegroup or zone name.

```
# radosgw-admin zonegroup rename --rgw-zonegroup default --
zonegroup-new-name=<name>
# 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.

■

```
# 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.

```
# 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.

```
# radosgw-admin user create --uid=<user-id> --display-name="
<display-name>"\
                                --access-key=<access-key> --secret=
<secret-key> --system
```

6. Commit the updated configuration.

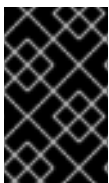
```
# radosgw-admin period update --commit
```

7. Finally, restart the Ceph Object Gateway.

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

After completing this procedure, proceed to [Configure a Secondary Zone](#) to create a secondary zone in the master zone group.

## 5.9. MIGRATING A 1.3 MULTI-SITE SYSTEM TO VERSION 2



### IMPORTANT

Customers wishing to migrate from a 1.3 multi-site system, formerly called "federated", contact customer support for advice on recommended steps so Red Hat Support can look at the configuration, environment, and data first.