Red Hat Ceph Storage 1.2.3 Ceph Object Gateway for Ubuntu x86 64

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

Red Hat Customer Content Services

Red Hat Ceph Storage 1.2.3 Ceph Object Gateway for Ubuntu x86 64

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

Legal Notice

Abstract

This document provides instructions for installing, configuring and administering a Ceph Storage Object Gateway on Ubuntu Precise and Ubuntu Trusty.

Table of Contents

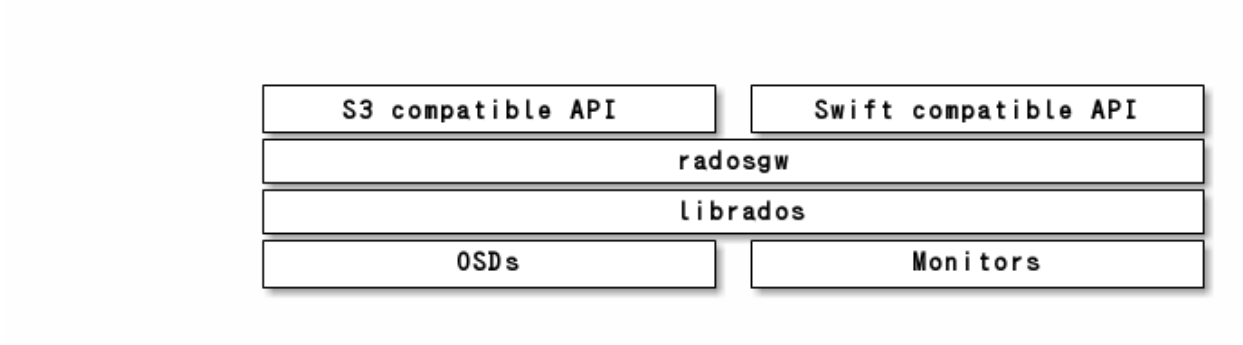
PREFACE

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 built on FastCGI 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 Filesystem clients or Ceph Block Device clients. The S3 and Swift APIs share a common namespace, so you may write data with one API and retrieve it with the other.



Note

Ceph Object Storage does **NOT** use the Ceph Metadata Server.

CHAPTER 1. INSTALL CEPH OBJECT GATEWAY

Note

To run the Ceph object gateway service on **Ubuntu 12.04 (Precise)** or **Ubuntu 14.04 (Trusty)**, you should have a running Ceph cluster and the gateway host should have access to storage and public networks.

The Ceph Object Gateway daemon runs on Apache and FastCGI.

To run a Ceph Object Storage service you must install Apache, FastCGI and Ceph Object Gateway daemon on the host that is going to provide the gateway service, i.e, the **`gateway host`**. To install the Ceph Object Gateway, first install and configure Apache and FastCGI. Then, install the Ceph Object Gateway daemon. If you plan to run a Ceph Object Storage service with a federated architecture (multiple regions and zones), you must also install the synchronization agent.

1. INSTALL APACHE/FASTCGI

On **`Ubuntu 12.04`** and **`Ubuntu 14.04`**, **`multiverse`** needs to be enabled in the package resource list file for **`apt`** i.e, **`/etc/apt/sources.list`**.

Execute the following to install Apache/FastCGI on the **`gateway host`**:

1. For **`Ubuntu 12.04`**, uncomment the following lines in **`/etc/apt/sources.list`**:

```
# deb http://archive.ubuntu.com/ubuntu precise multiverse
# deb-src http://archive.ubuntu.com/ubuntu precise multiverse
# deb http://archive.ubuntu.com/ubuntu precise-updates multiverse
# deb-src http://archive.ubuntu.com/ubuntu precise-updates
multiverse
```

2. For **`Ubuntu 14.04`**, uncomment the following lines in **`/etc/apt/sources.list`**:

```
# deb http://archive.ubuntu.com/ubuntu trusty multiverse
# deb-src http://archive.ubuntu.com/ubuntu trusty multiverse
# deb http://archive.ubuntu.com/ubuntu trusty-updates multiverse
# deb-src http://archive.ubuntu.com/ubuntu trusty-updates
multiverse
```

3. Update the package resource list:

```
sudo apt-get update
```

4. Install Apache and FastCGI:

```
sudo apt-get install apache2 libapache2-mod-fastcgi
```

2. CONFIGURE APACHE

Make the following changes on the **`gateway host`** to configure Apache:

2.1. Give the ServerName in apache2.conf

Add a line for the **`ServerName`** in the **`/etc/apache2/apache2.conf`**. Provide the fully qualified domain name of the server machine (e.g., **`hostname -f`**):

```
ServerName {fqdn}
```

2.2. Enable the URL rewrite modules for Apache and FastCGI

Execute the following:

```
sudo a2enmod rewrite
sudo a2enmod fastcgi
```

Note

When you install **`libapache2-mod-fastcgi`** on **`Ubuntu 14.04`**, it will automatically enable **`fastcgi`** module.

2.3. Start Apache service

```
sudo service apache2 start
```

3. ENABLE SSL

Some REST clients use HTTPS by default. So you should consider enabling SSL for Apache. Use the following procedures to enable SSL on the **`gateway host`**.

Note

You can use self-certified certificates. Some client APIs check for a trusted certificate authority. You may need to obtain a SSL certificate from a trusted authority to use those client APIs.

To enable SSL, execute the following steps:

1. Ensure that you have installed the dependencies:

   ```
   sudo apt-get install openssl ssl-cert
   ```

2. Enable the SSL module:

   ```
   sudo a2enmod ssl
   ```

3. Generate a certificate:

```
sudo mkdir /etc/apache2/ssl
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/apache2/ssl/apache.key -out /etc/apache2/ssl/apache.crt
```

4. Restart Apache:

```
sudo service apache2 restart
```

4. INSTALL CEPH OBJECT GATEWAY DAEMON

Ceph Object Storage services use the Ceph Object Gateway daemon (**`radosgw`**) to enable the gateway. For federated architectures, the synchronization agent (**`radosgw-agent`**) provides data and metadata synchronization between zones and regions.

To install the Ceph Object Gateway daemon on the **`gateway host`**, execute the following:

```
sudo apt-get install radosgw
```

To install the Ceph Object Gateway synchronization agent, execute the following:

```
sudo apt-get install radosgw-agent
```

CHAPTER 2. CONFIGURE CEPH OBJECT GATEWAY

Once you have installed the Ceph Object Gateway packages, the next step is to configure your Ceph Object Gateway. There are two approaches:

- **Simple:** A **`simple`** Ceph Object Gateway configuration implies that you are running a Ceph Object Storage service in a single data center. So you can configure the Ceph Object Gateway without regard to regions and zones.
- **Federated:** 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. This involves configuring your Ceph Object Gateway instances with regions and zones.

In this guide we shall proceed with a **`simple`** Ceph Object Gateway configuration.

The Ceph Object Gateway is a client of the Ceph Storage Cluster. As a Ceph Storage Cluster client, it requires:

- A name for the gateway instance. We use **`gateway`** in this guide.
- A storage cluster user name with appropriate permissions in a keyring.
- Pools to store its data.
- A data directory for the gateway instance.
- An instance entry in the Ceph Configuration file.
- A configuration file for the web server to interact with FastCGI.

The configuration steps are as follows:

1. CREATE A USER AND KEYRING

Each instance must have a user name and key to communicate with a Ceph Storage Cluster. In the following steps, we use an **`admin node`** to create a keyring. Then, we create a client user name and key. Next, we add the key to the Ceph Storage Cluster. Finally, we distribute the key ring to the node containing the gateway instance i.e, **`gateway host`**.

Monitor Key CAPS

When you provide CAPS to the key, you MUST provide read capability. However, you have the option of providing write capability for the monitor. This is an important choice. If you provide write capability to the key, the Ceph Object Gateway will have the ability to create pools automatically; however, it will create pools with either the default number of placement groups (not ideal) or the number of placement groups 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 first.

Execute the following steps on the **`admin node`** of your cluster:

1. Create a keyring for the gateway:

```
sudo ceph-authtool --create-keyring
/etc/ceph/ceph.client.radosgw.keyring
sudo chmod +r /etc/ceph/ceph.client.radosgw.keyring
```

2. Generate a Ceph Object Gateway user name and key for each instance. For exemplary purposes, we will use the name **gateway** after **client.radosgw**:

```
sudo ceph-authtool /etc/ceph/ceph.client.radosgw.keyring -n
client.radosgw.gateway --gen-key
```

3. Add capabilities to the key:

```
sudo ceph-authtool -n client.radosgw.gateway --cap osd 'allow
rwx' --cap mon 'allow rwx' /etc/ceph/ceph.client.radosgw.keyring
```

4. Once you have created a keyring and key to enable the Ceph Object Gateway with access to the Ceph Storage Cluster, add the key to your Ceph Storage Cluster. For example:

```
sudo ceph -k /etc/ceph/ceph.client.admin.keyring auth add
client.radosgw.gateway -i /etc/ceph/ceph.client.radosgw.keyring
```

5. Distribute the keyring to the **gateway host**:

```
sudo scp /etc/ceph/ceph.client.radosgw.keyring
ceph@{hostname}:/home/ceph
ssh {hostname}
sudo mv ceph.client.radosgw.keyring
/etc/ceph/ceph.client.radosgw.keyring
```

Note

The 5th step is optional if **admin node** is the **gateway host**.

2. CREATE POOLS

Ceph Object Gateways require Ceph Storage Cluster pools to store specific gateway data. If the user you created has permissions, the gateway will create the pools automatically. However, you should ensure that you have set an appropriate default number of placement groups per pool into your Ceph configuration file.

When configuring a gateway with the default region and zone, the naming convention for pools typically omits region and zone naming, but you can use any naming convention you prefer. For example:

- **.rgw**
- **.rgw.root**
- **.rgw.control**
- **.rgw.gc**
- **.rgw.buckets**
- **.rgw.buckets.index**
- **.log**

- **.intent-log**
- **.usage**
- **.users**
- **.users.email**
- **.users.swift**
- **.users.uid**

As already said, if write permission is given, Ceph Object Gateway will create pools automatically. To create a pool manually, execute the following:

```
ceph osd pool create {poolname} {pg-num} {pgp-num}
```

Note

When adding a large number of pools, it may take some time for your cluster to return to a **active + clean** state.

When you have completed this step, execute the following to ensure that you have created all of the foregoing pools:

```
rados lspools
```

3. ADD A GATEWAY CONFIGURATION TO CEPH

Add the Ceph Object Gateway configuration to your Ceph Configuration file in **admin node**. The Ceph Object Gateway configuration requires you to identify the Ceph Object Gateway instance. Then, you must specify the host name where you installed the Ceph Object Gateway daemon, a keyring (for use with cephx), the socket path for FastCGI and a log file.

Append the following configuration to **/etc/ceph/ceph.conf** in your **admin node**:

```
[client.radosgw.gateway]
host = {hostname}
keyring = /etc/ceph/ceph.client.radosgw.keyring
rgw socket path = /var/run/ceph/ceph.radosgw.gateway.fastcgi.sock
log file = /var/log/radosgw/client.radosgw.gateway.log
```

Note

Here, **{hostname}** is the short hostname (output of command **hostname -s**) of the node that is going to provide the gateway service i.e, the **gateway host**.

The **[client.radosgw.gateway]** portion of the gateway instance 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., **radosgw**).

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 to help troubleshoot any configuration issues. For example:

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

4. DISTRIBUTE UPDATED CEPH CONFIGURATION FILE

The updated Ceph configuration file needs to be distributed to all Ceph cluster nodes from the **admin node**.

It involves the following steps:

1. Pull the updated **ceph.conf** from **/etc/ceph/** to the root directory of the cluster in admin node (e.g. **ceph-config** directory). The contents of **ceph.conf** in **ceph-config** will get overwritten. To do so, execute the following:

   ```
   ceph-deploy --overwrite-conf config pull {hostname}
   ```

   Here, **{hostname}** is the short hostname of the Ceph admin node.

2. Push the updated **ceph.conf** file from the admin node to all other nodes in the cluster including the **gateway host**:

   ```
   ceph-deploy --overwrite-conf config push [HOST] [HOST...]
   ```

Give the hostnames of the other Ceph nodes in place of **[HOST] [HOST...]**.

5. COPY CEPH.CLIENT.ADMIN.KEYRING FROM ADMIN NODE TO GATEWAY HOST

As the **gateway host** can be a different node that is not part of the cluster, the **ceph.client.admin.keyring** needs to be copied from the **admin node** to the **gateway host**. To do so, execute the following on **admin node**:

```
sudo scp /etc/ceph/ceph.client.admin.keyring
ceph@{hostname}:/home/ceph
ssh {hostname}
sudo mv ceph.client.admin.keyring /etc/ceph/ceph.client.admin.keyring
```

Note

The above step need not be executed if **admin node** is the **gateway host**.

6. CREATE A CGI WRAPPER SCRIPT

The wrapper script provides the interface between the webserver and the radosgw process. This script needs to be in a web accessible location and should be executable.

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

1. Create the script:

```
sudo vi /var/www/html/s3gw.fcgi
```

2. Add the following content to the script:

```
#!/bin/sh
exec /usr/bin/radosgw -c /etc/ceph/ceph.conf -n
client.radosgw.gateway
```

3. Provide execute permissions to the script:

```
sudo chmod +x /var/www/html/s3gw.fcgi
```

7. CREATE DATA DIRECTORY

Deployment scripts may not create the default Ceph Object Gateway data directory. Create data directories for each instance of a **radosgw** daemon (if you haven't done so already). The **host** variables in the Ceph configuration file determine which host runs each instance of a **radosgw** daemon. The typical form specifies the **radosgw** daemon, the cluster name and the daemon ID.

To create the directory on the **gateway host**, execute the following:

```
sudo mkdir -p /var/lib/ceph/radosgw/ceph-radosgw.gateway
```

8. START RADOSGW SERVICE

The Ceph Object gateway daemon needs to be started. To do so, execute the following on the **gateway host**:

```
sudo /etc/init.d/radosgw start
```

9. CREATE A GATEWAY CONFIGURATION FILE

On the host where you installed the Ceph Object Gateway i.e, **gateway host**, create an **rgw.conf** file. Place the file in **/etc/apache2/sites-available** directory. It is a **Apache** configuration file which is needed for the radosgw service. This file must be readable by the web server.

We recommend deploying FastCGI as an external server, because allowing Apache to manage FastCGI sometimes introduces high latency. To manage FastCGI as an external server, use the **FastCgiExternalServer** directive.

Ceph Object Gateway requires a rewrite rule for the Amazon S3-compatible interface. It's required for passing in the **HTTP_AUTHORIZATION env** for S3, which is filtered out by Apache. The rewrite rule is not necessary for the OpenStack Swift-compatible interface.

You should configure Apache to allow encoded slashes, provide paths for log files and to turn off server signatures.

Execute the following steps :

1. Create the file:

```
sudo vi /etc/apache2/sites-available/rgw.conf
```

2. Add the following contents to the file:

```
FastCgiExternalServer /var/www/html/s3gw.fcgi -socket
/var/run/ceph/ceph.radosgw.gateway.fastcgi.sock

<VirtualHost *:80>
ServerName localhost
DocumentRoot /var/www/html

ErrorLog /var/log/apache2/rgw_error.log
CustomLog /var/log/apache2/rgw_access.log combined

# LogLevel debug

RewriteEngine On

RewriteRule ^/([a-zA-Z0-9-_.]*)([/]?.*) /s3gw.fcgi?
page=$1&params=$2&%{QUERY_STRING} [E=HTTP_AUTHORIZATION:%
{HTTP:Authorization},L]

<IfModule mod_fastcgi.c>
 <Directory /var/www/html>
 Options +ExecCGI
 AllowOverride All
 SetHandler fastcgi-script
 Order allow,deny
 Allow from all
 AuthBasicAuthoritative Off
 </Directory>
</IfModule>

AllowEncodedSlashes On
ServerSignature Off

</VirtualHost>
```

3. Disable the default site:

```
sudo s2dissite 000-default
```

4. Enable the configuration file:

```
sudo a2ensite rgw.conf
```

Important

In the **`rgw.conf`** file, although the **`RewriteRule`** appears to be in two lines, actually it is a single line and should be written as a single line only.

10. RESTART APACHE SERVICE

The Apache service needs to be restarted to accept the new configuration.

```
sudo service apache2 restart
```

CHAPTER 3. 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.

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.

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": []}
```

3. ACCESS VERIFICATION

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 apt-get 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 = '{FQDN}',
   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 **{FQDN}** with the full hostname i.e, the fully qualified domain name of the host where you have configured the gateway service i.e, the **gateway host**.

4. Run the script:

```
python s3test.py
```

The output will be something like the following:

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

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 apt-get 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}/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.

For example:

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

The output should be:

```
my-new-bucket
```