Red Hat Ceph Storage 4

開発者ガイド

Red Hat Ceph Storage の各種アプリケーションプログラミングインターフェイスの使用

Last Updated: 2023-08-23

Red Hat Ceph Storage 4 開発者ガイド

Red Hat Ceph Storage の各種アプリケーションプログラミングインターフェイスの使用

法律上の通知

概要

本ガイドでは、AMD64 および Intel 64 のアーキテクチャーで実行している Red Hat Ceph Storage のさまざまなアプリケーションプログラミングインターフェイスを使用する方法を説明します。Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、弊社の CTO、Chris Wright のメッセージ を参照してください。

目次

第1章 CEPH OBJECT GATEWAY 管理 API

開発者は、RESTful アプリケーションプログラムインターフェイス (API) と対話して Ceph Object Gateway を管理することができます。Ceph Object Gateway は、RESTful API の **radosgw-admin** コマンドの機能を利用できます。他の管理プラットフォームと統合できるユーザー、データ、クォータ、および使用方法を管理できます。

注記

Red Hat では、Ceph Object Gateway の設定時にコマンドラインインターフェイスを使用することを推奨します。

管理 API は以下の機能を提供します。

- 認証要求
- **ユーザーアカウントの管理**
  - 管理ユーザー
  - ユーザー情報の取得
  - 作成
  - 修正
  - 削除
  - サブユーザーの作成
  - サブユーザーの変更
  - サブユーザーの削除
- **ユーザーのケーパビリティー管理**
  - 追加
  - 削除
- **キー管理**
  - 作成
  - 削除
- **バケット管理**
  - バケット情報の取得
  - インデックスの確認
  - 削除
  - リンク
  - リンク解除

    - ポリシー
- **オブジェクト管理**
    - 削除
    - ポリシー
- **クォータ管理**
    - ユーザーの取得
    - ユーザーの設定
    - バケットの取得
    - バケットの設定
    - 個々のバケットのクォータの設定
- **使用方法の取得**
- **使用方法情報の削除**
- **標準エラーレスポンス**

1.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- RESTful クライアント。

1.2. 管理操作

管理アプリケーションプログラミングインターフェイス (API) リクエストは、設定可能な管理者リソースエントリーポイントで開始する URI で実行されます。管理 API の認可は S3 認可メカニズムを複製します。一部の操作では、ユーザーが特別な管理機能を保持する必要があります。XML または JSON のいずれかのレスポンスエンティティータイプはリクエストの format オプションとして指定され、指定されていないとデフォルトは JSON に設定されます。

例

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

usage=read
```

1.3. 管理認証要求

Amazon の S3 サービスは、アクセスキー、要求ヘッダーのハッシュ、および秘密鍵を使用して要求を認証します。これは、認証されたリクエスト (特に SSL オーバーヘッドのない大規模なアップロード) を提供する利点があります。

S3 API のほとんどのユースケースには、Java や Python Boto 用の Amazon SDK の **AmazonS3Client** などのオープンソースの S3 クライアントを使用します。これらのライブラリーは、Ceph Object Gateway 管理 API をサポートしません。これらのライブラリーをサブクラス化および拡張して、Ceph Admin API をサポートすることができます。一意のゲートウェイクライアントを作成できます。

execute() メソッドの作成

本セクションの CephAdminAPI のサンプルクラスでは、要求パラメーターの取得、リクエストの認証、Ceph 管理 API を呼び出してレスポンスを受け取ることができる **execute()** メソッドの作成方法を説明します。

CephAdminAPI クラスの例は、商用としてはサポートされず、そのように意図されてもいません。これは説明のみを目的としています。

Ceph Object Gateway の呼び出し

クライアントコード には、CRUD 操作を示すために Ceph Object Gateway への 5 つの呼び出しが含まれます。

- ユーザーの作成
- ユーザーの取得
- ユーザーの変更
- サブユーザーの作成
- ユーザーを削除します。

この例を使用するには、**httpcomponents-client-4.5.3** Apache HTTP コンポーネントを取得します。たとえば、http://hc.apache.org/downloads.cgi からダウンロードできます。その後、tar ファイルを展開して **lib** ディレクトリーに移動し、**JAVA_HOME** ディレクトリーの **/jre/lib/ext** ディレクトリーまたはカスタムクラスパスにコピーします。

CephAdminAPI クラスの例を検査する際に、**execute()** メソッドは HTTP メソッド、リクエストパス、オプションのサブリソース、未指定の場合は **null**、およびパラメーターのマップを取得することに注意してください。サブリソース (例: **subuser**、**key** など) で実行するには、サブリソースを **execute()** メソッドの引数として指定する必要があります。

方法の例を以下に示します。

1. URI をビルドします。
2. HTTP ヘッダー文字列をビルドします。
3. HTTP リクエストをインスタンス化します (例: **PUT**、**POST**、**GET**、**DELETE**)。
4. **Date** ヘッダーを HTTP ヘッダー文字列および要求ヘッダーに追加します。
5. **Authorization** ヘッダーを HTTP リクエストヘッダーに追加します。
6. HTTP クライアントをインスタンス化し、インスタンス化された HTTP リクエストを渡します。
7. 要求を行います。
8. レスポンスを返します。

ヘッダー文字列のビルド

ヘッダー文字列のビルドは、Amazon の S3 認証手順を伴うプロセスの一部です。特に、サンプルメソッドは以下を行います。

1. **PUT**、**POST**、**GET**、**DELETE** などのリスエストタイプを追加します。
2. 日付を追加します。
3. requestPath を追加します。

リクエストタイプは、先頭または最後の空白のない大文字である必要があります。空白を削除しないと、認証は失敗します。日付は GMT で表現される必要があります。さもないと、認証に失敗します。

例示的な方法には、他のヘッダーはありません。Amazon S3 認証手順は、**x-amz** ヘッダーの辞書式に並べ替えられます。したがって、**x-amz** ヘッダーを追加する場合は、必ず辞書式で追加する必要があります。

ヘッダー文字列をビルドしたら、次の手順は HTTP リクエストをインスタンス化し、URI を渡すことです。典型的なメソッドは、**PUT** を使用してユーザーおよびサブユーザーを作成し、**GET** を使用してユーザーを取得し、**POST** を使用してユーザーを変更し、**DELETE** を使用してユーザーを削除します。

リクエストをインスタンス化したら、**Date** ヘッダーに続けて **Authorization** ヘッダーを追加します。Amazon の S3 認証は標準の **Authorization** ヘッダーを使用し、以下の構造を持ちます。

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

CephAdminAPI のサンプルクラスには **base64Sha1Hmac()** メソッドがあります。これはヘッダー文字列と admin ユーザーの秘密鍵を取得し、SHA1 HMAC を base-64 でエンコードされた文字列として返します。それぞれの **execute()** 呼び出しは、同じコード行を呼び出して **Authorization** ヘッダーをビルドします。

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

以下の **CephAdminAPI** のサンプルクラスでは、アクセスキー、シークレットキー、およびエンドポイントをコンストラクターに渡す必要があります。クラスは実行時に変更するためのアクセスメソッドを提供します。

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);
 }
}

}
```

後続の **CephAdminAPIClient** の例は、**CephAdminAPI** クラスをインスタンス化する方法、リクエストパラメーターのマップをビルドし、**execute()** メソッドを使用してユーザーを作成、取得、更新、および削除する方法を示しています。

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();
 }
 }
}
```

関連情報

- 詳細は、**Red Hat Ceph Storage 開発者ガイド**の **S3 認証** セクションを参照してください。
- Amazon S3 認証手順の詳細は、Amazon Simple Storage Service ドキュメントの Signing and Authenticating REST Requests セクションを参照してください。

1.4. 管理ユーザーの作成

重要

Ceph Object Gateway ノードから **radosgw-admin** コマンドを実行するには、ノードに admin キーがあることを確認します。admin キーは、任意の Ceph Monitor ノードからコピーできます。

前提条件

- Ceph Object Gateway ノードへのルートレベルのアクセス。

手順

1. Object Gateway ユーザーを作成します。

   構文

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

   例

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

   radosgw-admin コマンドラインインターフェイスはユーザーを返します。

   出力例

   ```
   {
       "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": [],
       "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. 作成するユーザーに管理ケイパビリティーを割り当てます。

   構文

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

例

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

radosgw-admin コマンドラインインターフェイスはユーザーを返します。**"caps":** には、ユーザーに割り当てられたケイパビリティーがあります。

出力例

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

これで、管理者権限を持つユーザーが作成されます。

1.5. ユーザー情報の取得

ユーザーの情報の取得

機能

```
users=read
```

構文

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

表1.1 リクエストパラメーター

Name	説明	型	例	必須
uid	情報が要求されるユーザー。	文字列	**foo_user**	はい

表1.2 レスポンスエンティティー

Name	説明	タイプ	親
user	ユーザーデータ情報のコンテナー	コンテナー	該当なし
user_id	ユーザー ID。	文字列	**user**
display_name	ユーザーの表示名。	文字列	**user**
suspended	ユーザーが一時停止した場合は True。	ブール値	**user**
max_buckets	ユーザーが所有するバケットの最大数。	整数	**user**
subusers	このユーザーアカウントに関連付けられたサブユーザー。	コンテナー	**user**
keys	このユーザーアカウントに関連付けられた S3 キー。	コンテナー	**user**
swift_keys	このユーザーアカウントに関連付けられた Swift 鍵。	コンテナー	**user**
caps	ユーザーケイパビリティー。	コンテナー	**user**

成功すると、応答にはユーザー情報が含まれます。

特別なエラーレスポンス

なし。

1.6. ユーザーの作成

新しいユーザーを作成します。デフォルトでは、S3 キーペアが自動的に作成され、レスポンスで返されます。**access-key** または **secret-key** のいずれかのみを指定すると、省略キーが自動的に生成されます。デフォルトでは、生成されたキーは、既存のキーペアを置き換えることなくキーリングに追加され

ます。**access-key** が指定され、ユーザーが所有する既存のキーを参照すると、そのキーは変更されます。

機能

```
`users=write`
```

構文

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

表1.3 リクエストパラメーター

Name	説明	型	例	必須
uid	作成されるユーザー ID。	文字列	**foo_user**	はい
display-name	作成するユーザーの表示名。	文字列	**foo ユーザー**	はい
email	ユーザーに関連付けられたメールアドレス。	文字列	**foo@bar.com**	いいえ
key-type	生成されるキータイプ。オプションは swift、s3 (デフォルト) です。	文字列	**s3** [**s3**]	いいえ
access-key	アクセスキーを指定します。	文字列	**ABCD0EF12GHIJ2K34LMN**	いいえ
secret-key	シークレットキーを指定します。	文字列	**0AbCDEFg1h2i34JkIM5nop6QrSTUV+WxyzaBC7D8**	いいえ
user-caps	ユーザーケイパビリティー。	文字列	**usage=read, write; users=read**	いいえ
generate-key	新しいキーペアを生成し、既存のキーリングに追加します。	ブール値	True [True]	いいえ
max-buckets	ユーザーが所有できるバケットの最大数を指定します。	整数	500 [1000]	いいえ
suspended	ユーザーが一時停止されるかどうかを指定します。	ブール値	False [False]	No

表1.4 レスポンスエンティティー

Name	説明	タイプ	親
user	ユーザーデータ情報のコンテナー	コンテナー	該当なし
user_id	ユーザー ID。	文字列	**user**
display_name	ユーザーの表示名。	文字列	**user**
suspended	ユーザーが一時停止した場合は True。	ブール値	**user**
max_buckets	ユーザーが所有するバケットの最大数。	整数	**user**
subusers	このユーザーアカウントに関連付けられたサブユーザー。	コンテナー	**user**
keys	このユーザーアカウントに関連付けられた S3 キー。	コンテナー	**user**
swift_keys	このユーザーアカウントに関連付けられた Swift 鍵。	コンテナー	**user**
caps	ユーザーケイパビリティー。	コンテナー	**user**

成功すると、応答にはユーザー情報が含まれます。

表1.5 特別なエラーレスポンス

Name	説明	コード
UserExists	既存のユーザーの作成を試行。	409 Conflict
InvalidAccessKey	無効なアクセスキーが指定されている。	400 Bad Request
InvalidKeyType	無効なキータイプが指定されている。	400 Bad Request
InvalidSecretKey	無効なシークレットキーが指定されている。	400 Bad Request
InvalidKeyType	無効なキータイプが指定されている。	400 Bad Request
KeyExists	提供されたアクセスキーが存在し、別のユーザーに属している。	409 Conflict
EmailExists	提供されるメールアドレスが存在する。	409 Conflict
InvalidCap	無効な管理者ケイパビリティーの付与を試行。	400 Bad Request

関連情報

- サブユーザーの作成は、Red Hat Ceph Storage 開発者ガイドを参照してください。

1.7. ユーザーの変更

既存ユーザーの変更

機能

```
`users=write`
```

構文

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

表1.6 リクエストパラメーター

Name	説明	型	例	必須
uid	変更するユーザー ID。	文字列	**foo_user**	はい
display-name	変更されるユーザーの表示名。	文字列	**foo ユーザー**	いいえ
email	ユーザーに関連付けるメールアドレス。	文字列	**foo@bar.com**	いいえ
generate-key	新しいキーペアを生成し、既存のキーリングに追加します。	ブール値	True [False]	いいえ
access-key	アクセスキーを指定します。	文字列	**ABCD0EF12GHIJ2K34LMN**	いいえ
secret-key	シークレットキーを指定します。	文字列	**0AbCDEFg1h2i34JkIM5nop6QrSTUV+WxyzaBC7D8**	いいえ
key-type	生成されるキータイプ。オプションは swift、s3 (デフォルト) です。	文字列	**s3**	いいえ
user-caps	ユーザーケイパビリティー。	文字列	**usage=read, write; users=read**	いいえ
max-buckets	ユーザーが所有できるバケットの最大数を指定します。	整数	500 [1000]	いいえ

Name	説明	型	例	必須
suspended	ユーザーが一時停止されるかどうかを指定します。	ブール値	False [False]	No

表1.7 レスポンスエンティティー

Name	説明	タイプ	親
user	ユーザーデータ情報のコンテナー	コンテナー	該当なし
user_id	ユーザー ID。	文字列	**user**
display_name	ユーザーの表示名。	文字列	**user**
suspended	ユーザーが一時停止した場合は True。	ブール値	**user**
max_buckets	ユーザーが所有するバケットの最大数。	整数	**user**
subusers	このユーザーアカウントに関連付けられたサブユーザー。	コンテナー	**user**
keys	このユーザーアカウントに関連付けられた S3 キー。	コンテナー	**user**
swift_keys	このユーザーアカウントに関連付けられた Swift 鍵。	コンテナー	**user**
caps	ユーザーケイパビリティー。	コンテナー	**user**

成功すると、応答にはユーザー情報が含まれます。

表1.8 特別なエラーレスポンス

Name	説明	コード
InvalidAccessKey	無効なアクセスキーが指定されている。	400 Bad Request
InvalidKeyType	無効なキータイプが指定されている。	400 Bad Request
InvalidSecretKey	無効なシークレットキーが指定されている。	400 Bad Request
KeyExists	提供されたアクセスキーが存在し、別のユーザーに属している。	409 Conflict
EmailExists	提供されるメールアドレスが存在する。	409 Conflict

Name	説明	コード
InvalidCap	無効な管理者ケイパビリティーの付与を試行。	400 Bad Request

関連情報

- サブユーザーの変更については、Red Hat Ceph Storage 開発者ガイドを参照してください。

1.8. ユーザーの削除

既存のユーザーを削除します。

機能

```
`users=write`
```

構文

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

表1.9 リクエストパラメーター

Name	説明	型	例	必須
uid	削除するユーザー ID。	文字列	**foo_user**	はい。
purge-data	指定すると、ユーザーに属するバケットとオブジェクトも削除されます。	ブール値	True	No

レスポンスエンティティー

なし。

特別なエラーレスポンス

なし。

関連情報

- サブユーザーの削除については、Red Hat Ceph Storage 開発者ガイドを参照してください。

1.9. サブユーザーの作成

Swift API を使用するクライアントに主に役立つ新しいサブユーザーを作成します。

注記

有効なリクエストには、**gen-subuser** または **subuser** のいずれかが必要です。通常、サブユーザーには、**access** を指定してパーミッションを付与する必要があります。ユーザー作成 (**subuser** が **secret** なしで指定されている場合) と同様に、シークレットキーは自動的に生成されます。

機能

```
`users=write`
```

構文

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

表1.10 リクエストパラメーター

Name	説明	型	例	必須
uid	サブユーザーを作成するユーザー ID。	文字列	**foo_user**	はい
subuser	作成するサブユーザー ID を指定します。	文字列	**sub_foo**	必須 (または **gen-subuser**)
gen-subuser	作成するサブユーザー ID を指定します。	文字列	**sub_foo**	はい (または **subuser**)
secret-key	シークレットキーを指定します。	文字列	**0AbCDEFg1h2i34JklM5nop6QrSTUVWxyzaBC7D8**	いいえ
key-type	生成されるキータイプ。オプションは swift (デフォルト)、s3 です。	文字列	**swift** [**swift**]	いいえ
access	サブユーザーのアクセスパーミッションを設定する場合は、**read, write, readwrite, full** のいずれかである必要があります。	文字列	**read**	いいえ

Name	説明	型	例	必須
generate-secret	秘密鍵を生成します。	ブール値	True [False]	No

表1.11 レスポンスエンティティー

Name	説明	タイプ	親
subusers	ユーザーアカウントに関連付けられたサブユーザー	コンテナー	該当なし
id	サブユーザー ID。	文字列	**subusers**
permissions	ユーザーアカウントへのサブユーザーアクセス	文字列	**subusers**

成功すると、レスポンスにはサブユーザー情報が含まれます。

表1.12 特別なエラーレスポンス

Name	説明	コード
SubuserExists	指定したサブユーザーが存在する。	409 Conflict
InvalidKeyType	無効なキータイプが指定されている。	400 Bad Request
InvalidSecretKey	無効なシークレットキーが指定されている。	400 Bad Request
InvalidAccess	無効なサブユーザーアクセスが指定されている。	400 Bad Request

1.10. サブユーザーの変更

既存のサブユーザーを変更します。

機能

> `users=write`

構文

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

表1.13 リクエストパラメーター

Name	説明	型	例	必須
uid	サブユーザーを変更するユーザー ID。	文字列	**foo_user**	はい
subuser	変更するサブユーザー ID。	文字列	**sub_foo**	はい
generate-secret	サブユーザーの新しい秘密鍵を生成し、既存のキーを置き換えます。	ブール値	True [False]	いいえ
secret	シークレットキーを指定します。	文字列	**0AbCDEFg1h2i34JkIM5nop6QrSTUV+WxyzaBC7D8**	いいえ
key-type	生成されるキータイプ。オプションは swift (デフォルト)、s3 です。	文字列	**swift** [**swift**]	いいえ
access	サブユーザーのアクセスパーミッションを設定する場合は、**read, write, readwrite, full** のいずれかである必要があります。	文字列	**read**	いいえ

表1.14 レスポンスエンティティー

Name	説明	タイプ	親
subusers	ユーザーアカウントに関連付けられたサブユーザー	コンテナー	該当なし
id	サブユーザー ID。	文字列	**subusers**
permissions	ユーザーアカウントへのサブユーザーアクセス	文字列	**subusers**

成功すると、レスポンスにはサブユーザー情報が含まれます。

表1.15 特別なエラーレスポンス

Name	説明	コード
InvalidKeyType	無効なキータイプが指定されている。	400 Bad Request

Name	説明	コード
InvalidSecretKey	無効なシークレットキーが指定されている。	400 Bad Request
InvalidAccess	無効なサブユーザーアクセスが指定されている。	400 Bad Request

1.11. サブユーザーの削除

既存のサブユーザーを削除します。

機能

> `users=write`

構文

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

表1.16 リクエストパラメーター

Name	説明	型	例	必須
uid	サブユーザーを削除するユーザー ID。	文字列	**foo_user**	はい
subuser	削除されるサブユーザー ID。	文字列	**sub_foo**	はい
purge-keys	サブユーザーに属する鍵を削除します。	ブール値	True [True]	No

レスポンスエンティティー

なし。

特別なエラーレスポンス

なし。

1.12. ユーザーへの機能の追加

指定したユーザーに管理ケイパビリティーを追加します。

機能

> `users=write`

構文

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

表1.17 リクエストパラメーター

Name	説明	型	例	必須
uid	管理機能を追加するユーザー ID。	文字列	**foo_user**	はい
user-caps	ユーザーに追加する管理キャパシティー。	文字列	**usage=read, write**	はい

表1.18 レスポンスエンティティー

Name	説明	タイプ	親
user	ユーザーデータ情報のコンテナー	コンテナー	該当なし
user_id	ユーザー ID。	文字列	**user**
caps	ユーザーケイパビリティー。	コンテナー	**user**

成功すると、レスポンスにはユーザーのケイパビリティーが含まれます。

表1.19 特別なエラーレスポンス

Name	説明	コード
InvalidCap	無効な管理者ケイパビリティーの付与を試行。	400 Bad Request

1.13. ユーザーからの機能の削除

指定したユーザーから管理ケイパビリティーを削除します。

機能

```
`users=write`
```

構文

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

表1.20 リクエストパラメーター

Name	説明	型	例	必須
uid	管理ケイパビリティーを削除するユーザー ID。	文字列	**foo_user**	はい
user-caps	ユーザーから削除する管理ケイパビリティー。	文字列	**usage=read, write**	はい

表1.21 レスポンスエンティティー

Name	説明	タイプ	親
user	ユーザーデータ情報のコンテナー	コンテナー	該当なし
user_id	ユーザー ID。	文字列	**user**
caps	ユーザーケイパビリティー。	コンテナー	**user**

成功すると、レスポンスにはユーザーのケイパビリティーが含まれます。

表1.22 特別なエラーレスポンス

Name	説明	コード
InvalidCap	無効な管理ケイパビリティーの削除を試行します。	400 Bad Request
NoSuchCap	ユーザーは、指定されていないケイパビリティーです。	404 Not Found

1.14. キーの作成

新しいキーを作成します。**subuser** を指定すると、デフォルトで作成されたキーは swift タイプになります。**access-key** または **secret-key** のいずれかのみが指定された場合、コミットされたキーは自動的に生成されます。**secret-key** のみが指定されている場合、**access-key** は自動的に生成されます。デフォルトでは、生成されたキーは、既存のキーペアを置き換えることなくキーリングに追加されます。**access-key** が指定され、ユーザーが所有する既存のキーを参照すると、そのキーは変更されます。レスポンスは、作成された鍵と同じタイプの鍵をすべて一覧表示するコンテナーです。

注記

swift キーの作成時に、**access-key** オプションを指定しても効果はありません。また、ユーザーまたはサブユーザーごとに 1 つの swift キーのみを保持することができます。

機能

```
`users=write`
```

構文

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

表1.23 リクエストパラメーター

Name	説明	型	例	必須
uid	新しいキーを受け取るユーザー ID。	文字列	**foo_user**	はい
subuser	新しいキーを受け取るサブユーザー ID。	文字列	**sub_foo**	いいえ
key-type	生成されるキータイプ。オプションは swift、s3 (デフォルト) です。	文字列	**s3** [**s3**]	いいえ
access-key	アクセスキーを指定します。	文字列	**AB01C2D3EF45G6H7IJ8K**	いいえ
secret-key	秘密鍵を指定します。	文字列	**0ab/CdeFGhij1klmnopqRSTUv1WxyZabcDEFgHij**	いいえ
generate-key	新しいキーペアを生成し、既存のキーリングに追加します。	ブール値	True [**True**]	いいえ

表1.24 レスポンスエンティティー

Name	説明	タイプ	親
keys	このユーザーアカウントに関連付けられたタイプのキー。	コンテナー	該当なし
user	キーに関連付けられたユーザーアカウント。	文字列	**keys**
access-key	アクセスキー。	文字列	**keys**
secret-key	シークレットキー	文字列	**keys**

表1.25 特別なエラーレスポンス

Name	説明	コード
InvalidAccessKey	無効なアクセスキーが指定されている。	400 Bad Request
InvalidKeyType	無効なキータイプが指定されている。	400 Bad Request
InvalidSecretKey	無効なシークレットキーが指定されている。	400 Bad Request
InvalidKeyType	無効なキータイプが指定されている。	400 Bad Request
KeyExists	提供されたアクセスキーが存在し、別のユーザーに属している。	409 Conflict

1.15. 鍵の削除

既存のキーを削除します。

機能

```
`users=write`
```

構文

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

表1.26 リクエストパラメーター

Name	説明	型	例	必須
access-key	削除する S3 キーペアに属する S3 アクセスキー。	文字列	**AB01C2D3EF45G6H7IJ8K**	はい
uid	キーの削除元のユーザー。	文字列	**foo_user**	いいえ
subuser	キーの削除元のサブユーザー。	文字列	**sub_foo**	いいえ
key-type	削除するキータイプ。オプションは swift、s3 です。 **注記** swift キーを削除するために必要です。	String	**swift**	いいえ

特別なエラーレスポンス

なし。

レスポンスエンティティー

なし。

1.16. バケット通知

ストレージ管理者は、これらの API を使用してバケット通知メカニズムの設定および制御インターフェイスを提供できます。API トピックは、特定のエンドポイントの定義が含まれる名前が付けられたオブジェクトです。バケット通知では、トピックを特定のバケットに関連付けます。**S3 バケット操作** セクションでは、バケット通知の詳細が表示されます。

注記

すべてのトピックアクションでは、パラメーターは URL エンコードされ、**application/x-www-form-urlencoded** コンテンツタイプを使用してメッセージのボディーで送信されます。

注記

トピックの更新を有効にするには、トピックにすでに関連付けられているバケット通知を再作成する必要があります。

1.16.1. 前提条件

- Ceph Object Gateway 上にバケット通知を作成します。

1.16.2. トピックの作成

バケット通知を作成する前に、トピックを作成できます。トピックは Simple Notification Service (SNS) エンティティーで、すべてのトピック操作 (つまり **create**、**delete**、**list**、および **get**) は SNS 操作です。トピックには、バケット通知の作成時に使用されるエンドポイントパラメーターが必要です。リクエストが正常に行われると、レスポンスには、バケット通知要求でこのトピックを参照するために後で使用できるトピックの Amazon Resource Name (ARN) が含まれます。

注記

topic_arn はバケット通知設定を提供し、トピックの作成後に生成されます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ルートレベルのアクセス。
- Ceph Object Gateway のインストール
- ユーザーアクセスキーおよびシークレットキー。
- エンドポイントパラメーター。

手順

1. 以下の要求形式でトピックを作成します。

構文

```
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]
```

リクエストパラメーターを以下に示します。

- **Endpoint**: 通知を送信するエンドポイントの URL。
- **OpaqueData**: 不透明なデータはトピック設定で設定され、トピックによって発生するすべての通知に追加されます。
- HTTP エンドポイント:
  - **URL**: http[s]://**FQDN**[: **PORT**]
  - **ポートのデフォルト**: HTTP または HTTPS にそれぞれ 80 または 443 を使用します。
  - **verify-ssl**: サーバー証明書がクライアントによって検証されているかどうかを示します。デフォルトでは **true** です。
- AMQP0.9.1 エンドポイント:
  - **URL**: amqp://[**USER** : **PASSWORD** @] **FQDN** [: **PORT**][/**VHOST**].
  - ユーザーおよびグループのデフォルト値はそれぞれ **guest** と **guest** です。
  - ユーザーおよびパスワードは、HTTPS でのみ提供できます。そうしないと、トピック作成要求は拒否されます。
  - **Port のデフォルト値** は 5672 です。
  - **vhost** のデフォルトは/です。
  - **amqp-exchange**: 交換は存在し、トピックに基づいてメッセージをルーティングできる必要があります。これは AMQP0.9.1 の必須パラメーターです。同じエンドポイントを参照するさまざまなトピックが同じ交換を使用する必要があります。
  - **amqp-ack-level**: 最終宛先に送信される前にメッセージがブローカーで永続化される可能性があるため、終了確認は不要です。確認方法は 3 つあります。
    - **none**: ブローカーに送信された場合にメッセージが **配信されている** と見なされます。

- **broker**: デフォルトでは、メッセージはブローカーによって確認応答されると **配信されている** と見なされます。
- **routable**: ブローカーがコンシューマーにルーティングできる場合、メッセージは **配信されている** と見なされます。

注記

特定のパラメーターのキーと値は、同じ行または特定の順序で存在する必要はありませんが、同じインデックスを使用する必要があります。属性インデックスは、特定の値から順番にしたり、開始したりする必要はありません。

注記

topic-name は AMQP トピックに使用されます。

- Kafka エンドポイント:
  - **URL**: kafka://[**USER**: **PASSWORD** @] **FQDN**[: **PORT**].
  - **use-ssl** がデフォルトで **false** に設定される場合。**use-ssl** が **true** に設定されている場合は、ブローカーへの接続にセキュアな接続が使用されます。
  - **ca-location** が指定され、セキュアな接続が使用される場合は、ブローカーを認証するために、デフォルトの CA ではなく、指定された CA が使用されます。
  - ユーザーおよびパスワードは HTTP[S] でのみ提供できます。そうでない場合、トピック作成リクエストは拒否されます。
  - ユーザーおよびパスワードは **use-ssl** とのみ提供でき、ブローカーへの接続に失敗していました。
  - **Port のデフォルト値** は 9092 です。
  - **kafka-ack-level**: 最終宛先に送信される前にメッセージがブローカーで永続化される可能性があるため、終了確認は不要です。確認方法は 2 つあります。
    - **none**: ブローカーに送信された場合にメッセージが **配信されている** と見なされます。
    - **broker**: デフォルトでは、メッセージはブローカーによって確認応答されると **配信されている** と見なされます。

2. 次の形式でレスポンスを作成します。

構文

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

注記

レスポンスのトピックの Amazon Resource Name (ARN) の形式は、**arn:aws:sns:ZONE_GROUP:TENANT:TOPIC** になります。

以下は AMQP0.9.1 エンドポイントの例になります。

構文

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

1.16.3. トピック情報の取得

特定のトピックに関する情報を返します。これには、指定されている場合にはエンドポイント情報を含めることができます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ルートレベルのアクセス。
- Ceph Object Gateway のインストール
- ユーザーアクセスキーおよびシークレットキー。
- エンドポイントパラメーター。

手順

1. 以下の要求形式でトピック情報を取得します。

構文

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

レスポンスフォーマットの例を以下に示します。

```
<GetTopicResponse>
  <GetTopicRersult>
    <Topic>
      <User>
      </User>
      <Name>
      </Name>
      <EndPoint>
        <EndpointAddress>
        </EndpointAddress>
        <EndpointArgs>
        </EndpointArgs>
        <EndpointTopic>
```

```
      </EndpointTopic>
    </EndPoint>
    <TopicArn>
    </TopicArn>
    <OpaqueData>
    </OpaqueData>
   </Topic>
 </GetTopicResult>
 <ResponseMetadata>
  <RequestId>
  </RequestId>
 </ResponseMetadata>
</GetTopicResponse>
```

タグおよびその定義を以下に示します。

- **User**: トピックを作成したユーザーの名前。
- **Name**: トピックの名前。
- **EndpointAddress**: エンドポイントの URL。エンドポイント URL にユーザーおよびパスワード情報が含まれる場合、リクエストは HTTPS 経由で行う必要があります。そうでない場合、トピックの取得要求は拒否されます。
- **EndPointArgs**: エンドポイント引数。
- **EndpointTopic**: エンドポイントに送信されるトピック名は、上記のトピック名とは異なる場合があります。
- **TopicArn**: Topic ARN。

1.16.4. トピックの一覧表示

ユーザーが定義したトピックを一覧表示します。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ルートレベルのアクセス。
- Ceph Object Gateway のインストール
- ユーザーアクセスキーおよびシークレットキー。
- エンドポイントパラメーター。

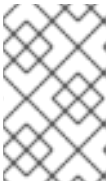
手順

1. 以下の要求形式でトピック情報を一覧表示します。

```
POST
Action=ListTopics
```

レスポンスフォーマットの例を以下に示します。

```
<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>
```



注記

エンドポイント URL にユーザーおよびパスワード情報が含まれる場合は、トピックのいずれかで要求を行う必要があります。そうでない場合、トピック一覧の要求は拒否されます。

1.16.5. トピックの削除

削除したトピックを削除すると、操作はなく、失敗は発生しません。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ルートレベルのアクセス。
- Ceph Object Gateway のインストール
- ユーザーアクセスキーおよびシークレットキー。
- エンドポイントパラメーター。

手順

1. 以下の要求形式でトピックを削除します。

構文

構文

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

レスポンスフォーマットの例を以下に示します。

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

1.16.6. イベントレコード

イベントは、Ceph Object Gateway によって行われる操作に関する情報を保持し、選択したエンドポイント (HTTP、HTTPS、Kafka、または AMQ0.9.1 など) 上のペイロードとして送信されます。イベントレコードは JSON 形式になります。

例

```
{"Records":[
    {
        "eventVersion":"2.1",
        "eventSource":"ceph:s3",
        "awsRegion":"us-east-1",
        "eventTime":"2019-11-22T13:47:35.124724Z",
        "eventName":"s3: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"
    }
]}
```

以下はイベントレコードのキーおよびその定義です。

- **awsRegion**: Zonegroup。
- **eventTime**: イベントがトリガーされたタイミングを示すタイムスタンプ。
- **eventName**: イベントのタイプ。
- **userIdentity.principalId**: イベントを開始したユーザーの ID。
- **requestParameters.sourceIPAddress**: イベントをトリガーしたクライアントの IP アドレス。このフィールドはサポートされません。
- **responseElements.x-amz-request-id**: イベントをトリガーしたリクエスト ID。
- **responseElements.x_amz_id_2**: イベントがトリガーされた Ceph Object Gateway の IP アドレスID 形式は **RGWID-ZONE-ZONEGROUP** です。
- **s3.configurationId**: イベントを作成した通知 ID。
- **s3.bucket.name**: バケットの名前。
- **s3.bucket.ownerIdentity.principalId**: バケットの所有者。
- **s3.bucket.arn**: バケットの Amazon Resource Name(ARN)。
- **s3.bucket.id**: バケットのアイデンティティー。
- **s3.object.key**: オブジェクトキー。
- **s3.object.size**: オブジェクトのサイズ
- **s3.object.eTag**: オブジェクト etag。
- **s3.object.version**: バージョン化されたバケットのオブジェクトバージョン。
- **s3.object.sequencer**: 16 進数形式でオブジェクトごとの変更識別子を増加させます。
- **s3.object.metadata**: **x-amz-meta** として送信されるオブジェクトにメタデータセット。
- **s3.object.tags**: オブジェクトに設定されたタグ。
- **s3.eventId**: イベントの一意のアイデンティティー
- **s3.opaqueData**: Opaque データはトピック設定で設定され、トピックによってトリガーされるすべての通知に追加されます。

関連情報

- 詳細は、Event Message Structure を参照してください。
- 詳細は、Red Hat Ceph Storage 開発者ガイドの サポート対象のイベントタイプ セクションを参照してください。

1.16.7. サポートされるイベントタイプ

以下のイベントタイプがサポートされます。

- **s3:ObjectCreated:***
- **s3:ObjectCreated:Put**
- **s3:ObjectCreated:Post**
- **s3:ObjectCreated:Copy**
- **s3:ObjectCreated:CompleteMultipartUpload**
- **s3:ObjectRemoved:***
- **s3:ObjectRemoved:Delete**
- **s3:ObjectRemoved:DeleteMarkerCreated**

1.16.8. 関連情報

- 詳細は、Red Hat Ceph Storage Object Gateway 設定および管理ガイドの バケット通知の作成 セクションを参照してください。

1.17. バケット情報の取得

既存のバケットのサブセットに関する情報を取得します。**uid** が **bucket** なしで指定されると、そのユーザーに属するすべてのバケットが返されます。**bucket** のみが指定されている場合は、その特定のバケットの情報を取得します。

機能

```
`buckets=read`
```

構文

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

表1.27 リクエストパラメーター

Name	説明	型	例	必須
bucket	情報を返すバケット。	文字列	**foo_bucket**	いいえ

Name	説明	型	例	必須
uid	バケット情報を取得するユーザー。	文字列	**foo_user**	いいえ
stats	バケットの統計を返します。	ブール値	True [False]	No

表1.28 レスポンスエンティティー

Name	説明	タイプ	親
stats	バケットごとの情報	コンテナー	該当なし
buckets	1つ以上のバケットコンテナーの一覧が含まれます。	コンテナー	**bucket**
単一バケット情報用のコンテナー。	コンテナー	**buckets**	**name**
バケットの名前。	文字列	**bucket**	**pool**
バケットが保存されているプール。	文字列	**bucket**	**id**
一意のバケット ID。	文字列	**bucket**	**marker**
内部バケットタグ。	文字列	**bucket**	**owner**
バケット所有者のユーザー ID。	文字列	**bucket**	**usage**
ストレージの使用情報。	コンテナー	**bucket**	**index**

要求に成功すると、必要なバケット情報が含まれるバケットコンテナーが返されます。

表1.29 特別なエラーレスポンス

Name	説明	コード
IndexRepairFailed	バケットのインデックスが失敗しました。	409 Conflict

1.18. バケットインデックスを確認します。

既存のバケットのインデックスを確認します。

注記

check-objects で複数パートオブジェクトアカウンティングを確認するには、**fix** を True に設定する必要があります。

機能

buckets=write

構文

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

表1.30 リクエストパラメーター

Name	説明	型	例	必須
bucket	情報を返すバケット。	文字列	**foo_bucket**	はい
check-objects	複数パートオブジェクトアカウンティングを確認します。	ブール値	True [False]	いいえ
fix	また、チェック時にバケットインデックスも修正します。	ブール値	False [False]	No

表1.31 レスポンスエンティティー

Name	説明	型
index	バケットインデックスのステータス。	文字列

表1.32 特別なエラーレスポンス

Name	説明	コード
IndexRepairFailed	バケットのインデックスが失敗しました。	409 Conflict

1.19. バケットの削除

既存のバケットを削除します。

機能

`buckets=write`

構文

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

表1.33 リクエストパラメーター

Name	説明	型	例	必須
bucket	削除するバケット。	文字列	**foo_bucket**	はい
purge-objects	削除前にバケットオブジェクトを削除します。	ブール値	True [False]	No

レスポンスエンティティー

なし。

表1.34 特別なエラーレスポンス

Name	説明	コード
BucketNotEmpty	空でないバケットの削除を試行しました。	409 Conflict
ObjectRemovalFailed	オブジェクトを削除できません。	409 Conflict

1.20. バケットのリンク

バケットを指定ユーザーにリンクし、直前のユーザーからバケットのリンクを解除します。

機能

`buckets=write`

構文

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

表1.35 リクエストパラメーター

Name	説明	型	例	必須
bucket	リンクを解除するバケット。	文字列	**foo_bucket**	はい

Name	説明	型	例	必須
uid	バケットをリンクするユーザー ID。	文字列	**foo_user**	はい

表1.36 レスポンスエンティティー

Name	説明	タイプ	親
bucket	単一バケット情報用のコンテナー。	コンテナー	該当なし
name	バケットの名前。	文字列	**bucket**
pool	バケットが保存されているプール。	文字列	**bucket**
id	一意のバケット ID。	文字列	**bucket**
marker	内部バケットタグ。	文字列	**bucket**
owner	バケット所有者のユーザー ID。	文字列	**bucket**
usage	ストレージの使用情報。	コンテナー	**bucket**
index	バケットインデックスのステータス。	文字列	**bucket**

表1.37 特別なエラーレスポンス

Name	説明	コード
BucketUnlinkFailed	指定のユーザーからバケットのリンクを解除できません。	409 Conflict
BucketLinkFailed	バケットを指定されたユーザーにリンクできません。	409 Conflict

1.21. バケットのリンクを解除します。

指定されたユーザーからバケットのリンクを解除します。主にバケットの所有権を変更するのに役立ちます。

機能

`buckets=write`

構文

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

表1.38 リクエストパラメーター

Name	説明	型	例	必須
bucket	リンクを解除するバケット。	文字列	**foo_bucket**	はい
uid	バケットのリンクを解除するユーザー ID。	文字列	**foo_user**	はい

レスポンスエンティティー

なし。

表1.39 特別なエラーレスポンス

Name	説明	コード
BucketUnlinkFailed	指定のユーザーからバケットのリンクを解除できません。	409 Conflict

1.22. バケットまたはオブジェクトポリシーを取得する

オブジェクトまたはバケットのポリシーを読み取ります。

機能

`buckets=read`

構文

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

表1.40 リクエストパラメーター

Name	説明	型	例	必須
bucket	ポリシーを読み取るバケット。	文字列	**foo_bucket**	はい

Name	説明	型	例	必須
object	ポリシーの読み取り元となるオブジェクト。	文字列	**foo.txt**	いいえ

表1.41 レスポンスエンティティー

Name	説明	タイプ	親
policy	アクセス制御ポリシー。	コンテナー	該当なし

成功した場合には、オブジェクトまたはバケットポリシーを返します。

表1.42 特別なエラーレスポンス

Name	説明	コード
IncompleteBody	バケットポリシー要求にバケットが指定されていないか、オブジェクトがオブジェクトポリシー要求に指定されていません。	400 Bad Request

1.23. オブジェクトの削除

既存のオブジェクトを削除します。

注記

所有者を一時停止せずに指定する必要はありません。

機能

```
`buckets=write`
```

構文

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

表1.43 リクエストパラメーター

Name	説明	型	例	必須
bucket	削除されるオブジェクトを含むバケット。	文字列	**foo_bucket**	はい

Name	説明	型	例	必須
object	削除するオブジェクト。	文字列	**foo.txt**	はい

レスポンスエンティティー

なし。

表1.44 特別なエラーレスポンス

Name	説明	コード
NoSuchObject	指定されたオブジェクトは存在しません。	404 Not Found
ObjectRemovalFailed	オブジェクトを削除できません。	409 Conflict

1.24. クォータ

管理操作 API を使用すると、ユーザーおよびユーザーが所有するバケットにクォータを設定できます。クォータには、バケットのオブジェクトの最大数と、メガバイト単位のストレージの最大サイズが含まれます。

クォータを表示するには、ユーザーに **users=read** ケイパビリティーが必要です。クォータを設定、変更、または無効にするには、ユーザーに **users=write** ケイパビリティーが必要です。

クォータの有効なパラメーターには以下が含まれます。

- Bucket: **bucket** オプションでは、ユーザーが所有するバケットのクォータを指定できます。
- Maximum Objects: **max-objects** 設定では、オブジェクトの最大数を指定できます。負の値を設定すると、この設定が無効になります。
- Maximum Size: **max-size** オプションでは、バイトの最大数のクォータを指定できます。負の値を設定すると、この設定が無効になります。
- Quota Scope: **quota-scope** オプションは、クォータのスコープを設定します。オプションは **bucket** と **user** です。

1.25. ユーザークォータの取得

クォータを取得するには、**read** パーミッションを持つ **users** ケイパビリティーが設定されている必要があります。

構文

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

1.26. ユーザークォータの設定

クォータを設定するには、ユーザーに **write** パーミッションを持つ **users** ケイパビリティーを設定する必要があります。

構文

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

コンテンツには、対応する読み取り操作でエンコードされているクォータ設定の JSON 表現が含まれている必要があります。

1.27. バケットクォータの取得

バケットクォータを取得するには、ユーザーは **read** 権限を設定した **user** 機能を持っている必要があります。

構文

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

1.28. バケットクォータの設定

クォータを設定するには、ユーザーに **write** パーミッションを持つ **users** ケイパビリティーを設定する必要があります。

構文

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

コンテンツには、対応する読み取り操作でエンコードされているクォータ設定の JSON 表現が含まれている必要があります。

1.29. 個別のバケットのクォータの設定

クォータを設定するには、ユーザーに **write** パーミッションを持つ **buckets** 機能が設定されている必要があります。

構文

```
PUT /admin/bucket?quota&uid=UID&bucket=BUCKET_NAME&quota
```

コンテンツには、クォータ設定の JSON 表現を含める必要があります。

1.30. 使用方法情報の取得

帯域幅の使用情報の要求。

機能

```
`usage=read`
```

構文

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

表1.45 リクエストパラメーター

Name	説明	型	必須
uid	情報が要求されるユーザー。	文字列。	はい
start	要求されたデータの開始時間を指定する日付および (任意の) 時間。(例: **2012-09-25 16:00:00**)	文字列	いいえ
end	要求されたデータの終了時間を指定する日付および (任意の) 時間。(例: **2012-09-25 16:00:00**)	文字列	いいえ
show-entries	データエントリーを返すかどうかを指定します。	ブール値	いいえ
show-summary	データ要約を返すかどうかを指定します。	ブール値	No

表1.46 レスポンスエンティティー

Name	説明	型
usage	使用方法に関する情報用のコンテナー。	コンテナー
entries	使用方法エントリー情報のコンテナー。	コンテナー
user	ユーザーデータ情報のコンテナー	コンテナー
owner	バケットを所有するユーザーの名前。	文字列
bucket	バケット名。	文字列
time	データが指定されている時間の下限 (最初の関連する時間の開始に丸められます)。	文字列
epoch	**1/1/1970** からの経過時間 (秒単位)。	文字列
categories	統計情報カテゴリーのコンテナー。	コンテナー
entry	stats エントリーのコンテナー。	コンテナー
category	統計が提供される要求カテゴリーの名前。	文字列
bytes_sent	Ceph Object Gateway によって送信されるバイト数。	整数
bytes_received	Ceph Object Gateway が受け取るバイト数。	整数

Name	説明	型
ops	演算の数。	整数
successful_ops	成功した操作の数。	整数
summary	統計情報の概要のコンテナー。	コンテナー
total	統計情報の概要集計合計のコンテナー。	コンテナー

成功すると、レスポンスには要求された情報が含まれます。

1.31. 使用方法に関する情報を削除

使用方法に関する情報を削除します。日付を指定しないと、すべての使用情報が削除されます。

機能

> `usage=write`

構文

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

表1.47 リクエストパラメーター

Name	説明	型	例	必須
uid	情報が要求されるユーザー。	文字列	**foo_user**	いいえ
start	要求されたデータの開始時間を指定する日付および (任意の) 時間。	文字列	**2012-09-25 16:00:00**	いいえ
end	要求されたデータの終了時間を指定する日付および 時間 (非包括的)。	文字列	**2012-09-25 16:00:00**	いいえ
remove-all	マルチユーザーデータの削除を確認するために **uid** が指定されていない場合に必須です。	ブール値	True [False]	No

1.32. 標準エラーレスポンス

以下の表は、標準的なエラーレスポンスと説明の詳細を示しています。

Name	説明	コード
AccessDenied	アクセスが拒否されました。	403 forbidden
InternalError	内部サーバーエラー。	500 Internal Server Error
NoSuchUser	ユーザーが存在しない。	404 Not Found
NoSuchBucket	バケットが存在しません。	404 Not Found
NoSuchKey	そのようなアクセスキーはありません。	404 Not Found

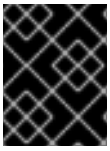
第2章 CEPH OBJECT GATEWAY および S3 API

開発者は、Amazon S3 データアクセスモデルと互換性のある RESTful アプリケーションプログラミングインターフェイス (API) を使用できます。Ceph Object Gateway を使用して、Red Hat Ceph Storage クラスターに保存されているバケットおよびオブジェクトを管理できます。

2.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- RESTful クライアント。

2.2. S3 の制限



重要

以下の制限事項を使用してください。お使いのハードウェアの選択には影響があるため、この要件を Red Hat アカウントチームと常に相談してください。

- **Amazon S3 を使用する場合の最大オブジェクトサイズ:** 個別の Amazon S3 オブジェクトは、最小の 0B から最大 5TB のサイズに制限できます。1 つの **PUT** でアップロードできる最大オブジェクトは 5 GB です。100MB を超えるオブジェクトの場合は、Multipart Upload ケイパビリティーの使用を検討してください。
- **Amazon S3 を使用する場合の最大メタデータサイズ:** オブジェクトに適用できるユーザーメタデータの合計サイズに定義された制限はありませんが、単一の HTTP リクエストは 16,000 バイトに制限されます。
- **Red Hat Ceph Storage クラスターでは、S3 オブジェクトおよびメタデータを保存するために生成するデータオーバーヘッドのデータ量:** 推定時間は 200-300 バイトとオブジェクト名の長さです。バージョン管理されたオブジェクトは、バージョン数に比例する領域を追加で使用します。また、マルチパートアップロードなどのトランザクション更新中に一時的なオーバーヘッドが発生しますが、これらのオーバーヘッドはガベージコレクション中にリカバリーされます。

関連情報

- 詳細は、**Red Hat Ceph Storage 開発者ガイド**の **サポートされないヘッダーフィールド** を参照してください。

2.3. S3 API を使用した CEPH OBJECT GATEWAY へのアクセス

開発者は、Amazon S3 API の使用を開始する前に、Ceph Object Gateway および Secure Token Service (STS) へのアクセスを設定する必要があります。

2.3.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- 実行中の Ceph Object Gateway。
- RESTful クライアント。

2.3.2. S3 認証

Ceph Object Gateway への要求は、認証または認証解除のいずれかになります。Ceph Object Gateway は、認証されていないリクエストが匿名ユーザーによって送信されることを前提としています。Ceph Object Gateway は、固定 ACL をサポートしています。

ほとんどのユースケースでは、クライアントは、Java や Python Boto 用の Amazon SDK の **AmazonS3Client** などの既存のオープンソースライブラリーを使用します。オープンソースライブラリーでは、アクセスキーおよびシークレットキーを渡すだけで、ライブラリーはユーザーの要求ヘッダーおよび認証署名をビルドします。ただし、リクエストを作成して署名することもできます。

リクエストの認証には、アクセスキーとベース 64 でエンコードされたハッシュベースのメッセージ認証コード (HMAC) が Ceph Object Gateway サーバーに送信される前に要求に追加する必要があります。Ceph Object Gateway は S3 互換の認証を使用します。

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
```

上記の例では、**ACCESS_KEY** をアクセスキー ID の値に置き換え、その後にコロン (**:**) を追加します。**HASH_OF_HEADER_AND_SECRET** を、正規化されたヘッダー文字列のハッシュとアクセスキー ID に対応するシークレットに置き換えます。

ヘッダー文字列およびシークレットのハッシュの生成

ヘッダー文字列およびシークレットのハッシュを生成するには、以下を実行します。

1. ヘッダー文字列の値を取得します。
2. 要求ヘッダー文字列を正規形式に正規化します。
3. SHA-1 ハッシュアルゴリズムを使用して HMAC を生成します。
4. **hmac** の結果を base-64 としてエンコードします。

ヘッダーを正規化

ヘッダーを正規の形式に正規化するには、以下を行います。

1. すべての **content-** ヘッダーを取得します。
2. **content-type** および **content-md5** 以外の **content-** ヘッダーをすべて削除します。
3. **content-** ヘッダー名が小文字であることを確認します。
4. **content-** ヘッダーの辞書式で並べ替えます。
5. **Date** ヘッダー AND があることを確認します。指定した日付が、オフセットではなく GMT を使用していることを確認してください。

6. **x-amz-** で始まるヘッダーをすべて取得します。
7. **x-amz-** ヘッダーがすべて小文字であることを確認します。
8. **x-amz-** ヘッダーの辞書式で並べ替えます。
9. 同じフィールド名の複数のインスタンスを単一のフィールドに組み合わせ、フィールド値をコンマで区切ります。
10. ヘッダー値の空白文字および改行文字を、単一スペースに置き換えます。
11. コロンの前後に空白を削除します。
12. 各ヘッダーの後に新しい行を追加します。
13. ヘッダーを要求ヘッダーにマージします。

HASH_OF_HEADER_AND_SECRET を、base-64 でエンコードされた HMAC 文字列に置き換えます。

関連情報

- 詳細は、Amazon Simple Storage Service ドキュメントの Signing and Authenticating REST Requests セクションを参照してください。

2.3.3. S3 サーバー側の暗号化

Ceph Object Gateway は、S3 アプリケーションプログラムインターフェイス (API) のアップロードされたオブジェクトのサーバー側の暗号化をサポートします。サーバー側の暗号化とは、S3 クライアントが暗号化されていない形式で HTTP 経由でデータを送信し、Ceph Object Gateway はそのデータを暗号化した形式で Red Hat Ceph Storage に保存することを意味します。

注記

Red Hat は、Static Large Object (SLO) または Dynamic Large Object (DLO) の S3 オブジェクト暗号化をサポートしません。

重要

暗号化を使用するには、クライアントリクエストは、SSL 接続上でリクエストを送信する **必要があります**。Red Hat は、Ceph Object Gateway が SSL を使用しない限り、クライアントからの S3 暗号化をサポートしません。ただし、テスト目的で、管理者は、ランタイム時に **rgw_crypt_require_ssl** 設定を **false** に設定し、Ceph 設定ファイルで **false** に設定して、Ansible 設定ファイルで **false** に設定し、Ceph Object Gateway の Ansible Playbook を再生して、テスト中に SSL を無効にすることができます。

実稼働環境では、SSL 経由で暗号化された要求を送信できない場合があります。このような場合は、サーバー側の暗号化で HTTP を使用して要求を送信します。

サーバー側の暗号化で HTTP を設定する方法は、以下の**関連情報**セクションを参照してください。

暗号化キーの管理には、以下の 2 つのオプションがあります。

お客様提供のキー

お客様が提供する鍵を使用する場合、S3 クライアントは暗号鍵を各リクエストと共に渡して、暗号化されたデータの読み取りまたは書き込みを行います。これらのキーを管理するのは、お客様の責任です。各オブジェクトの暗号化に使用する Ceph Object Gateway の鍵を覚えておく必要があります。

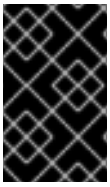
Ceph Object Gateway は、Amazon SSE-C 仕様に従って、S3 API で顧客提供のキー動作を実装します。

お客様がキー管理を処理し、S3 クライアントはキーを Ceph Object Gateway に渡すため、Ceph Object Gateway ではこの暗号化モードをサポートするための特別な設定は必要ありません。

キー管理サービス

キー管理サービスを使用する場合、セキュアなキー管理サービスはキーを格納し、Ceph Object Gateway はデータの暗号化または復号の要求に対応するためにキーをオンデマンドで取得します。

Ceph Object Gateway は、Amazon SSE-KMS 仕様に従って S3 API にキー管理サービスの動作を実装します。



重要

現時点で、テスト済み鍵管理の実装は HashiCorp Vault および OpenStack Barbican です。ただし、OpenStack Barbican はテクノロジープレビューであるため、実稼働システムでの使用はサポートされません。

関連情報

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

2.3.4. S3 アクセス制御リスト

Ceph Object Gateway は S3 と互換性のあるアクセス制御リスト (ACL) の機能をサポートします。ACL は、ユーザーがバケットまたはオブジェクトで実行できる操作を指定するアクセス権限の一覧です。それぞれの付与は、バケットに適用するか、オブジェクトに適用される場合の異なる意味を持ちます。

表2.1 ユーザー操作

パーミッション	バケット	Object
READ	パーミッションを得たユーザーは、バケットのオブジェクトを一覧表示できます。	パーミッションを得たユーザーは、オブジェクトを読み取りできます。
WRITE	パーミッションを得たユーザーは、バケットのオブジェクトを書き込みまたは削除できます。	該当なし
READ_ACP	パーミッションを得たユーザーは、バケット ACL を読み取ることができます。	パーミッションを得たユーザーは、オブジェクト ACL を読み取ることができます。

パーミッション	バケット	Object
WRITE_ACP	パーミッションを得たユーザーは、バケット ACL を書き込めます。	パーミッションを得たユーザーは、オブジェクト ACL に書き込めます。
FULL_CONTROL	Grantee にはバケットのオブジェクトに対する完全なパーミッションがあります。	パーミッションを得たユーザーは、オブジェクト ACL に読み取りまたは書き込みできます。

2.3.5. S3 を使用した Ceph Object Gateway へのアクセスの準備

ゲートウェイサーバーにアクセスする前に、Ceph Object Gateway ノードの前提条件に従う必要があります。

警告

Ceph 設定ファイルを変更してポート **80** を使用するようにし、**Civetweb** がデフォルトの Ansible に設定されたポートである **8080** を使用することは変更 **しないでください**。

前提条件

- Ceph Object Gateway ソフトウェアのインストール。
- Ceph Object Gateway ノードへのルートレベルのアクセス。

手順

1. **root** で、ファイアウォールのポート **8080** を開きます。

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

2. Object Gateway 設定および管理ガイド で説明されているように、ゲートウェイに使用する DNS サーバーにワイルドカードを追加します。
ローカル DNS キャッシュ用のゲートウェイノードを設定することもできます。これを実行するには、以下の手順を実行します。

   a. **root** で **dnsmasq** をインストールおよび設定します。

```
[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
```

IP_OF_GATEWAY_NODE および **FQDN_OF_GATEWAY_NODE** は、ゲートウェイノードの IP アドレスと FQDN に置き換えます。

b. **root** で NetworkManager を停止します。

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

c. **root** として、ゲートウェイサーバーの IP を名前空間として設定します。

```
[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
```

IP_OF_GATEWAY_NODE および **FQDN_OF_GATEWAY_NODE** は、ゲートウェイノードの IP アドレスと FQDN に置き換えます。

d. サブドメイン要求を確認します。

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

FQDN_OF_GATEWAY_NODE は、ゲートウェイノードの FQDN に置き換えます。

警告

ローカルの DNS キャッシュ用にゲートウェイサーバーを設定することは、テスト目的のみを目的としています。これを行った後は、外部ネットワークにはアクセスできなくなります。**Red Hat Ceph Storage クラスターおよびゲートウェイノードに適切な DNS サーバーを使用することを強く推奨します。**

3. Object Gateway の設定および管理ガイド に説明されているように、**S3** アクセスに **radosgw** ユーザーを作成し、生成した **access_key** および **secret_key** をコピーします。**S3** アクセス、およびそれ以降のバケット管理タスクには、これらのキーが必要です。

2.3.6. Ruby AWS S3 を使用した Ceph Object Gateway へのアクセス

Ruby プログラミング言語は、**S3** アクセスに **aws-s3** gem と共に使用できます。**Ruby AWS::S3** で Ceph Object Gateway サーバーにアクセスするために使用されるノードで以下の手順を実行します。

前提条件

- Ceph Object Gateway へのユーザーレベルのアクセス。
- Ceph Object Gateway にアクセスするノードへのルートレベルのアクセス。

- インターネットアクセス。

手順

1. **ruby** パッケージをインストールします。

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

注記

上記のコマンドは **ruby** と、**rubygems**、**ruby-libs** などの基本的な依存関係をインストールします。コマンドによってすべての依存関係がインストールされていない場合は、個別にインストールします。

2. Ruby パッケージ **aws-s3** をインストールします。

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

3. プロジェクトディレクトリーを作成します。

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

4. コネクションファイルを作成します。

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

5. **conn.rb** ファイルに以下のコンテンツを貼り付けます。

構文

```
#!/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'
)
```

FQDN_OF_GATEWAY_NODE は、Ceph Object Gateway ノードの FQDN に置き換えます。**MY_ACCESS_KEY** および **MY_SECRET_KEY** を、Red Hat Ceph Storage Object Gateway Configuration and Administration Guide に記載されているように、**S3** アクセス用の **radosgw** ユーザーを作成したときに生成された **access_key** および **secret_key** に置き換えます。

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'
)
```

ファイルを保存して、エディターを終了します。

6. ファイルを実行可能にします。

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

7. コマンドを実行します。

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

ファイルに正しく値を指定した場合は、コマンドの出力は **0** になります。

8. バケットを作成するための新規ファイルを作成します。

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

以下のコンテンツをファイルに貼り付けます。

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

load 'conn.rb'

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

ファイルを保存して、エディターを終了します。

9. ファイルを実行可能にします。

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

10. コマンドを実行します。

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

コマンドの出力が **true** の場合は、バケット **my-new-bucket1** が正常に作成されたことを意味します。

11. 所有されるバケットを一覧表示するために新規ファイルを作成します。

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

以下のコンテンツをファイルに貼り付けます。

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

load 'conn.rb'

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

ファイルを保存して、エディターを終了します。

12. ファイルを実行可能にします。

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

13. コマンドを実行します。

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

出力は以下のようになります。

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

14. オブジェクトを作成するための新規ファイルを作成します。

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

以下のコンテンツをファイルに貼り付けます。

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

load 'conn.rb'

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

ファイルを保存して、エディターを終了します。

15. ファイルを実行可能にします。

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

16. コマンドを実行します。

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

これで、文字列 **Hello World!** で **hello.txt** が作成されます。

17. バケットのコンテンツを一覧表示するための新規ファイルを作成します。

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

以下のコンテンツをファイルに貼り付けます。

```
#!/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
```

ファイルを保存して、エディターを終了します。

18. ファイルを実行可能にします。

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

19. コマンドを実行します。

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

出力は以下のようになります。

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

20. 空のバケットを削除するために新規ファイルを作成します。

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

以下のコンテンツをファイルに貼り付けます。

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

load 'conn.rb'

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

ファイルを保存して、エディターを終了します。

21. ファイルを実行可能にします。

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

22. コマンドを実行します。

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

バケットが正常に削除されると、コマンドは **0** を出力として返します。

注記

create_bucket.rb ファイルを編集し、空のバケットを作成します (例: **my-new-bucket4**、**my-new-bucket5**)。次に、空のバケットの削除を試みる前に、上記の **del_empty_bucket.rb** ファイルを適宜編集します。

23. 空でないバケットを削除する新規ファイルを作成します。

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

以下のコンテンツをファイルに貼り付けます。

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

load 'conn.rb'

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

ファイルを保存して、エディターを終了します。

24. ファイルを実行可能にします。

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

25. コマンドを実行します。

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

バケットが正常に削除されると、コマンドは **0** を出力として返します。

26. オブジェクトを削除する新しいファイルを作成します。

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

以下のコンテンツをファイルに貼り付けます。

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

load 'conn.rb'

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

ファイルを保存して、エディターを終了します。

27. ファイルを実行可能にします。

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

28. コマンドを実行します。

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

これにより、オブジェクト **hello.txt** が削除されます。

2.3.7. Ruby AWS SDK を使用した Ceph Object Gateway へのアクセス

Ruby プログラミング言語は、**S3** アクセスに **aws-sdk** gem と共に使用できます。**Ruby AWS::SDK** を使用して Ceph Object Gateway サーバーにアクセスするために使用されるノードで以下の手順を実行します。

前提条件

- Ceph Object Gateway へのユーザーレベルのアクセス。
- Ceph Object Gateway にアクセスするノードへのルートレベルのアクセス。
- インターネットアクセス。

手順

1. **ruby** パッケージをインストールします。

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

注記

上記のコマンドは **ruby** と、**rubygems**、**ruby-libs** などの基本的な依存関係をインストールします。コマンドによってすべての依存関係がインストールされていない場合は、個別にインストールします。

2. Ruby パッケージ **aws-sdk** をインストールします。

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

3. プロジェクトディレクトリーを作成します。

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

4. コネクションファイルを作成します。

```
[user@ruby_aws_sdk]$ vim conn.rb
```

5. **conn.rb** ファイルに以下のコンテンツを貼り付けます。

構文

```
#!/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'
)
```

FQDN_OF_GATEWAY_NODE は、Ceph Object Gateway ノードの FQDN に置き換えます。**MY_ACCESS_KEY** および **MY_SECRET_KEY** を、Red Hat Ceph Storage Object Gateway Configuration and Administration Guide に記載されているように、**S3** アクセス用の **radosgw** ユーザーを作成したときに生成された **access_key** および **secret_key** に置き換えます。

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'
)
```

ファイルを保存して、エディターを終了します。

6. ファイルを実行可能にします。

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

7. コマンドを実行します。

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

ファイルに正しく値を指定した場合は、コマンドの出力は **0** になります。

8. バケットを作成するための新規ファイルを作成します。

```
[user@ruby_aws_sdk]$ vim create_bucket.rb
```

以下のコンテンツをファイルに貼り付けます。

構文

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

load 'conn.rb'

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

ファイルを保存して、エディターを終了します。

9. ファイルを実行可能にします。

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

10. コマンドを実行します。

```
[user@ruby_aws_sdk]$ ./create_bucket.rb
```

コマンドの出力が **true** の場合は、バケット **my-new-bucket2** が正常に作成されていることを意味します。

11. 所有されるバケットを一覧表示するために新規ファイルを作成します。

```
[user@ruby_aws_sdk]$ vim list_owned_buckets.rb
```

以下のコンテンツをファイルに貼り付けます。

```
#!/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
```

ファイルを保存して、エディターを終了します。

12. ファイルを実行可能にします。

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

13. コマンドを実行します。

```
[user@ruby_aws_sdk]$ ./list_owned_buckets.rb
```

出力は以下のようになります。

```
my-new-bucket2 2022-04-21 10:33:19 UTC
```

14. オブジェクトを作成するための新規ファイルを作成します。

```
[user@ruby_aws_sdk]$ vim create_object.rb
```

以下のコンテンツをファイルに貼り付けます。

```
#!/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'
)
```

ファイルを保存して、エディターを終了します。

15. ファイルを実行可能にします。

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

16. コマンドを実行します。

```
[user@ruby_aws_sdk]$ ./create_object.rb
```

これで、文字列 **Hello World!** で **hello.txt** が作成されます。

17. バケットのコンテンツを一覧表示するための新規ファイルを作成します。

```
[user@ruby_aws_sdk]$ vim list_bucket_content.rb
```

以下のコンテンツをファイルに貼り付けます。

```
#!/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
```

ファイルを保存して、エディターを終了します。

18. ファイルを実行可能にします。

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

19. コマンドを実行します。

```
[user@ruby_aws_sdk]$ ./list_bucket_content.rb
```

出力は以下のようになります。

```
hello.txt    12    Fri, 22 Apr 2022 15:54:52 GMT
```

20. 空のバケットを削除するために新規ファイルを作成します。

```
[user@ruby_aws_sdk]$ vim del_empty_bucket.rb
```

以下のコンテンツをファイルに貼り付けます。

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

```
load 'conn.rb'

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

ファイルを保存して、エディターを終了します。

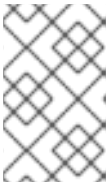
21. ファイルを実行可能にします。

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

22. コマンドを実行します。

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

バケットが正常に削除されると、コマンドは **0** を出力として返します。



注記

create_bucket.rb ファイルを編集し、空のバケットを作成します (例: **my-new-bucket6**、**my-new-bucket7**)。次に、空のバケットの削除を試みる前に、上記の **del_empty_bucket.rb** ファイルを適宜編集します。

23. 空でないバケットを削除する新規ファイルを作成します。

```
[user@ruby_aws_sdk]$ vim del_non_empty_bucket.rb
```

以下のコンテンツをファイルに貼り付けます。

```
#!/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')
```

ファイルを保存して、エディターを終了します。

24. ファイルを実行可能にします。

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

25. コマンドを実行します。

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

バケットが正常に削除されると、コマンドは **0** を出力として返します。

26. オブジェクトを削除する新しいファイルを作成します。

```
[user@ruby_aws_sdk]$ vim delete_object.rb
```

以下のコンテンツをファイルに貼り付けます。

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

load 'conn.rb'

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

ファイルを保存して、エディターを終了します。

27. ファイルを実行可能にします。

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

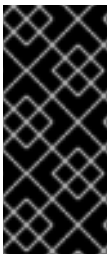
28. コマンドを実行します。

```
[user@ruby_aws_sdk]$ ./delete_object.rb
```

これにより、オブジェクト **hello.txt** が削除されます。

2.3.8. PHP を使用した Ceph Object Gateway へのアクセス

S3 アクセスには PHP スクリプトを使用できます。この手順では、バケットやオブジェクトの削除など、さまざまなタスクを実行する PHP スクリプトの例を提供します。



重要

以下は、**php v5.4.16** および **aws-sdk v2.8.24** に対してテストされています。**php >= 5.5+** が必要なため、**php** に **aws-sdk** の最新バージョンを使用 **しません**。**php 5.5** は、**RHEL 7** のデフォルトリポジトリーでは利用できません。**php 5.5** を使用する場合は、**epel** およびその他のサードパーティーのリポジトリーを有効にする必要があります。また、**php 5.5** および最新バージョンの **aws-sdk** の設定オプションも異なります。

前提条件

- 開発ワークステーションへのルートレベルのアクセス。
- インターネットアクセス。

手順

1. **php** パッケージをインストールします。

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

2. PHP 用に **aws-sdk** の zip アーカイブを ダウンロード し、展開します。
3. プロジェクトディレクトリーを作成します。

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

4. 展開した **aws** ディレクトリーをプロジェクトのディレクトリーにコピーします。以下に例を示します。

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

5. コネクションファイルを作成します。

```
[user@php_s3]$ vim conn.php
```

6. **conn.php** ファイルに以下のコンテンツを貼り付けます。

構文

```
<?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
));
?>
```

FQDN_OF_GATEWAY_NODE は、ゲートウェイノードの FQDN に置き換えます。**MY_ACCESS_KEY** および **MY_SECRET_KEY** を、Red Hat Ceph Storage Object Gateway Configuration and Administration Guide で説明されているように、**S3** アクセス用の **radosgw** ユーザーを作成するときに生成された **access_key** および **secret_key** に置き換えます。**PATH_TO_AWS** は、**php** プロジェクトディレクトリーにコピーした、展開した **aws** ディレクトリーへの絶対パスに置き換えます。

ファイルを保存して、エディターを終了します。

7. コマンドを実行します。

```
[user@php_s3]$ php -f conn.php | echo $?
```

ファイルに正しく値を指定した場合は、コマンドの出力は **0** になります。

8. バケットを作成するための新規ファイルを作成します。

```
[user@php_s3]$ vim create_bucket.php
```

新しいファイルに以下の内容を貼り付けます。

構文

```
<?php

include 'conn.php';

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

?>
```

ファイルを保存して、エディターを終了します。

9. コマンドを実行します。

```
[user@php_s3]$ php -f create_bucket.php
```

10. 所有されるバケットを一覧表示するために新規ファイルを作成します。

```
[user@php_s3]$ vim list_owned_buckets.php
```

以下のコンテンツをファイルに貼り付けます。

構文

```
<?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";
}

?>
```

ファイルを保存して、エディターを終了します。

11. コマンドを実行します。

```
[user@php_s3]$ php -f list_owned_buckets.php
```

出力は以下のようになります。

```
my-new-bucket3 2022-04-21 10:33:19 UTC
```

12. まず **hello.txt** という名前のソースファイルを作成するオブジェクトを作成します。

```
[user@php_s3]$ echo "Hello World!" > hello.txt
```

13. 新しい php ファイルを作成します。

```
[user@php_s3]$ vim create_object.php
```

以下のコンテンツをファイルに貼り付けます。

構文

```
<?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);

?>
```

ファイルを保存して、エディターを終了します。

14. コマンドを実行します。

```
[user@php_s3]$ php -f create_object.php
```

これにより、バケット **my-new-bucket3** でオブジェクト **hello.txt** が作成されます。

15. バケットのコンテンツを一覧表示するための新規ファイルを作成します。

```
[user@php_s3]$ vim list_bucket_content.php
```

以下のコンテンツをファイルに貼り付けます。

構文

```
<?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";
}
?>
```

ファイルを保存して、エディターを終了します。

16. コマンドを実行します。

```
[user@php_s3]$ php -f list_bucket_content.php
```

出力は以下のようになります。

```
hello.txt    12    Fri, 22 Apr 2022 15:54:52 GMT
```

17. 空のバケットを削除するために新規ファイルを作成します。

```
[user@php_s3]$ vim del_empty_bucket.php
```

以下のコンテンツをファイルに貼り付けます。

構文

```
<?php

include 'conn.php';

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

ファイルを保存して、エディターを終了します。

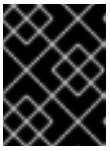
18. コマンドを実行します。

```
[user@php_s3]$ php -f del_empty_bucket.php | echo $?
```

バケットが正常に削除されると、コマンドは **0** を出力として返します。

注記

create_bucket.php ファイルを編集し、空のバケットを作成します (例: **my-new-bucket4**、**my-new-bucket5**)。次に、空のバケットの削除を試みる前に、上記の **del_empty_bucket.php** ファイルを適宜編集します。



重要

空でないバケットの削除は、現在 PHP 2 以降のバージョンの **aws-sdk** ではサポートされていません。

19. オブジェクトを削除する新しいファイルを作成します。

```
[user@php_s3]$ vim delete_object.php
```

以下のコンテンツをファイルに貼り付けます。

構文

```
<?php

include 'conn.php';

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

ファイルを保存して、エディターを終了します。

20. コマンドを実行します。

```
[user@php_s3]$ php -f delete_object.php
```

これにより、オブジェクト **hello.txt** が削除されます。

2.3.9. AWS CLI を使用して Ceph Object Gateway にアクセスする

S3 アクセスには AWS CLI を使用できます。この手順では、AWS CLI をインストールする手順と、MFA 削除が有効なバケットからオブジェクトを削除するなど、さまざまなタスクを実行するコマンドの例をいくつか示します。

前提条件

- Ceph Object Gateway へのユーザーレベルのアクセス。
- 開発ワークステーションへのルートレベルのアクセス。
- **radosgw-admin mfa create** を使用して、多要素認証 (MFA) TOTP トークンが作成されました

手順

1. **awscli** パッケージをインストールします。

```
[user@dev]$ pip3 install --user awscli
```

2. AWS CLI を使用して Ceph Object Storage にアクセスするように **awscli** を設定します。

構文

```
aws configure --profile=MY_PROFILE_NAME

AWS Access Key ID [None]: MY_ACCESS_KEY
AWS Secret Access Key [None]: MY_SECRET_KEY
Default region name [None]:
Default output format [None]:
```

MY_PROFILE_NAME を、このプロファイルを識別するために使用する名前に置き換えます。**MY_ACCESS_KEY** および **MY_SECRET_KEY** を、Red Hat Ceph Storage Object Gateway Configuration and Administration Guide で説明されているように、**S3** アクセス用の **radosgw** ユーザーを作成するときに生成された **access_key** および **secret_key** に置き換えます。

例

```
[user@dev]$ aws configure --profile=ceph

AWS Access Key ID [None]: 12345
AWS Secret Access Key [None]: 67890
Default region name [None]:
Default output format [None]:
```

3. Ceph Object Gateway ノードの FQDN を指すエイリアスを作成します。

構文

```
alias aws="aws --endpoint-url=http://FQDN_OF_GATEWAY_NODE:8080"
```

FQDN_OF_GATEWAY_NODE は、Ceph Object Gateway ノードの FQDN に置き換えます。

例

```
[user@dev]$ alias aws="aws --endpoint-url=http://testclient.englab.pnq.redhat.com:8080"
```

4. 新しいバケットを作成します。

構文

```
aws --profile=MY_PROFILE_NAME s3api create-bucket --bucket BUCKET_NAME
```

MY_PROFILE_NAME を、このプロファイルを使用するために作成した名前に置き換えます。**BUCKET_NAME** は、新規バケットの名前に置き換えます。

例

```
[user@dev]$ aws --profile=ceph s3api create-bucket --bucket mybucket
```

5. 所有するバケットを一覧表示します。

構文

```
aws --profile=MY_PROFILE_NAME s3api list-buckets
```

MY_PROFILE_NAME を、このプロファイルを使用するために作成した名前に置き換えます。

例

```
[user@dev]$ aws --profile=ceph s3api list-buckets
{
    "Buckets": [
        {
            "Name": "mybucket",
            "CreationDate": "2021-08-31T16:46:15.257Z"
        }
    ],
    "Owner": {
        "DisplayName": "User",
        "ID": "user"
    }
}
```

6. MFA-Delete のバケットを設定します。

構文

```
aws --profile=MY_PROFILE_NAME s3api put-bucket-versioning --bucket BUCKET_NAME --versioning-configuration '{"Status":"Enabled","MFADelete":"Enabled"}' --mfa 'TOTP_SERIAL TOTP_PIN'
```

- **MY_PROFILE_NAME** を、このプロファイルを使用するために作成した名前に置き換えます。
- **BUCKET_NAME** を新しいバケットの名前に置き換えます。
- **TOTP_SERIAL** を TOTP トークンの ID を表す文字列に置き換え、 **TOTP_PIN** を MFA 認証デバイスに表示されている現在の PIN に置き換えます。
- **TOTP_SERIAL** は、S3 の radosgw ユーザーを作成したときに指定した文字列です。
- MFA TOTP トークンの作成に関する詳細は、**Red Hat Ceph Storage Object Gateway Configuration and Administration Guide** の Creating a new multi-factor authentication TOTP token セクションを参照してください。
- oathtool を使用した MFA シードの作成に関する詳細は、**Red Hat Ceph Storage 開発者ガイド** の **oathtool を使用した多要素認証のシードの作成** セクションを参照してください。

例

```
[user@dev]$ aws --profile=ceph s3api put-bucket-versioning --bucket mybucket --versioning-configuration '{"Status":"Enabled","MFADelete":"Enabled"}' --mfa 'MFAtest 232009'
```

7. バケットバージョン管理状態の MFA-Delete ステータスを表示します。

構文

```
aws --profile=MY_PROFILE_NAME s3api get-bucket-versioning --bucket BUCKET_NAME
```

MY_PROFILE_NAME を、このプロファイルを使用するために作成した名前に置き換えます。**BUCKET_NAME** を新しいバケットの名前に置き換えます。

例

```
[user@dev]$ aws --profile=ceph s3api get-bucket-versioning --bucket mybucket
{
    "Status": "Enabled",
    "MFADelete": "Enabled"
}
```

8. MFA-Delete が有効なバケットにオブジェクトを追加します。

構文

```
aws --profile=MY_PROFILE_NAME s3api put-object --bucket BUCKET_NAME --key OBJECT_KEY --body LOCAL_FILE
```

- **MY_PROFILE_NAME** を、このプロファイルを使用するために作成した名前に置き換えます。

- **BUCKET_NAME** を新しいバケットの名前に置き換えます。
- **OBJECT_KEY** を、バケット内のオブジェクトを一意に識別する名前に置き換えます。
- **LOCAL_FILE** を、アップロードするローカルファイルの名前に置き換えます。

例

```
[user@dev]$ aws --profile=ceph s3api put-object --bucket mybucket --key example --
body testfile
{
    "ETag": "\"5679b828547a4b44cfb24a23fd9bb9d5\"",
    "VersionId": "3VyyYPTEuIofdvMPWbr1znlOu7lJE3r"
}
```

9. 特定のオブジェクトのオブジェクトバージョンを一覧表示します。

構文

```
aws --profile=MY_PROFILE_NAME s3api list-object-versions --bucket BUCKET_NAME --
key OBJEC_KEY]
```

- **MY_PROFILE_NAME** を、このプロファイルを使用するために作成した名前に置き換えます。
- **BUCKET_NAME** を新しいバケットの名前に置き換えます。
- **OBJECT_KEY** を、バケット内のオブジェクトを一意に識別するために指定された名前に置き換えます。

例

```
[user@dev]$ aws --profile=ceph s3api list-object-versions --bucket mybucket --key
example
{
    "IsTruncated": false,
    "KeyMarker": "example",
    "VersionIdMarker": "",
    "Versions": [
        {
            "ETag": "\"5679b828547a4b44cfb24a23fd9bb9d5\"",
            "Size": 196,
            "StorageClass": "STANDARD",
            "Key": "example",
            "VersionId": "3VyyYPTEuIofdvMPWbr1znlOu7lJE3r",
            "IsLatest": true,
            "LastModified": "2021-08-31T17:48:45.484Z",
            "Owner": {
                "DisplayName": "User",
                "ID": "user"
            }
        }
    ],
    "Name": "mybucket",
    "Prefix": "",
```

```
    "MaxKeys": 1000,
    "EncodingType": "url"
}
```

10. MFA-Delete が有効なバケット内のオブジェクトを削除します。

構文

```
aws --profile=MY_PROFILE_NAME s3api delete-object --bucket BUCKET_NAME --key
OBJECT_KEY --version-id VERSION_ID --mfa 'TOTP_SERIAL TOTP_PIN'
```

- **MY_PROFILE_NAME** を、このプロファイルを使用するために作成した名前に置き換えます。
- **BUCKET_NAME** を、削除するオブジェクトを含むバケットの名前に置き換えます。
- **OBJECT_KEY** を、バケット内のオブジェクトを一意に識別する名前に置き換えます。
- **VERSION_ID** を、削除するオブジェクトの特定のバージョンの VersionID に置き換えます。
- **TOTP_SERIAL** を TOTP トークンの ID を表す文字列に置き換え、 **TOTP_PIN** を MFA 認証デバイスに表示されている現在の PIN に置き換えます。

例

```
[user@dev]$ aws --profile=ceph s3api delete-object --bucket mybucket --key example --
version-id 3VyyYPTEuIofdvMPWbr1znlOu7lJE3r --mfa 'MFAtest 420797'
{
    "VersionId": "3VyyYPTEuIofdvMPWbr1znlOu7lJE3r"
}
```

MFA トークンが含まれていない場合、リクエストは次のエラーで失敗します。

例

```
[user@dev]$ aws --profile=ceph s3api delete-object --bucket mybucket --key example --
version-id 3VyyYPTEuIofdvMPWbr1znlOu7lJE3r
An error occurred (AccessDenied) when calling the DeleteObject operation: Unknown
```

11. オブジェクトのバージョンを一覧表示して、MFA 削除が有効なバケットからオブジェクトが削除されたことを確認します。

構文

```
aws --profile=MY_PROFILE_NAME s3api list-object-versions --bucket BUCKET_NAME --
key OBJECT_KEY
```

- **MY_PROFILE_NAME** を、このプロファイルを使用するために作成した名前に置き換えます。
- **BUCKET_NAME** をバケットの名前に置き換えます。
- **OBJECT_KEY** を、バケット内のオブジェクトを一意に識別する名前に置き換えます。

例

```
[user@dev]$ aws --profile=ceph s3api list-object-versions --bucket mybucket --key
example
{
    "IsTruncated": false,
    "KeyMarker": "example",
    "VersionIdMarker": "",
    "Name": "mybucket",
    "Prefix": "",
    "MaxKeys": 1000,
    "EncodingType": "url"
}
```

2.3.10. oathtool コマンドを使用して多要素認証のシードを作成する

マルチファクター認証 (MFA) を設定するには、時間ベースのワンタイムパスワード (TOTP) ジェネレーターとバックエンド MFA システムで使用するシークレットシードを作成する必要があります。**oathtool** を使用して 16 進数のシードを生成し、オプションで **qrencode** を使用して QR コードを作成し、トークンを MFA デバイスにインポートできます。

前提条件

- Linux システム。
- コマンドラインシェルへのアクセス。
- Linux システムへの **root** または **sudo** アクセス。

手順

1. **oathtool** パッケージをインストールします。

```
[root@dev]# dnf install oathtool
```

2. **qrencode** パッケージをインストールします。

```
[root@dev]# dnf install qrencode
```

3. **urandom** Linux デバイスファイルから 30 文字のシードを生成し、シェル変数 **SEED** に格納します。

例

```
[user@dev]$ SEED=$(head -10 /dev/urandom | sha512sum | cut -b 1-30)
```

4. **SEED** 変数で echo を実行して、シードを出力します。

例

```
[user@dev]$ echo $SEED
BA6GLJBJIKC3D7W7YFYXXAQ7
```

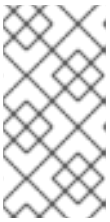
5. **SEED** を oathtool コマンドに入力します。

構文

```
oathtool -v -d6 $SEED
```

例

```
[user@dev]$ oathtool -v -d6 $SEED
Hex secret: 083c65a4294285b1fedfc1717b821f
Base32 secret: BA6GLJBJIKC3D7W7YFYXXAQ7
Digits: 6
Window size: 0
Start counter: 0x0 (0)

823182
```



注記

base32 シークレットは、MFA デバイスのオーセンティケーターアプリケーションにトークンを追加するために必要です。QR コードを使用してトークンを認証アプリケーションにインポートするか、base32 シークレットを使用して手動でトークンを追加できます。

6. オプション: QR コードイメージファイルを作成して、トークンをオーセンティケーターに追加します。

構文

```
qrencode -o /tmp/user.png 'otpauth://totp/TOTP_SERIAL?secret=_BASE32_SECRET'
```

TOTP_SERIAL を (TOTP) トークンの ID を表す文字列に置き換え、 **BASE32_SECRET** を oathtool によって生成された Base32 シークレットに置き換えます。

例

```
[user@dev]$ qrencode -o /tmp/user.png 'otpauth://totp/MFAtest?
secret=BA6GLJBJIKC3D7W7YFYXXAQ7'
```

7. 生成された QR コードイメージファイルをスキャンして、トークンを MFA デバイスの認証アプリケーションに追加します。

8. **radowgw-admin** コマンドを使用して、ユーザーの多要素認証 TOTP トークンを作成します。

関連情報

- MFA TOTP トークンの作成に関する詳細は、**Red Hat Ceph Storage Object Gateway Configuration and Administration Guide** の Creating a new multi-factor authentication TOTP token セクションを参照してください。

2.3.11. セキュアなトークンサービス

Amazon Web Services の Secure Token Service (STS) は、ユーザーを認証するための一時セキュリティー認証情報のセットを返します。Ceph Object Gateway は STS アプリケーションプログラミングインターフェイス (API) のサブセットを実装し、ID およびアクセス管理 (IAM) の一時的な認証情報を提供します。これらの一時的な認証情報を使用して、Ceph Object Gateway の STS エンジンを使用して S3 呼び出しを認証します。IAM ポリシーを使用すると、STS API に渡されるパラメーターである一時認証情報をさらに制限できます。

関連情報

- Amazon Web Services Secure Token Service の Welcome ページ。
- STS Lite および Keystone の詳細は、**Red Hat Ceph Storage 開発者ガイド**の **STS Lite および Keystone の設定および使用** セクションを参照してください。
- STS Lite および Keystone の制限の詳細は、**Red Hat Ceph Storage 開発者ガイド**の **Keystone のある STS Lite を使用する際の制限の回避** を参照してください。

2.3.11.1. Secure Token Service アプリケーションのプログラミングインターフェイス

Ceph Object Gateway は、以下の Secure Token Service (STS) アプリケーションプログラミングインターフェイス (API) を実装します。

AssumeRole

この API は、アカウント間のアクセスのための一時的な認証情報のセットを返します。これらの一時的な認証情報により、**Role** と、**AssumeRole** API で割り当てられるポリシーの両方に割り当てられるパーミッションポリシーを使用することができます。**RoleArn** および **RoleSessionName** リクエストパラメーターは必須ですが、他の要求パラメーターは任意です。

RoleArn

詳細

長さが 20 ~ 2048 文字の Amazon Resource Name (ARN) について想定するロール。

型

文字列

必須

はい

RoleSessionName

詳細

仮定するロールセッション名を特定します。ロールセッション名は、異なるプリンシパルや別の理由がロールを想定する場合にセッションを一意に識別できます。このパラメーターの値は、2 文字から 64 文字までです。**=**、**,**、**.**、**@**、および **-** 文字は使用できますが、スペースは使用できません。

型

文字列

必須

はい

ポリシー

詳細

インラインセッションで使用する JSON 形式の ID およびアクセス管理ポリシー (IAM)。このパラメーターの値は 1 文字から 2048 文字までです。

型

文字列

必須

いいえ

DurationSeconds

詳細

セッションの期間 (秒単位)。最小値は **900** 秒で、最大値は **43200** 秒です。デフォルト値は **3600** 秒です。

型

整数

必須

いいえ

ExternalId

詳細

別のアカウントのロールを想定する場合には、利用可能な場合は一意の外部識別子を指定します。このパラメーターの値は、2 文字から 1224 文字までになります。

型

文字列

必須

いいえ

SerialNumber

詳細

関連付けられたマルチファクター認証 (MFA) デバイスからのユーザーの識別番号。パラメーターの値は、9 文字から 256 文字までのハードウェアデバイスまたは仮想デバイスのシリアル番号になります。

型

文字列

必須

いいえ

TokenCode

詳細

信頼ポリシーに MFA が必要な場合は、マルチファクター認証 (MFA) デバイスから生成された値。MFA デバイスが必要で、このパラメーターの値が空または期限切れの場合には、**AssumeRole** の呼び出しは access denied エラーメッセージを返します。このパラメーターの値には、固定長は 6 文字です。

型

文字列

必須

いいえ

AssumeRoleWithWebIdentity

この API は、OpenID Connect や OAuth 2.0 アイデンティティープロバイダーなどのアプリケーションによって認証されたユーザーの一時認証情報のセットを返します。**RoleArn** および **RoleSessionName** リクエストパラメーターは必須ですが、他の要求パラメーターは任意です。

RoleArn

詳細

長さが 20 ~ 2048 文字の Amazon Resource Name (ARN) について想定するロール。

型

文字列

必須

はい

RoleSessionName

詳細

仮定するロールセッション名を特定します。ロールセッション名は、異なるプリンシパルや別の理由がロールを想定する場合にセッションを一意に識別できます。このパラメーターの値は、2 文字から 64 文字までです。**=**、**,**、**.**、**@**、および **-** 文字は使用できますが、スペースは使用できません。

型

文字列

必須

はい

ポリシー

詳細

インラインセッションで使用する JSON 形式の ID およびアクセス管理ポリシー (IAM)。このパラメーターの値は 1 文字から 2048 文字までです。

型

文字列

必須

いいえ

DurationSeconds

詳細

セッションの期間 (秒単位)。最小値は **900** 秒で、最大値は **43200** 秒です。デフォルト値は **3600** 秒です。

型

整数

必須

いいえ

ProviderId

詳細

アイデンティティープロバイダーからのドメイン名の完全修飾ホストコンポーネント。このパラメーターの値は、長さが 4 ~ 2048 文字の OAuth 2.0 アクセストークンでのみ有効です。

型

文字列

必須

いいえ

WebIdentityToken

詳細

アイデンティティープロバイダーから提供される OpenID Connect アイデンティティートークンまたは OAuth 2.0 アクセストークン。このパラメーターの値は、4 文字から 2048 文字までです。

型

文字列

必須

いいえ

関連情報

- 詳細は、**Red Hat Ceph Storage 開発者ガイド**の Secure Token Service API の使用例セクションを参照してください。
- Amazon Web Services Security Token Service (AssumeRole アクション)
- Amazon Web Services Security Token Service (AssumeRoleWithWebIdentity アクション)

2.3.11.2. セキュアなトークンサービスの設定

Ceph Ansible を使用して Ceph Object Gateway で使用する Secure Token Service (STS) を設定します。

注記

S3 と STS API は同じ名前空間に共存し、いずれも Ceph Object Gateway の同じエンドポイントからアクセスできます。

前提条件

- Ceph Ansible 管理ノード。
- 稼働中の Red Hat Ceph Storage クラスターがある。
- 実行中の Ceph Object Gateway。

手順

1. **group_vars/rgws.yml** ファイルを編集するために開きます。
   a. 以下の行を追加します。

```
rgw_sts_key = STS_KEY
rgw_s3_auth_use_sts = true
```

以下を置き換えます。

- **STS_KEY** は、セッショントークンの暗号化に使用される鍵に置き換えます。

2. **group_vars/rgws.yml** ファイルへの変更を保存します。
3. 適切な Ceph Ansible Playbook を再実行します。
   a. **ベアメタル** デプロイメント:

```
[user@admin ceph-ansible]$ ansible-playbook site.yml --limit rgws
```

   b. **コンテナー** デプロイメント:

```
[user@admin ceph-ansible]$ ansible-playbook site-docker.yml --limit rgws
```

関連情報

- STS API に関する詳細は、**Red Hat Ceph Storage 開発者ガイド**の Secure Token Service アプリケーションのプログラミングインターフェイス セクションを参照してください。

2.3.11.3. OpenID Connect プロバイダー用のユーザーの作成

Ceph Object Gateway と OpenID Connect Provider との間の信頼を確立するには、ユーザーエンティティーとロール信頼ポリシーを作成します。

前提条件

- Ceph Object Gateway ノードへのユーザーレベルのアクセス。

手順

1. 新しい Ceph ユーザーを作成します。

   構文

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

   例

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

2. Ceph ユーザー機能を設定します。

   構文

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

例

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

3. Secure Token Service (STS) API を使用してロール信頼ポリシーに条件を追加します。

構文

```
"{\"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

上記の構文例の **app_id** は、着信トークンの **AUD_FIELD** フィールドと一致させる必要があります。

関連情報

- Amazon の Web サイトの Obtaining the Root CA Thumbprint for an OpenID Connect Identity Provider を参照してください。
- STS API に関する詳細は、**Red Hat Ceph Storage 開発者ガイド**の Secure Token Service アプリケーションのプログラミングインターフェイス セクションを参照してください。
- 詳細は、**Red Hat Ceph Storage 開発者ガイド**の Secure Token Service API の使用例セクションを参照してください。

2.3.11.4. OpenID Connect プロバイダーのサムプリントの取得

OpenID Connect プロバイダー (IDP) の設定ドキュメントを取得するには、以下を実行します。

前提条件

- **openssl** パッケージおよび **curl** パッケージのインストール。

手順

1. IDP の URL から設定ドキュメントを取得します。

構文

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

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. IDP 証明書を取得します。

構文

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

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 .
```

3. 直前のコマンドから x5c 応答の結果をコピーし、それを **certificate.crt** ファイルに貼り付けます。冒頭に **—–BEGIN CERTIFICATE—–**、末尾に **—–END CERTIFICATE—–** を含めます。

4. 証明書のサムプリントを取得します。

構文

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

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. SHA1 フィンガープリントからコロンをすべて削除し、IAM 要求で IDP エンティティーを作成するための入力として使用します。

関連情報

- Amazon の Web サイトの Obtaining the Root CA Thumbprint for an OpenID Connect Identity Provider を参照してください。
- STS API に関する詳細は、**Red Hat Ceph Storage 開発者ガイド**の Secure Token Service アプリケーションのプログラミングインターフェイス セクションを参照してください。
- 詳細は、**Red Hat Ceph Storage 開発者ガイド**の Secure Token Service API の使用例セクションを参照してください。

2.3.11.5. Keystone での STS Lite の設定および使用 (テクノロジープレビュー)

Amazon Secure Token Service (STS) と S3 API は、同じ名前空間に共存します。STS オプションは、Keystone オプションと組み合わせて設定できます。

注記

S3 と STS の API の両方に、Ceph Object Gateway の同じエンドポイントを使用してアクセスできます。

前提条件

- Red Hat Ceph Storage 3.2 以降
- 実行中の Ceph Object Gateway。
- Boto Python モジュールのバージョン 3 以降のインストール

手順

1. **group_vars/rgws.yml** ファイルを以下のオプションで開き、編集します。

```
rgw_sts_key = STS_KEY
rgw_s3_auth_use_sts = true
```

以下を置き換えます。

- **STS_KEY** は、セッショントークンの暗号化に使用される鍵に置き換えます。

2. 適切な Ceph Ansible Playbook を再実行します。

a. **ベアメタル** デプロイメント:

```
[user@admin ceph-ansible]$ ansible-playbook site.yml --limit rgws
```

b. **コンテナー** デプロイメント:

```
[user@admin ceph-ansible]$ ansible-playbook site-docker.yml --limit rgws
```

3. EC2 認証情報を生成します。

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                        |
+------------+---------------------------------------------------------+
```

4. 生成された認証情報を使用して、**GetSessionToken** API を使用して一時的なセキュリティー認証情報のセットを取得します。

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
)
```

5. 一時認証情報の取得は、S3 呼び出しの作成に使用できます。

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'],
  )
```

6. 新しい **S3Access** ロールを作成し、ポリシーを設定します。

   a. 管理 CAPS でユーザーを割り当てます。

   構文

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

   例

   ```
   [user@client]$ radosgw-admin caps add --uid="gwadmin" --caps="roles=*"
   ```

   b. **S3Access** ロールを作成します。

構文

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

例

```
[user@client]$ 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. S3Access ロールにパーミッションポリシーを割り当てます。

構文

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

例

```
[user@client]$ 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. 別のユーザーが **gwadmin** ユーザーのロールを想定できるようになりました。たとえば、**gwuser** ユーザーは、**gwadmin** ユーザーのパーミッションを想定できます。

e. 仮定ユーザーの **access_key** および **secret_key** の値を書き留めておきます。

例

```
[user@client]$ radosgw-admin user info --uid=gwuser | grep -A1 access_key
```

7. AssumeRole API 呼び出しを使用し、仮定のユーザーから **access_key** および **secret_key** の値を提供します。

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 には **S3Access** ロールが必要です。

関連情報

- Boto Python モジュールのインストールに関する詳細は、**Red Hat Ceph Storage Object Gateway ガイド**の S3 アクセスのテスト セクションを参照してください。
- 詳細は、**Red Hat Ceph Storage Object Gateway ガイド**の ユーザーの作成 セクションを参照してください。

2.3.11.6. Keystone で STS Lite を使用するための制限の回避 (テクノロジープレビュー)

Keystone の制限は、STS リクエストをサポートしないことです。もう 1 つの制限は、ペイロードハッシュがリクエストに含まれていないことです。この 2 つの制限を回避するには、Boto 認証コードを変更する必要があります。

前提条件

- 稼働中の Red Hat Ceph Storage クラスター (バージョン 3.2 以降)。
- 実行中の Ceph Object Gateway。
- Boto Python モジュールのバージョン 3 以降のインストール

手順

1. Boto の **auth.py** ファイルを開いて編集します。
   a. 以下の 4 つの行をコードブロックに追加します。

```
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. 以下の 2 つの行をコードブロックに追加します。

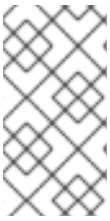
```
def _modify_request_before_signing(self, request):
        if 'Authorization' in request.headers:
            del request.headers['Authorization']
        self._set_necessary_date_headers(request)
        if self.credentials.token:
            if 'X-Amz-Security-Token' in request.headers:
      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

関連情報

- Boto Python モジュールのインストールに関する詳細は、**Red Hat Ceph Storage Object Gateway ガイド**の S3 アクセスのテスト セクションを参照してください。

2.3.12. STS での属性ベースのアクセス制御 (ABAC) のセッションタグ

セッションタグは、ユーザーのフェデレーション中に渡すことができるキーと値のペアです。これらは、セッションで **aws:PrincipalTag** として渡されるか、セキュアトークンサービス (STS) によって返される一時的な認証情報として渡されます。これらのプリンシパルタグは、Web トークンの一部として提供されるセッションタグと、引き受けるロールに関連付けられたタグで設定されます。



注記

現在、セッションタグは、**AssumeRoleWithWebIdentity** に渡される Web トークンの一部としてのみサポートされています。

タグは常に次の名前空間で指定する必要があります: **https://aws.amazon.com/tags**

重要

フェデレーションユーザーによって渡された Web トークンにセッションタグが含まれている場合、信頼ポリシーには **sts:TagSession** 権限が必要です。それ以外の場合、**AssumeRoleWithWebIdentity** アクションは失敗します。

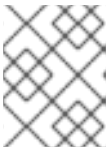
sts:TagSession を使用した信頼ポリシーの例:

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

プロパティー

セッションタグのプロパティーは次のとおりです。

- セッションタグは複数値にすることができます。



注記

複数値のセッションタグは、Amazon Webservice (AWS) ではサポートされていません。

- Keycloak は、最大 50 個のセッションタグを持つ OpenID Connect ID プロバイダー (IDP) として設定できます。
- 許可されるキーの最大サイズは 128 文字です。
- 許可される値の最大サイズは 256 文字です。
- タグまたは値は **aws:** で開始できません。

関連情報

- セキュアトークンサービスの詳細については、**Red Hat Ceph Storage 開発者ガイド** の セキュアトークンサービス のセクションを参照してください。

2.3.12.1. タグキー

以下は、ロール信頼ポリシーまたはロール許可ポリシーで使用できるタグキーです。

aws:RequestTag

説明

リクエストで渡されたキーと値のペアを、ロールの信頼ポリシーのキーと値のペアと比較します。
AssumeRoleWithWebIdentity の場合、セッションタグはロール信頼ポリシーで

aws:RequestTag として使用できます。これらのセッションタグは、Web トークンで Keycloak によって渡されます。その結果、フェデレーションユーザーはロールを引き受けることができます。

aws:PrincipalTag

説明

プリンシパルに関連付けられたキーと値のペアをポリシーのキーと値のペアと比較します。**AssumeRoleWithWebIdentity** の場合、セッションタグは、ユーザーが認証されると、一時的な認証情報にプリンシパルタグとして表示されます。これらのセッションタグは、Web トークンで Keycloak によって渡されます。これらは、ロールのアクセス許可ポリシーで **aws:PrincipalTag** として使用できます。

iam:ResourceTag

説明

リソースにアタッチされたキーと値のペアをポリシーのキーと値のペアと比較します。**AssumeRoleWithWebIdentity** の場合、ロールにアタッチされたタグが信頼ポリシーのタグと比較され、ユーザーがロールを引き受けることができます。

注記

Ceph Object Gateway は、ロールのタグ付け、タグの一覧表示、およびタグ付け解除アクションのための RESTful API をサポートするようになりました。

aws:TagKeys

説明

リクエスト内のタグとポリシー内のタグを比較します。**AssumeRoleWithWebIdentity** の場合、タグは、ユーザーがロールを引き受けることを許可される前に、ロール信頼ポリシーまたはアクセス許可ポリシーでタグキーをチェックするために使用されます。

s3:ResourceTag

説明

S3 リソース (バケットまたはオブジェクト) に存在するタグを、ロールのアクセス許可ポリシーのタグと比較します。
これは、Ceph Object Gateway で S3 操作を承認するために使用できます。ただし、これは AWS では許可されていません。

オブジェクトやバケットに付けられたタグを参照するためのキーです。オブジェクトまたはバケットに使用できる RESTful API を使用して、タグをオブジェクトまたはバケットにアタッチできます。

2.3.12.2. S3 リソースタグ

次のリストは、特定の操作を承認するためにサポートされている S3 リソースタグタイプを示しています。

タグタイプ: オブジェクトタグ

操作

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

タグタイプ: バケットタグ

操作

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

タグタイプ: バケット ACL のバケットタグ、オブジェクト ACL のオブジェクトタグ

操作

GetACLs、PutACLs

タグタイプ: ソースオブジェクトのオブジェクトタグ、宛先バケットのバケットタグ

操作

PutObject、CopyObject

2.4. S3 バケット操作

開発者は、Ceph Object Gateway 経由で Amazon S3 アプリケーションプログラミングインターフェイス (API) を使用してバケット操作を実行できます。

以下の表は、バケットの Amazon S3 機能操作と関数のサポートステータスを示しています。

表2.2 バケット操作

機能	状態	注記
バケットの一覧表示	サポート対象	
バケットの作成	サポート対象	固定 ACL のさまざまなセット。
バケットライフサイクル	一部サポート対象	**Expiration**、**NoncurrentVersionExpiration** および **AbortIncompleteMultipartUpload** がサポートされます。

機能	状態	注記
PUT バケットライフサイクル	一部サポート対象	**Expiration**、**NoncurrentVersionExpiration** および **AbortIncompleteMultipartUpload** がサポートされます。
バケットライフサイクルの削除	サポート対象	
バケットオブジェクトの取得	サポート対象	
バケットの場所	サポート対象	
バケットバージョンの取得	サポート対象	
バケットバージョンの送信	サポート対象	
バケットの削除	サポート対象	
バケット ACL の取得	サポート対象	固定 ACL のさまざまなセット
バケット ACL の送信	サポート対象	固定 ACL のさまざまなセット
バケットに関する CORS 設定情報を取得	サポート対象	
バケットに対し CORS 設定を行う	サポート対象	
バケットの CORS 設定を削除	サポート対象	
バケットオブジェクトバージョンの一覧表示	サポート対象	
HEAD バケット	サポート対象	
バケットマルチパートアップロードの一覧表示	サポート対象	
バケットポリシー	一部サポート対象	
バケットリクエストの支払いの取得	サポート対象	
バケットリクエストの支払いを行う	サポート対象	
マルチテナントバケット操作	サポート対象	

2.4.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- RESTful クライアント。

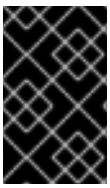
2.4.2. S3 create bucket notifications

バケットレベルでバケット通知を作成します。通知設定には、Red Hat Ceph Storage Object Gateway S3 イベント (**ObjectCreated** および **ObjectRemoved**) があります。これらは公開され、バケット通知を送信する宛先である必要があります。バケット通知は S3 オペレーションです。

s3:objectCreate および **s3:objectRemove** イベントバケット通知を作成するには、PUT を使用します。

例

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



重要

Red Hat は、**ObjectCreate** イベント (例: **put**、**post**、**multipartUpload**、および **copy**) をサポートします。また、Red Hat は、**object_delete**、**s3_multi_object_delete** などの **ObjectRemove** イベントをサポートしています。

リクエストエンティティー

NotificationConfiguration

詳細

TopicConfiguration エンティティーのリスト。

型

Container

Required

はい

TopicConfiguration

詳細

イベントエンティティーの **Id**、**Topic**、および **list**。

型

Container

Required

はい

id

詳細

通知の名前。

型

文字列

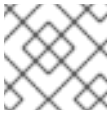
必須

はい

トピック

説明

トピック Amazon リソース名 (ARN)



注記

トピックは事前に作成する必要があります。

タイプ

文字列

必須

はい

Event

詳細

サポートされるイベントの一覧。複数のイベントエンティティーを使用できます。省略すると、すべてのイベントが処理されます。

型

文字列

必須

いいえ

フィルター

詳細

S3Key、**S3Metadata**、および **S3Tags** エンティティー。

型

Container

Required

いいえ

S3Key

詳細

オブジェクトキーに基づくフィルタリングの **FilterRule** エンティティーの一覧。3 つのエンティティーが一覧に含まれる場合があります。たとえば、**Name** は、**prefix**、**suffix**、または **regex** になります。リスト内のフィルタールールはすべて、フィルターが一致するために一致している必要があります。

型

Container

Required

いいえ

S3Metadata

詳細

オブジェクトメタデータに基づくフィルタリングの **FilterRule** エンティティーの一覧。リスト内のフィルタールールはすべて、オブジェクトで定義されたメタデータと一致する必要があります。ただし、フィルターにリストされていない他のメタデータエントリーがある場合には、オブジェクトは一致するままになります。

型

Container

Required

いいえ

S3Tags

詳細

オブジェクトタグに基づいてフィルタリングする **FilterRule** エンティティーの一覧。リスト内のフィルタールールはすべて、オブジェクトで定義されたタグと一致する必要があります。ただし、フィルターに他のタグがリストされていない場合、オブジェクトは引き続き一致します。

型

Container

Required

いいえ

S3Key.FilterRule

詳細

Name エンティティーおよび **Value** エンティティーです。Name は、**prefix**、**suffix**、または **regex** です。**Value** は、キー接頭辞、キー接尾辞、またはキーに一致する正規表現を保持します。

型

Container

Required

はい

S3Metadata.FilterRule

詳細

Name エンティティーおよび **Value** エンティティーです。Name は、メタデータ属性の名前です (例: **x-amz-meta-xxx**)。この値は、この属性で想定される値になります。

型

Container

Required

はい

S3Tags.FilterRule

詳細

Name エンティティーおよび **Value** エンティティーです。Name はタグキーで、値はタグの値です。

型

Container

Required

はい

HTTP レスポンス

400

ステータスコード

MalformedXML

詳細

XML は適していません。

400

ステータスコード

InvalidArgument

詳細

ID がないか、トピック ARN がないか無効であるか、イベントが無効です。

404

ステータスコード

NoSuchBucket

詳細

バケットが存在しません。

404

ステータスコード

NoSuchKey

詳細

トピックが存在しません。

id="s3-get-bucket-notifications_dev"]

2.4.3. S3 get bucket notifications

特定の通知を取得するか、バケットに設定されたすべての通知を一覧表示します。

構文

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

例

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

レスポンスの例

```
<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>
```

注記

notification サブリソースはバケット通知設定または空の **NotificationConfiguration** 要素を返します。呼び出し元はバケットの所有者である必要があります。

リクエストエンティティー

notification-id

詳細

通知の名前。ID が指定されていない場合は、すべての通知が一覧表示されます。

型

String

NotificationConfiguration

詳細

TopicConfiguration エンティティーのリスト。

型

Container

Required

はい

TopicConfiguration

詳細

イベントエンティティーの **Id**、**Topic**、および **list**。

型

Container

Required

はい

id

詳細

通知の名前。

型

文字列

必須

はい

トピック

説明

トピック Amazon リソース名 (ARN)

注記

トピックは事前に作成する必要があります。

タイプ

文字列

必須

はい

Event

詳細

処理されたイベント。複数のイベントエンティティーが存在する可能性があります。

型

文字列

必須

はい

フィルター

詳細

指定の設定のフィルター。

型

Container

Required

いいえ

HTTP レスポンス

404

ステータスコード

NoSuchBucket

詳細

バケットが存在しません。

404

ステータスコード

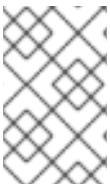
NoSuchKey

詳細

通知は、提供された場合に存在しません。

2.4.4. S3 delete bucket notifications

バケットから特定の通知またはすべての通知を削除します。



注記

通知の削除は、S3 通知 API の拡張機能です。バケットで定義された通知は、バケットの削除時に削除されます。不明な通知 (例: **double delete**) を削除しても、エラーとは見なされません。

特定の通知またはすべての通知を削除するには、DELETE を使用します。

構文

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

例

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

リクエストエンティティー

notification-id

詳細

通知の名前。通知 ID が指定されていない場合は、バケットのすべての通知が削除されます。

型

String

HTTP レスポンス

404

ステータスコード

NoSuchBucket

詳細

バケットが存在しません。

2.4.5. バケットのホスト名へのアクセス

バケットにアクセスするモードは 2 つあります。最初のメソッドは推奨されるメソッドで、バケットを URI の最上位ディレクトリーとして識別します。

例

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

2 番目のメソッドは、仮想バケットのホスト名経由でバケットを識別します。

例

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

ヒント

2 番目の方法では高価なドメイン認定と DNS ワイルドカードが必要なため、Red Hat は最初の方法を推奨します。

2.4.6. S3 list buckets

GET / は、ユーザーがリクエストを行うユーザーが作成するバケットの一覧を返します。**GET** / は、認証ユーザーが作成したバケットのみを返します。匿名のリクエストを行うことはできません。

構文

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

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

表2.3 レスポンスエンティティー

Name	型	詳細
Buckets	コンテナー	バケットの一覧用のコンテナー。
Bucket	コンテナー	バケット情報用のコンテナー
名前	文字列	バケット名。
CreationDate	Date	バケットが作成された時点の UTC 時間。
ListAllMyBucketsResult	コンテナー	結果のコンテナー。
Owner	コンテナー	バケット所有者の **ID** および **DisplayName** のコンテナー。
ID	文字列	バケット所有者の ID。
DisplayName	文字列	バケットの所有者の表示名。

2.4.7. S3 はバケットオブジェクトの一覧を返します。

バケットオブジェクトの一覧を返します。

構文

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

表2.4 パラメーター

名前	型	詳細
prefix	文字列	指定された接頭辞が含まれるオブジェクトのみを返します。
delimiter	文字列	接頭辞と他のオブジェクト名の間に挿入される区切り文字。
marker	文字列	返されるオブジェクトリストの開始インデックス。
max-keys	整数	返すキーの最大数。デフォルトは 1000 です。

表2.5 HTTP レスポンス

HTTP ステータス	ステータスコード	詳細
200	OK	取得されるバケット

GET /BUCKET は、以下のフィールドが含まれるバケットのコンテナーを返します。

表2.6 バケットレスポンスエンティティー

Name	型	詳細
ListBucketResult	エンティティー	オブジェクト一覧のコンテナー。
名前	文字列	コンテンツが返されるバケットの名前。
Prefix	文字列	オブジェクトキーの接頭辞。
Marker	文字列	返されるオブジェクトリストの開始インデックス。
MaxKeys	整数	返されるキーの最大数。
Delimiter	文字列	設定されている場合は、同じ接頭辞を持つオブジェクトが **CommonPrefixes** リストに表示されます。
IsTruncated	ブール値	**true** の場合、バケットの内容のサブセットのみが返されます。
CommonPrefixes	コンテナー	複数のオブジェクトに同じ接頭辞が含まれる場合は、この一覧に表示されます。

ListBucketResult にはオブジェクトが含まれ、各オブジェクトは **Contents** コンテナー内にあります。

表2.7 オブジェクトレスポンスエンティティー

Name	型	詳細
内容	Object	オブジェクトのコンテナー。
Key	文字列	オブジェクトのキー。
LastModified	Date	オブジェクトの最後に変更した日時。
ETag	文字列	オブジェクトの MD-5 ハッシュ (entity タグ)
サイズ	整数	オブジェクトのサイズ。

Name	型	詳細
StorageClass	文字列	常に **STANDARD** を返す必要があります。

2.4.8. S3 による新規バケットの作成

新規バケットを作成します。バケットを作成するには、要求を認証するためにユーザー ID および有効な AWS アクセスキー ID が必要です。バケットを匿名ユーザーとして作成することはできません。

制約

通常、バケット名はドメイン名の制約に従う必要があります。

- バケット名は一意である必要があります。
- バケット名は最初に指定し、小文字で終了する必要があります。
- バケット名にはダッシュ (-) を含めることができます。

構文

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

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

表2.8 パラメーター

名前	説明	有効な値	必須
x-amz-acl	固定 ACL。	**private**、**public-read**、**public-read-write**、**authenticated-read**	いいえ

HTTP レスポンス

バケット名が一意で、制約内で未使用であると、操作は成功します。同じ名前のバケットがすでに存在し、ユーザーがバケット所有者である場合は、操作が成功します。バケット名が使用中の場合は、操作が失敗します。

HTTP ステータス	ステータスコード	詳細
409	BucketAlreadyExists	バケットは、異なるユーザーの所有権に存在します。

2.4.9. S3 バケットの削除

バケットを削除します。バケットの削除が正常に行われた後にバケット名を再利用できます。

構文

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

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

表2.9 HTTP レスポンス

HTTP ステータス	ステータスコード	詳細
204	コンテンツなし	バケットが削除されました。

2.4.10. S3 bucket lifecycle

バケットのライフサイクル設定を使用してオブジェクトを管理し、そのオブジェクトが有効期間中効果的に保存されるようにすることができます。Ceph Object Gateway の S3 API は、AWS バケットライフサイクルアクションのサブセットをサポートします。

- **Expiration**: これはバケット内のオブジェクトの有効期間を定義します。オブジェクトが存続する日数または有効期限がかかり、その時点で Ceph Object Gateway がオブジェクトを削除します。バケットがバージョン管理を有効にしない場合、Ceph Object Gateway はオブジェクトを永続的に削除します。バケットがバージョン管理を有効化する場合、Ceph Object Gateway は現行バージョンの削除マーカーを作成し、現行バージョンを削除します。

- **NoncurrentVersionExpiration**: これはバケット内の最新バージョン以外のオブジェクトバージョンのライフサイクルを定義します。この機能を使用するには、バケットがバージョン管理を有効にする必要があります。最新バージョン以外のオブジェクトが存続する日数を取ります。この時点では、Ceph Object Gateway が最新バージョン以外のオブジェクトを削除します。

- **AbortIncompleteMultipartUpload**: これは、非完全なマルチパートアップロードが中止されるまでの日数を定義します。

ライフサイクル設定には、**<Rule>** 要素を使用した 1 つ以上のルールが含まれます。

例

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

ライフサイクルルールは、ライフサイクルルールに指定する **<Filter>** 要素に基づいてバケットの全オブジェクトまたはサブセットに適用できます。フィルターは複数の方法を指定できます。

- キーの接頭辞

- オブジェクトタグ

- キー接頭辞と 1 つ以上のオブジェクトタグの両方

キーの接頭辞

ライフサイクルルールは、キー名の接頭辞に基づいてオブジェクトのサブセットに適用できます。たとえば、**<keypre/>** を指定すると、**keypre/** で始まるオブジェクトに適用されます。

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

異なるキー接頭辞を持つオブジェクトに、異なるライフサイクルルールを適用することもできます。

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

オブジェクトタグ

ライフサイクルルールは、**<Key>** 要素および **<Value>** 要素を使用して、特定のタグを持つオブジェクトにのみ適用できます。

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

接頭辞および 1 つ以上のタグの両方

ライフサイクルルールでは、キーの接頭辞と 1 つ以上のタグの両方に基づいてフィルターを指定できます。これらは **<And>** 要素でラップする必要があります。フィルターには 1 つの接頭辞と、ゼロまたは複数のタグのみを使用できます。

```
<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>
```

関連情報

- バケットライフサイクルの取得に関する詳細は、**Red Hat Ceph Storage 開発者ガイド**の バケットライフサイクルの取得 を参照してください。
- バケットライフサイクルの作成に関する詳細は、**Red Hat Ceph Storage 開発者ガイド**を バケットライフサイクルの作成 を参照してください。
- バケットライフサイクルの削除に関する詳細は、**Red Hat Ceph Storage 開発者ガイド**の バケットライフサイクルの削除 を参照してください。

2.4.11. S3 GET bucket lifecycle

バケットのライフサイクルを取得するには、**GET** を使用して宛先バケットを指定します。

構文

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

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

リクエストヘッダー

詳細は、一般的なリクエストヘッダー を参照してください。

レスポンス

レスポンスには、バケットライフサイクルとその要素が含まれます。

2.4.12. S3 create or replace a bucket lifecycle

バケットライフサイクルを作成または置き換えるには、**PUT** を使用して宛先バケットとライフサイクル設定を指定します。Ceph Object Gateway は、S3 ライフサイクル機能のサブセットのみをサポートします。

構文

```
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>
```

表2.10 リクエストヘッダー

Name	説明	有効な値	必須
content-md5	メッセージの base64 でエンコードされた MD-5 ハッシュ	文字列。デフォルトや制約はありません。	No

関連情報

- 一般的な Amazon S3 リクエストヘッダーに関する詳細は、**Red Hat Ceph Storage 開発者ガイド**の Amazon S3 リクエストヘッダー を参照してください。
- Amazon S3 バケットライフサイクルに関する詳細は、**Red Hat Ceph Storage 開発者ガイド**の Amazon S3 バケットライフサイクル を参照してください。

2.4.13. S3 delete a bucket lifecycle

バケットライフサイクルを削除するには、**DELETE** を使用し、宛先バケットを指定します。

構文

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

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

リクエストヘッダー

リクエストには特別な要素が含まれません。

レスポンス

レスポンスは、一般的なレスポンスのステータスを返します。

関連情報

- Amazon S3 の一般的なリクエストヘッダーは、付録 A を参照してください。
- Amazon S3 の一般的なレスポンスステータスコードは、付録 B を参照してください。

2.4.14. S3 get bucket location

バケットのゾーングループを取得します。これを呼び出すには、ユーザーはバケット所有者である必要があります。PUT 要求時に **LocationConstraint** を指定して、バケットをゾーングループに制限できます。

以下のように **location** サブリソースをバケットリソースに追加します。

構文

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

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

表2.11 レスポンスエンティティー

Name	型	詳細
LocationConstraint	文字列	バケットが存在するゾーングループ (デフォルトゾーングループ用の空の文字列)

2.4.15. S3 によるバケットのバージョン管理を取得

バケットのバージョン状態を取得します。これを呼び出すには、ユーザーはバケット所有者である必要があります。

以下のように、**versioning** サブリソースをバケットリソースに追加します。

構文

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

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

2.4.16. S3 によるバケットのバージョン管理の設定

このサブリソースは、既存のバケットのバージョン管理状態を設定します。バージョン管理状態を設定するには、バケット所有者である必要があります。バージョン管理状態がバケットに設定されていないと、バージョンは管理されていません。GET バージョン管理リクエストを実行しても、バージョン管理状態の値は返されません。

バケットによるバージョン管理の状態を設定します。

Enabled: バケットのオブジェクトのバージョン管理を有効にします。バケットに追加したすべてのオブジェクトは、一意のバージョン ID を受信します。**Suspended**: バケットのオブジェクトのバージョン管理を無効にします。バケットに追加したすべてのオブジェクトは、バージョン ID の Null を受け取ります。

構文

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

表2.12 バケット要求のエンティティー

Name	型	詳細
VersioningConfiguration	コンテナー	要求のコンテナー。
状態	文字列	バケットのバージョン状態を設定します。有効な値: Suspended/Enabled

2.4.17. S3 でバケットのアクセス制御リストを取得

バケットのアクセス制御リストを取得します。ユーザーはバケットの所有者である必要があります。または、バケットで **READ_ACP** パーミッションが付与されている必要があります。

以下のように、**acl** サブリソースをバケット要求に追加します。

構文

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

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

表2.13 レスポンスエンティティー

Name	型	詳細
AccessControlPolicy	コンテナー	レスポンスのコンテナー。
AccessControlList	コンテナー	ACL 情報用のコンテナー
Owner	コンテナー	バケット所有者の **ID** および **DisplayName** のコンテナー。
ID	文字列	バケット所有者の ID。
DisplayName	文字列	バケットの所有者の表示名。
Grant	コンテナー	**Grantee** および **Permission** のコンテナー。
Grantee	コンテナー	パーミッションを付与されるユーザーの **DisplayName** および **ID** のコンテナー。

Name	型	詳細
Permission	文字列	**Grantee** バケットに指定されるパーミッション。

2.4.18. S3 でバケットのアクセス制御リストを取得

既存のバケットへのアクセス制御を設定します。ユーザーはバケットの所有者である必要があります。または、バケットの **WRITE_ACP** パーミッションが付与されている必要があります。

以下のように、**acl** サブリソースをバケット要求に追加します。

構文

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

表2.14 リクエストエンティティー

Name	型	詳細
AccessControlPolicy	コンテナー	要求のコンテナー。
AccessControlList	コンテナー	ACL 情報用のコンテナー
Owner	コンテナー	バケット所有者の **ID** および **DisplayName** のコンテナー。
ID	文字列	バケット所有者の ID。
DisplayName	文字列	バケットの所有者の表示名。
Grant	コンテナー	**Grantee** および **Permission** のコンテナー。
Grantee	コンテナー	パーミッションを付与されるユーザーの **DisplayName** および **ID** のコンテナー。
Permission	文字列	**Grantee** バケットに指定されるパーミッション。

2.4.19. S3 ではバケットの CORS 設定を取得

バケットに設定された CORS 設定情報を取得します。ユーザーはバケットの所有者である必要があります。または、バケットで **READ_ACP** パーミッションが付与されている必要があります。

以下に示すように、**cors** サブリソースをバケット要求に追加します。

構文

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

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

2.4.20. S3 put bucket cors

バケットの CORS 設定を設定します。ユーザーはバケットの所有者である必要があります。または、バケットで **READ_ACP** パーミッションが付与されている必要があります。

以下に示すように、**cors** サブリソースをバケット要求に追加します。

構文

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

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

2.4.21. S3 delete a bucket cors

バケットに設定された CORS 設定情報を削除します。ユーザーはバケットの所有者である必要があります。または、バケットで **READ_ACP** パーミッションが付与されている必要があります。

以下に示すように、**cors** サブリソースをバケット要求に追加します。

構文

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

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

2.4.22. S3 list bucket object versions

バケット内のすべてのバージョンのオブジェクトに関するメタデータの一覧を返します。バケットへの READ アクセスが必要です。

以下のように **versions** サブリソースをバケット要求に追加します。

構文

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

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

GET /BUCKET?versions のパラメーターを指定できますが、いずれも不要です。

表2.15 パラメーター

名前	型	詳細
prefix	文字列	指定の接頭辞が含まれるキーが含まれる進行中のアップロードを返します。
delimiter	文字列	接頭辞と他のオブジェクト名の間に挿入される区切り文字。
key-marker	文字列	アップロード一覧の最初のマーカー。
max-keys	整数	進行中のアップロードの最大数。デフォルト値は 1000 です。
version-id-marker	文字列	リストを開始するオブジェクトバージョンを指定します。

表2.16 レスポンスエンティティー

Name	型	詳細
KeyMarker	文字列	**key-marker** リクエストパラメーターによって指定されるキーマーカー (ある場合)。
NextKeyMarker	文字列	**IsTruncated** が **true** の場合に後続のリクエストで使用するキーマーカー。
NextUploadIdMarker	文字列	**IsTruncated** が **true** の場合に後続のリクエストで使用するアップロード ID マーカー。
IsTruncated	ブール値	**true** の場合は、バケットのアップロードコンテンツのサブセットのみが返されます。
サイズ	整数	アップロードした部分のサイズ。
DisplayName	文字列	所有者の表示名。
ID	文字列	所有者の ID。
Owner	コンテナー	オブジェクトを所有するユーザーの **ID** および **DisplayName** のコンテナー。
StorageClass	文字列	作成されるオブジェクトを保存するために使用されるメソッド。**STANDARD** または **REDUCED_REDUNDANCY**

Name	型	詳細
バージョン	コンテナー	バージョン情報のコンテナー
versionId	文字列	オブジェクトのバージョン ID。
versionIdMarker	文字列	省略されたレスポンスのキーの最後のバージョン。

2.4.23. S3 ヘッドバケット

バケットで HEAD を呼び出して、存在する場合は、呼び出し元にアクセス権限があるかどうかを判断します。バケットが存在し、呼び出し元にパーミッションがある場合は **200 OK** を返します。バケットが存在しない場合は **404 Not Found**、バケットが存在しますが呼び出し元にはアクセスパーミッションがない場合は **403 Forbidden** を返します。

構文

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

2.4.24. S3 list multipart uploads

GET /?uploads は、現在の進行中のマルチパートアップロードの一覧を返します。つまり、アプリケーションは複数パートごとのアップロードを開始しますが、サービスがすべてのアップロードを完了しているわけではありません。

構文

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

GET /BUCKET?uploads のパラメーターを指定できますが、いずれも不要です。

表2.17 パラメーター

名前	型	詳細
prefix	文字列	指定の接頭辞が含まれるキーが含まれる進行中のアップロードを返します。
delimiter	文字列	接頭辞と他のオブジェクト名の間に挿入される区切り文字。

名前	型	詳細
key-marker	文字列	アップロード一覧の最初のマーカー。
max-keys	整数	進行中のアップロードの最大数。デフォルト値は 1000 です。
max-uploads	整数	マルチパートアップロードの最大数。1-1000 の範囲。デフォルト値は 1000 です。
version-id-marker	文字列	**key-marker** が指定されていない場合は無視されます。辞書式順序で、またはその **ID** 以降でリストする最初のアップロードの **ID** を指定します。

表2.18 レスポンスエンティティー

Name	型	詳細
ListMultipartUploadsResult	コンテナー	結果のコンテナー
ListMultipartUploadsResult.Prefix	文字列	**prefix** 要求パラメーターで指定される接頭辞 (存在する場合)。
Bucket	文字列	バケットのコンテンツを受け取るバケット。
KeyMarker	文字列	**key-marker** リクエストパラメーターによって指定されるキーマーカー (ある場合)。
UploadIdMarker	文字列	**upload-id-marker** リクエストパラメーターによって指定されるマーカー (存在する場合)。
NextKeyMarker	文字列	**IsTruncated** が **true** の場合に後続のリクエストで使用するキーマーカー。
NextUploadIdMarker	文字列	**IsTruncated** が **true** の場合に後続のリクエストで使用するアップロード ID マーカー。
MaxUploads	整数	**max-uploads** リクエストパラメーターで指定される最大アップロード数。
Delimiter	文字列	設定されている場合は、同じ接頭辞を持つオブジェクトが **CommonPrefixes** リストに表示されます。
IsTruncated	ブール値	**true** の場合は、バケットのアップロードコンテンツのサブセットのみが返されます。

Name	型	詳細
Upload	コンテナー	**Key**、**UploadId**、**InitiatorOwner**、**StorageClass**、および **Initiated** 要素のコンテナー。
Key	文字列	マルチパートアップロードが完了した後のオブジェクトのキー。
UploadId	文字列	マルチパートアップロードを識別する **ID**。
Initiator	コンテナー	アップロードを開始したユーザーの **ID** と **DisplayName** が含まれます。
DisplayName	文字列	イニシエーターの表示名。
ID	文字列	イニシエーターの ID。
Owner	コンテナー	アップロードしたオブジェクトを所有するユーザーの **ID** および **DisplayName** のコンテナー。
StorageClass	文字列	作成されるオブジェクトを保存するために使用されるメソッド。**STANDARD** または **REDUCED_REDUNDANCY**
Initiated	Date	ユーザーがアップロードを開始した日時。
CommonPrefixes	コンテナー	複数のオブジェクトに同じ接頭辞が含まれる場合は、この一覧に表示されます。
CommonPrefixes.Prefix	文字列	**prefix** リクエストパラメーターで定義されている接頭辞の後にキーのサブ文字列。

2.4.25. S3 バケットポリシー

Ceph Object Gateway は、バケットに適用される Amazon S3 ポリシー言語のサブセットをサポートします。

作成および削除

Ceph Object Gateway は、CLI ツール **radosgw-admin** を使用するのではなく、標準の S3 操作を使用して S3 バケットポリシーを管理します。

管理者は、**s3cmd** コマンドを使用してポリシーを設定または削除できます。

例

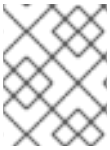
```
$ cat > examplepol
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {"AWS": ["arn:aws:iam::usfolks:user/fred"]},
    "Action": "s3:PutObjectAcl",
    "Resource": [
      "arn:aws:s3:::happybucket/*"
    ]
  }]
}

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

制限事項

Ceph Object Gateway がサポートするのは以下の S3 アクションだけです。

- **s3:AbortMultipartUpload**
- **s3:CreateBucket**
- **s3:DeleteBucketPolicy**
- **s3:DeleteBucket**
- **s3:DeleteBucketWebsite**
- **s3:DeleteObject**
- **s3:DeleteObjectVersion**
- **s3:GetBucketAcl**
- **s3:GetBucketCORS**
- **s3:GetBucketLocation**
- **s3:GetBucketPolicy**
- **s3:GetBucketRequestPayment**
- **s3:GetBucketVersioning**
- **s3:GetBucketWebsite**
- **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:PutLifecycleConfiguration**
- **s3:PutObjectAcl**
- **s3:PutObject**
- **s3:PutObjectVersionAcl**



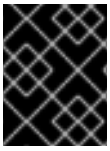
注記

Ceph Object Gateway は、ユーザー、グループ、またはロールへのポリシー設定をサポートしません。

Ceph Object Gateway は、Amazon の 12 桁のアカウント ID の代わりに RGW の tenant 識別子を使用します。Amazon Web Service (AWS) S3 と Ceph Object Gateway S3 との間でポリシーを使用する場合、Ceph Object Gateway は、ユーザーの作成時に Amazon アカウント ID をテナント ID として使用する必要があります。

AWS S3 では、すべてのテナントが単一の名前空間を共有します。対照的に、Ceph Object Gateway はすべてのテナントにバケットの独自の名前空間を提供します。現在、別のテナントに属するバケットにアクセスしようとしている Ceph Object Gateway クライアントは、S3 リクエストの **tenant:bucket** としてそれを処理する必要があります。

AWS では、バケットポリシーは別のアカウントへのアクセスを許可し、そのアカウントの所有者はユーザーパーミッションを持つ個々のユーザーにアクセス権限を付与できます。Ceph Object Gateway はユーザー、ロール、およびグループのパーミッションをサポートしていません。そのため、アカウントの所有者は個々のユーザーに直接アクセスを付与する必要があります。



重要

アカウント全体のアクセスをバケットに付与すると、そのアカウントのすべてのユーザーにアクセス権限が付与されます。

バケットポリシーは文字列の補正を **サポートしません**。

Ceph Object Gateway では、以下の条件キーがサポートされます。

- **aws:CurrentTime**
- **aws:EpochTime**
- **aws:PrincipalType**
- **aws:Referer**
- **aws:SecureTransport**
- **aws:SourceIp**
- **aws:UserAgent**
- **aws:username**

Ceph Object Gateway **のみ** は、**ListBucket** アクションの以下の条件キーをサポートします。

- **s3:prefix**
- **s3:delimiter**
- **s3:max-keys**

Swift への影響

Ceph Object Gateway は、Swift API にバケットポリシーを設定する機能はありません。ただし、S3 API で設定されているバケットポリシーは Swift と S3 のいずれの操作も管理します。

Ceph Object Gateway は、ポリシーで指定されたプリンシパルに対して Swift の認証情報と一致します。

2.4.26. S3 get the request payment configuration on a bucket

requestPayment サブリソースを使用してバケットの要求支払い設定を返します。ユーザーはバケットの所有者である必要があります。または、バケットで **READ_ACP** パーミッションが付与されている必要があります。

以下のように **requestPayment** サブリソースをバケット要求に追加します。

構文

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

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

2.4.27. S3 set the request payment configuration on a bucket

requestPayment サブリソースを使用してバケットの要求支払い設定を設定します。デフォルトでは、バケットの所有者はバケットからのダウンロードに対して支払います。この設定パラメーターにより、バケットの所有者は、ダウンロードを要求するすべてのユーザーが要求に対して要求およびバケットからダウンロードに対して課金されることを指定できます。

以下のように **requestPayment** サブリソースをバケット要求に追加します。

構文

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

表2.19 リクエストエンティティー

Name	型	詳細
Payer	列挙	ダウンロードおよびリクエストの費用の課金を指定します。
RequestPaymentConfiguration	コンテナー	**Payer** のコンテナー。

2.4.28. マルチテナントバケット操作

クライアントアプリケーションがバケットにアクセスする場合は、常に特定ユーザーの認証情報で動作します。Red Hat Ceph Storage クラスターでは、すべてのユーザーがテナントに属します。そのため、テナントが明示的に指定されていない場合は、すべてのバケット操作のコンテキストに暗黙的なテナントがあります。したがって、マルチテナンシーは、参照されるバケットと参照ユーザーが同じテナントに属する限り、以前のリリースと完全に後方互換性があります。

明示的なテナントの指定に使用される拡張機能は、使用されるプロトコルおよび認証システムによって異なります。

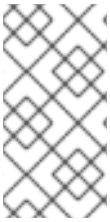
以下の例では、コロン文字はテナントとバケットを分離します。そのため、URL のサンプルは以下のようになります。

```
https://rgw.domain.com/tenant:bucket
```

一方、単純な Python の例は、バケットメソッド自体でテナントとバケットを分離します。

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")
```



注記

ホスト名に、コロンや、バケット名では有効ではない他の区切り文字を含めることができないため、マルチテナンシーを使用して S3 形式のサブドメインを使用することはできません。期間を使用するとあいまいな構文が作成されます。そのため、**bucket-in-URL-path** 形式をマルチテナンシーと併用する必要があります。

関連情報

- 詳細は、マルチテナンシー を参照してください。

2.4.29. 関連情報

- バケット Web サイトの設定に関する詳細は、**Red Hat Ceph Storage Object Gateway 設定および管理ガイド** を参照してください。

2.5. S3 オブジェクト操作

開発者は、Ceph Object Gateway 経由で Amazon S3 アプリケーションプログラミングインターフェイス (API) を使用してオブジェクト操作を行うことができます。

以下の表は、関数のサポートステータスとともに、オブジェクトの Amazon S3 の機能操作を示しています。

表2.20 オブジェクト操作

Get Object	サポート対象	
Get Object Information	サポート対象	
Put Object	サポート対象	
Delete Object	サポート対象	
Delete Multiple Objects	サポート対象	
Get Object ACLs	サポート対象	
Put Object ACLs	サポート対象	
Copy Object	サポート対象	
Post Object	サポート対象	
Options Object	サポート対象	
Initiate Multipart Upload	サポート対象	
Add a Part to a Multipart Upload	サポート対象	

Get Object	サポート対象	
List Parts of a Multipart Upload	サポート対象	
Assemble Multipart Upload	サポート対象	
Copy Multipart Upload	サポート対象	
Abort Multipart Upload	サポート対象	
マルチテナンシー	サポート対象	

2.5.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- RESTful クライアント。

2.5.2. S3 get an object from a bucket

バケットからオブジェクトを取得します。

構文

```
GET /BUCKET/OBJECT HTTP/1.1
```

versionId サブリソースを追加して、オブジェクトの特定のバージョンを取得します。

構文

```
GET /BUCKET/OBJECT?versionId=VERSION_ID HTTP/1.1
```

表2.21 リクエストヘッダー

Name	説明	有効な値	必須
範囲	取得するオブジェクトの範囲。	範囲: bytes=beginbyte-endbyte	No
if-modified-since	タイムスタンプ以降に変更した場合にのみ取得します。	タイムスタンプ	No
if-unmodified-since	タイムスタンプ以降変更されていない場合にのみ取得します。	タイムスタンプ	No
if-match	オブジェクトの ETag が ETag と一致する場合にのみ取得します。	エンティティータグ	No

Name	説明	有効な値	必須
if-none-match	オブジェクトの ETag が ETag と一致する場合にのみ取得します。	エンティティータグ	No

表2.22 レスポンスヘッダー

Name	説明
Content-Range	データ範囲 (範囲ヘッダーフィールドがリクエストに指定された場合のみを返します)。
x-amz-version-id	バージョン ID または Null を返します。

2.5.3. S3 get information on an object

オブジェクトに関する情報を返します。この要求は Get Object 要求と同じヘッダー情報を返しますが、オブジェクトデータペイロードではなくメタデータのみが含まれます。

オブジェクトの現行バージョンを取得します。

構文

```
HEAD /BUCKET/OBJECT HTTP/1.1
```

versionId サブリソースを追加して、特定バージョンの情報を取得します。

構文

```
HEAD /BUCKET/OBJECT?versionId=VERSION_ID HTTP/1.1
```

表2.23 リクエストヘッダー

Name	説明	有効な値	必須
範囲	取得するオブジェクトの範囲。	範囲: bytes=beginbyte-endbyte	No
if-modified-since	タイムスタンプ以降に変更した場合にのみ取得します。	タイムスタンプ	No
if-unmodified-since	タイムスタンプ以降変更されていない場合にのみ取得します。	タイムスタンプ	No
if-match	オブジェクトの ETag が ETag と一致する場合にのみ取得します。	エンティティータグ	No
if-none-match	オブジェクトの ETag が ETag と一致する場合にのみ取得します。	エンティティータグ	No

Name	説明	有効な値	必須

表2.24 レスポンスヘッダー

Name	説明
x-amz-version-id	バージョン ID または Null を返します。

2.5.4. S3 によりオブジェクトをバケットに追加

オブジェクトをバケットに追加します。この操作を実行するには、バケットに書き込みパーミッションが必要です。

構文

```
PUT /BUCKET/OBJECT HTTP/1.1
```

表2.25 リクエストヘッダー

Name	説明	有効な値	必須
content-md5	メッセージの base64 でエンコードされた MD-5 ハッシュ	文字列。デフォルトや制約はありません。	No
content-type	標準の MIME タイプ。	MIME タイプ。デフォルト: **binary/octet-stream**	いいえ
x-amz-meta-<...>	ユーザーのメタデータ。オブジェクトとともに保存されます。	8kb までの文字列。デフォルトはありません。	No
x-amz-acl	固定 ACL。	**private**、**public-read**、**public-read-write**、**authenticated-read**	いいえ

表2.26 レスポンスヘッダー

Name	説明
x-amz-version-id	バージョン ID または Null を返します。

2.5.5. S3 delete an object

オブジェクトを削除します。含まれるバケットに WRITE パーミッションを設定する必要があります。

オブジェクトを削除します。オブジェクトのバージョン管理が有効なの場合、マーカーが作成されます。

構文

```
DELETE /BUCKET/OBJECT HTTP/1.1
```

バージョン管理が有効な場合にオブジェクトを削除するには、**versionId** サブリソースおよび削除するオブジェクトのバージョンを指定する必要があります。

```
DELETE /BUCKET/OBJECT?versionId=VERSION_ID HTTP/1.1
```

2.5.6. S3 delete multiple objects

この API 呼び出しは、バケットから複数のオブジェクトを削除します。

構文

```
POST /BUCKET/OBJECT?delete HTTP/1.1
```

2.5.7. S3 get an object's Access Control List (ACL)

オブジェクトの現行バージョンの ACL を返します。

構文

```
GET /BUCKET/OBJECT?acl HTTP/1.1
```

versionId サブリソースを追加して、特定バージョンの ACL を取得します。

構文

```
GET /BUCKET/OBJECT?versionId=VERSION_ID&acl HTTP/1.1
```

表2.27 レスポンスヘッダー

Name	説明
x-amz-version-id	バージョン ID または Null を返します。

表2.28 レスポンスエンティティー

Name	型	詳細
AccessControlPolicy	コンテナー	レスポンスのコンテナー。
AccessControlList	コンテナー	ACL 情報用のコンテナー
Owner	コンテナー	オブジェクトの所有者の **ID** および **DisplayName** のコンテナー。
ID	文字列	オブジェクトの所有者の ID。
DisplayName	文字列	オブジェクトの所有者の表示名。
Grant	コンテナー	**Grantee** および **Permission** のコンテナー。
Grantee	コンテナー	パーミッションを付与されるユーザーの **DisplayName** および **ID** のコンテナー。
Permission	文字列	**Grantee** オブジェクトに指定されたパーミッション。

2.5.8. S3 によりオブジェクトのアクセス制御リスト (ACL) の設定

オブジェクトの現行バージョンのオブジェクト ACL を設定します。

構文

```
PUT /BUCKET/OBJECT?acl
```

表2.29 リクエストエンティティー

Name	型	詳細
AccessControlPolicy	コンテナー	レスポンスのコンテナー。
AccessControlList	コンテナー	ACL 情報用のコンテナー
Owner	コンテナー	オブジェクトの所有者の **ID** および **DisplayName** のコンテナー。
ID	文字列	オブジェクトの所有者の ID。
DisplayName	文字列	オブジェクトの所有者の表示名。

Name	型	詳細
Grant	コンテナー	**Grantee** および **Permission** のコンテナー。
Grantee	コンテナー	パーミッションを付与されるユーザーの **DisplayName** および **ID** のコンテナー。
Permission	文字列	**Grantee** オブジェクトに指定されたパーミッション。

2.5.9. S3 によるオブジェクトのコピー

オブジェクトをコピーするには、**PUT** を使用して宛先バケットとオブジェクト名を指定します。

構文

```
PUT /DEST_BUCKET/DEST_OBJECT HTTP/1.1
x-amz-copy-source: SOURCE_BUCKET/SOURCE_OBJECT
```

表2.30 リクエストヘッダー

Name	説明	有効な値	必須
x-amz-copy-source	ソースバケット名 + オブジェクト名。	BUCKET/OBJECT	はい
x-amz-acl	固定 ACL。	**private**、**public-read**、**public-read-write**、**authenticated-read**	いいえ
x-amz-copy-if-modified-since	タイムスタンプ以降に変更された場合のみコピーします。	タイムスタンプ	No
x-amz-copy-if-unmodified-since	タイムスタンプ以降変更されていない場合にのみコピーします。	タイムスタンプ	No
x-amz-copy-if-match	オブジェクトの ETag が ETag と一致する場合に限りコピーします。	エンティティータグ	No
x-amz-copy-if-none-match	オブジェクトの ETag が一致しない場合にのみコピーします。	エンティティータグ	No

表2.31 レスポンスエンティティー

Name	型	説明
CopyObjectResult	コンテナー	レスポンス要素のコンテナー。
LastModified	Date	ソースオブジェクトを最後に変更した日付。
Etag	文字列	新規オブジェクトの ETag。

関連情報

- <additional resource 1>
- <additional resource 2>

2.5.10. S3 により HTML フォームを使用してオブジェクトをバケットに追加

HTML フォームを使用してオブジェクトをバケットに追加します。この操作を実行するには、バケットに書き込みパーミッションが必要です。

構文

```
POST /BUCKET/OBJECT HTTP/1.1
```

2.5.11. S3 determine options for a request

特定の送信元、HTTP メソッド、およびヘッダーを使用して実際のリクエストを送信できるかどうかを判断するための事前要求です。

構文

```
OPTIONS /OBJECT HTTP/1.1
```

2.5.12. S3 initiate a multipart upload

複数パートからなるアップロードプロセスを開始します。追加部分の追加、パーツの一覧表示、および複数パートアップロードの完了または破棄時に指定できる **UploadId** を返します。

構文

```
POST /BUCKET/OBJECT?uploads
```

表2.32 リクエストヘッダー

Name	説明	有効な値	必須
content-md5	メッセージの base64 でエンコードされた MD-5 ハッシュ	文字列。デフォルトや制約はありません。	No

Name	説明	有効な値	必須
content-type	標準の MIME タイプ。	MIME タイプ。デフォルト: **binary/octet-stream**	いいえ
x-amz-meta-<...>	ユーザーのメタデータ。オブジェクトとともに保存されます。	8kb までの文字列。デフォルトはありません。	No
x-amz-acl	固定 ACL。	**private**、**public-read**、**public-read-write**、**authenticated-read**	いいえ

表2.33 レスポンスエンティティー

Name	型	詳細
InitiatedMultipartUploadsResult	コンテナー	結果のコンテナー
Bucket	文字列	オブジェクトの内容を受け取るバケット。
Key	文字列	**key** リクエストパラメーターで指定されるキー (存在する場合)。
UploadId	文字列	**upload-id** リクエストパラメーターで指定される ID で、マルチパートアップロードを特定します (存在する場合)。

2.5.13. S3 がマルチパートアップロードに部分を追加

マルチパートアップロードに部分を追加します。

複数パートのアップロードに部分を追加するために **uploadId** サブリソースとアップロード ID を指定します。

構文

```
PUT /BUCKET/OBJECT?partNumber=&uploadId=UPLOAD_ID HTTP/1.1
```

以下の HTTP レスポンスが返されます。

表2.34 HTTP レスポンス

HTTP ステータス	ステータスコード	詳細
404	NoSuchUpload	指定した upload-id がこのオブジェクトで開始されたアップロードと一致しません。

2.5.14. S3 でマルチパートアップロードの一部が一覧表示されます。

マルチパートアップロードの一部を一覧表示するために **uploadId** サブリソースとアップロード ID を指定します。

構文

```
GET /BUCKET/OBJECT?uploadId=UPLOAD_ID HTTP/1.1
```

表2.35 レスポンスエンティティー

Name	型	詳細
InitiatedMultipartUploadsResult	コンテナー	結果のコンテナー
Bucket	文字列	オブジェクトの内容を受け取るバケット。
Key	文字列	**key** リクエストパラメーターで指定されるキー (存在する場合)。
UploadId	文字列	**upload-id** リクエストパラメーターで指定される ID で、マルチパートアップロードを特定します (存在する場合)。
Initiator	コンテナー	アップロードを開始したユーザーの **ID** と **DisplayName** が含まれます。
ID	文字列	イニシエーターの ID。
DisplayName	文字列	イニシエーターの表示名。
Owner	コンテナー	アップロードしたオブジェクトを所有するユーザーの **ID** および **DisplayName** のコンテナー。
StorageClass	文字列	作成されるオブジェクトを保存するために使用されるメソッド。**STANDARD** または **REDUCED_REDUNDANCY**
PartNumberMarker	文字列	**IsTruncated** が **true** の場合に後続のリクエストで使用する部分マーカー。一覧の先頭に指定します。
NextPartNumberMarker	文字列	**IsTruncated** が **true** の場合は、後続のリクエストで使用する次の部分マーカー。リストの末尾。

Name	型	詳細
MaxParts	整数	**max-parts** リクエストパラメーターで指定されたレスポンスで許可される最大部分。
IsTruncated	ブール値	**true** の場合は、オブジェクトのアップロードコンテンツのサブセットのみが返されます。
Part	コンテナー	**Key**、**Part**、**InitiatorOwner**、**StorageClass**、および **Initiated** 要素のコンテナー。
PartNumber	整数	部分の識別番号。
ETag	文字列	コンポーネントのエンティティータグです。
サイズ	整数	アップロードした部分のサイズ。

2.5.15. S3 アップロードした部分のアセンブル

アップロードした部分を組み立て、新規オブジェクトを作成します。これにより、複数パートのアップロードが実行されます。

複数パートからなるアップロードを完了するには、**uploadId** サブリソースとアップロード ID を指定します。

構文

```
POST /BUCKET/OBJECT?uploadId=UPLOAD_ID HTTP/1.1
```

表2.36 リクエストエンティティー

Name	型	説明	必須
CompleteMultipartUpload	コンテナー	1 つ以上の部分で設定されるコンテナー。	はい
Part	コンテナー	**PartNumber** および **ETag** のコンテナー。	はい
PartNumber	整数	部分の識別子。	はい
ETag	文字列	コンポーネントのエンティティータグです。	Yes

表2.37 レスポンスエンティティー

Name	型	詳細
CompleteMultipartUploadResult	コンテナー	レスポンスのコンテナー。
場所	URI	新規オブジェクトのリソース識別子 (パス)。
Bucket	文字列	新規オブジェクトが含まれるバケットの名前。
Key	文字列	オブジェクトのキー。
ETag	文字列	新規オブジェクトのエンティティータグ。

2.5.16. S3 によるマルチパートのアップロード

既存のオブジェクトからデータをデータソースとしてコピーして、パーツをアップロードします。

複数パートからなるアップロードコピーを実行するには、**uploadId** サブリソースとアップロード ID を指定します。

構文

```
PUT /BUCKET/OBJECT?partNumber=PartNumber&uploadId=UPLOAD_ID HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

表2.38 リクエストヘッダー

Name	説明	有効な値	必須
x-amz-copy-source	ソースバケット名およびオブジェクト名。	BUCKET/OBJECT	はい
x-amz-copy-source-range	ソースオブジェクトからコピーするバイトの範囲。	範囲: **bytes=first-last** (ここで、最初のおよび最後は、コピーするゼロベースのバイトオフセットです)たとえば、**bytes=0-9** は、ソースの最初の 10 バイトをコピーすることを示しています。	いいえ

表2.39 レスポンスエンティティー

Name	型	詳細
CopyPartResult	コンテナー	すべてのレスポンス要素のコンテナー。
ETag	文字列	新しい部分の ETag を返します。

Name	型	詳細
LastModified	文字列	最後に変更した日付を返します。

.Additional Resources

- この機能の詳細は、Amazon S3 のサイト を参照してください。

2.5.17. S3 abort a multipart upload

複数パートアップロードを中止します。

マルチパートによるアップロードを中止するために **uploadId** サブリソースとアップロード ID を指定します。

構文

```
DELETE /BUCKET/OBJECT?uploadId=UPLOAD_ID HTTP/1.1
```

2.5.18. S3 Hadoop interoperability

HDFS (Hadoop Distributed File System) のアクセスを必要とするデータ解析アプリケーションは、Hadoop 用の Apache S3A コネクターを使用して Ceph Object Gateway にアクセスできます。S3A コネクターは、データが Ceph Object Gateway に保存される一方で、HDFS ファイルシステムがアプリケーションへのセマンティクスを読み取りおよび書き込みする S3 互換のオブジェクトストレージを HDFS ファイルシステムとして表示するオープンソースツールです。

Ceph Object Gateway は、Hadoop 2.7.3 に同梱される S3A コネクターと完全に互換性があります。

2.5.19. 関連情報

- マルチテナンシーの詳細は、Red Hat Ceph Storage Object Gateway 設定および管理ガイドを参照してください。

2.6. 関連情報

- Amazon S3 の一般的なリクエストヘッダーは、付録 A を参照してください。
- Amazon S3 の一般的なレスポンスステータスコードは、付録 B を参照してください。
- サポートされていないヘッダーフィールドは、付録 C を参照してください。

第3章 CEPH OBJECT GATEWAY および SWIFT API

開発者は、Swift API データアクセスモデルと互換性のある RESTful アプリケーションプログラミングインターフェイス (API) を使用できます。Ceph Object Gateway を使用して、Red Hat Ceph Storage クラスターに保存されているバケットおよびオブジェクトを管理できます。

以下の表は、現在の Swift 機能機能のサポート状況を示しています。

表3.1 機能

機能	状態	備考
認証	サポート対象	
アカウントメタデータの取得	サポート対象	カスタムメタデータなし
Swift ACL	サポート対象	Swift ACL のサブセットに対応
コンテナーの一覧表示	サポート対象	
コンテナーのオブジェクト一覧の表示	サポート対象	
コンテナーの作成	サポート対象	
コンテナーの削除	サポート対象	
コンテナーメタデータの取得	サポート対象	
コンテナーメタデータの追加/更新	サポート対象	
コンテナーメタデータの削除	サポート対象	
オブジェクトの取得	サポート対象	
オブジェクトの作成/更新	サポート対象	
大規模オブジェクトの作成	サポート対象	
オブジェクトの削除	サポート対象	
オブジェクトのコピー	サポート対象	
オブジェクトメタデータの取得	サポート対象	
オブジェクトメタデータの追加/更新	サポート対象	
一時 URL 操作	サポート対象	

機能	状態	備考
CORS	サポート対象外	
オブジェクトの期限設定	サポート対象	
オブジェクトのバージョン管理	サポート対象外	
静的な Web サイト	サポート対象外	

3.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- RESTful クライアント。

3.2. SWIFT API の制限

重要

以下の制限事項を使用してください。お使いのハードウェアの選択には影響があるため、この要件を Red Hat アカウントチームと常に相談してください。

- **Swift API を使用する場合の最大オブジェクトサイズ:** 5GB
- **Swift API を使用する場合のメタデータの最大サイズ:** オブジェクトに適用できるユーザーメタデータの合計サイズに定義された制限はありませんが、単一の HTTP 要求は 16,000 バイトに制限されます。

3.3. SWIFT ユーザーの作成

Swift インターフェイスをテストするには、Swift サブユーザーを作成します。Swift ユーザーの作成は 2 つの手順です。最初のステップでは、ユーザーを作成します。次のステップでは、秘密鍵を作成します。

注記

マルチサイトのデプロイメントでは、マスターゾーングループのマスターゾーンにあるホストでユーザーを作成します。

前提条件

- Ceph Object Gateway のインストール
- Ceph Object Gateway ノードへのルートレベルのアクセス。

手順

1. Swift ユーザーを作成します。

構文

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

NAME を Swift ユーザー名に置き換えます。以下に例を示します。

例

```
[root@rgw]# radosgw-admin subuser create --uid=testuser --subuser=testuser:swift --
access=full
{
    "user_id": "testuser",
    "display_name": "First User",
    "email": "",
    "suspended": 0,
    "max_buckets": 1000,
    "auid": 0,
    "subusers": [
        {
            "id": "testuser:swift",
            "permissions": "full-control"
        }
    ],
    "keys": [
        {
            "user": "testuser",
            "access_key": "O8JDE41XMI74O185EHKD",
            "secret_key": "i4Au2yxG5wtr1JK01mI8kjJPM93HNAoVWOSTdJd6"
        }
    ],
    "swift_keys": [
        {
            "user": "testuser:swift",
            "secret_key": "13TLtdEW7bCqgttQgPzxFxziu0AgabtOc6vM8DLA"
        }
    ],
    "caps": [],
    "op_mask": "read, write, delete",
    "default_placement": "",
    "placement_tags": [],
    "bucket_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
    "user_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
```

```
    "temp_url_keys": [],
    "type": "rgw"
}
```

2. シークレットキーを作成します。

構文

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

NAME を Swift ユーザー名に置き換えます。以下に例を示します。

例

```
[root@rgw]# radosgw-admin key create --subuser=testuser:swift --key-type=swift --gen-
secret
{
    "user_id": "testuser",
    "display_name": "First User",
    "email": "",
    "suspended": 0,
    "max_buckets": 1000,
    "auid": 0,
    "subusers": [
        {
            "id": "testuser:swift",
            "permissions": "full-control"
        }
    ],
    "keys": [
        {
            "user": "testuser",
            "access_key": "O8JDE41XMI74O185EHKD",
            "secret_key": "i4Au2yxG5wtr1JK01mI8kjJPM93HNAoVWOSTdJd6"
        }
    ],
    "swift_keys": [
        {
            "user": "testuser:swift",
            "secret_key": "a4ioT4jEP653CDcdU8p4OuhruwABBRZmyNUbnSSt"
        }
    ],
    "caps": [],
    "op_mask": "read, write, delete",
    "default_placement": "",
    "placement_tags": [],
    "bucket_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
    "user_quota": {
        "enabled": false,
```

```
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
    "temp_url_keys": [],
    "type": "rgw"
}
```


3.4. ユーザーの SWIFT 認証

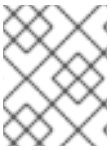
ユーザーを認証するには、**X-Auth-User** および **X-Auth-Key** を含むリクエストを作成します。

構文

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

レスポンスの例

```
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
```



注記

認証中に **X-Storage-Url** 値を使用して **GET** リクエストを実行すると、Ceph の Swift 互換サービスに関するデータを取得できます。

関連情報

- Swift リクエストヘッダーは Red Hat Ceph Storage 開発者ガイドを参照してください。
- Swift レスポンスヘッダーは Red Hat Ceph Storage 開発者ガイドを参照してください。

3.5. SWIFT コンテナー操作

開発者は、Ceph Object Gateway 経由で Swift アプリケーションのプログラミングインターフェイス (API) を使用してコンテナーの操作を行うことができます。コンテナーを一覧表示、作成、更新、および削除できます。コンテナーのメタデータを追加または更新できます。

3.5.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- RESTful クライアント。

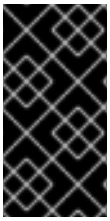
3.5.2. Swift コンテナー操作

コンテナーは、データオブジェクトを格納するメカニズムです。アカウントには多くのコンテナーを持たせることができますが、コンテナー名は一意でなければなりません。この API により、クライアントはコンテナーの作成、アクセス制御およびメタデータの設定、コンテナーのコンテンツの取得、およびコンテナーの削除を行うことができます。この API は特定のユーザーのアカウントの情報に関連するリクエストを行うため、コンテナーのアクセス制御が意図的に公開されていない限り、つまり匿名のリクエストを許可しない限り、この API のすべてのリクエストを認証する必要があります。

注記

Amazon S3 API はバケットという用語を使用してデータコンテナーを記述します。Swift API 内のバケットを参照すると、バケットという用語はコンテナーという用語と同じものになります。

オブジェクトストレージの 1 つは、階層パスやディレクトリーをサポートしないことです。代わりに、各コンテナーにオブジェクトがある 1 つ以上のコンテナーで設定される 1 つのレベルをサポートします。RADOS Gateway の Swift 互換 API は、疑似階層コンテナーの概念をサポートします。これは、オブジェクトの命名を使用してコンテナーをエミュレートする手段で、ストレージシステムで実際には実装されません。たとえば、photos/buildings/empire-state.jpg のように、疑似階層名でオブジェクトに名前を付けることができますが、コンテナー名にスラッシュ (/) 文字を含めることはできません。



重要

バージョン付けされた Swift コンテナーに大規模なオブジェクトをアップロードする場合は、**python-swiftclient** ユーティリティーで **--leave-segments** オプションを使用します。**--leave-segments** を使用しないと、マニフェストファイルが上書きされます。したがって、既存のオブジェクトは上書きされ、データが失われることになります。

3.5.3. Swift でコンテナーのアクセス制御リスト (ACL) の更新

ユーザーがコンテナーを作成すると、ユーザーはデフォルトでコンテナーへの読み取り/書き込みアクセスを持ちます。その他のユーザーがコンテナーのコンテンツを読み取りしたり、コンテナーに書き込むことを許可するには、ユーザーを明示的に有効にする必要があります。**X-Container-Read** または **X-Container-Write** に * を指定することもできます。これにより、すべてのユーザーがコンテナーから読み取るか、コンテナーへの書き込みが可能になります。* を設定すると、コンテナーが公開されます。これにより、匿名ユーザーがコンテナーから読み込むか、コンテナーに書き込むことができます。

構文

```
POST /AP_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
```

表3.2 リクエストヘッダー

Name	説明	型	必須
X-Container-Read	コンテナーの読み取りパーミッションを持つユーザー ID。	ユーザー ID のコンマ区切りの文字列値。	いいえ

Name	説明	型	必須
X-Container-Write	コンテナーの書き込みパーミッションを持つユーザー ID。	ユーザー ID のコンマ区切りの文字列値。	No

3.5.4. Swift 一覧コンテナー

API バージョンを指定し、アカウントは特定のユーザーアカウントのコンテナーの一覧を返す **GET** リクエスト。リクエストは特定のユーザーのコンテナーを返すため、リクエストには認証トークンが必要です。リクエストは匿名で行われません。

構文

```
GET /API_VERSION/ACCOUNT HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

表3.3 リクエストパラメーター

Name	説明	型	必須	有効な値
limit	結果の数を指定の値に制限します。	整数	No	該当なし
format	結果の形式を定義します。	文字列	いいえ	**json** または **xml**
marker	マーカー値よりも大きな結果の一覧を返します。	文字列	No	該当なし

レスポンスにはコンテナーの一覧が含まれるか、HTTP 204 レスポンスコードで返されます。

表3.4 レスポンスエンティティー

Name	説明	型
account	アカウント情報の一覧。	コンテナー
container	コンテナーの一覧。	コンテナー
name	コンテナーの名前。	文字列
bytes	コンテナーのサイズ。	整数

3.5.5. Swift でコンテナーオブジェクトの一覧表示

コンテナー内のオブジェクトを一覧表示するには、API バージョン、アカウント、およびコンテナーの名前を使用して **GET** リクエストを行います。クエリーパラメーターを指定して完全なリストをフィルタリングしたり、パラメーターを除外してコンテナーに保存されている最初の 10,000 オブジェクト名の一覧を返すこともできます。

構文

```
GET /AP_VERSION/TENANT:CONTAINER HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

表3.5 パラメーター

名前	説明	タイプ	有効な値	必須
format	結果の形式を定義します。	文字列	**json** または **xml**	いいえ
prefix	結果を、指定した接頭辞で始まるオブジェクトに制限します。	文字列	該当なし	いいえ
marker	マーカー値よりも大きな結果の一覧を返します。	文字列	該当なし	いいえ
limit	結果の数を指定の値に制限します。	整数	0 - 10,000	いいえ
delimiter	接頭辞と他のオブジェクト名の間に挿入される区切り文字。	文字列	該当なし	いいえ
path	オブジェクトの擬似階層パス。	文字列	該当なし	No

表3.6 レスポンスエンティティー

Name	説明	型
container	コンテナー	コンテナー
object	コンテナー内のオブジェクト。	コンテナー
name	コンテナー内のオブジェクトの名前。	文字列
hash	オブジェクトのコンテンツのハッシュコード。	文字列

Name	説明	型
last_modified	オブジェクトの内容を最後に変更した時間。	Date
content_type	オブジェクト内のコンテンツのタイプ。	文字列

3.5.6. Swift でコンテナーの作成

新規コンテナーを作成するには、API バージョン、アカウント、および新規コンテナーの名前で **PUT** 要求を行います。コンテナー名は一意である必要があります。スラッシュ (/) を含めることはできず、256 バイト未満でなければなりません。リクエストには、アクセス制御ヘッダーおよびメタデータヘッダーを含めることができます。一連の配置プールのキーを特定するストレージポリシーを含めることもできます。たとえば、**radosgw-admin zone get** を実行すると、**placement_pools** で利用可能なキーの一覧を確認します。ストレージポリシーを使用すると、SSD ベースのストレージなど、コンテナーの特別なプールセットを指定できます。操作には、べき等性があります。既存のコンテナーを作成するように要求すると、HTTP 202 戻りコードが返されますが、別のコンテナーは作成されません。

構文

```
PUT /AP_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
```

表3.7 ヘッダー

Name	説明	型	必須
X-Container-Read	コンテナーの読み取りパーミッションを持つユーザー ID。	ユーザー ID のコンマ区切りの文字列値。	いいえ
X-Container-Write	コンテナーの書き込みパーミッションを持つユーザー ID。	ユーザー ID のコンマ区切りの文字列値。	いいえ
X-Container-Meta-KEY	任意の文字列の値を取得するユーザー定義のメタデータキー。	文字列	いいえ

Name	説明	型	必須
X-Storage-Policy	Ceph Object Gateway の **placement_pools** 下にあるストレージポリシーを識別するキー。**radosgw-admin zone get** を実行し、利用可能なキーを取得します。	文字列	No

同じ名前のコンテナーがすでに存在し、ユーザーがコンテナー所有者である場合、操作は成功します。そうでないと、操作は失敗します。

表3.8 HTTP レスポンス

Name	説明	ステータスコード
409	コンテナーは、別のユーザーの所有権にすでに存在します。	**BucketAlreadyExists**

3.5.7. Swift コンテナーの削除

コンテナーを削除するには、API バージョン、アカウント、およびコンテナーの名前を使用して **DELETE** 要求を行います。コンテナーは空である必要があります。コンテナーが空であるかを確認する場合は、コンテナーに対して **HEAD** リクエストを実行します。コンテナーが正常に削除されると、コンテナー名を再利用できます。

構文

```
DELETE /AP_VERSION/ACCOUNT/TENANT:CONTAINER HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

表3.9 HTTP レスポンス

Name	説明	ステータスコード
204	コンテナーが削除されました。	**NoContent**

3.5.8. Swift がコンテナーのメタデータを追加または更新

コンテナーにメタデータを追加するには、API バージョン、アカウント、およびコンテナー名で **POST** 要求を行います。メタデータを追加または更新するには、コンテナーに対する書き込み権限が必要です。

構文

```
POST /AP_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
```

表3.10 リクエストヘッダー

Name	説明	型	必須
X-Container-Meta-KEY	任意の文字列の値を取得するユーザー定義のメタデータキー。	文字列	No

3.6. SWIFT オブジェクト操作

開発者は、Ceph Object Gateway 経由で Swift アプリケーションのプログラミングインターフェイス (API) を使用してオブジェクト操作を行うことができます。オブジェクトを一覧表示、作成、更新、および削除することができます。オブジェクトのメタデータを追加または更新することもできます。

3.6.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- RESTful クライアント。

3.6.2. Swift オブジェクト操作

オブジェクトは、データおよびメタデータを保存するコンテナーです。コンテナーには多くのオブジェクトがありますが、オブジェクト名は一意である必要があります。この API により、クライアントはオブジェクトの作成、アクセス制御およびメタデータの設定、オブジェクトのデータおよびメタデータの取得、およびオブジェクトの削除を行うことができます。この API は特定のユーザーのアカウントの情報に関連するリクエストを行うため、この API のすべてのリクエストを認証する必要があります。コンテナーまたはオブジェクトのアクセス制御が意図的に公開されていない限り、つまり匿名の要求を許可している場合を除きます。

3.6.3. Swift がオブジェクトを取得

オブジェクトを取得するには、API バージョン、アカウント、コンテナー、およびオブジェクト名を使用して **GET** リクエストを行います。コンテナー内のオブジェクトを取得するには、コンテナーの読み取り権限が必要です。

構文

```
GET /AP_VERSION/ACCOUNT/TENANT:CONTAINER/OBJECT HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

表3.11 リクエストヘッダー

Name	説明	型	必須
範囲	オブジェクトの内容のサブセットを取得するには、バイト範囲を指定します。	Date	いいえ

Name	説明	型	必須
If-Modified-Since	ソースオブジェクトの **last_modified** 属性の日時以降に変更された場合のみコピーします。	Date	いいえ
If-Unmodified-Since	ソースオブジェクトの **last_modified** 属性の日時以降に変更した場合のみコピーします。	Date	いいえ
Copy-If-Match	リクエストの ETag がソースオブジェクトの ETag と一致する場合にのみコピーします。	ETag.	いいえ
Copy-If-None-Match	リクエストの ETag がソースオブジェクトの ETag と一致しない場合にのみコピーします。	ETag.	No

表3.12 レスポンスヘッダー

Name	説明
Content-Range	オブジェクトコンテンツのサブセットの範囲。range ヘッダーフィールドがリクエストで指定されている場合にのみ返されます。

3.6.4. Swift でオブジェクトの作成または更新

新規オブジェクトを作成するには、API バージョン、アカウント、コンテナー名、および新規オブジェクトの名前を使用して **PUT** 要求を行います。オブジェクトを作成または更新するには、コンテナーに書き込みパーミッションが必要です。オブジェクト名は、コンテナー内で一意である必要があります。**PUT** リクエストはべき等ではないため、一意の名前を使用しないと、リクエストによりオブジェクトが更新されます。ただし、オブジェクト名に疑似階層構文を使用して、別の疑似階層ディレクトリーにある場合は、同じ名前の別のオブジェクトと区別することができます。リクエストには、アクセス制御ヘッダーおよびメタデータヘッダーを含めることができます。

構文

```
PUT /AP_VERSION/ACCOUNT/TENANT:CONTAINER HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

表3.13 リクエストヘッダー

Name	説明	型	必須	有効な値
ETag	オブジェクトの内容の MD5 ハッシュ。推奨されます。	文字列	No	該当なし
Content-Type	オブジェクトに含まれるコンテンツのタイプ。	文字列	No	該当なし

Name	説明	型	必須	有効な値
Transfer-Encoding	オブジェクトが大規模な集約オブジェクトの一部であるかどうかを示します。	文字列	いいえ	**chunked**

3.6.5. Swift でオブジェクトの削除

オブジェクトを削除するには、API バージョン、アカウント、コンテナー、およびオブジェクト名を使用して **DELETE** リクエストを行います。コンテナー内のオブジェクトを削除するには、コンテナーに対する書き込み権限が必要です。オブジェクトが正常に削除されると、オブジェクト名を再利用できます。

構文

```
DELETE /API_VERSION/ACCOUNT/TENANT:CONTAINER/OBJECT HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

3.6.6. Swift でオブジェクトのコピー

オブジェクトのコピーを使用すると、オブジェクトをダウンロードしたり、別のコンテナーにアップロードしたりしなくてもよいように、オブジェクトのサーバー側のコピーを作成できます。あるオブジェクトのコンテンツを別のオブジェクトにコピーするには、API バージョン、アカウント、およびコンテナー名で **PUT** 要求または **COPY** 要求を行います。

PUT 要求の場合は、要求で宛先コンテナーおよびオブジェクト名、および要求ヘッダーのソースコンテナーおよびオブジェクトを使用します。

Copy リクエストには、要求でソースコンテナーおよびオブジェクト、および要求ヘッダーの宛先コンテナーおよびオブジェクトを使用します。オブジェクトをコピーするには、コンテナーに書き込みパーミッションが必要です。宛先オブジェクト名は、コンテナー内で一意である必要があります。リクエストはべき等ではないため、一意の名前を使用しないと、リクエストにより宛先オブジェクトが更新されます。宛先オブジェクトが別の疑似階層ディレクトリーにある場合は、オブジェクト名に疑似階層構文を使用して、同じ名前のソースオブジェクトと区別できます。リクエストには、アクセス制御ヘッダーおよびメタデータヘッダーを含めることができます。

構文

```
PUT /AP_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
```

または、次のようになります。

構文

```
COPY /AP_VERSION/ACCOUNT/TENANT:SOURCE_CONTAINER/SOURCE_OBJECT HTTP/1.1
Destination: TENANT:DEST_CONTAINER/DEST_OBJECT
```

表3.14 リクエストヘッダー

Name	説明	型	必須
X-Copy-From	ソースコンテナー/オブジェクトパスを定義するために **PUT** リクエストで使用されます。	文字列	はい (**PUT** を使用している場合)
Destination	宛先コンテナー/オブジェクトパスを定義するために **COPY** 要求で使用されます。	文字列	はい (**COPY** を使用している場合)
If-Modified-Since	ソースオブジェクトの **last_modified** 属性の日時以降に変更された場合のみコピーします。	Date	いいえ
If-Unmodified-Since	ソースオブジェクトの **last_modified** 属性の日時以降に変更した場合のみコピーします。	Date	いいえ
Copy-If-Match	リクエストの ETag がソースオブジェクトの ETag と一致する場合にのみコピーします。	ETag.	いいえ
Copy-If-None-Match	リクエストの ETag がソースオブジェクトの ETag と一致しない場合にのみコピーします。	ETag.	No

3.6.7. Swift でオブジェクトメタデータの取得

オブジェクトのメタデータを取得するには、API バージョン、アカウント、コンテナー、およびオブジェクト名を使用して **HEAD** リクエストを行います。コンテナー内のオブジェクトからメタデータを取得するには、コンテナーの読み取り権限が必要です。このリクエストは、オブジェクト自体の要求と同じヘッダー情報を返しますが、オブジェクトのデータを返しません。

構文

```
HEAD /AP_VERSION/ACCOUNT/TENANT:CONTAINER/OBJECT HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

3.6.8. Swift によるオブジェクトメタデータの追加または更新

オブジェクトにメタデータを追加するには、API バージョン、アカウント、コンテナー、およびオブジェクト名で **POST** リクエストを行います。メタデータを追加または更新するには、親コンテナーに対する書き込み権限が必要です。

構文

```
POST /AP_VERSION/ACCOUNT/TENANT:CONTAINER/OBJECT HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

表3.15 リクエストヘッダー

Name	説明	型	必須
X-Object-Meta-KEY	任意の文字列の値を取得するユーザー定義のメタデータキー。	文字列	No

3.7. SWIFT の一時 URL 操作

一時的なアクセスを可能にするため、**radosgw** の swift エンドポイントによりサポートされます。たとえば、GET リクエストは、認証情報を共有せずにオブジェクトに送信されます。

この機能には、最初に **X-Account-Meta-Temp-URL-Key** の値を設定し、必要に応じて **X-Account-Meta-Temp-URL-Key-2** を設定する必要があります。Temp URL 機能は、これらの秘密鍵に対する HMAC-SHA1 署名に依存します。

3.7.1. Swift が一時 URL オブジェクトを取得

一時 URL は、以下の要素を含む暗号化 HMAC-SHA1 署名を使用します。

- Request メソッドの値 (例:GET)
- エポックからの経過時間 (秒単位)。つまり Unix 時間です。
- v1 以降のリクエストパス

上記の項目は、それらの間に新しい行が追加されて正規化され、HMAC は前述の Temp URL キーのいずれかに対して SHA-1 ハッシュアルゴリズムを使用して生成されます。

上記を示すサンプルの Python スクリプトを以下に示します。

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
```

出力例

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

3.7.2. Swift POST 一時 URL キー

必要なキーを持つ swift アカウントへの **POST** リクエストは、一時 URL アクセスをアカウントに提供できるアカウントのシークレット一時 URL キーを設定します。最大 2 つのキーがサポートされ、一時 URL を無効化せずに鍵をローテーションできるように、両方のキーに対して署名がチェックされます。

構文

```
POST /API_VERSION/ACCOUNT HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

表3.16 リクエストヘッダー

Name	説明	型	必須
X-Account-Meta-Temp-URL-Key	任意の文字列値を取るユーザー定義のキー。	文字列	はい
X-Account-Meta-Temp-URL-Key-2	任意の文字列値を取るユーザー定義のキー。	文字列	No

3.8. SWIFT マルチテナンシーコンテナーの操作

クライアントアプリケーションがコンテナーにアクセスする場合は、常に特定ユーザーの認証情報で動作します。Red Hat Ceph Storage クラスターでは、すべてのユーザーがテナントに属します。そのため、テナントが明示的に指定されていない場合、すべてのコンテナー操作のコンテキストに暗黙的なテナントがあります。したがって、マルチテナンシーは、参照されるコンテナーと、参照しているユーザーが同じテナントに属する限り、以前のリリースと完全に後方互換性があります。

明示的なテナントの指定に使用される拡張機能は、使用されるプロトコルおよび認証システムによって異なります。

テナントとコンテナーはコロンで区切ります。したがって、URL は以下のようになります。

例

```
https://rgw.domain.com/tenant:container
```

一方、**create_container()** メソッドでは、コンテナーメソッド自体でテナントとコンテナーを分離します。

例

```
create_container("tenant:container")
```

3.9. 関連情報

- マルチテナンシーの詳細は、**Red Hat Ceph Storage Object Gateway 設定および管理ガイド**を参照してください。
- Swift リクエストヘッダーの **付録 D** を参照してください。
- Swift レスポンスヘッダーの **付録 E** を参照してください。

付録A S3 の一般的なリクエストヘッダー

以下の表には、有効な一般的なリクエストヘッダーとその説明をまとめています。

表A.1 リクエストヘッダー

リクエストヘッダー	詳細
CONTENT_LENGTH	リクエストボディーの長さ。
DATE	要求の日時と日付 (UTC 単位)。
HOST	ホストサーバーの名前。
AUTHORIZATION	承認トークン。

付録B S3 の 一般的なレスポンスステータスコード

以下の表は、有効な一般的な HTTP レスポンスステータスと対応するコードを示しています。

表B.1 レスポンスのステータス

HTTP ステータス	レスポンスコード
100	Continue
200	Success
201	Created
202	Accepted
204	NoContent
206	Partial content
304	NotModified
400	InvalidArgument
400	InvalidDigest
400	BadDigest
400	InvalidBucketName
400	InvalidObjectName
400	UnresolvableGrantByEmailAddress
400	InvalidPart
400	InvalidPartOrder
400	RequestTimeout
400	EntityTooLarge
403	AccessDenied
403	UserSuspended
403	RequestTimeTooSkewed

HTTP ステータス	レスポンスコード
404	NoSuchKey
404	NoSuchBucket
404	NoSuchUpload
405	MethodNotAllowed
408	RequestTimeout
409	BucketAlreadyExists
409	BucketNotEmpty
411	MissingContentLength
412	PreconditionFailed
416	InvalidRange
422	UnprocessableEntity
500	InternalError

付録C S3 サポートされないヘッダーフィールド

表C.1 サポートされないヘッダーフィールド

Name	型
x-amz-security-token	リクエスト
Server	レスポンス
x-amz-delete-marker	レスポンス
x-amz-id-2	レスポンス
x-amz-request-id	レスポンス
x-amz-version-id	レスポンス

付録D SWIFT リクエストヘッダー

表D.1 リクエストヘッダー

Name	説明	型	必須
X-Auth-User	認証するキーの Ceph Object Gateway のユーザー名。	文字列	はい
X-Auth-Key	Ceph Object Gateway のユーザー名に関連付けられたキー。	文字列	はい

付録E SWIFT レスポンスヘッダー

サーバーからのレスポンスには、**X-Auth-Token** の値が含まれている必要があります。レスポンスには、API のドキュメント全体で他のリクエストに指定される **API_VERSION/ACCOUNT** 接頭辞を提供する **X-Storage-Url** も含まれる可能性があります。

表E.1 レスポンスヘッダー

Name	説明	型
X-Storage-Token	要求に指定された **X-Auth-User** の承認トークン。	文字列
X-Storage-Url	ユーザーの URL および **API_VERSION/ACCOUNT** パス。	文字列

付録F SECURE TOKEN SERVICE API の使用例

これらの例は、Python の **boto3** モジュールを使用して、Ceph Object Gateway の Secure Token Service (STS) の実装と対話しています。これらの例では、**TESTER2** は **TESTER1** によって作成されたロールを想定しています。これは、ロールに割り当てられたパーミッションポリシーに基づいて **TESTER1** が所有する S3 リソースにアクセスするためです。

AssumeRole のサンプルはロールを作成し、ポリシーをロールに割り当てます。次に、一時認証情報を取得し、それらの一時認証情報を使用して S3 リソースにアクセスするロールを想定します。

AssumeRoleWithWebIdentity の例は、OpenID Connect ID プロバイダーである Keycloak を使用して外部アプリケーションを使用してユーザーを認証し、ロールのアクセス許可ポリシーに従って一時的な認証情報を取得して S3 リソースにアクセスするロールを引き受けます。

AssumeRole の例

```
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 の例

```
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\":\
{\"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 = 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()
```

関連情報

- Python の **boto** モジュールの使用に関する詳細は、**Red Hat Ceph Storage Object Gateway 設定および管理ガイド**の S3 アクセスのテスト セクションを参照してください。

付録G STS で属性ベースのアクセス制御にセッションタグを使用する例

次のリストには、STS での属性ベースのアクセス制御 (ABAC) のセッションタグの使用例が含まれています。

Web トークンで Keycloak によって渡されるセッションタグの例

```
{
    "jti": "947960a3-7e91-4027-99f6-da719b0d4059",
    "exp": 1627438044,
    "nbf": 0,
    "iat": 1627402044,
    "iss": "http://localhost:8080/auth/realms/quickstart",
    "aud": "app-profile-jsp",
    "sub": "test",
    "typ": "ID",
    "azp": "app-profile-jsp",
    "auth_time": 0,
    "session_state": "3a46e3e7-d198-4a64-8b51-69682bcfc670",
    "preferred_username": "test",
    "email_verified": false,
    "acr": "1",
    "https://aws.amazon.com/tags": [
        {
            "principal_tags": {
                "Department": [
                    "Engineering",
                    "Marketing"
                ]
            }
        }
    ],
    "client_id": "app-profile-jsp",
    "username": "test",
    "active": true
}
```

aws:RequestTag の例

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

aws:PrincipalTag の例

```
{
      "Version":"2012-10-17",
      "Statement":[
      {
         "Effect":"Allow",
         "Action":["s3:*"],
      "Resource":["arn:aws:s3::t1tenant:my-test-bucket","arn:aws:s3::t1tenant:my-test-bucket/*"],"+
         "Condition":{"StringEquals":{"aws:PrincipalTag/Department":"Engineering"}}
      }]
   }
```

aws:ResourceTag の例

```
{
      "Version":"2012-10-17",
      "Statement":[
      {
         "Effect":"Allow",
         "Action":["sts:AssumeRoleWithWebIdentity","sts:TagSession"],
         "Principal":{"Federated":["arn:aws:iam:::oidc-provider/localhost:8080/auth/realms/quickstart"]},
         "Condition":{"StringEquals":{"iam:ResourceTag/Department":"Engineering"}}(1)
      }]
   }
```

(1) (1) (1) 上記を機能させるには、'Department=Engineering' タグをロールにアタッチする必要があります。

aws:TagKeys の例

```
{
      "Version":"2012-10-17",
      "Statement":[
      {
         "Effect":"Allow",
         "Action":["sts:AssumeRoleWithWebIdentity","sts:TagSession"],
         "Principal":{"Federated":["arn:aws:iam:::oidc-provider/localhost:8080/auth/realms/quickstart"]},
         "Condition":{"ForAllValues:StringEquals":{"aws:TagKeys":["Marketing,Engineering"]}}(1)
      }]
   }
```

(1) **ForAllValues:StringEquals** は、リクエスト内のすべてのタグキーがポリシー内のタグキーのサブセットであるかどうかをテストします。したがって、この条件は、リクエストで渡されるタグキーを制限します。

s3:ResourceTag の例

```
{
   "Version":"2012-10-17",
   "Statement":[
   {
      "Effect":"Allow",
      "Action":["s3:PutBucketTagging"],
```

```
        "Resource":["arn:aws:s3::t1tenant:my-test-bucket\","arn:aws:s3::t1tenant:my-test-bucket/*"]
    },
    {
        "Effect":"Allow",
        "Action":["s3:*"],
        "Resource":["*"],
        "Condition":{"StringEquals":{"s3:ResourceTag/Department":\"Engineering"}}
    }
}
```

1 上記を機能させるには、このポリシーを適用するバケットまたはオブジェクトに Department=Engineering タグを添付する必要があります。

iam:ResourceTag を使用した aws:RequestTag の例

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

1 これは、受信リクエスト内のタグとロールに付けられたタグを照合することで、ロールを引き受けることです。**aws:RequestTag** は JSON Web Token (JWT) の受信タグであり、**iam:ResourceTag** は引き受けるロールに添付されたタグです。

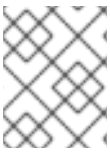
s3:ResourceTag を使用した aws:PrincipalTag の例

```
{
    "Version":"2012-10-17",
    "Statement":[
    {
        "Effect":"Allow",
        "Action":["s3:PutBucketTagging"],
        "Resource":["arn:aws:s3::t1tenant:my-test-bucket\","arn:aws:s3::t1tenant:my-test-bucket/*"]
    },
    {
        "Effect":"Allow",
        "Action":["s3:*"],
        "Resource":["*"],
        "Condition":{"StringEquals":{"s3:ResourceTag/Department":"${aws:PrincipalTag/Department}"}}
    }
    }
}
```

1 これは、プリンシパルタグと S3 リソースタグを照合して、ロールのアクセス許可ポリシーを評価するためです。**aws:PrincipalTag** は、一時的な認証情報とともに渡されるタグであり、**s3:ResourceTag** は、オブジェクトまたはバケットである S3 リソースに添付されたタグです。

付録H セッションタグの使用方法を示すサンプルコード

以下は、ロール、バケット、またはオブジェクトにタグを付け、ロールの信頼とロールの許可ポリシーでタグキーを使用するためのサンプルコードです。



注記

この例では、タグ **Department=Engineering** が Keycloak によって JSON Web トークン (JWT) アクセストークンで渡されることを前提としています。

```
# -*- coding: utf-8 -*-

import boto3
import json
from nose.tools import eq_ as eq

access_key = 'TESTER'
secret_key = 'test123'
endpoint = 'http://s3.us-east.localhost:8000'

s3client = boto3.client('s3',
aws_access_key_id = access_key,
aws_secret_access_key = secret_key,
endpoint_url = endpoint,
region_name='',)

s3res = boto3.resource('s3',
        aws_access_key_id = access_key,
        aws_secret_access_key = secret_key,
        endpoint_url = endpoint,
        region_name='',)

iam_client = boto3.client('iam',
aws_access_key_id=access_key,
aws_secret_access_key=secret_key,
endpoint_url=endpoint,
region_name=''
)

bucket_name = 'test-bucket'
s3bucket = s3client.create_bucket(Bucket=bucket_name)

bucket_tagging = s3res.BucketTagging(bucket_name)
Set_Tag = bucket_tagging.put(Tagging={'TagSet':[{'Key':'Department', 'Value': 'Engineering'}]})
try:
    response = iam_client.create_open_id_connect_provider(
        Url='http://localhost:8080/auth/realms/quickstart',
        ClientIDList=[
            'app-profile-jsp',
            'app-jee-jsp'
        ],
        ThumbprintList=[
            'F7D7B3515DD0D319DD219A43A9EA727AD6065287'
    ]
    )
```

```
except ClientError as e:
    print ("Provider already exists")

policy_document = "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",\"Principal\":
{\"Federated\":[\"arn:aws:iam:::oidc-provider/localhost:8080/auth/realms/quickstart\"]},\"Action\":
[\"sts:AssumeRoleWithWebIdentity\",\"sts:TagSession\"],\"Condition\":{\"StringEquals\":
{\"aws:RequestTag/Department\":\"${iam:ResourceTag/Department}\"}}}]}"
role_response = ""

print ("\n Getting Role \n")

try:
    role_response = iam_client.get_role(
        RoleName='S3Access'
    )
    print (role_response)
except ClientError as e:
    if e.response['Code'] == 'NoSuchEntity':
        print ("\n Creating Role \n")
        tags_list = [
            {'Key':'Department','Value':'Engineering'},
        ]
        role_response = iam_client.create_role(
            AssumeRolePolicyDocument=policy_document,
            Path='/',
            RoleName='S3Access',
            Tags=tags_list,
        )
        print (role_response)
    else:
        print("Unexpected error: %s" % e)

role_policy = "{\"Version\":\"2012-10-17\",\"Statement\":
{\"Effect\":\"Allow\",\"Action\":\"s3:*\",\"Resource\":\"arn:aws:s3:::*\",\"Condition\":{\"StringEquals\":
{\"s3:ResourceTag/Department\":[\"${aws:PrincipalTag/Department}\"]}}}}"

response = iam_client.put_role_policy(
            RoleName='S3Access',
            PolicyName='Policy1',
            PolicyDocument=role_policy
        )

sts_client = boto3.client('sts',
aws_access_key_id='abc',
aws_secret_access_key='def',
endpoint_url = endpoint,
region_name = '',
)


print ("\n Assuming Role with Web Identity\n")
response = sts_client.assume_role_with_web_identity(
RoleArn=role_response['Role']['Arn'],
RoleSessionName='Bob',
DurationSeconds=900,
WebIdentityToken='<web-token>')
```

```
s3client2 = 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='http://s3.us-east.localhost:8000',
region_name='',)

bucket_body = 'this is a test file'
tags = 'Department=Engineering'
key = "test-1.txt"
s3_put_obj = s3client2.put_object(Body=bucket_body, Bucket=bucket_name, Key=key,
Tagging=tags)
eq(s3_put_obj['ResponseMetadata']['HTTPStatusCode'],200)

s3_get_obj = s3client2.get_object(Bucket=bucket_name, Key=key)
eq(s3_get_obj['ResponseMetadata']['HTTPStatusCode'],200)
```