Red Hat Ceph Storage 7

开发人员指南

为 Red Hat Ceph Storage 使用各种应用程序编程接口

Last Updated: 2024-06-14

Red Hat Ceph Storage 7 开发人员指南

为 Red Hat Ceph Storage 使用各种应用程序编程接口

法律通告

摘要

本文档提供了有关为在 AMD64 和 Intel 64 构架中运行的 Red Hat Ceph Storage 使用各种应用程序编程接口的说明。 红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看 CTO Chris Wright 信息。

目录

第 1 章 CEPH RESTFUL API

作为存储管理员，您可以使用 Ceph RESTful API，或者只是由 Red Hat Ceph Storage Dashboard 提供的 Ceph API，以便与 Red Hat Ceph Storage 集群交互。您可以显示有关 Ceph Monitor 和 OSD 的信息，以及它们对应的配置选项。您甚至可以创建或编辑 Ceph 池。

Ceph API 使用以下标准：

- HTTP 1.1
- JSON
- MIME 和 HTTP 内容协商
- JWT

这些标准符合 OpenAPI 3.0，符合 API 语法、语义、内容编码、版本控制、身份验证和授权。

先决条件

- 一个正常运行的 Red Hat Ceph Storage 集群。
- 访问运行 Ceph Manager 的节点。

1.1. CEPH API 的版本

Ceph RESTful API 的主要目标是提供稳定的接口。为了实现稳定的接口，Ceph API 基于以下原则构建：

- 所有端点都有一个强制的显式默认版本，以避免隐式默认值。
- 精细更改每个端点的控制。
  - HTTP 标头中声明了来自特定端点的预期版本。

    语法

    ```
    Accept: application/vnd.ceph.api.v*MAJOR.MINOR*+json
    ```

    示例

    ```
    Accept: application/vnd.ceph.api.v1.0+json
    ```

    如果当前的 Ceph API 服务器无法解决该特定版本，则将返回 **415 - Unsupported Media Type** 响应。
- 使用语义版本.
  - 主要变化是向后不兼容的。更改可能会导致对请求进行非附加更改，以及特定端点的响应格式。
  - 次要变化是向后和向前兼容的。更改包括对特定端点的请求或响应格式的添加更改。

1.2. CEPH API 的身份验证和授权

访问 Ceph RESTful API 会经过两个检查点。第一种方法是验证请求是否代表有效和现有用户完成。其次，授权之前经过身份验证的用户可以在目标端点上创建、读取、更新或删除特定操作。

用户开始使用 Ceph API 之前，需要有效的 JSON Web 令牌(JWT)。您可以通过 **/api/auth** 端点来检索这个令牌。

示例

```
[root@mon ~]# curl -X POST "https://example.com:8443/api/auth" \
  -H  "Accept: application/vnd.ceph.api.v1.0+json" \
  -H  "Content-Type: application/json" \
  -d '{"username": user1, "password": password1}'
```

此令牌必须与每个 API 请求一起使用，方法是将其放在 **Authorization** HTTP 标头中。

语法

```
curl -H "Authorization: Bearer TOKEN" ...
```

其它资源

- 如需了解更多详细信息，请参阅 *Red Hat Ceph Storage Administration Guide* 中的 Ceph 用户管理一章。 *https://access.redhat.com/documentation/zh-cn/red_hat_ceph_storage/7/html-single/administration_guide/#ceph-user-management*

1.3. 启用和保护 CEPH API 模块

Red Hat Ceph Storage Dashboard 模块通过 SSL 保护的连接提供存储集群的 RESTful API 访问。

重要

如果禁用 SSL，则用户名和密码会以未加密的形式发送到 Red Hat Ceph Storage 仪表板。

先决条件

- *Ceph 监控节点的根级别访问权限.*
- *确保至少有一个 **ceph-mgr** 守护进程处于活跃状态。*
- *如果您使用防火墙，请确保 TCP 端口 **8443** （对于 SSL）和 TCP 端口 **8080** （没有 SSL）已在带有活跃 **ceph-mgr** 守护进程的节点上打开。*

流程

1. *登录到 Cephadm shell：*

   示例

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

2. *启用 RESTful 插件：*

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

3. 配置 SSL 证书。

   a. 如果机构的证书颁发机构(CA)提供证书，则使用证书文件设置：

      语法

      ```
      ceph dashboard set-ssl-certificate HOST_NAME -i CERT_FILE
      ceph dashboard set-ssl-certificate-key HOST_NAME -i KEY_FILE
      ```

      示例

      ```
      [ceph: root@host01 /]# ceph dashboard set-ssl-certificate -i dashboard.crt
      [ceph: root@host01 /]# ceph dashboard set-ssl-certificate-key -i dashboard.key
      ```

      如果要设置基于节点的唯一证书，请在命令中添加 HOST_NAME ：

      示例

      ```
      [ceph: root@host01 /]# ceph dashboard set-ssl-certificate host01 -i dashboard.crt
      [ceph: root@host01 /]# ceph dashboard set-ssl-certificate-key host01 -i dashboard.key
      ```

   b. 或者，您可以生成自签名证书。但是，使用自签名证书无法提供 HTTPS 协议的完整安全优势：

      ```
      [ceph: root@host01 /]# ceph dashboard create-self-signed-cert
      ```

      警告

      大多数现代 Web 浏览器都会提示自签名证书，这需要您在建立安全连接前进行确认。

4. 创建用户，设置密码并设置角色：

   语法

   ```
   echo -n "PASSWORD" > PATH_TO_FILE/PASSWORD_FILE
   ceph dashboard ac-user-create USER_NAME -i PASSWORD_FILE ROLE
   ```

   示例

   ```
   [ceph: root@host01 /]# echo -n "p@ssw0rd" > /root/dash-password.txt
   [ceph: root@host01 /]# ceph dashboard ac-user-create user1 -i /root/dash-password.txt
   administrator
   ```

   这个示例创建一个名为 **user1** 的用户，该用户具有 **administrator** 角色。

5. 连接到 RESTful 插件网页。打开 Web 浏览器，并输入以下 URL ：

语法

```
https://HOST_NAME:8443
```

示例

```
https://host01:8443
```

如果您使用了自签名证书，请确认安全例外。

其它资源

- **ceph dashboard --help** 命令。
- **https://HOST_NAME:8443/doc** 页面，其中 HOST_NAME 是运行 **ceph-mgr** 实例的节点的 IP 地址或名称。
- 如需更多信息，请参阅红帽客户门户网站中的 Red Hat Enterprise Linux 产品文档中的 安全强化指南。

1.4. 问题和答案

1.4.1. 获取信息

本节介绍如何使用 Ceph API 查看有关存储集群、Ceph 监控器、OSD、池和主机的信息。

1.4.1.1. 如何查看所有集群配置选项？

本节论述了如何使用 RESTful 插件查看集群配置选项及其值。

curl 命令
在命令行中使用：

```
curl --silent --user USER 'https://CEPH_MANAGER:CEPH_MANAGER_PORT/api/cluster_conf'
```

替换：

- 使用用户名替代 **USER**
- 使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER**
- **CEPH_MANAGER_PORT** 和 TCP 端口号。默认 TCP 端口号为 8443。

提示时输入用户密码。

如果您使用自签名证书，请使用 **--insecure** 选项：

```
curl --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/cluster_conf'
```

Python
在 Python 解释器中，输入：

```
$ python
```

```
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/cluster_conf', auth=("USER",
"PASSWORD"))
>> print result.json()
```

替换：

- *使用带有活跃 **ceph-mgr** 实例的节点的IP地址或短主机名替换 **CEPH_MANAGER***
- *使用用户名替代 **USER***
- *使用用户的密码替换 **PASSWORD***

*如果您使用自签名证书，请使用 **verify=False** 选项：*

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/cluster_conf', auth=("USER",
"PASSWORD"), verify=False)
>> print result.json()
```

Web 浏览器
在 Web 浏览器中，输入：

```
https://CEPH_MANAGER:8080/api/cluster_conf
```

替换：

- *使用带有活跃 **ceph-mgr** 实例的节点的IP地址或短主机名替换 **CEPH_MANAGER***

提示时输入用户名和密码。

其它资源

- *Red Hat Ceph Storage 7 的 配置指南*

1.4.1.2. 如何查看 Particular 集群配置选项？

本节论述了如何查看特定集群选项及其值。

curl 命令
在命令行中使用：

```
curl --silent --user USER 'https://CEPH_MANAGER:8080/api/cluster_conf/ARGUMENT'
```

替换：

- *使用用户名替代 **USER***
- *使用带有活跃 **ceph-mgr** 实例的节点的IP地址或短主机名替换 **CEPH_MANAGER***
- *使用您要查看的配置选项替换 **ARGUMENT***

提示时输入用户密码。

*如果您使用自签名证书，请使用 **--insecure** 选项：*

```
curl --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/cluster_conf/ARGUMENT'
```

Python
在 Python 解释器中，输入：

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/cluster_conf/ARGUMENT', auth=
("USER", "PASSWORD"))
>> print result.json()
```

替换：

- *使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER***
- *使用您要查看的配置选项替换 **ARGUMENT***
- *使用用户名替代 **USER***
- *使用用户的密码替换 **PASSWORD***

*如果您使用自签名证书，请使用 **verify=False** 选项：*

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/cluster_conf/ARGUMENT', auth=
("USER", "PASSWORD"), verify=False)
>> print result.json()
```

Web 浏览器
在 Web 浏览器中，输入：

```
https://CEPH_MANAGER:8080/api/cluster_conf/ARGUMENT
```

替换：

- *使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER***
- *使用您要查看的配置选项替换 **ARGUMENT***

提示时输入用户名和密码。

其它资源

- *Red Hat Ceph Storage 7 的 配置指南*

1.4.1.3. 如何查看 OSD 的所有配置选项？

本节论述了如何查看 OSD 的所有配置选项及其值。

curl 命令
在命令行中使用：

```
curl --silent --user USER 'https://CEPH_MANAGER:8080/api/osd/flags'
```

替换：

- *使用用户名替代* ***USER***
- *使用带有活跃* ***ceph-mgr*** *实例的节点的IP地址或短主机名替换* ***CEPH_MANAGER***

提示时输入用户密码。

如果您使用自签名证书，请使用 ***--insecure*** *选项：*

```
curl --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/osd/flags'
```

Python
在 Python 解释器中，输入：

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/osd/flags', auth=("USER",
"PASSWORD"))
>> print result.json()
```

替换：

- *使用带有活跃* ***ceph-mgr*** *实例的节点的IP地址或短主机名替换* ***CEPH_MANAGER***
- *使用用户名替代* ***USER***
- *使用用户的密码替换* ***PASSWORD***

如果您使用自签名证书，请使用 ***verify=False*** *选项：*

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/osd/flags', auth=("USER",
"PASSWORD"), verify=False)
>> print result.json()
```

Web 浏览器
在 Web 浏览器中，输入：

```
https://CEPH_MANAGER:8080/api/osd/flags
```

替换：

- *使用带有活跃* ***ceph-mgr*** *实例的节点的IP地址或短主机名替换* ***CEPH_MANAGER***

提示时输入用户名和密码。

其它资源

- *Red Hat Ceph Storage 7 的 配置指南*

1.4.1.4. 如何查看 CRUSH 规则？

本节论述了如何查看 CRUSH 规则。

curl 命令
在命令行中使用：

```
curl --silent --user USER 'https://CEPH_MANAGER:8080/api/crush_rule'
```

替换：

- 使用用户名替代 **USER**
- 使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER**

提示时输入用户密码。

如果您使用自签名证书，请使用 **--insecure** 选项：

```
curl --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/crush_rule'
```

Python
在 Python 解释器中，输入：

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/crush_rule', auth=("USER",
"PASSWORD"))
>> print result.json()
```

替换：

- 使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER**
- 使用用户名替代 **USER**
- 使用用户的密码替换 **PASSWORD**

如果您使用自签名证书，请使用 **verify=False** 选项：

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/crush_rule', auth=("USER",
"PASSWORD"), verify=False)
>> print result.json()
```

Web 浏览器
在 Web 浏览器中，输入：

```
https://CEPH_MANAGER:8080/api/crush_rule
```

替换：

- 使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER**

提示时输入用户名和密码。

其它资源

- *Red Hat Ceph Storage 7 管理指南中的 CRUSH Rules 部分。*

1.4.1.5. 如何查看有关 monitor 的信息？

本节论述了如何查看特定 monitor 的信息，例如：

- *IP 地址*
- *名称*
- *仲裁状态*

curl 命令
在命令行中使用：

```
curl --silent --user USER 'https://CEPH_MANAGER:8080/api/monitor'
```

替换：

- *使用用户名替代 **USER***
- *使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER***

提示时输入用户密码。

*如果您使用自签名证书，请使用 **--insecure** 选项：*

```
curl --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/monitor'
```

Python
在 Python 解释器中，输入：

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/monitor', auth=("USER",
"PASSWORD"))
>> print result.json()
```

替换：

- *使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER***
- *使用用户名替代 **USER***
- *使用用户的密码替换 **PASSWORD***

*如果您使用自签名证书，请使用 **verify=False** 选项：*

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/monitor', auth=("USER",
```

```
"PASSWORD"), verify=False)
>> print result.json()
```

Web 浏览器
在 Web 浏览器中，输入：

```
https://CEPH_MANAGER:8080/api/monitor
```

替换：

- 使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER**

提示时输入用户名和密码。

1.4.1.6. 如何查看关于 Particular monitor 的信息？

本节论述了如何查看特定 monitor 的信息，例如：

- IP 地址
- 名称
- 仲裁状态

curl 命令
在命令行中使用：

```
curl --silent --user USER 'https://CEPH_MANAGER:8080/api/monitor/NAME'
```

替换：

- 使用用户名替代 **USER**
- 使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER**
- 使用 monitor 短主机名替换 **NAME**

提示时输入用户密码。

如果您使用自签名证书，请使用 **--insecure** 选项：

```
curl --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/monitor/NAME'
```

Python
在 Python 解释器中，输入：

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/monitor/NAME', auth=("USER",
"PASSWORD"))
>> print result.json()
```

替换：

- 使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER**

- *使用 monitor 短主机名替换 **NAME***
- *使用用户名替代 **USER***
- *使用用户的密码替换 **PASSWORD***

*如果您使用自签名证书，请使用 **verify=False** 选项：*

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/monitor/NAME', auth=("USER",
"PASSWORD"), verify=False)
>> print result.json()
```

Web 浏览器
在 Web 浏览器中，输入：

```
https://CEPH_MANAGER:8080/api/monitor/NAME
```

替换：

- *使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER***
- *使用 monitor 短主机名替换 **NAME***

提示时输入用户名和密码。

1.4.1.7. 如何查看 OSD 的信息？

本节论述了如何查看 OSD 的信息，例如：

- *IP 地址*
- *其池*
- *关联性*
- *Weight*

curl 命令
在命令行中使用：

```
curl --silent --user USER 'https://CEPH_MANAGER:8080/api/osd'
```

替换：

- *使用用户名替代 **USER***
- *使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER***

提示时输入用户密码。

*如果您使用自签名证书，请使用 **--insecure** 选项：*

```
curl --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/osd'
```

Python
在 Python 解释器中，输入：

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/osd/', auth=("USER", "PASSWORD"))
>> print result.json()
```

替换：

- *使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER***
- *使用用户名替代 **USER***
- *使用用户的密码替换 **PASSWORD***

*如果您使用自签名证书，请使用 **verify=False** 选项：*

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/osd/', auth=("USER", "PASSWORD"),
verify=False)
>> print result.json()
```

Web 浏览器
在 Web 浏览器中，输入：

```
https://CEPH_MANAGER:8080/api/osd
```

替换：

- *使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER***

提示时输入用户名和密码。

1.4.1.8. 如何查看 Particular OSD 的信息？

本节论述了如何查看特定 OSD 的信息，例如：

- *IP 地址*
- *其池*
- *关联性*
- *Weight*

curl 命令
在命令行中使用：

```
curl --silent --user USER 'https://CEPH_MANAGER:8080/api/osd/ID'
```

替换：

- *使用用户名替代 **USER***

- 使用带有活跃 **ceph-mgr** 实例的节点的IP地址或短主机名替换 **CEPH_MANAGER**
- 使用 **osd** 字段中列出的OSD 的ID 替换 **ID**

提示时输入用户密码。

如果您使用自签名证书，请使用 **--insecure** 选项：

```
curl --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/osd/ID'
```

Python
在 Python 解释器中，输入：

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/osd/ID', auth=("USER", "PASSWORD"))
>> print result.json()
```

替换：

- 使用带有活跃 **ceph-mgr** 实例的节点的IP地址或短主机名替换 **CEPH_MANAGER**
- 使用 **osd** 字段中列出的OSD 的ID 替换 **ID**
- 使用用户名替代 **USER**
- 使用用户的密码替换 **PASSWORD**

如果您使用自签名证书，请使用 **verify=False** 选项：

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/osd/ID', auth=("USER", "PASSWORD"),
verify=False)
>> print result.json()
```

Web 浏览器
在 Web 浏览器中，输入：

```
https://CEPH_MANAGER:8080/api/osd/ID
```

替换：

- 使用带有活跃 **ceph-mgr** 实例的节点的IP地址或短主机名替换 **CEPH_MANAGER**
- 使用 **osd** 字段中列出的OSD 的ID 替换 **ID**

提示时输入用户名和密码。

1.4.1.9. 如何确定 OSD 上可以调度哪些进程？

本节介绍如何使用 RESTful 插件查看清理或深度清理等进程，可以调度到 OSD 上。

curl 命令
在命令行中使用：

```
curl --silent --user USER 'https://CEPH_MANAGER:8080/api/osd/ID/command'
```

替换：

- *使用用户名替代* ***USER***
- *使用带有活跃* ***ceph-mgr*** *实例的节点的 IP 地址或短主机名替换* ***CEPH_MANAGER***
- *使用* ***osd*** *字段中列出的 OSD 的 ID 替换* ***ID***

提示时输入用户密码。

如果您使用自签名证书，请使用 ***--insecure*** *选项：*

```
curl --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/osd/ID/command'
```

Python
在 Python 解释器中，输入：

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/osd/ID/command', auth=("USER",
"PASSWORD"))
>> print result.json()
```

替换：

- *使用带有活跃* ***ceph-mgr*** *实例的节点的 IP 地址或短主机名替换* ***CEPH_MANAGER***
- *使用* ***osd*** *字段中列出的 OSD 的 ID 替换* ***ID***
- *使用用户名替代* ***USER***
- *使用用户的密码替换* ***PASSWORD***

如果您使用自签名证书，请使用 ***verify=False*** *选项：*

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/osd/ID/command', auth=("USER",
"PASSWORD"), verify=False)
>> print result.json()
```

Web 浏览器
在 Web 浏览器中，输入：

```
https://CEPH_MANAGER:8080/api/osd/ID/command
```

替换：

- *使用带有活跃* ***ceph-mgr*** *实例的节点的 IP 地址或短主机名替换* ***CEPH_MANAGER***
- *使用* ***osd*** *字段中列出的 OSD 的 ID 替换* ***ID***

提示时输入用户名和密码。

1.4.1.10. 如何查看关于池的信息？

本节论述了如何查看池的信息，例如：

- 标记
- 大小
- 放置组数量

curl 命令
在命令行中使用：

```
curl --silent --user USER 'https://CEPH_MANAGER:8080/api/pool'
```

替换：

- 使用用户名替代 **USER**
- 使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER**

提示时输入用户密码。

如果您使用自签名证书，请使用 **--insecure** 选项：

```
curl --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/pool'
```

Python
在 Python 解释器中，输入：

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/pool', auth=("USER", "PASSWORD"))
>> print result.json()
```

替换：

- 使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER**
- 使用用户名替代 **USER**
- 使用用户的密码替换 **PASSWORD**

如果您使用自签名证书，请使用 **verify=False** 选项：

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/pool', auth=("USER", "PASSWORD"),
verify=False)
>> print result.json()
```

Web 浏览器
在 Web 浏览器中，输入：

```
https://CEPH_MANAGER:8080/api/pool
```

替换：

- *使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER***

提示时输入用户名和密码。

1.4.1.11. 如何查看关于 Particular 池的信息？

本节论述了如何查看特定池的信息，例如：

- *标记*
- *大小*
- *放置组数量*

curl 命令
在命令行中使用：

```
curl --silent --user USER 'https://CEPH_MANAGER:8080/api/pool/ID'
```

替换：

- *使用用户名替代 **USER***
- *使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER***
- *使用 **pool** 字段中列出的池的 ID 替换 **ID***

提示时输入用户密码。

*如果您使用自签名证书，请使用 **--insecure** 选项：*

```
curl --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/pool/ID'
```

Python
在 Python 解释器中，输入：

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/pool/ID', auth=("USER", "PASSWORD"))
>> print result.json()
```

替换：

- *使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER***
- *使用 **pool** 字段中列出的池的 ID 替换 **ID***
- *使用用户名替代 **USER***
- *使用用户的密码替换 **PASSWORD***

*如果您使用自签名证书，请使用 **verify=False** 选项：*

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/pool/ID', auth=("USER", "PASSWORD"),
verify=False)
>> print result.json()
```

Web 浏览器
在 Web 浏览器中，输入：

```
https://CEPH_MANAGER:8080/api/pool/ID
```

替换：

- 使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER**
- 使用 **pool** 字段中列出的池的 ID 替换 **ID**

提示时输入用户名和密码。

1.4.1.12. 如何查看关于主机的信息？

本节论述了如何查看主机的信息，例如：

- 主机名
- Ceph 守护进程及其 ID
- Ceph 版本

curl 命令
在命令行中使用：

```
curl --silent --user USER 'https://CEPH_MANAGER:8080/api/host'
```

替换：

- 使用用户名替代 **USER**
- 使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER**

提示时输入用户密码。

如果您使用自签名证书，请使用 **--insecure** 选项：

```
curl --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/host'
```

Python
在 Python 解释器中，输入：

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/host', auth=("USER", "PASSWORD"))
>> print result.json()
```

替换：

- *使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER***
- *使用用户名替代 **USER***
- *使用用户的密码替换 **PASSWORD***

*如果您使用自签名证书，请使用 **verify=False** 选项：*

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/host', auth=("USER", "PASSWORD"),
verify=False)
>> print result.json()
```

Web 浏览器
在 Web 浏览器中，输入：

```
https://CEPH_MANAGER:8080/api/host
```

替换：

- *使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER***

提示时输入用户名和密码。

1.4.1.13. 如何查看关于 Particular 主机的信息？

本节论述了如何查看特定主机的信息，例如：

- *主机名*
- *Ceph 守护进程及其 ID*
- *Ceph 版本*

curl 命令
在命令行中使用：

```
curl --silent --user USER 'https://CEPH_MANAGER:8080/api/host/HOST_NAME'
```

替换：

- *使用用户名替代 **USER***
- *使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER***
- *使用在 **hostname** 项中列出的主机名替换 **HOST_NAME***

提示时输入用户密码。

*如果您使用自签名证书，请使用 **--insecure** 选项：*

```
curl --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/host/HOST_NAME'
```

Python
在 Python 解释器中，输入：

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/host/HOST_NAME', auth=("USER",
"PASSWORD"))
>> print result.json()
```

替换：

- *使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER***
- *使用在 **hostname** 项中列出的主机名替换 **HOST_NAME***
- *使用用户名替代 **USER***
- *使用用户的密码替换 **PASSWORD***

*如果您使用自签名证书，请使用 **verify=False** 选项：*

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/host/HOST_NAME', auth=("USER",
"PASSWORD"), verify=False)
>> print result.json()
```

Web 浏览器
在 Web 浏览器中，输入：

```
https://CEPH_MANAGER:8080/api/host/HOST_NAME
```

替换：

- *使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER***
- *使用在 **hostname** 项中列出的主机名替换 **HOST_NAME***

提示时输入用户名和密码。

1.4.2. 更改配置

本节介绍如何使用 Ceph API 更改 OSD 配置选项、OSD 的状态以及池的相关信息。

1.4.2.1. 如何更改 OSD 配置选项？

本节论述了如何使用 RESTful 插件更改 OSD 配置选项。

curl 命令
在命令行中使用：

```
echo -En '{"OPTION": VALUE}' | curl --request PATCH --data @- --silent --user USER
'https://CEPH_MANAGER:8080/api/osd/flags'
```

替换：

- *使用要修改的选项替换* ***OPTION***，***pause***, ***noup***, ***nodown***, ***noout***, ***noin***, ***nobackfill***, ***norecover***, ***noscrub***, ***nodeep-scrub***
- *带有* ***true*** *或* ***false****的* ***VALUE***
- *使用用户名替代* ***USER***
- *使用带有活跃* ***ceph-mgr*** *实例的节点的 IP 地址或短主机名替换* ***CEPH_MANAGER***

提示时输入用户密码。

如果您使用自签名证书，请使用 ***--insecure*** *选项：*

```
echo -En '{"OPTION": VALUE}' | curl --request PATCH --data @- --silent --insecure --user USER
'https://CEPH_MANAGER:8080/api/osd/flags'
```

Python
在 Python 解释器中，输入：

```
$ python
>> import requests
>> result = requests.patch('https://CEPH_MANAGER:8080/api/osd/flags', json={"OPTION": VALUE},
auth=("USER", "PASSWORD"))
>> print result.json()
```

替换：

- *使用带有活跃* ***ceph-mgr*** *实例的节点的 IP 地址或短主机名替换* ***CEPH_MANAGER***
- *使用要修改的选项替换* ***OPTION***，***pause***, ***noup***, ***nodown***, ***noout***, ***noin***, ***nobackfill***, ***norecover***, ***noscrub***, ***nodeep-scrub***
- *使用* ***True*** *或* ***False*** *作为* ***VALUE***
- *使用用户名替代* ***USER***
- *使用用户的密码替换* ***PASSWORD***

如果您使用自签名证书，请使用 ***verify=False*** *选项：*

```
$ python
>> import requests
>> result = requests.patch('https://CEPH_MANAGER:8080/api/osd/flags', json={"OPTION": VALUE},
auth=("USER", "PASSWORD"), verify=False)
>> print result.json()
```

1.4.2.2. 如何更改 OSD 状态？

本节介绍如何使用 RESTful 插件更改 OSD 的状态。

curl 命令
在命令行中使用：

```
echo -En '{"STATE": VALUE}' | curl --request PATCH --data @- --silent --user USER
'https://CEPH_MANAGER:8080/api/osd/ID'
```

替换：

- *使用要改变的状态（**in** 或 **up**）替换 **STATE***
- *带有 **true** 或 **false**的 **VALUE***
- *使用用户名替代 **USER***
- *使用带有活跃 **ceph-mgr** 实例的节点的IP地址或短主机名替换 **CEPH_MANAGER***
- *使用 **osd** 字段中列出的OSD 的ID 替换 **ID***

提示时输入用户密码。

*如果您使用自签名证书，请使用 **--insecure** 选项：*

```
echo -En '{"STATE": VALUE}' | curl --request PATCH --data @- --silent --insecure --user USER
'https://CEPH_MANAGER:8080/api/osd/ID'
```

Python
在 Python 解释器中，输入：

```
$ python
>> import requests
>> result = requests.patch('https://CEPH_MANAGER:8080/api/osd/ID', json={"STATE": VALUE},
auth=("USER", "PASSWORD"))
>> print result.json()
```

替换：

- *使用带有活跃 **ceph-mgr** 实例的节点的IP地址或短主机名替换 **CEPH_MANAGER***
- *使用 **osd** 字段中列出的OSD 的ID 替换 **ID***
- *使用要改变的状态（**in** 或 **up**）替换 **STATE***
- *使用 **True** 或 **False** 作为 **VALUE***
- *使用用户名替代 **USER***
- *使用用户的密码替换 **PASSWORD***

*如果您使用自签名证书，请使用 **verify=False** 选项：*

```
$ python
>> import requests
>> result = requests.patch('https://CEPH_MANAGER:8080/api/osd/ID', json={"STATE": VALUE},
auth=("USER", "PASSWORD"), verify=False)
>> print result.json()
```

1.4.2.3. 如何重新加权 OSD？

本节论述了如何更改 OSD 的权重。

curl 命令
在命令行中使用：

```
echo -En '{"reweight": VALUE}' | curl --request PATCH --data @- --silent --user USER
'https://CEPH_MANAGER:8080/api/osd/ID'
```

替换：

- *带有新权重的 **VALUE***
- *使用用户名替代 **USER***
- *使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER***
- *使用 **osd** 字段中列出的 OSD 的 ID 替换 **ID***

提示时输入用户密码。

*如果您使用自签名证书，请使用 **--insecure** 选项：*

```
echo -En '{"reweight": VALUE}' | curl --request PATCH --data @- --silent --insecure --user USER
'https://CEPH_MANAGER:8080/api/osd/ID'
```

Python
在 Python 解释器中，输入：

```
$ python
>> import requests
>> result = requests.patch('https://CEPH_MANAGER:8080/osd/ID', json={"reweight": VALUE}, auth=
("USER", "PASSWORD"))
>> print result.json()
```

替换：

- *使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER***
- *使用 **osd** 字段中列出的 OSD 的 ID 替换 **ID***
- *带有新权重的 **VALUE***
- *使用用户名替代 **USER***
- *使用用户的密码替换 **PASSWORD***

*如果您使用自签名证书，请使用 **verify=False** 选项：*

```
$ python
>> import requests
>> result = requests.patch('https://CEPH_MANAGER:8080/api/osd/ID', json={"reweight": VALUE},
auth=("USER", "PASSWORD"), verify=False)
>> print result.json()
```

1.4.2.4. 如何更改池的信息？

这部分论述了如何使用 RESTful 插件更改特定池的信息。

curl 命令
在命令行中使用：

```
echo -En '{"OPTION": VALUE}' | curl --request PATCH --data @- --silent --user USER
'https://CEPH_MANAGER:8080/api/pool/ID'
```

替换：

- *使用要修改的选项替换* ***OPTION***
- *使用选项的新值替换* ***VALUE***
- *使用用户名替代* ***USER***
- *使用带有活跃* ***ceph-mgr*** *实例的节点的 IP 地址或短主机名替换* ***CEPH_MANAGER***
- *使用* ***pool*** *字段中列出的池的 ID 替换* ***ID***

提示时输入用户密码。

如果您使用自签名证书，请使用 ***--insecure*** *选项：*

```
echo -En '{"OPTION": VALUE}' | curl --request PATCH --data @- --silent --insecure --user USER
'https://CEPH_MANAGER:8080/api/pool/ID'
```

Python
在 Python 解释器中，输入：

```
$ python
>> import requests
>> result = requests.patch('https://CEPH_MANAGER:8080/api/pool/ID', json={"OPTION": VALUE},
auth=("USER, "PASSWORD"))
>> print result.json()
```

替换：

- *使用带有活跃* ***ceph-mgr*** *实例的节点的 IP 地址或短主机名替换* ***CEPH_MANAGER***
- *使用* ***pool*** *字段中列出的池的 ID 替换* ***ID***
- *使用要修改的选项替换* ***OPTION***
- *使用选项的新值替换* ***VALUE***
- *使用用户名替代* ***USER***
- *使用用户的密码替换* ***PASSWORD***

如果您使用自签名证书，请使用 ***verify=False*** *选项：*

```
$ python
>> import requests
>> result = requests.patch('https://CEPH_MANAGER:8080/api/pool/ID', json={"OPTION": VALUE},
```

```
auth=("USER, "PASSWORD"), verify=False)
>> print result.json()
```

1.4.3. 管理集群

本节介绍如何使用 Ceph API 在 OSD 上初始化清理或深度清理，创建池或从池中移除数据、删除请求或创建请求。

1.4.3.1. 如何在 OSD 上运行调度的进程？

本节论述了如何使用 RESTful API 在 OSD 上运行调度的进程，如清理或深度清理。

curl 命令
在命令行中使用：

```
echo -En '{"command": "COMMAND"}' | curl --request POST --data @- --silent --user USER
'https://CEPH_MANAGER:8080/api/osd/ID/command'
```

替换：

- 带有要启动的进程(**清理、深度清理 或修复)的 COMMAND**。验证 OSD 上支持该进程。详情请查看 第 1.4.1.9 节 "如何确定 OSD 上可以调度哪些进程？"。
- 使用用户名替代 **USER**
- 使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER**
- 使用 **osd** 字段中列出的 OSD 的 ID 替换 **ID**

提示时输入用户密码。

如果您使用自签名证书，请使用 **--insecure** 选项：

```
echo -En '{"command": "COMMAND"}' | curl --request POST --data @- --silent --insecure --user
USER 'https://CEPH_MANAGER:8080/api/osd/ID/command'
```

Python
在 Python 解释器中，输入：

```
$ python
>> import requests
>> result = requests.post('https://CEPH_MANAGER:8080/api/osd/ID/command', json={"command":
"COMMAND"}, auth=("USER", "PASSWORD"))
>> print result.json()
```

替换：

- 使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER**
- 使用 **osd** 字段中列出的 OSD 的 ID 替换 **ID**
- 带有要启动的进程(**清理、深度清理 或修复)的 COMMAND**。验证 OSD 上支持该进程。详情请查看 第 1.4.1.9 节 "如何确定 OSD 上可以调度哪些进程？"。
- 使用用户名替代 **USER**

- 使用用户的密码替换 **PASSWORD**

如果您使用自签名证书，请使用 **verify=False** 选项：

```
$ python
>> import requests
>> result = requests.post('https://CEPH_MANAGER:8080/api/osd/ID/command', json={"command":
"COMMAND"}, auth=("USER", "PASSWORD"), verify=False)
>> print result.json()
```

1.4.3.2. 如何创建新池？

这部分论述了如何使用 RESTful 插件创建新池。

curl 命令
在命令行中使用：

```
echo -En '{"name": "NAME", "pg_num": NUMBER}' | curl --request POST --data @- --silent --user
USER 'https://CEPH_MANAGER:8080/api/pool'
```

替换：

- 使用新池的名称替换 **NAME**
- 使用放置组数量替换 **NUMBER**
- 使用用户名替代 **USER**
- 使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER**

提示时输入用户密码。

如果您使用自签名证书，请使用 **--insecure** 选项：

```
echo -En '{"name": "NAME", "pg_num": NUMBER}' | curl --request POST --data @- --silent --
insecure --user USER 'https://CEPH_MANAGER:8080/api/pool'
```

Python
在 Python 解释器中，输入：

```
$ python
>> import requests
>> result = requests.post('https://CEPH_MANAGER:8080/api/pool', json={"name": "NAME",
"pg_num": NUMBER}, auth=("USER", "PASSWORD"))
>> print result.json()
```

替换：

- 使用带有活跃 **ceph-mgr** 实例的节点的 IP 地址或短主机名替换 **CEPH_MANAGER**
- 使用新池的名称替换 **NAME**
- 使用放置组数量替换 **NUMBER**
- 使用用户名替代 **USER**

- *使用用户的密码替换* ***PASSWORD***

如果您使用自签名证书，请使用 ***verify=False*** *选项：*

```
$ python
>> import requests
>> result = requests.post('https://CEPH_MANAGER:8080/api/pool', json={"name": "NAME",
"pg_num": NUMBER}, auth=("USER", "PASSWORD"), verify=False)
>> print result.json()
```

1.4.3.3. 如何删除池？

这部分论述了如何使用 RESTful 插件删除池。

默认情况下禁止此请求。若要允许它，请将以下参数添加到 Ceph 配置指南中。

```
mon_allow_pool_delete = true
```

curl 命令
在命令行中使用：

```
curl --request DELETE --silent --user USER 'https://CEPH_MANAGER:8080/api/pool/ID'
```

替换：

- *使用用户名替代* ***USER***
- *使用带有活跃* ***ceph-mgr*** *实例的节点的 IP 地址或短主机名替换* ***CEPH_MANAGER***
- *使用* ***pool*** *字段中列出的池的 ID 替换* ***ID***

提示时输入用户密码。

如果您使用自签名证书，请使用 ***--insecure*** *选项：*

```
curl --request DELETE --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/pool/ID'
```

Python
在 Python 解释器中，输入：

```
$ python
>> import requests
>> result = requests.delete('https://CEPH_MANAGER:8080/api/pool/ID', auth=("USER",
"PASSWORD"))
>> print result.json()
```

替换：

- *使用带有活跃* ***ceph-mgr*** *实例的节点的 IP 地址或短主机名替换* ***CEPH_MANAGER***
- *使用* ***pool*** *字段中列出的池的 ID 替换* ***ID***
- *使用用户名替代* ***USER***

- *使用用户的密码替换* ***PASSWORD***

如果您使用自签名证书，请使用 ***verify=False*** *选项：*

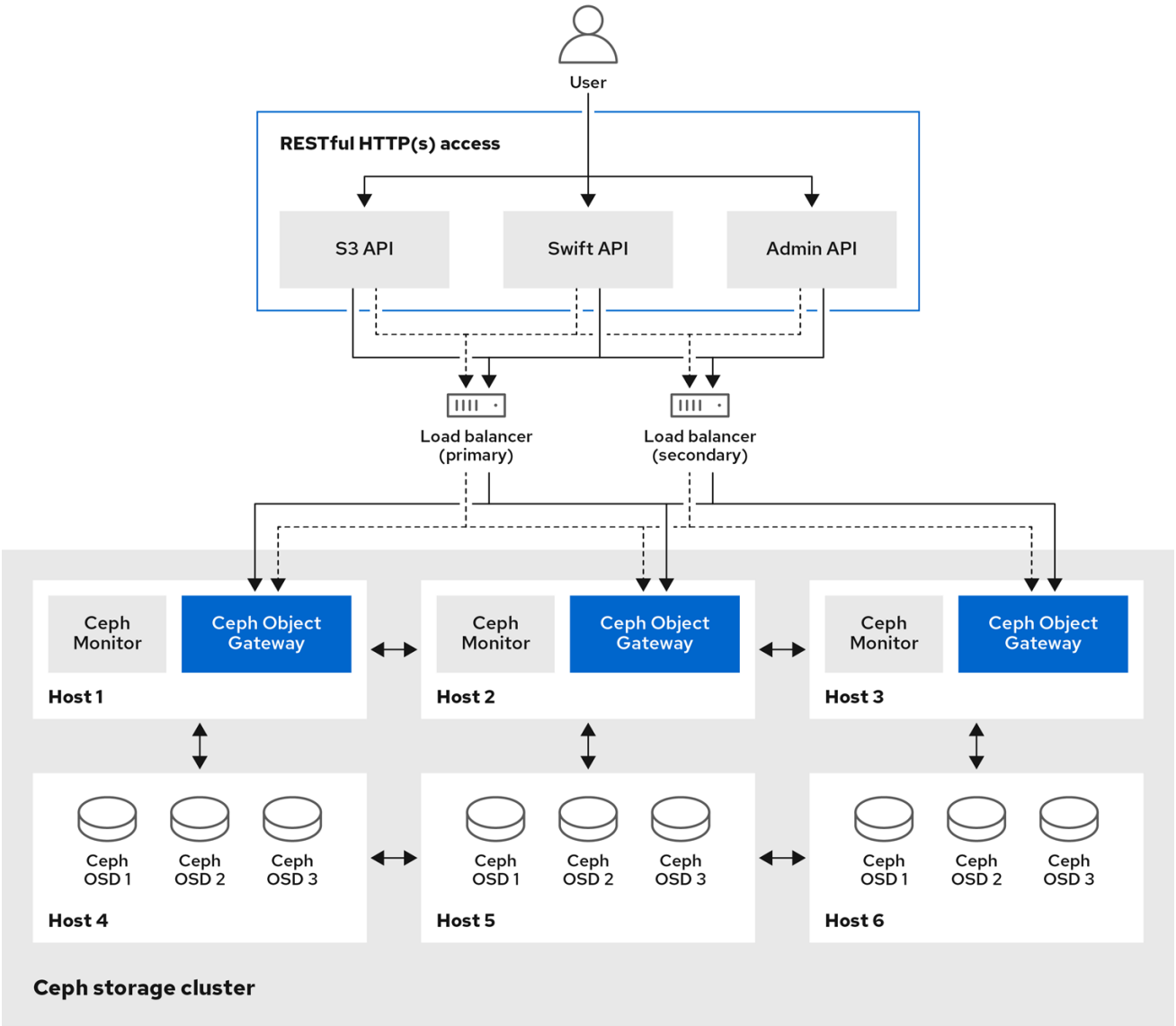
```
$ python
>> import requests
>> result = requests.delete('https://CEPH_MANAGER:8080/api/pool/ID', auth=("USER",
"PASSWORD"), verify=False)
>> print result.json()
```

第 2 章 CEPH 对象网关管理 API

作为开发者，您可以通过与 RESTful 应用程序编程接口(API)交互来管理 Ceph 对象网关。Ceph 对象网关在 RESTful API 中提供了 ***radosgw-admin*** *命令的功能。您可以管理用户、数据、配额和使用，并可与其他管理平台集成。*

注意

红帽建议在配置 Ceph 对象网关时使用命令行界面。



管理 API 提供以下功能：

- *身份验证请求*
- ***用户帐户管理***
  - *管理用户*
  - *获取用户信息*

- 创建
- 修改
- 删除
- 创建子用户
- 修改子用户
- 删除子用户

- **用户功能管理**
  - 添加
  - 删除
- **密钥管理**
  - 创建
  - 删除
- **bucket 管理**
  - 获取 Bucket 信息
  - 检查索引
  - 删除
  - 链接
  - 取消链接
  - policy
- **对象管理**
  - 删除
  - policy
- **配额管理**
  - 获取用户
  - 设置用户
  - 获取 Bucket
  - 设置 Bucket
- **获取使用信息**
- **删除使用信息**
- **标准错误响应**

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- RESTful 客户端。

2.1. 管理操作

管理应用程序编程接口(API)请求将在以可配置的"admin"资源入口点开头的 URI 上进行。管理 API 的授权重复了 S3 的授权机制。有些操作需要用户具有特殊的管理功能。响应实体类型(XML 或 JSON)可能会指定为请求中的 'format' 选项，如果未指定，则默认为 JSON。

示例

```
PUT /admin/user?caps&format=json HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
Content-Type: text/plain
Authorization: AUTHORIZATION_TOKEN

usage=read
```

2.2. 管理身份验证请求

Amazon 的 S3 服务使用访问密钥和请求标头的哈希和 secret 密钥来验证请求。它有提供经过身份验证的请求的好处，特别是进行大型上传，而无需 SSL 开销。

S3 API 的大部分用例都涉及使用开源 S3 客户端，如 Amazon SDK for Java 或 Python Boto 的 **AmazonS3Client**。这些库不支持 Ceph 对象网关管理员 API。您可以使用子类并扩展这些库来支持 Ceph Admin API。或者，您可以创建唯一的网关客户端。

创建 execute （） 方法

本节中的 CephAdminAPI 示例类演示了如何创建能够获取请求参数的 **execute()** 方法，验证请求，调用 Ceph Admin API 并收到响应。

CephAdminAPI 类示例不被支持，不适用于商业用途。它只用于演示目的。

调用 Ceph 对象网关

client code 包含对 Ceph 对象网关的五个调用，以演示 CRUD 操作：

- 创建用户
- 获取用户
- 修改用户
- 创建子用户
- 删除用户

要使用此示例，请获取 **httpcomponents-client-4.5.3** Apache HTTP 组件。您可以在此处下载该文件：http://hc.apache.org/downloads.cgi。然后解压缩 tar 文件，进入其 **lib** 目录，并将内容复制到 **JAVA_HOME** 目录的 **/jre/lib/ext** 目录中，或自定义 classpath。

当您检查 CephAdminAPI 类示例时，请注意 **execute()** 采用 HTTP 方法、请求路径、可选的子资源、**null**（如果未指定），以及参数映射。要使用子资源（如 **subuser** 和 **key**）执行，您需要在 **execute （）** 方法中指定子资源作为参数。

示例方法：

1. 构建 URI。
2. 构建 HTTP 标头字符串。
3. 实例化 HTTP 请求，如 **PUT**、**POST**、**GET**、**DELETE**。
4. 在 HTTP 标头字符串和请求标头中添加 **Date** 标头。
5. 在 HTTP 请求标头中添加 **Authorization** 标头。
6. 实例化 HTTP 客户端，并传递实例化的 HTTP 请求。
7. 发出请求。
8. 返回响应。

构建标头字符串

构建标头字符串是涉及 Amazon 的 S3 身份验证过程的过程的一部分。具体来说，示例方法如下：

1. 添加请求类型，如 **PUT**、**POST**、**GET** 和**DELETE**。
2. 添加日期。
3. 添加 requestPath。

请求类型应该为大写，没有前导或尾随空格。如果没有修剪空格，身份验证将失败。日期需要 以 GMT 表示，否则身份验证将失败。

exemplary 方法没有任何其他标头。Amazon S3 身份验证流程按 **x-amz** 标头顺序排序。因此，如果您要添加 **x-amz** 标头，确保以字典顺序添加它们。

构建了标头字符串后，下一步是实例化 HTTP 请求并传递 URI。exemplary 方法使用 **PUT** 来创建用户和子用户，**GET** 获取用户，**POST** 用于修改用户和 **DELETE** 删除用户。

实例化了一个请求后，添加 **Date** 标头后跟 **Authorization** 标头。Amazon 的 S3 身份验证使用标准 **Authorization** 标头，且结构如下：

```
Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

CephAdminAPI 示例类具有 **base64Sha1Hmac （）** 方法，它使用标头字符串和 admin 用户的 secret 密钥，并将 SHA1 HMAC 返回为 base-64 编码字符串。每个 **execute()** 调用都将调用同一行代码来构建 **Authorization** 标头：

```
httpRequest.addHeader("Authorization", "AWS " + this.getAccessKey() + ":" +
base64Sha1Hmac(headerString.toString(), this.getSecretKey()));
```

以下 **CephAdminAPI** 示例类要求您将访问密钥、secret 密钥和端点传递给构造器。类提供在运行时更改它们的访问者方法。

示例

```
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.time.OffsetDateTime;
import java.time.format.DateTimeFormatter;
import java.time.ZoneId;

import org.apache.http.HttpEntity;
import org.apache.http.NameValuePair;
import org.apache.http.Header;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpRequestBase;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.methods.HttpPut;
import org.apache.http.client.methods.HttpDelete;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.message.BasicNameValuePair;
import org.apache.http.util.EntityUtils;
import org.apache.http.client.utils.URIBuilder;

import java.util.Base64;
import java.util.Base64.Encoder;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import javax.crypto.spec.SecretKeySpec;
import javax.crypto.Mac;

import java.util.Map;
import java.util.Iterator;
import java.util.Set;
import java.util.Map.Entry;

public class CephAdminAPI {

 /*
  * Each call must specify an access key, secret key, endpoint and format.
  */
 String accessKey;
 String secretKey;
 String endpoint;
 String scheme = "http"; //http only.
 int port = 80;

 /*
  * A constructor that takes an access key, secret key, endpoint and format.
  */
 public CephAdminAPI(String accessKey, String secretKey, String endpoint){
  this.accessKey = accessKey;
  this.secretKey = secretKey;
  this.endpoint = endpoint;
 }

 /*
```

```
 * Accessor methods for access key, secret key, endpoint and format.
 */
public String getEndpoint(){
 return this.endpoint;
}

public void setEndpoint(String endpoint){
 this.endpoint = endpoint;
}

public String getAccessKey(){
 return this.accessKey;
}

public void setAccessKey(String accessKey){
 this.accessKey = accessKey;
}

public String getSecretKey(){
 return this.secretKey;
}

public void setSecretKey(String secretKey){
 this.secretKey = secretKey;
}

/*
 * Takes an HTTP Method, a resource and a map of arguments and
 * returns a CloseableHTTPResponse.
 */
public CloseableHttpResponse execute(String HTTPMethod, String resource,
                                     String subresource, Map arguments) {

 String httpMethod = HTTPMethod;
 String requestPath = resource;
 StringBuffer request = new StringBuffer();
 StringBuffer headerString = new StringBuffer();
 HttpRequestBase httpRequest;
 CloseableHttpClient httpclient;
 URI uri;
 CloseableHttpResponse httpResponse = null;

 try {

  uri = new URIBuilder()
   .setScheme(this.scheme)
   .setHost(this.getEndpoint())
   .setPath(requestPath)
   .setPort(this.port)
   .build();


  if (subresource != null){
   uri = new URIBuilder(uri)
    .setCustomQuery(subresource)
    .build();
```

```
 }


 for (Iterator iter = arguments.entrySet().iterator();
 iter.hasNext();) {
  Entry entry = (Entry)iter.next();
  uri = new URIBuilder(uri)
   .setParameter(entry.getKey().toString(),
                 entry.getValue().toString())
   .build();

 }

 request.append(uri);

 headerString.append(HTTPMethod.toUpperCase().trim() + "\n\n\n");

 OffsetDateTime dateTime = OffsetDateTime.now(ZoneId.of("GMT"));
 DateTimeFormatter formatter = DateTimeFormatter.RFC_1123_DATE_TIME;
 String date = dateTime.format(formatter);

 headerString.append(date + "\n");
 headerString.append(requestPath);

 if (HTTPMethod.equalsIgnoreCase("PUT")){
  httpRequest = new HttpPut(uri);
 } else if (HTTPMethod.equalsIgnoreCase("POST")){
  httpRequest = new HttpPost(uri);
 } else if (HTTPMethod.equalsIgnoreCase("GET")){
  httpRequest = new HttpGet(uri);
 } else if (HTTPMethod.equalsIgnoreCase("DELETE")){
  httpRequest = new HttpDelete(uri);
 } else {
  System.err.println("The HTTP Method must be PUT,
  POST, GET or DELETE.");
  throw new IOException();
 }

 httpRequest.addHeader("Date", date);
 httpRequest.addHeader("Authorization", "AWS " + this.getAccessKey()
 + ":" + base64Sha1Hmac(headerString.toString(),
 this.getSecretKey()));

 httpclient = HttpClients.createDefault();
 httpResponse = httpclient.execute(httpRequest);

}  catch  (URISyntaxException e){
 System.err.println("The URI is not formatted properly.");
 e.printStackTrace();
}  catch (IOException e){
 System.err.println("There was an error making the request.");
 e.printStackTrace();
}
 return httpResponse;
}
```

```
 /*
  * Takes a uri and a secret key and returns a base64-encoded
  * SHA-1 HMAC.
  */
 public String base64Sha1Hmac(String uri, String secretKey) {
  try {

   byte[] keyBytes = secretKey.getBytes("UTF-8");
   SecretKeySpec signingKey = new SecretKeySpec(keyBytes, "HmacSHA1");

   Mac mac = Mac.getInstance("HmacSHA1");
   mac.init(signingKey);

   byte[] rawHmac = mac.doFinal(uri.getBytes("UTF-8"));

   Encoder base64 = Base64.getEncoder();
   return base64.encodeToString(rawHmac);

  } catch (Exception e) {
   throw new RuntimeException(e);
  }
 }

}
```

*后续 **CephAdminAPIClient** 示例演示了如何实例化 **CephAdminAPI** 类、构建请求参数映射，并使用 **execute （）** 方法来创建、获取、更新和删除用户。*

示例

```
import java.io.IOException;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.HttpEntity;
import org.apache.http.util.EntityUtils;
import java.util.*;


public class CephAdminAPIClient {

 public static void main (String[] args){

  CephAdminAPI adminApi = new CephAdminAPI ("FFC6ZQ6EMIF64194158N",
                          "Xac39eCAhlTGcCAUreuwe1ZuH5oVQFa51lbEMVoT",
                          "ceph-client");

  /*
   * Create a user
   */
  Map requestArgs = new HashMap();
  requestArgs.put("access", "usage=read, write; users=read, write");
  requestArgs.put("display-name", "New User");
  requestArgs.put("email", "new-user@email.com");
  requestArgs.put("format", "json");
  requestArgs.put("uid", "new-user");

  CloseableHttpResponse response =
```

```
 adminApi.execute("PUT", "/admin/user", null, requestArgs);

System.out.println(response.getStatusLine());
HttpEntity entity = response.getEntity();

try {
 System.out.println("\nResponse Content is: "
  + EntityUtils.toString(entity, "UTF-8") + "\n");
 response.close();
} catch (IOException e){
 System.err.println ("Encountered an I/O exception.");
 e.printStackTrace();
}

/*
 * Get a user
 */
requestArgs = new HashMap();
requestArgs.put("format", "json");
requestArgs.put("uid", "new-user");

response = adminApi.execute("GET", "/admin/user", null, requestArgs);

System.out.println(response.getStatusLine());
entity = response.getEntity();

try {
 System.out.println("\nResponse Content is: "
  + EntityUtils.toString(entity, "UTF-8") + "\n");
 response.close();
} catch (IOException e){
 System.err.println ("Encountered an I/O exception.");
 e.printStackTrace();
}

/*
 * Modify a user
 */
requestArgs = new HashMap();
requestArgs.put("display-name", "John Doe");
requestArgs.put("email", "johndoe@email.com");
requestArgs.put("format", "json");
requestArgs.put("uid", "new-user");
requestArgs.put("max-buckets", "100");

response = adminApi.execute("POST", "/admin/user", null, requestArgs);

System.out.println(response.getStatusLine());
entity = response.getEntity();

try {
 System.out.println("\nResponse Content is: "
  + EntityUtils.toString(entity, "UTF-8") + "\n");
 response.close();
} catch (IOException e){
 System.err.println ("Encountered an I/O exception.");
```

```
 e.printStackTrace();
}


/*
 * Create a subuser
 */
requestArgs = new HashMap();
requestArgs.put("format", "json");
requestArgs.put("uid", "new-user");
requestArgs.put("subuser", "foobar");

response = adminApi.execute("PUT", "/admin/user", "subuser", requestArgs);
System.out.println(response.getStatusLine());
entity = response.getEntity();

try {
 System.out.println("\nResponse Content is: "
  + EntityUtils.toString(entity, "UTF-8") + "\n");
 response.close();
} catch (IOException e){
 System.err.println ("Encountered an I/O exception.");
 e.printStackTrace();
}


/*
 * Delete a user
 */
requestArgs = new HashMap();
requestArgs.put("format", "json");
requestArgs.put("uid", "new-user");

response = adminApi.execute("DELETE", "/admin/user", null, requestArgs);
System.out.println(response.getStatusLine());
entity = response.getEntity();

try {
 System.out.println("\nResponse Content is: "
  + EntityUtils.toString(entity, "UTF-8") + "\n");
 response.close();
} catch (IOException e){
 System.err.println ("Encountered an I/O exception.");
 e.printStackTrace();
 }
 }
}
```

其它资源

- *如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 S3 身份验证 部分。*
- *有关 Amazon S3 身份验证流程的更广泛的解释，请参阅 Amazon Simple Storage Service 文档中的 Signing and Authenticating REST Requests 部分。*

2.3. 创建管理用户

重要

要从 Ceph 对象网关节点运行 **radosgw-admin** 命令，请确保节点具有 admin 密钥。admin 密钥可以从任何 Ceph 监控节点复制。

先决条件

- Ceph 对象网关节点的根级别访问权限.

流程

1. 创建对象网关用户：

   语法

   ```
   radosgw-admin user create --uid="USER_NAME" --display-name="DISPLAY_NAME"
   ```

   示例

   ```
   [user@client ~]$ radosgw-admin user create --uid="admin-api-user" --display-name="Admin API User"
   ```

   radosgw-admin 命令行界面将返回用户。

   输出示例

   ```
   {
       "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": "gr1VEGIV7rxcP3xvXDFCo4UDwwl2YoNrmtRIIAty"
           }
       ],
       "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. *为您创建的用户分配管理功能：*

语法

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

示例

```
[user@client ~]$ radosgw-admin caps add --uid=admin-api-user --caps="users=*"
```

***radosgw-admin** 命令行界面将返回用户。**"caps":** 将具有您分配给用户的能力：*

输出示例

```
{
    "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": []
    }
```

现在您有具有管理特权的用户。

2.4. 获取用户信息

获取用户信息。必须将 **users** 或 **user-info-without-keys** 的 cap 设置为 **read** 才能运行此操作。如果将 cap **user-info-without-keys** 设置为 **read** 或 *，则 S3 密钥和 Swift 密钥不会包含在响应中，除非运行此操作的用户是系统用户、管理员用户，或者将 cap 用户设置为 **read**。

功能

```
users=read or user-info-without-keys=read
```

语法

```
GET /admin/user?format=json HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
```

请求参数

uid

描述

请求信息的用户。

Type

字符串

示例

foo_user

必需

是

access-key

描述

请求信息的用户的 S3 访问密钥。

Type

字符串

示例

ABCD0EF12GHIJ2K34LMN

必需

否

响应实体

user

描述

用于用户数据信息的容器。

Type

Container

父

不适用

user_id

描述

用户 ID。

Type

字符串

父

user

display_name

描述

显示用户的名称。

Type

字符串

父

user

suspended

描述

如果用户已被挂起，则为 true。

Type

布尔值

父

user

max_buckets

描述

用户所有的 bucket 的最大数量。

Type

整数

父

user

子用户

描述

与此用户帐户关联的子用户。

Type

Container

父

user

keys

描述

与这个用户帐户关联的 S3 密钥。

Type

Container

父

user

swift_keys

描述

与此用户帐户关联的 Swift 密钥。

Type

Container

父

user

caps

描述

用户能力。

Type

Container

父

user

如果成功，响应将包含用户信息。

特殊错误响应

无。

2.5. 创建用户

创建新用户。默认情况下，将自动创建 S3 密钥对并在响应中返回。如果只提供了 access-key 或 secret-key，则忽略的密钥会自动生成。默认情况下，生成的密钥会添加到密钥环中，而不替换现有的密钥对。如果指定了 access-key，并引用用户拥有的现有密钥，则会修改它。

功能

> `*users=write*`

语法

```
PUT /admin/user?format=json HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
```

请求参数

uid

描述

要创建的用户 ID。

Type

字符串

示例

foo_user

必需

是

display-name

描述

要创建的用户的显示名称。

Type

字符串

示例

foo_user

必需

是

email

描述

与用户关联的电子邮件地址。

Type

字符串

示例

foo@bar.com

必需

否

key-type

描述

*要生成的密钥类型，选项为：**swift**、**s3**（默认）。*

Type

字符串

示例

s3 [s3]

必需

否

access-key

描述

指定访问密钥。

Type

字符串

示例

ABCD0EF12GHIJ2K34LMN

必需

否

secret-key

描述

*指定 **secret** 密钥。*

Type

字符串

示例

0AbCDEFg1h2i34JkIM5nop6QrSTUV+WxyzaBC7D8

必需

否

user-caps

描述

用户能力。

Type

字符串

示例

usage=read, write; users=read

必需

否

generate-key

描述

生成新密钥对并添加到现有密钥环中。

Type

布尔值

示例

True [True]

必需

否

max-buckets

描述

指定用户可以拥有的最大存储桶数。

Type

整数

示例

500 [1000]

必需

否

suspended

描述

指定是否应该暂停用户

Type

布尔值

示例

false [False]

必需

否

响应实体

user

描述

指定是否应该暂停用户

Type

布尔值

父

否

user_id

描述

用户 ID。

Type

字符串

父

user

display_name

描述

显示用户的名称。

Type

字符串

父

user

suspended

描述

如果用户已被挂起，则为 true。

Type

布尔值

父

user

max_buckets

描述

用户所有的 bucket 的最大数量。

Type

整数

父

user

子用户

描述

与此用户帐户关联的子用户。

Type

Container

父

user

keys

描述

与这个用户帐户关联的 S3 密钥。

Type

Container

父

user

swift_keys

描述

与此用户帐户关联的 Swift 密钥。

Type

Container

父

user

caps

描述

用户能力。

Type

Container

父

如果成功，响应将包含用户信息。

特殊错误响应

UserExists

描述

尝试创建现有用户。

代码

409 冲突

InvalidAccessKey

描述

指定无效的访问密钥。

代码

400 错误请求

InvalidKeyType

描述

指定了无效的密钥类型。

代码

400 错误请求

InvalidSecretKey

描述

指定了无效的 secret key。

代码

400 错误请求

KeyExists

描述

提供的访问密钥存在，但属于另一用户。

代码

409 冲突

EmailExists

描述

提供的电子邮件地址已存在。

代码

409 冲突

InvalidCap

描述

尝试授予无效的 admin 功能。

代码

400 错误请求

其它资源

- 有关创建子用户，请参阅 Red Hat Ceph Storage 开发人员指南。

2.6. 修改用户

修改现有用户。

功能

> `users=write`

语法

```
POST /admin/user?format=json HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
```

请求参数

uid

描述

要创建的用户 ID。

Type

字符串

示例

foo_user

必需

是

display-name

描述

要创建的用户的显示名称。

Type

字符串

示例

foo_user

必需

是

email

描述

与用户关联的电子邮件地址。

Type

字符串

示例

foo@bar.com

必需

否

generate-key

描述

生成新密钥对并添加到现有密钥环中。

Type

布尔值

示例

True [False]

必需

否

access-key

描述

指定访问密钥。

Type

字符串

示例

ABCD0EF12GHIJ2K34LMN

必需

否

secret-key

描述

指定 secret 密钥。

Type

字符串

示例

0AbCDEFg1h2i34JklM5nop6QrSTUV+WxyzaBC7D8

必需

否

key-type

描述

要生成的密钥类型，选项为：swift、s3（默认）。

Type

字符串

示例

s3

必需

否

user-caps

描述

用户能力。

Type

字符串

示例

usage=read, write; users=read

必需

否

max-buckets

描述

指定用户可以拥有的最大存储桶数。

Type

整数

示例

500 [1000]

必需

否

suspended

描述

指定是否应该暂停用户

Type

布尔值

示例

false [False]

必需

否

响应实体

user

描述

指定是否应该暂停用户

Type

布尔值

父

否

user_id

描述

用户 ID。

Type

字符串

父

user

display_name

描述

显示用户的名称。

Type

字符串

父

user

suspended

描述

*如果用户已被挂起，则为 **true**。*

Type

布尔值

父

user

max_buckets

描述

*用户所有的 **bucket** 的最大数量。*

Type

整数

父

user

子用户

描述

与此用户帐户关联的子用户。

Type

Container

父

user

keys

描述

与这个用户帐户关联的 S3 密钥。

Type

Container

父

user

swift_keys

描述

与此用户帐户关联的 Swift 密钥。

Type

Container

父

user

caps

描述

用户能力。

Type

Container

父

如果成功，响应将包含用户信息。

特殊错误响应

InvalidAccessKey

描述

指定无效的访问密钥。

代码

400 错误请求

InvalidKeyType

描述

指定了无效的密钥类型。

代码

400 错误请求

InvalidSecretKey

描述

指定了无效的 secret key。

代码

400 错误请求

KeyExists

描述

提供的访问密钥存在，但属于另一用户。

代码

409 冲突

EmailExists

描述

提供的电子邮件地址已存在。

代码

409 冲突

InvalidCap

描述

尝试授予无效的 admin 功能。

代码

400 错误请求

其它资源

- 有关修改子用户，请参阅 Red Hat Ceph Storage 开发人员指南。

2.7. 删除用户

移除现有用户。

功能

`users=write`

语法

```
DELETE /admin/user?format=json HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
```

请求参数

uid

描述

要删除的用户 **ID**。

Type

字符串

示例

foo_user

必需

是

purge-data

描述

指定属于用户的 **bucket** 和对象时，也会被删除。

Type

布尔值

示例

true

必需

否

响应实体

无。

特殊错误响应

无。

其它资源

- 有关删除子用户，请参阅 Red Hat Ceph Storage 开发人员指南。

2.8. 创建子用户

创建一个新的子用户，主要用于使用 Swift API 的客户端。

注意

有效请求需要 gen-subuser 或 subuser。通常，要使子用户很有用，必须通过指定 访问权限 来授予权限。与创建用户一样，如果在没有 secret 的情况下指定子用户，则会自动生成 secret 密钥。

功能

`users=write`

语法

```
PUT /admin/user?subuser&format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

请求参数

uid

描述

要创建子用户的用户 ID。

Type

字符串

示例

foo_user

必需

是

subuser

描述

指定要创建的子用户 ID。

Type

字符串

示例

sub_foo

必需

是（或 gen-subuser）

gen-subuser

描述

指定要创建的子用户 ID。

Type

字符串

示例

sub_foo

必需

是（或 gen-subuser）

secret-key

描述

指定 secret 密钥。

Type

字符串

示例

0AbCDEFg1h2i34JklM5nop6QrSTUV+WxyzaBC7D8

必需

否

key-type

描述

要生成的密钥类型，选项为：swift（默认）、s3。

Type

字符串

示例

swift [swift]

必需

否

access

描述

*为子用户设置访问权限，应为 **read, write, readwrite, full 之一。***

Type

字符串

示例

读取

必需

否

generate-secret

描述

生成 secret 密钥。

Type

布尔值

示例

True [False]

必需

否

*响应实**体***

子用户

描述

与用户帐户关联的子用户。

Type

Container

父

不适用

权限

描述

对用户帐户的子用户访问。

Type

字符串

父

子用户

如果成功，响应包含子用户信息。

特殊错误响应

SubuserExists

描述

存在指定的子用户。

代码

409 冲突

InvalidKeyType

描述

指定了无效的密钥类型。

代码

400 错误请求

InvalidSecretKey

描述

指定了无效的 secret key。

代码

400 错误请求

InvalidAccess

描述

指定了无效的子用户访问

代码

400 错误请求

2.9. 修改子用户

修改现有子用户。

功能

> `*users=write*`

语法

```
POST /admin/user?subuser&format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

请求参数

uid

描述

要创建子用户的用户 ID。

Type

字符串

示例

foo_user

必需

是

subuser

描述

要修改的子用户 ID。

Type

字符串

示例

sub_foo

必需

generate-secret

描述

为 subuser 生成新 secret 密钥，并替换现有的密钥。

Type

布尔值

示例

True [False]

必需

否

secret

描述

指定 secret 密钥。

Type

字符串

示例

0AbCDEFg1h2i34JklM5nop6QrSTUV+WxyzaBC7D8

必需

否

key-type

描述

要生成的密钥类型，选项为：swift（默认）、s3。

Type

字符串

示例

swift [swift]

必需

否

access

描述

为子用户设置访问权限，应为 read, write, readwrite, full 之一。

Type

字符串

示例

读取

必需

否

响应实体

子用户

描述

与用户帐户关联的子用户。

Type

Container

父

不适用

id

描述

子用户 ID

Type

字符串

父

子用户

权限

描述

对用户帐户的子用户访问。

Type

字符串

父

子用户

如果成功，响应包含子用户信息。

特殊错误响应

InvalidKeyType

描述

指定了无效的密钥类型。

代码

400 错误请求

InvalidSecretKey

描述

指定了无效的 secret key。

代码

400 错误请求

InvalidAccess

描述

指定了无效的子用户访问

代码

400 错误请求

2.10. 删除子用户

移除现有的子用户。

功能

```
`users=write`
```

语法

```
DELETE /admin/user?subuser&format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

请求参数

uid

描述

要删除的用户 ID。

Type

字符串

示例

foo_user

必需

是

subuser

描述

要删除的子用户 ID。

Type

字符串

示例

sub_foo

必需

是

purge-keys

描述

删除属于子用户的密钥。

Type

布尔值

示例

True [True]

必需

否

响应实体

无。

特殊错误响应

无。

2.11. 为用户添加功能

向指定用户添加管理权限。

功能

```
`users=write`
```

语法

```
PUT /admin/user?caps&format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

请求参数

uid

描述

要添加的管理功能的用户 ID。

Type

字符串

示例

foo_user

必需

是

user-caps

描述

添加用户的管理功能。

Type

字符串

示例

usage=read, write

必需

是

响应实体

响应实体

user

描述

用于用户数据信息的容器。

Type

Container

父

不适用

user_id

描述

用户 ID

Type

字符串

父

user

caps

描述

用户能力，

Type

Container

父

user

如果成功，响应将包含用户的功能。

特殊错误响应

InvalidCap

描述

尝试授予无效的 admin 功能。

代码

400 错误请求

2.12. 从用户中删除功能

从指定用户删除管理权限。

功能

```
`users=write`
```

语法

```
DELETE /admin/user?caps&format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

请求参数

uid

描述

从中删除管理功能的用户 ID。

Type

字符串

示例

foo_user

必需

是

user-caps

描述

从用户中删除的管理功能。

Type

字符串

示例

usage=read, write

必需

是

响应实体

user

描述

用于用户数据信息的容器。

Type

Container

父

不适用

user_id

描述

用户 ID。

Type

字符串

父

user

caps

描述

用户能力。

Type

Container

父

user

如果成功，响应将包含用户的功能。

特殊错误响应

InvalidCap

描述

尝试删除无效的管理员功能。

代码

400 错误请求

NoSuchCap

描述

用户没有指定的能力。

代码

404 not Found

2.13. 创建密钥

创建新密钥。如果指定了 subuser，那么默认创建的密钥将是 swift 类型。如果只提供一个 access-key 或 secret-key，则会自动生成提交的密钥，如果只指定了 secret-key，则会自动生成 access-key。默认情况下，生成的密钥会添加到密钥环中，而不替换现有的密钥对。如果指定了 access-key，并引用用户拥有的现有密钥，则会修改它。响应是一个容器，它列出了与创建的密钥相同的所有类型。

注意

在创建 swift 密钥时，指定 access-key 选项将无效。另外，每个用户或子用户只能保留一个 swift 密钥。

功能

> `users=write`

语法

```
PUT /admin/user?key&format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

请求参数

uid

描述

用于接收新密钥的用户 ID。

Type

字符串

示例

foo_user

必需

是

subuser

描述

用于接收新密钥的子用户 ID。

Type

字符串

示例

sub_foo

必需

否

key-type

描述

要生成的密钥类型，选项为：swift、s3（默认）。

Type

字符串

示例

s3 [s3]

必需

否

access-key

描述

指定访问密钥。

Type

字符串

示例

AB01C2D3EF45G6H7IJ8K

必需

否

secret-key

描述

指定 secret 密钥。

Type

字符串

示例

0ab/CdeFGhij1klmnopqRSTUv1WxyZabcDEFgHij

必需

否

generate-key

描述

生成新密钥对并添加到现有密钥环中。

Type

布尔值

示例

True [True]

必需

否

响应实体

keys

描述

与此用户帐户关联的类型密钥。

Type

Container

父

不适用

user

描述

与密钥关联的用户帐户。

Type

字符串

父

keys

access-key

描述

access key。

Type

字符串

父

keys

secret-key

描述

secret 密钥。

Type

字符串

父

keys

特殊错误响应

InvalidAccessKey

描述

指定无效的访问密钥。

代码

400 错误请求

InvalidKeyType

描述

指定了无效的密钥类型。

代码

400 错误请求

InvalidSecretKey

描述

指定了无效的 secret key。

代码

400 错误请求

InvalidKeyType

描述

指定了无效的密钥类型。

代码

400 错误请求

KeyExists

描述

提供的访问密钥存在，但属于另一用户。

代码

409 冲突

2.14. 删除密钥

删除现有密钥。

功能

`users=write`

语法

```
DELETE /admin/user?key&format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

请求参数

access-key

描述

属于 S3 密钥对的 S3 访问密钥，用于移除。

Type

字符串

示例

AB01C2D3EF45G6H7IJ8K

必需

是

uid

描述

从中删除该密钥的用户。

Type

字符串

示例

foo_user

必需

否

subuser

描述

从中删除该密钥的子用户。

Type

字符串

示例

sub_foo

必需

否

key-type

描述

要移除的关键类型，选项为：swift、s3。

注意

移除 swift 密钥是必需的。

Type

字符串

示例

swift

必需

否

特殊错误响应

无。

响应实体

无。

2.15. BUCKET 通知

作为存储管理员，您可以使用这些 API 为存储桶通知机制提供配置和部署接口。API 主题是命名的对象，其中包含特定端点的定义。bucket 通知将主题与特定 bucket 关联。S3 bucket operations 部分提供有关存储桶通知的更多详情。

注意

在所有主题操作中，参数都是 URL 编码，并使用 application/x-www-form-urlencoded 内容类型在消息正文中发送。

注意

需要重新创建与这个主题相关联的任何存储桶通知，以使主题更新生效。

先决条件

- 在 Ceph 对象网关上创建 bucket 通知。

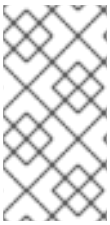
2.15.1. 存储桶通知概述

bucket 通知提供了一种方式，可以在 bucket 中发生特定事件时从 Ceph 对象网关发送信息。bucket 通知可以发送到 HTTP、AMQP0.9.1 和 Kafka 端点。必须创建一个通知条目，以便为特定存储桶上的事件和特定主题发送存储桶通知。可以在事件类型的子集上创建 bucket 通知，也可以默认为所有事件类型创建 bucket 通知。bucket 通知可以根据密钥前缀或后缀、匹配键的正则表达式以及附加到对象或对象标签的元数据属性过滤出事件。bucket 通知具有 REST API，用于为 bucket 通知机制提供配置和控制接口。

当对象同步到区域时，发送存储桶通知可让外部系统在对象级别获取信息到区域同步状态。当通过存储桶通知机制配置时，当通过存储桶通知机制配置时，存储桶通知事件类型 s3:ObjectSynced:Created，在成功同步对象时从同步 RGW 中发送通知事件。在发送通知事件的每个区域中，应单独执行主题和通知配置。

2.15.2. 持久性通知

永久通知支持可靠和异步将通知从 Ceph 对象网关传输到主题中配置的端点。常规通知也可靠，因为发送到端点的发送是在请求期间同步执行的。使用持久通知时，Ceph 对象网关即使端点停机或操作过程中存在网络问题，即使未成功传送到端点，也会重试发送通知。只有在与通知的操作相关的所有其他操作成功后，才会发送通知。如果端点在较长时间内停机，通知队列会填充，并且为这些端点配置了通知的 S3 操作将失败。

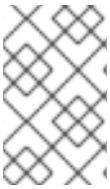


注意

使用 kafka-ack-level=none 时，没有指示消息失败，因此当代理再次上线时，发送的消息不会重试。再次设置代理后，只会看到新的通知。

2.15.3. 创建主题

您可以在创建存储桶通知前创建主题。主题是 Simple Notification Service (SNS)实体和所有主题操作，即 创建、删除、列出 和 get 是 SNS 操作。主题需要具有创建存储桶通知时使用的端点参数。请求成功后，响应中包含稍后可用来在存储桶通知请求中引用此主题的 Amazon Resource Name(ARN)主题。



注意

topic_arn 提供存储桶通知配置，并在创建主题后生成。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 根级别访问权限.
- 安装 Ceph 对象网关.
- 用户 access key 和 secret key。
- 端点参数.

流程

1. 创建带有请求格式的主题：

语法

```
POST
Action=CreateTopic
&Name=TOPIC_NAME
[&Attributes.entry.1.key=amqp-exchange&Attributes.entry.1.value=EXCHANGE]
[&Attributes.entry.2.key=amqp-ack-level&Attributes.entry.2.value=none|broker|routable]
[&Attributes.entry.3.key=verify-ssl&Attributes.entry.3.value=true|false]
[&Attributes.entry.4.key=kafka-ack-level&Attributes.entry.4.value=none|broker]
[&Attributes.entry.5.key=use-ssl&Attributes.entry.5.value=true|false]
[&Attributes.entry.6.key=ca-location&Attributes.entry.6.value=FILE_PATH]
```

```
[&Attributes.entry.7.key=OpaqueData&Attributes.entry.7.value=OPAQUE_DATA]
[&Attributes.entry.8.key=push-endpoint&Attributes.entry.8.value=ENDPOINT]
[&Attributes.entry.9.key=persistent&Attributes.entry.9.value=true|false]
```

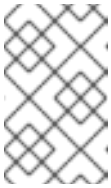
以下是请求参数：

- 端点 ：要将通知发送到的端点的 URL。
- OpaqueData: 不透明数据在主题配置中设置，并添加到该主题触发的所有通知。
- persistent ：指示对此端点的通知是否是持久的，是否为异步的。默认值为 false。
- HTTP 端点：
  - URL: https://FQDN:PORT
  - port defaults to: 分别为 HTTP[S] 使用 80/443。
  - verify-ssl ：指示服务器证书是否由客户端验证。默认情况下， 它是 true。
- AMQP0.9.1 端点：
  - URL: amqp://USER:PASSWORD@FQDN:PORT[/VHOST].
  - 用户和密码分别默认为： guest 和 guest。
  - 应当通过 HTTPS 提供用户和密码详细信息，否则主题创建请求将被拒绝。

- 端口默认为: 5672。
- vhost 默认为："/"
- amqp-exchange ：该交换必须存在，并且能够根据主题路由消息。这是用于 AMQP0.9.1 的强制参数。指向同一端点的不同主题必须使用相同的交换。
- AMQP-ack 级别: 不需要端到端确认，因为消息可能会在代理中被传送到最终目的地前保留。有三个确认方法：
  - none ：当发送到代理，则消息被视为已发送。
  - 代理 ：默认情况下，如果代理确认，则消息被视为 已发送。
  - routable ：如果代理可路由到使用者，则消息被视为已发送。

注意

特定参数的键和值不必驻留在同一行中，或者以任何特定顺序使用，但必须使用相同的索引。属性索引不需要是连续的，或从任何特定值开始。



注意

topic-name 用于 AMQP 主题。

- Kafka 端点：
  - URL: kafka://USER:PASSWORD@FQDN:PORT.
  - use-ssl 默认被设置为 false。如果将 use-ssl 设置为 true，则与代理的连接会使用安全的连接。

- 如果提供了 ca-location，并且使用安全连接，则会使用指定的 CA 而不是默认的端口来验证代理。
- 用户和密码只能通过 HTTP[S] 提供。否则，主题创建请求将被拒绝。
- 用户和密码只能与 use-ssl 一起提供，否则与代理的连接将失败。
- 端口默认为: 9092。
- Kafka-ack 级别 ：不需要结束确认，因为消息可能会在代理中被传送到最终目的地前保留。存在两种确认方法：
  - none ：当发送到代理，则消息被视为已发送。
  - 代理 ：默认情况下，如果代理确认，则消息被视为 已发送。

以下是响应格式的示例：

示例

```
<CreateTopicResponse xmlns="https://sns.amazonaws.com/doc/2010-03-31/">
    <CreateTopicResult>
        <TopicArn></TopicArn>
    </CreateTopicResult>
    <ResponseMetadata>
        <RequestId></RequestId>
    </ResponseMetadata>
</CreateTopicResponse>
```

注意

响应中的 Amazon 资源名称(ARN)将具有以下格式：
arn:aws:sns:ZONE_GROUP:TENANT:TOPIC

以下是一个 AMQP0.9.1 端点的示例：

示例

```
client.create_topic(Name='my-topic' , Attributes={'push-endpoint': 'amqp://127.0.0.1:5672', 'amqp-exchange': 'ex1', 'amqp-ack-level': 'broker'}) "
```

2.15.4. 获取主题信息

返回有关特定主题的信息。如果提供，这可包括端点信息。

先决条件

- *一个正在运行的 Red Hat Ceph Storage 集群。*
- *根级别访问权限.*
- *安装 Ceph 对象网关.*
- *用户 access key 和 secret key。*
- *端点参数.*

流程

1. *使用以下请求格式获取主题信息：*

语法

```
POST
Action=GetTopic
&TopicArn=TOPIC_ARN
```

以下是响应格式的示例：

```
<GetTopicResponse>
<GetTopicRersult>
<Topic>
<User></User>
<Name></Name>
<EndPoint>
<EndpointAddress></EndpointAddress>
<EndpointArgs></EndpointArgs>
<EndpointTopic></EndpointTopic>
<HasStoredSecret></HasStoredSecret>
<Persistent></Persistent>
</EndPoint>
<TopicArn></TopicArn>
<OpaqueData></OpaqueData>
</Topic>
</GetTopicResult>
<ResponseMetadata>
<RequestId></RequestId>
</ResponseMetadata>
</GetTopicResponse>
```

以下是标签和定义：

- ***User ：创建该主题的用户的名称。***
- ***Name ：主题的名称.***

- JSON 格式的端点包括：
  - EndpointAddress ：端点 URL。如果端点 URL 包含用户和密码信息，则必须通过 HTTPS 进行请求。otheriwse，主题 get 请求将被拒绝。
    - EndPointArgs ：端点参数。
    - EndpointTopic ：发送到端点的主题名称可以与上例主题名称不同。
    - HasStoredSecret：当端点 URL 包含用户和密码信息时为 true。
    - Persistent: 当主题是持久的则为 true。
- TopicArn: Topic ARN。
- OpaqueData ：这是在主题上设置的不透明数据。

2.15.5. 列出主题

列出用户已定义的主题。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 根级别访问权限.
- 安装 Ceph 对象网关.
- 用户 access key 和 secret key。

- *端点参数.*

流程

1. *使用以下请求格式列出主题信息：*

语法

```
POST
Action=ListTopics
```

以下是响应格式的示例：

```
<ListTopicdResponse xmlns="https://sns.amazonaws.com/doc/2020-03-31/">
<ListTopicsRersult>
<Topics>
<member>
<User></User>
<Name></Name>
<EndPoint>
<EndpointAddress></EndpointAddress>
<EndpointArgs></EndpointArgs>
<EndpointTopic></EndpointTopic>
</EndPoint>
<TopicArn></TopicArn>
<OpaqueData></OpaqueData>
</member>
</Topics>
</ListTopicsResult>
<ResponseMetadata>
<RequestId></RequestId>
</ResponseMetadata>
</ListTopicsResponse>
```

注意

如果端点 URL 包含用户和密码信息，在任何主题中，则必须通过 HTTPS 进行请求。否则，主题列表请求将被拒绝。

2.15.6. 删除主题

删除删除的主题会导致没有操作，且不是失败。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 根级别访问权限.
- 安装 Ceph 对象网关.
- 用户 access key 和 secret key。
- 端点参数.

流程

1. 使用以下请求格式删除主题：

语法

```
POST
Action=DeleteTopic
&TopicArn=TOPIC_ARN
```

以下是响应格式的示例：

```
<DeleteTopicResponse xmlns="https://sns.amazonaws.com/doc/2020-03-31/">
<ResponseMetadata>
<RequestId></RequestId>
</ResponseMetadata>
</DeleteTopicResponse>
```

-

2.15.7. 使用命令行界面进行主题管理

您可以使用命令行界面列出、获取和删除主题。

先决条件

- Ceph 对象网关节点的根级别访问权限.

流程

1. 获取用户的所有主题的列表：

   语法

   ```
   radosgw-admin topic list --uid=USER_ID
   ```

   示例

   ```
   [root@rgw ~]# radosgw-admin topic list --uid=example
   ```

2. 获取特定主题的配置：

   语法

   ```
   radosgw-admin topic get --uid=USER_ID --topic=TOPIC_NAME
   ```

示例

```
[root@rgw ~]# radosgw-admin topic get --uid=example --topic=example-topic
```

3. *删除特定主题：*

语法

```
radosgw-admin topic rm --uid=USER_ID --topic=TOPIC_NAME
```

示例

```
[root@rgw ~]# radosgw-admin topic rm --uid=example --topic=example-topic
```

2.15.8. 管理通知配置

您可以使用命令行界面列出、获取和删除存储桶的通知配置。

先决条件

- *一个正在运行的 Red Hat Ceph Storage 集群。*
- *配置 Ceph 对象网关。*

流程

- ***列出所有存储桶通知配置：***

语法

```
radosgw-admin notification list --bucket=BUCKET_NAME
```

示例

```
[root@host04 ~]# radosgw-admin notification list --bucket bkt2
{
    "notifications": [
        {
            "TopicArn": "arn:aws:sns:default::topic1",
            "Id": "notif1",
            "Events": [
                "s3:ObjectCreated:*",
                "s3:ObjectRemoved:*"
            ],
            "Filter": {
                "S3Key": {},
                "S3Metadata": {},
                "S3Tags": {}
            }
        },
        {
            "TopicArn": "arn:aws:sns:default::topic1",
            "Id": "notif2",
            "Events": [
                "s3:ObjectSynced:*"
            ],
            "Filter": {
                "S3Key": {},
                "S3Metadata": {},
                "S3Tags": {}
            }
        }
    ]
}
```

-

- ***获取存储桶通知配置：***

语法

```
radosgw-admin notification get --bucket BUCKET_NAME --notification-id NOTIFICATION_ID
```

示例

```
[root@host04 ~]# radosgw-admin notification get --bucket bkt2 --notification-id notif2
{
    "TopicArn": "arn:aws:sns:default::topic1",
    "Id": "notif2",
    "Events": [
        "s3:ObjectSynced:*"
    ],
    "Filter": {
        "S3Key": {},
        "S3Metadata": {},
        "S3Tags": {}
    }
}
```

- ***删除特定的存储桶通知配置：***

语法

```
radosgw-admin notification rm --bucket BUCKET_NAME [--notification-id
NOTIFICATION_ID]
```

在这里，NOTIFICATION_ID 是可选的。如果没有指定，该命令会删除该存储桶的所有通知配置。

示例

```
[root@host04 ~]# radosgw-admin notification rm --bucket bkt2 --notification-id notif1
```

2.15.9. 事件记录

事件包含有关 Ceph 对象网关完成的操作的信息，并通过所选端点（如 HTTP、HTTPS、Kafka 或 AMQ0.9.1）作为有效负载发送。事件记录采用 JSON 格式。

支持以下 ObjectLifecycle:Expiration 事件：

- **ObjectLifecycle:Expiration:Current**
- **ObjectLifecycle:Expiration:NonCurrent**
- **ObjectLifecycle:Expiration:DeleteMarker**
- **ObjectLifecycle:Expiration:AbortMultipartUpload**

示例

```
{"Records":[
    {
        "eventVersion":"2.1",
        "eventSource":"ceph:s3",
        "awsRegion":"us-east-1",
        "eventTime":"2019-11-22T13:47:35.124724Z",
        "eventName":"ObjectCreated:Put",
        "userIdentity":{
            "principalId":"tester"
        },
        "requestParameters":{
```

```
        "sourceIPAddress":""
      },
      "responseElements":{
        "x-amz-request-id":"503a4c37-85eb-47cd-8681-2817e80b4281.5330.903595",
        "x-amz-id-2":"14d2-zone1-zonegroup1"
      },
      "s3":{
        "s3SchemaVersion":"1.0",
        "configurationId":"mynotif1",
        "bucket":{
          "name":"mybucket1",
          "ownerIdentity":{
            "principalId":"tester"
          },
          "arn":"arn:aws:s3:us-east-1::mybucket1",
          "id":"503a4c37-85eb-47cd-8681-2817e80b4281.5332.38"
        },
        "object":{
          "key":"myimage1.jpg",
          "size":"1024",
          "eTag":"37b51d194a7513e45b56f6524f2d51f2",
          "versionId":"",
          "sequencer": "F7E6D75DC742D108",
          "metadata":[],
          "tags":[]
        }
      },
      "eventId":"",
      "opaqueData":"me@example.com"
    }
  ]}
```

以下是事件记录密钥及其定义：

- **awsRegion: Zonegroup。**
- **eventTime: 指示触发事件的时间的时间戳。**
- **eventName ：事件类型。它可以是 ObjectCreated、ObjectRemoved 或 ObjectLifecycle:Expiration**
- **userIdentity.principalId ：触发该事件的用户的身份。**

- *requestParameters.sourceIPAddress ：触发该事件的客户端的 IP 地址。不支持此字段。*
- *responseElements.x-amz-request-id ：触发事件的请求 ID。*
- *responseElements.x_amz_id_2 ：触发事件的 Ceph 对象网关的身份。身份格式为 RGWID-ZONE-ZONEGROUP。*
- *s3.configurationId ：创建事件的通知 ID。*
- *s3.bucket.name ：存储桶的名称。*
- *s3.bucket.ownerIdentity.principalId ：存储桶的所有者。*
- *s3.bucket.arn: bucket 的 Amazon Resource Name(ARN)。*
- *s3.bucket.id ：存储桶的身份。*
- *s3.object.key ：对象键。*
- *s3.object.size ：对象的大小。*
- *s3.object.eTag ：对象 etag。*
- *s3.object.version ：版本控制存储桶中的对象版本。*
- *s3.object.sequencer ：以十六进制格式为每个对象增大更改的标识符。*
- *s3.object.metadata ：对象上设置的任何元数据，作为 x-amz-meta。*

- *s3.object.tags ：对象上设置的任何标签。*
- *s3.eventId ：事件的唯一标识。*
- *s3.opaqueData: Opaque 数据在主题配置中设置，并添加到该主题触发的所有通知中。*

其它资源

- *如需更多信息，请参阅事件消息结构。*

2.15.10. 获取存储桶信息

获取有关现有 bucket 子集的信息。如果在没有 bucket 的情况下指定 uid，则返回属于该用户的所有存储桶。如果只指定 存储桶，则会检索该特定存储桶的信息。

功能

> *`buckets=read`*

语法

```
GET /admin/bucket?format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

请求参数

bucket

描述

用于返回信息的存储桶。

Type

字符串

示例

foo_bucket

必需

否

uid

描述

检索存储桶信息的用户。

Type

字符串

示例

foo_user

必需

否

stats

描述

返回存储桶统计信息。

Type

布尔值

示例

True [False]

必需

否

响应实体

stats

描述

每个存储桶信息。

Type

Container

父

不适用

bucket

描述

包含一个或多个 bucket 容器的列表。

Type

Container

父

bucket

bucket

描述

用于单一存储桶信息的容器。

Type

Container

父

bucket

名称

描述

bucket 的名称。

Type

字符串

父

bucket

pool

描述

bucket 存储在其中的池。

Type

字符串

父

bucket

id

描述

唯一的存储桶 ID。

Type

字符串

父

bucket

marker

描述

内部存储桶标签。

Type

字符串

父

bucket

owner

描述

bucket 所有者的用户 ID。

Type

字符串

父

bucket

usage

描述

存储使用信息。

Type

Container

父

bucket

index

描述

bucket 索引的状态。

Type

字符串

父

bucket

如果成功，请求会返回带有存储桶信息的 bucket 容器。

特殊错误响应

IndexRepairFailed

描述

bucket 索引修复失败。

代码

409 冲突

2.15.11. 检查存储桶索引

检查现有存储桶的索引。

注意

要检查带有 check-objects 的多部分对象核算，fix 必须设为 True。

功能

buckets=write

语法

```
GET /admin/bucket?index&format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

请求参数

bucket

描述

用于返回信息的存储桶。

Type

字符串

示例

foo_bucket

必需

是

check-objects

描述

*检查 **multipart** 对象核算。*

Type

布尔值

示例

True [False]

必需

否

fix

描述

另外，在检查时修复存储桶索引。

Type

布尔值

示例

false [False]

必需

否

响应实体

index

描述

bucket 索引的状态。

Type

字符串

特殊错误响应

IndexRepairFailed

描述

bucket 索引修复失败。

代码

409 冲突

2.15.12. 删除存储桶

删除现有的存储桶。

功能

> *`buckets=write`*

语法

```
DELETE /admin/bucket?format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

请求参数

bucket

描述

要删除的存储桶。

类型

字符串

示例

foo_bucket

必需

是

purge-objects

描述

在删除前删除存储桶的对象。

类型

布尔值

示例

True [False]

必需

否

响应实体

无。

特殊错误响应

BucketNotEmpty

描述

尝试删除非空存储桶。

代码

409 冲突

ObjectRemovalFailed

描述

无法删除对象。

代码

409 冲突

2.15.13. 链接存储桶

将存储桶链接到指定用户，从任何先前用户取消链接存储桶。

功能

> *`buckets=write`*

语法

```
PUT /admin/bucket?format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

-

请求参数

bucket

描述

取消链接的存储桶。

Type

字符串

示例

foo_bucket

必需

是

uid

描述

将存储桶链接到的用户 ID。

Type

字符串

示例

foo_user

必需

是

响应实体

bucket

描述

用于单一存储桶信息的容器。

Type

Container

父

不适用

名称

描述

bucket 的名称。

Type

字符串

父

bucket

pool

描述

bucket 存储在其中的池。

Type

字符串

父

bucket

id

描述

唯一的存储桶 ID。

Type

字符串

父

bucket

marker

描述

内部存储桶标签。

Type

字符串

父

bucket

owner

描述

bucket 所有者的用户 ID。

Type

字符串

父

bucket

usage

描述

存储使用信息。

Type

Container

父

bucket

index

描述

bucket 索引的状态。

Type

字符串

父

bucket

特殊错误响应

BucketUnlinkFailed

描述

无法从指定用户取消链接存储桶。

代码

409 冲突

BucketLinkFailed

描述

无法将存储桶链接到指定用户。

代码

409 冲突

2.15.14. 取消链接存储桶

从指定用户取消链接存储桶。主要用于更改 bucket 所有权。

功能

> `buckets=write`

语法

```
POST /admin/bucket?format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

请求参数

bucket

描述

取消链接的存储桶。

Type

字符串

示例

foo_bucket

必需

是

uid

描述

将存储桶链接到的用户 ID。

Type

字符串

示例

foo_user

必需

是

响应实体

无。

特殊错误响应

BucketUnlinkFailed

描述

无法从指定用户取消链接存储桶。

Type

409 冲突

2.15.15. 获取存储桶或对象策略

对一个对象或 bucket 的读取策略。

功能

```
`buckets=read`
```

语法

```
GET /admin/bucket?policy&format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

请求参数

bucket

描述

要从中读取策略的存储桶。

Type

字符串

示例

foo_bucket

必需

是

object

描述

从中读取策略的对象。

Type

字符串

示例

foo.txt

必需

否

响应实体

policy

描述

访问控制策略。

Type

Container

父

不适用

如果成功，返回对象或存储桶策略

特殊错误响应

IncompleteBody

描述

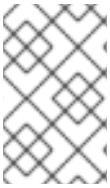
没有为存储桶策略请求指定存储桶，或未为对象策略请求指定对象。

代码

400 错误请求

2.15.16. 删除对象

删除现有对象。



注意

不需要所有者才能终止。

功能

`buckets=write`

语法

```
DELETE /admin/bucket?object&format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

请求参数

bucket

描述

包含要删除的对象的存储桶。

Type

字符串

示例

foo_bucket

必需

是

object

描述

要删除的对象

Type

字符串

示例

foo.txt

必需

是

响应实体

无。

特殊错误响应

NoSuchObject

描述

指定对象不存在。

代码

404 not Found

ObjectRemovalFailed

描述

无法删除对象。

代码

409 冲突

2.15.17. 配额

管理操作 API 允许您在用户和用户拥有的 bucket 上设置配额。配额包括 bucket 中对象的最大数量，以及最大存储大小（以 MB 为单位）。

要查看配额，用户必须具有 users=read 能力。要设置，修改或禁用配额，用户必须具有 users=write 功能。

配额的有效参数包括：

- *Bucket：bucket 选项允许您为用户拥有的 bucket 指定配额。*
- *最大对象： --max-objects 设置允许您指定对象的最大数量。负值将禁用此设置。*

- **Maximum Size: 您可以使用 max-size 选项为最大字节数指定配额。负值将禁用此设置。**
- **Quota Scope: quota-scope 选项设定配额的范围。选项为 bucket 和 user。**

2.15.18. 获取用户配额

要获得配额，用户必须具有带有 read 权限的 users 能力。

语法

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

2.15.19. 设置用户配额

要获得配额，用户必须具有带有 write 权限的 users 能力。

语法

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

内容必须包括配额设置的 JSON 表示，如相应读取操作中编码。

2.15.20. 获取存储桶配额

获取有关现有 bucket 子集的信息。如果在没有 bucket 的情况下指定 uid，则返回属于该用户的所有存储桶。如果只指定 存储桶，则会检索该特定存储桶的信息。

功能

> `buckets=read`

语法

```
GET /admin/bucket?format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

请求参数

bucket

描述

用于返回信息的存储桶。

Type

字符串

示例

foo_bucket

必需

否

uid

描述

检索存储桶信息的用户。

Type

字符串

示例

foo_user

必需

否

stats

描述

返回存储桶统计信息。

Type

布尔值

示例

True [False]

必需

否

响应实体

stats

描述

每个存储桶信息。

Type

Container

父

不适用

bucket

描述

包含一个或多个 bucket 容器的列表。

Type

Container

父

不适用

bucket

描述

用于单一存储桶信息的容器。

Type

Container

父

bucket

名称

描述

bucket 的名称。

Type

字符串

父

bucket

pool

描述

bucket 存储在其中的池。

Type

字符串

父

bucket

id

描述

唯一的存储桶 ID。

Type

字符串

父

bucket

marker

描述

内部存储桶标签。

Type

字符串

父

bucket

owner

描述

bucket 所有者的用户 ID。

Type

字符串

父

bucket

usage

描述

存储使用信息。

Type

Container

父

bucket

index

描述

bucket 索引的状态。

Type

字符串

父

bucket

如果成功，请求会返回带有存储桶信息的 bucket 容器。

特殊错误响应

IndexRepairFailed

描述

bucket 索引修复失败。

代码

409 冲突

2.15.21. 设置存储桶配额

要获得配额，用户必须具有带有 write 权限的 users 能力。

语法

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

内容必须包括配额设置的 JSON 表示，如相应读取操作中编码。

2.15.22. 获取用法信息

请求带宽使用量信息。

功能

```
`usage=read`
```

语法

```
GET /admin/usage?format=json HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
```

请求参数

uid

描述

请求信息的用户。

Type

字符串

必需

是

start

描述

数据请求的开始时间以及可选的日期。例如，2012-09-25 16:00:00。

Type

字符串

必需

否

end

描述

数据请求结束的日期以及可选。例如，2012-09-25 16:00:00。

Type

字符串

必需

否

show-entries

描述

指定是否应返回数据条目。

Type

布尔值

必需

否

show-summary

描述

指定是否应返回数据条目。

Type

布尔值

必需

否

响应实体

usage

描述

用于用法信息的容器。

Type

Container

entries

描述

用于用法条目信息的容器。

Type

Container

user

描述

用于用户数据信息的容器。

Type

Container

owner

描述

拥有存储桶的用户的名称。

Type

字符串

bucket

描述

bucket 名称。

Type

字符串

time

描述

指定数据的时间下限，将其舍入到第一个相关小时的开头。

Type

字符串

epoch

描述

自 1/1/1970 起的时间（以秒为单位）。

Type

字符串

类别

描述

用于统计类别的容器。

Type

Container

entry

描述

用于统计条目的容器。

Type

Container

category

描述

提供统计的请求类别的名称。

Type

字符串

bytes_sent

描述

Ceph 对象网关发送的字节数。

Type

整数

bytes_received

描述

Ceph 对象网关收到的字节数。

Type

整数

ops

描述

操作数量。

Type

整数

successful_ops

描述

成功操作数量。

Type

整数

summary

描述

成功操作数量。

Type

Container

total

描述

容器用于统计摘要聚合的总数。

Type

Container

如果成功，响应包含请求的信息。

2.15.23. 删除使用信息

删除使用信息。如果未指定日期，可删除所有使用信息。

功能

> *`usage=write`*

语法

```
DELETE /admin/usage?format=json HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
```

请求参数

uid

描述

请求信息的用户。

Type

字符串

示例

foo_user

必需

是

start

描述

数据请求的开始时间以及可选的日期。例如， ***2012-09-25 16:00:00。***

Type

字符串

示例

2012-09-25 16:00:00

必需

否

end

描述

数据请求结束的日期以及可选。例如，2012-09-25 16:00:00。

Type

字符串

示例

2012-09-25 16:00:00

必需

否

remove-all

描述

未指定 uid 时需要此项以确认删除多用户数据。

Type

布尔值

示例

True [False]

必需

否

2.15.24. 标准错误响应

以下列表详细介绍了标准错误响应及其描述。

AccessDenied

描述

拒绝访问。

代码

403 Forbidden

InternalError

描述

内部服务器错误.

代码

500 内部服务器错误

NoSuchUser

描述

用户不存在。

代码

404 not Found

NoSuchBucket

描述

bucket 不存在。

代码

404 not Found

NoSuchKey

描述

没有这样的访问密钥。

代码

404 not Found

第 3 章 CEPH 对象网关和 S3 API

作为开发者，您可以使用与 Amazon S3 数据访问模型兼容的 RESTful 应用程序编程接口(API)。您可以通过 Ceph 对象网关管理 Red Hat Ceph Storage 集群中存储的 bucket 和对象。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- RESTful 客户端。

3.1. S3 限制

重要

应谨慎使用以下限制。您的硬件选择会有影响，因此您应该始终与您的红帽客户团队讨论这些要求。

- Maximum object size when using Amazon S3: 单个 Amazon S3 对象的大小范围可以为从 0B 到最多 5TB。单个 PUT 中可以上传的最大对象为 5GB。对于大于 100MB 的对象，您应该考虑使用多部分上传功能。
- 使用 Amazon S3 时的最大元数据大小，对可应用到对象的用户元数据的总大小没有定义限制，但单个 HTTP 请求限制为 16,000 字节。
- Red Hat Ceph Storage 集群的数据开销量为存储 S3 对象和元数据： 这里的估计为 200-300 字节，以及对象名称的长度。版本化的对象会消耗与版本数量成比例的额外空间。另外，在多部分上传和其他事务更新过程中会产生临时开销，但这些开销会在垃圾回收期间恢复。

其它资源

- 有关不支持的标头字段 的详细信息，请参阅 Red Hat Ceph Storage Developer Guide。

3.2. 使用 S3 API 访问 CEPH 对象网关

作为开发者，您必须配置对 Ceph 对象网关和安全令牌服务(STS)的访问权限，然后才能开始使用

Amazon S3 API。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 正在运行的 Ceph 对象网关.
- RESTful 客户端。

3.2.1. S3 身份验证

对 Ceph 对象网关的请求可以是经过身份验证的用户或未经身份验证的请求。Ceph 对象网关假定匿名用户发送了未经身份验证的请求。Ceph 对象网关支持 ACL。

对于大多数用例，客户端使用现有开源库，如 Amazon SDK 的 AmazonS3Client 用于 Java 和 Python Boto。使用开源库，只需传递访问密钥和密钥，库会为您构建请求标头和身份验证签名。但是，您可以创建请求并签署它们。

在向 Ceph 对象网关服务器发送前，在请求中包括访问密钥和基于 64 编码的消息身份验证代码 (HMAC)的要求。Ceph 对象网关使用兼容 S3 的身份验证方法。

示例

```
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
```

在上例中，将 ACCESS_KEY 替换为 access key ID 的值，后跟一个冒号(:)。将 HASH_OF_HEADER_AND_SECRET 替换为一个规范标头字符串的哈希值，以及与访问密钥 ID 对应的

secret。

生成标头字符串和 secret 的哈希值

生成标头字符串和 secret 的哈希值：

1. *获取标头字符串的值。*
2. *将请求标头字符串规范化为规范形式。*
3. *使用 SHA-1 哈希算法生成 HMAC。*
4. *将 hmac 结果编码为 base-64。*

规范化标头

将标头规范化为规范形式：

1. *获取 所有内容 标头。*
2. *删除除 content-type 和 content-md5 外的所有 content- 标头。*
3. *确保 content- 标头名称是小写。*
4. *以字典顺序对 content- 标题排序。*
5. *确保带有 Date 标头，并确保指定的日期使用 GMT 而不是偏移量。*
6. *获取以 x-amz- 开头的所有标题。*
7. *确保 x-amz- 标头都是小写。*

8. 以字典顺序对 x-amz- 标头排序。

9. 将同一字段名称的多个实例合并到一个字段中，并使用逗号分隔字段值。

10. 使用单个空格替换标题值中的空格和换行符。

11. 在冒号前后删除空格。

12. 在每个标头后面添加一个新行。

13. 将标头合并回请求标头。

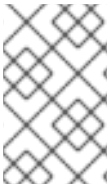
将 HASH_OF_HEADER_AND_SECRET 替换为 base-64 编码的 HMAC 字符串。

其它资源

- 如需了解更多详细信息，请参阅 Amazon Simple Storage Service 文档中的签名和授权 REST Requests 部分。

3.2.2. S3-server-side encryption

Ceph 对象网关支持为 S3 应用编程接口(API)上传对象的服务器端加密。服务器端加密意味着 S3 客户端以未加密的形式通过 HTTP 发送数据，而 Ceph 对象网关以加密的形式将数据存储在 Red Hat Ceph Storage 集群中。



注意

红帽不支持静态大型对象(SLO)或动态大对象(DLO)的 S3 对象加密。

重要

若要使用加密，客户端请求需要通过 SSL 连接发送请求。除非 Ceph 对象网关使用 SSL，否则红帽不支持从客户端进行 S3 加密。但是，出于测试目的，管理员可以使用 ceph config set client.rgw 命令，在测试期间禁用 SSL，在运行时将 rgw_crypt_require_ssl 配置设置设置为 false，然后重新启动 Ceph 对象网关实例。

在生产环境中，可能无法通过 SSL 发送加密的请求。在这种情况下，使用 HTTP 和服务器端加密发送请求。

有关如何使用服务器端加密配置 HTTP 的详情，请参考下面的附加资源部分。

管理加密密钥有两个选项：

客户提供的键

在使用客户提供的密钥时，S3 客户端会传递加密密钥以及每个请求来读取或写入加密数据。客户负责管理这些密钥。客户必须记住用于加密每个对象的 Ceph 对象网关的关键是什么。

Ceph 对象网关根据 Amazon SSE-C 规范在 S3 API 中实施客户提供的关键行为。

由于客户处理密钥管理，并且 S3 客户端将密钥传递到 Ceph 对象网关，因此 Ceph 对象网关不需要特殊配置来支持这种加密模式。

密钥管理服务

在使用密钥管理服务时，安全密钥管理服务存储密钥，Ceph 对象网关则按需检索密钥，为数据加密或解密请求提供服务。

Ceph 对象网关根据 Amazon SSE-KMS 规范在 S3 API 中实施关键管理服务行为。

重要

目前，唯一测试的关键管理实施是 HashiCorp Vault 和 OpenStack Barbican。但是，OpenStack Barbican 是一个技术预览，不支持在生产环境中使用。

其它资源

- *Amazon SSE-C*
- *Amazon SSE-KMS*
- *配置服务器端加密*
- *HashiCorp Vault*

3.2.3. S3 访问控制列表

Ceph 对象网关支持 S3 兼容访问控制列表(ACL)功能。ACL 是访问权限列表，可指定用户可以对存储桶或对象执行的操作。在应用到存储桶而不是应用到对象时，每个授权都有不同的意义：

表 3.1. 用户操作

权限	Bucket	Object
READ	Grantee 可以列出存储桶中的对象。	Grantee 可以读取对象。
写	Grantee 可以编写或删除存储桶中的对象。	不适用
READ_ACP	Grantee 可以读取存储桶 ACL。	Grantee 可以读取对象 ACL。
WRITE_ACP	Grantee 可以编写存储桶 ACL。	Grantee 可以写入对象 ACL。
FULL_CONTROL	Grantee 具有存储桶中对象的完整权限。	Grantee 可以读取和写入对象 ACL。

3.2.4. 使用 S3 准备对 Ceph 对象网关的访问

在尝试访问网关服务器前，您必须在 Ceph 对象网关节点上遵循一些前提条件。

先决条件

- *安装 Ceph 对象网关软件.*

- **Ceph 对象网关节点的根级别访问权限.**

流程

1. **以 root 用户身份，在防火墙上打开端口 8080 ：**

```
[root@rgw ~]# firewall-cmd --zone=public --add-port=8080/tcp --permanent
[root@rgw ~]# firewall-cmd --reload
```

2. **如 对象网关配置和管理指南 中所述，为网关使用的 DNS 服务器添加通配符。**

   您还可以为本地 DNS 缓存设置网关节点。要做到这一点，请执行以下步骤：

   a. **以 root 用户身份，安装和设置 dnsmasq ：**

```
[root@rgw ~]# yum install dnsmasq
[root@rgw ~]# echo
"address=/.FQDN_OF_GATEWAY_NODE/IP_OF_GATEWAY_NODE" | tee --append
/etc/dnsmasq.conf
[root@rgw ~]# systemctl start dnsmasq
[root@rgw ~]# systemctl enable dnsmasq
```

   将 IP_OF_GATEWAY_NODE 和 FQDN_OF_GATEWAY_NODE 替换为网关节点的 IP 地址和 FQDN。

   b. **作为 root 用户，停止 NetworkManager：**

```
[root@rgw ~]# systemctl stop NetworkManager
[root@rgw ~]# systemctl disable NetworkManager
```

   c. **以 root 用户身份，将网关服务器的 IP 设置为名称服务器：**

```
[root@rgw ~]# echo "DNS1=IP_OF_GATEWAY_NODE" | tee --append
/etc/sysconfig/network-scripts/ifcfg-eth0
[root@rgw ~]# echo "IP_OF_GATEWAY_NODE FQDN_OF_GATEWAY_NODE" | tee --
append /etc/hosts
[root@rgw ~]# systemctl restart network
[root@rgw ~]# systemctl enable network
[root@rgw ~]# systemctl restart dnsmasq
```

将 IP_OF_GATEWAY_NODE 和 FQDN_OF_GATEWAY_NODE 替换为网关节点的 IP 地址和 FQDN。

d. 验证子域请求：

```
[user@rgw ~]$ ping mybucket.FQDN_OF_GATEWAY_NODE
```

将 FQDN_OF_GATEWAY_NODE 替换为网关节点的 FQDN。

警告

为本地 DNS 缓存设置网关服务器仅用于测试目的。执行此操作后，您将无法访问外部网络。强烈建议您为 Red Hat Ceph Storage 集群和网关节点使用正确的 DNS 服务器。

3. 如 对象网关配置和管理指南中所述，为 S3 访问创建 radosgw 用户， 并复制生成的 access_key 和 secret_key。您需要这些密钥进行 S3 访问和随后的 bucket 管理任务。

3.2.5. 使用 Ruby AWS S3 访问 Ceph 对象网关

您可以使用 Ruby 编程语言以及 aws-s3 gem 用于 S3 访问。在用于通过 Ruby AWS::S3 访问 Ceph 对象网关服务器的节点上执行下方所述的步骤。

先决条件

- Ceph 对象网关用户级访问权限.
- 访问 Ceph 对象网关的 root 级别访问节点。
- 互联网访问。

流程

1. **安装 ruby 软件包：**

```
[root@dev ~]# yum install ruby
```

注意

以上命令将安装 ruby 及其基本依赖项，如 rubygems 和 ruby-libs。如果某种方式命令没有安装所有依赖项，请单独安装它们。

2. **安装 aws-s3 Ruby 软件包：**

```
[root@dev ~]# gem install aws-s3
```

3. **创建项目目录：**

```
[user@dev ~]$ mkdir ruby_aws_s3
[user@dev ~]$ cd ruby_aws_s3
```

4. **创建连接文件：**

```
[user@dev ~]$ vim conn.rb
```

5. **将以下内容粘贴到 conn.rb 文件中：**

语法

```
#!/usr/bin/env ruby

require 'aws/s3'
require 'resolv-replace'

AWS::S3::Base.establish_connection!(
        :server            => 'FQDN_OF_GATEWAY_NODE',
        :port          => '8080',
        :access_key_id     => 'MY_ACCESS_KEY',
        :secret_access_key => 'MY_SECRET_KEY'
)
```

将 FQDN_OF_GATEWAY_NODE 替换为 Ceph 对象网关节点的 FQDN。将 MY_ACCESS_KEY 和 MY_SECRET_KEY 替换为您在为 S3 访问创建 radosgw 用户时生成的 access_key 和 secret_key，如 Red Hat Ceph Storage Object Gateway Configuration and Administration Guide 所述。

示例

```
#!/usr/bin/env ruby

require 'aws/s3'
require 'resolv-replace'

AWS::S3::Base.establish_connection!(
        :server            => 'testclient.englab.pnq.redhat.com',
        :port          => '8080',
        :access_key_id     => '98J4R9P22P5CDL65HKP8',
        :secret_access_key => '6C+jcaP0dp0+FZfrRNgyGA9EzRy25pURldwje049'
)
```

保存文件并退出编辑器。

6. 使文件可执行：

```
[user@dev ~]$ chmod +x conn.rb
```

7. 运行该文件：

```
[user@dev ~]$ ./conn.rb | echo $?
```

如果您在文件中正确提供了值，命令的输出将是 0。

8. 创建存储桶的新文件：

```
[user@dev ~]$ vim create_bucket.rb
```

将以下内容粘贴到文件中：

```
#!/usr/bin/env ruby

load 'conn.rb'

AWS::S3::Bucket.create('my-new-bucket1')
```

保存文件并退出编辑器。

9. 使文件可执行：

```
[user@dev ~]$ chmod +x create_bucket.rb
```

10. 运行该文件：

```
[user@dev ~]$ ./create_bucket.rb
```

如果命令的输出为 **true**，这表示 **bucket my-new-bucket1** 已创建成功。

11. 创建用于列出所拥有的存储桶的新文件：

```
[user@dev ~]$ vim list_owned_buckets.rb
```

将以下内容粘贴到文件中：

```
#!/usr/bin/env ruby

load 'conn.rb'

AWS::S3::Service.buckets.each do |bucket|
        puts "{bucket.name}\t{bucket.creation_date}"
end
```

保存文件并退出编辑器。

12. 使文件可执行：

```
[user@dev ~]$ chmod +x list_owned_buckets.rb
```

13. 运行该文件：

```
[user@dev ~]$ ./list_owned_buckets.rb
```

输出应类似如下：

```
my-new-bucket1 2020-01-21 10:33:19 UTC
```

14. 创建用于创建对象的新文件：

```
[user@dev ~]$ vim create_object.rb
```

将以下内容粘贴到文件中：

```
#!/usr/bin/env ruby

load 'conn.rb'

AWS::S3::S3Object.store(
        'hello.txt',
        'Hello World!',
        'my-new-bucket1',
        :content_type => 'text/plain'
)
```

保存文件并退出编辑器。

15. 使文件可执行：

```
[user@dev ~]$ chmod +x create_object.rb
```

16. 运行该文件：

```
[user@dev ~]$ ./create_object.rb
```

-

这将创建一个含有字符串 **Hello World!** 的文件 **hello.txt**。

17. 创建用于列出存储桶内容的新文件：

```
[user@dev ~]$ vim list_bucket_content.rb
```

将以下内容粘贴到文件中：

```
#!/usr/bin/env ruby

load 'conn.rb'

new_bucket = AWS::S3::Bucket.find('my-new-bucket1')
new_bucket.each do |object|
        puts "{object.key}\t{object.about['content-length']}\t{object.about['last-modified']}"
end
```

保存文件并退出编辑器。

18. 使文件成为可执行文件。

```
[user@dev ~]$ chmod +x list_bucket_content.rb
```

19. 运行该文件：

```
[user@dev ~]$ ./list_bucket_content.rb
```

输出将类似如下：

```
hello.txt     12     Fri, 22 Jan 2020 15:54:52 GMT
```

20. 创建用于删除空存储桶的新文件：

```
[user@dev ~]$ vim del_empty_bucket.rb
```

将以下内容粘贴到文件中：

```
#!/usr/bin/env ruby

load 'conn.rb'

AWS::S3::Bucket.delete('my-new-bucket1')
```

保存文件并退出编辑器。

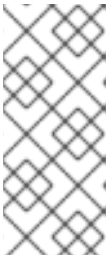
21. 使文件可执行：

```
[user@dev ~]$ chmod +x del_empty_bucket.rb
```

22. 运行该文件：

```
[user@dev ~]$ ./del_empty_bucket.rb | echo $?
```

如果删除了存储桶，命令会返回 0 作为输出。



注意

编辑 **create_bucket.rb** 文件，以创建空存储桶，如 **my-new-bucket4**、**my-new-bucket5**。然后，在尝试删除空存储桶前相应地编辑上述 **del_empty_bucket.rb** 文件。

23. 创建用于删除非空存储桶的新文件：

```
[user@dev ~]$ vim del_non_empty_bucket.rb
```

将以下内容粘贴到文件中：

```
#!/usr/bin/env ruby

load 'conn.rb'

AWS::S3::Bucket.delete('my-new-bucket1', :force => true)
```

保存文件并退出编辑器。

24. 使文件可执行：

```
[user@dev ~]$ chmod +x del_non_empty_bucket.rb
```

25. 运行该文件：

```
[user@dev ~]$ ./del_non_empty_bucket.rb | echo $?
```

如果删除了存储桶，命令会返回 0 作为输出。

26. 创建用于删除对象的新文件：

```
[user@dev ~]$ vim delete_object.rb
```

将以下内容粘贴到文件中：

```
#!/usr/bin/env ruby

load 'conn.rb'

AWS::S3::S3Object.delete('hello.txt', 'my-new-bucket1')
```

保存文件并退出编辑器。

27. 使文件可执行：

```
[user@dev ~]$ chmod +x delete_object.rb
```

28. 运行该文件：

```
[user@dev ~]$ ./delete_object.rb
```

这将删除对象 **hello.txt**。

3.2.6. 使用 Ruby AWS SDK 访问 Ceph 对象网关

您可以使用 **Ruby** 编程语言以及 **aws-sdk gem** 用于 **S3** 访问。在用于通过 **Ruby AWS::SDK** 访问 **Ceph** 对象网关服务器的节点上执行下方所述的步骤。

先决条件

- **Ceph** 对象网关用户级访问权限.
- 访问 **Ceph** 对象网关的 **root** 级别访问节点。
- 互联网访问。

流程

1. 安装 **ruby** 软件包：

```
[root@dev ~]# yum install ruby
```

注意

以上命令将安装 **ruby** 及其基本依赖项，如 **rubygems** 和 **ruby-libs**。如果某种方式命令没有安装所有依赖项，请单独安装它们。

2. 安装 **aws-sdk Ruby** 软件包：

```
[root@dev ~]# gem install aws-sdk
```

3. 创建项目目录：

```
[user@dev ~]$ mkdir ruby_aws_sdk
[user@dev ~]$ cd ruby_aws_sdk
```

4. *创建连接文件：*

```
[user@dev ~]$ vim conn.rb
```

5. *将以下内容粘贴到 conn.rb 文件中：*

语法

```
#!/usr/bin/env ruby

require 'aws-sdk'
require 'resolv-replace'

Aws.config.update(
        endpoint: 'http://FQDN_OF_GATEWAY_NODE:8080',
        access_key_id: 'MY_ACCESS_KEY',
        secret_access_key: 'MY_SECRET_KEY',
        force_path_style: true,
        region: 'us-east-1'
)
```

将 FQDN_OF_GATEWAY_NODE 替换为 Ceph 对象网关节点的 FQDN。将 MY_ACCESS_KEY 和 MY_SECRET_KEY 替换为您在为 S3 访问创建 radosgw 用户时生成的 access_key 和 secret_key，如 Red Hat Ceph Storage Object Gateway Configuration and Administration Guide 所述。

示例

```
#!/usr/bin/env ruby

require 'aws-sdk'
require 'resolv-replace'

Aws.config.update(
        endpoint: 'http://testclient.englab.pnq.redhat.com:8080',
        access_key_id: '98J4R9P22P5CDL65HKP8',
        secret_access_key: '6C+jcaP0dp0+FZfrRNgyGA9EzRy25pURldwje049',
        force_path_style: true,
        region: 'us-east-1'
)
```

保存文件并退出编辑器。

6. 使文件可执行：

```
[user@dev ~]$ chmod +x conn.rb
```

7. 运行该文件：

```
[user@dev ~]$ ./conn.rb | echo $?
```

如果您在文件中正确提供了值，命令的输出将是 0。

8. 创建存储桶的新文件：

```
[user@dev ~]$ vim create_bucket.rb
```

将以下内容粘贴到文件中：

语法

```
#!/usr/bin/env ruby

load 'conn.rb'

s3_client = Aws::S3::Client.new
s3_client.create_bucket(bucket: 'my-new-bucket2')
```

保存文件并退出编辑器。

9. 使文件可执行：

```
[user@dev ~]$ chmod +x create_bucket.rb
```

10. 运行该文件：

```
[user@dev ~]$ ./create_bucket.rb
```

如果命令的输出为 **true**，这表示 **bucket my-new-bucket2** 已创建成功。

11. 创建用于列出所拥有的存储桶的新文件：

```
[user@dev ~]$ vim list_owned_buckets.rb
```

将以下内容粘贴到文件中：

```
#!/usr/bin/env ruby

load 'conn.rb'

s3_client = Aws::S3::Client.new
s3_client.list_buckets.buckets.each do |bucket|
      puts "{bucket.name}\t{bucket.creation_date}"
end
```

保存文件并退出编辑器。

12. 使文件可执行：

```
[user@dev ~]$ chmod +x list_owned_buckets.rb
```

13. 运行该文件：

```
[user@dev ~]$ ./list_owned_buckets.rb
```

输出应类似如下：

```
-
```

```
my-new-bucket2 2020-01-21 10:33:19 UTC
```

14. 创建用于创建对象的新文件：

```
[user@dev ~]$ vim create_object.rb
```

将以下内容粘贴到文件中：

```
#!/usr/bin/env ruby

load 'conn.rb'

s3_client = Aws::S3::Client.new
s3_client.put_object(
        key: 'hello.txt',
        body: 'Hello World!',
        bucket: 'my-new-bucket2',
        content_type: 'text/plain'
)
```

保存文件并退出编辑器。

15. 使文件可执行：

```
[user@dev ~]$ chmod +x create_object.rb
```

16. 运行该文件：

```
[user@dev ~]$ ./create_object.rb
```

这将创建一个含有字符串 **Hello World!** 的文件 **hello.txt**。

17. 创建用于列出存储桶内容的新文件：

```
[user@dev ~]$ vim list_bucket_content.rb
```

将以下内容粘贴到文件中：

```
#!/usr/bin/env ruby

load 'conn.rb'

s3_client = Aws::S3::Client.new
s3_client.list_objects(bucket: 'my-new-bucket2').contents.each do |object|
        puts "{object.key}\t{object.size}"
end
```

保存文件并退出编辑器。

18. 使文件成为可执行文件。

```
[user@dev ~]$ chmod +x list_bucket_content.rb
```

19. 运行该文件：

```
[user@dev ~]$ ./list_bucket_content.rb
```

输出将类似如下：

```
hello.txt     12     Fri, 22 Jan 2020 15:54:52 GMT
```

20. 创建用于删除空存储桶的新文件：

```
[user@dev ~]$ vim del_empty_bucket.rb
```

将以下内容粘贴到文件中：

```
#!/usr/bin/env ruby

load 'conn.rb'

s3_client = Aws::S3::Client.new
s3_client.delete_bucket(bucket: 'my-new-bucket2')
```

保存文件并退出编辑器。

21. 使文件可执行：

```
[user@dev ~]$ chmod +x del_empty_bucket.rb
```

22. 运行该文件：

```
[user@dev ~]$ ./del_empty_bucket.rb | echo $?
```

如果删除了存储桶，命令会返回 0 作为输出。

注意

编辑 **create_bucket.rb** 文件，以创建空存储桶，如 **my-new-bucket6**、**my-new-bucket7**。然后，在尝试删除空存储桶前相应地编辑上述 **del_empty_bucket.rb** 文件。

23. 创建用于删除非空存储桶的新文件：

```
[user@dev ~]$ vim del_non_empty_bucket.rb
```

将以下内容粘贴到文件中：

```
#!/usr/bin/env ruby

load 'conn.rb'

s3_client = Aws::S3::Client.new
Aws::S3::Bucket.new('my-new-bucket2', client: s3_client).clear!
s3_client.delete_bucket(bucket: 'my-new-bucket2')
```

保存文件并退出编辑器。

24. 使文件可执行：

```
[user@dev ~]$ chmod +x del_non_empty_bucket.rb
```

25.

25. 运行该文件：

```
[user@dev ~]$ ./del_non_empty_bucket.rb | echo $?
```

如果删除了存储桶，命令会返回 0 作为输出。

26. 创建用于删除对象的新文件：

```
[user@dev ~]$ vim delete_object.rb
```

将以下内容粘贴到文件中：

```
#!/usr/bin/env ruby

load 'conn.rb'

s3_client = Aws::S3::Client.new
s3_client.delete_object(key: 'hello.txt', bucket: 'my-new-bucket2')
```

保存文件并退出编辑器。

27. 使文件可执行：

```
[user@dev ~]$ chmod +x delete_object.rb
```

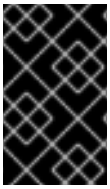
28. 运行该文件：

```
[user@dev ~]$ ./delete_object.rb
```

这将删除对象 **hello.txt**。

3.2.7. 使用 PHP 访问 Ceph 对象网关

您可以使用 PHP 脚本进行 S3 访问。此流程提供一些示例 PHP 脚本来执行各种任务，如删除存储桶或对象。



重要

以下给出的示例针对 php v5.4.16 和 aws-sdk v2.8.24 测试。

先决条件

- 根级访问开发工作站。
- 互联网访问。

流程

1. 安装 php 软件包：

```
[root@dev ~]# yum install php
```

2. 下载适用于 PHP 的 aws-sdk 的 zip 存档并解压缩。

3. 创建项目目录：

```
[user@dev ~]$ mkdir php_s3
[user@dev ~]$ cd php_s3
```

4. 将提取的 aws 目录复制到项目目录中。例如：

```
[user@dev ~]$ cp -r ~/Downloads/aws/ ~/php_s3/
```

5. 创建连接文件：

```
[user@dev ~]$ vim conn.php
```

6. 将以下内容粘贴到 conn.php 文件中：

语法

```
<?php
define('AWS_KEY', 'MY_ACCESS_KEY');
define('AWS_SECRET_KEY', 'MY_SECRET_KEY');
define('HOST', 'FQDN_OF_GATEWAY_NODE');
define('PORT', '8080');

// require the AWS SDK for php library
require '/PATH_TO_AWS/aws-autoloader.php';

use Aws\S3\S3Client;

// Establish connection with host using S3 Client
client = S3Client::factory(array(
    'base_url' => HOST,
    'port' => PORT,
    'key'      => AWS_KEY,
    'secret'   => AWS_SECRET_KEY
));
?>
```

将 FQDN_OF_GATEWAY_NODE 替换为网关节点的 FQDN。将 MY_ACCESS_KEY 和 MY_SECRET_KEY 替换为创建 radosgw 用户以进行 S3 访问时生成的 access_key 和 secret_key，如 Red Hat Ceph Storage Object Gateway Configuration and Administration Guide 所述。将 PATH_TO_AWS 替换为您复制到 php 项目目录的提取 aws 目录的绝对路径。

保存文件并退出编辑器。

7. **运行该文件：**

```
[user@dev ~]$ php -f conn.php | echo $?
```

如果您在文件中正确提供了值，命令的输出将是 0。

8. **创建存储桶的新文件：**

```
[user@dev ~]$ vim create_bucket.php
```

将以下内容粘贴到新文件中：

语法

```
<?php

include 'conn.php';

client->createBucket(array('Bucket' => 'my-new-bucket3'));

?>
```

保存文件并退出编辑器。

9. *运行该文件：*

```
[user@dev ~]$ php -f create_bucket.php
```

10. *创建用于列出所拥有的存储桶的新文件：*

```
[user@dev ~]$ vim list_owned_buckets.php
```

将以下内容粘贴到文件中：

语法

```
<?php

include 'conn.php';

blist = client->listBuckets();
echo "Buckets belonging to " . blist['Owner']['ID'] . ":\n";
foreach (blist['Buckets'] as b) {
    echo "{b['Name']}\t{b['CreationDate']}\n";
}

?>
```

保存文件并退出编辑器。

11. 运行该文件：

```
[user@dev ~]$ php -f list_owned_buckets.php
```

输出应类似于如下：

```
my-new-bucket3 2020-01-21 10:33:19 UTC
```

12. 首先创建名为 **hello.txt** 的源文件，以创建对象：

```
[user@dev ~]$ echo "Hello World!" > hello.txt
```

13. 创建新 **php** 文件：

```
[user@dev ~]$ vim create_object.php
```

将以下内容粘贴到文件中：

语法

```
<?php

include 'conn.php';

key         = 'hello.txt';
source_file = './hello.txt';
acl         = 'private';
bucket      = 'my-new-bucket3';
client->upload(bucket, key, fopen(source_file, 'r'), acl);

?>
```

保存文件并退出编辑器。

14. 运行该文件：

```
[user@dev ~]$ php -f create_object.php
```

这将在 bucket my-new-bucket3 中创建对象 hello.txt。

15. 创建用于列出存储桶内容的新文件：

```
[user@dev ~]$ vim list_bucket_content.php
```

将以下内容粘贴到文件中：

语法

```
<?php

include 'conn.php';

o_iter = client->getIterator('ListObjects', array(
    'Bucket' => 'my-new-bucket3'
));
foreach (o_iter as o) {
    echo "{o['Key']}\t{o['Size']}\t{o['LastModified']}\n";
}
?>
```

保存文件并退出编辑器。

16. 运行该文件：

```
[user@dev ~]$ php -f list_bucket_content.php
```

输出类似如下：

```
hello.txt    12    Fri, 22 Jan 2020 15:54:52 GMT
```

17. 创建用于删除空存储桶的新文件：

```
[user@dev ~]$ vim del_empty_bucket.php
```

将以下内容粘贴到文件中：

语法

```
<?php

include 'conn.php';

client->deleteBucket(array('Bucket' => 'my-new-bucket3'));
?>
```

保存文件并退出编辑器。

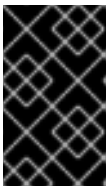
18. 运行该文件：

```
[user@dev ~]$ php -f del_empty_bucket.php | echo $?
```

如果删除了存储桶，命令会返回 0 作为输出。

注意

编辑 **create_bucket.php** 文件，以创建空存储桶，如 **my-new-bucket4**、**my-new-bucket5**。然后，在尝试删除空存储桶前相应地编辑上述 **del_empty_bucket.php** 文件。



重要

PHP 2 和 aws-sdk 的新版本不支持删除非空存储桶。

19. 创建用于删除对象的新文件：

```
[user@dev ~]$ vim delete_object.php
```

将以下内容粘贴到文件中：

语法

```
<?php

include 'conn.php';

client->deleteObject(array(
    'Bucket' => 'my-new-bucket3',
    'Key'    => 'hello.txt',
));
?>
```

保存文件并退出编辑器。

20. 运行该文件：

```
[user@dev ~]$ php -f delete_object.php
```

这将删除对象 hello.txt。

3.2.8. 安全令牌服务

Amazon Web Services 的安全令牌服务(STS)返回一组临时安全凭证来验证用户。

Red Hat Ceph Storage Object Gateway 支持用于身份和访问管理(IAM)的 Amazon STS 应用程序编程接口(API)的子集。

用户首先对 STS 进行身份验证并接收一个简短的 S3 访问密钥和 secret 密钥，可在后续请求中使用。

Red Hat Ceph Storage 可以通过配置 OIDC 供应商与单点登录集成来验证 S3 用户。此功能使对象存储用户能够对企业身份提供程序而不是本地 Ceph 对象网关数据库进行身份验证。例如，如果 SSO 连接到后端中的企业 IDP，则对象存储用户可以使用其企业凭据进行身份验证并可访问 Ceph 对象网关 S3 端点。

通过使用 STS 和 IAM 角色策略功能，您可以创建精心调优的授权策略来控制对数据的访问。这可让您为对象存储数据实施基于角色的访问控制(RBAC)或基于属性的访问控制(ABAC)授权模型，以便您完全控制谁可以访问数据。

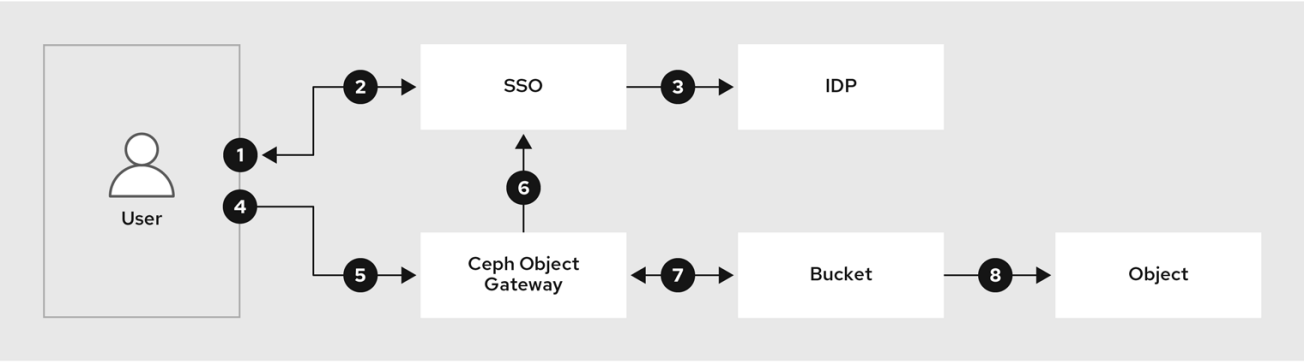
使用 STS 简化访问 S3 资源的工作流

1. 用户希望访问 Red Hat Ceph Storage 中的 S3 资源。
2. 用户需要针对 SSO 提供程序进行身份验证。
3. SSO 提供程序与 IDP 联合，检查用户凭据是否有效，用户通过身份验证，并且 SSO 为用户提供令牌。
4. 使用 SSO 提供的令牌，用户访问 Ceph 对象网关 STS 端点，要求假定一个 IAM 角色提供对 S3 资源的访问权限。
5. Red Hat Ceph Storage 网关接收用户令牌，并要求 SSO 验证令牌。

6. 在 SSO 验证令牌后，允许用户假定角色。通过 STS，该用户具有临时访问和 secret 密钥，用于授予用户对 S3 资源的访问权限。

7. 根据附加到用户假定的 IAM 角色的策略，用户可以访问一组 S3 资源。

8. 例如，存储桶 A 的读取并写入存储桶 B。



550_Ceph_0224

其它资源

- Amazon Web Services Secure Token Service 欢迎页面。
- 如需了解有关 STS Lite 和 Keystone 的详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 配置和使用 STS Lite with Keystone 部分。
- 如需了解有关 STS Lite 和 Keystone 的限制，请参阅 Red Hat Ceph Storage Developer Guide 中的使用 STS Lite 和 Keystone 的限制。

3.2.8.1. 安全令牌服务应用程序编程接口

Ceph 对象网关实施下列安全令牌服务(STS)应用程序编程接口(API)：

AssumeRole

此 API 返回一组临时凭证，用于跨帐户访问。这些临时凭证都允许，附加了 AssumeRole API 的 Role 和策略的权限策略。RoleArn 和 RoleSessionName 请求参数是必需的，但其他请求参数是可选的。

RoleArn

描述

假定为 Amazon 资源名(ARN)的角色，长度为 20 到 2048 个字符。

Type

字符串

必需

是

RoleSessionName

描述

识别要假定的角色会话名称。当不同的主体或不同的原因假设角色时，角色会话名称可以唯一标识一个会话。此参数的值的长度为 2 到 64 个字符。允许 =、、、、@ 和 - 字符，但不允许使用空格。

Type

字符串

必需

是

policy

描述

以 JSON 格式的身份和访问管理策略(IAM)，用于在内联会话中使用。此参数的值的长度为 1 到 2048 个字符。

Type

字符串

必需

否

DurationSeconds

描述

会话持续时间（以秒为单位），最小值为 900 秒，最大值为 43200 秒。默认值为 3600 秒。

Type

整数

必需

否

ExternalId

描述

假设另一个帐户的角色时，请提供一个唯一的外部标识符（如果可用）。这个参数的值的长度为 2 到 1224 个字符。

Type

字符串

必需

否

SerialNumber

描述

用户从关联的多因素身份验证(MFA)设备识别号。参数的值可以是硬件设备或虚拟设备的序列号，长度为 9 到 256 个字符。

Type

字符串

必需

否

TokenCode

描述

如果信任策略需要 MFA，则从多因素身份验证(MFA)设备生成的值。如果需要 MFA 设备，如果此参数的值为空或过期，则 AssumeRole 调用会返回 "access denied" 错误消息。这个参数的值的固定长度为 6 个字符。

Type

字符串

必需

否

AssumeRoleWithWebIdentity

此 API 为应用进行身份验证的用户返回一组临时凭据，如 OpenID Connect 或 OAuth 2.0 身份提供程序。RoleArn 和 RoleSessionName 请求参数是必需的，但其他请求参数是可选的。

RoleArn

描述

假定为 Amazon 资源名(ARN)的角色，长度为 20 到 2048 个字符。

Type

字符串

必需

是

RoleSessionName

描述

识别要假定的角色会话名称。当不同的主体或不同的原因假设角色时，角色会话名称可以唯一标识一个会话。此参数的值的长度为 2 到 64 个字符。允许 =、、、、、@ 和 - 字符，但不允许使用空格。

Type

字符串

必需

是

policy

描述

以 JSON 格式的身份和访问管理策略(IAM)，用于在内联会话中使用。此参数的值的长度为 1 到 2048 个字符。

Type

字符串

必需

否

DurationSeconds

描述

会话持续时间（以秒为单位），最小值为 900 秒，最大值为 43200 秒。默认值为 3600 秒。

Type

整数

必需

否

ProviderId

描述

身份提供程序中域名的完全限定主机组件。此参数的值仅对 OAuth 2.0 访问令牌有效，长度为 4 到 2048 个字符。

Type

字符串

必需

否

WebIdentityToken

描述

从身份提供程序提供的 OpenID Connect 身份令牌或 OAuth 2.0 访问令牌。此参数的值的长度为 4 到 2048 个字符。

Type

字符串

必需

否

其它资源

- *如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的使用 安全令牌服务 API 部分的示例。*
- *Amazon Web Services Security Token Service， AssumeRole 操作。*
- *Amazon Web Services Security Token Service， AssumeRoleWithWebIdentity 操作。*

3.2.8.2. 配置安全令牌服务

通过设置 rgw_sts_key 和 rgw_s3_auth_use_sts 选项，配置与 Ceph 对象网关搭配使用的安全令牌服务(STS)。

注意

S3 和 STS API 共同存在于同一命名空间中，两者都可从 Ceph 对象网关中的同一端点访问。

先决条件

- *一个正在运行的 Red Hat Ceph Storage 集群。*
- *正在运行的 Ceph 对象网关.*
- *Ceph 管理器节点的 root 级别访问。*

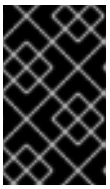
流程

1. 为 Ceph 对象网关客户端设置以下配置选项：

语法

```
ceph config set RGW_CLIENT_NAME rgw_sts_key STS_KEY
ceph config set RGW_CLIENT_NAME rgw_s3_auth_use_sts true
```

rgw_sts_key 是用于加密或解密会话令牌的 STS 密钥，且正好为 16 十六进制字符。



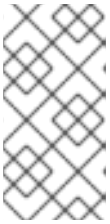
重要

STS 密钥需要是字母数字。

示例

```
[root@mgr ~]# ceph config set client.rgw rgw_sts_key 7f8fd8dd4700mnop
[root@mgr ~]# ceph config set client.rgw rgw_s3_auth_use_sts true
```

2. 重新启动 Ceph 对象网关，使添加的密钥生效。



注意

使用 ceph orch ps 命令的输出，在 NAME 列下获取 SERVICE_TYPE。ID 信息。

a. 在存储集群中的单个节点上重启 Ceph 对象网关：

语法

```
systemctl restart ceph-CLUSTER_ID@SERVICE_TYPE.ID.service
```

示例

```
[root@host01 ~]# systemctl restart ceph-c4b34c6f-8365-11ba-dc31-
529020a7702d@rgw.realm.zone.host01.gwasto.service
```

b. **在存储集群的所有节点上重启 Ceph 对象网关：**

语法

```
ceph orch restart SERVICE_TYPE
```

示例

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

其它资源

- **有关 STS API 的详情，请参阅 Red Hat Ceph Storage Developer Guide 中的 安全令牌服务应用程序接口 部分。**
-

有关使用 Ceph 配置数据库的更多详细信息，请参见 Red Hat Ceph Storage 配置指南中的 Ceph 配置的基础知识。

3.2.8.3. 为 OpenID Connect 供应商创建用户

要在 Ceph 对象网关和 OpenID Connect 供应商间建立信任，请创建一个用户实体和角色信任策略。

先决条件

- Ceph 对象网关节点的用户级访问权限。
- 配置了安全令牌服务。

流程

1. 创建新 Ceph 用户：

语法

```
radosgw-admin --uid USER_NAME --display-name "DISPLAY_NAME" --access_key
USER_NAME --secret SECRET user create
```

示例

```
[user@rgw ~]$ radosgw-admin --uid TESTER --display-name "TestUser" --access_key
TESTER --secret test123 user create
```

2. 配置 Ceph 用户功能：

语法

```
radosgw-admin caps add --uid="USER_NAME" --caps="oidc-provider=*"
```

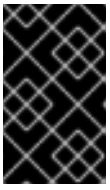
示例

```
[user@rgw ~]$ radosgw-admin caps add --uid="TESTER" --caps="oidc-provider=*"
```

3. **使用 Secure Token Service(STS)API 在角色信任策略中添加条件：**

语法

```
"{\"Version\":\"2020-01-17\",\"Statement\":[{\"Effect\":\"Allow\",\"Principal\":{\"Federated\":
[\"arn:aws:iam:::oidc-provider/IDP_URL\"]},\"Action\":
[\"sts:AssumeRoleWithWebIdentity\"],\"Condition\":{\"StringEquals\":
{\"IDP_URL:app_id\":\"AUD_FIELD\"}\}\}\]\}"
```



重要

上述语法示例中的 app_id 必须与传入令牌的 AUD_FIELD 字段匹配。

其它资源

- 请参阅 **Amazon** 网站上的 **OpenID Connect** 身份提供程序的 **Obtaining the Root CA Thumbprint**。
- 有关 **STS API** 的详情，请参阅 **Red Hat Ceph Storage** 开发者指南中的 **安全令牌服务应用程序编程接口** 部分。

- 如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的使用 安全令牌服务 API 部分的示例。

3.2.8.4. 获取 OpenID Connect 供应商的 thumbprint

获取 OpenID Connect 供应商(IDP)配置文档。

任何遵循 OIDC 协议标准的 SSO 都应该使用 Ceph 对象网关。红帽已使用以下 SSO 供应商测试：

- Red Hat Single Sing-on
- Keycloak

先决条件

- 安装 openssl 和 curl 软件包。

流程

1. 从 IDP 的 URL 获取配置文档：

语法

```
curl -k -v \
    -X GET \
    -H "Content-Type: application/x-www-form-urlencoded" \
    "IDP_URL:8000/CONTEXT/realms/REALM/.well-known/openid-configuration" \
  | jq .
```

示例

```
[user@client ~]$ curl -k -v \
```

```
    -X GET \
    -H "Content-Type: application/x-www-form-urlencoded" \
    "http://www.example.com:8000/auth/realms/quickstart/.well-known/openid-configuration" \
  | jq .
```

2. ***获取 IDP 证书：***

语法

```
curl -k -v \
    -X GET \
    -H "Content-Type: application/x-www-form-urlencoded" \
    "IDP_URL/CONTEXT/realms/REALM/protocol/openid-connect/certs" \
    | jq .
```

示例

```
[user@client ~]$ curl -k -v \
    -X GET \
    -H "Content-Type: application/x-www-form-urlencoded" \
    "http://www.example.com/auth/realms/quickstart/protocol/openid-connect/certs" \
    | jq .
```

注意

x5c 证书可以在 /certs 路径或 /jwks 路径中可用，具体取决于 SSO 提供程序。

3. ***复制上一命令中的"x5c"响应的结果，并将它粘贴到 certificate.crt 文件中。在结尾处包括 --BEGIN CERTIFICATE-- 和 --END CERTIFICATE--。***

示例

```
-----BEGIN CERTIFICATE-----

MIIDYjCCAkqgAwIBAgIEEEd2CDANBgkqhkiG9w0BAQsFADBzMQkwBwYDVQQGEwAxCTA
HBgNVBAgTADEJMAcGA1UEBxMAMQkwBwYDVQQKEwAxCTAHBgNVBAsTADE6MDgGA1
UEAxMxYXV0aHN2Yy1pbmxpbmVtZmEuZGV2LnZlcmlmeS5pYm1jbG91ZHNlY3VyaXR5LmN
vbTAeFw0yMTA3MDUxMzU2MzZaFw0zMTA3MDMxMzU2MzZaMHMxCTAHBgNVBAYTADE
JMAcGA1UECBMAMQkwBwYDVQQHEwAxCTAHBgNVBAoTADEJMAcGA1UECxMAMTowO
AYDVQQDEzFhdXRoc3ZjLWlubGluZW1mYS5kZXYudmVyaWZ5LmlibWNsb3Vkc2VjdXJpdHk
uY29tMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAphyu3HaAZ14JH/EXetZxtNn
erNuqcnfxcmLhBz9SsTlFD59ta+BOVlRnK5SdYEqO3ws2iGEzTvC55rczF+hDVHFZEBJLVLQe
8ABmi22RAtG1P0dA/Bq8ReFxpOFVWJUBc31QM+ummW0T4yw44wQJI51LZTMz7PznB0Sc
pObxKe+frFKd1TCMXPlWOSzmTeFYKzR83Fg9hsnz7Y8SKGxi+RoBbTLT+ektfWpR7O+oWZ
If4INe1VYJRxZvn+qWcwI5uMRCtQkiMknc3Rj6Eupiqq6FlAjDs0p//EzsHAIW244jMYnHCGq0UP
3oE7vViLJyiOmZw7J3rvs3m9mOQiPLoQIDAQABMA0GCSqGSIb3DQEBCwUAA4IBAQCeVq
AzSh7Tp8LgaTIFUuRbdjBAKXC9Nw3+pRBHoiUTdhqO3ualyGih9m/js/clb8Vq/39zl0VPeaslWl2
NNX9zaK7xo+ckVIOY3ucCaTC04ZUn1KzZu/7azlN0C5XSWg/CfXgU2P3BeMNzc1UNY1BAS
GyWn2lEplIVWKLaDZpNdSyyGyaoQAIBdzxeNCyzDfPCa2oSO8WH1czmFiNPqR5kdknHl96C
msQdi+DT4jwzVsYgrLfcHXmiWyIAb883hR3Pobp+Bsw7LUnxebQ5ewccjYmrJzOk5Wb5FpXBh
aJH1B3AEd6RGalRUyc/zUKdvEy0nIRMDS9x2BP3NVvZSADD

-----END CERTIFICATE-----
```

4. **获取证书指纹：**

语法

```
openssl x509 -in CERT_FILE -fingerprint -noout
```

示例

```
[user@client ~]$ openssl x509 -in certificate.crt -fingerprint -noout
SHA1 Fingerprint=F7:D7:B3:51:5D:D0:D3:19:DD:21:9A:43:A9:EA:72:7A:D6:06:52:87
```

5. 从 SHA1 指纹中删除所有冒号，并将其用作在 IAM 请求中创建 IDP 实体的输入。

其它资源

- 请参阅 Amazon 网站上的 OpenID Connect 身份提供程序的 Obtaining the Root CA Thumbprint。
- 有关 STS API 的详情，请参阅 Red Hat Ceph Storage 开发者指南中的 安全令牌服务应用程序编程接口 部分。
- 如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的使用 安全令牌服务 API 部分的示例。

3.2.8.5. 注册 OpenID Connect 供应商

注册 OpenID Connect 供应商(IDP)配置文档。

先决条件

- 安装 openssl 和 curl 软件包。
- 配置了安全令牌服务。
- 为 OIDC 供应商创建的用户。
- 获取 OIDC 的指纹。

流程

1. 从令牌中提取 URL。

示例

```
[root@host01 ~]# bash check_token_isv.sh | jq .iss

"https://keycloak-sso.apps.ocp.example.com/auth/realms/ceph"
```

2. *使用 Ceph 对象网关注册 OIDC 供应商。*

示例

```
[root@host01 ~]# aws --endpoint https://cephproxy1.example.com:8443 iam create-open-id-connect-provider --url https://keycloak-sso.apps.ocp.example.com/auth/realms/ceph --thumbprint-list 00E9CFD697E0B16DD13C86B0FFDC29957E5D24DF
```

3. *验证 OIDC 供应商是否已添加到 Ceph 对象网关。*

示例

```
[root@host01 ~]# aws --endpoint https://cephproxy1.example.com:8443 iam list-open-id-connect-providers

{
 "OpenIDConnectProviderList": [
 {
 "Arn":
"arn:aws:iam:::oidc-provider/keycloak-sso.apps.ocp.example.com/auth/realms/ceph"
 }
 ]
}
```

3.2.8.6. 创建 IAM 角色和策略

创建 IAM 角色和策略。

先决条件

- *安装 openssl 和 curl 软件包。*
- *配置了安全令牌服务。*
- *为 OIDC 供应商创建的用户。*
- *获取 OIDC 的指纹。*
- *注册的 Ceph 对象网关中的 OIDC 供应商。*

流程

1. *检索并验证 JWT 令牌。*

   示例

```
[root@host01 ~]# curl -k -q -L -X POST
"https://keycloak-sso.apps.example.com/auth/realms/ceph/protocol/openid-connect/
token" \
-H 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'client_id=ceph' \
--data-urlencode 'grant_type=password' \
--data-urlencode 'client_secret=XXXXXXXXXXXXXXXXXXXXXXXX' \
--data-urlencode 'scope=openid' \
--data-urlencode "username=SSOUSERNAME" \
--data-urlencode "password=SSOPASSWORD"
```

2. *验证令牌。*

示例

```
[root@host01 ~]# cat check_token.sh
USERNAME=$1
PASSWORD=$2
KC_CLIENT="ceph"
KC_CLIENT_SECRET="7sQXqyMSzHIeMcSALoKaljB6sNIBDRjU"
KC_ACCESS_TOKEN="$(./get_web_token.sh $USERNAME $PASSWORD | jq -r
'.access_token')"
KC_SERVER="https://keycloak-sso.apps.ocp.stg.local"
KC_CONTEXT="auth"
KC_REALM="ceph"
curl -k -s -q \
-X POST \
-u "$KC_CLIENT:$KC_CLIENT_SECRET" \
-d "token=$KC_ACCESS_TOKEN" \
"$KC_SERVER/$KC_CONTEXT/realms/$KC_REALM/protocol/openid-
connect/token/introspect" | jq .


[root@host01 ~]# ./check_token.sh s3admin passw0rd | jq .sub
"ceph"
```

在本例中，令牌中的子字段使用 jq 过滤器，并设置为 ceph。

3. **创建具有角色属性的 JSON 文件。将 statement 设置为 Allow，将 Action 设为 AssumeRoleWithWebIdentity。允许使用与 sub:ceph 条件匹配的 JWT 令牌访问任何用户。**

示例

```
[root@host01 ~]# cat role-rgwadmins.json
{
 "Version": "2012-10-17",
 "Statement": [
   {
     "Effect": "Allow",
     "Principal": {
       "Federated": [
         "arn:aws:iam:::oidc-provider/keycloak-sso.apps.example.com/auth/realms/ceph"
       ]
     },
     "Action": [
```

```
      "sts:AssumeRoleWithWebIdentity"
    ],
    "Condition": {
      "StringLike": { "keycloak-sso.apps.example.com/auth/realms/ceph:sub":"ceph" }
    }
  }
 ]
}
```

4. **使用 JSON 文件创建 Ceph 对象网关角色。**

示例

```
[root@host01 ~]# radosgw-admin role create --role-name rgwadmins \
--assume-role-policy-doc=$(jq -rc . /root/role-rgwadmins.json)
```

.

3.2.8.7. 访问 S3 资源

使用 STS 凭证验证 Assume Role 以访问 S3 资源。

先决条件

- **安装 openssl 和 curl 软件包。**
- **配置了安全令牌服务。**
- **为 OIDC 供应商创建的用户。**
- **获取 OIDC 的指纹。**

- *注册的 Ceph 对象网关中的 OIDC 供应商。*
- *已创建 IAM 角色和策略*

流程

1. *以下是假设 Role with STS 为获取临时访问和 secret 密钥来访问 S3 资源的示例。*

```
[roo@host01 ~]# cat test-assume-role.sh
#!/bin/bash
export AWS_CA_BUNDLE="/etc/pki/ca-trust/source/anchors/cert.pem"
unset AWS_ACCESS_KEY_ID
unset AWS_SECRET_ACCESS_KEY
unset AWS_SESSION_TOKEN
KC_ACCESS_TOKEN=$(curl -k -q -L -X POST
"https://keycloak-sso.apps.ocp.example.com/auth/realms/ceph/protocol/openid-connect/
token" \
-H 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'client_id=ceph' \
--data-urlencode 'grant_type=password' \
--data-urlencode 'client_secret=XXXXXXXXXXXXXXXXXXXXXXXX' \
--data-urlencode 'scope=openid' \
--data-urlencode "<varname>SSOUSERNAME</varname>" \
--data-urlencode "<varname>SSOPASSWORD</varname>" | jq -r .access_token)
echo ${KC_ACCESS_TOKEN}
IDM_ASSUME_ROLE_CREDS=$(aws sts assume-role-with-web-identity --role-arn
"arn:aws:iam:::role/$3" --role-session-name testbr
--endpoint=https://cephproxy1.example.com:8443
--web-identity-token="$KC_ACCESS_TOKEN")
echo "aws sts assume-role-with-web-identity --role-arn "arn:aws:iam:::role/$3"
--role-session-name testb --endpoint=https://cephproxy1.example.com:8443
--web-identity-token="$KC_ACCESS_TOKEN""
echo $IDM_ASSUME_ROLE_CREDS
export AWS_ACCESS_KEY_ID=$(echo $IDM_ASSUME_ROLE_CREDS | jq -r
.Credentials.AccessKeyId)
export AWS_SECRET_ACCESS_KEY=$(echo $IDM_ASSUME_ROLE_CREDS | jq -r
.Credentials.SecretAccessKey)
export AWS_SESSION_TOKEN=$(echo $IDM_ASSUME_ROLE_CREDS | jq -r
.Credentials.SessionToken)
```

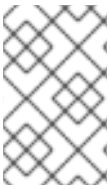
2. *运行脚本。*

示例

```
[root@host01 ~]# source ./test-assume-role.sh s3admin passw0rd rgwadmins
[root@host01 ~]# aws s3 mb s3://testbucket
[root@host01 ~]# aws s3 ls
```

3.2.9. 在 Keystone 中配置和使用 STS Lite（技术预览）

同一命名空间中的 Amazon 安全令牌服务(STS)和 S3 API 共存。STS 选项可以和 Keystone 选项一起配置。



注意

S3 和 STS API 都可以使用 Ceph 对象网关中的同一端点来访问。

先决条件

- Red Hat Ceph Storage 5.0 或更高版本。
- 正在运行的 Ceph 对象网关.
- 安装 Boto Python 模块，版本 3 或更高版本。
- Ceph 管理器节点的 root 级别访问。
- OpenStack 节点的用户级访问权限。

流程

1. 为 Ceph 对象网关客户端设置以下配置选项：

语法

```
ceph config set RGW_CLIENT_NAME rgw_sts_key STS_KEY
ceph config set RGW_CLIENT_NAME rgw_s3_auth_use_sts true
```

rgw_sts_key 是用于加密或解密会话令牌的 STS 密钥，且正好为 16 十六进制字符。

重要

STS 密钥需要是字母数字。

示例

```
[root@mgr ~]# ceph config set client.rgw rgw_sts_key 7f8fd8dd4700mnop
[root@mgr ~]# ceph config set client.rgw rgw_s3_auth_use_sts true
```

2. **在 OpenStack 节点上生成 EC2 凭证：**

示例

```
[user@osp ~]$ openstack ec2 credentials create

+------------+---------------------------------------------------------+
| Field      | Value                                                   |
+------------+---------------------------------------------------------+
| access     | b924dfc87d454d15896691182fdeb0ef                        |
| links      | {u'self': u'http://192.168.0.15/identity/v3/users/      |
|            | 40a7140e424f493d8165abc652dc731c/credentials/           |
|            | OS-EC2/b924dfc87d454d15896691182fdeb0ef'}               |
| project_id | c703801dccaf4a0aaa39bec8c481e25a                        |
| secret     | 6a2142613c504c42a94ba2b82147dc28                        |
| trust_id   | None                                                    |
| user_id    | 40a7140e424f493d8165abc652dc731c                        |
+------------+---------------------------------------------------------+
```

3. *使用生成的凭证，通过 **GetSessionToken API** 来备份一组临时安全凭证：*

示例

```
import boto3

access_key = b924dfc87d454d15896691182fdeb0ef
secret_key = 6a2142613c504c42a94ba2b82147dc28

client = boto3.client('sts',
aws_access_key_id=access_key,
aws_secret_access_key=secret_key,
endpoint_url=https://www.example.com/rgw,
region_name='',
)

response = client.get_session_token(
    DurationSeconds=43200
)
```

4. *获取临时凭证可用于发出 **S3** 调用：*

示例

```
s3client = boto3.client('s3',
  aws_access_key_id = response['Credentials']['AccessKeyId'],
  aws_secret_access_key = response['Credentials']['SecretAccessKey'],
  aws_session_token = response['Credentials']['SessionToken'],
  endpoint_url=https://www.example.com/s3,
  region_name='')

bucket = s3client.create_bucket(Bucket='my-new-shiny-bucket')
response = s3client.list_buckets()
for bucket in response["Buckets"]:
  print "{name}\t{created}".format(
    name = bucket['Name'],
    created = bucket['CreationDate'],
  )
```

5. *创建新的 S3Access 角色并配置策略。*

   a. *使用管理 CAPS 为用户分配：*

   语法

   ```
   radosgw-admin caps add --uid="USER" --caps="roles=*"
   ```

   示例

   ```
   [root@mgr ~]# radosgw-admin caps add --uid="gwadmin" --caps="roles=*"
   ```

   b. *创建 S3Access 角色：*

   语法

   ```
   radosgw-admin role create --role-name=ROLE_NAME --path=PATH --assume-role-
   policy-doc=TRUST_POLICY_DOC
   ```

   示例

   ```
   [root@mgr ~]# radosgw-admin role create --role-name=S3Access --
   ```

```
path=/application_abc/component_xyz/ --assume-role-policy-doc=\{\"Version\":\"2012-10-17\",\"Statement\":\[\{\"Effect\":\"Allow\",\"Principal\":\{\"AWS\":\[\"arn:aws:iam:::user/TESTER\"\]\},\"Action\":\[\"sts:AssumeRole\"\]\}\]\}
```

c. ***将权限策略附加到 S3Access 角色：***

语法

```
radosgw-admin role-policy put --role-name=ROLE_NAME --policy-name=POLICY_NAME --policy-doc=PERMISSION_POLICY_DOC
```

示例

```
[root@mgr ~]# radosgw-admin role-policy put --role-name=S3Access --policy-name=Policy --policy-doc=\{\"Version\":\"2012-10-17\",\"Statement\":\[\{\"Effect\":\"Allow\",\"Action\":\[\"s3:*\"\],\"Resource\":\"arn:aws:s3:::example_bucket\"\}\]\}
```

d. ***现在，另一个用户可以假定 gwadmin 用户的角色。例如，gwuser 用户可以假定 gwadmin 用户权限。***

e. ***记录假定用户的 access_key 和 secret_key 值。***

示例

```
[root@mgr ~]# radosgw-admin user info --uid=gwuser | grep -A1 access_key
```

6. 使用 AssumeRole API 调用，提供来自假定用户的 access_key 和 secret_key 值：

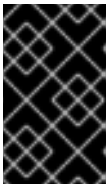
示例

```
import boto3

access_key = 11BS02LGFB6AL6H1ADMW
secret_key = vzCEkuryfn060dfee4fgQPqFrncKEIkh3ZcdOANY

client = boto3.client('sts',
aws_access_key_id=access_key,
aws_secret_access_key=secret_key,
endpoint_url=https://www.example.com/rgw,
region_name='',
)

response = client.assume_role(
RoleArn='arn:aws:iam:::role/application_abc/component_xyz/S3Access',
RoleSessionName='Bob',
DurationSeconds=3600
)
```



重要

AssumeRole API 需要 S3Access 角色。

其它资源

- 有关安装 Boto Python 模块的更多信息，请参阅 Red Hat Ceph Storage Object Gateway 指南中的 Test S3 Access 部分。
- 如需更多信息，请参阅 Red Hat Ceph Storage 对象网关指南中的创建用户部分。

3.2.10. 围绕将 STS Lite 与 Keystone 搭配使用的限制（技术预览）

Keystone 的一个限制是它不支持安全令牌服务(STS)请求。请求中不包含另一个限制的有效负载哈希。要解决这两个限制，必须修改 Boto 身份验证代码。

先决条件

- *正在运行的 Red Hat Ceph Storage 集群，版本 5.0 或更高版本。*
- *正在运行的 Ceph 对象网关.*
- *安装 Boto Python 模块，版本 3 或更高版本。*

流程

1. *打开并编辑 Boto 的 auth.py 文件。*

   a. *将以下四行添加到代码块中：*

```
class SigV4Auth(BaseSigner):
  """
  Sign a request with Signature V4.
  """
  REQUIRES_REGION = True

  def __init__(self, credentials, service_name, region_name):
     self.credentials = credentials
     # We initialize these value here so the unit tests can have
     # valid values.  But these will get overriden in ``add_auth``
     # later for real requests.
     self._region_name = region_name
     if service_name == 'sts': (1)
        self._service_name = 's3' (2)
     else: (3)
        self._service_name = service_name (4)
```

   b. *将以下两行添加到代码块中：*

```
def _modify_request_before_signing(self, request):
       if 'Authorization' in request.headers:
          del request.headers['Authorization']
       self._set_necessary_date_headers(request)
       if self.credentials.token:
          if 'X-Amz-Security-Token' in request.headers:
             del request.headers['X-Amz-Security-Token']
          request.headers['X-Amz-Security-Token'] = self.credentials.token

       if not request.context.get('payload_signing_enabled', True):
```

```
        if 'X-Amz-Content-SHA256' in request.headers:
            del request.headers['X-Amz-Content-SHA256']
        request.headers['X-Amz-Content-SHA256'] = UNSIGNED_PAYLOAD (1)
    else: (2)
        request.headers['X-Amz-Content-SHA256'] = self.payload(request)
```

其它资源

- *有关安装 Boto Python 模块的更多信息，请参阅 Red Hat Ceph Storage Object Gateway 指南中的 Test S3 Access 部分。*

3.3. S3 存储桶操作

作为开发者，您可以通过 Ceph 对象网关通过 Amazon S3 应用编程接口(API)执行存储桶操作。

下表列出了存储桶的 Amazon S3 功能操作，以及功能的支持状态。

表 3.2. bucket 操作

功能	Status	备注
List Buckets	支持	
创建 Bucket	支持	不同的一组可实施 ACL。
Put Bucket Website	支持	
Get Bucket Website	支持	
Delete Bucket Website	支持	
放置 Bucket 复制	支持	
Get Bucket 复制	支持	
删除 Bucket 复制	支持	
bucket 生命周期	部分支持	支持 **Expiration**, **NoncurrentVersionExpiration** 和 **AbortIncompleteMultipartUpload**。
Put Bucket Lifecycle	部分支持	支持 **Expiration**, **NoncurrentVersionExpiration** 和 **AbortIncompleteMultipartUpload**。

功能	Status	备注
Delete Bucket Lifecycle	支持	
Get Bucket Objects	支持	
bucket 位置	支持	
Get Bucket Version	支持	
Put Bucket Version	支持	
删除 Bucket	支持	
获取 Bucket ACL	支持	不同的可处理 ACL 集
放置 Bucket ACL	支持	不同的可处理 ACL 集
Get Bucket cors	支持	
Put Bucket cors	支持	
Delete Bucket cors	支持	
列出 Bucket 对象版本	支持	
head Bucket	支持	
List Bucket Multipart Uploads	支持	
bucket 策略	部分支持	
Get a Bucket Request Payment	支持	
Put a Bucket Request Payment	支持	
多租户 Bucket 操作	支持	
GET PublicAccessBlock	支持	
PUT PublicAccessBlock	支持	
删除 PublicAccessBlock	支持	

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- RESTful 客户端。

3.3.1. S3 创建存储桶通知

在 bucket 级别上创建 bucket 通知。通知配置具有 Red Hat Ceph Storage Object Gateway S3 事件、ObjectCreated、ObjectRemoved 和 ObjectLifecycle:Expiration。这些需要发布，以及发送 bucket 通知的目的地。bucket 通知是 S3 操作。

要为 s3:objectCreate,s3:objectRemove 和 s3:ObjectLifecycle:Expiration 事件创建存储桶通知，请使用 PUT：

示例

```
client.put_bucket_notification_configuration(
   Bucket=bucket_name,
   NotificationConfiguration={
      'TopicConfigurations': [
         {
            'Id': notification_name,
            'TopicArn': topic_arn,
            'Events': ['s3:ObjectCreated:*', 's3:ObjectRemoved:*', 's3:ObjectLifecycle:Expiration:*']
         }]})
```

重要

红帽支持 ObjectCreate 事件，如 put, post, multipartUpload, 和 copy。红帽还支持 ObjectRemove 事件，如 object_delete 和 s3_multi_object_delete。

请求实体

NotificationConfiguration

描述

TopicConfiguration 实体列表。

Type

Container

必需

是

TopicConfiguration

描述

事件主题的 Id, Topic 和 list。

Type

Container

必需

是

id

描述

通知的名称。

Type

字符串

必需

是

Topic

描述

Topic Amazon Resource Name(ARN)

注意

必须事先创建主题。

Type

字符串

必需

是

事件

描述

支持的事件列表。可以使用多个事件实体。如果省略，则处理所有事件。

Type

字符串

必需

否

Filter

描述

S3Key、S3Metadata 和 S3Tags 实体。

Type

Container

必需

否

S3Key

描述

FilterRule 实体列表，用于根据对象密钥进行过滤。列表中最多可以有 3 个实体，例如，Name 为 prefix, suffix, 或 regex。列表中的所有过滤规则必须与过滤器匹配。

Type

Container

必需

否

S3Metadata

描述

FilterRule 实体列表，用于根据对象元数据进行过滤。列表中的所有过滤规则必须与对象中定义的元数据匹配。但是，如果对象具有过滤器中没有列出的其他元数据条目，则对象仍然匹配。

Type

Container

必需

否

S3Tags

描述

FilterRule 实体列表，用于根据对象标签进行过滤。列表中的所有过滤规则必须与对象中定义的标签匹配。但是，如果对象没有列在过滤器中，则对象仍然匹配。

Type

Container

必需

否

S3Key.FilterRule

描述

Name 和 Value 实体。Name 为： prefix,suffix, 或 regex.Value 将保存用于匹配键的密钥前缀、密钥后缀或正则表达式。

Type

Container

必需

是

S3Metadata.FilterRule

描述

Name 和 Value 实体。name 是 x-amz-meta-xxx 等 metadata 属性的名称。该值是此属性的预期值。

Type

Container

必需

是

S3Tags.FilterRule

描述

Name 和 Value 实体。name 是 tag 键，值为 tag 值。

Type

Container

必需

是

HTTP 响应

400

状态代码

MalformedXML

描述

XML 不正确。

400

状态代码

InvalidArgument

描述

缺少 Id 或缺失或无效主题 ARN 或无效的事件。

404

状态代码

NoSuchBucket

描述

bucket 不存在。

404

状态代码

NoSuchKey

描述

主题不存在。

3.3.2. S3 获取存储桶通知

获取特定的通知，或列出存储桶中配置的所有通知。

语法

```
Get /BUCKET?notification=NOTIFICATION_ID HTTP/1.1
Host: cname.domain.com
Date: date
Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

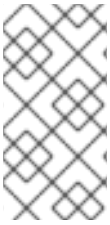
示例

```
Get /testbucket?notification=testnotificationID HTTP/1.1
Host: cname.domain.com
```

```
Date: date
Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

响应示例

```
<NotificationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <TopicConfiguration>
      <Id></Id>
      <Topic></Topic>
      <Event></Event>
      <Filter>
         <S3Key>
            <FilterRule>
               <Name></Name>
               <Value></Value>
            </FilterRule>
          </S3Key>
         <S3Metadata>
            <FilterRule>
               <Name></Name>
               <Value></Value>
            </FilterRule>
         </S3Metadata>
         <S3Tags>
            <FilterRule>
               <Name></Name>
               <Value></Value>
            </FilterRule>
         </S3Tags>
       </Filter>
   </TopicConfiguration>
</NotificationConfiguration>
```



注意

notification 子资源返回存储桶通知配置或空 notification Configuration 元素。调用者必须是存储桶所有者。

请求实体

notification-id

描述

通知的名称。如果未提供 ID，则会列出所有通知。

Type

字符串

NotificationConfiguration

描述

TopicConfiguration 实体列表。

Type

Container

必需

是

TopicConfiguration

描述

事件主题的 Id, Topic 和 list。

Type

Container

必需

是

id

描述

通知的名称。

Type

字符串

必需

是

Topic

描述

Topic Amazon Resource Name(ARN)

注意

必须事先创建主题。

Type

字符串

必需

是

事件

描述

处理的事件。可能存在多个事件实体。

Type

字符串

必需

是

Filter

描述

指定配置的过滤器。

Type

Container

必需

否

HTTP 响应

404

状态代码

NoSuchBucket

描述

bucket 不存在。

404

状态代码

NoSuchKey

描述

如果提供了通知，则不会存在。

3.3.3. S3 删除存储桶通知

从存储桶中删除特定或所有通知。

注意

通知删除是对 S3 通知 API 的扩展。当存储桶被删除时，存储桶上所有定义的通知都会被删除。删除一个未知的示例 double delete 的通知不会被视为错误。

要删除特定或所有通知，请使用 DELETE：

语法

```
DELETE /BUCKET?notification=NOTIFICATION_ID HTTP/1.1
```

示例

```
DELETE /testbucket?notification=testnotificationID HTTP/1.1
```

请求实体

notification-id

描述

通知的名称。如果没有提供通知 ID，则存储桶上的所有通知都会被删除。

Type

字符串

HTTP 响应

404

状态代码

NoSuchBucket

描述

bucket 不存在。

3.3.4. 访问存储桶主机名

访问存储桶时有两种不同的模式。第一种方法和首选的方法将 bucket 标识为 URI 中的顶级目录。

示例

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

第二种方法通过虚拟 bucket 主机名标识 bucket。

示例

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

提示

红帽首选第一种方法，因为第二种方法需要昂贵的域认证和 DNS 通配符。

3.3.5. S3 列表存储桶

GET / 返回由发出请求的用户创建的 bucket 列表。GET / 仅返回由经过身份验证的用户创建的存储桶。您不能发出匿名请求。

语法

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

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

响应实体

Buckets

描述

用于存储桶列表的容器。

Type

Container

Bucket

描述

用于存储桶信息的容器。

Type

Container

名称

描述

bucket 名称.

Type

字符串

CreationDate

描述

创建存储桶时的 UTC 时间。

Type

Date

ListAllMyBucketsResult

描述

用于结果的容器。

Type

Container

所有者

描述

bucket 所有者的 ID 和 DisplayName 的容器。

Type

Container

ID

描述

bucket 所有者的 ID。

Type

字符串

DisplayName

描述

bucket 所有者的显示名称。

Type

字符串

3.3.6. S3 返回存储桶对象列表

返回存储桶对象列表。

语法

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

参数

prefix

描述

仅返回包含指定前缀的对象。

Type

字符串

delimiter

描述

前缀与其他对象名称之间的分隔符。

Type

字符串

marker

描述

返回的对象列表的开头索引。

Type

字符串

max-keys

描述

要返回的最大密钥数。默认值为 1000。

Type

整数

HTTP 响应

200

状态代码

确定

描述

检索的 bucket。

GET /BUCKET 返回存储桶的容器，并带有以下字段：

bucket 响应实体

ListBucketResult

描述

容器，用于对象列表。

Type

实体

名称

描述

其内容要返回的存储桶的名称。

Type

字符串

prefix

描述

对象密钥的前缀。

Type

字符串

Marker

描述

返回的对象列表的开头索引。

Type

字符串

MaxKeys

描述

返回的最大键数。

Type

整数

Delimiter

描述

如果设置，具有相同前缀的对象将显示在 CommonPrefixes 列表中。

Type

字符串

IsTruncated

描述

如果为 true，则仅返回存储桶内容的子集。

Type

布尔值

CommonPrefixes

描述

如果多个对象包含相同的前缀，它们将显示在此列表中。

Type

Container

ListBucketResult 包含对象，其中每个对象都在一个 Contents 容器中。

对象响应实体

内容

描述

对象的容器。

Type

Object

键

描述

对象的密钥。

Type

字符串

LastModified

描述

对象的最后修改日期和时间。

Type

Date

ETag

描述

对象的 MD-5 哈希。ETag 是一个实体标签。

Type

字符串

大小

描述

对象的大小。

Type

整数

StorageClass

描述

应始终返回 STANDARD。

Type

字符串

3.3.7. S3 创建新存储桶

创建新存储桶。要创建存储桶，必须有用户 ID 和有效的 AWS 访问密钥 ID 才能验证请求。您不能以匿名用户身份创建存储桶。

约束

通常，存储桶名称应该遵循域名约束。

- bucket 名称必须是唯一的。
- bucket 名称不能格式化为 IP 地址。
- bucket 名称长度可在 3 到 63 个字符之间。
- bucket 名称不得包含大写字符或下划线。
- bucket 名称必须以小写或数字开头。
- bucket 名称可以包含短划线(-)。
- bucket 名称必须是由一个或多个标签组成的系列。相邻标签由单个句点(.)分隔。bucket 名称可以包含小写字母、数字和连字符。每个标签必须以小写或数字开头和结尾。

注意

如果 **rgw_relaxed_s3_bucket_names** 设为 **true**，则上述限制会被放松。**bucket** 名称仍必须是唯一的，不能格式化为 **IP** 地址，可以包含字母、数字、句点、短划线和下划线，最多 **255** 个字符。

语法

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

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

参数

x-amz-acl

描述

可以拒绝 **ACL**。

有效值

私有,**public-read,public-read-write,authenticated-read**

必需

否

HTTP 响应

如果存储桶名称是唯一的，在约束范围内，且未使用，则操作会成功。如果具有相同名称的存储桶已经存在，并且该用户是 **bucket** 所有者，则操作将成功。如果存储桶名称已经在使用中，则操作将失败。

409

状态代码

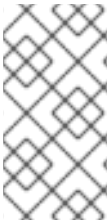
BucketAlreadyExists

描述

bucket 已存在于不同用户的所有权下。

3.3.8. S3 put bucket 网站

put bucket 网站 API 设置在 website 子资源中指定的网站配置。要将存储桶配置为网站，可在存储桶中添加 website 子资源。



注意

Put 操作需要 S3:PutBucketWebsite 权限。默认情况下，只有 bucket 所有者才能配置附加到存储桶的网站。

语法

```
PUT /BUCKET?website-configuration=HTTP/1.1
```

示例

```
PUT /testbucket?website-configuration=HTTP/1.1
```

其它资源

- *有关此 API 调用的更多信息，请参阅 S3 API。*

3.3.9. S3 get bucket 网站

get bucket 网站 API 检索在 website 子资源中指定的网站配置。

注意

Get 操作需要 S3:GetBucketWebsite 权限。默认情况下，只有存储桶所有者才能读取存储桶网站配置。

语法

```
GET /BUCKET?website-configuration=HTTP/1.1
```

示例

```
GET /testbucket?website-configuration=HTTP/1.1
```

其它资源

- *有关此 API 调用的更多信息，请参阅 S3 API。*

3.3.10. S3 删除存储桶网站

删除存储桶网站 API 会删除存储桶的网站配置。

语法

```
DELETE /BUCKET?website-configuration=HTTP/1.1
```

示例

```
DELETE /testbucket?website-configuration=HTTP/1.1
```

其它资源

- *有关此 API 调用的更多信息，请参阅 S3 API。*

3.3.11. S3 放置存储桶复制

放置存储桶复制 API 为存储桶配置复制配置，或替换现有的配置。

语法

```
PUT /BUCKET?replication HTTP/1.1
```

示例

```
PUT /testbucket?replication HTTP/1.1
```

3.3.12. S3 获取存储桶复制

get bucket 复制 API 返回存储桶的复制配置。

语法

```
GET /BUCKET?replication HTTP/1.1
```

示例

```
GET /testbucket?replication HTTP/1.1
```

3.3.13. S3 删除存储桶复制

删除存储桶复制 API 从存储桶中删除复制配置。

语法

```
DELETE /BUCKET?replication HTTP/1.1
```

示例

```
DELETE /testbucket?replication HTTP/1.1
```

3.3.14. S3 删除存储桶

删除存储桶。您可以在成功删除存储桶后重复使用存储桶名称。

语法

```
DELETE /BUCKET HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

HTTP 响应

204

状态代码

无内容

描述

已删除存储桶。

3.3.15. S3 存储桶生命周期

您可以使用存储桶生命周期配置来管理对象，以便在其生命周期内有效存储它们。Ceph 对象网关中的 S3 API 支持 AWS 存储桶生命周期操作的子集：

- ***Expiration: 定义存储桶内对象的寿命。它取对象应实时或过期日期的天数，该日期指向 Ceph 对象网关将删除对象。如果存储桶没有启用版本控制，Ceph 对象网关将永久删除对象。如果 bucket 启用版本控制，Ceph 对象网关为当前版本创建删除标记，然后删除当前版本。***

- ***NoncurrentVersionExpiration ：这定义了存储桶中非当前对象版本的期限。要使用此功能，您必须启用存储桶版本控制。它取非当前对象应处于活动状态的天数，此时 Ceph 对象网关将删除非当前对象。***

- ***NewerNoncurrentVersions ：指定要保留的非当前对象版本数量。您可以指定要保留的最多 100 个非当前版本。如果要保留的指定数量超过 100，则删除额外的非当前版本。***

- ***AbortIncompleteMultipartUpload ：这定义了未完成的多部分上传在中止前的天数。***

- **BlockPublicPolicy reject ：此操作用于公共访问块。如果指定的策略（用于访问点或底层存储桶）允许公共访问，它将调用通过访问点进行的 PUT 访问策略和 PUT bucket 策略。Amazon S3 Block Public Access 功能包括在 Red Hat Ceph Storage 5.x/ Ceph Pacific 版本中。它提供访问点、存储桶和帐户的设置，以帮助您管理 Amazon S3 资源的公共访问权限。默认情况下，新的 bucket、接入点和对象不允许公共访问。但是，您可以修改存储桶策略、访问点策略或对象权限来允许公共访问。S3 Block Public Access 设置覆盖这些策略和权限，以便您可以限制这些资源的公共访问权限。**

生命周期配置包含使用 <Rule> 元素的一个或多个规则。

示例

```
<LifecycleConfiguration>
  <Rule>
    <Prefix/>
    <Status>Enabled</Status>
    <Expiration>
      <Days>10</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

生命周期规则可应用于存储桶中所有或对象子集，具体取决于您在生命周期规则中指定的 <Filter > 元素。您可以通过几种方法指定过滤器：

- **密钥前缀**
- **对象标签**
- **密钥前缀和一个或多个对象标签**

密钥前缀

您可以根据密钥名称前缀，将生命周期规则应用到对象的子集。例如，指定 <keypre/> 将应用到以 keypre/ 开头的对象：

-

```
<LifecycleConfiguration>
   <Rule>
      <Status>Enabled</Status>
      <Filter>
         <Prefix>keypre/</Prefix>
      </Filter>
   </Rule>
</LifecycleConfiguration>
```

您还可以将不同的生命周期规则应用到具有不同密钥前缀的对象：

```
<LifecycleConfiguration>
   <Rule>
      <Status>Enabled</Status>
      <Filter>
         <Prefix>keypre/</Prefix>
      </Filter>
   </Rule>
   <Rule>
      <Status>Enabled</Status>
      <Filter>
         <Prefix>mypre/</Prefix>
      </Filter>
   </Rule>
</LifecycleConfiguration>
```

对象标签

您可以使用 <Key> 和 <Value> 元素将生命周期规则应用到带有特定标签的对象：

```
<LifecycleConfiguration>
   <Rule>
      <Status>Enabled</Status>
      <Filter>
         <Tag>
            <Key>key</Key>
            <Value>value</Value>
         </Tag>
      </Filter>
   </Rule>
</LifecycleConfiguration>
```

前缀和一个或多个标签

在生命周期规则中，您可以根据密钥前缀和一个或多个标签指定过滤器。它们必须包括在 <And> 项中。过滤器只能有一个前缀，以及零个或多个标签：

```
<LifecycleConfiguration>
   <Rule>
```

```
        <Status>Enabled</Status>
        <Filter>
          <And>
            <Prefix>key-prefix</Prefix>
            <Tag>
              <Key>key1</Key>
              <Value>value1</Value>
            </Tag>
            <Tag>
              <Key>key2</Key>
              <Value>value2</Value>
            </Tag>
            ...
          </And>
        </Filter>
      </Rule>
    </LifecycleConfiguration>
```

其它资源

- 有关 *获取存储桶生命周期的详细信息，请参阅* **Red Hat Ceph Storage Developer Guide 中的 S3 GET 存储桶生命周期 部分。**
- 有关创建 *存储桶生命周期的详细信息，请参阅* **Red Hat Ceph Storage Developer Guide 中的 S3 创建或替换 存储桶生命周期部分。**
- 有关删除 *存储桶生命周期的详细信息，请参阅* **Red Hat Ceph Storage Developer Guide 中的 S3 删除存储桶生命周期部分。**

3.3.16. S3 GET 存储桶生命周期

要获取存储桶生命周期，请使用 GET 并指定目标存储桶。

语法

```
GET /BUCKET?lifecycle HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

请求 Headers

有关常见请求标头的更多信息，请参见附录 B 中的 S3 通用请求标头。

响应

响应中包含存储桶生命周期及其元素。

3.3.17. S3 创建或替换存储桶生命周期

要创建或替换存储桶生命周期，请使用 PUT，并指定目标存储桶和生命周期配置。Ceph 对象网关仅支持 S3 生命周期功能的子集。

语法

```
PUT /BUCKET?lifecycle HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
<LifecycleConfiguration>
  <Rule>
    <Expiration>
      <Days>10</Days>
    </Expiration>
  </Rule>
    ...
  <Rule>
  </Rule>
</LifecycleConfiguration>
```

请求 Headers

content-md5

描述

消息的 base64 编码 MD-5 哈希

有效值

字符串没有默认值或限制。

必需

否

其它资源

- 有关 Amazon S3 通用请求标头 的更多信息，请参阅 Red Hat Ceph Storage Developer Guide 附录 B 中的 S3 通用请求标头部分。

- 有关 Amazon S3 存储桶生命周期 的更多信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 S3 存储桶生命周期部分。

3.3.18. S3 删除存储桶生命周期

要删除存储桶生命周期，请使用 DELETE 并指定目标存储桶。

语法

```
DELETE /BUCKET?lifecycle HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

请求 Headers

请求不包含任何特殊元素。

响应

响应返回常见的响应状态。

其它资源

- 有关 Amazon S3 通用请求标头 的更多信息，请参阅 Red Hat Ceph Storage Developer Guide 附录 B 中的 S3 通用请求标头部分。

- 有关 Amazon S3 通用响应状态代码 的更多信息，请参阅 Red Hat Ceph Storage Developer Guide 附录 C 中的 S3 通用响应状态代码部分。

3.3.19. S3 get bucket 位置

检索 bucket 的 zone group。用户需要是存储桶所有者才能调用它。通过在 PUT 请求期间提供 LocationConstraint，可以将 bucket 限制到 zone group。

将 location 子资源添加到存储桶资源，如下所示。

语法

```
GET /BUCKET?location HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

响应实体

LocationConstraint

描述

bucket 所在的 zone 组，即 default zone group 的空字符串。

类型

字符串

3.3.20. S3 获取存储桶版本

检索 bucket 的版本控制状态。用户需要是存储桶所有者才能调用它。

将 versioning 子资源添加到存储桶资源，如下所示。

语法

```
GET /BUCKET?versioning HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

3.3.21. S3 放置存储桶版本

此子资源设置现有存储桶的 versioning 状态。用户需要是存储桶所有者来设置版本控制状态。如果存储桶上没有设置 versioning 状态，则没有版本状态。执行 GET 版本请求不会返回 versioning 状态值。

设置存储桶版本状态：

启用 ：为存储桶中的对象启用版本控制。添加到存储桶的所有对象都会获得一个唯一的版本 ID。暂停：禁用存储桶中对象的版本控制。添加到存储桶的所有对象都会接收版本 ID null。

语法

```
PUT /BUCKET?versioning HTTP/1.1
```

示例

```
PUT /testbucket?versioning HTTP/1.1
```

bucket 请求实体

VersioningConfiguration

描述

用于请求的容器。

Type

Container

Status

描述

设置存储桶的版本控制状态。有效值： Suspended/Enabled

Type

字符串

3.3.22. S3 获取存储桶访问控制列表

检索 bucket 访问控制列表。用户需要是存储桶所有者，或被授予存储桶的 READ_ACP 权限。

将 acl 子资源添加到存储桶请求，如下所示。

语法

```
GET /BUCKET?acl HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

响应实体

AccessControlPolicy

描述

用于响应的容器。

Type

Container

AccessControlList

描述

ACL 信息的容器。

Type

Container

所有者

描述

bucket 所有者的 ID 和 DisplayName 的容器。

Type

Container

ID

描述

bucket 所有者的 ID。

Type

字符串

DisplayName

描述

bucket 所有者的显示名称。

Type

字符串

Grant

描述

适用于 Grantee 和 Permission 的容器。

Type

Container

Grantee

描述

允许权限的用户的 DisplayName 和 ID 的容器。

Type

Container

权限

描述

提供给 Grantee 存储桶的权限。

Type

字符串

3.3.23. S3 放置存储桶访问控制列表

设置到现有存储桶的访问控制。用户需要是存储桶所有者，或被授予存储桶的 WRITE_ACP 权限。

将 acl 子资源添加到存储桶请求，如下所示。

语法

```
PUT /BUCKET?acl HTTP/1.1
```

请求实体

S3 列表多部分上传

AccessControlList

描述

ACL 信息的容器。

Type

Container

所有者

描述

bucket 所有者的 ID 和 DisplayName 的容器。

Type

Container

ID

描述

bucket 所有者的 ID。

Type

字符串

DisplayName

描述

bucket 所有者的显示名称。

Type

字符串

Grant

描述

适用于 Grantee 和 Permission 的容器。

Type

Container

Grantee

描述

允许权限的用户的 DisplayName 和 ID 的容器。

Type

Container

权限

描述

提供给 Grantee 存储桶的权限。

Type

字符串

3.3.24. S3 get bucket cors

检索为存储桶设置的 cors 配置信息。用户需要是存储桶所有者，或被授予存储桶的 READ_ACP 权限。

将 cors 子资源添加到存储桶请求，如下所示。

语法

```
GET /BUCKET?cors HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

3.3.25. S3 put bucket cors

为存储桶设置 cors 配置。用户需要是存储桶所有者，或被授予存储桶的 READ_ACP 权限。

将 cors 子资源添加到存储桶请求，如下所示。

语法

```
PUT /BUCKET?cors HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

3.3.26. S3 删除存储桶 cors

删除为存储桶设置的 cors 配置信息。用户需要是存储桶所有者，或被授予存储桶的 READ_ACP 权限。

将 cors 子资源添加到存储桶请求，如下所示。

语法

```
DELETE /BUCKET?cors HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

3.3.27. S3 列表存储桶对象版本

返回存储桶中所有对象版本的元数据列表。需要指向存储桶的 READ 访问权限。

将 version 子资源添加到存储桶请求，如下所示。

语法

```
GET /BUCKET?versions HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

您可以为 GET /BUCKET?version 指定参数，但不需要它们。

参数

prefix

描述

返回的 in-progress 上传，其密钥包含指定前缀。

Type

字符串

delimiter

描述

前缀与其他对象名称之间的分隔符。

Type

字符串

key-marker

描述

上传列表的开头标记。

Type

字符串

max-keys

描述

最大的 in-progress 上传数。默认值为 1000。

Type

整数

version-id-marker

描述

指定用于开始列表的对象版本。

Type

字符串

响应实体

KeyMarker

描述

通过 key-marker 请求参数指定的密钥标记（如果有）。

Type

字符串

NextKeyMarker

描述

如果 IsTruncated 为 true，则后续请求中使用的密钥标记。

Type

字符串

NextUploadIdMarker

描述

如果 IsTruncated 为 true，则上传 ID 标记用于后续请求。

Type

字符串

IsTruncated

描述

如果为 true，则仅返回存储桶上传内容的子集。

Type

布尔值

大小

描述

上传部分的大小。

Type

整数

DisplayName

描述

所有者的显示名称。

Type

字符串

ID

描述

所有者的 ID。

Type

字符串

所有者

描述

用于拥有对象的用户的 ID 和 DisplayName 的容器。

Type

Container

StorageClass

描述

用于存储生成的对象的方法。STANDARD 或 REDUCED_REDUNDANCY

Type

字符串

版本

描述

版本信息的容器。

Type

Container

versionId

描述

对象的版本 ID。

Type

字符串

versionIdMarker

描述

已截断响应中密钥的最后一个版本。

Type

字符串

3.3.28. S3 头存储桶

在存储桶上调用 HEAD，以确定它是否存在以及调用者是否有访问权限。如果存储桶存在并且调用者具有权限，则返回 200 OK ；如果存储桶不存在，则返回 404 Not Found。如果存储桶存在，但调用者没有访问权限，则返回 403 Forbidden。

语法

```
HEAD /BUCKET HTTP/1.1
Host: cname.domain.com
Date: date
Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

3.3.29. S3 列表多部分上传

GET /?uploads 返回当前 in-progress 多部分上传的列表，即应用程序启动多部分上传，但该服务尚未完成所有上传。

语法

```
GET /BUCKET?uploads HTTP/1.1
```

您可以为 GET /BUCKET?uploads 指定参数，但参数都不是必需的。

参数

prefix

描述

返回的 in-progress 上传，其密钥包含指定前缀。

Type

字符串

delimiter

描述

前缀与其他对象名称之间的分隔符。

Type

字符串

key-marker

描述

上传列表的开头标记。

Type

字符串

max-keys

描述

最大的 in-progress 上传数。默认值为 1000。

Type

整数

max-uploads

描述

多部分上传的最大数量。范围为 1 到1000。默认值为 1000。

Type

整数

version-id-marker

描述

*如果没有指定 **key-marker**，则忽略。指定第一个上传的 **ID** 以字典顺序（位于 ）或遵循 **ID** 的顺序来列出。*

Type

字符串

响应实体

ListMultipartUploadsResult

描述

用于结果的容器。

Type

Container

ListMultipartUploadsResult.Prefix

描述

***prefix** 请求参数中指定的前缀（若有）。*

Type

字符串

Bucket

描述

将接收存储桶内容的存储桶。

Type

字符串

KeyMarker

描述

*通过 **key-marker** 请求参数指定的密钥标记（如果有）。*

Type

字符串

UploadIdMarker

描述

upload-id-marker 请求参数中指定的标记（若有）。

Type

字符串

NextKeyMarker

描述

如果 IsTruncated 为 true，则后续请求中使用的密钥标记。

Type

字符串

NextUploadIdMarker

描述

如果 IsTruncated 为 true，则上传 ID 标记用于后续请求。

Type

字符串

MaxUploads

描述

max-uploads 请求参数指定的最大上传。

Type

整数

Delimiter

描述

如果设置，具有相同前缀的对象将显示在 CommonPrefixes 列表中。

Type

字符串

IsTruncated

描述

如果为 true，则仅返回存储桶上传内容的子集。

Type

布尔值

上传

描述

Key、UploadId、InitiatorOwner、StorageClass 和 Initiated 元素的容器。

Type

Container

键

描述

在多部分上传完成后，对象的键。

Type

字符串

UploadId

描述

标识多部分上传的 ID。

Type

字符串

initiator

描述

包含开始上传的用户的 ID 和 DisplayName。

Type

Container

DisplayName

描述

启动器的显示名称。

Type

字符串

ID

描述

启动器的 ID。

Type

字符串

所有者

描述

拥有上传对象的用户的 ID 和 DisplayName 的容器。

Type

Container

StorageClass

描述

用于存储生成的对象的方法。STANDARD 或 REDUCED_REDUNDANCY

Type

字符串

启动

描述

用户开始上传的日期和时间。

Type

Date

CommonPrefixes

描述

如果多个对象包含相同的前缀，它们将显示在此列表中。

Type

Container

CommonPrefixes.Prefix

描述

前缀后的键子字符串，如 prefix request 参数所定义。

Type

字符串

3.3.30. S3 存储桶策略

Ceph 对象网关支持应用到 bucket 的 Amazon S3 策略语言的子集。

创建和删除

Ceph 对象网关通过标准 S3 操作来管理 S3 Bucket 策略，而不使用 radosgw-admin CLI 工具。

管理员可以使用 s3cmd 命令来设置或删除策略。

示例

```
$ cat > examplepol
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {"AWS": ["arn:aws:iam::usfolks:user/fred"]},
    "Action": "s3:PutObjectAcl",
    "Resource": [
      "arn:aws:s3:::happybucket/*"
    ]
  }]
}

$ s3cmd setpolicy examplepol s3://happybucket
$ s3cmd delpolicy s3://happybucket
```

限制

***Ceph** 对象网关仅支持以下 **S3** 操作：*

- ***s3:AbortMultipartUpload***
- ***s3:CreateBucket***
- ***s3:DeleteBucketPolicy***
- ***s3:DeleteBucket***
- ***s3:DeleteBucketWebsite***
- ***s3:DeleteBucketReplication***

- *s3:DeleteReplicationConfiguration*
- *s3:DeleteObject*
- *s3:DeleteObjectVersion*
- *s3:GetBucketAcl*
- *s3:GetBucketCORS*
- *s3:GetBucketLocation*
- *s3:GetBucketPolicy*
- *s3:GetBucketRequestPayment*
- *s3:GetBucketVersioning*
- *s3:GetBucketWebsite*
- *s3:GetBucketReplication*
- *s3:GetReplicationConfiguration*
- *s3:GetLifecycleConfiguration*
- *s3:GetObjectAcl*

- *s3:GetObject*
- *s3:GetObjectTorrent*
- *s3:GetObjectVersionAcl*
- *s3:GetObjectVersion*
- *s3:GetObjectVersionTorrent*
- *s3:ListAllMyBuckets*
- *s3:ListBucketMultiPartUploads*
- *s3:ListBucket*
- *s3:ListBucketVersions*
- *s3:ListMultipartUploadParts*
- *s3:PutBucketAcl*
- *s3:PutBucketCORS*
- *s3:PutBucketPolicy*
- *s3:PutBucketRequestPayment*

- *s3:PutBucketVersioning*
- *s3:PutBucketWebsite*
- *s3:PutBucketReplication*
- *s3:PutReplicationConfiguration*
- *s3:PutLifecycleConfiguration*
- *s3:PutObjectAcl*
- *s3:PutObject*
- *s3:PutObjectVersionAcl*

注意

Ceph 对象网关不支持在用户、组或角色上设置策略。

Ceph 对象网关使用 RGW 租户标识符来代替 Amazon twelve-digit 帐户 ID。希望使用 Amazon Web Service(AWS)S3 和 Ceph 对象网关 S3 之间的策略，必须在创建用户时将 Amazon 帐户 ID 用作租户 ID。

使用 AWS S3 时，所有租户共享一个命名空间。与之相反，Ceph 对象网关为每个租户提供自己的 bucket 命名空间。目前，Ceph 对象网关客户端试图访问属于另一个租户 MUST 地址的 bucket，作为 S3 请求中的 tenant:bucket。

在 AWS 中，存储桶策略可以授予其他帐户的访问权限，然后该帐户所有者可以向具有用户权限的单独用户授予访问权限。由于 Ceph 对象网关尚不支持用户、角色和组权限，因此帐户所有者需要直接向个别用户授予访问权限。

重要

授予 bucket 的完整帐户访问权限，授予该帐户中所有用户的访问权限。

bucket 策略 不支持 字符串插值。

Ceph 对象网关支持以下条件键：

- aws:CurrentTime
- aws:EpochTime
- aws:PrincipalType
- aws:Referer
- aws:SecureTransport
- aws:SourceIp
- aws:UserAgent
- aws:username

Ceph 对象网关仅支持以下条件键进行 ListBucket 操作：

- s3:prefix
- s3:delimiter

- *s3:max-keys*

对 Swift 的影响

Ceph 对象网关提供在 Swift API 下设置 bucket 策略的功能。但是，使用 S3 API 管理 Swift 和 S3 操作设置的存储桶策略。

Ceph 对象网关将 Swift 凭据与策略中指定的主体匹配。

3.3.31. S3 获取存储桶上的请求支付配置

使用 requestPayment 子资源返回存储桶的请求支付配置。用户需要是存储桶所有者，或被授予存储桶的 READ_ACP 权限。

将 requestPayment 子资源添加到存储桶请求，如下所示。

语法

```
GET /BUCKET?requestPayment HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

3.3.32. S3 在存储桶上设置请求支付配置

使用 requestPayment 子资源来设置存储桶的请求支付配置。默认情况下，存储桶所有者需要从存储桶下载。此配置参数可让 bucket 所有者指定请求者将被请求及从存储桶下载的数据。

将 requestPayment 子资源添加到存储桶请求，如下所示。

语法

```
PUT /BUCKET?requestPayment HTTP/1.1
Host: cname.domain.com
```

请求实体

Payer

描述

指定下载和请求费用的支付费用。

Type

Enum

RequestPaymentConfiguration

描述

用于 Payer 的容器.

Type

Container

3.3.33. 多租户存储桶操作

当客户端应用访问 bucket 时，它始终与特定用户的凭据一同运行。在 Red Hat Ceph Storage 集群中，每个用户都属于一个租户。因此，如果没有明确指定租户，每个 bucket 操作在其上下文中都有一个隐式租户。因此，多租户与之前的版本完全向后兼容，只要引用的存储桶和引用用户属于同一租户。

已根据所使用的协议和身份验证系统，使用扩展来指定明确的租户会有所不同。

在以下示例中，冒号分隔租户和 bucket。因此，一个示例 URL 是：

```
https://rgw.domain.com/tenant:bucket
```

相反，一个简单的 Python 示例将租户和存储桶方法本身分开：

示例

```
from boto.s3.connection import S3Connection, OrdinaryCallingFormat
 c = S3Connection(
  aws_access_key_id="TESTER",
  aws_secret_access_key="test123",
  host="rgw.domain.com",
  calling_format = OrdinaryCallingFormat()
 )
 bucket = c.get_bucket("tenant:bucket")
```

注意

无法利用多租户使用 S3-style 子域，因为主机名不能包含冒号或任何已在存储桶名称中无效的其他分隔符。使用句点会创建模糊的语法。因此，bucket-in-URL-path 格式必须与多租户一起使用。

其它资源

- *如需了解更多详细信息，请参阅 Red Hat Ceph Storage Object Gateway Guide 中的 Multi Tenancy 部分。*

3.3.34. S3 块公共访问

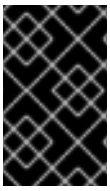
您可以使用 S3 Block Public Access 功能来设置存储桶和用户，以帮助您管理 Red Hat Ceph Storage 对象存储 S3 资源的公共访问权限。

使用此功能、存储桶策略、访问点策略和对象权限可以被覆盖，以允许公共访问。默认情况下，新的 bucket、接入点和对象不允许公共访问。

Ceph 对象网关中的 S3 API 支持 AWS 公共访问设置的子集：

- *BlockPublicPolicy ：定义允许用户管理访问点和存储桶策略的设置。此设置不允许用户公开共享存储桶或其包含的对象。通过启用此设置，现有的访问点和存储桶策略不会受到影响。将这个选项设置为 TRUE 会导致 S3:*

- 拒绝 PUT Bucket 策略的调用。
- 拒绝对所有 bucket 的相同访问点点的 PUT 访问点的调用。



重要

在用户 级别应用此设置，以便用户无法更改特定存储桶的块公共访问设置。

注意

TRUE 设置仅在指定的策略允许公共访问时才有效。

- RestrictPublicBuckets ：定义设置来限制对存储桶的访问或使用公共策略的接入点。限制适用于存储桶所有者帐户和访问点所有者帐户内的 AWS 服务主体和授权用户。这会阻止跨帐户访问访问访问点或 bucket，但指定的情况除外，同时仍然允许帐户中的用户管理访问点或存储桶。启用此设置不会影响现有的访问点或存储桶策略。它仅定义 Amazon S3 会阻止公共和跨帐户访问，以及从任何公共访问点或 bucket 策略派生的跨帐户访问，包括特定帐户的非公共委托。

注意

Red Hat Ceph Storage 目前不支持访问控制列表(ACL)。

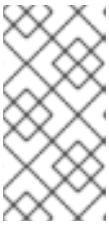
除非另有定义，否则假定存储桶策略为 public。要阻止公共访问，存储桶策略必须只授予以下一个或多个值的访问权限：

注意

固定值不包含通配符(*)或 AWS Identity and Access Management Policy Variable。

- AWS 主体、用户、角色或服务主体

- *使用 aws:SourceIp的无类别域间路由(CIDRs)集*
- *aws:SourceArn*
- *aws:SourceVpc*
- *aws:SourceVpce*
- *aws:SourceOwner*
- *aws:SourceAccount*
- *s3:x-amz-server-side-encryption-aws-kms-key-id*
- *aws:userid，在模式 AROLEID:*之外*
- *s3:DataAccessPointArn*



注意

在存储桶策略中使用时，这个值可以包含访问点名称的通配符，而无需渲染策略 public，只要帐户 ID 已修复。

- *s3:DataAccessPointPointAccount*

以下示例策略被视为 public。

示例

```
{
  "Principal": "*",
  "Resource": "*",
  "Action": "s3:PutObject",
  "Effect": "Allow",
  "Condition": { "StringLike": {"aws:SourceVpc": "vpc-*"}}
 }
```

要使策略非公共，请包含具有固定值的任何条件键。

示例

```
{
  "Principal": "*",
  "Resource": "*",
  "Action": "s3:PutObject",
  "Effect": "Allow",
  "Condition": {"StringEquals": {"aws:SourceVpc": "vpc-91237329"}}
 }
```

其它资源

- 有关获取 PublicAccessBlock 的详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 S3 GET 'PublicAccessBlock' 部分。
- 有关创建或修改 PublicAccessBlock 的详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 S3 PUT 'PublicAccessBlock' 部分。
- 有关删除 PublicAccessBlock 的详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 S3 Delete 'PublicAccessBlock' 部分。
- 如需了解有关存储桶策略的详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 S3 bucket policies 部分。
-

请参阅 Amazon Simple Storage Service (S3)文档中的 Blocking public access to your Amazon S3 storage 部分。

3.3.35. S3 GET PublicAccessBlock

要获得配置的 S3 Block Public Access 功能，请使用 GET 并指定目标 AWS 帐户。

语法

```
GET /v20180820/configuration/publicAccessBlock HTTP/1.1
Host: cname.domain.com
x-amz-account-id: _ACCOUNTID_
```

请求 Headers

有关 常见请求标头的更多信息，请参见附录 B 中的 S3 通用请求标头。

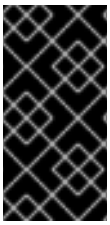
响应

响应是 HTTP 200 响应，以 XML 格式返回。

3.3.36. S3 PUT PublicAccessBlock

使用它来为 S3 存储桶创建或修改 PublicAccessBlock 配置。

要使用此操作，您必须有 s3:PutBucketPublicAccessBlock 权限。



重要

如果 PublicAccessBlock 配置在存储桶和帐户之间不同，Amazon S3 将使用存储桶级别和帐户级别设置的最严格的组合。

语法

```
PUT /?publicAccessBlock HTTP/1.1
Host: Bucket.s3.amazonaws.com
Content-MD5: ContentMD5
x-amz-sdk-checksum-algorithm: ChecksumAlgorithm
x-amz-expected-bucket-owner: ExpectedBucketOwner
<?xml version="1.0" encoding="UTF-8"?>
<PublicAccessBlockConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <BlockPublicAcls>boolean</BlockPublicAcls>
  <IgnorePublicAcls>boolean</IgnorePublicAcls>
  <BlockPublicPolicy>boolean</BlockPublicPolicy>
  <RestrictPublicBuckets>boolean</RestrictPublicBuckets>
</PublicAccessBlockConfiguration>
```

请求 Headers

有关 常见请求标头的更多信息，请参见附录 B 中的 S3 通用请求标头。

响应

响应是一个 HTTP 200 响应，它会返回一个空的 HTTP 正文。

3.3.37. S3 删除 PublicAccessBlock

使用它来删除 S3 存储桶的 PublicAccessBlock 配置。

语法

```
DELETE /v20180820/configuration/publicAccessBlock HTTP/1.1
Host: s3-control.amazonaws.com
x-amz-account-id: AccountId
```

请求 Headers

有关 常见请求标头的更多信息，请参见附录 B 中的 S3 通用请求标头。

响应

响应是一个 HTTP 200 响应，它会返回一个空的 HTTP 正文。

3.4. S3 对象操作

作为开发者，您可以通过 Ceph 对象网关通过 Amazon S3 应用编程接口(API)执行对象操作。

下表列出了对象的 Amazon S3 功能操作，以及功能的支持状态。

表 3.3. 对象操作

功能	Status
获取对象	支持
Get Object Information	支持
Put Object Lock	支持
Get Object Lock	支持
Put Object Legal Hold	支持
Get Object Legal Hold	支持
Put Object Retention	支持
Get Object Retention	支持
Put Object Tagging	支持
Get Object Tagging	支持
Delete Object Tagging	支持
放置对象	支持
删除对象	支持
Delete Multiple Objects	支持
获取对象 ACL	支持

功能	Status
放置对象 ACL	支持
复制对象	支持
后对象	支持
选项对象	支持
启动多部分上传	支持
Add a Part to a Multipart Upload	支持
List Parts of a Multipart Upload	支持
assemble Multipart Upload	支持
Copy Multipart Upload	支持
Abort Multipart Upload	支持
Multi-Tenancy	支持

先决条件

- *一个正在运行的 Red Hat Ceph Storage 集群。*
- *RESTful 客户端。*

3.4.1. S3 从存储桶获取对象

从存储桶检索对象：

语法

```
GET /BUCKET/OBJECT HTTP/1.1
```

添加 versionId 子资源以检索对象的特定版本：

语法

```
GET /BUCKET/OBJECT?versionId=VERSION_ID HTTP/1.1
```

请求 Headers

PartNumber

描述

被读取的对象的一部分号。这为指定部分启用范围 GET 请求。使用此请求可用于只下载对象的一部分。

有效值

1 到 10,000 之间的正整数。

必需

否

range

描述

要检索的对象范围。

注意

不支持每个 GET 请求的多个范围。

有效值

Range:bytes=beginbyte-endbyte

必需

否

if-modified-since

描述

仅当自时间戳起修改时才会获得。

有效值

Timestamp

必需

否

if-unmodified-since

描述

只有自时间戳起没有修改时才会获得。

有效值

Timestamp

必需

否

if-match

描述

仅在对象 ETag 匹配 ETag 时才会获得。

有效值

实体标签

必需

否

if-none-match

描述

仅在对象 ETag 不匹配 ETag 时才会获得。

有效值

实体标签

必需

否

带有请求标头的 Sytnax

```
GET /BUCKET/OBJECT?partNumber=PARTNUMBER&versionId=VersionId HTTP/1.1
Host: Bucket.s3.amazonaws.com
If-Match: IfMatch
If-Modified-Since: IfModifiedSince
If-None-Match: IfNoneMatch
If-Unmodified-Since: IfUnmodifiedSince
Range: Range
```

响应标头

Content-Range

描述

只有在请求中指定范围标头字段时，才会返回数据范围

x-amz-version-id

描述

返回版本 ID 或 null。

3.4.2. S3 获取对象信息

返回有关对象的信息。此请求将返回与 Get Object 请求相同的标头信息，但将仅包含元数据，而不包括对象数据有效负载。

检索对象的当前版本：

语法

```
HEAD /BUCKET/OBJECT HTTP/1.1
```

添加 versionId 子资源以检索特定版本的信息：

语法

```
HEAD /BUCKET/OBJECT?versionId=VERSION_ID HTTP/1.1
```

请求 Headers

range

描述

要检索的对象范围。

有效值

Range:bytes=beginbyte-endbyte

必需

否

if-modified-since

描述

仅当自时间戳起修改时才会获得。

有效值

Timestamp

必需

否

if-match

描述

仅在对象 ETag 匹配 ETag 时才会获得。

有效值

实体标签

必需

否

if-none-match

描述

仅在对象 ETag 匹配 ETag 时才会获得。

有效值

实体标签

必需

否

响应标头

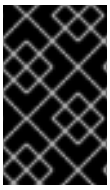
x-amz-version-id

描述

返回版本 ID 或 null。

3.4.3. S3 put 对象锁定

put 对象锁定 API 将锁定配置放在所选存储桶中。使用对象锁定，您可以使用 Write-Once-Read-Many (WORM)模型存储对象。对象锁定可确保，在一个固定的时间端内或无限期内，对象不会被删除或被覆盖。对象锁定配置中指定的规则默认应用于放置在所选存储桶中的每个新对象。



重要

除非创建存储桶时启用对象锁定，否则操作会失败。

语法

```
PUT /BUCKET?object-lock HTTP/1.1
```

示例

```
PUT /testbucket?object-lock HTTP/1.1
```

请求实体

ObjectLockConfiguration

描述

用于请求的容器。

Type

Container

必需

是

ObjectLockEnabled

描述

指明此存储桶是否启用了对象锁定配置。

Type

字符串

必需

是

规则

描述

指定存储桶的对象锁定规则。

Type

Container

必需

否

DefaultRetention

描述

应用到放置在指定存储桶中的新对象的默认保留周期。

Type

Container

必需

否

模式

描述

默认对象锁定保留模式。有效值：GOVERNANCE/COMPLIANCE.

Type

Container

必需

是

Days

描述

为默认保留周期指定的天数。

Type

整数

必需

否

Years

描述

在默认保留周期内指定的年数。

Type

整数

必需

否

HTTP 响应

400

状态代码

MalformedXML

描述

XML 不正确。

409

状态代码

InvalidBucketState

描述

没有启用存储桶对象锁定。

其它资源

- *有关此 API 调用的更多信息，请参阅 S3 API。*

3.4.4. S3 get 对象锁定

get 对象锁定 API 检索存储桶的锁定配置。

语法

```
GET /BUCKET?object-lock HTTP/1.1
```

示例

```
GET /testbucket?object-lock HTTP/1.1
```

响应实体

ObjectLockConfiguration

描述

用于请求的容器。

Type

Container

必需

是

ObjectLockEnabled

描述

指明此存储桶是否启用了对象锁定配置。

Type

字符串

必需

是

规则

描述

指定存储桶的对象锁定规则就位。

Type

Container

必需

否

DefaultRetention

描述

应用到放置在指定存储桶中的新对象的默认保留周期。

Type

Container

必需

否

模式

描述

默认对象锁定保留模式。有效值：GOVERNANCE/COMPLIANCE.

Type

Container

必需

是

Days

描述

为默认保留周期指定的天数。

Type

整数

必需

否

Years

描述

在默认保留周期内指定的年数。

Type

整数

必需

否

其它资源

- 有关此 API 调用的更多信息，请参阅 S3 API。

3.4.5. S3 put 对象法律持有

放置对象法律持有 API 将法律持有的配置应用到所选对象。在法律持有的情况下，您无法覆盖或删除对象版本。法律持有没有关联的保留周期，除非明确删除为止。

语法

```
PUT /BUCKET/OBJECT?legal-hold&versionId= HTTP/1.1
```

示例

```
PUT /testbucket/testobject?legal-hold&versionId= HTTP/1.1
```

versionId 子资源检索对象的特定版本。

请求实体

LegalHold

描述

用于请求的容器。

Type

Container

必需

是

Status

描述

指明指定对象是否有法律持有。有效值： ON/OFF

Type

字符串

必需

是

其它资源

- *有关此 API 调用的更多信息，请参阅 S3 API。*

3.4.6. S3 get 对象法律持有

get 对象法律持有 API 检索对象当前法律持有状态。

语法

```
GET /BUCKET/OBJECT?legal-hold&versionId= HTTP/1.1
```

示例

```
GET /testbucket/testobject?legal-hold&versionId= HTTP/1.1
```

versionId 子资源检索对象的特定版本。

响应实体

LegalHold

描述

用于请求的容器。

Type

Container

必需

是

Status

描述

指明指定对象是否有法律持有。有效值： ON/OFF

Type

字符串

必需

是

其它资源

- *有关此 API 调用的更多信息，请参阅 S3 API。*

3.4.7. S3 放置对象保留

放置对象保留 API 将对象保留配置放在对象中。保留周期为固定时间保护对象版本。有两种模式：GOVERNANCE 和 COMPLIANCE。这两个保留模式对对象应用不同的保护级别。

注意

在此期间，您的对象是 **Write-Once-Read-Many-protected (WORM-protected)**，且无法覆盖或删除。

语法

```
PUT /BUCKET/OBJECT?retention&versionId= HTTP/1.1
```

示例

```
PUT /testbucket/testobject?retention&versionId= HTTP/1.1
```

versionId 子资源检索对象的特定版本。

请求实体

保留

描述

用于请求的容器。

Type

Container

必需

是

模式

描述

指定对象的保留模式。有效值：GOVERNANCE、COMPLIANCE.

Type

字符串

必需

是

RetainUntilDate

描述

保留日期。

格式

2020-01-05T00:00:00.000Z

Type

Timestamp

必需

是

其它资源

- *有关此 API 调用的更多信息，请参阅 S3 API。*

3.4.8. S3 get 对象保留

get 对象保留 API 会检索对象上的对象保留配置。

语法

```
GET /BUCKET/OBJECT?retention&versionId= HTTP/1.1
```

示例

```
GET /testbucket/testobject?retention&versionId= HTTP/1.1
```

***versionId** 子资源检索对象的特定版本。*

*响应实**体***

保留

描述

用于请求的容器。

Type

Container

必需

是

模式

描述

指定对象的保留模式。有效值： GOVERNANCE/COMPLIANCE

Type

字符串

必需

是

RetainUntilDate

描述

保留日期。格式：2020-01-05T00:00:00.000Z

Type

Timestamp

必需

是

其它资源

- *有关此 API 调用的更多信息，请参阅 S3 API。*

3.4.9. S3 put 对象标记

put 对象标记 API 将标签与对象相关联。标签是键值对。若要放置任何其他版本的标签，请使用 versionId 查询参数。您必须具有执行 s3:PutObjectTagging 操作的权限。默认情况下，bucket 所有者具有这个权限，并可向其他用户授予此权限。

语法

```
PUT /BUCKET/OBJECT?tagging&versionId= HTTP/1.1
```

示例

```
PUT /testbucket/testobject?tagging&versionId= HTTP/1.1
```

请求实体

Tagging

描述

用于请求的容器。

Type

Container

必需

是

TagSet

描述

一组标签的集合。

Type

字符串

必需

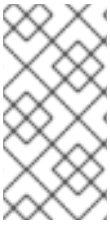
是

其它资源

- *有关此 API 调用的更多信息，请参阅 S3 API。*

3.4.10. S3 get 对象标记

get 对象标记 API 返回对象标签。默认情况下，GET 操作返回对象当前版本的信息。



注意

对于版本化的存储桶，您可以在存储桶中拥有多个版本的对象。要检索任何其他版本的标签，请在请求中添加 versionId 查询参数。

语法

```
GET /BUCKET/OBJECT?tagging&versionId= HTTP/1.1
```

示例

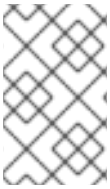
```
GET /testbucket/testobject?tagging&versionId= HTTP/1.1
```

其它资源

- *有关此 API 调用的更多信息，请参阅 S3 API。*

3.4.11. S3 删除对象标记

delete 对象标记 API 从指定对象中删除整个标签集。您必须具有执行 s3:DeleteObjectTagging 操作的权限才能使用此操作。



注意

要删除特定对象版本的标签，请在请求中添加 versionId 查询参数。

语法

```
DELETE /BUCKET/OBJECT?tagging&versionId= HTTP/1.1
```

示例

```
DELETE /testbucket/testobject?tagging&versionId= HTTP/1.1
```

其它资源

- *有关此 API 调用的更多信息，请参阅 S3 API。*

3.4.12. S3 将对象添加到存储桶

添加对象到存储桶。您必须在存储桶上具有写入权限才能执行此操作。

语法

```
PUT /BUCKET/OBJECT HTTP/1.1
```

请求 Headers

content-md5

描述

消息的 base64 编码 MD-5 哈希。

有效值

字符串无默认值或限制。

必需

否

content-type

描述

标准 MIME 类型。

有效值

任何 MIME 类型。默认：binary/octet-stream。

必需

否

x-amz-meta-<...>*

描述

用户元数据。与对象存储。

有效值

字符串最多 8kb。无默认值。

必需

否

x-amz-acl

描述

可以拒绝 ACL。

有效值

私有,public-read,public-read-write,authenticated-read

必需

否

响应标头

x-amz-version-id

描述

返回版本 ID 或 null。

3.4.13. S3 删除对象

删除对象。要求在包含存储桶上设置 WRITE 权限。

删除对象。如果对象版本控制为 on，它会创建一个标记。

语法

```
DELETE /BUCKET/OBJECT HTTP/1.1
```

要在对对象进行版本控制时删除对象，您必须指定 versionId 子资源以及要删除的对象的版本。

```
DELETE /BUCKET/OBJECT?versionId=VERSION_ID HTTP/1.1
```

3.4.14. S3 删除多个对象

此 API 调用从存储桶中删除多个对象。

语法

```
POST /BUCKET/OBJECT?delete HTTP/1.1
```

3.4.15. S3 获取对象的访问控制列表(ACL)

返回对象的当前版本的 ACL：

语法

```
GET /BUCKET/OBJECT?acl HTTP/1.1
```

添加 versionId 子资源以检索特定版本的 ACL：

语法

```
GET /BUCKET/OBJECT?versionId=VERSION_ID&acl HTTP/1.1
```

响应标头

x-amz-version-id

描述

返回版本 ID 或 null。

响应实体

AccessControlPolicy

描述

用于响应的容器。

Type

Container

AccessControlList

描述

ACL 信息的容器。

Type

Container

所有者

描述

bucket 所有者的 ID 和 DisplayName 的容器。

Type

Container

ID

描述

bucket 所有者的 ID。

Type

字符串

DisplayName

描述

bucket 所有者的显示名称。

Type

字符串

Grant

描述

适用于 Grantee 和 Permission 的容器。

Type

Container

Grantee

描述

允许权限的用户的 DisplayName 和 ID 的容器。

Type

Container

权限

描述

提供给 Grantee 存储桶的权限。

Type

字符串

3.4.16. S3 设置对象的访问控制列表(ACL)

为对象的当前版本设置对象 ACL。

语法

```
PUT /BUCKET/OBJECT?acl
```

请求实体

AccessControlPolicy

描述

用于响应的容器。

Type

Container

AccessControlList

描述

ACL 信息的容器。

Type

Container

所有者

描述

bucket 所有者的 ID 和 DisplayName 的容器。

Type

Container

ID

描述

bucket 所有者的 ID。

Type

字符串

DisplayName

描述

bucket 所有者的显示名称。

Type

字符串

Grant

描述

适用于 Grantee 和 Permission 的容器。

Type

Container

Grantee

描述

允许权限的用户的 DisplayName 和 ID 的容器。

Type

Container

权限

描述

提供给 Grantee 存储桶的权限。

Type

字符串

3.4.17. S3 复制一个对象

要复制对象，请使用 PUT 并指定目标存储桶和对象名称。

语法

```
PUT /DEST_BUCKET/DEST_OBJECT HTTP/1.1
x-amz-copy-source: SOURCE_BUCKET/SOURCE_OBJECT
```

请求 Headers

x-amz-copy-source

描述

源 bucket 名称 + 对象名称。

有效值

BUCKET/OBJECT

必需

是

x-amz-acl

描述

可以拒绝 ACL。

有效值

私有,public-read,public-read-write,authenticated-read

必需

否

x-amz-copy-if-modified-since

描述

仅在自时间戳以来修改时才复制。

有效值

Timestamp

必需

否

x-amz-copy-if-unmodified-since

描述

仅在自时间戳起未修改时才复制。

有效值

Timestamp

必需

否

x-amz-copy-if-match

描述

只有在对象 ETag 匹配 ETag 时复制。

有效值

实体标签

必需

否

x-amz-copy-if-none-match

描述

只有在对象 ETag 匹配 ETag 时复制。

有效值

实体标签

必需

否

响应实体

CopyObjectResult

描述

用于响应元素的容器。

Type

Container

LastModified

描述

源对象的最后修改日期。

Type

Date

Etag

描述

新对象的 ETag。

Type

字符串

3.4.18. S3 使用 HTML 表单向存储桶添加对象

使用 HTML 表单将对象添加到存储桶。您必须在存储桶上具有写入权限才能执行此操作。

语法

```
POST /BUCKET/OBJECT HTTP/1.1
```

3.4.19. S3 确定请求选项

用于确定实际请求是否可以通过特定原始卷、HTTP 方法和标头发送的 preflight 请求。

语法

```
OPTIONS /OBJECT HTTP/1.1
```

3.4.20. S3 启动多部分上传

启动多部分上传过程。返回 UploadId，您可以在添加额外部分、列出部分和完成或带出多部分上传时指定。

语法

```
POST /BUCKET/OBJECT?uploads
```

请求 Headers

content-md5

描述

消息的 base64 编码 MD-5 哈希。

有效值

字符串无默认值或限制。

必需

否

content-type

描述

标准 MIME 类型。

有效值

任何 MIME 类型。默认： binary/octet-stream

必需

否

x-amz-meta-<...>

描述

用户元数据。与对象存储。

有效值

字符串最多 8kb。无默认值。

必需

否

x-amz-acl

描述

可以拒绝 ACL。

有效值

私有,public-read,public-read-write,authenticated-read

必需

否

响应实体

InitiatedMultipartUploadsResult

描述

用于结果的容器。

Type

Container

Bucket

描述

将接收对象内容的存储桶。

Type

字符串

键

描述

key 请求参数指定的密钥（若有）。

Type

字符串

UploadId

描述

由 upload-id 请求参数指定的 ID，标识多部分上传（如果有）。

Type

字符串

3.4.21. S3 在多部分上传中添加部分

添加部分到多部分上传。

指定 uploadId 子资源以及上传 ID，将部分添加到多部分上传：

语法

```
PUT /BUCKET/OBJECT?partNumber=&uploadId=UPLOAD_ID HTTP/1.1
```

可能会返回以下 HTTP 响应：

HTTP 响应

404

状态代码

NoSuchUpload

描述

指定 upload-id 与此对象上启动的任何上传都不匹配。

3.4.22. S3 列出多部分上传

指定 uploadId 子资源以及上传 ID 来列出多部分上传：

语法

```
GET /BUCKET/OBJECT?uploadId=UPLOAD_ID HTTP/1.1
```

响应实体

InitiatedMultipartUploadsResult

描述

用于结果的容器。

Type

Container

Bucket

描述

将接收对象内容的存储桶。

Type

字符串

键

描述

key 请求参数指定的密钥（若有）。

Type

字符串

UploadId

描述

由 upload-id 请求参数指定的 ID，标识多部分上传（如果有）。

Type

字符串

initiator

描述

包含开始上传的用户的 ID 和 DisplayName。

Type

Container

ID

描述

启动器的 ID。

Type

字符串

DisplayName

描述

启动器的显示名称。

Type

字符串

所有者

描述

拥有上传对象的用户的 ID 和 DisplayName 的容器。

Type

Container

StorageClass

描述

用于存储生成的对象的方法。STANDARD 或 REDUCED_REDUNDANCY

Type

字符串

PartNumberMarker

描述

如果 IsTruncated 为 true，则后续请求中使用的部分标记。在列表前面。

Type

字符串

NextPartNumberMarker

描述

如果 IsTruncated 为 true，则后续请求中使用的下一部分标记。列表的末尾。

Type

字符串

IsTruncated

描述

如果为 true，则仅返回对象上传内容的子集。

Type

布尔值

Part

描述

Key, Part, InitiatorOwner, StorageClass, 和 Initiated 元素的容器。

Type

Container

PartNumber

描述

Key, Part, InitiatorOwner, StorageClass, 和 Initiated 元素的容器。

Type

整数

ETag

描述

部分的实体标签。

Type

字符串

大小

描述

上传部分的大小。

Type

整数

3.4.23. S3 汇编上传的部分

装配已上传的部分并创建新的对象，从而完成多部分上传。

指定 uploadId 子资源以及上传 ID 以完成多部分上传：

语法

```
POST /BUCKET/OBJECT?uploadId=UPLOAD_ID HTTP/1.1
```

请求实体

CompleteMultipartUpload

描述

由一个或多个部分组成的容器。

Type

Container

必需

是

Part

描述

适用于 PartNumber 和 ETag 的容器。

Type

Container

必需

是

PartNumber

描述

部分的标识符。

Type

整数

必需

是

ETag

描述

部分的实体标签。

Type

字符串

必需

是

响应实体

CompleteMultipartUploadResult

描述

用于响应的容器。

Type

Container

位置

描述

新对象的资源标识符(path)。

Type

URI

bucket

描述

包含新对象的存储桶的名称。

Type

字符串

键

描述

对象的密钥。

Type

字符串

ETag

描述

新对象的实体标签。

Type

字符串

3.4.24. S3 复制多部分上传

通过从现有对象复制数据作为数据源来上传部分。

指定 **uploadId** 子资源以及上传 **ID**，以执行多部分上传副本：

语法

```
PUT /BUCKET/OBJECT?partNumber=PartNumber&uploadId=UPLOAD_ID HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

请求 Headers

x-amz-copy-source

描述

源存储桶名称和对象名称。

有效值

BUCKET/OBJECT

必需

是

x-amz-copy-source-range

描述

从源对象复制的字节数。

有效值

范围： bytes=first-last，其中第一个和最后一个是要复制的零字节偏移。例如， bytes=0-9 表示您想要复制源的前 10 字节。

必需

否

响应实体

CopyPartResult

描述

用于所有响应元素的容器。

Type

Container

ETag

描述

返回新部分的 ETag。

Type

字符串

LastModified

描述

返回部分上次修改的日期。

Type

字符串

其它资源

- 有关此功能的更多信息，请参阅 Amazon S3 网站。

3.4.25. S3 中止多部分上传

中止多部分上传。

指定 uploadId 子资源以及上传 ID 以中止多部分上传：

语法

```
DELETE /BUCKET/OBJECT?uploadId=UPLOAD_ID HTTP/1.1
```

3.4.26. S3 Hadoop 互操作性

对于需要 Hadoop 分布式文件系统(HDFS)访问的数据分析应用，可以使用用于 Hadoop 的 Apache S3A 连接器来访问 Ceph 对象网关。S3A 连接器是一个开源工具，它将 S3 兼容对象存储呈现为 HDFS 文件系统，在数据存储在 Ceph 对象网关中时，对应用程序进行读写语义。

Ceph 对象网关与 Hadoop 2.7.3 附带的 S3A 连接器完全兼容。

其它资源

- 如需了解有关多租户的详细信息，请参阅 Red Hat Ceph Storage 对象网关指南。

3.5. S3 选择操作

作为开发者，您可以运行 S3 选择来加快吞吐量。用户可以在没有介质器的情况下直接运行 S3 选择查询。

有三个 S3 选择工作流 - CSV、Apache Parquet (Parquet)和 JSON，为 CSV、Bquet 和 JSON 对象提供 S3 选择操作：

- CSV 文件以纯文本格式存储表格数据。文件的每一行都是数据记录。
- Parquet 是一个开源、面向列的数据文件格式，旨在高效数据存储和检索。它提供了高效率的数据压缩和编码方案，以提高性能，以批量处理复杂数据。Parquet 启用 S3 select-engine 跳过列和块，从而减少 IOPS （与 CSV 和 JSON 格式持续）。
- JSON 是格式结构。S3 选择引擎使用 JSON 格式输入数据顶部的 SQL 语句，从而可以扫描高度嵌套和复杂的 JSON 格式数据。

例如，带有数 GB 数据的 CSV、Parquet 或 JSON S3 对象允许用户提取单个列，使用以下查询过滤另一个列：

示例

```
select customerid from s3Object where age>30 and age<65;
```

目前，S3 对象必须通过 Ceph 对象网关从 Ceph OSD 检索数据，然后才能过滤和提取数据。当对象较大且查询更为具体时，性能会提高性能。比 CSV 更高效地处理 Parquet 格式。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。

- RESTful 客户端。
- 创建的用户访问权限的 S3 用户。

3.5.1. S3 从对象中选择内容

选择对象内容 API 通过结构化查询语言(SQL)过滤对象的内容。如需清单对象中应驻留的内容的说明，请参阅 AWS 系统管理器用户指南中的清单收集的元数据部分。清单内容会影响应针对该清单运行的查询类型。可能提供重要信息的 SQL 语句数量较大，但 S3 选择是类 SQL 的实用程序，因此不支持某些运算符，如 group-by 并加入。

仅对于 CSV，您必须将数据序列化格式指定为以逗号分隔的对象值，以检索指定的内容。Parquet 没有分隔符，因为它采用二进制格式。Amazon Web Services (AWS)命令行界面(CLI)选择对象内容使用 CSV 或 Parquet 格式将对象数据解析到记录中，仅返回查询中指定的记录。

您必须为响应指定数据序列化格式。这个操作必须具有 s3:GetObject 权限。

注意

- InputSerialization 元素描述了正在查询的对象中的数据格式。对象可以是 CSV 或 Parquet 格式。
- OutputSerialization 元素是 AWS-CLI 用户客户端的一部分，并描述了如何格式化输出数据。Ceph 为 AWS-CLI 实施了服务器客户端，因此，根据当前仅 CSV 的 OutputSerialization 提供相同的输出。
- InputSerialization 的格式不需要与 OutputSerialization 的格式匹配。例如，您可以在 InputSerialization 中指定 Parquet，在 OutputSerialization 中指定 CSV。

语法

```
POST /BUCKET/KEY?select&select-type=2 HTTP/1.1\r\n
```

示例

```
POST /testbucket/sample1csv?select&select-type=2 HTTP/1.1\r\n
POST /testbucket/sample1parquet?select&select-type=2 HTTP/1.1\r\n
```

请求实体

Bucket

描述

要从中选择对象内容的存储桶。

Type

字符串

必需

是

键

描述

对象键。

长度限制

最小长度为 1.

Type

字符串

必需

是

SelectObjectContentRequest

描述

选择对象内容请求参数的根级别标签。

Type

字符串

必需

是

表达式

描述

用于查询对象的表达式。

Type

字符串

必需

是

ExpressionType

描述

示例 SQL 提供的表达式的类型。

Type

字符串

有效值

SQL

必需

是

InputSerialization

描述

描述正在查询的对象中数据的格式。

Type

字符串

必需

是

OutputSerialization

描述

以逗号分隔符和新行返回的数据格式。

Type

字符串

必需

是

响应实体

如果操作成功，服务会返回 HTTP 200 响应。服务以 XML 格式返回数据：

payload

描述

有效负载参数的根级别标签。

Type

字符串

必需

是

Records

描述

记录事件。

Type

base64 编码的二进制数据对象

必需

否

Stats

描述

stats 事件。

Type

Long

必需

否

Ceph 对象网关支持以下响应：

示例

```
{:event-type,records} {:content-type,application/octet-stream} {:message-type,event}
```

语法（对于 CSV）

```
aws --endpoint-URL http://localhost:80 s3api select-object-content
 --bucket BUCKET_NAME
 --expression-type 'SQL'
 --input-serialization
 '{"CSV": {"FieldDelimiter": "," , "QuoteCharacter": "\"" , "RecordDelimiter" : "\n" ,
"QuoteEscapeCharacter" : "\\" , "FileHeaderInfo": "USE" }, "CompressionType": "NONE"}'
```

```
 --output-serialization '{"CSV": {}}'
 --key OBJECT_NAME.csv
 --expression "select count(0) from s3object where int(_1)<10;" output.csv
```

*示例（对于 **CSV**）*

```
aws --endpoint-url http://localhost:80 s3api select-object-content
 --bucket testbucket
 --expression-type 'SQL'
 --input-serialization
 '{"CSV": {"FieldDelimiter": "," , "QuoteCharacter": "\"" , "RecordDelimiter" : "\n" ,
"QuoteEscapeCharacter" : "\\" , "FileHeaderInfo": "USE" }, "CompressionType": "NONE"}'
 --output-serialization '{"CSV": {}}'
 --key testobject.csv
 --expression "select count(0) from s3object where int(_1)<10;" output.csv
```

*语法（针对 **Parquet**）*

```
aws --endpoint-url http://localhost:80 s3api select-object-content
 --bucket BUCKET_NAME
 --expression-type 'SQL'
 --input-serialization
 '{"Parquet": {}, {"CompressionType": "NONE"}'
 --output-serialization '{"CSV": {}}'
 --key OBJECT_NAME.parquet
 --expression "select count(0) from s3object where int(_1)<10;" output.csv
```

*示例（对于 **Parquet**）*

```
aws --endpoint-url http://localhost:80 s3api select-object-content
 --bucket testbucket
 --expression-type 'SQL'
 --input-serialization
 '{"Parquet": {}, {"CompressionType": "NONE"}'
```

```
 --output-serialization '{"CSV": {}}'
 --key testobject.parquet
 --expression "select count(0) from s3object where int(_1)<10;" output.csv
```

语法（用于 JSON）

```
aws --endpoint-URL http://localhost:80 s3api select-object-content
 --bucket BUCKET_NAME
 --expression-type 'SQL'
 --input-serialization
 '{"JSON": {"CompressionType": "NONE"}'
 --output-serialization '{"CSV": {}}}'
 --key OBJECT_NAME.json
 --expression "select count(0) from s3object where int(_1)<10;" output.csv
```

示例（用于 JSON）

```
aws --endpoint-url http://localhost:80 s3api select-object-content
 --bucket testbucket
 --expression-type 'SQL'
 --input-serialization
 '{"JSON": {"CompressionType": "NONE"}'
 --output-serialization '{"CSV": {}}}'
 --key testobject.json
 --expression "select count(0) from s3object where int(_1)<10;" output.csv
```

示例（用于 BOTO3）

```
import pprint
import boto3
from botocore.exceptions import ClientError

def
run_s3select(bucket,key,query,column_delim=",",row_delim="\n",quot_char='"',esc_char='\\',csv_header
_info="NONE"):
```

```
    s3 = boto3.client('s3',
        endpoint_url=endpoint,
        aws_access_key_id=access_key,
        region_name=region_name,
        aws_secret_access_key=secret_key)

    result = ""
    try:
        r = s3.select_object_content(
        Bucket=bucket,
        Key=key,
        ExpressionType='SQL',
        InputSerialization = {"CSV": {"RecordDelimiter" : row_delim, "FieldDelimiter" :
column_delim,"QuoteEscapeCharacter": esc_char, "QuoteCharacter": quot_char, "FileHeaderInfo":
csv_header_info}, "CompressionType": "NONE"},
        OutputSerialization = {"CSV": {}},
        Expression=query,
        RequestProgress = {"Enabled": progress})

    except ClientError as c:
        result += str(c)
        return result

    for event in r['Payload']:
            if 'Records' in event:
                result = ""
                records = event['Records']['Payload'].decode('utf-8')
                result += records
            if 'Progress' in event:
                print("progress")
                pprint.pprint(event['Progress'],width=1)
            if 'Stats' in event:
                print("Stats")
                pprint.pprint(event['Stats'],width=1)
            if 'End' in event:
                print("End")
                pprint.pprint(event['End'],width=1)

    return result




 run_s3select(
 "my_bucket",
 "my_csv_object",
 "select int(_1) as a1, int(_2) as a2 , (a1+a2) as a3 from s3object where a3>100 and a3<300;")
```

支持的功能

目前只支持 AWS s3 选择命令的一部分：

功能	详情	描述	示例
算术运算符	^ * % / + - ( )		select (int(_1)+int(_2))*int(_9) from s3object;
算术运算符	% modulo		select count cycles from s3object where cast (_1 as int)%2 = 0;
算术运算符	^ power-of		从 s3object 选择 cast（2^10，为 int）；
比较运算符	> < >= ⇐ == !=		select _1,_2 from s3object where (int(_1)+int(_3))>int(_5);
逻辑运算符	AND OR NOT		select count(*) from s3object where not (int(1)>123 and int(_5) <200);
逻辑运算符	is null	为表达式中的 null 返回 true/false	
逻辑运算符和 NULL	is not null	为表达式中的 null 返回 true/false	
逻辑运算符和 NULL	未知状态	查看 null-handle，并观察使用 NULL 的逻辑操作的结果。查询返回 **0**。	**select count(*) from s3object where null and (3>2);**
带有 NULL 的算术运算符	未知状态	查看 null-handle，并观察使用 NULL 的二进制操作的结果。查询返回 **0**。	**select count(*) from s3object where (null+1) and (3>2);**
与 NULL 进行比较	未知状态	回顾空客户端并观察与 NULL 比较操作的结果。查询返回 **0**。	**select count(*) from s3object where (null*1.5) != 3;**
缺少列	未知状态		**select count(*) from s3object where _1 is null;**
projection 列	与 if or then or else 类似		选择问题单 **when (1+1== (2+1)*3) then 'case_1' when 4*3）==（12 then 'case_2' else 'case_else' end, age*2 from s3object;**

功能	详情	描述	示例
projection 列	与 switch/case 类似		选择 **case cast (_1 as int)+ 1 when 2 then "a" when 3 then "b" other "c" end from s3object;**
逻辑运算符		**coalesce** 返回第一个非null 参数	**select coalesce(nullif(5,5),nullif(1,1.0),age+12) from s3object;**
逻辑运算符		如果两个参数都相等，则 **nullif** 返回 null，否则第一个参数为**nullif (1,1)=NULL nullif (null,1)=NULL nullif (2,1)=2**	**select nullif(cast(_1 as int),cast(_2 as int)) from s3object;**
逻辑运算符		**{expression} in (.. {expression} ..)**	**select count cycles from s3object where 'ben' in (trim (_5),substring （_1,char_length (_1)-3,3),last_name）;**
逻辑运算符		**{expression} 和 {expression} 之间的 {expression}**	**select _1 from s3object where cast (_1 as int) between 800 和 900; select count cycles from stdin where substring (_3,char_length (_3),1) between "x" and trim (_1)和 substring (_3,char_length (_3)-1,1)= ":";**
逻辑运算符		**{expression} like {match-pattern}**	**select count () from s3object where first_name like '%de_'; select count () from s3object where _1 like "%a[r-s];**
casting operator			**select cast(123 as int)%2 from s3object;**
casting operator			**select cast(123.456 as float)%2 from s3object;**
casting operator			**select cast ('ABC0-9' as string),cast (substr ('ab12cd',3,2) as int)*4 from s3object;**

功能	详情	描述	示例
casting operator			**select cast (substring ('publish on 2007-01-01',12,10) as timestamp) from s3object;**
非 AWS casting operator			**select int(_1),int(1.2 + 3.4) from s3object;**
非 AWS casting operator			**select float(1.2) from s3object;**
非 AWS casting operator			**select to_timestamp ('1999-10-10T12:23:44Z') from s3object;**
聚合功能	sun		**select sum(int(_1)) from s3object;**
聚合功能	avg		**select avg (cast (_1 as float)+ cast (_2 as int)) from s3object;**
聚合功能	分钟		**select avg(cast(_1 a float) + cast(_2 as int)) from s3object;**
聚合功能	max		**select max(float(_1)),min(int(_5)) from s3object;**
聚合功能	数量		**select count(*) from s3object where (int(1)+int(_3))>int(_5);**
时间戳功能	extract		**select count cycles from s3object where extract (year from to_timestamp (_2))> 1950 and extract (year from to_timestamp (_1))< 1960;**
时间戳功能	dateadd		**select count (0) from s3object where date_diff (year,to_timestamp (_1),date_add （day,366,to_timestamp (_1))）= 1;**

功能	详情	描述	示例
时间戳功能	datediff		**select count (0) from s3object where date_diff (month,to_timestamp (_1),to_timestamp (_2))= 2;**
时间戳功能	utcnow		**select count (0) from s3object where date_diff (hour,utcnow （),date_add (day,1,utcnow （））） = 24**
时间戳功能	to_string		**select to_string (to_timestamp ("2009-09-17T17:56:06.234567Z"), "yyyyMMdd-H:m:s") from s3object;**
字符串函数	子字符串		**select count(0) from s3object where int(substring(_1,1,4))>1950 and int(substring(_1,1,4)) <1960;**
字符串函数	子字符串	来自负数的子字符串被视为第一个有效	**select substring ("123456789" from -4) from s3object;**
字符串函数	子字符串	来自零的子字符串用于越界数字，就像(first,last)一样有效。	**select substring ("123456789" from 0 for 100) from s3object;**
字符串函数	trim		**select trim (' foobar ') from s3object;**
字符串函数	trim		**select trim (trailing from ' foobar ') from s3object;**
字符串函数	trim		**select trim (leading from ' foobar ') from s3object;**
字符串函数	trim		**select trim (both '12' from '1112211foobar22211122') from s3object;**
字符串函数	lower 或 upper		**select lower ('ABcD12#$e') from s3object;**

功能	详情	描述	示例
字符串函数	char_length, character_length		**select count cycles from s3object where char_length (_3)=3;**
复杂的查询			**select sum (cast (_1 as int)),max (cast (_3 as int)), substring ('abcdefghijklm', (2-1)*3+sum （cast (_1 as int))/sum (cast (_1 as int))+1, (count （)+ count (0)）/count (0)）from s3object;**
别名支持			**select int(_1) as a1, int(_2) as a2 , (a1+a2) as a3 from s3object where a3>100 and a3<300;**

其它资源

- *如需了解更多详细信息，请参阅 Amazon 的 S3 Select Object Content API。*

3.5.2. S3 支持的选择功能

S3 选择支持以下功能：.Timestamp

to_timestamp(string)

描述

将字符串转换为时间戳基本类型。在字符串格式中，任何缺少的 'time' 值都会填充零；对于缺少的月份和天值，1 是默认值。'timezone' 格式是 +/-H:mm 或 Z，其中字母 'Z' 表示协调通用时间(UTC)。timezone 的值可以包括 - 12:00 和 +14:00。

支持

目前，它可以将以下字符串格式转换为时间戳：

- ***YYYY-MM-DDTHH:mm:ss.SSSSSS+/-HH:mm***
- ***YYYY-MM-DDTHH:mm:ss.SSSSSSZ***

- *YYYY-MM-DDTHH:mm:ss+/-HH:mm*
- *YYYY-MM-DDTHH:mm:ssZ*
- *YYYY-MM-DDTHH:mm+/-HH:mm*
- *YYYY-MM-DDTHH:mmZ*
- *YYYY-MM-DDT*
- *YYYYT*

to_string (timestamp, format_pattern)

描述

以给定的输入字符串格式返回输入时间戳的字符串。

参数

格式	示例	描述
yy	69	2 年数字.
y	1969	4 年数字。
YYYY	1969	零添加的 4 位数年。
M	1	年月。
MM	01	每年的零添加月数.
MMM	1 月	每年名称的缩写月。
MMMM	1 月	年全月名称。

格式	示例	描述
MMMMM	J	年前一个字母的月。与 **to_timestamp** 函数一起使用无效。
d	2	日期(1-31)。
dd	02	每月零添加一天(01-31)。
a	AM	每天的 AM 或 PM。
h	3	天中的小时(1-12)。
hh	03	零添加小时的一天(01-12)。
H	3	天中的小时(0-23)。
HH	03	每天的零添加小时(00-23)。
m	4	小时的分钟(0-59)。
mm	04	小时的零添加分钟(00-59)。
s	5	第二分钟(0-59)。
ss	05	分钟的零添加秒(00-59)。
S	1	第二部分（精度： 0.1，范围： 0.0-0.9）。
SS	12	第二部分（精度： 0.01，范围： 0.0-0.99）。
SSS	123	第二部分（精度： 0.01，范围： 0.0-0.999）。
SSSS	1234	第二部分（精度： 0.001，范围： 0.0-0.9999）。
SSSSSS	123456	第二部分（最大精度）：1纳秒，范围： 0.0-0.999999）。
n	60000000	second 的 nano。
X	+07 或 Z	如果偏移为 0，则以小时或"Z"表示偏移量。

格式	示例	描述
XX 或 XXXX	+0700 或 Z	如果偏移为 0，则以小时和"Z"表示偏移量。
XXX 或 XXXXX	+07:00 或 Z	如果偏移为 0，则以小时和"Z"表示偏移量。
x	7	以小时为单位进行偏移。
XX 或 xxxx	700	以小时和分钟为单位的偏移。
xxx 或 xxxxx	+07:00	以小时和分钟为单位的偏移。

extract (date-part from timestamp)

描述

根据 date-part 从输入时间戳中提取的整数。

支持

year, month, week, day, hour, minute, second, timezone_hour, timezone_minute.

date_add(date-part ,integer,timestamp)

描述

返回时间戳，根据输入时间戳和日期部分的结果计算。

支持

年，月份、天、小时、分钟、秒。

date_diff(date-part,timestamp,timestamp)

描述

返回整数，根据 date-part 在两个时间戳之间计算的结果。

支持

年，月份、天、小时、分钟、秒。

utcnow()

描述

返回当前时间的时间戳。

聚合

count()

描述

根据与条件匹配的行数返回整数（如果存在）。

sum(expression)

描述

如果出现某个条件，则每行上返回表达式摘要。

avg(expression)

描述

如果出现某个条件，每行中返回一个平均表达式。

max(expression)

描述

如果出现某个条件，则返回与条件匹配的所有表达式的最大结果。

min(expression)

描述

返回与条件匹配的所有表达式的最小结果。

字符串

子字符串（字符串、from、for）

描述

为输入返回字符串从输入字符串中提取的字符串。

Char_length

描述

返回字符串中的多个字符。Character_length 也实现相同的目的。

trim ([[leading | trailing | both remove_chars] from] string)

描述

从目标字符串中修剪前/尾部（或两者）字符。默认值为空白字符。

Upper\lower

描述

将字符转换为大写或小写。

NULL

NULL 值缺失或未知，即 NULL 无法为任何算术操作生成一个值。这同样适用于算术比较，对 NULL 的任何比较都是未知的 NULL。

表 3.4. NULL 用例

A is NULL	result (NULL=UNKNOWN)
不是 A	**NULL**
A 或 False	**NULL**
A or True	**true**
A 或 A	**NULL**
A 和 False	**False**
A 和 True	**NULL**
A 和 A	**NULL**

其它资源

- *如需了解更多详细信息，请参阅 Amazon 的 S3 Select Object Content API。*

3.5.3. S3 别名编程结构

别名编程结构是 s3 选择语言的重要组成部分，因为它可以为包含许多列或复杂查询的对象启用更好的编程。当解析带有别名结构的声明时，它会将别名替换为对右投射列和查询执行的引用，该引用将象任何其他表达式一样评估。别名维护结果缓存，如果别名被多次使用，则不会评估相同的表达式，并返回相同的结果，因为使用了缓存的结果。目前，红帽支持列别名。

示例

```
select int(_1) as a1, int(_2) as a2 , (a1+a2) as a3 from s3object where a3>100 and a3<300;")
```

3.5.4. S3 解析解释

S3 选择引擎具有所有三个文件格式的解析器：CSV、Brquet 和 JSON，后者将命令划分为更多可处理组件，然后附加到定义每个组件的标签中。

3.5.4.1. S3 CSV 解析

带有输入序列化的 CSV 定义使用以下默认值：

- 将 {\n}' 用于 row-delimiter。
- 使用 {"} 括起内容。
- 使用 {\} 转义字符。

csv-header-info 会在 AWS-CLI 中显示 USE 时解析，这是包含该模式的输入对象中的第一行。目前，不支持输出序列化和压缩类型。S3 选择引擎具有 CSV 解析器，它解析 S3-objects：

- 每行以 row-delimiter 结尾。
- field-separator 会分离相邻的列。

- *successive 字段分隔符定义 NULL 列。*
- *quote-character 覆盖 field-separator；即，字段分隔符是引号之间的任何字符。*
- *转义字符禁用除行分隔符之外的任何特殊字符。*

以下是 CSV 解析规则的示例：

表 3.5. CSV 解析

功能	描述	输入(Tokens)
NULL	successive 字段分隔符	**,,1,,2, ==⇒ {null}{null}{1}{null}{2}{null}**
QUOTE	quote 字符覆盖字段分隔符。	**11,22,"a,b,c,d",last ==⇒ {11}{22}{"a,b,c,d"}{last}**
Escape	转义字符覆盖了元字符。	对象所有者 **ID** 和 **DisplayName** 的容器
行分隔符	没有关闭的引号；行分隔符是右行。	**11,22,a="str,44,55,66 ==⇒ {11}{22}{a="str,44,55,66}**
CSV 标头信息	FileHeaderInfo 标签	USE 值表示第一行中的每个令牌都是列名称；IGNORE 值意味着跳过第一行。

其它资源

- *如需了解更多详细信息，请参阅 Amazon 的 S3 Select Object Content API。*

3.5.4.2. S3 Parquet 解析

Apache Parquet 是一个开源列式数据文件格式，旨在高效数据存储和检索。

S3 选择引擎的 Parquet 解析器解析 S3-objects，如下所示：

示例

```
4-byte magic number "PAR1"
<Column 1 Chunk 1 + Column Metadata>
<Column 2 Chunk 1 + Column Metadata>
...
<Column N Chunk 1 + Column Metadata>
<Column 1 Chunk 2 + Column Metadata>
<Column 2 Chunk 2 + Column Metadata>
...
<Column N Chunk 2 + Column Metadata>
...
<Column 1 Chunk M + Column Metadata>
<Column 2 Chunk M + Column Metadata>
...
<Column N Chunk M + Column Metadata>
File Metadata
4-byte length in bytes of file metadata
4-byte magic number "PAR1"
```

- 在上例中，此表中有 N 列，被分成 M 行组。文件元数据包含所有列元数据启动位置。
- 元数据在数据后写入，以允许进行一次传递写入。
- 所有列块都可以在文件元数据中找到，之后应按顺序读取。
- 格式被明确设计为将元数据与数据分开。这允许将列分成多个文件，并有一个元数据文件引用多个 parquet 文件。

3.5.4.3. S3 JSON 解析

JSON 文档启用在没有限制的情况下的对象或数组中嵌套值。在 S3 选择引擎的 JSON 文档中查询特定值时，该值的位置通过 SELECT 语句中的路径指定。

JSON 文档的通用结构没有 CSV 和 Parquet 等行和列结构。相反，SQL 语句本身是查询 JSON 文档时定义行和列的 SQL 语句本身。

S3 选择引擎的 JSON 解析器解析 S3-objects，如下所示：

- *SELECT 语句中的 FROM 子句定义行边界。*
- *JSON 文档中的一行与如何为 CSV 对象定义行的分隔符类似，以及如何使用行组定义 Parquet 对象的行*
- *考虑以下示例：*

示例

```
{
    "firstName": "Joe",
    "lastName": "Jackson",
    "gender": "male",
    "age": "twenty"
},

{
    "firstName": "Joe_2",
    "lastName": "Jackson_2",
    "gender": "male",
    "age": 21
},

"phoneNumbers":
[
    { "type": "home1", "number": "734928_1","addr": 11 },
    { "type": "home2", "number": "734928_2","addr": 22 }
],

"key_after_array": "XXX",

"description" :
{
    "main_desc" : "value_1",
    "second_desc" : "value_2"
}

# the from-clause define a single row.
# _1 points to root object level.
# _1.age appears twice in Documnet-row, the last value is used for the operation.
query = "select
_1.firstname,_1.key_after_array,_1.age+4,_1.description.main_desc,_1.description.second_des
```

```
c from s3object[*].aa.bb.cc;";

expected_result = Joe_2,XXX,25,value_1,value_2
```

- 语句指示读取器搜索路径 aa.bb.cc，并根据此路径的发生定义行边界。
- 当读取器遇到路径时，行开始，当读取器退出路径的最顶层部分时结束，本例中为对象 cc。

3.5.5. 将 Ceph 对象网关与 Trino 集成

将 Ceph 对象网关与 Trino 集成，这是一个重要的实用程序，允许用户在 S3 对象上更快地运行 SQL 查询 9x。

以下是使用 Trino 的一些优点：

- Trino 是一个完整的 SQL 引擎。
- 推送 S3 选择 Trino 引擎中的请求标识在服务器端运行具有成本效益的 SQL 语句的一部分。
- 使用 Ceph/S3select 的优化规则来提高性能。
- 利用 Red Hat Ceph Storage 可扩展性，将原始对象划分为多个等部分，执行 S3 选择请求并合并请求。

重要

如果 s3select 语法在查询 trino 时无法正常工作，请使用 SQL 语法。

先决条件

- 正在运行的 Red Hat Ceph Storage 集群安装有 Ceph 对象网关。
- 安装了 Docker 或 Podman。
- bucket 已创建。
- 对象已上传。

流程

1. 部署 Trino 和 hive。

示例

```
[cephuser@host01 ~]$ git clone https://github.com/ceph/s3select.git
[cephuser@host01 ~]$ cd s3select
```

2. 使用 S3 端点、访问密钥和 secret 密钥修改 hms_trino.yaml 文件。

示例

```
[cephuser@host01 s3select]$ cat container/trino/hms_trino.yaml
version: '3'
services:
  hms:
    image: galsl/hms:dev
    container_name: hms
    environment:
      # S3_ENDPOINT the CEPH/RGW end-point-url
      - S3_ENDPOINT=http://rgw_ip:port
      - S3_ACCESS_KEY=abc
      - S3_SECRET_KEY=abc
    # the container starts with booting the hive metastore
    command: sh -c '. ~/.bashrc; start_hive_metastore'
```

```
    ports:
      - 9083:9083
    networks:
      - trino_hms

  trino:
    image: trinodb/trino:405
    container_name: trino
    volumes:
      # the trino directory contains the necessary configuration
      - ./trino:/etc/trino
    ports:
      - 8080:8080
    networks:
      - trino_hms

networks:
  trino_hm
```

3. ***使用 S3 端点、访问密钥和 secret 密钥修改 hive.properties 文件。***

示例

```
[cephuser@host01 s3select]$ cat container/trino/trino/catalog/hive.properties
connector.name=hive
hive.metastore.uri=thrift://hms:9083

#hive.metastore.warehouse.dir=s3a://hive/

hive.allow-drop-table=true
hive.allow-rename-table=true
hive.allow-add-column=true
hive.allow-drop-column=true
hive.allow-rename-column=true

hive.non-managed-table-writes-enabled=true
hive.s3select-pushdown.enabled=true
hive.s3.aws-access-key=abc
hive.s3.aws-secret-key=abc

# should modify per s3-endpoint-url
hive.s3.endpoint=http://rgw_ip:port
#hive.s3.max-connections=1
#hive.s3select-pushdown.max-connections=1

hive.s3.connect-timeout=100s
```

```
hive.s3.socket-timeout=100s
hive.max-splits-per-second=10000
hive.max-split-size=128MB
```

4. 启动 **Trino 容器，以集成 Ceph 对象网关。**

示例

```
[cephuser@host01 s3select]$ sudo docker compose -f ./container/trino/hms_trino.yaml up -d
```

5. 验证**集成。**

示例

```
[cephuser@host01 s3select]$ sudo docker exec -it trino /bin/bash
trino@66f753905e82:/$ trino
trino> create schema hive.csvbkt1schema;
trino> create table hive.csvbkt1schema.polariondatacsv(c1 varchar,c2 varchar, c3 varchar,
c4 varchar, c5 varchar, c6 varchar, c7 varchar, c8 varchar, c9 varchar) WITH (
external_location = 's3a://csvbkt1/',format = 'CSV');
trino> select * from hive.csvbkt1schema.polariondatacsv;
```

注意

外部位置必须指向存储桶名称或目录，而不是文件末尾。

第 4 章 CEPH 对象网关和 SWIFT API

作为开发者，您可以使用与 Swift API 数据访问模型兼容的 RESTful 应用程序编程接口(API)。您可以通过 Ceph 对象网关管理 Red Hat Ceph Storage 集群中存储的 bucket 和对象。

下表描述了当前 Swift 功能功能的支持状态：

表 4.1. 功能

功能	Status	备注
身份验证	支持	
获取帐户元数据	支持	没有自定义元数据
Swift ACL	支持	支持 Swift ACL 的子集
列出容器	支持	
列出容器的对象	支持	
创建容器	支持	
删除容器	支持	
获取容器元数据	支持	
添加/更新容器元数据	支持	
删除容器元数据	支持	
获取对象	支持	
创建/更新对象	支持	
创建大对象	支持	
删除对象	支持	
复制对象	支持	
获取对象元数据	支持	
添加/更新对象元数据	支持	

功能	Status	备注
临时 URL 操作	支持	
CORS	不支持	
过期对象	支持	
对象版本控制	不支持	
静态网站	不支持	

先决条件

- *一个正在运行的 Red Hat Ceph Storage 集群。*
- *RESTful 客户端。*

4.1. SWIFT API 限制

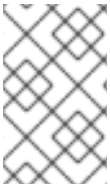
重要

应谨慎使用以下限制。您的硬件选择会有影响，因此您应该始终与您的红帽客户团队讨论这些要求。

- *使用 Swift API 时的最大对象大小： 5GB*
- *使用 Swift API 时的最大元数据大小： 对可应用到对象的用户元数据的总大小没有定义限制，但单个 HTTP 请求限制为 16,000 字节。*

4.2. 创建 SWIFT 用户

要测试 Swift 接口，请创建一个 Swift 子用户。创建 Swift 用户包含两个步骤。第一步是创建用户。第二步是创建机密密钥。



注意

在多站点部署中，始终在 **master zone group** 的 **master zone** 中的主机上创建用户。

先决条件

- 安装 **Ceph** 对象网关.
- **Ceph** 对象网关节点的根级别访问权限.

流程

1. 创建 **Swift** 用户：

语法

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

使用 **Swift** 用户名替换 **NAME**，例如：

示例

```
[root@host01 ~]# radosgw-admin subuser create --uid=testuser --subuser=testuser:swift --
access=full
{
    "user_id": "testuser",
    "display_name": "First User",
    "email": "",
    "suspended": 0,
    "max_buckets": 1000,
    "auid": 0,
    "subusers": [
        {
            "id": "testuser:swift",
            "permissions": "full-control"
        }
```

```
    ],
    "keys": [
        {
            "user": "testuser",
            "access_key": "O8JDE41XMI74O185EHKD",
            "secret_key": "i4Au2yxG5wtr1JK01mI8kjJPM93HNAoVWOSTdJd6"
        }
    ],
    "swift_keys": [
        {
            "user": "testuser:swift",
            "secret_key": "13TLtdEW7bCqgttQgPzxFxziu0AgabtOc6vM8DLA"
        }
    ],
    "caps": [],
    "op_mask": "read, write, delete",
    "default_placement": "",
    "placement_tags": [],
    "bucket_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
    "user_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
    "temp_url_keys": [],
    "type": "rgw"
}
```

2. ***创建 secret 密钥：***

语法

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

使用 Swift 用户名替换 NAME，例如：

示例

```
[root@host01 ~]# radosgw-admin key create --subuser=testuser:swift --key-type=swift --gen-
secret
{
    "user_id": "testuser",
    "display_name": "First User",
    "email": "",
    "suspended": 0,
    "max_buckets": 1000,
    "auid": 0,
    "subusers": [
        {
            "id": "testuser:swift",
            "permissions": "full-control"
        }
    ],
    "keys": [
        {
            "user": "testuser",
            "access_key": "O8JDE41XMI74O185EHKD",
            "secret_key": "i4Au2yxG5wtr1JK01mI8kjJPM93HNAoVWOSTdJd6"
        }
    ],
    "swift_keys": [
        {
            "user": "testuser:swift",
            "secret_key": "a4ioT4jEP653CDcdU8p4OuhruwABBRZmyNUbnSSt"
        }
    ],
    "caps": [],
    "op_mask": "read, write, delete",
    "default_placement": "",
    "placement_tags": [],
    "bucket_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
    "user_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
```

```
    "temp_url_keys": [],
    "type": "rgw"
}
```

4.3. SWIFT 验证用户

要对用户进行身份验证，请在标题中创建一个包含 X-Auth-User 和 X-Auth-Key 的请求。

语法

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

响应示例

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

注意

您可以通过在身份验证过程中使用 X-Storage-Url 值执行 GET 请求来检索有关 Ceph Swift 兼容服务的数据。

其它资源

- 如需 Swift 请求标头，请参阅 Red Hat Ceph Storage Developer Guide。

- 如需 Swift 响应标头，请参阅 Red Hat Ceph Storage Developer Guide。

4.4. SWIFT 容器操作

作为开发者，您可以通过 Ceph 对象网关通过 Swift 应用编程接口(API)执行容器操作。您可以列出、创建、更新和删除容器。您还可以添加或更新容器的元数据。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。

- RESTful 客户端。

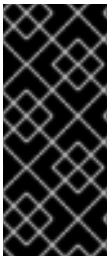
4.4.1. Swift 容器操作

容器是用于存储数据对象的机制。帐户可以有许多容器，但容器名称必须是唯一的。此 API 允许客户端创建容器，设置访问控制和元数据，检索容器的内容，以及删除容器。由于此 API 对特定用户帐户中的信息发出与请求相关的请求，因此此 API 中的所有请求都必须进行身份验证，除非容器访问控制被有意公开访问，即允许匿名请求。

注意

Amazon S3 API 使用术语"bucket"来描述数据容器。当您听到在 Swift API 中提到术语 'bucket' 时，这个术语 'bucket' 可能被认为与 "container" 这一术语相当。

对象存储的一个方面是它不支持分层路径或目录。相反，它支持由一个或多个容器组成的一个级别，每个容器可能具有对象。RADOS 网关的 Swift API 支持 'pseudo-hierarchical 容器' 的概念，这是一种使用对象命名来模拟容器或目录层次结构，而无需实际在存储系统中实施。您可以使用伪层次结构名称来命名对象，例如 photos/buildings/empire-state.jpg，但容器名称不能包含正斜杠(/)字符。



重要

将大型对象上传到版本的 Swift 容器时，在 python-swiftclient 程序中使用 --leave-segments 选项。不要使用 --leave-segments 覆盖清单文件。因此，现有对象会被覆盖，这会导致数据丢失。

4.4.2. Swift 更新容器的访问控制列表(ACL)

当用户创建容器时，用户默认对容器具有读写权限。要允许其他用户读取容器的内容或写入容器，您必须特别启用该用户。您也可以在 X-Container-Read 或 X-Container-Write 设置中指定 *，这样可以有效地让所有用户从或写入容器。设置 * 使容器变为公共容器。这是它允许匿名用户从容器读取或写入到容器。

语法

```
POST /API_VERSION/ACCOUNT/TENANT:CONTAINER HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
X-Container-Read: *
X-Container-Write: UID1, UID2, UID3
```

请求 Headers

x-Container-Read

描述

具有容器读取权限的用户 ID。

Type

用户 ID 的以逗号分隔的字符串值。

必需

否

X-Container-Write

描述

具有容器写入权限的用户 ID。

Type

用户 ID 的以逗号分隔的字符串值。

必需

否

4.4.3. Swift 列出容器

指定 API 版本并且帐户返回特定用户帐户的容器列表的 GET 请求。由于请求返回特定用户的容器，因此请求需要身份验证令牌。无法匿名发出请求。

语法

```
GET /API_VERSION/ACCOUNT HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

请求参数

limit

描述

将结果数量限制为指定的值。

Type

整数

有效值

不适用

必需

是

格式

描述

将结果数量限制为指定的值。

Type

整数

有效值

json 或 xml

必需

否

marker

描述

返回大于标记值的结果列表。

Type

字符串

有效值

不适用

必需

否

响应包含容器列表，或使用 HTTP 204 响应代码返回。

响应实体

account

描述

帐户信息列表。

Type

Container

container

描述

容器列表。

Type

Container

名称

描述

容器的名称。

Type

字符串

bytes

描述

容器的大小。

Type

整数

4.4.4. Swift 列出容器的对象

要列出容器内的对象，请使用 API 版本、帐户和容器名称来创建一个 GET 请求。您可以指定查询参数来过滤完整列表，或用参数返回存储在容器中的前 10,000 项名称的列表。

语法

```
GET /API_VERSION/TENANT:CONTAINER HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

请求参数

格式

描述

将结果数量限制为指定的值。

Type

整数

有效值

json 或 xml

必需

否

prefix

描述

将结果设置为以指定前缀开头的对象。

Type

字符串

有效值

不适用

必需

否

marker

描述

返回大于标记值的结果列表。

Type

字符串

有效值

不适用

必需

否

limit

描述

将结果数量限制为指定的值。

Type

整数

有效值

0 - 10,000

必需

否

delimiter

描述

前缀与其他对象名称之间的分隔符。

Type

字符串

有效值

不适用

必需

否

path

描述

对象的伪层次结构路径。

Type

字符串

有效值

不适用

必需

否

响应实体

container

描述

容器。

Type

Container

object

描述

容器内的对象。

Type

Container

名称

描述

容器中对象的名称。

Type

字符串

hash

描述

对象内容的散列代码。

Type

字符串

last_modified

描述

最后一次修改对象内容的时间。

Type

Date

content_type

描述

对象中的内容类型。

Type

字符串

4.4.5. Swift 创建容器

若要创建新容器，请为 PUT 请求提供 API 版本、帐户和新容器的名称。容器名称必须是唯一的，不得包含正斜杠(/)字符，且应小于 256 字节。您可以在请求中包含访问控制标头和元数据标头。您还可以包含指定一组 PG 的密钥的存储策略。例如，执行 radosgw-admin zone get 以查看 placement_pools 下的可用键的列表。存储策略允许您为容器指定一组特殊的池，如基于 SSD 的存储。操作是幂等的。如果您请求创建已存在的容器，它会返回 HTTP 202 返回代码，但不会创建另一个容器。

语法

```
PUT /API_VERSION/ACCOUNT/TENANT:CONTAINER HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
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
```

Headers

x-Container-Read

描述

具有容器读取权限的用户 ID。

Type

用户 ID 的以逗号分隔的字符串值。

必填

否

X-Container-Write

描述

具有容器写入权限的用户 ID。

Type

用户 ID 的以逗号分隔的字符串值。

必填

否

X-Container-Meta-KEY

描述

用户定义的元数据键，它采用任意字符串值。

Type

字符串

必填

否

x-Storage-Policy

描述

在 Ceph 对象网关的 placement_pools 下标识存储策略的密钥。为可用的密钥执行 radosgw-admin zone get。

Type

字符串

必填

否

如果容器已存在具有相同名称的容器，并且用户是容器所有者，则操作将成功。否则，操作将失败。

HTTP 响应

409

状态代码

BucketAlreadyExists

描述

容器已存在于其他用户的所有权下。

4.4.6. Swift 删除容器

要删除容器，请使用 API 版本、帐户和容器名称进行 DELETE 请求。容器必须为空。如果您想检查容器是否为空，请对容器执行 HEAD 请求。成功移除容器后，您将能够重复利用容器名称。

语法

```
DELETE /API_VERSION/ACCOUNT/TENANT:CONTAINER HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

HTTP 响应

204

状态代码

NoContent

描述

容器已被删除。

4.4.7. Swift 添加或更新容器元数据

要向容器添加元数据，请使用 API 版本、帐户和容器名称发出 POST 请求。容器必须具有写入权限才能添加或更新元数据。

语法

```
POST /API_VERSION/ACCOUNT/TENANT:CONTAINER HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
X-Container-Meta-Color: red
X-Container-Meta-Taste: salty
```

请求 Headers

X-Container-Meta-KEY

描述

用户定义的元数据键，它采用任意字符串值。

Type

字符串

必填

否

4.5. SWIFT 对象操作

作为开发者，您可以通过 Ceph 对象网关通过 Swift 应用编程接口(API)执行对象操作。您可以列出、创建、更新和删除对象。您还可以添加或更新对象的元数据。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- RESTful 客户端。

4.5.1. Swift 对象操作

对象是用于存储数据和元数据的容器。容器可能有许多对象，但对象名称必须是唯一的。此 API 允许客户端创建对象，设置访问控制和元数据，检索对象的数据和元数据，以及删除对象。由于此 API 发出与特定用户帐户中信息相关的请求，因此此 API 中的所有请求都必须经过身份验证。除非容器或对象的访问控制被有意公开访问，否则这是允许匿名请求。

4.5.2. Swift 获取对象

要检索对象，请使用 API 版本、帐户、容器和对象名称发出 GET 请求。必须具有容器的读取权限，以检索对象。

语法

```
GET /API_VERSION/ACCOUNT/TENANT:CONTAINER/OBJECT HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

请求 Headers

range

描述

要检索对象内容的子集，您可以指定一个字节范围。

Type

Date

必需

否

if-Modified-Since

描述

仅在自源对象的 last_modified 属性的日期和时间修改时才会复制。

Type

Date

必需

否

If-Unmodified-Since

描述

仅在不修改源对象的 last_modified 属性的日期和时间时复制。

Type

Date

必需

否

Copy-If-Match

描述

只有在请求中的 ETag 与源对象的 ETag 匹配时才复制。

Type

ETag

必需

否

Copy-If-None-Match

描述

只有在请求中的 ETag 不与源对象的 ETag 匹配时才复制。

Type

ETag

必需

否

响应标头

Content-Range

描述

对象内容子集的范围。仅在请求中指定范围标头字段时返回。

4.5.3. Swift 创建或更新对象

要创建新对象，请使用 API 版本、帐户、容器名称和新对象的名称发出 PUT 请求。您必须具有容器的写入权限才能创建或更新对象。对象名称在容器内必须是唯一的。PUT 请求不是幂等的，因此如果您不使用唯一名称，则请求将更新对象。但是，您可以在对象名称中使用伪层次结构语法，如果它位于不同的伪层次结构目录中，将其与同一名称的另一个对象区分开来。您可以在请求中包含访问控制标头和元数据标头。

语法

```
PUT /API_VERSION/ACCOUNT/TENANT:CONTAINER HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

请求 Headers

ETag

描述

对象内容的 MD5 哈希。推荐选项。

Type

字符串

有效值

不适用

必需

否

Content-Type

描述

对象内容的 MD5 哈希。

Type

字符串

有效值

不适用

必需

否

Transfer-Encoding

描述

指明对象是更大聚合对象的一部分。

Type

字符串

有效值

chunked

必需

否

4.5.4. Swift 删除对象

要删除对象，请使用 API 版本、帐户、容器和对象名称发出 DELETE 请求。您必须具有容器的写入权限，才能删除其中的对象。成功删除对象后，您将能够重复利用对象名称。

语法

```
DELETE /API_VERSION/ACCOUNT/TENANT:CONTAINER/OBJECT HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

4.5.5. Swift 复制对象

通过复制对象，您可以制作对象的服务器端副本，因此您不必下载对象并将其上传到其他容器下。要将一个对象的内容复制到另一个对象，您可以使用 API 版本、帐户和容器名称发出 PUT 请求或 COPY 请求。

对于 PUT 请求，请使用请求中的目标容器和对象名称，以及请求标头中的源容器和对象。

对于 Copy request，请使用请求中的源容器和对象，以及请求标头中的目标容器和对象。您必须具有容器的写入权限才能复制对象。目标对象名称在容器内必须是唯一的。请求不是幂等的，因此如果您不使

用唯一名称，则请求将更新目标对象。如果目标名称位于不同的伪层次结构目录中，您可以使用伪层次结构语法，将目的地对象与同一名称的源对象区分开。您可以在请求中包含访问控制标头和元数据标头。

语法

```
PUT /API_VERSION/ACCOUNT/TENANT:CONTAINER HTTP/1.1
X-Copy-From: TENANT:SOURCE_CONTAINER/SOURCE_OBJECT
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

或者：

语法

```
COPY /API_VERSION/ACCOUNT/TENANT:SOURCE_CONTAINER/SOURCE_OBJECT HTTP/1.1
Destination: TENANT:DEST_CONTAINER/DEST_OBJECT
```

请求 Headers

X-Copy-From

描述

与 PUT 请求一起使用，以定义源容器/对象路径。

Type

字符串

必需

是，如果使用 PUT。

目的地

描述

与 COPY 请求一起使用以定义目标容器/对象路径。

Type

字符串

必需

是，如果使用 COPY。

If-Modified-Since

描述

仅在自源对象的 last_modified 属性的日期和时间修改时才会复制。

Type

Date

必需

否

If-Unmodified-Since

描述

仅在不修改源对象的 last_modified 属性的日期和时间时复制。

Type

Date

必需

否

Copy-If-Match

描述

只有在请求中的 ETag 与源对象的 ETag 匹配时才复制。

Type

ETag

必需

否

Copy-If-None-Match

描述

只有在请求中的 ETag 不与源对象的 ETag 匹配时才复制。

Type

ETag

必需

否

4.5.6. Swift 获取对象元数据

要检索对象的元数据，请使用 API 版本、帐户、容器和对象名称发出 HEAD 请求。您必须具有容器的读取权限，以便从容器内的对象检索元数据。此请求返回与对象本身的请求相同的标头信息，但不会返回对象的数据。

语法

```
HEAD /API_VERSION/ACCOUNT/TENANT:CONTAINER/OBJECT HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

4.5.7. Swift 添加或更新对象元数据

要将元数据添加到对象，请使用 API 版本、帐户、容器和对象名称发出 POST 请求。父容器必须具有写入权限才能添加或更新元数据。

语法

```
POST /API_VERSION/ACCOUNT/TENANT:CONTAINER/OBJECT HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

请求 Headers

X-Object-Meta-KEY

描述

用户定义的 meta 数据键，它采用任意字符串值。

Type

字符串

必填

否

4.6. SWIFT 临时 URL 操作

要允许临时访问，temp url 功能由 radosgw 的 swift 端点支持。例如，GET 请求到对象，无需共享凭据。

对于此功能，首先应设置 X-Account-Meta-Temp-URL-Key 和 optionally X-Account-Meta-Temp-URL-URL-Key-2 的值。Temp URL 功能依赖于针对这些 secret 密钥的 HMAC-SHA1 签名。

4.7. SWIFT 获取临时 URL 对象

临时 URL 使用加密 HMAC-SHA1 签名，其中包含以下元素：

- *Request 方法的值，实例"GET"*

- 自 epoch （即 Unix 时间）以来的过期时间（以秒为单位）
- 从 "v1" 开始的请求路径

以上项目在它们之间附加了换行符，并使用 SHA-1 哈希算法（之前发布的 Temp URL 键）生成 HMAC。

要演示上述的 python 脚本的示例如下：

示例

```
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
```

输出示例

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

4.8. SWIFT POST 临时 URL 密钥

使用所需密钥向 swift 帐户发出 POST 请求，将为帐户提供临时 URL 访问的帐户的 secret 临时 URL 密钥。支持两个密钥，并且会针对密钥检查签名（如果存在），因此这些密钥可以被轮转，而不会无效的临时 URL。

语法

```
POST /API_VERSION/ACCOUNT HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

请求 Headers

X-Account-Meta-Temp-URL-Key

描述

使用任意字符串值的用户定义的键。

Type

字符串

必填

是

X-Account-Meta-Temp-URL-Key-2

描述

使用任意字符串值的用户定义的键。

Type

字符串

必填

否

4.9. SWIFT 多租户容器操作

当客户端应用访问容器时，它始终使用特定用户的凭证运行。在 Red Hat Ceph Storage 集群中，每个用户都属于一个租户。因此，如果没有明确指定租户，则每个容器操作在其上下文中都有一个隐式租户。因此，多租户与之前的版本完全向后兼容，只要引用容器并引用用户所属的租户。

已根据所使用的协议和身份验证系统，使用扩展来指定明确的租户会有所不同。

冒号分隔租户和容器，因此一个示例 URL 为：

示例

```
https://rgw.domain.com/tenant:container
```

相反，在 create_container （） 方法中，只需将租户和容器分隔到容器方法本身中：

示例

```
create_container("tenant:container")
```

附录 A. CEPH RESTFUL API 规格

作为存储管理员，您可以通过 Ceph RESTful API 端点访问各种 Ceph 子系统。这是可用 Ceph RESTful API 方法的参考指南。

可用的 Ceph API 端点：

- 第 A.27 节 “Telemetry”
- 第 A.28 节 “Ceph 用户”

先决条件

- 了解如何使用 RESTful API。
- 一个正常运行的 Red Hat Ceph Storage 集群。
- 启用 Ceph Manager dashboard 模板。

A.1. CEPH 概述

方法引用，使用 Ceph RESTful API summary 端点显示 Ceph 概述详情。

GET /api/summary

描述

显示 Ceph 详细信息的摘要。

示例

```
GET /api/summary HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。

- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

其它资源

- 如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。

A.2. 认证

方法引用，使用 Ceph RESTful API auth 端点来发起与 Red Hat Ceph Storage 的会话。

POST /api/auth

curl 示例

```
curl -i -k --location -X POST 'https://192.168.0.44:8443/api/auth' -H 'Accept:
application/vnd.ceph.api.v1.0+json' -H 'Content-Type: application/json' --data '{"password":
"admin@123", "username": "admin"}'
```

示例

```
POST /api/auth HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "password": "STRING",
  "username": "STRING"
}
```

状态代码

- 201 created - 资源已创建。
- 202 accepted - 操作仍在执行。请检查任务队列。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。

- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

POST /api/auth/check

描述

检查身份验证令牌的要求。

示例

```
POST /api/auth/check?token=STRING HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "token": "STRING"
}
```

状态代码

- 201 created - 资源已创建。
- 202 accepted - 操作仍在执行。请检查任务队列。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

POST /api/auth/logout

状态代码

- *201 created - 资源已创建。*
- *202 accepted - 操作仍在执行。请检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

其它资源

- *如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。*

A.3. CEPH 文件系统

方法引用，使用 Ceph RESTful API cephfs 端点管理 Ceph 文件系统(CephFS)。

GET /api/cephfs

示例

```
GET /api/cephfs HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*

- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/cephfs/FS_ID

参数

- *使用 Ceph 文件系统标识符字符串替换 FS_ID。*

示例

```
GET /api/cephfs/FS_ID HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

DELETE /api/cephfs/FS_ID/client/CLIENT_ID

参数

- 使用 Ceph 文件系统标识符字符串替换 FS_ID。
- 将 CLIENT_ID 替换为 Ceph 客户端标识符字符串。

状态代码

- 202 accepted - 操作仍在执行。请检查任务队列。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

GET /api/cephfs/FS_ID/clients

参数

- 使用 Ceph 文件系统标识符字符串替换 FS_ID。

示例

```
GET /api/cephfs/FS_ID/clients HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。

- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/cephfs/FS_ID/get_root_directory

描述

使用 ls_dir API 调用无法获取的根目录。

参数

- *使用 Ceph 文件系统标识符字符串替换 FS_ID。*

示例

```
GET /api/cephfs/FS_ID/get_root_directory HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/cephfs/FS_ID/ls_dir

描述

列出指定路径的目录。

参数

- *使用 Ceph 文件系统标识符字符串替换 FS_ID。*
- *queries:*
  - *path - 要开始列表的字符串值。默认路径为 /，如果没有指定。*
  - *depth - 整数值指定要减少目录树的步骤数。*

示例

```
GET /api/cephfs/FS_ID/ls_dir HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/cephfs/FS_ID/mds_counters

参数

- 使用 Ceph 文件系统标识符字符串替换 FS_ID。
- queries:
  - 计数器 - 整数值。

示例

```
GET /api/cephfs/FS_ID/mds_counters HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

GET /api/cephfs/FS_ID/quota

描述

显示给定路径的 CephFS 配额。

参数

- 使用 Ceph 文件系统标识符字符串替换 FS_ID。
- queries:

- path - 指定目录路径所需的字符串值。

示例

```
GET /api/cephfs/FS_ID/quota?path=STRING HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

PUT /api/cephfs/FS_ID/quota

描述

为给定路径设置配额。

参数

- 使用 Ceph 文件系统标识符字符串替换 FS_ID。
- max_bytes - 定义字节限制的字符串值。
- max_files - 定义文件限制的字符串值。
- path - 定义到目录或文件的路径的字符串值。

示例

```
PUT /api/cephfs/FS_ID/quota HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "max_bytes": "STRING",
    "max_files": "STRING",
    "path": "STRING"
}
```

状态代码

- **200 OK - 确定。**
- **202 accepted - Operation 仍在执行，检查任务队列。**
- **400 Bad Request – Operation exception.请检查响应正文以了解详细信息。**
- **401 未授权 - 未验证的访问.请首先登录。**
- **403 Forbidden – Unauthorized access.请检查您的权限。**
- **500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。**

DELETE /api/cephfs/FS_ID/snapshot

描述

删除一个快照。

参数

- **使用 Ceph 文件系统标识符字符串替换 FS_ID。**
- **queries:**

- name - 指定快照名称所需的字符串值。
- path - 定义到目录的路径所需的字符串值。

状态代码

- 202 accepted - Operation 仍在执行，检查任务队列。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

POST /api/cephfs/FS_ID/snapshot

描述

创建快照。

参数

- 使用 Ceph 文件系统标识符字符串替换 FS_ID。
- name - 指定快照名称的字符串值。如果没有指定名称，则会生成使用 RFC3339 UTC 格式的当前时间的名称。
- path - 定义到目录的路径的字符串值。

示例

```
POST /api/cephfs/FS_ID/snapshot HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "name": "STRING",
  "path": "STRING"
}
```

状态代码

- **201 created - 资源已创建。**
- **202 accepted - Operation 仍在执行，检查任务队列。**
- **400 Bad Request – Operation exception.请检查响应正文以了解详细信息。**
- **401 未授权 - 未验证的访问.请首先登录。**
- **403 Forbidden – Unauthorized access.请检查您的权限。**
- **500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。**

DELETE /api/cephfs/FS_ID/tree

描述

删除目录。

参数

- **使用 Ceph 文件系统标识符字符串替换 FS_ID。**
- **queries:**

- path - 定义到目录的路径所需的字符串值。

状态代码

- 202 accepted - Operation 仍在执行，检查任务队列。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

POST /api/cephfs/FS_ID/tree

描述

创建一个目录。

参数

- 使用 Ceph 文件系统标识符字符串替换 FS_ID。
- path - 定义到目录的路径的字符串值。

示例

```
POST /api/cephfs/FS_ID/tree HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "path": "STRING"
}
```

状态代码

- *201 created - 资源已创建。*
- *202 accepted - Operation 仍在执行，检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

其它资源

- *如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。*

A.4. 存储集群配置

方法引用，使用 Ceph RESTful API cluster_conf 端点管理 Red Hat Ceph Storage 集群。

GET /api/cluster_conf

示例

```
GET /api/cluster_conf HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*

- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

POST /api/cluster_conf

示例

```
POST /api/cluster_conf HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "name": "STRING",
  "value": "STRING"
}
```

状态代码

- *201 created - 资源已创建。*
- *202 accepted - Operation 仍在执行，检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

PUT /api/cluster_conf

示例

```
PUT /api/cluster_conf HTTP/1.1
Host: example.com
Content-Type: application/json

{
   "options": "STRING"
}
```

状态代码

- **200 OK - 确定。**
- **202 accepted - Operation 仍在执行，检查任务队列。**
- **400 Bad Request – Operation exception.请检查响应正文以了解详细信息。**
- **401 未授权 - 未验证的访问.请首先登录。**
- **403 Forbidden – Unauthorized access.请检查您的权限。**
- **500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。**

GET /api/cluster_conf/filter

描述

按名称显示存储集群配置。

参数

- **queries:**
  - **names - 配置选项名称的字符串值。**

示例

```
GET /api/cluster_conf/filter HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

DELETE /api/cluster_conf/NAME

参数

- 使用存储集群配置名称替换 NAME。
- queries:
  - section - 所需的字符串值。

状态代码

- 202 accepted - Operation 仍在执行，检查任务队列。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。

- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/cluster_conf/NAME

参数

- *使用存储集群配置名称替换 NAME。*

示例

```
GET /api/cluster_conf/NAME HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

其它资源

- *如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。*

A.5. CRUSH 规则

方法引用，使用 Ceph RESTful API crush_rule 端点来管理 CRUSH 规则。

GET /api/crush_rule

描述

列出 CRUSH 规则配置。

示例

```
GET /api/crush_rule HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

POST /api/crush_rule

示例

```
POST /api/crush_rule HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "device_class": "STRING",
  "failure_domain": "STRING",
  "name": "STRING",
  "root": "STRING"
}
```

状态代码

- *201 created - 资源已创建。*
- *202 accepted - Operation 仍在执行，检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

DELETE /api/crush_rule/NAME

参数

- *使用规则名称替换 NAME。*

状态代码

- *202 accepted - Operation 仍在执行，检查任务队列。*
- *204 No Content – Resource deleted.*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*

- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/crush_rule/NAME

参数

- *使用规则名称替换 NAME。*

示例

```
GET /api/crush_rule/NAME HTTP/1.1
Host: example.com
```

状态代码

- *202 accepted - Operation 仍在执行，检查任务队列。*
- *204 No Content – Resource deleted.*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

其它资源

- *如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。*

A.6. 纠删代码 PROFILE

使用 Ceph RESTful API erasure_code_profile 端点来管理 Ersaure 编码的配置集引用。

GET /api/erasure_code_profile

描述

列出纠删代码 profile 信息。

示例

```
GET /api/erasure_code_profile HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

POST /api/erasure_code_profile

示例

```
POST /api/erasure_code_profile HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "name": "STRING"
}
```

状态代码

- 201 created - 资源已创建。

- *202 accepted - Operation 仍在执行，检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

DELETE /api/erasure_code_profile/NAME

参数

- *使用配置集名称替换 NAME。*

状态代码

- *202 accepted - Operation 仍在执行，检查任务队列。*
- *204 No Content – Resource deleted.*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/erasure_code_profile/NAME

参数

- *使用配置集名称替换 NAME。*

示例

```
GET /api/erasure_code_profile/NAME HTTP/1.1
Host: example.com
```

状态代码

- *202 accepted - Operation 仍在执行，检查任务队列。*
- *204 No Content – Resource deleted.*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

其它资源

- *如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。*

A.7. 功能切换

方法引用，使用 Ceph RESTful API feature_toggles 端点来管理 CRUSH 规则。

GET /api/feature_toggles

描述

列出 Red Hat Ceph Storage 的功能。

示例

```
GET /api/feature_toggles HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

其它资源

- 如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。

A.8. GRAFANA

方法引用，使用 Ceph RESTful API grafana 端点管理 Grafana。

POST /api/grafana/dashboards

状态代码

- 201 created - 资源已创建。
- 202 accepted - 操作仍在执行。请检查任务队列。

- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/grafana/url

描述

列出 Grafana URL 实例。

示例

```
GET /api/grafana/url HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/grafana/validation/PARAMS

参数

- 使用字符串值替换 PARAMS。

示例

```
GET /api/grafana/validation/PARAMS HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

其它资源

- 如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。

A.9. 存储集群健康状况

使用 Ceph RESTful API health 端点来显示存储集群健康状况详细信息和状态的方法参考。

GET /api/health/full

示例

```
GET /api/health/full HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/health/minimal

描述

显示存储集群的最小健康报告。

示例

```
GET /api/health/minimal HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

其它资源

- 如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。

A.10. 主机

使用 Ceph RESTful API host 端点来显示主机（也称为节点）的方法参考。

GET /api/host

描述

列出主机规格。

参数

- queries:
  - sources - 主机源的字符串值。

示例

```
GET /api/host HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

POST /api/host

示例

```
POST /api/host HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "hostname": "STRING",
    "status": "STRING"
}
```

状态代码

- **201 created - 资源已创建。**
- **202 accepted - 操作仍在执行。请检查任务队列。**
- **400 Bad Request – Operation exception.请检查响应正文以了解详细信息。**
- **401 未授权 - 未验证的访问.请首先登录。**
- **403 Forbidden – Unauthorized access.请检查您的权限。**
- **500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。**

DELETE /api/host/HOST_NAME

参数

- **将 HOST_NAME 替换为节点的名称。**

状态代码

- **202 accepted - 操作仍在执行。请检查任务队列。**
- **204 No Content – Resource deleted.**

- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/host/HOST_NAME

描述

显示给定主机的信息。

参数

- *将 HOST_NAME 替换为节点的名称。*

示例

```
GET /api/host/HOST_NAME HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

PUT /api/host/HOST_NAME

描述

更新给定主机的信息。只有在启用 Ceph Orchestrator 时，才支持此方法。

参数

- *将 HOST_NAME 替换为节点的名称。*
- *force - 强制主机进入维护模式。*
- *Labels - 标签列表。*
- *Maintenance - 输入或退出维护模式。*
- *update_labels - 更新标签。*

示例

```
PUT /api/host/HOST_NAME HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "force": true,
  "labels": [
    "STRING"
  ],
  "maintenance": true,
  "update_labels": true
}
```

状态代码

- *200 OK - 确定。*
- *202 accepted - 操作仍在执行。请检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*

- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/host/HOST_NAME/daemons

参数

- *将 HOST_NAME 替换为节点的名称。*

示例

```
GET /api/host/HOST_NAME/daemons HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/host/HOST_NAME/devices

参数

- *将 HOST_NAME 替换为节点的名称。*

示例

```
GET /api/host/HOST_NAME/devices HTTP/1.1
Host: example.com
```

状态代码

- **200 OK - 确定。**
- **400 Bad Request – Operation exception.请检查响应正文以了解详细信息。**
- **401 未授权 - 未验证的访问.请首先登录。**
- **403 Forbidden – Unauthorized access.请检查您的权限。**
- **500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。**

POST /api/host/HOST_NAME/identify_device

描述

通过打开设备的 light 达到指定秒数来识别设备。

参数

- **将 HOST_NAME 替换为节点的名称。**
- **Device - 设备 ID，如 /dev/dm-0 或 ABC1234DEF567-1R1234_ABC8DE0Q。**
- **duration - 设备的 LED 应该闪存的秒数。**

示例

```
POST /api/host/HOST_NAME/identify_device HTTP/1.1
Host: example.com
Content-Type: application/json

{
```

```
    "device": "STRING",
    "duration": "STRING"
}
```

状态代码

- **201 created - 资源已创建。**
- **202 accepted - 操作仍在执行。请检查任务队列。**
- **400 Bad Request – Operation exception.请检查响应正文以了解详细信息。**
- **401 未授权 - 未验证的访问.请首先登录。**
- **403 Forbidden – Unauthorized access.请检查您的权限。**
- **500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。**

GET /api/host/HOST_NAME/inventory

描述

显示主机的清单。

参数

- **将 HOST_NAME 替换为节点的名称。**
- **queries:**
  - **refresh - 触发异步刷新的字符串值。**

示例

```
GET /api/host/HOST_NAME/inventory HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/host/HOST_NAME/smart

参数

- *将 HOST_NAME 替换为节点的名称。*

示例

```
GET /api/host/HOST_NAME/smart HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

其它资源

- 如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。

A.11. 日志

使用 Ceph RESTful API logs 端点来显示日志信息的方法参考。

GET /api/logs/all

描述

查看所有日志配置。

示例

```
GET /api/logs/all HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

其它资源

- 如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。

A.12. CEPH MANAGER 模块

方法引用，使用 Ceph RESTful API mgr/module 端点来管理 Ceph 管理器模块。

GET /api/mgr/module

描述

查看受管模块的列表。

示例

```
GET /api/mgr/module HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

GET /api/mgr/module/MODULE_NAME

描述

检索持久配置设置的值。

参数

- 将 MODULE_NAME 替换为 Ceph Manager 模块名称。

示例

```
GET /api/mgr/module/MODULE_NAME HTTP/1.1
Host: example.com
```

```
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

PUT /api/mgr/module/MODULE_NAME

描述

设置持久配置设置的值。

参数

- 将 MODULE_NAME 替换为 Ceph Manager 模块名称。
- config - 模块选项的值。

示例

```
PUT /api/mgr/module/MODULE_NAME HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "config": "STRING"
}
```

状态代码

- 200 OK - 确定。

- *202 accepted - 操作仍在执行。请检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

POST /api/mgr/module/MODULE_NAME/disable

描述

禁用给定的 Ceph Manager 模块。

参数

- *将 MODULE_NAME 替换为 Ceph Manager 模块名称。*

状态代码

- *201 created - 资源已创建。*
- *202 accepted - 操作仍在执行。请检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*

- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

POST /api/mgr/module/MODULE_NAME/enable

描述

启用给定的 Ceph Manager 模块。

参数

- *将 MODULE_NAME 替换为 Ceph Manager 模块名称。*

状态代码

- *201 created - 资源已创建。*
- *202 accepted - 操作仍在执行。请检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/mgr/module/MODULE_NAME/options

描述

查看给定 Ceph Manager 模块的选项。

参数

- *将 MODULE_NAME 替换为 Ceph Manager 模块名称。*

示例

```
GET /api/mgr/module/MODULE_NAME/options HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

其它资源

- *如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。*

A.13. CEPH MONITOR

方法引用，使用 Ceph RESTful API monitor 端点来显示 Ceph 监控器的信息。

GET /api/monitor

描述

查看 Ceph 监控详情。

示例

```
GET /api/monitor HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

其它资源

- *如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。*

A.14. CEPH OSD

方法引用，使用 Ceph RESTful API osd 端点管理 Ceph OSD。

GET /api/osd

示例

```
GET /api/osd HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*

- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

POST /api/osd

示例

```
POST /api/osd HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "data": "STRING",
  "method": "STRING",
  "tracking_id": "STRING"
}
```

状态代码

- *201 created - 资源已创建。*
- *202 accepted - 操作仍在执行。请检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/osd/flags

描述

查看 Ceph OSD 标记。

示例

```
GET /api/osd/flags HTTP/1.1
Host: example.com
```

状态代码

- **200 OK - 确定。**
- **400 Bad Request – Operation exception.请检查响应正文以了解详细信息。**
- **401 未授权 - 未验证的访问.请首先登录。**
- **403 Forbidden – Unauthorized access.请检查您的权限。**
- **500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。**

PUT /api/osd/flags

描述

为整个存储集群设置 Ceph OSD 标志。

参数

- **recovery_deletes, sortbitwise, 和 pglog_hardlimit 标志不能取消设置。**
- **无法设置 purged_snapshots 标志。**

重要

您必须包含这四个标记才能成功操作。

示例

```
PUT /api/osd/flags HTTP/1.1
Host: example.com
Content-Type: application/json
```

```
{
  "flags": [
    "STRING"
  ]
}
```

状态代码

- **200 OK - 确定。**
- **202 accepted - 操作仍在执行。请检查任务队列。**
- **400 Bad Request – Operation exception.请检查响应正文以了解详细信息。**
- **401 未授权 - 未验证的访问.请首先登录。**
- **403 Forbidden – Unauthorized access.请检查您的权限。**
- **500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。**

GET /api/osd/flags/individual

描述

查看各个 Ceph OSD 标志。

示例

```
GET /api/osd/flags/individual HTTP/1.1
Host: example.com
```

状态代码

- **200 OK - 确定。**
- **400 Bad Request – Operation exception.请检查响应正文以了解详细信息。**

- **401 未授权 - 未验证的访问.请首先登录。**
- **403 Forbidden – Unauthorized access.请检查您的权限。**
- **500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。**

PUT /api/osd/flags/individual

描述

更新 noout， noin、 nodown 和 noup 标志用于独立 Ceph OSD 子集。

示例

```
PUT /api/osd/flags/individual HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "flags": {
    "nodown": true,
    "noin": true,
    "noout": true,
    "noup": true
  },
  "ids": [
    1
  ]
}
```

状态代码

- **200 OK - 确定。**
- **202 accepted - 操作仍在执行。请检查任务队列。**
- **400 Bad Request – Operation exception.请检查响应正文以了解详细信息。**
- **401 未授权 - 未验证的访问.请首先登录。**

- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/osd/safe_to_delete

参数

- *queries:*
  - *svc_ids - Ceph OSD 服务标识符的必要字符串。*

示例

```
GET /api/osd/safe_to_delete?svc_ids=STRING HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/osd/safe_to_destroy

描述

检查 Ceph OSD 是否安全销毁。

参数

- *queries:*
  - *id - Ceph OSD 服务标识符的必要字符串。*

示例

```
GET /api/osd/safe_to_destroy?ids=STRING HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

DELETE /api/osd/SVC_ID

参数

- *将 SVC_ID 替换为 Ceph OSD 服务标识符的字符串值。*
- *queries:*
  - *preserve_id - 字符串值。*
  - *force - 字符串值。*

状态代码

- *202 accepted - 操作仍在执行。请检查任务队列。*
- *204 No Content – Resource deleted.*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/osd/SVC_ID

描述

返回有关 Ceph OSD 收集的数据。

参数

- *将 SVC_ID 替换为 Ceph OSD 服务标识符的字符串值。*

示例

```
GET /api/osd/SVC_ID HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*

- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

PUT /api/osd/SVC_ID

参数

- *将 SVC_ID 替换为 Ceph OSD 服务标识符的字符串值。*

示例

```
PUT /api/osd/SVC_ID HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "device_class": "STRING"
}
```

状态代码

- *200 OK - 确定。*
- *202 accepted - 操作仍在执行。请检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

POST /api/osd/SVC_ID/destroy

描述

将 Ceph OSD 标记为被销毁。Ceph OSD 在被销毁前必须标记为 down。此操作会使 Ceph OSD 标识符保持不变，但会删除 Cephx 密钥、配置密钥数据和锁定键。

警告

此操作会使数据永久可读。

参数

- 将 SVC_ID 替换为 Ceph OSD 服务标识符的字符串值。

状态代码

- 201 created - 资源已创建。
- 202 accepted - 操作仍在执行。请检查任务队列。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

GET /api/osd/SVC_ID/devices

参数

- 将 SVC_ID 替换为 Ceph OSD 服务标识符的字符串值。

示例

```
GET /api/osd/SVC_ID/devices HTTP/1.1
Host: example.com
```

状态代码

- **200 OK - 确定。**
- **400 Bad Request – Operation exception.请检查响应正文以了解详细信息。**
- **401 未授权 - 未验证的访问.请首先登录。**
- **403 Forbidden – Unauthorized access.请检查您的权限。**
- **500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。**

GET /api/osd/SVC_ID/histogram

描述

返回 Ceph OSD 直方图数据。

参数

- **将 SVC_ID 替换为 Ceph OSD 服务标识符的字符串值。**

示例

```
GET /api/osd/SVC_ID/histogram HTTP/1.1
Host: example.com
```

状态代码

- **200 OK - 确定。**
- **400 Bad Request – Operation exception.请检查响应正文以了解详细信息。**

- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

PUT /api/osd/SVC_ID/mark

描述

将 Ceph OSD 标记为 out, in, down, 和 lost。

注意

Ceph OSD 在标记为 lost 前，需要先将它标记为 down。

参数

- 将 SVC_ID 替换为 Ceph OSD 服务标识符的字符串值。

示例

```
PUT /api/osd/SVC_ID/mark HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "action": "STRING"
}
```

状态代码

- 200 OK - 确定。
- 202 accepted - 操作仍在执行。请检查任务队列。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。

- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

POST /api/osd/SVC_ID/purge

描述

从 CRUSH map 移除 Ceph OSD。

注意

在移除前，Ceph OSD 必须标记为 down。

参数

- 将 SVC_ID 替换为 Ceph OSD 服务标识符的字符串值。

状态代码

- 201 created - 资源已创建。
- 202 accepted - 操作仍在执行。请检查任务队列。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。

- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

POST /api/osd/SVC_ID/reweight

描述

临时重新加权 Ceph OSD。当 Ceph OSD 标记为 out 时，OSD 的权重设为 0。当 Ceph OSD 标记为 back 时，OSD 的权重设置为 1。

参数

- 将 SVC_ID 替换为 Ceph OSD 服务标识符的字符串值。

示例

```
POST /api/osd/SVC_ID/reweight HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "weight": "STRING"
}
```

状态代码

- 201 created - 资源已创建。
- 202 accepted - 操作仍在执行。请检查任务队列。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

POST /api/osd/SVC_ID/scrub

参数

- 将 SVC_ID 替换为 Ceph OSD 服务标识符的字符串值。
- queries:
  - deep - 布尔值，可以是 true 或 false。

示例

```
POST /api/osd/SVC_ID/scrub HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "deep": true
}
```

状态代码

- 201 created - 资源已创建。
- 202 accepted - 操作仍在执行。请检查任务队列。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

GET /api/osd/SVC_ID/smart

参数

- 将 SVC_ID 替换为 Ceph OSD 服务标识符的字符串值。

示例

```
GET /api/osd/SVC_ID/smart HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

其它资源

- 如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。

A.15. CEPH 对象网关

方法引用，使用 Ceph RESTful API rgw 端点管理 Ceph 对象网关。

GET /api/rgw/status

描述

显示 Ceph 对象网关状态。

示例

```
GET /api/rgw/status HTTP/1.1
Host: example.com
```

```
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

GET /api/rgw/daemon

描述

显示 Ceph 对象网关守护进程。

示例

```
GET /api/rgw/daemon HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。

- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/rgw/daemon/SVC_ID

参数

- *将 SVC_ID 替换为服务标识符作为字符串值。*

示例

```
GET /api/rgw/daemon/SVC_ID HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/rgw/site

参数

- *queries:*
  - *query - 字符串值。*
  - *daemon_name - 守护进程的名称作为字符串值。*

示例

```
GET /api/rgw/site HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

bucket 管理

GET /api/rgw/bucket

参数

- *queries:*
  - *stats - bucket 统计的布尔值。*
  - *daemon_name - 守护进程的名称作为字符串值。*

示例

```
GET /api/rgw/bucket HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

POST /api/rgw/bucket

示例

```
POST /api/rgw/bucket HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "bucket": "STRING",
  "daemon_name": "STRING",
  "lock_enabled": "false",
  "lock_mode": "STRING",
  "lock_retention_period_days": "STRING",
  "lock_retention_period_years": "STRING",
  "placement_target": "STRING",
  "uid": "STRING",
  "zonegroup": "STRING"
}
```

状态代码

- *201 created - 资源已创建。*
- *202 accepted - 操作仍在执行。请检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*

- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

DELETE /api/rgw/bucket/BUCKET

参数

- *将 BUCKET 替换为存储桶名称作为字符串值。*
- *queries:*
  - *purge_objects - 字符串值。*
  - *daemon_name - 守护进程的名称作为字符串值。*

状态代码

- *202 accepted - 操作仍在执行。请检查任务队列。*
- *204 No Content – Resource deleted.*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/rgw/bucket/BUCKET

参数

- ***将 BUCKET 替换为存储桶名称作为字符串值。***
- ***queries:***
  - ***daemon_name - 守护进程的名称作为字符串值。***

示例

```
GET /api/rgw/bucket/BUCKET HTTP/1.1
Host: example.com
```

状态代码

- ***200 OK - 确定。***
- ***400 Bad Request – Operation exception.请检查响应正文以了解详细信息。***
- ***401 未授权 - 未验证的访问.请首先登录。***
- ***403 Forbidden – Unauthorized access.请检查您的权限。***
- ***500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。***

PUT /api/rgw/bucket/BUCKET

参数

- ***将 BUCKET 替换为存储桶名称作为字符串值。***

示例

```
PUT /api/rgw/bucket/BUCKET HTTP/1.1
Host: example.com
Content-Type: application/json
```

```
{
    "bucket_id": "STRING",
    "daemon_name": "STRING",
    "lock_mode": "STRING",
    "lock_retention_period_days": "STRING",
    "lock_retention_period_years": "STRING",
    "mfa_delete": "STRING",
    "mfa_token_pin": "STRING",
    "mfa_token_serial": "STRING",
    "uid": "STRING",
    "versioning_state": "STRING"
}
```

状态代码

- **200 OK - 确定。**
- **202 accepted - 操作仍在执行。请检查任务队列。**
- **400 Bad Request – Operation exception.请检查响应正文以了解详细信息。**
- **401 未授权 - 未验证的访问.请首先登录。**
- **403 Forbidden – Unauthorized access.请检查您的权限。**
- **500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。**

用户管理

GET /api/rgw/user

描述

显示 **Ceph** 对象网关用户。

参数

- **queries:**

- ***daemon_name - 守护进程的名称作为字符串值。***

示例

```
GET /api/rgw/user HTTP/1.1
Host: example.com
```

状态代码

- ***200 OK - 确定。***
- ***400 Bad Request – Operation exception.请检查响应正文以了解详细信息。***
- ***401 未授权 - 未验证的访问.请首先登录。***
- ***403 Forbidden – Unauthorized access.请检查您的权限。***
- ***500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。***

POST /api/rgw/user

示例

```
POST /api/rgw/user HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "access_key": "STRING",
  "daemon_name": "STRING",
  "display_name": "STRING",
  "email": "STRING",
  "generate_key": "STRING",
  "max_buckets": "STRING",
  "secret_key": "STRING",
  "suspended": "STRING",
  "uid": "STRING"
}
```

状态代码

- *201 created - 资源已创建。*
- *202 accepted - 操作仍在执行。请检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/rgw/user/get_emails

参数

- *queries:*
  - *daemon_name - 守护进程的名称作为字符串值。*

示例

```
GET /api/rgw/user/get_emails HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*

- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

DELETE /api/rgw/user/UID

参数

- *使用用户标识符替换 UID 作为字符串。*
- *queries:*
  - *daemon_name - 守护进程的名称作为字符串值。*

状态代码

- *202 accepted - 操作仍在执行。请检查任务队列。*
- *204 No Content – Resource deleted.*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/rgw/user/UID

参数

- 使用用户标识符替换 UID 作为字符串。
- queries:
  - daemon_name - 守护进程的名称作为字符串值。
  - stats - 用户统计的布尔值。

示例

```
GET /api/rgw/user/UID HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

PUT /api/rgw/user/UID

参数

- 使用用户标识符替换 UID 作为字符串。

示例

```
PUT /api/rgw/user/UID HTTP/1.1
Host: example.com
```

```
Content-Type: application/json

{
    "daemon_name": "STRING",
    "display_name": "STRING",
    "email": "STRING",
    "max_buckets": "STRING",
    "suspended": "STRING"
}
```

状态代码

- **200 OK - 确定。**
- **202 accepted - 操作仍在执行。请检查任务队列。**
- **400 Bad Request – Operation exception.请检查响应正文以了解详细信息。**
- **401 未授权 - 未验证的访问.请首先登录。**
- **403 Forbidden – Unauthorized access.请检查您的权限。**
- **500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。**

DELETE /api/rgw/user/UID/capability

参数

- **使用用户标识符替换 UID 作为字符串。**
- **queries:**
  - **daemon_name - 守护进程的名称作为字符串值。**
  - **类型 - 必需。字符串值。**

- perm - 必需。字符串值。

状态代码

- 202 accepted - 操作仍在执行。请检查任务队列。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

POST /api/rgw/user/UID/capability

参数

- 使用用户标识符替换 UID 作为字符串。

示例

```
POST /api/rgw/user/UID/capability HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "daemon_name": "STRING",
  "perm": "STRING",
  "type": "STRING"
}
```

状态代码

- 201 created - 资源已创建。

- *202 accepted - 操作仍在执行。请检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

DELETE /api/rgw/user/UID/key

参数

- *使用用户标识符替换 UID 作为字符串。*
- *queries:*
  - *daemon_name - 守护进程的名称作为字符串值。*
  - *key_type - 字符串值。*
  - *subuser - 字符串值。*
  - *access_key - 字符串值。*

状态代码

- *202 accepted - 操作仍在执行。请检查任务队列。*
- *204 No Content – Resource deleted.*

- ***400 Bad Request – Operation exception.请检查响应正文以了解详细信息。***
- ***401 未授权 - 未验证的访问.请首先登录。***
- ***403 Forbidden – Unauthorized access.请检查您的权限。***
- ***500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。***

POST /api/rgw/user/UID/key

参数

- ***使用用户标识符替换 UID 作为字符串。***

示例

```
POST /api/rgw/user/UID/key HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "access_key": "STRING",
    "daemon_name": "STRING",
    "generate_key": "true",
    "key_type": "s3",
    "secret_key": "STRING",
    "subuser": "STRING"
}
```

状态代码

- ***201 created - 资源已创建。***
- ***202 accepted - 操作仍在执行。请检查任务队列。***
- ***400 Bad Request – Operation exception.请检查响应正文以了解详细信息。***

- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/rgw/user/UID/quota

参数

- *使用用户标识符替换 UID 作为字符串。*

示例

```
GET /api/rgw/user/UID/quota HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

PUT /api/rgw/user/UID/quota

参数

- *使用用户标识符替换 UID 作为字符串。*

示例

```
PUT /api/rgw/user/UID/quota HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "daemon_name": "STRING",
  "enabled": "STRING",
  "max_objects": "STRING",
  "max_size_kb": 1,
  "quota_type": "STRING"
}
```

状态代码

- **200 OK - 确定。**
- **202 accepted - 操作仍在执行。请检查任务队列。**
- **400 Bad Request – Operation exception.请检查响应正文以了解详细信息。**
- **401 未授权 - 未验证的访问.请首先登录。**
- **403 Forbidden – Unauthorized access.请检查您的权限。**
- **500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。**

POST /api/rgw/user/UID/subuser

参数

- **使用用户标识符替换 UID 作为字符串。**

示例

```
POST /api/rgw/user/UID/subuser HTTP/1.1
Host: example.com
Content-Type: application/json

{
```

```
    "access": "STRING",
    "access_key": "STRING",
    "daemon_name": "STRING",
    "generate_secret": "true",
    "key_type": "s3",
    "secret_key": "STRING",
    "subuser": "STRING"
}
```

状态代码

- 201 created - 资源已创建。
- 202 accepted - 操作仍在执行。请检查任务队列。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

DELETE /api/rgw/user/UID/subuser/SUBUSER

参数

- 使用用户标识符替换 UID 作为字符串。
- 将 SUBUSER 替换为字符串的子用户名。
- queries:
  - purge_keys - 设置为 false 以不清除密钥。这只适用于 S3 子用户。

- daemon_name - 守护进程的名称作为字符串值。

状态代码

- 202 accepted - 操作仍在执行。请检查任务队列。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

其它资源

- 如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。

A.16. 用于操作角色的 REST API

除了 radosgw-admin 角色 命令外，您还可以使用 REST API 来处理角色。

若要调用 REST admin API，请创建一个具有 admin caps 的用户。

示例

```
[root@host01 ~]# radosgw-admin --uid TESTER --display-name "TestUser" --access_key TESTER --
secret test123 user create
[root@host01 ~]# radosgw-admin caps add --uid="TESTER" --caps="roles=*"
```

- ***创建角色：***

语法

```
POST “<hostname>?
Action=CreateRole&RoleName=ROLE_NAME&Path=PATH_TO_FILE&AssumeRolePolicyDo
cument=TRUST_RELATIONSHIP_POLICY_DOCUMENT”
```

示例

```
POST “<hostname>?
Action=CreateRole&RoleName=S3Access&Path=/application_abc/component_xyz/&AssumeRo
ePolicyDocument={"Version":"2022-06-17","Statement":[{"Effect":"Allow","Principal":{"AWS":
["arn:aws:iam:::user/TESTER"]},"Action":["sts:AssumeRole"]}]}”
```

响应示例

```
<role>
  <id>8f41f4e0-7094-4dc0-ac20-074a881ccbc5</id>
  <name>S3Access</name>
  <path>/application_abc/component_xyz/</path>
  <arn>arn:aws:iam:::role/application_abc/component_xyz/S3Access</arn>
  <create_date>2022-06-23T07:43:42.811Z</create_date>
  <max_session_duration>3600</max_session_duration>
  <assume_role_policy_document>{"Version":"2022-06-17","Statement":
[{"Effect":"Allow","Principal":{"AWS":["arn:aws:iam:::user/TESTER"]},"Action":
["sts:AssumeRole"]}]}</assume_role_policy_document>
</role>
```

- ***获取角色：***

语法

```
POST “<hostname>?Action=GetRole&RoleName=ROLE_NAME”
```

示例

```
POST “<hostname>?Action=GetRole&RoleName=S3Access”
```

响应示例

```
<role>
  <id>8f41f4e0-7094-4dc0-ac20-074a881ccbc5</id>
  <name>S3Access</name>
  <path>/application_abc/component_xyz/</path>
  <arn>arn:aws:iam:::role/application_abc/component_xyz/S3Access</arn>
  <create_date>2022-06-23T07:43:42.811Z</create_date>
  <max_session_duration>3600</max_session_duration>
  <assume_role_policy_document>{"Version":"2022-06-17","Statement":
[{"Effect":"Allow","Principal":{"AWS":["arn:aws:iam:::user/TESTER"]},"Action":
["sts:AssumeRole"]}]}</assume_role_policy_document>
</role>
```

- ***列出角色：***

语法

```
POST “<hostname>?
Action=GetRole&RoleName=ROLE_NAME&PathPrefix=PATH_PREFIX”
```

请求示例

```
POST “<hostname>?Action=ListRoles&RoleName=S3Access&PathPrefix=/application”
```

响应示例

```
<role>
  <id>8f41f4e0-7094-4dc0-ac20-074a881ccbc5</id>
  <name>S3Access</name>
  <path>/application_abc/component_xyz/</path>
  <arn>arn:aws:iam:::role/application_abc/component_xyz/S3Access</arn>
  <create_date>2022-06-23T07:43:42.811Z</create_date>
  <max_session_duration>3600</max_session_duration>
  <assume_role_policy_document>{"Version":"2022-06-17","Statement":
[{"Effect":"Allow","Principal":{"AWS":["arn:aws:iam:::user/TESTER"]},"Action":
["sts:AssumeRole"]}]}</assume_role_policy_document>
</role>
```

- ***更新 assume 角色策略文档：***

语法

```
POST “<hostname>?
Action=UpdateAssumeRolePolicy&RoleName=ROLE_NAME&PolicyDocument=TRUST_RE
LATIONSHIP_POLICY_DOCUMENT”
```

示例

```
POST “<hostname>?
Action=UpdateAssumeRolePolicy&RoleName=S3Access&PolicyDocument=
{"Version":"2022-06-17","Statement":[{"Effect":"Allow","Principal":{"AWS":
["arn:aws:iam:::user/TESTER2"]},"Action":["sts:AssumeRole"]}]}”
```

- **更新附加到角色的策略：**

语法

```
POST “<hostname>?
Action=PutRolePolicy&RoleName=ROLE_NAME&PolicyName=POLICY_NAME&PolicyDocu
ment=TRUST_RELATIONSHIP_POLICY_DOCUMENT”
```

示例

```
POST “<hostname>?
Action=PutRolePolicy&RoleName=S3Access&PolicyName=Policy1&PolicyDocument=
{"Version":"2022-06-17","Statement":[{"Effect":"Allow","Action":
["s3:CreateBucket"],"Resource":"arn:aws:s3:::example_bucket"}]}”
```

- **列出附加到角色的权限策略名称：**

语法

```
POST “<hostname>?Action=ListRolePolicies&RoleName=ROLE_NAME”
```

示例

```
POST “<hostname>?Action=ListRolePolicies&RoleName=S3Access”

<PolicyNames>
  <member>Policy1</member>
</PolicyNames>
```

- **获取附加到角色的权限策略：**

语法

```
POST “<hostname>?
Action=GetRolePolicy&RoleName=ROLE_NAME&PolicyName=POLICY_NAME”
```

示例

```
POST “<hostname>?Action=GetRolePolicy&RoleName=S3Access&PolicyName=Policy1”

<GetRolePolicyResult>
  <PolicyName>Policy1</PolicyName>
  <RoleName>S3Access</RoleName>
  <Permission_policy>{"Version":"2022-06-17","Statement":[{"Effect":"Allow","Action":
["s3:CreateBucket"],"Resource":"arn:aws:s3:::example_bucket"}]}</Permission_policy>
</GetRolePolicyResult>
```

- ***删除附加到角色的策略：***

语法

```
POST “hostname>?
Action=DeleteRolePolicy&RoleName=ROLE_NAME&PolicyName=POLICY_NAME“
```

示例

```
POST “<hostname>?Action=DeleteRolePolicy&RoleName=S3Access&PolicyName=Policy1”
```

- ***删除角色：***

注意

只有在没有附加任何权限策略时，才能删除角色。

语法

```
POST “<hostname>?Action=DeleteRole&RoleName=ROLE_NAME"
```

示例

```
POST “<hostname>?Action=DeleteRole&RoleName=S3Access"
```

其它资源

- 详情请参阅 Red Hat Ceph Storage 对象网关指南中的角色管理部分。

A.17. NFS GANESHA

方法引用，使用 Ceph RESTful API nfs-ganesha 端点管理 Ceph NFS 网关。

GET /api/nfs-ganesha/daemon

描述

查看 NFS Ganesha 守护进程的相关信息。

示例

```
GET /api/nfs-ganesha/daemon HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

GET /api/nfs-ganesha/export

描述

查看所有 NFS Ganesha 导出。

示例

```
GET /api/nfs-ganesha/export HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

POST /api/nfs-ganesha/export

描述

创建新的 NFS Ganesha 导出。

示例

```
POST /api/nfs-ganesha/export HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "access_type": "STRING",
  "clients": [
    {
      "access_type": "STRING",
      "addresses": [
        "STRING"
      ],
      "squash": "STRING"
    }
  ],
  "cluster_id": "STRING",
```

```
  "daemons": [
    "STRING"
  ],
  "fsal": {
    "filesystem": "STRING",
    "name": "STRING",
    "rgw_user_id": "STRING",
    "sec_label_xattr": "STRING",
    "user_id": "STRING"
  },
  "path": "STRING",
  "protocols": [
    1
  ],
  "pseudo": "STRING",
  "reload_daemons": true,
  "security_label": "STRING",
  "squash": "STRING",
  "tag": "STRING",
  "transports": [
    "STRING"
  ]
}
```

状态代码

- **201 created - 资源已创建。**
- **202 accepted - 操作仍在执行。请检查任务队列。**
- **400 Bad Request – Operation exception.请检查响应正文以了解详细信息。**
- **401 未授权 - 未验证的访问.请首先登录。**
- **403 Forbidden – Unauthorized access.请检查您的权限。**
- **500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。**

DELETE /api/nfs-ganesha/export/CLUSTER_ID/EXPORT_ID

描述

删除 NFS Ganesha 导出。

参数

- 使用存储集群标识符字符串替换 CLUSTER_ID。
- 将 EXPORT_ID 替换为一个整数的导出标识符。
- queries:
  - reload_daemons - 一个布尔值，用于触发 NFS Ganesha 守护进程配置的重新加载。

状态代码

- 202 accepted - 操作仍在执行。请检查任务队列。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

GET /api/nfs-ganesha/export/CLUSTER_ID/EXPORT_ID

描述

查看 NFS Ganesha 导出信息。

参数

- 使用存储集群标识符字符串替换 CLUSTER_ID。
- 将 EXPORT_ID 替换为一个整数的导出标识符。

示例

```
GET /api/nfs-ganesha/export/CLUSTER_ID/EXPORT_ID HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

PUT /api/nfs-ganesha/export/CLUSTER_ID/EXPORT_ID

描述

更新 NFS Ganesha 导出信息。

参数

- 使用存储集群标识符字符串替换 CLUSTER_ID。
- 将 EXPORT_ID 替换为一个整数的导出标识符。

示例

```
PUT /api/nfs-ganesha/export/CLUSTER_ID/EXPORT_ID HTTP/1.1
```

```
Host: example.com
Content-Type: application/json

{
  "access_type": "STRING",
  "clients": [
    {
      "access_type": "STRING",
      "addresses": [
        "STRING"
      ],
      "squash": "STRING"
    }
  ],
  "daemons": [
    "STRING"
  ],
  "fsal": {
    "filesystem": "STRING",
    "name": "STRING",
    "rgw_user_id": "STRING",
    "sec_label_xattr": "STRING",
    "user_id": "STRING"
  },
  "path": "STRING",
  "protocols": [
    1
  ],
  "pseudo": "STRING",
  "reload_daemons": true,
  "security_label": "STRING",
  "squash": "STRING",
  "tag": "STRING",
  "transports": [
    "STRING"
  ]
}
```

状态代码

- **200 OK - 确定。**
- **202 accepted - 操作仍在执行。请检查任务队列。**
- **400 Bad Request – Operation exception.请检查响应正文以了解详细信息。**
- **401 未授权 - 未验证的访问.请首先登录。**

- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/nfs-ganesha/status

描述

查看 NFS Ganesha 管理功能的状态信息。

示例

```
GET /api/nfs-ganesha/status HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

其它资源

- *如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。*
- *如需更多信息，请参阅 Red Hat Ceph Storage 对象网关指南中的 将命名空间导出到 NFS-Ganesha 部分。*

A.18. CEPH 编排器

使用 Ceph RESTful API orchestrator 端点的方法引用，以显示 Ceph 编排器状态。

GET /api/orchestrator/status

描述

显示 Ceph 编排器状态。

示例

```
GET /api/orchestrator/status HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

其它资源

- 如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。

A.19. 池

使用 Ceph RESTful API pool 端点来管理存储池的方法参考。

GET /api/pool

描述

显示池列表。

参数

- *queries:*
  - attrs - 池属性的字符串值。
  - stats - 池统计数据的布尔值。

示例

```
GET /api/pool HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

POST /api/pool

示例

```
POST /api/pool HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "application_metadata": "STRING",
```

```
    "configuration": "STRING",
    "erasure_code_profile": "STRING",
    "flags": "STRING",
    "pg_num": 1,
    "pool": "STRING",
    "pool_type": "STRING",
    "rule_name": "STRING"
}
```

状态代码

- **201 created - 资源已创建。**
- **202 accepted - 操作仍在执行。请检查任务队列。**
- **400 Bad Request – Operation exception.请检查响应正文以了解详细信息。**
- **401 未授权 - 未验证的访问.请首先登录。**
- **403 Forbidden – Unauthorized access.请检查您的权限。**
- **500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。**

DELETE /api/pool/POOL_NAME

参数

- **将 POOL_NAME 替换为池的名称。**

状态代码

- **202 accepted - 操作仍在执行。请检查任务队列。**
- **204 No Content – Resource deleted.**
- **400 Bad Request – Operation exception.请检查响应正文以了解详细信息。**

- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/pool/POOL_NAME

参数

- *将 POOL_NAME 替换为池的名称。*
- *queries:*
  - *attrs - 池属性的字符串值。*
  - *stats - 池统计数据的布尔值。*

示例

```
GET /api/pool/POOL_NAME HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*

- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

PUT /api/pool/POOL_NAME

参数

- *将 POOL_NAME 替换为池的名称。*

示例

```
PUT /api/pool/POOL_NAME HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "application_metadata": "STRING",
  "configuration": "STRING",
  "flags": "STRING"
}
```

状态代码

- *200 OK - 确定。*
- *202 accepted - 操作仍在执行。请检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/pool/POOL_NAME/configuration

参数

- *将 POOL_NAME 替换为池的名称。*

示例

```
GET /api/pool/POOL_NAME/configuration HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

其它资源

- *如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。*

A.20. PROMETHEUS

方法引用，使用 Ceph RESTful API prometheus 端点管理 Prometheus。

GET /api/prometheus

示例

```
GET /api/prometheus/rules HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/prometheus/rules

示例

```
GET /api/prometheus/rules HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

POST /api/prometheus/silence

状态代码

- *201 created - 资源已创建。*
- *202 accepted - 操作仍在执行。请检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

DELETE /api/prometheus/silence/S_ID

参数

- *使用字符串值替换 S_ID。*

状态代码

- *202 accepted - 操作仍在执行。请检查任务队列。*
- *204 No Content – Resource deleted.*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*

- ***500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。***

GET /api/prometheus/silences

示例

```
GET /api/prometheus/silences HTTP/1.1
Host: example.com
```

状态代码

- ***200 OK - 确定。***
- ***400 Bad Request – Operation exception.请检查响应正文以了解详细信息。***
- ***401 未授权 - 未验证的访问.请首先登录。***
- ***403 Forbidden – Unauthorized access.请检查您的权限。***
- ***500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。***

GET /api/prometheus/notifications

示例

```
GET /api/prometheus/notifications HTTP/1.1
Host: example.com
```

状态代码

- ***200 OK - 确定。***
- ***400 Bad Request – Operation exception.请检查响应正文以了解详细信息。***
- ***401 未授权 - 未验证的访问.请首先登录。***

- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

其它资源

- *如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。*

A.21. RADOS 块设备

使用 Ceph RESTful API block 端点来管理 RADOS 块设备(RBD)的方法参考。此参考包含所有可用的 RBD 功能端点，例如：

- *RBD 命名空间*
- *RBD 快照*
- *rbd Trash*
- *RBD 镜像功能*
  - *RBD 镜像功能概述*
  - *RBD 镜像池 Bootstrap*
  - *RBD 镜像池模式*
  - *RBD 镜像池 Peer*

RBD 镜像

RBD 镜像

GET /api/block/image

描述

查看 **RBD** 镜像。

参数

- **queries:**
  - **pool_name - 池名称作为字符串。**

示例

```
GET /api/block/image HTTP/1.1
Host: example.com
```

状态代码

- **200 OK - 确定。**
- **400 Bad Request – Operation exception.请检查响应正文以了解详细信息。**
- **401 未授权 - 未验证的访问.请首先登录。**
- **403 Forbidden – Unauthorized access.请检查您的权限。**
- **500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。**

POST /api/block/image

示例

```
POST /api/block/image HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "configuration": "STRING",
  "data_pool": "STRING",
```

```
    "features": "STRING",
    "name": "STRING",
    "namespace": "STRING",
    "obj_size": 1,
    "pool_name": "STRING",
    "size": 1,
    "stripe_count": 1,
    "stripe_unit": "STRING"
}
```

状态代码

- **201 created - 资源已创建。**
- **202 accepted - 操作仍在执行。请检查任务队列。**
- **400 Bad Request – Operation exception.请检查响应正文以了解详细信息。**
- **401 未授权 - 未验证的访问.请首先登录。**
- **403 Forbidden – Unauthorized access.请检查您的权限。**
- **500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。**

GET /api/block/image/clone_format_version

描述

返回 RBD 克隆格式版本。

示例

```
GET /api/block/image/clone_format_version HTTP/1.1
Host: example.com
```

状态代码

- **200 OK - 确定。**

- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/block/image/default_features

示例

```
GET /api/block/image/default_features HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/block/image/default_features

示例

```
GET /api/block/image/default_features HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

DELETE /api/block/image/IMAGE_SPEC

参数

- *将 IMAGE_SPEC 替换为镜像名称，作为字符串值。*

状态代码

- *202 accepted - 操作仍在执行。请检查任务队列。*
- *204 No Content – Resource deleted.*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/block/image/IMAGE_SPEC

参数

- 将 IMAGE_SPEC 替换为镜像名称，作为字符串值。

示例

```
GET /api/block/image/IMAGE_SPEC HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

PUT /api/block/image/IMAGE_SPEC

参数

- 将 IMAGE_SPEC 替换为镜像名称，作为字符串值。

示例

```
PUT /api/block/image/IMAGE_SPEC HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "configuration": "STRING",
  "features": "STRING",
  "name": "STRING",
  "size": 1
}
```

状态代码

- *200 OK - 确定。*
- *202 accepted - 操作仍在执行。请检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

POST /api/block/image/IMAGE_SPEC/copy

参数

- *将 IMAGE_SPEC 替换为镜像名称，作为字符串值。*

示例

```
POST /api/block/image/IMAGE_SPEC/copy HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "configuration": "STRING",
  "data_pool": "STRING",
  "dest_image_name": "STRING",
  "dest_namespace": "STRING",
  "dest_pool_name": "STRING",
  "features": "STRING",
  "obj_size": 1,
  "snapshot_name": "STRING",
  "stripe_count": 1,
  "stripe_unit": "STRING"
}
```

状态代码

-

- *201 created - 资源已创建。*
- *202 accepted - 操作仍在执行。请检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

POST /api/block/image/IMAGE_SPEC/flatten

参数

- *将 IMAGE_SPEC 替换为镜像名称，作为字符串值。*

状态代码

- *201 created - 资源已创建。*
- *202 accepted - 操作仍在执行。请检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*

- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

POST /api/block/image/IMAGE_SPEC/move_trash

描述

将镜像移动到回收站中。克隆主动使用的镜像可以移到回收站中，并在以后删除。

参数

- *将 IMAGE_SPEC 替换为镜像名称，作为字符串值。*

示例

```
POST /api/block/image/IMAGE_SPEC/move_trash HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "delay": 1
}
```

状态代码

- *201 created - 资源已创建。*
- *202 accepted - 操作仍在执行。请检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

RBD 镜像功能

GET /api/block/mirroring/site_name

描述

*显示 **RBD** 镜像站点名称。*

示例

```
GET /api/block/mirroring/site_name HTTP/1.1
Host: example.com
```

状态代码

- ***200 OK - 确定。***
- ***400 Bad Request – Operation exception.请检查响应正文以了解详细信息。***
- ***401 未授权 - 未验证的访问.请首先登录。***
- ***403 Forbidden – Unauthorized access.请检查您的权限。***
- ***500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。***

PUT /api/block/mirroring/site_name

示例

```
PUT /api/block/mirroring/site_name HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "site_name": "STRING"
}
```

状态代码

- ***200 OK - 确定。***
- ***202 accepted - 操作仍在执行。请检查任务队列。***

- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

RBD 镜像池 Bootstrap

POST /api/block/mirroring/pool/POOL_NAME/bootstrap/peer

参数

- *将 POOL_NAME 替换为池名称作为字符串。*

示例

```
POST /api/block/mirroring/pool/POOL_NAME/bootstrap/peer HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "direction": "STRING",
  "token": "STRING"
}
```

状态代码

- *201 created - 资源已创建。*
- *202 accepted - 操作仍在执行。请检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*

- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

POST /api/block/mirroring/pool/POOL_NAME/bootstrap/token

参数

- *将 POOL_NAME 替换为池名称作为字符串。*

状态代码

- *201 created - 资源已创建。*
- *202 accepted - 操作仍在执行。请检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

RBD 镜像池模式

GET /api/block/mirroring/pool/POOL_NAME

描述

显示 RBD 镜像摘要。

参数

- *将 POOL_NAME 替换为池名称作为字符串。*

示例

```
GET /api/block/mirroring/pool/POOL_NAME HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

PUT /api/block/mirroring/pool/POOL_NAME

参数

- *将 POOL_NAME 替换为池名称作为字符串。*

示例

```
PUT /api/block/mirroring/pool/POOL_NAME HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "mirror_mode": "STRING"
}
```

状态代码

- *200 OK - 确定。*

- *202 accepted - 操作仍在执行。请检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

RBD 镜像池 Peer

GET /api/block/mirroring/pool/POOL_NAME/peer

参数

- *将 POOL_NAME 替换为池名称作为字符串。*

示例

```
GET /api/block/mirroring/pool/POOL_NAME/peer HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*

- ***500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。***

POST /api/block/mirroring/pool/POOL_NAME/peer

参数

- ***将 POOL_NAME 替换为池名称作为字符串。***

示例

```
POST /api/block/mirroring/pool/POOL_NAME/peer HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "client_id": "STRING",
  "cluster_name": "STRING",
  "key": "STRING",
  "mon_host": "STRING"
}
```

状态代码

- ***201 created - 资源已创建。***
- ***202 accepted - 操作仍在执行。请检查任务队列。***
- ***400 Bad Request – Operation exception.请检查响应正文以了解详细信息。***
- ***401 未授权 - 未验证的访问.请首先登录。***
- ***403 Forbidden – Unauthorized access.请检查您的权限。***
- ***500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。***

DELETE /api/block/mirroring/pool/POOL_NAME/peer/PEER_UUID

参数

- 将 POOL_NAME 替换为池名称作为字符串。
- 将 PEER_UUID 替换为字符串的对等 UUID。

状态代码

- 202 accepted - 操作仍在执行。请检查任务队列。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

GET /api/block/mirroring/pool/POOL_NAME/peer/PEER_UUID

参数

- 将 POOL_NAME 替换为池名称作为字符串。
- 将 PEER_UUID 替换为字符串的对等 UUID。

示例

```
GET /api/block/mirroring/pool/POOL_NAME/peer/PEER_UUID HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。

- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

PUT /api/block/mirroring/pool/POOL_NAME/peer/PEER_UUID

参数

- *将 POOL_NAME 替换为池名称作为字符串。*
- *将 PEER_UUID 替换为字符串的对等 UUID。*

示例

```
PUT /api/block/mirroring/pool/POOL_NAME/peer/PEER_UUID HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "client_id": "STRING",
  "cluster_name": "STRING",
  "key": "STRING",
  "mon_host": "STRING"
}
```

状态代码

- *200 OK - 确定。*
- *202 accepted - 操作仍在执行。请检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*

- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

RBD 镜像功能概述

GET /api/block/mirroring/summary

描述

显示 RBD 镜像摘要。

示例

```
GET /api/block/mirroring/summary HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

RBD 命名空间

GET /api/block/pool/POOL_NAME/namespace

参数

- *将 POOL_NAME 替换为池名称作为字符串。*

示例

```
GET /api/block/pool/POOL_NAME/namespace HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

POST /api/block/pool/POOL_NAME/namespace

参数

- *将 POOL_NAME 替换为池名称作为字符串。*

示例

```
POST /api/block/pool/POOL_NAME/namespace HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "namespace": "STRING"
}
```

状态代码

- *201 created - 资源已创建。*

- *202 accepted - 操作仍在执行。请检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

DELETE /api/block/pool/POOL_NAME/namespace/NAMESPACE

参数

- *将 POOL_NAME 替换为池名称作为字符串。*
- *将 NAMESPACE 替换为命名空间作为字符串。*

状态代码

- *202 accepted - 操作仍在执行。请检查任务队列。*
- *204 No Content – Resource deleted.*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*

- ***500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。***

RBD 快照

POST /api/block/image/IMAGE_SPEC/snap

参数

- ***将 IMAGE_SPEC 替换为镜像名称，作为字符串值。***

示例

```
POST /api/block/image/IMAGE_SPEC/snap HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "snapshot_name": "STRING"
}
```

状态代码

- ***201 created - 资源已创建。***
- ***202 accepted - 操作仍在执行。请检查任务队列。***
- ***400 Bad Request – Operation exception.请检查响应正文以了解详细信息。***
- ***401 未授权 - 未验证的访问.请首先登录。***
- ***403 Forbidden – Unauthorized access.请检查您的权限。***
- ***500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。***

DELETE /api/block/image/IMAGE_SPEC/snap/SNAPSHOT_NAME

参数

- 将 IMAGE_SPEC 替换为镜像名称，作为字符串值。
- 将 SNAPSHOT_NAME 替换为快照名称（一个字符串值）。

状态代码

- 202 accepted - 操作仍在执行。请检查任务队列。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

PUT /api/block/image/IMAGE_SPEC/snap/SNAPSHOT_NAME

参数

- 将 IMAGE_SPEC 替换为镜像名称，作为字符串值。
- 将 SNAPSHOT_NAME 替换为快照名称（一个字符串值）。

示例

```
PUT /api/block/image/IMAGE_SPEC/snap/SNAPSHOT_NAME HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "is_protected": true,
  "new_snap_name": "STRING"
}
```

状态代码

- *200 OK - 确定。*
- *202 accepted - 操作仍在执行。请检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

POST /api/block/image/IMAGE_SPEC/snap/SNAPSHOT_NAME/clone

描述

将快照克隆到镜像。

参数

- *将 IMAGE_SPEC 替换为镜像名称，作为字符串值。*
- *将 SNAPSHOT_NAME 替换为快照名称（一个字符串值）。*

示例

```
POST /api/block/image/IMAGE_SPEC/snap/SNAPSHOT_NAME/clone HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "child_image_name": "STRING",
  "child_namespace": "STRING",
  "child_pool_name": "STRING",
  "configuration": "STRING",
  "data_pool": "STRING",
  "features": "STRING",
```

```
    "obj_size": 1,
    "stripe_count": 1,
    "stripe_unit": "STRING"
}
```

状态代码

- 201 created - 资源已创建。
- 202 accepted - 操作仍在执行。请检查任务队列。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

POST /api/block/image/IMAGE_SPEC/snap/SNAPSHOT_NAME/rollback

参数

- 将 IMAGE_SPEC 替换为镜像名称，作为字符串值。
- 将 SNAPSHOT_NAME 替换为快照名称（一个字符串值）。

状态代码

- 201 created - 资源已创建。
- 202 accepted - 操作仍在执行。请检查任务队列。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。

- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

rbd Trash

GET /api/block/image/trash

描述

按池名称显示所有 RBD 回收条目，或者显示 RBD 回收详细信息。

参数

- *queries:*
  - *pool_name - 池的名称作为字符串值。*

示例

```
GET /api/block/image/trash HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*

- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

POST /api/block/image/trash/purge

描述

从回收站中删除所有已过期的镜像。

参数

- queries:
  - pool_name - 池的名称作为字符串值。

示例

```
POST /api/block/image/trash/purge HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "pool_name": "STRING"
}
```

状态代码

- 201 created - 资源已创建。
- 202 accepted - 操作仍在执行。请检查任务队列。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

DELETE /api/block/image/trash/IMAGE_ID_SPEC

描述

从回收站中删除镜像.如果镜像延迟时间没有过期，除非您使用 force，否则无法将其删除。通过克隆或具有快照来主动使用的镜像，无法删除它。

参数

- *将 IMAGE_ID_SPEC 替换为镜像名称，作为字符串值。*
- *queries:*
  - *force - 强制从垃圾中删除镜像的布尔值。*

状态代码

- *202 accepted - 操作仍在执行。请检查任务队列。*
- *204 No Content – Resource deleted.*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

POST /api/block/image/trash/IMAGE_ID_SPEC/restore

描述

从回收站中恢复镜像。

参数

- 将 IMAGE_ID_SPEC 替换为镜像名称，作为字符串值。

示例

```
POST /api/block/image/trash/IMAGE_ID_SPEC/restore HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "new_image_name": "STRING"
}
```

状态代码

- 201 created - 资源已创建。
- 202 accepted - 操作仍在执行。请检查任务队列。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

其它资源

- 如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。

A.22. 性能计数器

方法引用，使用 Ceph RESTful API perf_counters 端点显示各种 Ceph 性能计数器。此参考包括所有可用的性能计数器端点，例如：

- *Ceph 元数据服务器(MDS)*
- *Ceph Manager*
- *Ceph monitor*
- *Ceph OSD*
- *Ceph 对象网关*
- *Ceph RADOS 块设备(RBD)镜像功能*
- *TCMU Runner*

GET /api/perf_counters

描述

显示性能计数器。

示例

```
GET /api/perf_counters HTTP/1.1
Host: example.com
```

状态代码

- **200 OK - 确定。**
- **400 Bad Request – Operation exception.请检查响应正文以了解详细信息。**
- **401 未授权 - 未验证的访问.请首先登录。**

- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

Ceph 元数据服务器

GET /api/perf_counters/mds/SERVICE_ID

参数

- *将 SERVICE_ID 替换为所需的服务标识符作为字符串。*

示例

```
GET /api/perf_counters/mds/SERVICE_ID HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

Ceph Manager

GET /api/perf_counters/mgr/SERVICE_ID

参数

- *将 SERVICE_ID 替换为所需的服务标识符作为字符串。*

示例

```
GET /api/perf_counters/mgr/SERVICE_ID HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

Ceph monitor

GET /api/perf_counters/mon/SERVICE_ID

参数

- *将 SERVICE_ID 替换为所需的服务标识符作为字符串。*

示例

```
GET /api/perf_counters/mon/SERVICE_ID HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*

- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

Ceph OSD

GET /api/perf_counters/osd/SERVICE_ID

参数

- 将 SERVICE_ID 替换为所需的服务标识符作为字符串。

示例

```
GET /api/perf_counters/osd/SERVICE_ID HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

Ceph RADOS 块设备(RBD)镜像功能

GET /api/perf_counters/rbd-mirror/SERVICE_ID

参数

- **将 SERVICE_ID 替换为所需的服务标识符作为字符串。**

示例

```
GET /api/perf_counters/rbd-mirror/SERVICE_ID HTTP/1.1
Host: example.com
```

状态代码

- **200 OK - 确定。**
- **400 Bad Request – Operation exception.请检查响应正文以了解详细信息。**
- **401 未授权 - 未验证的访问.请首先登录。**
- **403 Forbidden – Unauthorized access.请检查您的权限。**
- **500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。**

Ceph 对象网关

GET /api/perf_counters/rgw/SERVICE_ID

参数

- **将 SERVICE_ID 替换为所需的服务标识符作为字符串。**

示例

```
GET /api/perf_counters/rgw/SERVICE_ID HTTP/1.1
Host: example.com
```

状态代码

- **200 OK - 确定。**

- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

TCMU Runner

GET /api/perf_counters/tcmu-runner/SERVICE_ID

参数

- *将 SERVICE_ID 替换为所需的服务标识符作为字符串。*

示例

```
GET /api/perf_counters/tcmu-runner/SERVICE_ID HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

其它资源

- *如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。*

A.23. 角色

使用 Ceph RESTful API role 端点来管理 Ceph 中各种用户角色的方法参考。

GET /api/role

描述

显示角色列表。

示例

```
GET /api/role HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

POST /api/role

示例

```
POST /api/role HTTP/1.1
Host: example.com
Content-Type: application/json

{
```

```
    "description": "STRING",
    "name": "STRING",
    "scopes_permissions": "STRING"
}
```

状态代码

- 201 created - 资源已创建。
- 202 accepted - 操作仍在执行。请检查任务队列。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

DELETE /api/role/NAME

参数

- 将 NAME 替换为字符串的角色名称。

状态代码

- 202 accepted - 操作仍在执行。请检查任务队列。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。

- ***403 Forbidden – Unauthorized access.请检查您的权限。***
- ***500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。***

GET /api/role/NAME

参数

- ***将 NAME 替换为字符串的角色名称。***

示例

```
GET /api/role/NAME HTTP/1.1
Host: example.com
```

状态代码

- ***200 OK - 确定。***
- ***400 Bad Request – Operation exception.请检查响应正文以了解详细信息。***
- ***401 未授权 - 未验证的访问.请首先登录。***
- ***403 Forbidden – Unauthorized access.请检查您的权限。***
- ***500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。***

PUT /api/role/NAME

参数

- ***将 NAME 替换为字符串的角色名称。***

示例

```
PUT /api/role/NAME HTTP/1.1
Host: example.com
```

```
Content-Type: application/json

{
   "description": "STRING",
   "scopes_permissions": "STRING"
}
```

状态代码

- **200 OK - 确定。**
- **202 accepted - 操作仍在执行。请检查任务队列。**
- **400 Bad Request – Operation exception.请检查响应正文以了解详细信息。**
- **401 未授权 - 未验证的访问.请首先登录。**
- **403 Forbidden – Unauthorized access.请检查您的权限。**
- **500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。**

POST /api/role/NAME/clone

参数

- **将 NAME 替换为字符串的角色名称。**

示例

```
POST /api/role/NAME/clone HTTP/1.1
Host: example.com
Content-Type: application/json

{
   "new_name": "STRING"
}
```

状态代码

- *201 created - 资源已创建。*
- *202 accepted - 操作仍在执行。请检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

其它资源

- *如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。*

A.24. 服务

方法引用，使用 Ceph RESTful API service 端点管理 Ceph 服务。

GET /api/service

参数

- *queries:*
  - *service_name - 作为字符串的服务名称。*

示例

```
GET /api/service HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

POST /api/service

参数

- service_spec - 服务规格作为 JSON 文件。
- service_name - 服务的名称。

示例

```
POST /api/service HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "service_name": "STRING",
  "service_spec": "STRING"
}
```

状态代码

- 201 created - 资源已创建。
- 202 accepted - 操作仍在执行。请检查任务队列。

- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/service/known_types

描述

显示已知服务类型列表。

示例

```
GET /api/service/known_types HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

DELETE /api/service/SERVICE_NAME

参数

- 将 SERVICE_NAME 替换为服务的名称作为字符串。

状态代码

- 202 accepted - 操作仍在执行。请检查任务队列。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

GET /api/service/SERVICE_NAME

参数

- 将 SERVICE_NAME 替换为服务的名称作为字符串。

示例

```
GET /api/service/SERVICE_NAME HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。

- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/service/SERVICE_NAME/daemons

参数

- *将 SERVICE_NAME 替换为服务的名称作为字符串。*

示例

```
GET /api/service/SERVICE_NAME/daemons HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

其它资源

- *如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。*

A.25. 设置

方法引用，使用 Ceph RESTful API settings 端点来管理各种 Ceph 设置。

GET /api/settings

描述

显示可用选项列表

参数

- queries:
  - names - 以逗号分隔的选项名称列表。

示例

```
GET /api/settings HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

PUT /api/settings

状态代码

- 200 OK - 确定。

- *202 accepted - 操作仍在执行。请检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

DELETE /api/settings/NAME

参数

- *使用选项名称替换 NAME 作为字符串。*

状态代码

- *202 accepted - 操作仍在执行。请检查任务队列。*
- *204 No Content – Resource deleted.*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/settings/NAME

描述

显示给定选项。

参数

- *使用选项名称替换 NAME 作为字符串。*

示例

```
GET /api/settings/NAME HTTP/1.1
Host: example.com
```

状态代码

- ***200 OK - 确定。***
- ***400 Bad Request – Operation exception.请检查响应正文以了解详细信息。***
- ***401 未授权 - 未验证的访问.请首先登录。***
- ***403 Forbidden – Unauthorized access.请检查您的权限。***
- ***500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。***

PUT /api/settings/NAME

参数

- *使用选项名称替换 NAME 作为字符串。*

示例

```
PUT /api/settings/NAME HTTP/1.1
Host: example.com
Content-Type: application/json
```

```
{
  "value": "STRING"
}
```

状态代码

- 200 OK - 确定。
- 202 accepted - 操作仍在执行。请检查任务队列。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

其它资源

- 如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。

A.26. CEPH 任务

方法引用，使用 Ceph RESTful API task 端点来显示 Ceph 任务。

GET /api/task

描述

显示 Ceph 任务。

参数

- queries:

- name - 任务的名称。

示例

```
GET /api/task HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

其它资源

- 如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。

A.27. TELEMETRY

方法引用，使用 Ceph RESTful API telemetry 端点管理遥测 Ceph Manager 模块的数据。

PUT /api/telemetry

描述

启用或禁用遥测模块发送所收集数据。

参数

- ***enable - 布尔值。***
- ***license_name - 字符串值，如 shared -1-0。确保用户已了解并接受共享遥测数据的许可证。***

示例

```
PUT /api/telemetry HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "enable": true,
  "license_name": "STRING"
}
```

状态代码

- ***200 OK - 确定。***
- ***202 accepted - 操作仍在执行。请检查任务队列。***
- ***400 Bad Request – Operation exception.请检查响应正文以了解详细信息。***
- ***401 未授权 - 未验证的访问.请首先登录。***
- ***403 Forbidden – Unauthorized access.请检查您的权限。***
- ***500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。***

GET /api/telemetry/report

描述

显示 Ceph 和设备上的报告数据。

示例

```
GET /api/telemetry/report HTTP/1.1
Host: example.com
```

状态代码

- 200 OK - 确定。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

其它资源

- 如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。
- 有关使用 Ceph 仪表板管理的详细信息，请参阅 Red Hat Ceph Storage 仪表板指南中的 激活和停用遥测 一章。

A.28. CEPH 用户

方法引用，使用 Ceph RESTful API user 端点显示 Ceph 用户详细信息，以及管理 Ceph 用户密码。

GET /api/user

描述

显示用户列表。

示例

```
GET /api/user HTTP/1.1
Host: example.com
```

状态代码

- *200 OK - 确定。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

POST /api/user

示例

```
POST /api/user HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "email": "STRING",
  "enabled": true,
  "name": "STRING",
  "password": "STRING",
  "pwdExpirationDate": "STRING",
  "pwdUpdateRequired": true,
  "roles": "STRING",
  "username": "STRING"
}
```

状态代码

- *201 created - 资源已创建。*
- *202 accepted - 操作仍在执行。请检查任务队列。*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*

- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

DELETE /api/user/USER_NAME

参数

- *将 USER_NAME 替换为用户的名称作为字符串。*

状态代码

- *202 accepted - 操作仍在执行。请检查任务队列。*
- *204 No Content – Resource deleted.*
- *400 Bad Request – Operation exception.请检查响应正文以了解详细信息。*
- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

GET /api/user/USER_NAME

参数

- *将 USER_NAME 替换为用户的名称作为字符串。*

示例

```
GET /api/user/USER_NAME HTTP/1.1
Host: example.com
```

状态代码

- **200 OK - 确定。**
- **400 Bad Request – Operation exception.请检查响应正文以了解详细信息。**
- **401 未授权 - 未验证的访问.请首先登录。**
- **403 Forbidden – Unauthorized access.请检查您的权限。**
- **500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。**

PUT /api/user/USER_NAME

参数

- **将 USER_NAME 替换为用户的名称作为字符串。**

示例

```
PUT /api/user/USER_NAME HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "email": "STRING",
  "enabled": "STRING",
  "name": "STRING",
  "password": "STRING",
  "pwdExpirationDate": "STRING",
  "pwdUpdateRequired": true,
  "roles": "STRING"
}
```

状态代码

- **200 OK - 确定。**

- 202 accepted - 操作仍在执行。请检查任务队列。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。
- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

POST /api/user/USER_NAME/change_password

参数

- 将 USER_NAME 替换为用户的名称作为字符串。

示例

```
POST /api/user/USER_NAME/change_password HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "new_password": "STRING",
  "old_password": "STRING"
}
```

状态代码

- 201 created - 资源已创建。
- 202 accepted - 操作仍在执行。请检查任务队列。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。
- 401 未授权 - 未验证的访问.请首先登录。

- 403 Forbidden – Unauthorized access.请检查您的权限。
- 500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。

POST /api/user/validate_password

描述

检查密码以查看它是否满足密码策略。

参数

- Password - 用于验证的密码。
- 用户名 - 可选。用户名称。
- old_password - 可选。旧密码。

示例

```
POST /api/user/validate_password HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "old_password": "STRING",
  "password": "STRING",
  "username": "STRING"
}
```

状态代码

- 201 created - 资源已创建。
- 202 accepted - 操作仍在执行。请检查任务队列。
- 400 Bad Request – Operation exception.请检查响应正文以了解详细信息。

- *401 未授权 - 未验证的访问.请首先登录。*
- *403 Forbidden – Unauthorized access.请检查您的权限。*
- *500 Internal Server Error – 意外错误。请检查堆栈追踪的响应正文。*

其它资源

- *如需了解更多详细信息，请参阅 Red Hat Ceph Storage Developer Guide 中的 Ceph RESTful API 章节。*

附录 B. S3 通用请求标头

下表列出了有效的通用请求标头及其描述。

表 B.1. 请求 Headers

请求标头	描述
CONTENT_LENGTH	请求正文的长度。
DATE	请求时间和日期（以 UTC 为单位）。
HOST	主机服务器的名称。
授权	身份验证令牌。

附录 C. S3 通用响应状态代码

下表列出了有效的通用 HTTP 响应状态及其相应的代码。

表 C.1. 响应状态

HTTP 状态	响应代码
100	继续
200	成功
201	已创建
202	已接受
204	NoContent
206	部分内容
304	NotModified
400	InvalidArgument
400	InvalidDigest
400	BadDigest
400	InvalidBucketName
400	InvalidObjectName
400	UnresolvableGrantByEmailAddress
400	InvalidPart
400	InvalidPartOrder
400	RequestTimeout
400	EntityTooLarge
403	AccessDenied
403	UserSuspended

HTTP 状态	响应代码
403	RequestTimeTooSkewed
404	NoSuchKey
404	NoSuchBucket
404	NoSuchUpload
405	MethodNotAllowed
408	RequestTimeout
409	BucketAlreadyExists
409	BucketNotEmpty
411	MissingContentLength
412	PreconditionFailed
416	InvalidRange
422	UnprocessableEntity
500	InternalError

附录 D. S3 支持和不支持的操作操作

此信息列出了最新支持和不支持的 S3 操作动词。

表 D.1. 支持的操作动词

操作	API
AbortMultipartUpload	S3
CompleteMultipartUpload	S3
CopyObject	S3
CreateBucket	S3
CreateMultipartUpload	S3
DeleteBucket	S3
DeleteBucketCORS	S3
DeleteBucketEncryption	S3
DeleteBucketLifecycle	S3
DeleteBucketPolicy	S3
DeleteBucketReplication	S3
DeleteBucketTagging	S3
DeleteBucketWebsite	S3
DeleteObject	S3
DeleteObjects	S3
DeleteObjectTagging	S3
GetBucketAcl	S3
GetBucketCORS	S3
GetBucketEncryption	S3

操作	API
GetBucketLocation	S3
GetBucketNotificationConfiguration	S3
GetBucketPolicy	S3
GetBucketPolicyStatus	S3
GetBucketReplication	S3
GetBucketRequestPayment	S3
GetBucketTagging	S3
GetBucketVersioning	S3
GetBucketWebsite	S3
GetObject	S3
GetObjectAcl	S3
GetObjectAttributes	S3
GetObjectLegalHold	S3
GetObjectLockConfiguration	S3
GetObjectRetention	S3
GetObjectTagging	S3
GetObjectTorrent	S3
HeadBucket	S3
HeadObject	S3
listBuckets	S3
ListMultipartUploads	S3
ListObjects	S3

操作	API
ListObjectsV2	S3
ListObjectVersions	S3
ListParts	S3
PutBucketAcl	S3
PutBucketCORS	S3
PutBucketEncryption	S3
PutBucketLifecycle	S3
PutBucketLifecycleConfiguration	S3
PutBucketNotificationConfiguration	S3
PutBucketPolicy	S3
PutBucketReplication	S3
PutBucketRequestPayment	S3
PutBucketTagging	S3
PutBucketVersioning	S3
PutBucketWebsite	S3
PutObject	S3
PutObjectAcl	S3
PutObjectLegalHold	S3
PutObjectLockConfiguration	S3
PutObjectRetention	S3
PutObjectTagging	S3
SelectObjectContent	S3

操作	API
UploadPart	S3
UploadPartCopy	S3
AssumeRole	STS
AssumeRoleWithWebIdentity	STS
GetSessionToken	STS
CreateOpenIDConnectProvider	IAM
CreateRole	IAM
DeleteOpenIDConnectProvider	IAM
DeleteRole	IAM
GetOpenIDConnectProvider	IAM
GetRole	IAM
ListRoles	IAM
DeleteBucketNotification	S3 扩展
CreateTopic	SNS
GetTopicAttributes	SNS
DeleteTopic	SNS
ListTopics	SNS

表 D.2. 不支持的操作操作

操作	API
DeleteBucketInventoryConfiguration	S3
DeleteIntelligentTieringConfiguration	S3
GetBucketInventoryConfiguration	S3

操作	API
GetBucketLogging	S3
GetIntelligentTieringConfiguration	S3
ListBucketIntelligentTieringConfigurations	S3
ListBucketInventoryConfigurations	S3
PutBucketIntelligentTieringConfiguration	S3
PutBucketInventoryConfiguration	S3
PutBucketLogging	S3
RestoreObject	S3
AssumeRoleWithSAML	STS
DecodeAuthorizationMessage	STS
GetAccessKeyInfo	STS
GetCallerIdentity	STS
GetFederationToken	STS
AttachGroupPolicy	IAM
AttachRolePolicy	IAM
AttachUserPolicy	IAM
CreatePolicy	IAM
CreateSAMLProvider	IAM
DeletePolicy	IAM
DeleteSAMLProviders	IAM
GetPolicy	IAM
GetSAMLProviders	IAM
ListOpenIDConnectProviders	IAM

操作	API
ListPolicies	IAM
ListSAMLProviders	IAM

附录 E. S3 不支持的标头字段

表 E.1. 不支持的标题字段

名称	Type
x-amz-security-token	Request（请求）
Server	响应
x-amz-delete-marker	响应
x-amz-id-2	响应
x-amz-request-id	响应
x-amz-version-id	响应

附录 F. SWIFT 请求标头

表 F.1. 请求 Headers

名称	描述	Type	必需
X-Auth-User	进行身份验证的主要 Ceph 对象网关用户名。	字符串	是
x-Auth-Key	与 Ceph 对象网关用户名关联的密钥。	字符串	是

附录 G. SWIFT 响应标头

服务器的响应应包含 X-Auth-Token 值。响应可能还包含 X-Storage-Url，它提供 API_VERSION/ACCOUNT 前缀，该前缀是在 API 文档中其他请求中指定的。

表 G.1. 响应标头

名称	描述	Type
x-Storage-Token	请求中指定的 **X-Auth-User** 的授权令牌。	字符串
X-Storage-Url	用户的 URL 和 ***API_VERSION/ACCOUNT*** 路径。	字符串

附录 H. 使用安全令牌服务 API 的示例

这些示例使用 Python 的 boto3 模块与 Ceph 对象网关安全令牌服务(STS)的接口。在这些示例中，TESTER2 假设一个由 TESTER1 创建的角色，因为根据附加到该角色的权限策略访问由 TESTER1 拥有的 S3 资源。

AssumeRole 示例创建一个角色，为角色分配一个策略，然后假定一个角色获取临时凭证，并使用这些临时凭证访问 S3 资源。

AssumeRoleWithWebIdentity 示例使用带有 Keycloak 的外部应用程序（OpenID Connect 身份提供程序）验证用户，它假定一个角色获取临时凭证，并根据角色的权限策略访问 S3 资源。

AssumeRole 示例

```
import boto3

iam_client = boto3.client('iam',
aws_access_key_id=ACCESS_KEY_OF_TESTER1,
aws_secret_access_key=SECRET_KEY_OF_TESTER1,
endpoint_url=<IAM URL>,
region_name=''
)

policy_document = "{\"Version\":\"2012-10-17\",\"Statement\":
[{\"Effect\":\"Allow\",\"Principal\":{\"AWS\":[\"arn:aws:iam:::user/TESTER1\"]},\"Action\":
[\"sts:AssumeRole\"]}]}"

role_response = iam_client.create_role(
AssumeRolePolicyDocument=policy_document,
Path='/',
RoleName='S3Access',
)

role_policy = "{\"Version\":\"2012-10-17\",\"Statement\":
{\"Effect\":\"Allow\",\"Action\":\"s3:*\",\"Resource\":\"arn:aws:s3:::*\"}}"

response = iam_client.put_role_policy(
RoleName='S3Access',
PolicyName='Policy1',
PolicyDocument=role_policy
)

sts_client = boto3.client('sts',
aws_access_key_id=ACCESS_KEY_OF_TESTER2,
aws_secret_access_key=SECRET_KEY_OF_TESTER2,
endpoint_url=<STS URL>,
region_name='',
```

```
)

response = sts_client.assume_role(
RoleArn=role_response['Role']['Arn'],
RoleSessionName='Bob',
DurationSeconds=3600
)

s3client = boto3.client('s3',
aws_access_key_id = response['Credentials']['AccessKeyId'],
aws_secret_access_key = response['Credentials']['SecretAccessKey'],
aws_session_token = response['Credentials']['SessionToken'],
endpoint_url=<S3 URL>,
region_name='',)

bucket_name = 'my-bucket'
s3bucket = s3client.create_bucket(Bucket=bucket_name)
resp = s3client.list_buckets()
```

AssumeRoleWithWebIdentity 示例

```
import boto3

iam_client = boto3.client('iam',
aws_access_key_id=ACCESS_KEY_OF_TESTER1,
aws_secret_access_key=SECRET_KEY_OF_TESTER1,
endpoint_url=<IAM URL>,
region_name=''
)

oidc_response = iam_client.create_open_id_connect_provider(
   Url=<URL of the OpenID Connect Provider>,
   ClientIDList=[
      <Client id registered with the IDP>
   ],
   ThumbprintList=[
      <IDP THUMBPRINT>
 ]
)

policy_document = "{\"Version\":\"2012-10-17\",\"Statement\":\[\
{\"Effect\":\"Allow\",\"Principal\":\{\"Federated\":\[\"arn:aws:iam:::oidc-
provider/localhost:8080/auth/realms/demo\"\]\},\"Action\":\
[\"sts:AssumeRoleWithWebIdentity\"\],\"Condition\":\{\"StringEquals\":\
{\"localhost:8080/auth/realms/demo:app_id\":\"customer-portal\"\}\}\}\]\}"
role_response = iam_client.create_role(
AssumeRolePolicyDocument=policy_document,
Path='/',
RoleName='S3Access',
```

```
)

role_policy = "{\"Version\":\"2012-10-17\",\"Statement\":
{\"Effect\":\"Allow\",\"Action\":\"s3:*\",\"Resource\":\"arn:aws:s3:::*\"}}"

response = iam_client.put_role_policy(
    RoleName='S3Access',
    PolicyName='Policy1',
    PolicyDocument=role_policy
)

sts_client = boto3.client('sts',
aws_access_key_id=ACCESS_KEY_OF_TESTER2,
aws_secret_access_key=SECRET_KEY_OF_TESTER2,
endpoint_url=<STS URL>,
region_name='',
)

response = sts_client.assume_role_with_web_identity(
RoleArn=role_response['Role']['Arn'],
RoleSessionName='Bob',
DurationSeconds=3600,
WebIdentityToken=<Web Token>
)

s3client = boto3.client('s3',
aws_access_key_id = response['Credentials']['AccessKeyId'],
aws_secret_access_key = response['Credentials']['SecretAccessKey'],
aws_session_token = response['Credentials']['SessionToken'],
endpoint_url=<S3 URL>,
region_name='',)

bucket_name = 'my-bucket'
s3bucket = s3client.create_bucket(Bucket=bucket_name)
resp = s3client.list_buckets()
```

其它资源

- *有关使用 Python 的 boto 模块的更多详细信息，请参阅 Red Hat Ceph Storage Object Gateway Configuration and Administration Guide 中的 Test S3 Access 部分。*