Red Hat Ceph Storage 1.3 Ceph Object Gateway for RHEL x86 64

Installing, configuring and administering the Ceph Storage Object Gateway on RHEL x86_64.

Red Hat Customer Content Services

Red Hat Ceph Storage 1.3 Ceph Object Gateway for RHEL x86 64

Installing, configuring and administering the Ceph Storage Object Gateway on RHEL x86_64.

Legal Notice

Abstract

This document provides instructions for installing, configuring and administering a Ceph Storage Object Gateway on RHEL 7.

Table of Contents

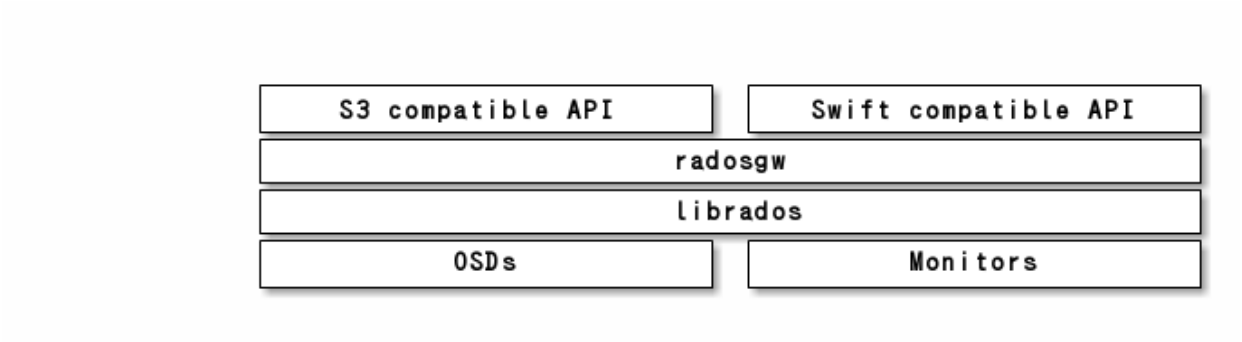
PREFACE

Designed for cloud infrastructures and web-scale object storage, Red Hat® Ceph Storage is a massively scalable, open, software-defined storage platform that combines the most stable version of Ceph with a Ceph management platform, deployment tools, and support services. Providing the tools to flexibly and cost-effectively manage petabyte-scale data deployments in the enterprise, Red Hat Ceph Storage manages cloud data so enterprises can focus on managing their businesses.

Ceph Object Gateway is an object storage interface built on top of **`librados`** to provide applications with a RESTful gateway to Ceph Storage Clusters. Ceph Object Storage 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.

Ceph Object Storage uses the Ceph Object Gateway daemon (**`radosgw`**), which 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, you will use separate pools. The S3 and Swift APIs share a common namespace, so you may write data with one API and retrieve it with the other.



PART I. INSTALLATION

For Red Hat Ceph Storage v1.3, Red Hat supports the Ceph Object Gateway running on Civetweb (embedded into the **`ceph-radosgw`** daemon) instead of Apache and FastCGI. Using Civetweb simplifies the Ceph Object Gateway installation and configuration.

Note

To run the Ceph Object Gateway service, you should have a running Ceph storage cluster, and the gateway host should have access to the public network.

Note

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

CHAPTER 1. EXECUTE THE PRE-INSTALLATION PROCEDURE

Refer to the Red Hat Ceph Storage Installation Guide, and execute the pre-installation procedures on your Ceph Object Gateway node. Specifically, you should disable **`requiretty`** on your Ceph Deplooy user, set SELinux to **`Permissive`** and set up a Ceph Deploy user with password-less **`sudo`**. For Ceph Object Gateways, you will need to open the port that Civetweb will use in production.

Note

Civetweb runs on port **`7480`** by default.

CHAPTER 2. ENABLE CEPH CLIENT REPOSITORY

Red Hat packages the Ceph Object Gateway in the **rhel-7-server-rhceph-1.3-tools-rpms** repository. To ensure you are using the same version of Ceph as your storage cluster, execute the following to enable the repository:

```
sudo subscription-manager repos --enable=rhel-7-server-rhceph-1.3-
tools-rpms
```

CHAPTER 3. INSTALL CEPH OBJECT GATEWAY

From the working directory of your administration server, install the Ceph Object Gateway package on the Ceph Object Gateway node. For example:

```
ceph-deploy install --rgw <gateway-node1> [<gateway-node2> ...]
```

The **ceph-common** package is a dependency, so **ceph-deploy** will install this too. The **ceph** CLI tools are intended for administrators. To make your Ceph Object Gateway node an administrator node, execute the following from the working directory of your administration server.

```
ceph-deploy admin <node-name>
```

CHAPTER 4. CREATE A GATEWAY INSTANCE

From the working directory of your administration server, create an instance of the Ceph Object Gateway on the Ceph Object Gateway. For example:

```
ceph-deploy rgw create <gateway-node1>
```

Once the gateway is running, you should be able to access it on port **7480** with an unauthenticated request like this:

```
http://client-node:7480
```

If the gateway instance is working properly, you should receive a response like this:

```
<?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>
```

If at any point you run into trouble and you want to start over, execute the following to purge the configuration:

```
ceph-deploy purge <gateway-node1> [<gateway-node2>]
ceph-deploy purgedata <gateway-node1> [<gateway-node2>]
```

If you execute **purge**, you must re-install Ceph.

CHAPTER 5. CHANGE THE 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 the working 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:

```
[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**.

Push the updated configuration file to your Ceph Object Gateway node (and other Ceph nodes).

```
ceph-deploy --overwrite-conf config push <gateway-node> [<other-nodes>]
```

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

```
sudo 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:

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

CHAPTER 6. 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 back to the description for installing a Ceph Object Gateway with **`ceph-deploy`**, notice that the configuration file only has one setting **`rgw_frontends`** (and that's assuming you elected to change the default port). The **`ceph-deploy`** utility generates the data directory and the keyring for you—placing the keyring in **`/var/lib/ceph/radosgw/{rgw-intance}`**. The daemon looks in default locations, whereas you may have specified different settings in your Ceph configuration file. 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.radosgw.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.radosgw.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, restart the Ceph Object Gateway.

```
sudo 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.

CHAPTER 7. CONFIGURE BUCKET SHARDING

A Ceph Object Gateway stores bucket index data in the **index_pool**, which defaults to **.rgw.buckets.index**. Sometimes users like to put many objects (hundreds of thousands to millions of objects) in a single bucket. If you do not use the gateway administration interface to set quotas for the maximum number of objects per bucket, the bucket index can suffer significant performance degradation when users place large numbers of objects into a bucket.

In Ceph 1.3, you may shard bucket indices to help prevent performance bottlenecks when you allow a high number of objects per bucket. The **rgw_override_bucket_index_max_shards** setting allows you to set a maximum number of shards per bucket. The default value is **0**, which means bucket index sharding is off by default.

To turn bucket index sharding on, set **rgw_override_bucket_index_max_shards** to a value greater than **0**.

For simple configurations, you may add **rgw_override_bucket_index_max_shards** to your Ceph configuration file. Add it under **[global]** to create a system-wide value. You can also set it for each instance in your Ceph configuration file.

Once you have changed your bucket sharding configuration in your Ceph configuration file, restart your gateway.

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

For federated configurations, each zone may have a different **index_pool** setting for failover. To make the value consistent for a region's zones, you may set **rgw_override_bucket_index_max_shards** in a gateway's region configuration. For example:

```
radosgw-admin region get > region.json
```

Open the **region.json** file and edit the **bucket_index_max_shards** setting for each named zone. Save the **region.json** file and reset the region. For example:

```
radosgw-admin region set < region.json
```

Once you've updated your region, update the region map. For example:

```
radosgw-admin regionmap update --name client.rgw.ceph-client
```

Where **client.rgw.ceph-client** is the name of the gateway user.

Note

Mapping the index pool (for each zone, if applicable) to a CRUSH ruleset of SSD-based OSDs may also help with bucket index performance.

CHAPTER 8. ADD WILDCARD TO DNS

To use Ceph with S3-style subdomains (e.g., bucket-name.domain-name.com), you need to add a wildcard to the DNS record of the DNS server you use with the **ceph-radosgw** daemon.

The address of the DNS must also be specified in the Ceph configuration file with the **rgw dns name = {hostname}** setting.

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 your **ceph-radosgw** daemon can process the subdomain requests:

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

For example:

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

CHAPTER 9. ADD DEBUGGING (IF NEEDED)

Once you finish the setup procedure, if you encounter issues with your configuration, you can add debugging to 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
```

CHAPTER 10. USING THE GATEWAY

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

10.1. CREATE 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,
 "auid": 0,
 "subusers": [],
 "keys": [{
  "user": "testuser",
  "access_key": "I0PJDPCIYZ665MW88W9R",
  "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.

10.2. CREATE 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": "I0PJDPCIYZ665MW88W9R",
  "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,
 "auid": 0,
 "subusers": [{
  "id": "testuser:swift",
  "permissions": "full-control"
 }],
 "keys": [{
  "user": "testuser:swift",
  "access_key": "3Y1LNW4Q6X0Y53A52DET",
  "secret_key": ""
 }, {
  "user": "testuser",
  "access_key": "I0PJDPCIYZ665MW88W9R",
  "secret_key": "dxaXZ8U90SXydYzyS5ivamEP20hkLSUViiaR+ZDA"
 }],
 "swift_keys": [{
  "user": "testuser:swift",
  "secret_key": "244+fz2gSqoHwR3lYtSbIyomyPHf3i7rgSJrF\/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": []
 }
```

10.3. ACCESS VERIFICATION

10.3.1. Test S3 access

You need to write and run a Python test script for verifying S3 access. The S3 access test script will connect to the **radosgw**, create a new bucket and list all buckets. The values for **aws_access_key_id** and **aws_secret_access_key** are taken from the values of **access_key** and **secret_key** returned by the **radosgw_admin** command.

Execute the following steps:

1. You will need to install the **python-boto** package.

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

2. Create the Python script:

   ```
   vi s3test.py
   ```

3. Add the following contents to the file:

   ```
   import boto
   import boto.s3.connection

   access_key = 'I0PJDPCIYZ665MW88W9R'
   secret_key = 'dxaXZ8U90SXydYzyS5ivamEP20hkLSUViiaR+ZDA'

   conn = boto.connect_s3(
    aws_access_key_id = access_key,
    aws_secret_access_key = secret_key,
    host = '{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 **{hostname}** with the hostname of the host where you have configured the gateway service i.e, the **gateway host**. Replace {port} with the port number you are using with Civetweb.

4. Run the script:

```
python s3test.py
```

The output will be something like the following:

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

10.3.2. Test swift access

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

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

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

To test swift access, execute the following:

```
swift -A http://{IP ADDRESS}:{port}/auth/1.0 -U testuser:swift -K
'{swift_secret_key}' list
```

Replace **{IP ADDRESS}** with the public IP address of the gateway server and **{swift_secret_key}** with its value from the output of **radosgw-admin key create** command executed for the **swift** user. Replace {port} with the port number you are using with Civetweb (e.g., **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
```

PART II. ADMINISTRATION (CLI)

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

- User Management
- Quota Management
- Usage Tracking

CHAPTER 11. 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. 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:

```
sudo 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.

11.1. 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.

```
sudo 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:

```
radosgw-admin user create --uid=janedoe --display-name="Jane Doe" --
email=jane@example.com
```

```
{ "user_id": "janedoe",
  "display_name": "Jane Doe",
  "email": "jane@example.com",
  "suspended": 0,
  "max_buckets": 1000,
  "auid": 0,
  "subusers": [],
```

```
  "keys": [
        { "user": "janedoe",
          "access_key": "11BS02LGFB6AL6H1ADMW",
          "secret_key": "vzCEkuryfn060dfee4fgQPqFrncKEIkh3ZcdOANY"}],
  "swift_keys": [],
  "caps": [],
  "op_mask": "read, write, delete",
  "default_placement": "",
  "placement_tags": [],
  "bucket_quota": { "enabled": false,
      "max_size_kb": -1,
      "max_objects": -1},
  "user_quota": { "enabled": false,
      "max_size_kb": -1,
      "max_objects": -1},
  "temp_url_keys": []}
```

Important

Check the key output. Sometimes **`radosgw-admin`** generates a JSON escape (\) character, and some clients do not know how to handle JSON escape characters. Remedies include removing the JSON escape character (\), encapsulating the string in quotes, regenerating the key and ensuring that it does not have a JSON escape character or specify the key and secret manually.

11.2. 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.

```
radosgw-admin subuser create --uid={uid} --subuser={uid} --access=[
read | write | readwrite | full ]
```

For example:

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

```
{ "user_id": "janedoe",
  "display_name": "Jane Doe",
  "email": "jane@example.com",
  "suspended": 0,
  "max_buckets": 1000,
  "auid": 0,
```

```
  "subusers": [
        { "id": "janedoe:swift",
          "permissions": "full-control"}],
  "keys": [
        { "user": "janedoe",
          "access_key": "11BS02LGFB6AL6H1ADMW",
          "secret_key": "vzCEkuryfn060dfee4fgQPqFrncKEIkh3ZcdOANY"}],
  "swift_keys": [],
  "caps": [],
  "op_mask": "read, write, delete",
  "default_placement": "",
  "placement_tags": [],
  "bucket_quota": { "enabled": false,
      "max_size_kb": -1,
      "max_objects": -1},
  "user_quota": { "enabled": false,
      "max_size_kb": -1,
      "max_objects": -1},
  "temp_url_keys": []}
```

11.3. GET USER INFO

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

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

11.4. MODIFY USER INFO

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:

```
radosgw-admin user modify --uid=janedoe --display-name="Jane E. Doe"
```

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

```
radosgw-admin subuser modify --uid=janedoe:swift --access=full
```

11.5. USER ENABLE/SUSPEND

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. :

```
radosgw-admin user suspend --uid=johndoe
```

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

```
radosgw-admin user enable --uid=johndoe
```

Note

Disabling the user disables the subuser.

11.6. 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.

```
radosgw-admin user rm --uid=<uid> [--purge-keys] [--purge-data]
```

For example:

```
radosgw-admin user rm --uid=johndoe --purge-data
```

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

```
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.

11.7. 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. :

```
radosgw-admin subuser rm --uid=johndoe:swift
```

Options include:

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

11.8. 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:

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

```
{ "user_id": "johndoe",
  "rados_uid": 0,
  "display_name": "John Doe",
```

```
  "email": "john@example.com",
  "suspended": 0,
  "subusers": [
     { "id": "johndoe:swift",
       "permissions": "full-control"}],
  "keys": [
    { "user": "johndoe",
      "access_key": "QFAMEDSJP5DEKJO0DDXY",
      "secret_key": "iaSFLDVvDdQt6lkNzHyW4fPLZugBAI1g17LO0+87"}],
  "swift_keys": [
    { "user": "johndoe:swift",
      "secret_key": "E9T2rUZNu2gxUjcwUBO8n\/Ev4KX6\/GprEuH4qhu1"}]}
```

11.9. ADD / 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. :

```
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. :

```
radosgw-admin key rm --uid=johndoe
```

11.10. ADD / 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:

```
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:

```
radosgw-admin caps add --uid=johndoe --caps="users=*"
```

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

```
radosgw-admin caps remove --uid=johndoe --caps={caps}
```

CHAPTER 12. 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.

12.1. SET USER QUOTA

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

```
radosgw-admin quota set --quota-scope=user --uid=<uid> [--max-objects=
<num objects>] [--max-size=<max size>]
```

For example:

```
radosgw-admin quota set --quota-scope=user --uid=johndoe --max-
objects=1024 --max-size=1024
```

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

12.2. ENABLE/DISABLE USER QUOTA

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

```
radosgw-admin quota enable --quota-scope=user --uid=<uid>
```

You may disable an enabled user quota. For example:

```
radosgw-admin quota-disable --quota-scope=user --uid=<uid>
```

12.3. SET BUCKET QUOTA

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

```
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.

12.4. ENABLE/DISABLE BUCKET QUOTA

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

```
radosgw-admin quota enable --quota-scope=bucket --uid=<uid>
```

You may disable an enabled bucket quota. For example:

```
radosgw-admin quota-disable --quota-scope=bucket --uid=<uid>
```

12.5. GET QUOTA SETTINGS

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

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

12.6. 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. :

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

12.7. GET USER USAGE STATS

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

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

Note

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

12.8. READING / WRITING GLOBAL QUOTAS

You can read and write quota settings in a region map. To get a region map, execute the following. :

```
radosgw-admin regionmap get > regionmap.json
```

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

```
radosgw-admin region set < regionmap.json
```

Note

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

CHAPTER 13. 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.

13.1. 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
```

13.2. 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. :

```
radosgw-admin usage trim --start-date=2010-01-01 --end-date=2010-12-31
radosgw-admin usage trim --uid=johndoe
radosgw-admin usage trim --uid=johndoe --end-date=2013-12-31
```

CHAPTER 14. CEPH OBJECT GATEWAY CONFIG REFERENCE

The following settings may added to the Ceph configuration file (i.e., 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.

Name	Description	Type	Default
`rgw_data`	Sets the location of the data files for Ceph Object Gateway.	String	**`/var/lib/ceph/radosgw/$cluster-$id`**
`rgw_enable_apis`	Enables the specified APIs.	String	**`s3, swift, swift_auth, admin`** All APIs.
`rgw_cache_enabled`	Whether the Ceph Object Gateway cache is enabled.	Boolean	**`true`**
`rgw_cache_lru_size`	The number of entries in the Ceph Object Gateway cache.	Integer	**`10000`**
`rgw_socket_path`	The socket path for the domain socket. **`FastCgiExternalServer`** uses this socket. If you do not specify a socket path, Ceph Object Gateway will not run as an external server. The path you specify here must be the same as the path specified in the **`rgw.conf`** file.	String	N/A
`rgw_host`	The host for the Ceph Object Gateway instance. Can be an IP address or a hostname.	String	**`0.0.0.0`**
`rgw_port`	Port the instance listens for requests. If not specified, Ceph Object Gateway runs external FastCGI.	String	None
`rgw_dns_name`	The DNS name of the served domain. See also the **`hostnames`** setting within regions.	String	None

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

Name	Description	Type	Default
rgw_gc_obj_min_wait	The minimum wait time before the object may be removed and handled by garbage collection processing.	Integer	**2 * 3600**
rgw_gc_processor_max_time	The maximum time between the beginning of two consecutive garbage collection processing cycles.	Integer	**3600**
rgw_gc_processor_period	The cycle time for garbage collection processing.	Integer	**3600**
rgw_s3 success_create_obj_status	The alternate success status response for **create-obj**.	Integer	**0**
rgw_resolve_cname	Whether **rgw** should use DNS CNAME record of the request hostname field (if hostname is not equal to **rgw_dns name**).	Boolean	**false**
rgw_object_stripe_size	The size of an object stripe for Ceph Object Gateway objects.	Integer	**4 << 20**
rgw_extended_http_attrs	Add new set of attributes that could be set on an object. These extra attributes can be set through HTTP header fields when putting the objects. If set, these attributes will return as HTTP fields when doing GET/HEAD on the object.	String	None. For example: "content_foo, content_bar"
rgw_exit_timeout_secs	Number of seconds to wait for a process before exiting unconditionally.	Integer	**120**

Name	Description	Type	Default
rgw_get_obj_window_size	The window size in bytes for a single object request.	Integer	**16 << 20**
rgw_get_obj_max_req_size	The maximum request size of a single get operation sent to the Ceph Storage Cluster.	Integer	**4 << 20**
rgw_relaxed s3_bucket_names	Enables relaxed S3 bucket names rules for US region buckets.	Boolean	**false**
rgw_list buckets_max_chunk	The maximum number of buckets to retrieve in a single operation when listing user buckets.	Integer	**1000**
rgw_num_zone_opstate_shards	The maximum number of shards for keeping inter-region copy progress information.	Integer	**128**
rgw_opstate_ratelimit_sec	The minimum time between opstate updates on a single upload. **0** disables the ratelimit.	Integer	**30**
rgw_curl_wait_timeout_ms	The timeout in milliseconds for certain **curl** calls.	Integer	**1000**
rgw_copy_obj_progress	Enables output of object progress during long copy operations.	Boolean	**true**
rgw_copy_obj_progress_every_bytes	The minimum bytes between copy progress output.	Integer	**1024 * 1024**
rgw_admin_entry	The entry point for an admin request URL.	String	**admin**
rgw_content_length_compat	Enable compatability handling of FCGI requests with both CONTENT_LENGTH AND HTTP_CONTENT_LENGTH set.	Boolean	**false**

14.1. REGIONS

The Ceph Object Gateway supports federated deployments and a global namespace via the notion of regions. A region defines the geographic location of one or more Ceph Object Gateway instances within one or more zones.

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

14.1.1. List Regions

A Ceph cluster contains a list of regions. To list the regions, execute:

```
sudo radosgw-admin regions list
```

The **`radosgw-admin`** returns a JSON formatted list of regions.

```
{ "default_info": { "default_region": "default"},
  "regions": [
        "default"]}
```

14.1.2. Get a Region Map

To list the details of each region, execute:

```
sudo radosgw-admin region-map get
```

Note

If you receive a **`failed to read region map`** error, run **`sudo radosgw-admin region-map update`** first.

14.1.3. Get a Region

To view the configuration of a region, execute:

```
radosgw-admin region get [--rgw-region=<region>]
```

The **`default`** region looks like this:

```
{"name": "default",
 "api_name": "",
 "is_master": "true",
 "endpoints": [],
 "hostnames": [],
 "master_zone": "",
 "zones": [
   {"name": "default",
    "endpoints": [],
    "log_meta": "false",
```

```
    "log_data": "false"}
  ],
 "placement_targets": [
   {"name": "default-placement",
    "tags": [] }],
 "default_placement": "default-placement"}
```

14.1.4. Set a Region

Defining a region consists of creating a JSON object, specifying at least the required settings:

1. **`name`**: The name of the region. Required.
2. **`api_name`**: The API name for the region. Optional.
3. **`is_master`**: Determines if the region is the master region. Required. **note:** You can only have one master region.
4. **`endpoints`**: A list of all the endpoints in the region. For example, you may use multiple domain names to refer to the same region. 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 region. For example, you may use multiple domain names to refer to the same region. Optional. The **`rgw dns name`** setting will automatically be included in this list. You should restart the **`radosgw`** daemon(s) after changing this setting.
6. **`master_zone`**: The master zone for the region. Optional. Uses the default zone if not specified. **note:** You can only have one master zone per region.
7. **`zones`**: A list of all zones within the region. 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 region, create a JSON object consisting of the required fields, save the object to a file (e.g., **`region.json`**); then, execute the following command:

```
sudo radosgw-admin region set --infile region.json
```

Where **`region.json`** is the JSON file you created.

Important

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

Finally, update the map. :

```
sudo radosgw-admin region-map update
```

14.1.5. Set a Region Map

Setting a region map consists of creating a JSON object consisting of one or more regions, and setting the **master_region** for the cluster. Each region in the region map consists of a key/value pair, where the **key** setting is equivalent to the **name** setting for an individual region configuration, and the **val** is a JSON object consisting of an individual region configuration.

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

```
{ "regions": [
     { "key": "default",
       "val": { "name": "default",
       "api_name": "",
       "is_master": "true",
       "endpoints": [],
       "hostnames": [],
       "master_zone": "",
       "zones": [
         { "name": "default",
           "endpoints": [],
           "log_meta": "false",
            "log_data": "false"}],
            "placement_targets": [
              { "name": "default-placement",
                "tags": []}],
                "default_placement": "default-placement"
              }
          }
       ],
   "master_region": "default"
}
```

To set a region map, execute the following:

```
sudo radosgw-admin region-map set --infile regionmap.json
```

Where **regionmap.json** is the JSON file you created. Ensure that you have zones created for the ones specified in the region map. Finally, update the map.

```
sudo radosgw-admin regionmap update
```

14.2. 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.

14.2.1. List Zones

To list the zones in a cluster, execute:

```
sudo radosgw-admin zone list
```

14.2.2. Get a Zone

To get the configuration of a zone, execute:

```
sudo 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"}
      }
    ]
  }
```

14.2.3. 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. See Pools_ for details of configuring pools.

To set a zone, create a JSON object consisting of the pools, save the object to a file (e.g., **`zone.json`**); then, execute the following command, replacing **`{zone-name}`** with the name of the zone:

```
sudo radosgw-admin zone set --rgw-zone={zone-name} --infile zone.json
```

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

14.3. REGION/ZONE SETTINGS

You may include the following settings in your Ceph configuration file under each **`[client.radosgw.{instance-name}]`** instance.

Name	Description	Type	Default
`rgw_zone`	The name of the zone for the gateway instance.	String	None
`rgw_region`	The name of the region for the gateway instance.	String	None
`rgw_default_region_info_oid`	The OID for storing the default region. We do not recommend changing this setting.	String	**`default.region`**

14.4. 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`**
- **`.rgw.control`**
- **`.rgw.gc`**
- **`.log`**
- **`.intent-log`**
- **`.usage`**
- **`.users`**
- **`.users.email`**
- **`.users.swift`**

- **.users.uid**

You have significant discretion in determining how you want a zone to access pools. You can create pools on a per zone basis, or use the same pools for multiple zones. As a best practice, we recommend having a separate set of pools for your master zone and your secondary zones in each region. When creating pools for a specific zone, consider prepending the region name and zone name to the default pool names. For example:

- **.region1-zone1.domain.rgw**
- **.region1-zone1.rgw.control**
- **.region1-zone1.rgw.gc**
- **.region1-zone1.log**
- **.region1-zone1.intent-log**
- **.region1-zone1.usage**
- **.region1-zone1.users**
- **.region1-zone1.users.email**
- **.region1-zone1.users.swift**
- **.region1-zone1.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 **.rgw.buckets.index** and for the data pool for default placement is **.rgw.buckets**.

Name	Description	Type	Default
rgw_region_root_pool	The pool for storing all region-specific information.	String	**.rgw.root**
rgw_zone_root_pool	The pool for storing zone-specific information.	String	**.rgw.root**

14.5. SWIFT SETTINGS

Name	Description	Type	Default
rgw_enforce_swift_acls	Enforces the Swift Access Control List (ACL) settings.	Boolean	**true**

Name	Description	Type	Default
rgw_swift_token_expiration	The time in seconds for expiring a Swift token.	Integer	**24 * 3600**
rgw_swift_url	The URL for the Ceph Object Gateway Swift API.	String	None
rgw_swift_url_prefix	The URL prefix for the Swift API (e.g., **http://fqdn.com/swift**).	**swift**	N/A
rgw_swift_auth_url	Default URL for verifying v1 auth tokens (if not using internal Swift auth).	String	None
rgw_swift_auth_entry	The entry point for a Swift auth URL.	String	**auth**

14.6. LOGGING SETTINGS

Name	Description	Type	Default
rgw_log_nonexistent_bucket	Enables Ceph Object Gateway to log a request for a non-existent bucket.	Boolean	**false**
rgw_log_object_name	The logging format for an object name. See manpage date for details about format specifiers.	Date	**%Y-%m-%d-%H-%i-%n**
rgw_log_object_name_utc	Whether a logged object name includes a UTC time. If **false**, it uses the local time.	Boolean	**false**
rgw_usage_max_shards	The maximum number of shards for usage logging.	Integer	**32**
rgw_usage_max_user_shards	The maximum number of shards used for a single user's usage logging.	Integer	**1**

Name	Description	Type	Default
rgw_enable_ops_log	Enable logging for each successful Ceph Object Gateway operation.	Boolean	**false**
rgw_enable_usage_log	Enable the usage log.	Boolean	**false**
rgw_ops_log_rados	Whether the operations log should be written to the Ceph Storage Cluster backend.	Boolean	**true**
rgw_ops_log_socket_path	The Unix domain socket for writing operations logs.	String	None
rgw_ops_log_data-backlog	The maximum data backlog data size for operations logs written to a Unix domain socket.	Integer	**5 << 20**
rgw_usage_log_flush_threshold	The number of dirty merged entries in the usage log before flushing synchronously.	Integer	1024
rgw_usage_log_tick_interval	Flush pending usage log data every **n** seconds.	Integer	**30**
rgw_intent_log_object_name	The logging format for the intent log object name. See manpage date for details about format specifiers.	Date	**%Y-%m-%d-%i-%n**
rgw_intent_log_object_name_utc	Whether the intent log object name includes a UTC time. If **false**, it uses the local time.	Boolean	**false**
rgw_data_log_window	The data log entries window in seconds.	Integer	**30**
rgw_data_log_changes_size	The number of in-memory entries to hold for the data changes log.	Integer	**1000**

Name	Description	Type	Default
rgw_data_log_num_shards	The number of shards (objects) on which to keep the data changes log.	Integer	**128**
rgw_data_log_obj_prefix	The object name prefix for the data log.	String	**data_log**
rgw_replica_log_obj_prefix	The object name prefix for the replica log.	String	**replica log**
rgw_md_log_max_shards	The maximum number of shards for the metadata log.	Integer	**64**

14.7. KEYSTONE SETTINGS

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

PART III. OBJECT GATEWAY ADMIN API

The Ceph Object Gateway exposes features of the **`radosgw-admin`** CLI in a RESTful API too. We recommend using the CLI interface when setting up your Ceph Object Gateway. When you want to manage users, data, quotas and usage, the Ceph Object Gateway's Admin API provides a RESTful interface that you can integrate with your management platform(s). Typical Admin API features include:

- Create/Get/Modify/Delete User/Subuser
- User Capabilities Management
- Create/Delete Key
- Get/Trim Usage
- Bucket Operations
  - Get Bucket Info
  - Check Bucket Index
  - Remove Bucket
  - Link/Unlink Bucket
- Object Operations
- Quotas

CHAPTER 15. CREATE AN ADMIN USER

To use the Ceph Object Gateway Admin API, you must first:

1. Create an object gateway user.

```
radosgw-admin user create --uid="admin-api-user" --display-
name="Admin API User"
```

   The **`radosgw-admin`** CLI will return the user. It will look something like this:

```
{
    "user_id": "admin-api-user",
    "display_name": "Admin API User",
    "email": "",
    "suspended": 0,
    "max_buckets": 1000,
    "auid": 0,
    "subusers": [],
    "keys": [
        {
            "user": "admin-api-user",
            "access_key": "NRWGT19TWMYOB1YDBV1Y",
            "secret_key":
"gr1VEGIV7rxcP3xvXDFCo4UDwwl2YoNrmtRlIAty"
        }
    ],
    "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": []
}
```

2. Assign administrative capabilities to the user you create.

```
radosgw-admin caps add --uid=admin-api-user --caps="users=*"
```

   The **`radosgw-admin`** CLI will return the user. The **`"caps":`** will have the capabilities you assigned to the user:

```
{
    "user_id": "admin-api-user",
```

```
    "display_name": "Admin API User",
    "email": "",
    "suspended": 0,
    "max_buckets": 1000,
    "auid": 0,
    "subusers": [],
    "keys": [
        {
            "user": "admin-api-user",
            "access_key": "NRWGT19TWMYOB1YDBV1Y",
            "secret_key":
"gr1VEGIV7rxcP3xvXDFCo4UDwwl2YoNrmtRlIAty"
        }
    ],
    "swift_keys": [],
    "caps": [
        {
            "type": "users",
            "perm": "*"
        }
    ],
    "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": []
}
```

Now you have a user with administrative privileges.

Amazon's S3 service uses the access key and a hash of the request header and the secret key to authenticate the request, which has the benefit of providing an authenticated request (especially large uploads) without SSL overhead.

```
Authorization: AWS {access-key}:{hash-of-header-and-secret}
```

See the S3 API authentication section for additional details.

CHAPTER 16. ADMIN OPERATIONS API

An admin API request will be done on a URI that starts with the configurable *admin* resource entry point. Authorization for the admin API duplicates the S3 authorization mechanism. Some operations require that the user holds special administrative capabilities. The response entity type (XML or JSON) may be specified as the *format* option in the request and defaults to JSON if not specified.

16.1. GET USAGE

Request bandwidth usage information.

caps

`usage=read`

16.1.1. Syntax

```
GET /{admin}/usage?format=json HTTP/1.1
Host: {fqdn}
```

16.1.2. Request Parameters

Name	Description	Type	Required
`uid`	The user for which the information is requested.	String.	Yes
`start`	Date and (optional) time that specifies the start time of the requested data. E.g., **`2012-09-25 16:00:00`**	String	No
`end`	Date and (optional) time that specifies the end time of the requested data (non-inclusive). E.g., **`2012-09-25 16:00:00`**	String	No
`show-entries`	Specifies whether data entries should be returned.	Boolean	No
`show-summary`	Specifies whether data summary should be returned.	Boolean	No

16.1.3. Response Entities

If successful, the response contains the requested information.

Name	Description	Type
usage	A container for the usage information.	Container
entries	A container for the usage entries information.	Container
user	A container for the user data information.	Container
owner	The name of the user that owns the buckets.	String
bucket	The bucket name.	String
time	Time lower bound for which data is being specified (rounded to the beginning of the first relevant hour).	String
epoch	The time specified in seconds since **1/1/1970**.	String
categories	A container for stats categories.	Container
entry	A container for stats entry.	Container
category	Name of request category for which the stats are provided.	String
bytes_sent	Number of bytes sent by the Ceph Object Gateway.	Integer
bytes_received	Number of bytes received by the Ceph Object Gateway.	Integer
ops	Number of operations.	Integer

Name	Description	Type
successful_ops	Number of successful operations.	Integer
summary	A container for stats summary.	Container
total	A container for stats summary aggregated total.	Container

16.2. TRIM USAGE

Remove usage information. With no dates specified, removes all usage information.

caps

usage=write

16.2.1. Syntax

```
DELETE /{admin}/usage?format=json HTTP/1.1
Host: {fqdn}
```

16.2.2. Request Parameters

Name	Description	Type	Example	Required
uid	The user for which the information is requested.	String	**foo_user**	No
start	Date and (optional) time that specifies the start time of the requested data.	String	**2012-09-25 16:00:00**	No
end	Date and (optional) time that specifies the end time of the requested data (none inclusive).	String	**2012-09-25 16:00:00**	No

Name	Description	Type	Example	Required
remove-all	Required when **uid** is not specified, in order to acknowledge multi-user data removal.	Boolean	True [False]	No

16.3. GET USER INFO

Get user information.

caps

users=read

16.3.1. Syntax

```
GET /{admin}/user?format=json HTTP/1.1
Host: {fqdn}
```

16.3.2. Request Parameters

Name	Description	Type	Example	Required
uid	The user for which the information is requested.	String	**foo_user**	Yes

16.3.3. Response Entities

If successful, the response contains the user information.

Name	Description	Type	Parent
user	A container for the user data information.	Container	N/A
user_id	The user ID.	String	**user**

Name	Description	Type	Parent
display_name	Display name for the user.	String	**user**
suspended	True if the user is suspended.	Boolean	**user**
max_buckets	The maximum number of buckets to be owned by the user.	Integer	**user**
subusers	Subusers associated with this user account.	Container	**user**
keys	S3 keys associated with this user account.	Container	**user**
swift_keys	Swift keys associated with this user account.	Container	**user**
caps	User capabilities.	Container	**user**

16.3.4. Special Error Responses

None.

16.4. CREATE USER

Create a new user. By Default, a S3 key pair will be created automatically and returned in the response. If only one of **access-key** or **secret-key** is provided, the omitted key will be automatically generated. By default, a generated key is added to the keyring without replacing an existing key pair. If **access-key** is specified and refers to an existing key owned by the user then it will be modified.

caps

```
users=write
```

16.4.1. Syntax

```
PUT /{admin}/user?format=json HTTP/1.1
Host: {fqdn}
```

16.4.2. Request Parameters

Name	Description	Type	Example	Required
uid	The user ID to be created.	String	**foo_user**	Yes
display-name	The display name of the user to be created.	String	**foo user**	Yes
email	The email address associated with the user.	String	**foo@bar.com**	No
key-type	Key type to be generated, options are: swift, s3 (default).	String	**s3** [**s3**]	No
access-key	Specify access key.	String	**ABCD0EF12GHIJ2K34LMN**	No
secret-key	Specify secret key.	String	**0AbCDEFg1h2i34JklM5nop6QrSTUV+WxyzaBC7D8**	No
user-caps	User capabilities.	String	**usage=read, write; users=read**	No
generate-key	Generate a new key pair and add to the existing keyring.	Boolean	True [True]	No
max-buckets	Specify the maximum number of buckets the user can own.	Integer	500 [1000]	No

Name	Description	Type	Example	Required
suspended	Specify whether the user should be suspended.	Boolean	False [False]	No

16.4.3. Response Entities

If successful, the response contains the user information.

Name	Description	Type	Parent
user	A container for the user data information.	Container	N/A
user_id	The user ID.	String	**user**
display_name	Display name for the user.	String	**user**
suspended	True if the user is suspended.	Boolean	**user**
max_buckets	The maximum number of buckets to be owned by the user.	Integer	**user**
subusers	Subusers associated with this user account.	Container	**user**
keys	S3 keys associated with this user account.	Container	**user**
swift_keys	Swift keys associated with this user account.	Container	**user**
caps	User capabilities.	Container	**user**

Name	Description	Type	Parent

16.4.4. Special Error Responses

Name	Description	Code
`UserExists`	Attempt to create existing user.	409 Conflict
`InvalidAccessKey`	Invalid access key specified.	400 Bad Request
`InvalidKeyType`	Invalid key type specified.	400 Bad Request
`InvalidSecretKey`	Invalid secret key specified.	400 Bad Request
`InvalidKeyType`	Invalid key type specified.	400 Bad Request
`KeyExists`	Provided access key exists and belongs to another user.	409 Conflict
`EmailExists`	Provided email address exists.	409 Conflict
`InvalidCap`	Attempt to grant invalid admin capability.	400 Bad Request

16.5. MODIFY USER

Modify a user.

`caps`

`users=write`

16.5.1. Syntax

```
POST /{admin}/user?format=json HTTP/1.1
Host: {fqdn}
```

16.5.2. Request Parameters

Name	Description	Type	Example	Required
uid	The user ID to be modified.	String	**foo_user**	Yes
display-name	The display name of the user to be modified.	String	**foo user**	No
email	The email address to be associated with the user.	String	**foo@bar.com**	No
generate-key	Generate a new key pair and add to the existing keyring.	Boolean	True [False]	No
access-key	Specify access key.	String	**ABCD0EF12GHIJ2K34LMN**	No
secret-key	Specify secret key.	String	**0AbCDEFg1h2i34JklM5nop6QrSTUV+WxyzaBC7D8**	No
key-type	Key type to be generated, options are: swift, s3 (default).	String	**s3**	No
user-caps	User capabilities.	String	**usage=read, write; users=read**	No
max-buckets	Specify the maximum number of buckets the user can own.	Integer	500 [1000]	No

Name	Description	Type	Example	Required
suspended	Specify whether the user should be suspended.	Boolean	False [False]	No

16.5.3. Response Entities

If successful, the response contains the user information.

Name	Description	Type	Parent
user	A container for the user data information.	Container	N/A
user_id	The user ID.	String	**user**
display_name	Display name for the user.	String	**user**
suspended	True if the user is suspended.	Boolean	**user**
max_buckets	The maximum number of buckets to be owned by the user.	Integer	**user**
subusers	Subusers associated with this user account.	Container	**user**
keys	S3 keys associated with this user account.	Container	**user**
swift_keys	Swift keys associated with this user account.	Container	**user**

Name	Description	Type	Parent
caps	User capabilities.	Container	**user**

16.5.4. Special Error Responses

Name	Description	Code
InvalidAccessKey	Invalid access key specified.	400 Bad Request
InvalidKeyType	Invalid key type specified.	400 Bad Request
InvalidSecretKey	Invalid secret key specified.	400 Bad Request
KeyExists	Provided access key exists and belongs to another user.	409 Conflict
EmailExists	Provided email address exists.	409 Conflict
InvalidCap	Attempt to grant invalid admin capability.	400 Bad Request

16.6. REMOVE USER

Remove an existing user.

caps

```
users=write
```

16.6.1. Syntax

```
DELETE /{admin}/user?format=json HTTP/1.1
Host: {fqdn}
```

16.6.2. Request Parameters

16.6.2. Request Parameters

Name	Description	Type	Example	Required
uid	The user ID to be removed.	String	**foo_user**	Yes.
purge-data	When specified the buckets and objects belonging to the user will also be removed.	Boolean	True	No

16.6.3. Response Entities

None

16.6.4. Special Error Responses

None.

16.7. CREATE SUBUSER

Create a new subuser (primarily useful for clients using the Swift API). Note that either **gen-subuser** or **subuser** is required for a valid request. Note that in general for a subuser to be useful, it must be granted permissions by specifying **access**. As with user creation if **subuser** is specified without **secret**, then a secret key will be automatically generated.

caps

users=write

16.7.1. Syntax

```
PUT /{admin}/user?subuser&format=json HTTP/1.1
Host {fqdn}
```

16.7.2. Request Parameters

Name	Description	Type	Example	Required
uid	The user ID under which a subuser is to be created.	String	**foo_user**	Yes

Name	Description	Type	Example	Required
subuser	Specify the subuser ID to be created.	String	**sub_foo**	No
secret-key	Specify secret key.	String	**0AbCDEFg1h2i34JklM5nop6QrSTUVWxyzaBC7D8**	No
key-type	Key type to be generated, options are: swift (default), s3.	String	**swift** [**swift**]	No
access	Set access permissions for sub-user, should be one of **read, write, readwrite, full**.	String	**read**	No
generate-secret	Generate the secret key.	Boolean	True [False]	No

16.7.3. Response Entities

If successful, the response contains the subuser information.

Name	Description	Type	Parent
subusers	Subusers associated with the user account.	Container	N/A
id	Subuser ID.	String	**subusers**

Name	Description	Type	Parent
permissions	Subuser access to user account.	String	**subusers**

16.7.4. Special Error Responses

Name	Description	Code
SubuserExists	Specified subuser exists.	409 Conflict
InvalidKeyType	Invalid key type specified.	400 Bad Request
InvalidSecretKey	Invalid secret key specified.	400 Bad Request
InvalidAccess	Invalid subuser access specified.	400 Bad Request

16.8. MODIFY SUBUSER

Modify an existing subuser

caps

users=write

16.8.1. Syntax

```
POST /{admin}/user?subuser&format=json HTTP/1.1
Host {fqdn}
```

16.8.2. Request Parameters

Name	Description	Type	Example	Required
uid	The user ID under which the subuser is to be modified.	String	**foo_user**	Yes
subuser	The subuser ID to be modified.	String	**sub_foo**	Yes
generate-secret	Generate a new secret key for the subuser, replacing the existing key.	Boolean	True [False]	No
secret	Specify secret key.	String	**0AbCDEFg1h2i34JklM5nop6QrSTUV+WxyzaBC7D8**	No
key-type	Key type to be generated, options are: swift (default), s3.	String	**swift** [**swift**]	No
access	Set access permissions for sub-user, should be one of **read, write, readwrite, full**.	String	**read**	No

16.8.3. Response Entities

If successful, the response contains the subuser information.

Name	Description	Type	Parent
subusers	Subusers associated with the user account.	Container	N/A
id	Subuser ID.	String	**subusers**

Name	Description	Type	Parent
permissions	Subuser access to user account.	String	**subusers**

16.8.4. Special Error Responses

Name	Description	Code
InvalidKeyType	Invalid key type specified.	400 Bad Request
InvalidSecretKey	Invalid secret key specified.	400 Bad Request
InvalidAccess	Invalid subuser access specified.	400 Bad Request

16.9. REMOVE SUBUSER

Remove an existing subuser

caps

users=write

16.9.1. Syntax

```
DELETE /{admin}/user?subuser&format=json HTTP/1.1
Host {fqdn}
```

16.9.2. Request Parameters

Name	Description	Type	Example	Required
uid	The user ID under which the subuser is to be removed.	String	**foo_user**	Yes

Name	Description	Type	Example	Required
subuser	The subuser ID to be removed.	String	**sub_foo**	Yes
purge-keys	Remove keys belonging to the subuser.	Boolean	True [True]	No

16.9.3. Response Entities

None.

16.9.4. Special Error Responses

None.

16.10. CREATE KEY

Create a new key. If a **subuser** is specified then by default created keys will be swift type. If only one of **access-key** or **secret-key** is provided the committed key will be automatically generated, that is if only **secret-key** is specified then **access-key** will be automatically generated. By default, a generated key is added to the keyring without replacing an existing key pair. If **access-key** is specified and refers to an existing key owned by the user then it will be modified. The response is a container listing all keys of the same type as the key created. Note that when creating a swift key, specifying the option **access-key** will have no effect. Additionally, only one swift key may be held by each user or subuser.

caps

users=write

16.10.1. Syntax

```
PUT /{admin}/user?key&format=json HTTP/1.1
Host {fqdn}
```

16.10.2. Request Parameters

Name	Description	Type	Example	Required
uid	The user ID to receive the new key.	String	**foo_user**	Yes

Name	Description	Type	Example	Required
subuser	The subuser ID to receive the new key.	String	**sub_foo**	No
key-type	Key type to be generated, options are: swift, s3 (default).	String	**s3** [**s3**]	No
access-key	Specify the access key.	String	**AB01C2D3EF45G6H7IJ8K**	No
secret-key	Specify the secret key.	String	**0ab/CdeFGhij1klmnopqRSTUv1WxyZabcDEFgHij**	No
generate-key	Generate a new key pair and add to the existing keyring.	Boolean	True [**True**]	No

16.10.3. Response Entities

Name	Description	Type	Parent
keys	Keys of type created associated with this user account.	Container	N/A
user	The user account associated with the key.	String	**keys**
access-key	The access key.	String	**keys**

Name	Description	Type	Parent
secret-key	The secret key	String	**keys**

16.10.4. Special Error Responses

Name	Description	Code
InvalidAccessKey	Invalid access key specified.	400 Bad Request
InvalidKeyType	Invalid key type specified.	400 Bad Request
InvalidSecretKey	Invalid secret key specified.	400 Bad Request
InvalidKeyType	Invalid key type specified.	400 Bad Request
KeyExists	Provided access key exists and belongs to another user.	409 Conflict

16.11. REMOVE KEY

Remove an existing key.

caps

users=write

16.11.1. Syntax

```
DELETE /{admin}/user?key&format=json HTTP/1.1
Host {fqdn}
```

16.11.2. Request Parameters

Name	Description	Type	Example	Required
access-key	The S3 access key belonging to the S3 key pair to remove.	String	**AB01C2D3EF45G6H7IJ8K**	Yes
uid	The user to remove the key from.	String	**foo_user**	No
subuser	The subuser to remove the key from.	String	**sub_foo**	No
key-type	Key type to be removed, options are: swift, s3. NOTE: Required to remove swift key.	String	**swift**	No

16.11.3. Special Error Responses

None.

16.11.4. Response Entities

None.

16.12. GET BUCKET INFO

Get information about a subset of the existing buckets. If **uid** is specified without **bucket** then all buckets beloning to the user will be returned. If **bucket** alone is specified, information for that particular bucket will be retrieved.

caps

buckets=read

16.12.1. Syntax

```
GET /{admin}/bucket?format=json HTTP/1.1
Host {fqdn}
```

16.12.2. Request Parameters

Name	Description	Type	Example	Required
bucket	The bucket to return info on.	String	**foo_bucket**	No
uid	The user to retrieve bucket information for.	String	**foo_user**	No
stats	Return bucket statistics.	Boolean	True [False]	No

16.12.3. Response Entities

If successful the request returns a buckets container containing the desired bucket information.

Name	Description	Type	Parent
stats	Per bucket information.	Container	N/A
buckets	Contains a list of one or more bucket containers.	Container	**bucket**
Container for single bucket information.	Container	**buckets**	**name**
The name of the bucket.	String	**bucket**	**pool**
The pool the bucket is stored in.	String	**bucket**	**id**
The unique bucket ID.	String	**bucket**	**marker**

Name	Description	Type	Parent
Internal bucket tag.	String	**bucket**	**owner**
The user ID of the bucket owner.	String	**bucket**	**usage**
Storage usage information.	Container	**bucket**	**index**

16.12.4. Special Error Responses

Name	Description	Code
IndexRepairFailed	Bucket index repair failed.	409 Conflict

16.13. CHECK BUCKET INDEX

Check the index of an existing bucket. NOTE: to check multipart object accounting with **check-objects**, **fix** must be set to True.

caps

buckets=write

16.13.1. Syntax

```
GET /{admin}/bucket?index&format=json HTTP/1.1
Host {fqdn}
```

16.13.2. Request Parameters

Name	Description	Type	Example	Required
bucket	The bucket to return info on.	String	**foo_bucket**	Yes

Name	Description	Type	Example	Required
check-objects	Check multipart object accounting.	Boolean	True [False]	No
fix	Also fix the bucket index when checking.	Boolean	False [False]	No

16.13.3. Response Entities

Name	Description	Type
index	Status of bucket index.	String

16.13.4. Special Error Responses

Name	Description	Code
IndexRepairFailed	Bucket index repair failed.	409 Conflict

16.14. REMOVE BUCKET

Delete an existing bucket.

caps

buckets=write

16.14.1. Syntax

```
DELETE /{admin}/bucket?format=json HTTP/1.1
Host {fqdn}
```

16.14.2. Request Parameters

Name	Description	Type	Example	Required
bucket	The bucket to remove.	String	**foo_bucket**	Yes
purge-objects	Remove a buckets objects before deletion.	Boolean	True [False]	No

16.14.3. Response Entities

None.

16.14.4. Special Error Responses

Name	Description	Code
BucketNotEmpty	Attempted to delete non-empty bucket.	409 Conflict
ObjectRemovalFailed	Unable to remove objects.	409 Conflict

16.15. UNLINK BUCKET

Unlink a bucket from a specified user. Primarily useful for changing bucket ownership.

caps

buckets=write

16.15.1. Syntax

```
POST /{admin}/bucket?format=json HTTP/1.1
Host {fqdn}
```

16.15.2. Request Parameters

Name	Description	Type	Example	Required
bucket	The bucket to unlink.	String	**foo_bucket**	Yes
uid	The user ID to unlink the bucket from.	String	**foo_user**	Yes

16.15.3. Response Entities

None.

16.15.4. Special Error Responses

Name	Description	Code
BucketUnlinkFailed	Unable to unlink bucket from specified user.	409 Conflict

16.16. LINK BUCKET

Link a bucket to a specified user, unlinking the bucket from any previous user.

caps

buckets=write

16.16.1. Syntax

```
PUT /{admin}/bucket?format=json HTTP/1.1
Host {fqdn}
```

16.16.2. Request Parameters

Name	Description	Type	Example	Required
bucket	The bucket to unlink.	String	**foo_bucket**	Yes

Name	Description	Type	Example	Required
uid	The user ID to link the bucket to.	String	**foo_user**	Yes

16.16.3. Response Entities

Name	Description	Type	Parent
bucket	Container for single bucket information.	Container	N/A
name	The name of the bucket.	String	**bucket**
pool	The pool the bucket is stored in.	String	**bucket**
id	The unique bucket ID.	String	**bucket**
marker	Internal bucket tag.	String	**bucket**
owner	The user ID of the bucket owner.	String	**bucket**
usage	Storage usage information.	Container	**bucket**
index	Status of bucket index.	String	**bucket**

16.16.4. Special Error Responses

Name	Description	Code
`BucketUnlinkFailed`	Unable to unlink bucket from specified user.	409 Conflict
`BucketLinkFailed`	Unable to link bucket to specified user.	409 Conflict

16.17. REMOVE OBJECT

Remove an existing object. NOTE: Does not require owner to be non-suspended.

caps

`buckets=write`

16.17.1. Syntax

```
DELETE /{admin}/bucket?object&format=json HTTP/1.1
Host {fqdn}
```

16.17.2. Request Parameters

Name	Description	Type	Example	Required
`bucket`	The bucket containing the object to be removed.	String	**`foo_bucket`**	Yes
`object`	The object to remove.	String	**`foo.txt`**	Yes

16.17.3. Response Entities

None.

16.17.4. Special Error Responses

Name	Description	Code
NoSuchObject	Specified object does not exist.	404 Not Found
ObjectRemovalFailed	Unable to remove objects.	409 Conflict

16.18. GET BUCKET OR OBJECT POLICY

Read the policy of an object or bucket.

caps

buckets=read

16.18.1. Syntax

```
GET /{admin}/bucket?policy&format=json HTTP/1.1
Host {fqdn}
```

16.18.2. Request Parameters

Name	Description	Type	Example	Required
bucket	The bucket to read the policy from.	String	**foo_bucket**	Yes
object	The object to read the policy from.	String	**foo.txt**	No

16.18.3. Response Entities

If successful, returns the object or bucket policy

|**policy** | Access control policy.|Container

16.18.4. Special Error Responses

Name	Description	Code
`IncompleteBody`	Either bucket was not specified for a bucket policy request or bucket and object were not specified for an object policy request.	400 Bad Request

16.19. ADD A USER CAPABILITY

Add an administrative capability to a specified user.

caps

`users=write`

16.19.1. Syntax

```
PUT /{admin}/user?caps&format=json HTTP/1.1
Host {fqdn}
```

16.19.2. Request Parameters

Name	Description	Type	Example	Required
`uid`	The user ID to add an administrative capability to.	String	**`foo_user`**	Yes
`user-caps`	The administrative capability to add to the user.	String	**`usage=read, write`**	Yes

16.19.3. Response Entities

If successful, the response contains the user's capabilities.

Name	Description	Type	Parent
`user`	A container for the user data information.	Container	N/A

Name	Description	Type	Parent
user_id	The user ID.	String	**user**
caps	User capabilities.	Container	**user**

16.19.4. Special Error Responses

Name	Description	Code
InvalidCap	Attempt to grant invalid admin capability.	400 Bad Request

16.19.5. Example Request

```
PUT /{admin}/user?caps&format=json HTTP/1.1
Host: {fqdn}
Content-Type: text/plain
Authorization: {your-authorization-token}

usage=read
```

16.20. REMOVE A USER CAPABILITY

Remove an administrative capability from a specified user.

caps

users=write

16.20.1. Syntax

```
DELETE /{admin}/user?caps&format=json HTTP/1.1
Host {fqdn}
```

16.20.2. Request Parameters

Name	Description	Type	Example	Required
`uid`	The user ID to remove an administrative capability from.	String	`foo_user`	Yes
`user-caps`	The administrative capabilities to remove from the user.	String	`usage=read, write`	Yes

16.20.3. Response Entities

If successful, the response contains the user's capabilities.

Name	Description	Type	Parent
`user`	A container for the user data information.	Container	N/A
`user_id`	The user ID.	String	`user`
`caps`	User capabilities.	Container	`user`

16.20.4. Special Error Responses

Name	Description	Code
`InvalidCap`	Attempt to remove an invalid admin capability.	400 Bad Request
`NoSuchCap`	User does not possess specified capability.	404 Not Found

16.20.5. Special Error Responses

None.

16.21. OUOTAS

The Admin Operations API enables you to set quotas on users and on bucket owned by users. See Quota Management for additional details. Quotas include the maximum number of objects in a bucket and the maximum storage size in megabytes.

To view quotas, the user must have a **`users=read`** capability. To set, modify or disable a quota, the user must have **`users=write`** capability. See the Administration (CLI) for details.

Valid parameters for quotas include:

- **Bucket:** The **`bucket`** option allows you to specify a quota for buckets owned by a user.
- **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`**.

16.21.1. Get User Quota

To get a quota, the user must have **`users`** capability set with **`read`** permission.

```
GET /admin/user?quota&uid=<uid>&quota-type=user
```

16.21.2. Set User Quota

To set a quota, the user must have **`users`** capability set with **`write`** permission.

```
PUT /admin/user?quota&uid=<uid>&quota-type=user
```

The content must include a JSON representation of the quota settings as encoded in the corresponding read operation.

16.21.3. Get Bucket Quota

To get a quota, the user must have **`users`** capability set with **`read`** permission.

```
GET /admin/user?quota&uid=<uid>&quota-type=bucket
```

16.21.4. Set Bucket Quota

To set a quota, the user must have **`users`** capability set with **`write`** permission.

```
PUT /admin/user?quota&uid=<uid>&quota-type=bucket
```

The content must include a JSON representation of the quota settings as encoded in the corresponding read operation.

16.22. STANDARD ERROR RESPONSES

Name	Description	Code
`AccessDenied`	Access denied.	403 Forbidden
`InternalError`	Internal server error.	500 Internal Server Error
`NoSuchUser`	User does not exist.	404 Not Found
`NoSuchBucket`	Bucket does not exist.	404 Not Found
`NoSuchKey`	No such access key.	404 Not Found

PART IV. CEPH OBJECT GATEWAY S3 API

Red Hat Ceph Object Gateway supports a RESTful API that is compatible with the basic data access model of the Amazon S3 API.

CHAPTER 17. API

17.1. FEATURES SUPPORT

The following table describes the support status for current Amazon S3 functional features:

Feature	Status	Remarks
List Buckets	Supported	
Create Bucket	Supported	Different set of canned ACLs
Get Bucket	Supported	
Get Bucket Location	Supported	
Delete Bucket	Supported	
Bucket ACLs (Get, Put)	Supported	Different set of canned ACLs
Bucket Object Versions	Supported	
Get Bucket Info (HEAD)	Supported	
List Bucket Multipart Uploads	Supported	
Bucket Lifecycle	Not Supported	
Policy (Buckets, Objects)	Not Supported	ACLs are supported
Bucket Website	Not Supported	
Bucket Notification	Not Supported	

Feature	Status	Remarks
Bucket Request Payment	Not Supported	
Put Object	Supported	
Delete Object	Supported	
Get Object	Supported	
Object ACLs (Get, Put)	Supported	
Get Object Info (HEAD)	Supported	
Copy Object	Supported	
Initiate Multipart Upload	Supported	
Initiage Multipart Upload Part	Supported	
List Multipart Upload Parts	Supported	
Complete Multipart Upload	Supported	
Abort Multipart Upload	Supported	
Multipart Uploads	Supported	(missing Copy Part)

17.2. UNSUPPORTED HEADER FIELDS

The following common request header fields are not supported:

Name	Type
x-amz-security-token	Request
Server	Response
x-amz-delete-marker	Response
x-amz-id-2	Response
x-amz-request-id	Response
x-amz-version-id	Response

CHAPTER 18. COMMON ENTITIES

18.1. BUCKET AND HOST NAME

There are two different modes of accessing the buckets. The first (preferred) method identifies the bucket as the top-level directory in the URI.

```
GET /mybucket HTTP/1.1
Host: cname.domain.com
```

The second method identifies the bucket via a virtual bucket host name. For example:

```
GET / HTTP/1.1
Host: mybucket.cname.domain.com
```

Tip

We prefer the first method, because the second method requires expensive domain certification and DNS wild cards.

18.2. COMMON REQUEST HEADERS

Request Header	Description
CONTENT_LENGTH	Length of the request body.
DATE	Request time and date (in UTC).
HOST	The name of the host server.
AUTHORIZATION	Authorization token.

18.3. COMMON RESPONSE STATUS

HTTP Status	Response Code
100	Continue

HTTP Status	Response Code
200	Success
201	Created
202	Accepted
204	NoContent
206	Partial content
304	NotModified
400	InvalidArgument
400	InvalidDigest
400	BadDigest
400	InvalidBucketName
400	InvalidObjectName
400	UnresolvableGrantByEmailAddress
400	InvalidPart
400	InvalidPartOrder
400	RequestTimeout

HTTP Status	Response Code
400	EntityTooLarge
403	AccessDenied
403	UserSuspended
403	RequestTimeTooSkewed
404	NoSuchKey
404	NoSuchBucket
404	NoSuchUpload
405	MethodNotAllowed
408	RequestTimeout
409	BucketAlreadyExists
409	BucketNotEmpty
411	MissingContentLength
412	PreconditionFailed
416	InvalidRange
422	UnprocessableEntity

HTTP Status	Response Code
`500`	InternalError

CHAPTER 19. AUTHENTICATION AND ACLS

Requests to the Ceph Object Gateway can be either authenticated or unauthenticated. Ceph Object Gateway assumes unauthenticated requests are sent by an anonymous user. Ceph Object Gateway supports canned ACLs.

19.1. AUTHENTICATION

Authenticating a request requires including an access key and a Hash-based Message Authentication Code (HMAC) in the request before it is sent to the Ceph Object Gateway server. Ceph Object Gateway uses an S3-compatible authentication approach.

```
HTTP/1.1
PUT /buckets/bucket/object.mpeg
Host: cname.domain.com
Date: Mon, 2 Jan 2012 00:01:01 +0000
Content-Encoding: mpeg
Content-Length: 9999999

Authorization: AWS {access-key}:{hash-of-header-and-secret}
```

In the foregoing example, replace **`{access-key}`** with the value for your access key ID followed by a colon (**`:`**). Replace **`{hash-of-header-and-secret}`** with a hash of the header string and the secret corresponding to the access key ID.

To generate the hash of the header string and secret, you must:

1. Get the value of the header string.
2. Normalize the request header string into canonical form.
3. Generate an HMAC using a SHA-1 hashing algorithm.
4. Encode the **`hmac`** result as base-64.

To normalize the header into canonical form:

1. Get all fields beginning with **`x-amz-`**.
2. Ensure that the fields are all lowercase.
3. Sort the fields lexicographically.
4. Combine multiple instances of the same field name into a single field and separate the field values with a comma.
5. Replace white space and line breaks in field values with a single space.
6. Remove white space before and after colons.
7. Append a new line after each field.
8. Merge the fields back into the header.

Replace the **`{hash-of-header-and-secret}`** with the base-64 encoded HMAC string.

19.2. ACCESS CONTROL LISTS (ACLS)

Ceph Object Gateway supports S3-compatible ACL functionality. An ACL is a list of access grants that specify which operations a user can perform on a bucket or on an object. Each grant has a different meaning when applied to a bucket versus applied to an object:

Permission	Bucket	Object
`READ`	Grantee can list the objects in the bucket.	Grantee can read the object.
`WRITE`	Grantee can write or delete objects in the bucket.	N/A
`READ_ACP`	Grantee can read bucket ACL.	Grantee can read the object ACL.
`WRITE_ACP`	Grantee can write bucket ACL.	Grantee can write to the object ACL.
`FULL_CONTROL`	Grantee has full permissions for object in the bucket.	Grantee can read or write to the object ACL.

CHAPTER 20. SERVICE OPERATIONS

20.1. LIST BUCKETS

GET / returns a list of buckets created by the user making the request. **GET /** only returns buckets created by an authenticated user. You cannot make an anonymous request.

20.1.1. Syntax

```
GET / HTTP/1.1
Host: cname.domain.com

Authorization: AWS {access-key}:{hash-of-header-and-secret}
```

20.1.2. Response Entities

Name	Type	Description
Buckets	Container	Container for list of buckets.
Bucket	Container	Container for bucket information.
Name	String	Bucket name.
CreationDate	Date	UTC time when the bucket was created.
ListAllMyBucketsResult	Container	A container for the result.
Owner	Container	A container for the bucket owner's **ID** and **DisplayName**.
ID	String	The bucket owner's ID.
DisplayName	String	The bucket owner's display name.

CHAPTER 21. BUCKET OPERATIONS

21.1. PUT BUCKET

Creates a new bucket. To create a bucket, you must have a user ID and a valid AWS Access Key ID to authenticate requests. You may not create buckets as an anonymous user.

21.1.1. Constraints

In general, bucket names should follow domain name constraints.

- Bucket names must be unique.
- Bucket names must begin and end with a lowercase letter.
- Bucket names may contain a dash (-).

21.1.2. Syntax

```
PUT /{bucket} HTTP/1.1
Host: cname.domain.com
x-amz-acl: public-read-write

Authorization: AWS {access-key}:{hash-of-header-and-secret}
```

21.1.3. Parameters

Name	Description	Valid Values	Required
`x-amz-acl`	Canned ACLs.	**`private`**, **`public-read`**, **`public-read-write`**, **`authenticated-read`**	No

21.1.4. HTTP Response

If the bucket name is unique, within constraints and unused, the operation will succeed. If a bucket with the same name already exists and the user is the bucket owner, the operation will succeed. If the bucket name is already in use, the operation will fail.

HTTP Status	Status Code	Description
`409`	BucketAlreadyExists	Bucket already exists under different user's ownership.

21.2. DELETE BUCKET

Deletes a bucket. You can reuse bucket names following a successful bucket removal.

21.2.1. Syntax

```
DELETE /{bucket} HTTP/1.1
Host: cname.domain.com

Authorization: AWS {access-key}:{hash-of-header-and-secret}
```

21.2.2. HTTP Response

HTTP Status	Status Code	Description
204	No Content	Bucket removed.

21.3. GET BUCKET

Returns a list of bucket objects.

21.3.1. Syntax

```
GET /{bucket}?max-keys=25 HTTP/1.1
Host: cname.domain.com
```

21.3.2. Parameters

Name	Type	Description
prefix	String	Only returns objects that contain the specified prefix.
delimiter	String	The delimiter between the prefix and the rest of the object name.
marker	String	A beginning index for the list of objects returned.
max-keys	Integer	The maximum number of keys to return. Default is 1000.

21.3.3. HTTP Response

HTTP Status	Status Code	Description
200	OK	Buckets retrieved

21.3.4. Bucket Response Entities

GET /{bucket} returns a container for buckets with the following fields.

Name	Type	Description
ListBucketResult	Entity	The container for the list of objects.
Name	String	The name of the bucket whose contents will be returned.
Prefix	String	A prefix for the object keys.
Marker	String	A beginning index for the list of objects returned.
MaxKeys	Integer	The maximum number of keys returned.
Delimiter	String	If set, objects with the same prefix will appear in the **CommonPrefixes** list.
IsTruncated	Boolean	If **true**, only a subset of the bucket's contents were returned.
CommonPrefixes	Container	If multiple objects contain the same prefix, they will appear in this list.

21.3.5. Object Response Entities

The **ListBucketResult** contains objects, where each object is within a **Contents** container.

Name	Type	Description
`Contents`	Object	A container for the object.
`Key`	String	The object's key.
`LastModified`	Date	The object's last-modified date/time.
`ETag`	String	An MD-5 hash of the object. (entity tag)
`Size`	Integer	The object's size.
`StorageClass`	String	Should always return **`STANDARD`**.

21.4. GET BUCKET LOCATION

Retrieves the bucket's region. The user needs to be the bucket owner to call this. A bucket can be constrained to a region by providing **`LocationConstraint`** during a PUT request.

21.4.1. Syntax

Add the **`location`** subresource to bucket resource as shown below.

```
GET /{bucket}?location HTTP/1.1
Host: cname.domain.com

Authorization: AWS {access-key}:{hash-of-header-and-secret}
```

21.4.2. Response Entities

Name	Type	Description
`LocationConstraint`	String	The region where bucket resides, empty string for defult region

21.5. GET BUCKET ACL

Retrieves the bucket access control list. The user needs to be the bucket owner or to have been granted **`READ_ACP`** permission on the bucket.

21.5.1. Syntax

Add the **acl** subresource to the bucket request as shown below.

```
GET /{bucket}?acl HTTP/1.1
Host: cname.domain.com

Authorization: AWS {access-key}:{hash-of-header-and-secret}
```

21.5.2. Response Entities

Name	Type	Description
AccessControlPolicy	Container	A container for the response.
AccessControlList	Container	A container for the ACL information.
Owner	Container	A container for the bucket owner's **ID** and **DisplayName**.
ID	String	The bucket owner's ID.
DisplayName	String	The bucket owner's display name.
Grant	Container	A container for **Grantee** and **Permission**.
Grantee	Container	A container for the **DisplayName** and **ID** of the user receiving a grant of permission.
Permission	String	The permission given to the **Grantee** bucket.

21.6. PUT BUCKET ACL

Sets an access control to an existing bucket. The user needs to be the bucket owner or to have been granted **WRITE_ACP** permission on the bucket.

21.6.1. Syntax

Add the **acl** subresource to the bucket request as shown below.

```
PUT /{bucket}?acl HTTP/1.1
```

21.6.2. Request Entities

Name	Type	Description
AccessControlPolicy	Container	A container for the request.
AccessControlList	Container	A container for the ACL information.
Owner	Container	A container for the bucket owner's **ID** and **DisplayName**.
ID	String	The bucket owner's ID.
DisplayName	String	The bucket owner's display name.
Grant	Container	A container for **Grantee** and **Permission**.
Grantee	Container	A container for the **DisplayName** and **ID** of the user receiving a grant of permission.
Permission	String	The permission given to the **Grantee** bucket.

21.7. LIST BUCKET OBJECT VERSIONS

Returns a list of metadata about all the version of objects within a bucket. Requires READ access to the bucket.

21.7.1. Syntax

Add the **versions** subresource to the bucket request as shown below.

```
GET /{bucket}?versions HTTP/1.1
Host: cname.domain.com

Authorization: AWS {access-key}:{hash-of-header-and-secret}
```

21.7.2. Parameters

You may specify parameters for **`GET /{bucket}?versions`**, but none of them are required.

Name	Type	Description
`prefix`	String	Returns in-progress uploads whose keys contains the specified prefix.
`delimiter`	String	The delimiter between the prefix and the rest of the object name.
`key-marker`	String	The beginning marker for the list of uploads.
`max-keys`	Integer	The maximum number of in-progress uploads. The default is 1000.
`version-id-marker`	String	Specifies the object version to begin the list.

21.7.3. Response Entities

Name	Type	Description
`KeyMarker`	String	The key marker specified by the **`key-marker`** request parameter (if any).
`NextKeyMarker`	String	The key marker to use in a subsequent request if **`IsTruncated`** is **`true`**.
`NextUploadIdMarker`	String	The upload ID marker to use in a subsequent request if **`IsTruncated`** is **`true`**.

Name	Type	Description
IsTruncated	Boolean	If **true**, only a subset of the bucket's upload contents were returned.
Size	Integer	The size of the uploaded part.
DisplayName	String	The owners's display name.
ID	String	The owners's ID.
Owner	Container	A container for the **ID** and **DisplayName** of the user who owns the object.
StorageClass	String	The method used to store the resulting object. **STANDARD** or **REDUCED_REDUNDANCY**
Version	Container	Container for the version information.
versionId	String	Version ID of an object.
versionIdMarker	String	The last version of the key in a truncated response.

21.8. LIST BUCKET MULTIPART UPLOADS

GET /?uploads returns a list of the current in-progress multipart uploads—i.e., the application initiates a multipart upload, but the service hasn't completed all the uploads yet.

21.8.1. Syntax

```
GET /{bucket}?uploads HTTP/1.1
```

21.8.2. Parameters

You may specify parameters for **`GET /{bucket}?uploads`**, but none of them are required.

Name	Type	Description
`prefix`	String	Returns in-progress uploads whose keys contains the specified prefix.
`delimiter`	String	The delimiter between the prefix and the rest of the object name.
`key-marker`	String	The beginning marker for the list of uploads.
`max-keys`	Integer	The maximum number of in-progress uploads. The default is 1000.
`max-uploads`	Integer	The maximum number of multipart uploads. The range from 1-1000. The default is 1000.
`version-id-marker`	String	Ignored if **`key-marker`** isn't specified. Specifies the **`ID`** of first upload to list in lexicographical order at or following the **`ID`**.

21.8.3. Response Entities

Name	Type	Description
`ListMultipartUploadsResult`	Container	A container for the results.
`ListMultipartUploadsResult.Prefix`	String	The prefix specified by the **`prefix`** request parameter (if any).
`Bucket`	String	The bucket that will receive the bucket contents.
`KeyMarker`	String	The key marker specified by the **`key-marker`** request parameter (if any).

Name	Type	Description
UploadIdMarker	String	The marker specified by the **upload-id-marker** request parameter (if any).
NextKeyMarker	String	The key marker to use in a subsequent request if **IsTruncated** is **true**.
NextUploadIdMarker	String	The upload ID marker to use in a subsequent request if **IsTruncated** is **true**.
MaxUploads	Integer	The max uploads specified by the **max-uploads** request parameter.
Delimiter	String	If set, objects with the same prefix will appear in the **CommonPrefixes** list.
IsTruncated	Boolean	If **true**, only a subset of the bucket's upload contents were returned.
Upload	Container	A container for **Key**, **UploadId**, **InitiatorOwner**, **StorageClass**, and **Initiated** elements.
Key	String	The key of the object once the multipart upload is complete.
UploadId	String	The **ID** that identifies the multipart upload.
Initiator	Container	Contains the **ID** and **DisplayName** of the user who initiated the upload.
DisplayName	String	The initiator's display name.
ID	String	The initiator's ID.

Name	Type	Description
`Owner`	Container	A container for the **`ID`** and **`DisplayName`** of the user who owns the uploaded object.
`StorageClass`	String	The method used to store the resulting object. **`STANDARD`** or **`REDUCED_REDUNDANCY`**
`Initiated`	Date	The date and time the user initiated the upload.
`CommonPrefixes`	Container	If multiple objects contain the same prefix, they will appear in this list.
`CommonPrefixes.Prefix`	String	The substring of the key after the prefix as defined by the **`prefix`** request parameter.

CHAPTER 22. OBJECT OPERATIONS

22.1. PUT OBJECT

Adds an object to a bucket. You must have write permissions on the bucket to perform this operation.

22.1.1. Syntax

```
PUT /{bucket}/{object} HTTP/1.1
```

22.1.2. Request Headers

Name	Description	Valid Values	Required
content-md5	A base64 encoded MD-5 hash of the message.	A string. No defaults or constraints.	No
content-type	A standard MIME type.	Any MIME type. Default: **`binary/octet-stream`**	No
x-amz-meta-<...>	User metadata. Stored with the object.	A string up to 8kb. No defaults.	No
x-amz-acl	A canned ACL.	**`private`**, **`public-read`**, **`public-read-write`**, **`authenticated-read`**	No

22.1.3. Response Headers

Name	Description
x-amz-version-id	Returns the version ID or null.

22.2. COPY OBJECT

To copy an object, use **`PUT`** and specify a destination bucket and the object name.

22.2.1. Syntax

```
PUT /{dest-bucket}/{dest-object} HTTP/1.1
x-amz-copy-source: {source-bucket}/{source-object}
```

22.2.2. Request Headers

Name	Description	Valid Values	Required
x-amz-copy-source	The source bucket name + object name.	{bucket}/{obj}	Yes
x-amz-acl	A canned ACL.	**`private`**, **`public-read`**, **`public-read-write`**, **`authenticated-read`**	No
x-amz-copy-if-modified-since	Copies only if modified since the timestamp.	Timestamp	No
x-amz-copy-if-unmodified-since	Copies only if unmodified since the timestamp.	Timestamp	No
x-amz-copy-if-match	Copies only if object ETag matches ETag.	Entity Tag	No
x-amz-copy-if-none-match	Copies only if object ETag doesn't match.	Entity Tag	No

22.2.3. Response Entities

Name	Type	Description
CopyObjectResult	Container	A container for the response elements.

Name	Type	Description
LastModified	Date	The last modified date of the source object.
Etag	String	The ETag of the new object.

22.3. REMOVE OBJECT

Removes an object. Requires WRITE permission set on the containing bucket.

22.3.1. Syntax

Deletes an object. If object versioning is on, it creates a marker.

```
DELETE /{bucket}/{object} HTTP/1.1
```

To delete an object when versioning is on, you must specify the **`versionId`** subresource and the version of the object to delete.

```
DELETE /{bucket}/{object}?versionId={versionID} HTTP/1.1
```

22.4. GET OBJECT

Retrieves an object from a bucket.

22.4.1. Syntax

```
GET /{bucket}/{object} HTTP/1.1
```

Add the **`versionId`** subresource to retrieve a particular version of the object.

```
GET /{bucket}/{object}?versionId={versionID} HTTP/1.1
```

22.4.2. Request Headers

Name	Description	Valid Values	Required
range	The range of the object to retrieve.	Range: bytes=beginbyte-endbyte	No

Name	Description	Valid Values	Required
if-modified-since	Gets only if modified since the timestamp.	Timestamp	No
if-unmodified-since	Gets only if not modified since the timestamp.	Timestamp	No
if-match	Gets only if object ETag matches ETag.	Entity Tag	No
if-none-match	Gets only if object ETag matches ETag.	Entity Tag	No

22.4.3. Response Headers

Name	Description
Content-Range	Data range, will only be returned if the range header field was specified in the request
x-amz-version-id	Returns the version ID or null.

22.5. GET OBJECT INFO

Returns information about an object. This request will return the same header information as with the Get Object request, but will include the metadata only, not the object data payload.

22.5.1. Syntax

Retrieves the current version of the object.

```
HEAD /{bucket}/{object} HTTP/1.1
```

Add the **`versionId`** subresource to retrieve info for a particular version.

```
HEAD /{bucket}/{object}?versionId={versionID} HTTP/1.1
```

22.5.2. Request Headers

Name	Description	Valid Values	Required
range	The range of the object to retrieve.	Range: bytes=beginbyte-endbyte	No
if-modified-since	Gets only if modified since the timestamp.	Timestamp	No
if-unmodified-since	Gets only if not modified since the timestamp.	Timestamp	No
if-match	Gets only if object ETag matches ETag.	Entity Tag	No
if-none-match	Gets only if object ETag matches ETag.	Entity Tag	No

22.5.3. Response Headers

Name	Description
x-amz-version-id	Returns the version ID or null.

22.6. GET OBJECT ACL

22.6.1. Syntax

Returns the ACL for the current version of the object.

```
GET /{bucket}/{object}?acl HTTP/1.1
```

Add the **`versionId`** subresource to retrieve the ACL for a particular version.

```
GET /{bucket}/{object}versionId={versionID}&acl HTTP/1.1
```

22.6.2. Response Headers

Name	Description
x-amz-version-id	Returns the version ID or null.

22.6.3. Response Entities

Name	Type	Description
`AccessControlPolicy`	Container	A container for the response.
`AccessControlList`	Container	A container for the ACL information.
`Owner`	Container	A container for the object owner's **`ID`** and **`DisplayName`**.
`ID`	String	The object owner's ID.
`DisplayName`	String	The object owner's display name.
`Grant`	Container	A container for **`Grantee`** and **`Permission`**.
`Grantee`	Container	A container for the **`DisplayName`** and **`ID`** of the user receiving a grant of permission.
`Permission`	String	The permission given to the **`Grantee`** object.

22.7. SET OBJECT ACL

Sets an object ACL for the current version of the object.

22.7.1. Syntax

```
PUT /{bucket}/{object}?acl
```

22.7.2. Request Entities

Name	Type	Description
`AccessControlPolicy`	Container	A container for the response.
`AccessControlList`	Container	A container for the ACL information.
`Owner`	Container	A container for the object owner's **`ID`** and **`DisplayName`**.
`ID`	String	The object owner's ID.
`DisplayName`	String	The object owner's display name.
`Grant`	Container	A container for **`Grantee`** and **`Permission`**.
`Grantee`	Container	A container for the **`DisplayName`** and **`ID`** of the user receiving a grant of permission.
`Permission`	String	The permission given to the **`Grantee`** object.

22.8. INITIATE MULTI-PART UPLOAD

Initiates a multi-part upload process. Returns a **`UploadId`**, which you may specify when adding additional parts, listing parts, and completing or abandoning a multi-part upload.

22.8.1. Syntax

```
POST /{bucket}/{object}?uploads
```

22.8.2. Request Headers

Name	Description	Valid Values	Required
content-md5	A base64 encoded MD-5 hash of the message.	A string. No defaults or constraints.	No
content-type	A standard MIME type.	Any MIME type. Default: **binary/octet-stream**	No
x-amz-meta-<...>	User metadata. Stored with the object.	A string up to 8kb. No defaults.	No
x-amz-acl	A canned ACL.	**private**, **public-read**, **public-read-write**, **authenticated-read**	No

22.8.3. Response Entities

Name	Type	Description
InitiatedMultipartUploadsResult	Container	A container for the results.
Bucket	String	The bucket that will receive the object contents.
Key	String	The key specified by the **key** request parameter (if any).
UploadId	String	The ID specified by the **upload-id** request parameter identifying the multipart upload (if any).

22.9. MULTIPART UPLOAD PART

Adds a part to a multi-part upload.

22.9.1. Syntax

Specify the **uploadId** subresource and the upload ID to add a part to a multi-part upload.

```
PUT /{bucket}/{object}?partNumber=&uploadId={upload-id} HTTP/1.1
```

22.9.2. HTTP Response

The following HTTP response may be returned:

HTTP Status	Status Code	Description
404	NoSuchUpload	Specified upload-id does not match any initiated upload on this object

22.10. LIST MULTIPART UPLOAD PARTS

22.10.1. Syntax

Specify the **uploadId** subresource and the upload ID to list the parts of a multi-part upload.

```
GET /{bucket}/{object}?uploadId={upload-id} HTTP/1.1
```

22.10.2. Response Entities

Name	Type	Description
InitiatedMultipartUploadsResult	Container	A container for the results.
Bucket	String	The bucket that will receive the object contents.
Key	String	The key specified by the **key** request parameter (if any).
UploadId	String	The ID specified by the **upload-id** request parameter identifying the multipart upload (if any).
Initiator	Container	Contains the **ID** and **DisplayName** of the user who initiated the upload.

Name	Type	Description
ID	String	The initiator's ID.
DisplayName	String	The initiator's display name.
Owner	Container	A container for the **ID** and **DisplayName** of the user who owns the uploaded object.
StorageClass	String	The method used to store the resulting object. **STANDARD** or **REDUCED_REDUNDANCY**
PartNumberMarker	String	The part marker to use in a subsequent request if **IsTruncated** is **true**. Precedes the list.
NextPartNumberMarker	String	The next part marker to use in a subsequent request if **IsTruncated** is **true**. The end of the list.
MaxParts	Integer	The max parts allowed in the response as specified by the **max-parts** request parameter.
IsTruncated	Boolean	If **true**, only a subset of the object's upload contents were returned.
Part	Container	A container for **Key**, **Part**, **InitiatorOwner**, **StorageClass**, and **Initiated** elements.
PartNumber	Integer	The identification number of the part.
ETag	String	The part's entity tag.

Name	Type	Description
`Size`	Integer	The size of the uploaded part.

22.11. COMPLETE MULTIPART UPLOAD

Assembles uploaded parts and creates a new object, thereby completing a multipart upload.

22.11.1. Syntax

Specify the **`uploadId`** subresource and the upload ID to complete a multi-part upload.

```
POST /{bucket}/{object}?uploadId= HTTP/1.1
```

22.11.2. Request Entities

Name	Type	Description	Required
`CompleteMultipartUpload`	Container	A container consisting of one or more parts.	Yes
`Part`	Container	A container for the **`PartNumber`** and **`ETag`**.	Yes
`PartNumber`	Integer	The identifier of the part.	Yes
`ETag`	String	The part's entity tag.	Yes

22.11.3. Response Entities

Name	Type	Description
`CompleteMultipartUploadResult`	Container	A container for the response.
`Location`	URI	The resource identifier (path) of the new object.

Name	Type	Description
`Bucket`	String	The name of the bucket that contains the new object.
`Key`	String	The object's key.
`ETag`	String	The entity tag of the new object.

22.12. ABORT MULTIPART UPLOAD

Aborts a multipart upload.

22.12.1. Syntax

Specify the **`uploadId`** subresource and the upload ID to abort a multi-part upload.

```
DELETE /{bucket}/{object}?uploadId={upload-id} HTTP/1.1
```

PART V. CEPH OBJECT GATEWAY SWIFT API

Ceph supports a RESTful API that is compatible with the basic data access model of the Swift API.

CHAPTER 23. API

CHAPTER 24. FEATURES SUPPORT

The following table describes the support status for current Swift functional features:

Feature	Status	Remarks
Authentication	Supported	
Get Account Metadata	Supported	No custom metadata
Swift ACLs	Supported	Supports a subset of Swift ACLs
List Containers	Supported	
Delete Container	Supported	
Create Container	Supported	
Get Container Metadata	Supported	
Update Container Metadata	Supported	
Delete Container Metadata	Supported	
List Objects	Supported	
Static Website	Not Supported	
Create/Update an Object	Supported	
Create Large Object	Supported	
Delete Object	Supported	

Feature	Status	Remarks
Get Object	Supported	
Copy Object	Supported	
Get Object Metadata	Supported	
Add/Update Object Metadata	Supported	
Temp URL Operations	Supported	
Expiring Objects	Not Supported	
Object Versioning	Not Supported	
CORS	Not Supported	

CHAPTER 25. AUTHENTICATION

Swift API requests that require authentication must contain an **X-Storage-Token** authentication token in the request header. The token may be retrieved from Ceph Object Gateway, or from another authenticator. To obtain a token from Ceph Object Gateway, you must create a user. For example:

```
sudo radosgw-admin user create --uid="{username}" --display-name="
{Display Name}"
```

25.1. AUTH GET

To authenticate a user, make a request containing an **X-Auth-User** and a **X-Auth-Key** in the header.

25.1.1. Syntax

```
GET /auth HTTP/1.1
Host: swift.radosgwhost.com
X-Auth-User: johndoe
X-Auth-Key: R7UUOLFDI2ZI9PRCQ53K
```

25.1.2. Request Headers

Name	Description	Type	Required
X-Auth-User	The key Ceph Object Gateway username to authenticate.	String	Yes
X-Auth-Key	The key associated to a Ceph Object Gateway username.	String	Yes

25.1.3. Response Headers

The response from the server should include an **X-Auth-Token** value. The response may also contain a **X-Storage-Url** that provides the **{api version}/{account}** prefix that is specified in other requests throughout the API documentation.

Name	Description	Type
X-Storage-Token	The authorization token for the **X-Auth-User** specified in the request.	String

Name	Description	Type
X-Storage-Url	The URL and **{api version}/{account}** path for the user.	String

A typical response looks like this:

```
HTTP/1.1 204 No Content
Date: Mon, 16 Jul 2012 11:05:33 GMT
Server: swift
X-Storage-Url: https://swift.radosgwhost.com/v1/ACCT-12345
X-Auth-Token: UOlCCC8TahFKlWuv9DB09TWHF0nDjpPElha0kAa
Content-Length: 0
Content-Type: text/plain; charset=UTF-8
```

CHAPTER 26. SERVICE OPERATIONS

To retrieve data about our Swift-compatible service, you may execute **GET** requests using the **X-Storage-Url** value retrieved during authentication.

26.1. LIST CONTAINERS

A **GET** request that specifies the API version and the account will return a list of containers for a particular user account. Since the request returns a particular user's containers, the request requires an authentication token. The request cannot be made anonymously.

26.1.1. Syntax

```
GET /{api version}/{account} HTTP/1.1
Host: {fqdn}
X-Auth-Token: {auth-token}
```

26.1.2. Request Parameters

Name	Description	Type	Required	Valid Values
limit	Limits the number of results to the specified value.	Integer	No	N/A
format	Defines the format of the result.	String	No	**json** or **xml**
marker	Returns a list of results greater than the marker value.	String	No	N/A

26.1.3. Response Entities

The response contains a list of containers, or returns with an HTTP 204 response code

Name	Description	Type
account	A list for account information.	Container

Name	Description	Type
container	The list of containers.	Container
name	The name of a container.	String
bytes	The size of the container.	Integer

CHAPTER 27. CONTAINER OPERATIONS

A container is a mechanism for storing data objects. An account may have many containers, but container names must be unique. This API enables a client to create a container, set access controls and metadata, retrieve a container's contents, and delete a container. Since this API makes requests related to information in a particular user's account, all requests in this API must be authenticated unless a container's access control is deliberately made publicly accessible (i.e., allows anonymous requests).

Note

The Amazon S3 API uses the term *bucket* to describe a data container. When you hear someone refer to a *bucket* within the Swift API, the term *bucket* may be construed as the equivalent of the term *container.*

One facet of object storage is that it does not support hierarchical paths or directories. Instead, it supports one level consisting of one or more containers, where each container may have objects. The RADOS Gateway's Swift-compatible API supports the notion of *pseudo-hierarchical containers,* which is a means of using object naming to emulate a container (or directory) hierarchy without actually implementing one in the storage system. You may name objects with pseudo-hierarchical names (e.g., photos/buildings/empire-state.jpg), but container names cannot contain a forward slash (`/`) character.

27.1. CREATE A CONTAINER

To create a new container, make a **`PUT`** request with the API version, account, and the name of the new container. The container name must be unique, must not contain a forward-slash (/) character, and should be less than 256 bytes. You may include access control headers and metadata headers in the request. You may also include a storage policy identifying a key for a set of placement pools (e.g., execute **`radosgw-admin zone get`** to see a list of available keys under **`placement_pools`**). A storage policy enables you to specify a special set of pools for the container (e.g., SSD-based storage). The operation is idempotent; that is, if you make a request to create a container that already exists, it will return with a HTTP 202 return code, but will not create another container.

27.1.1. Syntax

```
PUT /{api version}/{account}/{container} HTTP/1.1
Host: {fqdn}
X-Auth-Token: {auth-token}
X-Container-Read: {comma-separated-uids}
X-Container-Write: {comma-separated-uids}
X-Container-Meta-{key}: {value}
X-Storage-Policy: {placement-pools-key}
```

27.1.2. Headers

Name	Description	Type	Required
`X-Container-Read`	The user IDs with read permissions for the container.	Comma-separated string values of user IDs.	No
`X-Container-Write`	The user IDs with write permissions for the container.	Comma-separated string values of user IDs.	No
`X-Container-Meta-{key}`	A user-defined meta data key that takes an arbitrary string value.	String	No
`X-Storage-Policy`	The key that identifies the storage policy under **`placement_pools`** for the Ceph Object Gateway. Execute **`radosgw-admin zone get`** for available keys.	String	No

27.1.3. HTTP Response

If a container with the same name already exists, and the user is the container owner then the operation will succeed. Otherwise the operation will fail.

Name	Description	Status Code
`409`	The container already exists under a different user's ownership.	**`BucketAlreadyExists`**

27.2. LIST A CONTAINER'S OBJECTS

To list the objects within a container, make a **GET** request with the with the API version, account, and the name of the container. You can specify query parameters to filter the full list, or leave out the parameters to return a list of the first 10,000 object names stored in the container.

27.2.1. Syntax

```
GET /{api version}/{container} HTTP/1.1
 Host: {fqdn}
 X-Auth-Token: {auth-token}
```

27.2.2. Parameters

Name	Description	Type	Valid Values	Required
format	Defines the format of the result.	String	**json** or **xml**	No
prefix	Limits the result set to objects beginning with the specified prefix.	String	N/A	No
marker	Returns a list of results greater than the marker value.	String	N/A	No
limit	Limits the number of results to the specified value.	Integer	0 - 10,000	No
delimiter	The delimiter between the prefix and the rest of the object name.	String	N/A	No
path	The pseudo-hierarchical path of the objects.	String	N/A	No

27.2.3. Response Entities

Name	Description	Type

Name	Description	Type
container	The container.	Container
object	An object within the container.	Container
name	The name of an object within the container.	String
hash	A hash code of the object's contents.	String
last_modified	The last time the object's contents were modified.	Date
content_type	The type of content within the object.	String

27.3. UPDATE A CONTAINER'S ACLS

When a user creates a container, the user has read and write access to the container by default. To allow other users to read a container's contents or write to a container, you must specifically enable the user. You may also specify ***** in the **X-Container-Read** or **X-Container-Write** settings, which effectively enables all users to either read from or write to the container. Setting ***** makes the container public. That is it enables anonymous users to either read from or write to the container.

27.3.1. Syntax

```
POST /{api version}/{account}/{container} HTTP/1.1
Host: {fqdn}
 X-Auth-Token: {auth-token}
 X-Container-Read: *
 X-Container-Write: {uid1}, {uid2}, {uid3}
```

27.3.2. Request Headers

Name	Description	Type	Required

Name	Description	Type	Required
X-Container-Read	The user IDs with read permissions for the container.	Comma-separated string values of user IDs.	No
X-Container-Write	The user IDs with write permissions for the container.	Comma-separated string values of user IDs.	No

27.4. ADD/UPDATE CONTAINER METADATA

To add metadata to a container, make a **POST** request with the API version, account, and container name. You must have write permissions on the container to add or update metadata.

27.4.1. Syntax

```
POST /{api version}/{account}/{container} HTTP/1.1
Host: {fqdn}
 X-Auth-Token: {auth-token}
 X-Container-Meta-Color: red
 X-Container-Meta-Taste: salty
```

27.4.2. Request Headers

Name	Description	Type	Required
X-Container-Meta-{key}	A user-defined meta data key that takes an arbitrary string value.	String	No

27.5. DELETE A CONTAINER

To delete a container, make a **DELETE** request with the API version, account, and the name of the container. The container must be empty. If you'd like to check if the container is empty, execute a **HEAD** request against the container. Once you've successfully removed the container, you'll be able to reuse the container name.

27.5.1. Syntax

```
DELETE /{api version}/{account}/{container} HTTP/1.1
Host: {fqdn}
X-Auth-Token: {auth-token}
```

27.5.2. HTTP Response

Name	Description	Status Code
204	The container was removed.	**NoContent**

CHAPTER 28. OBJECT OPERATIONS

An object is a container for storing data and metadata. A container may have many objects, but the object names must be unique. This API enables a client to create an object, set access controls and metadata, retrieve an object's data and metadata, and delete an object. Since this API makes requests related to information in a particular user's account, all requests in this API must be authenticated unless the container or object's access control is deliberately made publicly accessible (i.e., allows anonymous requests).

28.1. CREATE/UPDATE AN OBJECT

To create a new object, make a **PUT** request with the API version, account, container name and the name of the new object. You must have write permission on the container to create or update an object. The object name must be unique within the container. The **PUT** request is not idempotent, so if you do not use a unique name, the request will update the object. However, you may use pseudo-hierarchical syntax in your object name to distinguish it from another object of the same name if it is under a different pseudo-hierarchical directory. You may include access control headers and metadata headers in the request.

28.1.1. Syntax

```
PUT /{api version}/{account}/{container}/{object} HTTP/1.1
 Host: {fqdn}
 X-Auth-Token: {auth-token}
```

28.1.2. Request Headers

Name	Description	Type	Required	Valid Values
ETag	An MD5 hash of the object's contents. Recommended.	String	No	N/A
Content-Type	The type of content the object contains.	String	No	N/A
Transfer-Encoding	Indicates whether the object is part of a larger aggregate object.	String	No	**chunked**

28.2. COPY AN OBJECT

Copying an object allows you to make a server-side copy of an object, so that you don't have to download it and upload it under another container/name. To copy the contents of one object to another object, you may make either a **PUT** request or a **COPY** request with the API version, account, and the container name. For a **PUT** request, use the destination container and object name in the request, and the source container and object in the request header. For a **Copy** request, use

the source container and object in the request, and the destination container and object in the request header. You must have write permission on the container to copy an object. The destination object name must be unique within the container. The request is not idempotent, so if you do not use a unique name, the request will update the destination object. However, you may use pseudo-hierarchical syntax in your object name to distinguish the destination object from the source object of the same name if it is under a different pseudo-hierarchical directory. You may include access control headers and metadata headers in the request.

28.2.1. Syntax

```
PUT /{api version}/{account}/{dest-container}/{dest-object} HTTP/1.1
X-Copy-From: {source-container}/{source-object}
Host: {fqdn}
X-Auth-Token: {auth-token}
```

or alternatively:

```
COPY /{api version}/{account}/{source-container}/{source-object}
HTTP/1.1
Destination: {dest-container}/{dest-object}
```

28.2.2. Request Headers

Name	Description	Type	Required
`X-Copy-From`	Used with a **`PUT`** request to define the source container/object path.	String	Yes, if using **`PUT`**
`Destination`	Used with a **`COPY`** request to define the destination container/object path.	String	Yes, if using **`COPY`**
`If-Modified-Since`	Only copies if modified since the date/time of the source object's **`last_modified`** attribute.	Date	No
`If-Unmodified-Since`	Only copies if not modified since the date/time of the source object's **`last_modified`** attribute.	Date	No
`Copy-If-Match`	Copies only if the ETag in the request matches the source object's ETag.	ETag.	No

Name	Description	Type	Required
`Copy-If-None-Match`	Copies only if the ETag in the request does not match the source object's ETag.	ETag.	No

28.3. DELETE AN OBJECT

To delete an object, make a **`DELETE`** request with the API version, account, container and object name. You must have write permissions on the container to delete an object within it. Once you've successfully deleted the object, you'll be able to reuse the object name.

28.3.1. Syntax

```
DELETE /{api version}/{account}/{container}/{object} HTTP/1.1
Host: {fqdn}
X-Auth-Token: {auth-token}
```

28.4. GET AN OBJECT

To retrieve an object, make a **`GET`** request with the API version, account, container and object name. You must have read permissions on the container to retrieve an object within it.

28.4.1. Syntax

```
GET /{api version}/{account}/{container}/{object} HTTP/1.1
Host: {fqdn}
X-Auth-Token: {auth-token}
```

28.4.2. Request Headers

Name	Description	Type	Required
`range`	To retrieve a subset of an object's contents, you may specify a byte range.	Date	No
`If-Modified-Since`	Only copies if modified since the date/time of the source object's **`last_modified`** attribute.	Date	No

Name	Description	Type	Required
`If-Unmodified-Since`	Only copies if not modified since the date/time of the source object's **`last_modified`** attribute.	Date	No
`Copy-If-Match`	Copies only if the ETag in the request matches the source object's ETag.	ETag.	No
`Copy-If-None-Match`	Copies only if the ETag in the request does not match the source object's ETag.	ETag.	No

28.4.3. Response Headers

Name	Description
`Content-Range`	The range of the subset of object contents. Returned only if the range header field was specified in the request.

28.5. GET OBJECT METADATA

To retrieve an object's metadata, make a **`HEAD`** request with the API version, account, container and object name. You must have read permissions on the container to retrieve metadata from an object within the container. This request returns the same header information as the request for the object itself, but it does not return the object's data.

28.5.1. Syntax

```
HEAD /{api version}/{account}/{container}/{object} HTTP/1.1
Host: {fqdn}
X-Auth-Token: {auth-token}
```

28.6. ADD/UPDATE OBJECT METADATA

To add metadata to an object, make a **`POST`** request with the API version, account, container and object name. You must have write permissions on the parent container to add or update metadata.

28.6.1. Syntax

```
POST /{api version}/{account}/{container}/{object} HTTP/1.1
Host: {fqdn}
X-Auth-Token: {auth-token}
```

28.6.2. Request Headers

Name	Description	Type	Required
`X-Object-Meta-{key}`	A user-defined meta data key that takes an arbitrary string value.	String	No

CHAPTER 29. TEMP URL OPERATIONS

To allow temporary access (for eg for GET requests) to objects without the need to share credentials, temp url functionality is supported by swift endpoint of radosgw. For this functionality, initially the value of X-Account-Meta-Temp-URL-Key and optionally X-Account-Meta-Temp-URL-Key-2 should be set. The Temp URL functionality relies on a HMAC-SHA1 signature against these secret keys.

29.1. POST TEMP-URL KEYS

A **POST** request to the swift account with the required Key will set the secret temp url key for the account against which temporary url access can be provided to accounts. Up to two keys are supported, and signatures are checked against both the keys, if present, so that keys can be rotated without invalidating the temporary urls.

29.1.1. Syntax

```
POST /{api version}/{account} HTTP/1.1
Host: {fqdn}
X-Auth-Token: {auth-token}
```

29.1.2. Request Headers

Name	Description	Type	Required
X-Account-Meta-Temp-URL-Key	A user-defined key that takes an arbitrary string value.	String	Yes
X-Account-Meta-Temp-URL-Key-2	A user-defined key that takes an arbitrary string value.	String	No

29.2. GET TEMP-URL OBJECTS

Temporary URL uses a cryptographic HMAC-SHA1 signature, which includes the following elements:

1. The value of the Request method, "GET" for instance
2. The expiry time, in format of seconds since the epoch, ie Unix time
3. The request path starting from "v1" onwards

The above items are normalized with newlines appended between them, and a HMAC is generated using the SHA-1 hashing algorithm against one of the Temp URL Keys posted earlier.

A sample python script to demonstrate the above is given below:

```
import hmac
from hashlib import sha1
from time import time

method = 'GET'
host = 'https://objectstore.example.com'
duration_in_seconds = 300  # Duration for which the url is valid
expires = int(time() + duration_in_seconds)
path = '/v1/your-bucket/your-object'
key = 'secret'
hmac_body = '%s\n%s\n%s' % (method, expires, path)
hmac_body = hmac.new(key, hmac_body, sha1).hexdigest()
sig = hmac.new(key, hmac_body, sha1).hexdigest()
rest_uri = "{host}{path}?temp_url_sig={sig}&temp_url_expires=
{expires}".format(
     host=host, path=path, sig=sig, expires=expires)
print rest_uri

# Example Output
# https://objectstore.example.com/v1/your-bucket/your-object?
temp_url_sig=ff4657876227fc6025f04fcf1e82818266d022c6&temp_url_expires=
1423200992
```

PART VI. CEPH FEDERATED OBJECT GATEWAY FOR RHEL X86_64

A **`federated`** Ceph Object Gateway configuration implies that you are running a Ceph Object Storage service in a geographically distributed manner for fault tolerance and failover.

In Red Hat Ceph Storage v1.3, you may configure each Ceph Object Gateway on a **RHEL 7** node to participate in a federated architecture, with multiple regions, and with multiple zones for a region.

- **Region**: A region represents a **`logical`** geographic area and contains one or more zones. A cluster with multiple regions must specify a master region.
- **Zone**: A zone is a **`logical`** grouping of one or more Ceph Object Gateway instance(s). A region has a master zone that processes client requests.

Important

Only write objects to the master zone in a region. You may read objects from secondary zones. Currently, the Gateway does not prevent you from writing to a secondary zone, but **DON'T DO IT**.

CHAPTER 30. BACKGROUND

When you deploy a Ceph Object Store service that spans geographical locales, configuring Ceph Object Gateway regions and metadata synchronization agents enables the service to maintain a global namespace, even though Ceph Object Gateway instances run in different geographic locales and potentially on different Ceph Storage Clusters. When you separate one or more Ceph Object Gateway instances within a region into separate logical containers to maintain an extra copy (or copies) of the data, configuring Ceph Object Gateway zones and data synchronization agents enables the service to maintain one or more copies of the master zone's data. Extra copies of the data are important for failover, backup and disaster recovery.

You may deploy a single Ceph Storage Cluster with a federated architecture if you have low latency network connections (this isn't recommended). You may also deploy one Ceph Storage Cluster per region with a separate set of pools for each zone (typical). You may also deploy a separate Ceph Storage Cluster for each zone if your requirements and resources warrant this level of redundancy.

CHAPTER 31. CONFIGURATION

In the following sections, we will demonstrate how to configure a federated cluster in two logical steps:

- **Configure a Master Region:** This section of the guide describes how to set up a region with multiple zones, and how to synchronize data between the master zone and the secondary zone(s) within the master region.
- **Configure a Secondary Region:** This section of the guide describes how to repeat the section on setting up a master region and multiple zones so that you have two regions with intra-zone synchronization in each region. Finally, you will learn how to set up a metadata synchronization agent so that you can maintain a global namespace for the regions in your cluster.

31.1. CONFIGURE A MASTER REGION

This section provides an exemplary procedure for setting up a region, and two zones within the region. The cluster will comprise two gateway daemon instances, one per zone. This region will serve as the master region.

31.1.1. Naming for the Master Region

Before configuring the cluster, defining region, zone and instance names will help you manage your cluster. Let's assume the region represents the United States, and we refer to it by its standard abbreviation.

- United States: **`us`**

Let's assume the zones represent the Eastern and Western United States. For continuity, our naming convention will use **`{region name}-{zone name}`** format, but you can use any naming convention you prefer.

- United States, East Region: **`us-east`**
- United States, West Region: **`us-west`**

Finally, let's assume that zones may have more than one Ceph Object Gateway instance per zone. For continuity, our naming convention will use **`{region name}-{zone name}-{instance}`** format, but you can use any naming convention you prefer.

- United States Region, Master Zone, Instance 1: **`us-east-1`**
- United States Region, Secondary Zone, Instance 1: **`us-west-1`**

31.1.2. Installation

For Red Hat Ceph Storage v1.3, Red Hat supports the Ceph Object Gateway running on Civetweb (embedded into the **`ceph-radosgw`** daemon) instead of Apache and FastCGI. Using Civetweb simplifies the installation and configuration.

Note

To run the Ceph Object Gateway service, you should have a running Ceph storage cluster, and Ceph Object Gateway nodes should have access to the public network.

Note

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

We will configure one Ceph Object Gateway instance per node and for ease of understanding, we will use the Ceph Object Gateway instance names **`us-east-1`**, **`us-west-1`** mentioned in Naming for the Master Region as the short hostnames (hostname -s) for Ceph Object Gateway nodes of **`master`** region and the Ceph Object Gateway instance names **`eu-east-1`** and **`eu-west-1`** mentioned in Naming for the Secondary Region as the short hostnames (hostname -s) for Ceph Object Gateway nodes of **`secondary`** region.

Note

You will have a total of four gateway instances after you create the master region and the secondary region.

31.1.3. Execute the Pre-Installation Procedure

Refer to the Red Hat Ceph Storage Installation Guide for RHEL, and execute the pre-installation procedures on your Ceph Object Gateway nodes of **`master`** region i.e, **`us-east-1`** and **`us-west-1`**. Specifically, you should enable password-less **`ssh`** to these nodes from **`admin node`**, disable **`requiretty`**, set SELinux to **`Permissive`** and set up a Ceph Deploy user with password-less **`sudo`** access on these nodes. For Ceph Object Gateways, you will need to open the port that Civetweb will use in production.

Note

Civetweb runs on port **`7480`** by default.

Note

You will have to repeat the same procedure for Ceph Object Gateway nodes of **`secondary`** region, i.e, **`eu-east-1`** and **`eu-west-1`**.

31.1.4. Enable Ceph Client Repository

Red Hat packages the Ceph Object Gateway in the **`rhel-7-server-rhceph-1.3-tools-rpms`** repository. To ensure you are using the same version of Ceph as your storage cluster, execute the following to enable the repository on your Ceph Object Gateway nodes of **`master`** region i.e, **`us-east-1`** and **`us-west-1`**:

```
sudo subscription-manager repos --enable=rhel-7-server-rhceph-1.3-
tools-rpms
```

Note

You will have to repeat the same procedure for Ceph Object Gateway nodes of **secondary** region, i.e, **eu-east-1** and **eu-west-1**.

31.1.5. Install Ceph Object Gateway

From the working directory of your **admin node**, install the Ceph Object Gateway package on your Ceph Object Gateway nodes of **master** region i.e, **us-east-1** and **us-west-1**:

```
ceph-deploy install --rgw us-east-1 us-west-1
```

The **ceph-common** package is a dependency, so **ceph-deploy** will install this too along with **ceph-radosgw** package. The **ceph** CLI tools are intended for administrators. To make your Ceph Object Gateway nodes administrator nodes, execute the following from the working directory of your **admin node** (e.g. **ceph-config**):

```
ceph-deploy admin us-east-1 us-west-1
```

Note

You will have to repeat the same procedure for Ceph Object Gateway nodes of **secondary** region, i.e, **eu-east-1** and **eu-west-1**.

31.1.6. Install Ceph Object Gateway Synchronization Agent

The Ceph Object Gateway synchronization agent **radosgw-agent** is required for data synchronization between zones within a region and metadata synchronization between two regions.

To install the Ceph Object Gateway synchronization agent, execute the following on your Ceph Object Gateway nodes of **master** region i.e, **us-east-1** and **us-west-1**:

```
sudo yum install radosgw-agent
```

Note

You will have to repeat the same procedure for Ceph Object Gateway nodes of **secondary** region, i.e, **eu-east-1** and **eu-west-1**.

31.1.7. Create Gateway Instances

From the working directory of your **admin node**, create instances of Ceph Object Gateway on your Ceph Object Gateway nodes of **master** region i.e, **us-east-1** and **us-west-1**:

```
ceph-deploy rgw create us-east-1 us-west-1
```

The above command creates gateway username, keyring, data directory for each Ceph Object Gateway node and places the keyring in the newly created data directory **`/var/lib/ceph/radosgw/{rgw-intance}`**. The command also provides write capability to the keyring so that each instance can create pools automatically.

Once a gateway instance is running, you should be able to access it on port **`7480`** with an unauthenticated request like this:

```
http://us-east-1:7480
```

If the gateway instance is working properly, you should receive a response like this:

```
<?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>
```

If at any point you run into trouble and you want to start over, execute the following to purge the configuration:

```
ceph-deploy purge <gateway-node1> [<gateway-node2>]
ceph-deploy purgedata <gateway-node1> [<gateway-node2>]
```

If you execute **`purge`**, you must re-install Ceph.

Note

You will have to repeat the same procedure for Ceph Object Gateway nodes of **`secondary`** region, i.e, **`eu-east-1`** and **`eu-west-1`**.

31.1.8. Copy Gateway Keyring to Ceph Configuration Directory

As mentioned in the previous section, **`ceph-deploy`** places the generated gateway keyring in **`/var/lib/ceph/radosgw/{rgw-instance}`** directory for each Ceph Object Gateway node. **`radosgw-admin`** utility expects the keyring to be in Ceph configuration directory i.e, **`/etc/ceph`** of the Ceph Object Gateway node and will throw error when executed if the keyring isn't there. So, to ensure **`radosgw-admin`** utility runs properly, ssh to each Ceph Object Gateway node and copy the gateway keyring to **`/etc/ceph/`** directory.

For example:

```
ssh us-east-1
sudo cp /var/lib/ceph/radosgw/ceph-rgw.us-east-1/keyring /etc/ceph
exit
```

```
ssh us-west-1
sudo cp /var/lib/ceph/radosgw/ceph-rgw.us-west-1/keyring /etc/ceph
exit
```

Note

You will have to repeat the same procedure for Ceph Object Gateway nodes of **secondary** region, i.e, **eu-east-1** and **eu-west-1**.

31.1.9. Add Gateway Configuration to Ceph

Modify your Ceph configuration file in the **/etc/ceph** directory of your **admin node** to add an entry for each Ceph Object Gateway node. Add sections entitled **[client.rgw.<gateway-node>]**, replacing **<gateway-node>** with the short host names of your Ceph Object Gateway nodes (i.e., **hostname -s**) of **master** region i.e, **us-east-1** and **us-west-1**. Also, Civetweb runs on port **7480** by default. You can change the default port (e.g. to port **80**).

Add sections like this after the **[global]** section:

```
[client.rgw.us-east-1]
rgw region = us
rgw region root pool = .us.rgw.root
rgw zone = us-east
rgw zone root pool = .us-east.rgw.root
rgw dns name = {hostname}
rgw_frontends = "civetweb port=80"
host = {hostname}

[client.rgw.us-west-1]
rgw region = us
rgw region root pool = .us.rgw.root
rgw zone = us-west
rgw zone root pool = .us-west.rgw.root
rgw dns name = {hostname}
rgw_frontends = "civetweb port=80"
host = {hostname}
```

Here, {hostname} is the short hostname (output of command **hostname -s**) of the Ceph Object Gateway nodes of **master** region i.e, **us-east-1** and **us-west-1**.

Note

Ensure that you leave no whitespace between **port=<port-number>** in the **rgw_frontends** key/value pair. **[client.rgw.us-east-1]** and **[client.rgw.us-west-1]** headings identify these portions of the Ceph configuration file as configuring Ceph Storage Cluster clients where each client is a Ceph Object Gateway (i.e.,**rgw**), and the names of the instances are **us-east-1** and **us-west-1**.

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.

Note

You will have to add sections for the secondary region as well with **`eu-east-1`** and **`eu-west-1`** instances when you configure the **`secondary`** region and you have to repeat this entire procedure.

31.1.10. Distribute Updated Ceph Configuration File

Pull the updated configuration file from **`/etc/ceph`** directory to the local directory of your cluster (e.g. ceph-config directory) and push it to your Ceph Object Gateway nodes and other Ceph nodes.

```
ceph-deploy --overwrite-conf config pull <admin-node>
ceph-deploy --overwrite-conf config push [<gateway-node> ...] [<other-
nodes>]
```

To make the new setting take effect, restart the Ceph Object Gateway on each Ceph Object Gateway node:

```
sudo systemctl restart ceph-radosgw
sudo chkconfig ceph-radosgw on
```

Note

You will have to repeat the same procedure for Ceph Object Gateway nodes of **`secondary`** region, i.e, **`eu-east-1`** and **`eu-west-1`**.

31.1.11. Adjust Firewall Settings

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

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

Note

You will have to repeat the same procedure for Ceph Object Gateway nodes of **`secondary`** region, i.e, **`eu-east-1`** and **`eu-west-1`**.

31.1.12. 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 back to the description for installing a Ceph Object Gateway with **ceph-deploy**, notice that the configuration file has one setting **rgw_frontends** (and that's assuming you elected to change the default port). The **ceph-deploy** utility generates the data directory and the keyring for you—placing the keyring in **/var/lib/ceph/radosgw/{rgw-intance}**. The daemon looks in default locations, whereas you may have specified different settings in your Ceph configuration file. 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.radosgw.gateway]
host = {hostname}
keyring = /etc/ceph/ceph.client.radosgw.keyring
rgw socket path = ""
log file = /var/log/radosgw/client.radosgw.gateway.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.radosgw.gateway]
host = {hostname}
keyring = /etc/ceph/ceph.client.radosgw.keyring
log file = /var/log/radosgw/client.radosgw.gateway.log
rgw_frontends = civetweb port=80
```

Finally, restart the Ceph Object Gateway.

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

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

Note

You have to repeat the same procedure for Ceph Object Gateway nodes of **secondary** region, i.e, **eu-east-1** and **eu-west-1** if they **Apache/FastCGI** configured in them.

31.1.13. Add Debugging (if needed)

Once you finish the setup procedure, if you encounter issues with your configuration, you can add debugging to the **[global]** section of your Ceph configuration file in **/etc/ceph** directory of **admin node**, pull it to the local cluster directory, push it to the gateway nodes 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
```

31.1.14. Add Wildcard to DNS

To use Ceph with S3-style subdomains (e.g., bucket-name.domain-name.com), you need to add a wildcard to the DNS record of the DNS server you use with the **ceph-radosgw** daemon.

The address of the DNS must also be specified in the Ceph configuration file with the **rgw dns name = {hostname}** setting. This setting has been mentioned in Add Gateway Configuration to Ceph

This step is important and you will find it's use in Multi Site Data Replication and Inter-Region Metadata Replication.

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=/.us-west-1/192.168.122.75
```

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

```
$TTL    604800
@       IN      SOA     us-west-1. root.us-west-1. (
                              2         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      us-west-1.
@       IN      A       192.168.122.113
*       IN      CNAME   @
```

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

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

For example:

```
ping mybucket.us-west-1
```

31.1.15. Create a Region

31.1.15. Create a Region

A region is designated using a **.json** file which is used to configure a **master** region or a **secondary** region in a **Ceph Object Gateway node** with specifications for **master** zone and **secondary** zone in the region. The region infile is created on a single **Ceph Object Gateway node** and that **Ceph Object Gateway node** is also configured as a **master** zone within that region using an **endpoint** address. Another **Ceph Object Gateway node** is added as a **secondary** zone in the infile using it's **endpoint** address. In this section we will create a region infile for **master** region. Same steps will be repeated for creating a region infile for **secondary** region in Configuring a Secondary Region, only the name for the **.json** region file will be different.

To maintain consistency, we will create the **region infile** in **admin node** of the cluster and copy it to the desired **Ceph Object Gateway nodes**.

Execute the following steps for creating and setting up a **region**:

1. Configure a region infile called **us.json** for the **us** master region in the working directory of **admin node** (e.g. **ceph-config** directory):

   ```
   cd ceph-config
   vi us.json
   ```

   Copy the contents of the following example to the new file **us.json**. Set **is_master** to **true**. Replace **{fqdn}** with the fully-qualified domain name of the endpoint. It will specify a master zone as **us-east** and list it in the **zones** list along with the **us-west** zone:

   ```
   { "name": "us",
     "api_name": "us",
     "is_master": "true",
     "endpoints": [
           "http:\/\/{fqdn}:80\/"],
     "master_zone": "us-east",
     "zones": [
           { "name": "us-east",
             "endpoints": [
                   "http:\/\/{fqdn}:80\/"],
             "log_meta": "true",
             "log_data": "true"},
           { "name": "us-west",
             "endpoints": [
                   "http:\/\/{fqdn}:80\/"],
             "log_meta": "true",
             "log_data": "true"}],
     "placement_targets": [
      {
        "name": "default-placement",
        "tags": []
      }
     ],
     "default_placement": "default-placement"}
   ```

2. Copy the **us.json** file to the **/etc/ceph** directory of Ceph Object Gateway node **us-east-1**:

   ```
   scp us.json ceph@us-east-1:~
   ```

```
ssh us-east-1
sudo mv us.json /etc/ceph/
exit
```

3. Copy the **us.json** file to the **/etc/ceph** directory of Ceph Object Gateway node **eu-east-1** of **secondary** region as well for creating the **master** region in **secondary region** too. You will find it's use in Configuring a Secondary Region.

   Execute the following:

   ```
   scp us.json ceph@eu-east-1:~
   ssh eu-east-1
   sudo mv us.json /etc/ceph/
   exit
   ```

4. Create the **us** region in **us-east-1** Ceph Object Gateway node using the **us.json** infile:

   ```
   ssh us-east-1
   sudo radosgw-admin region set --infile /etc/ceph/us.json --name
   client.rgw.us-east-1
   ```

5. Delete the default region (if it exists):

   ```
   sudo rados -p .us.rgw.root rm region_info.default
   ```

6. Set the **us** region as the default region:

   ```
   sudo radosgw-admin region default --rgw-region=us --name
   client.rgw.us-east-1
   ```

   Only one region can be the default region for a cluster.

7. Update the region map:

   ```
   sudo radosgw-admin regionmap update --name client.rgw.us-east-1
   ```

If you use different Ceph Storage Cluster instances for regions, you should repeat steps 4, 6 and 7 in by executing them with **--name client.rgw-us-west-1**. You may also export the region map from the initial gateway instance and import it followed by updating the region map.

Note

When you use this procedure to configure the secondary region, replace **us** with **eu**. You will have a total of two regions **after** you create the master region and the secondary region.

31.1.16. Create Zones

Like a region, a zone is also divided into two categories, **master** and **secondary**. A master zone is basically where you write data, and a secondary zone is where you failover if things go wrong. In this section we will create a zone infile for **master** zone in **us-east-1** Ceph Object Gateway node and a zone infile for **secondary** zone in **us-west-1** Ceph Object Gateway node.

Again, to maintain consistency create the files in **admin node** of the cluster and copy them to the desired **Ceph Object Gateway nodes** for **master** and **secondary** zones.

Execute the following steps to create and setup zones:

1. Configure a zone infile called **us-east.json** for the **us-east** zone in the working directory of **admin node** (e.g. **ceph-config** directory):

   ```
   cd ceph-config
   vi us-east.json
   ```

   Copy the contents of the following example to the new file **us-east.json**. This configuration uses pool names prepended with the region name and zone name:

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

2. Copy the **us-east.json** zone file to the **/etc/ceph** directory of **us-east-1** Ceph Object Gateway node:

   ```
   scp us-east.json ceph@us-east-1:~
   ssh us-east-1
   sudo mv us-east.json /etc/ceph/
   exit
   ```

3. Copy the **us-east.json** zone file to the **/etc/ceph** directory of **eu-east-1** Ceph Object Gateway node that you will configure as a **master** zone for secondary region as this file will be required for creating this **master** zone of **master** region in **secondary** region as well. You will find it's use in Configuring a Secondary Region.

   Execute the following:

   ```
   scp us-east.json ceph@eu-east-1:~
   ssh eu-east-1
   sudo mv us-east.json /etc/ceph/
   exit
   ```

4. Add the **us-east** zone using the **us-east.json** infile in east zone pool:

```
ssh us-east-1
sudo radosgw-admin zone set --rgw-zone=us-east --infile
/etc/ceph/us-east.json --name client.rgw.us-east-1
```

Repeat step 1 to create a zone infile **us-west.json** for **us-west**. Copy the file to **/etc/ceph** directory of both **us-west-1** and **eu-west-1** Ceph Object Gateway nodes. Then add the zone using the **us-west.json** infile in west zone pool:

```
ssh us-west-1
sudo radosgw-admin zone set --rgw-zone=us-west --infile
/etc/ceph/us-west.json --name client.rgw.us-west-1
```

The **east** and **west** zone pools mentioned here are the data pools that are automatically created by **client.rgw.us-east-1** and **client.rgw.us-west-1** gateway instances respectively. The gateway instances would write to the respective zone data pools. **Do not confuse them for us-east and us-west zone.**

5. Delete the default zone (if it exists):

```
ssh us-east-1
sudo rados -p .rgw.root rm zone_info.default
exit

ssh us-west-1
sudo rados -p .rgw.root rm zone_info.default
exit
```

6. Update the region map:

```
ssh us-east-1
sudo radosgw-admin regionmap update --name client.rgw.us-east-1
exit

ssh us-east-1
sudo radosgw-admin regionmap update --name client.rgw.us-west-1
exit
```

Note

When you use this procedure to configure the secondary region, replace **us-** with **eu-**. You will have a total of four zones **after** you create the master zone and the secondary zone in each region.

31.1.17. Create Zone Users

Ceph Object Gateway stores zone users in the zone pools. So you must create zone users after configuring the zones. Copy the **access_key** and **secret_key** fields for each zone user so you can update your zone configuration once you complete this step:

```
ssh us-east-1
sudo radosgw-admin user create --uid="us-east" --display-name="Region-
US Zone-East" --name client.rgw.us-east-1 --system
```

```
exit

ssh us-west-1
sudo radosgw-admin user create --uid="us-west" --display-name="Region-
US Zone-West" --name client.rgw.us-west-1 --system
exit
```

Note

When you use this procedure to configure the secondary region, replace **`us-`** with **`eu-`**. You will have a total of four zone users **after** you create the master region and the secondary region and their zones.

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.

31.1.18. Update Zone Configurations

You must update the zone configuration with zone users so that the synchronization agents can authenticate with the zones.

1. Open your **`us-east.json`** zone configuration file in the **`admin node`** and paste the contents of the **`access_key`** and **`secret_key`** fields of **`us-east`** zone user, that you got in the output while creating the user in Create Zone Users section, into the **`system_key`** field of your zone configuration infile:

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

```
                "data_pool": ".us-east.rgw.buckets"}
    }
  ]
}
```

2. Save the **us-east.json** file. Copy it again to **/etc/ceph** directory of **us-east-1** and **eu-east-1** Ceph Object Gateway nodes:

```
scp us-east.json ceph@us-east-1
ssh us-east-1
sudo mv us-east.json /etc/ceph/
exit

scp us-east.json ceph@eu-east-1
ssh eu-east-1
sudo mv us-east.json /etc/ceph/
exit
```

3. Then, update your zone configuration:

```
ssh us-east-1
sudo radosgw-admin zone set --rgw-zone=us-east --infile
/etc/ceph/us-east.json --name client.rgw.us-east-1
```

4. Repeat step 1 to update the zone infile for **us-west**. Copy it again to **us-west-1** and **eu-west-1** Ceph Object Gateway nodes. Then, update your zone configuration:

```
ssh us-west-1
sudo radosgw-admin zone set --rgw-zone=us-west --infile
/etc/ceph/us-west.json --name client.rgw.us-west-1
```

Note

When you use this procedure to configure the secondary region, replace **us-** with **eu-**. You will have a total of four zones **after** you create the master zone and the secondary zone in each region.

31.1.19. Configure Bucket Sharding

A Ceph Object Gateway stores bucket index data in the **index_pool**, which defaults to **.rgw.buckets.index**. Sometimes users like to put many objects (hundreds of thousands to millions of objects) in a single bucket. If you do not use the gateway administration interface to set quotas for the maximum number of objects per bucket, the bucket index can suffer significant performance degradation when users place large numbers of objects into a bucket.

In Red Hat Ceph Storage v1.3, you may shard bucket indices to help prevent performance bottlenecks when you allow a high number of objects per bucket. The **rgw_override_bucket_index_max_shards** setting allows you to set a maximum number of shards per bucket. The default value is **0**, which means bucket index sharding is off by default.

To turn bucket index sharding on, set **rgw_override_bucket_index_max_shards** to a value greater than **0**.

For federated configurations, each zone may have a different **index_pool** setting for failover. To make the value consistent for a region's zones, you may set **rgw_override_bucket_index_max_shards** in a gateway's region configuration. For example:

```
ssh us-east-1
sudo radosgw-admin region get > /etc/ceph/us.json
```

Here, **us.json** is the region infile for **master** region.

Open the **us.json** file and edit the **bucket_index_max_shards** setting for each named zone. Save the **us.json** file and reset the region. For example:

```
sudo radosgw-admin region set < /etc/ceph/us.json
```

Once you've updated your region, update the region map. For example:

```
sudo radosgw-admin regionmap update --name client.rgw.us-east-1
exit
```

Where **client.rgw.us-east-1** is the name of the gateway user.

Note

Mapping the index pool (for each zone, if applicable) to a CRUSH ruleset of SSD-based OSDs may also help with bucket index performance.

31.1.20. Restart Services

Once you have redeployed your Ceph configuration files, we recommend restarting your Ceph Storage Cluster(s).

Execute from **admin node** of the cluster:

```
sudo /etc/init.d/ceph restart
```

31.1.21. Restart Gateway Instances

The Ceph Object Gateway daemon needs to be started. To do so, execute the following on each **Ceph Object Gateway node**:

```
sudo systemctl restart ceph-radosgw
```

If you are running multiple instances on the same host, you must specify the user name:

```
sudo systemctl start ceph-radosgw --name client.rgw.us-east-1
```

31.2. CONFIGURE A SECONDARY REGION

This section provides an exemplary procedure for setting up a cluster with multiple regions. Configuring a cluster that spans regions requires maintaining a global namespace, so that there are no namespace clashes among object names stored across in different regions.

This section extends the procedure in Configure a Master Region but changes the region name and modifies a few procedures. See the following sections for details.

31.2.1. Naming for the Secondary Region

Before configuring the cluster, defining region, zone and instance names will help you manage your cluster. Let's assume the region represents the European Union, and we refer to it by its standard abbreviation.

- European Union: **eu**

Let's assume the zones represent the Eastern and Western European Union. For continuity, our naming convention will use **{region name}-{zone name}** format, but you can use any naming convention you prefer.

- European Union, East Region: **eu-east**
- European Union, West Region: **eu-west**

Finally, let's assume that zones may have more than one Ceph Object Gateway instance per zone. For continuity, our naming convention will use **{region name}-{zone name}-{instance}** format, but you can use any naming convention you prefer.

- European Union Region, Master Zone, Instance 1: **eu-east-1**
- European Union Region, Secondary Zone, Instance 1: **eu-west-1**

31.2.2. Configuring a Secondary Region

Repeat the exemplary procedure of Configure a Master Region with the following differences:

1. Use Naming for the Secondary Region in lieu of Naming for the Master Region.
2. Installation in **eu-east-1** and **eu-west-1**.
3. Execute the Pre-Installation requirements for **eu-east-1** and **eu-west-1**.
4. Enable Ceph Client Repository for **eu-east-1** and **eu-west-1**.
5. Install Ceph Object Gateway in **eu-east-1** and **eu-west-1**
6. Install Ceph Object Gateway Synchronization Agent in **eu-east-1** and **eu-west-1**
7. Create Gateway Instances in **eu-east-1** and **eu-west-1**
8. Copy Gateway Keyring to Ceph Configuration Directory in **eu-east-1** and **eu-west-1**.
9. Add Gateway Configuration to Ceph for **eu-east-1** and **eu-west-1**.
10. Distribute Updated Ceph Configuration File to **eu-east-1** and **eu-west-1**.
11. Adjust Firewall Settings for **eu-east-1** and **eu-west-1**.

12. Migrating from Apache to Civetweb (if required) for **eu-east-1** and **eu-west-1**.

13. Create a Region using **eu** instead of **us**. Set **is_master** to **false**. For consistency, create the master region in the secondary region too with the **us.json** file that you had earlier copied to the **secondary** region from **master** region in Create a Region:

```
ssh eu-east-1
sudo radosgw-admin region set --infile /etc/ceph/us.json --name
client.rgw.eu-east-1
```

14. Create Zones using **eu** instead of **us**. Ensure that you update the user name (i.e., **--name**) so that you create the zones in the correct cluster.

15. Create Zone Users for **eu-east** and **eu-west** zone.

16. Update Zone Configurations using **eu** instead of **us**.

    Copy the **eu-east.json** and **eu-west.json** zone files to **/etc/ceph** directory of **us-east-1** and **us-west-1** Ceph Object Gateway nodes as well after you copy them to **eu-east-1** and **eu-west-1** Ceph Object Gateway nodes. They will be required for creating zones from **secondary** region in **master** region.

17. Create zones from master region in the secondary region:

```
ssh eu-east-1
sudo radosgw-admin zone set --rgw-zone=us-east --infile
/etc/ceph/us-east.json --name client.rgw.eu-east-1
exit

ssh eu-west-1
sudo radosgw-admin zone set --rgw-zone=us-west --infile
/etc/ceph/us-west.json --name client.rgw.eu-west-1
exit
```

18. Create zones from secondary region in the master region:

```
ssh us-east-1
sudo radosgw-admin zone set --rgw-zone=eu-east --infile
/etc/ceph/eu-east.json --name client.rgw.us-east-1
exit

ssh us-west-1
sudo radosgw-admin zone set --rgw-zone=eu-west --infile
/etc/ceph/eu-west.json --name client.rgw.us-west-1
exit
```

19. Configure Bucket Sharding for **eu** region.

20. Restart Services.

21. Restart Gateway Instances.

31.3. ACCESS VERIFICATION

You need to verify if the zone users are able to access the gateway. The zone users that you created in Create Zone Users are S3 zone users. You can the verify the gateway access as a S3 user or as a Swift user.

31.3.1. Test S3 access

You need to write and run a Python test script for verifying S3 access. The S3 access test script will connect to the **radosgw**, create a new bucket and list all buckets. The values for **aws_access_key_id** and **aws_secret_access_key** in the Python script are taken from the values of **access_key** and **secret_key** returned by the **radosgw_admin** command while creating the S3 zone users in Create Zone Users.

Execute the following steps on a **master** or **secondary** zone Ceph Object Gateway node:

1. Login to the **Ceph Object Gateway node**.

   For example:

   ```
   ssh us-east-1
   ```

2. Install **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 = 'I0PJDPCIYZ665MW88W9R'
   secret_key = 'dxaXZ8U90SXydYzyS5ivamEP20hkLSUViiaR+ZDA'

   conn = boto.connect_s3(
    aws_access_key_id = access_key,
    aws_secret_access_key = secret_key,
    host = 'us-east-1',
    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,
   )
   ```

   Here, the value for **access_key** and **secret_key** are taken from the output of **sudo radosgw-admin user create --uid="us-east" ...** command that was executed

to create **us-east** zone user in Create Zone Users. The **host** is also taken as **us-east-1**. When you test for other zone users replace these values with relevant ones for a specific zone. Replace {port} with the port number you are using with Civetweb (e.g, 7480 is the **default**). If you already changed the port to **80**, remove the **port = {port},** line from the script.

5. Run the script:

```
python s3test.py
```

The output will be something like the following:

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

31.3.2. Test swift access

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

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

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

To test swift access, you will need the **swift_secret_key** of a swift zone user.

To create a swift zone user for a particular zone like **us-east** execute the following:

```
ssh us-east-1
sudo radosgw-admin subuser create --uid=us-east --subuser=us-east:swift
--name client.rgw.us-east-1 --system --access=full
```

It will create a swift zone user **us-east:swift** with an **access_key**. You will need the **secret_key** for this user as well.

To generate a **secret key** for the swift user, execute:

```
sudo radosgw-admin key create --subuser=us-east:swift --name
client.rgw.us-east-1 --system --key-type=swift --gen-secret
```

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 are using port **80** for Civetweb, there is no need to provide **{port}**.

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
```

31.4. MULTI-SITE DATA REPLICATION

The data synchronization agent replicates the data of a master zone to a secondary zone. The master zone of a region is the source for the secondary zone of the region and it gets selected automatically.

To configure the synchronization agent, retrieve the access key and secret for the source and destination, and the destination URL and port.

You may use **`radosgw-admin zone list`** to get a list of zone names. You may use **`radosgw-admin zone get`** to identify the key and secret for the zone. You may refer to the gateway configuration file you created under Create a Gateway Configuration_ to identify the port number.

You only need the hostname and port for a single instance (assuming all gateway instances in a region/zone access the same Ceph Storage Cluster). Specify these values in a configuration file (e.g., **`region-data-sync.conf`**), and include a **`log_file`** name.

For example:

```
ssh us-east-1
sudo vi /etc/ceph/region-data-sync.conf
```

Paste the following content in the file:

```
src_access_key: {source-access-key}
src_secret_key: {source-secret-key}
destination: https://zone-name.fqdn.com:port
dest_access_key: {destination-access-key}
dest_secret_key: {destination-secret-key}
log_file: {log.filename}
```

A concrete example may look like this:

```
src_access_key: DG8RE354EFPZBICHIAF0
src_secret_key: i3U0HiRP8CXaBWrcF8bbh6CbsxGYuPPwRkixfFSb
destination: https://us-west.storage.net:80
dest_access_key: U60RFI6B08F32T2PD30G
dest_secret_key: W3HuUor7Gl1Ee93pA2pq2wFk1JMQ7hTrSDecYExl
log_file: /var/log/radosgw/radosgw-sync-us-east-west.log
```

To activate the data synchronization agent, open a terminal and execute the following:

```
sudo radosgw-agent -c /etc/ceph/region-data-sync.conf
```

When the synchronization agent is running, you should see output indicating that the agent is synchronizing shards of data:

```
INFO:radosgw_agent.sync:Starting incremental sync
INFO:radosgw_agent.worker:17910 is processing shard number 0
INFO:radosgw_agent.worker:shard 0 has 0 entries after ''
INFO:radosgw_agent.worker:finished processing shard 0
INFO:radosgw_agent.worker:17910 is processing shard number 1
INFO:radosgw_agent.sync:1/64 shards processed
INFO:radosgw_agent.worker:shard 1 has 0 entries after ''
INFO:radosgw_agent.worker:finished processing shard 1
INFO:radosgw_agent.sync:2/64 shards processed
...
```

Note

You must have an agent for each source-destination pair.

31.5. INTER-REGION METADATA REPLICATION

The data synchronization agent replicates the metadata of master zone in the master region to a master zone in a secondary region. Metadata consists of gateway users and buckets, but not the objects within the buckets—ensuring a unified namespace across the cluster. The master zone of the master region is the source for the master zone of the secondary region and it gets selected automatically.

Follow the same steps in Multi-Site Data Replication by specifying the master zone of the master region as the source zone and the master zone of the secondary region as the secondary zone. When activating the **`radosgw-agent`**, specify **`--metadata-only`** so that it only copies metadata. For example:

```
sudo radosgw-agent -c /etc/ceph/inter-region-data-sync.conf --metadata-
only
```

Once you have completed the foregoing procedure, you should have a cluster consisting of a master region (**`us`**) and a secondary region (**`eu`**) where there is a unified namespace between the two regions.