Red Hat Ceph Storage 4

개발자 가이드

Red Hat Ceph Storage에 다양한 애플리케이션 프로그래밍 인터페이스 사용

Last Updated: 2023-09-02

Red Hat Ceph Storage 4 개발자 가이드

Red Hat Ceph Storage에 다양한 애플리케이션 프로그래밍 인터페이스 사용

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

법적 공지

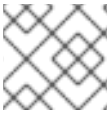
초록

이 문서에서는 AMD64 및 Intel 64 아키텍처에서 실행되는 Red Hat Ceph Storage에 다양한 애플리케이션 프로그래밍 인터페이스를 사용하는 방법을 설명합니다. Red Hat은 코드, 문서, 웹 속성에서 문제가 있는 용어를 교체하기 위해 최선을 다하고 있습니다. 먼저 마스터(master), 슬레이브(slave), 블랙리스트(blacklist), 화이트리스트(whitelist) 등 네 가지 용어를 교체하고 있습니다. 이러한 변경 작업은 작업 범위가 크므로 향후 여러 릴리스에 걸쳐 점차 구현할 예정입니다. 자세한 내용은 CTO Chris Wright의 메시지에서 참조하십시오.

차례

1장. CEPH OBJECT GATEWAY 관리 API

개발자는 RESTful 애플리케이션 프로그래밍 인터페이스(API)와 상호 작용하여 Ceph Object Gateway를 관리할 수 있습니다. Ceph Object Gateway를 사용하면 RESTful API에서 **radosgw-admin** 명령 기능을 사용할 수 있습니다. 다른 관리 플랫폼과 통합할 수 있는 사용자, 데이터, 할당량 및 사용을 관리할 수 있습니다.



참고

Ceph Object Gateway를 구성할 때 명령줄 인터페이스를 사용하는 것이 좋습니다.

관리 API는 다음과 같은 기능을 제공합니다.

- 인증 요청
- **사용자 계정 관리**
  - 관리 사용자
  - 사용자 정보 얻기
  - 생성
  - 수정
  - 제거
  - Subuser 생성
  - Subuser 수정
  - Subuser 제거
- **사용자 기능 관리**
  - 추가
  - 제거
- **키 관리**
  - 생성
  - 제거
- **bucket Management**
  - Bucket 정보 얻기
  - 인덱스 확인
  - 제거
  - 연결
  - unlinking

1.1. 사전 요구 사항

- 실행 중인 Red Hat Ceph Storage 클러스터.
- RESTful 클라이언트.

1.2. 관리 작업

관리 API(Application Programming Interface) 요청은 구성 가능한 'admin' 리소스 진입점으로 시작하는 URI에서 수행됩니다. 관리 API에 대한 권한 부여는 S3 권한 부여 메커니즘을 복제합니다. 일부 작업을 수행하려면 사용자가 특수 관리 기능을 보유해야 합니다. 응답 엔티티 유형(XML 또는 JSON)은 요청에서 'format' 옵션으로 지정되고 지정되지 않은 경우 기본값은 JSON으로 지정될 수 있습니다.

예제

```
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. 관리 인증 요청

Amazon의 S3 서비스는 액세스 키와 요청 헤더의 해시와 시크릿 키를 사용하여 요청을 인증합니다. 인증된 요청, 특히 대규모 업로드를 SSL 오버헤드 없이 제공하는 이점이 있습니다.

S3 API의 대부분의 사용 사례는 Amazon SDK for Java 또는 Python Boto에서 **AmazonS3Client** 와 같은 오픈 소스 S3 클라이언트를 사용하는 것입니다. 이러한 라이브러리는 Ceph Object Gateway Admin API를

지원하지 않습니다. 이러한 라이브러리를 하위 클래스로 확장하여 Ceph Admin API를 지원할 수 있습니다. 또는 고유한 게이트웨이 클라이언트를 만들 수 있습니다.

execute() 메서드 생성

이 섹션의 CephAdminAPI 예제 클래스에서는 요청 매개 변수를 사용하고, 요청을 인증하고, Ceph Admin API를 호출하고, 응답을 수신할 수 있는 **execute()** 메서드를 생성하는 방법을 보여줍니다.

CephAdminAPI 클래스 예제에서는 지원되지 않거나 상업적 용도로 사용할 수 없습니다. 이는 예시적인 목적으로만 사용됩니다.

Ceph 개체 게이트웨이 호출

클라이언트 코드에 는 CRUD 작업을 시연하기 위해 Ceph Object Gateway에 대한 5개의 호출이 포함되어 있습니다.

- 사용자 만들기
- 사용자 가져오기
- 사용자 수정
- 하위 사용자 만들기
- 사용자 삭제

이 예제를 사용하려면 **httpcomponents-client-4.5.3** Apache HTTP 구성 요소를 가져옵니다. 예를 들어 여기에서 다운로드할 수 있습니다. http://hc.apache.org/downloads.cgi. 그런 다음 tar 파일의 압축을 풀고 **lib** 디렉토리로 이동하여 **JAVA_HOME** 디렉터리의 **/jre/lib/ext** 디렉터리 또는 사용자 지정 classpath에 콘텐츠를 복사합니다.

CephAdminAPI 클래스 예제를 검사할 때 **execute()** 메서드는 HTTP 메서드, 요청 경로, 선택적 하위 리소스, 지정되지 않은 경우 **null**, 매개변수 맵을 사용합니다. 하위 리소스(예: **subuser**, **key**)로 실행하려면 하위 리소스를 **execute()** 메서드에서 인수로 지정해야 합니다.

예제 메서드:

1. URI를 빌드합니다.
2. HTTP 헤더 문자열을 작성합니다.
3. HTTP 요청을 인스턴스화합니다(예: **PUT**,**POST**,**GET**,**DELETE**).
4. HTTP 헤더 문자열 및 요청 헤더에 **Date** 헤더를 추가합니다.
5. **Authorization** 헤더를 HTTP 요청 헤더에 추가합니다.
6. HTTP 클라이언트를 인스턴스화하고 인스턴스화한 HTTP 요청을 전달합니다.
7. 요청을 합니다.
8. 응답을 반환합니다.

헤더 문자열 빌드

헤더 문자열을 빌드하는 것은 Amazon의 S3 인증 프로시저를 포함하는 프로세스의 일부입니다. 특히 예제 메서드는 다음을 수행합니다.

1. 요청 유형(예: **PUT**,**POST**,**GET**,**DELETE**)을 추가합니다.

2. 날짜를 추가합니다.

3. requestPath를 추가합니다.

요청 유형은 선행 또는 후행 공백이 없는 대문자이어야 합니다. 공백을 비우지 않으면 인증이 실패합니다. 날짜는 kafka로 표시되거나 인증이 실패합니다.

예시적인 방법에는 다른 헤더가 없습니다. Amazon S3 인증 절차에서는 사전 단위로 **x-amz** 헤더를 정렬합니다. **x-amz** 헤더를 추가하는 경우 사전순으로 추가해야 합니다.

헤더 문자열을 빌드했으면 다음 단계는 HTTP 요청을 인스턴스화하고 URI를 전달하는 것입니다. 시험 분할 방법은 **PUT** 을 사용하여 사용자 및 하위 사용자 생성, 사용자 변경을 위한 **POST**, 사용자 삭제용 **DELETE** 를 사용합니다.

요청을 인스턴스화하면 Date 헤더와 Authorization 헤더를 추가합니다. Amazon의 S3 인증은 표준 Authorization 헤더를 사용하며 다음과 같은 구조를 갖습니다.

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

CephAdminAPI 예제 클래스에는 base64Sha1Hmac() 메서드가 있습니다. 이 방법은 헤더 문자열과 admin 사용자의 시크릿 키를 사용하고 SHA1 HMAC를 base-64 인코딩 문자열로 반환합니다. 각 execute() 호출은 동일한 코드 줄을 호출하여 Authorization 헤더를 빌드합니다.

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

다음 CephAdminAPI 예제 클래스를 사용하려면 access 키, secret 키 및 끝점을 생성자에 전달해야 합니다. 클래스는 런타임 시 변경할 수 있는 접근자 메서드를 제공합니다.

예제

```
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);
  }
 }

}
```

후속 **CephAdminAPIClient** 예제에서는 **CephAdminAPI** 클래스를 인스턴스화하고, 요청 매개 변수에 대한 맵을 빌드하며, **execute()** 메서드를 사용하여 사용자를 생성, 가져오기, 업데이트 및 삭제하는 방법을 보여줍니다.

예제

```
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();
```

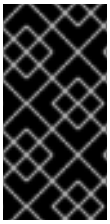
```
try {
 System.out.println("\nResponse Content is: "
  + EntityUtils.toString(entity, "UTF-8") + "\n");
 response.close();
} catch (IOException e){
 System.err.println ("Encountered an I/O exception.");
 e.printStackTrace();
}

/*
 * 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();
}
 }
}
```

추가 리소스

- 자세한 내용은 ***Red Hat Ceph Storage** 개발자 가이드* 의 ***S3** 인증 섹션을* 참조하십시오.
- **Amazon S3** 인증 절차에 대한 자세한 내용은 **Amazon Simple Storage Service** 설명서의 **REST** 요청 서명 및 인증 섹션을 참조하십시오.

1.4. 관리 사용자 생성



중요

Ceph Object Gateway 노드에서 **radosgw-admin** 명령을 실행하려면 노드에 **admin** 키가 있는지 확인합니다. **admin** 키는 모든 **Ceph Monitor** 노드에서 복사할 수 있습니다.

사전 요구 사항

- **Ceph Object Gateway** 노드에 대한 루트 수준 액세스.

절차

1. 오브젝트 게이트웨이 사용자를 생성합니다.

구문

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

예제

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

radosgw-admin 명령줄 인터페이스는 사용자를 반환합니다.

출력 예

-

```
{
    "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. **생성하는 사용자에게 관리 기능을 할당합니다.**

구문

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

예제

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

radosgw-admin 명령줄 인터페이스는 사용자를 반환합니다. **"caps":** 에는 사용자에게 할당된 기능이 있습니다.

출력 예

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

이제 관리 권한이 있는 사용자가 있습니다.

1.5. 사용자 정보 가져오기

사용자 정보를 가져옵니다.

capabilities

```
users=read
```

구문

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

표 1.1. 요청 매개변수

이름	설명	유형	예제	필수 항목
uid	정보가 요청된 사용자입니다.	문자열	**foo_user**	있음

표 1.2. 응답 엔티티

이름	설명	유형	parent
user	사용자 데이터 정보를 위한 컨테이너입니다.	컨테이너	해당 없음
user_id	사용자 ID입니다.	문자열	**user**

이름	설명	유형	parent
display_name	사용자의 이름을 표시합니다.	문자열	**user**
일시 중지됨	사용자가 일시 중지되면 True입니다.	부울	**user**
max_buckets	사용자가 소유할 최대 버킷 수입니다.	정수	**user**
subusers	이 사용자 계정과 연결된 하위 사용자입니다.	컨테이너	**user**
keys	이 사용자 계정과 연결된 S3 키입니다.	컨테이너	**user**
swift_keys	이 사용자 계정과 연결된 Swift 키입니다.	컨테이너	**user**
caps	사용자 기능.	컨테이너	**user**

성공하면 응답에 사용자 정보가 포함됩니다.

특수 오류 응답

없음.

1.6. 사용자 생성

새 사용자를 만듭니다. 기본적으로 S3 키 쌍이 자동으로 생성되고 응답에서 반환됩니다. access-key 또는 secret-key 중 하나만 제공되면 생략된 키가 자동으로 생성됩니다. 기본적으로 기존 키 쌍을 교체하지 않고 생성된 키가 인증 키에 추가됩니다. access-key 가 지정되어 사용자가 소유한 기존 키를 참조하는 경우 수정됩니다.

capabilities

> **`users=write`**

구문

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

표 1.3. 요청 매개변수

이름	설명	유형	예제	필수 항목
uid	생성할 사용자 ID입니다.	문자열	**foo_user**	있음
display-name	생성할 사용자의 표시 이름입니다.	문자열	**foo user**	있음
email	사용자와 연결된 이메일 주소입니다.	문자열	**foo@bar.com**	없음
key-type	생성할 키 유형은 swift, s3(기본값)입니다.	문자열	**s3** [**s3**]	없음
액세스 키	액세스 키를 지정합니다.	문자열	**ABCD0EF12GHIJ2K34LMN**	없음
secret-key	시크릿 키를 지정합니다.	문자열	**0AbCDEFg1h2i34JkIM5nop6QrSTUV+WxyzaBC7D8**	없음
user-caps	사용자 기능.	문자열	**usage=read, write; users=read**	없음
generate-key	새 키 쌍을 생성하고 기존 인증 키에 를 추가합니다.	부울	True [True]	없음
max-buckets	사용자가 소유할 수 있는 최대 버킷 수를 지정합니다.	정수	500 [1000]	없음
일시 중지됨	사용자를 중지할지 여부를 지정합니다.	부울	false [False]	없음

표 1.4. 응답 엔티티

이름	설명	유형	parent
user	사용자 데이터 정보를 위한 컨테이너입니다.	컨테이너	해당 없음
user_id	사용자 ID입니다.	문자열	**user**

이름	설명	유형	parent
display_name	사용자의 이름을 표시합니다.	문자열	**user**
일시 중지됨	사용자가 일시 중지되면 True입니다.	부울	**user**
max_buckets	사용자가 소유할 최대 버킷 수입니다.	정수	**user**
subusers	이 사용자 계정과 연결된 하위 사용자입니다.	컨테이너	**user**
keys	이 사용자 계정과 연결된 S3 키입니다.	컨테이너	**user**
swift_keys	이 사용자 계정과 연결된 Swift 키입니다.	컨테이너	**user**
caps	사용자 기능.	컨테이너	**user**

성공하면 응답에 사용자 정보가 포함됩니다.

표 1.5. 특수 오류 응답

이름	설명	코드
UserExists	기존 사용자 생성을 시도합니다.	409 충돌
InvalidAccessKey	잘못된 액세스 키가 지정되어 있습니다.	400 잘못된 요청
InvalidKeyType	잘못된 키 유형이 지정되어 있습니다.	400 잘못된 요청
InvalidSecretKey	잘못된 보안 키가 지정되어 있습니다.	400 잘못된 요청
InvalidKeyType	잘못된 키 유형이 지정되어 있습니다.	400 잘못된 요청
KeyExists	제공된 액세스 키가 존재하며 다른 사용자에게 속합니다.	409 충돌
EmailExists	제공된 이메일 주소가 있습니다.	409 충돌
InvalidCap	잘못된 admin 기능을 부여하려고 합니다.	400 잘못된 요청

추가 리소스

- 하위 사용자를 *생성하려면 **Red Hat Ceph Storage** 개발자 가이드* 를 참조하십시오.

1.7. 사용자 수정

기존 사용자를 수정합니다.

capabilities

> **`users=write`**

구문

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

표 1.6. 요청 매개변수

이름	설명	유형	예제	필수 항목
uid	수정할 사용자 ID입니다.	문자열	**foo_user**	있음
display-name	수정할 사용자의 표시 이름입니다.	문자열	**foo user**	없음
email	사용자와 연결할 이메일 주소입니다.	문자열	**foo@bar.com**	없음
generate-key	새 키 쌍을 생성하고 기존 인증 키에 를 추가합니다.	부울	true [False]	없음
액세스 키	액세스 키를 지정합니다.	문자열	**ABCD0EF12GHIJ2K34LMN**	없음

이름	설명	유형	예제	필수 항목
secret-key	시크릿 키를 지정합니다.	문자열	**0AbCDEFg1h2i34JkIM5nop6QrSTUV+WxyzaBC7D8**	없음
key-type	생성할 키 유형은 swift, s3(기본값)입니다.	문자열	**s3**	없음
user-caps	사용자 기능.	문자열	**usage=read, write; users=read**	없음
max-buckets	사용자가 소유할 수 있는 최대 버킷 수를 지정합니다.	정수	500 [1000]	없음
일시 중지됨	사용자를 중지할지 여부를 지정합니다.	부울	false [False]	없음

표 1.7. 응답 엔티티

이름	설명	유형	parent
user	사용자 데이터 정보를 위한 컨테이너입니다.	컨테이너	해당 없음
user_id	사용자 ID입니다.	문자열	**user**
display_name	사용자의 이름을 표시합니다.	문자열	**user**
일시 중지됨	사용자가 일시 중지되면 True입니다.	부울	**user**
max_buckets	사용자가 소유할 최대 버킷 수입니다.	정수	**user**
subusers	이 사용자 계정과 연결된 하위 사용자입니다.	컨테이너	**user**
keys	이 사용자 계정과 연결된 S3 키입니다.	컨테이너	**user**
swift_keys	이 사용자 계정과 연결된 Swift 키입니다.	컨테이너	**user**
caps	사용자 기능.	컨테이너	**user**

성공하면 응답에 사용자 정보가 포함됩니다.

표 1.8. 특수 오류 응답

이름	설명	코드
InvalidAccessKey	잘못된 액세스 키가 지정되어 있습니다.	400 잘못된 요청
InvalidKeyType	잘못된 키 유형이 지정되어 있습니다.	400 잘못된 요청
InvalidSecretKey	잘못된 보안 키가 지정되어 있습니다.	400 잘못된 요청
KeyExists	제공된 액세스 키가 존재하며 다른 사용자에게 속합니다.	409 충돌
EmailExists	제공된 이메일 주소가 있습니다.	409 충돌
InvalidCap	잘못된 admin 기능을 부여하려고 합니다.	400 잘못된 요청

추가 리소스

- 하위 사용자를 *수정하려면 **Red Hat Ceph Storage** 개발자 가이드* 를 참조하십시오.

1.8. 사용자 제거

기존 사용자를 제거합니다.

capabilities

> **`users=write`**

구문

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

표 1.9. 요청 매개변수

이름	설명	유형	예제	필수 항목
uid	제거할 사용자 ID입니다.	문자열	**foo_user**	네, 필요합니다.
purge-data	지정된 경우 사용자에게 속한 버킷과 오브젝트도 제거됩니다.	부울	True	없음

응답 엔티티

없음.

특수 오류 응답

없음.

추가 리소스

- **하위 사용자를 제거하려면 *Red Hat Ceph Storage 개발자 가이드* 를 참조하십시오.**

1.9. 하위 사용자 생성

Swift API를 사용하여 클라이언트에 주로 유용한 새 하위 사용자를 만듭니다.

참고

유효한 요청에는 gen- subuser 또는 subuser가 필요합니다. 일반적으로 하위 사용자를 유용하게 사용하려면 액세스를 지정하여 권한을 부여해야 합니다. 하위 사용자를 보안 없이 지정하는 경우 사용자 생성과 마찬가지로 시크릿 키가 자동으로 생성됩니다.

capabilities

`users=write`

구문

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

표 1.10. 요청 매개변수

이름	설명	유형	예제	필수 항목
uid	하위 사용자를 생성할 사용자 ID입니다.	문자열	**foo_user**	있음
subuser	생성할 하위 사용자 ID를 지정합니다.	문자열	**sub_foo**	예(또는 **gen-subuser**)
gen-subuser	생성할 하위 사용자 ID를 지정합니다.	문자열	**sub_foo**	예(또는 **하위 사용자**)
secret-key	시크릿 키를 지정합니다.	문자열	**0AbCDEFg1h2i34JklM5nop6QrSTUVWxyzaBC7D8**	없음
key-type	생성할 키 유형은 swift(기본값), s3입니다.	문자열	**swift** [**swift**]	없음
액세스	하위 사용자에 대한 액세스 권한을 설정합니다. 읽기, **쓰기, 읽기 쓰기, 전체** 중 하나여야 합니다.	문자열	**read**	없음
generate-secret	비밀 키를 생성합니다.	부울	true [False]	없음

표 1.11. 응답 엔티티

이름	설명	유형	parent
subusers	사용자 계정과 연결된 하위 사용자입니다.	컨테이너	해당 없음

이름	설명	유형	parent
id	하위 사용자 ID.	문자열	**subusers**
권한	사용자 계정에 대한 하위 사용자 액세스.	문자열	**subusers**

성공하면 응답에 하위 사용자 정보가 포함됩니다.

표 1.12. 특수 오류 응답

이름	설명	코드
SubuserExists	지정된 하위 사용자가 있습니다.	409 충돌
InvalidKeyType	잘못된 키 유형이 지정되어 있습니다.	400 잘못된 요청
InvalidSecretKey	잘못된 보안 키가 지정되어 있습니다.	400 잘못된 요청
InvalidAccess	잘못된 하위 사용자 액세스 권한이 지정되어 있습니다.	400 잘못된 요청

1.10. 하위 사용자 수정

기존 하위 사용자를 수정합니다.

capabilities

> **`users=write`**

구문

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

표 1.13. 요청 매개변수

이름	설명	유형	예제	필수 항목
uid	하위 사용자를 수정할 사용자 ID입니다.	문자열	**foo_user**	있음
subuser	수정할 하위 사용자 ID입니다.	문자열	**sub_foo**	있음
generate-secret	하위 사용자에 대한 새 비밀 키를 생성하여 기존 키를 교체합니다.	부울	true [False]	없음
Secret	시크릿 키를 지정합니다.	문자열	**0AbCDEFg1h2i34JkIM5nop6QrSTUV+WxyzaBC7D8**	없음
key-type	생성할 키 유형은 swift(기본값), s3입니다.	문자열	**swift** [**swift**]	없음
액세스	하위 사용자에 대한 액세스 권한을 설정합니다. 읽기, **쓰기, 읽기 쓰기, 전체** 중 하나여야 합니다.	문자열	**read**	없음

표 1.14. 응답 엔티티

이름	설명	유형	parent
subusers	사용자 계정과 연결된 하위 사용자입니다.	컨테이너	해당 없음
id	하위 사용자 ID.	문자열	**subusers**
권한	사용자 계정에 대한 하위 사용자 액세스.	문자열	**subusers**

성공하면 응답에 하위 사용자 정보가 포함됩니다.

표 1.15. 특수 오류 응답

이름	설명	코드
InvalidKeyType	잘못된 키 유형이 지정되어 있습니다.	400 잘못된 요청
InvalidSecretKey	잘못된 보안 키가 지정되어 있습니다.	400 잘못된 요청
InvalidAccess	잘못된 하위 사용자 액세스 권한이 지정되어 있습니다.	400 잘못된 요청

1.11. 하위 사용자 제거

기존 하위 사용자를 제거합니다.

capabilities

> **`users=write`**

구문

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

표 1.16. 요청 매개변수

이름	설명	유형	예제	필수 항목
uid	하위 사용자를 제거할 사용자 ID입니다.	문자열	**foo_user**	있음

이름	설명	유형	예제	필수 항목
subuser	제거할 하위 사용자 ID입니다.	문자열	**sub_foo**	있음
purge-keys	하위 사용자에 속하는 키를 제거합니다.	부울	True [True]	없음

응답 엔티티

없음.

특수 오류 응답

없음.

1.12. 사용자에게 기능 추가

지정된 사용자에게 관리 기능을 추가합니다.

capabilities

> **`users=write`**

구문

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

표 1.17. 요청 매개변수

이름	설명	유형	예제	필수 항목
uid	관리 기능을 추가할 사용자 ID입니다.	문자열	**foo_user**	있음
user-caps	사용자에게 추가할 관리 기능.	문자열	**usage=read, write**	있음

표 1.18. 응답 엔티티

이름	설명	유형	parent
user	사용자 데이터 정보를 위한 컨테이너입니다.	컨테이너	해당 없음
user_id	사용자 ID입니다.	문자열	**user**
caps	사용자 기능.	컨테이너	**user**

성공하면 응답에 사용자의 기능이 포함됩니다.

표 1.19. 특수 오류 응답

이름	설명	코드
InvalidCap	잘못된 admin 기능을 부여하려고 합니다.	400 잘못된 요청

1.13. 사용자의 기능 제거

지정된 사용자의 관리 기능을 제거합니다.

capabilities

> **`users=write`**

구문

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

표 1.20. 요청 매개변수

이름	설명	유형	예제	필수 항목
uid	관리 기능을 제거할 수 있는 사용자 ID입니다.	문자열	**foo_user**	있음
user-caps	사용자에서 제거할 관리 기능.	문자열	**usage=read, write**	있음

표 1.21. 응답 엔티티

이름	설명	유형	parent
user	사용자 데이터 정보를 위한 컨테이너입니다.	컨테이너	해당 없음
user_id	사용자 ID입니다.	문자열	**user**
caps	사용자 기능.	컨테이너	**user**

성공하면 응답에 사용자의 기능이 포함됩니다.

표 1.22. 특수 오류 응답

이름	설명	코드
InvalidCap	잘못된 관리자 기능을 제거하려고 합니다.	400 잘못된 요청
NoSuchCap	사용자에게 지정된 기능이 없습니다.	404 찾을 수 없음

1.14. 키 생성

새 키를 만듭니다. 하위 사용자를 지정하면 기본적으로 생성된 키가 swift 유형이 됩니다. access-key

또는 **secret-key** 중 하나만 제공되면 커밋된 키가 자동으로 생성되며, 이는 **secret-key** 만 지정된 경우 **access-key** 가 자동으로 생성됩니다. 기본적으로 기존 키 쌍을 교체하지 않고 생성된 키가 인증 키에 추가됩니다. **access-key** 가 지정되어 사용자가 소유한 기존 키를 참조하는 경우 수정됩니다. 응답은 생성된 키와 동일한 유형의 모든 키를 나열하는 컨테이너입니다.

참고

swift 키를 만들 때 **access-key** 옵션을 지정하면 효과가 없습니다. 또한 각 사용자 또는 하위 사용자가 하나의 **swift** 키만 보유할 수 있습니다.

capabilities

`users=write`

구문

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

표 1.23. 요청 매개변수

이름	설명	유형	예제	필수 항목
uid	새 키를 받을 사용자 ID입니다.	문자열	**foo_user**	있음
subuser	새 키를 수신할 하위 사용자 ID입니다.	문자열	**sub_foo**	없음
key-type	생성할 키 유형은 swift, s3(기본값)입니다.	문자열	**s3** [**s3**]	없음
액세스 키	액세스 키를 지정합니다.	문자열	**AB01C2D3EF45 G6H7IJ8K**	없음

이름	설명	유형	예제	필수 항목
secret-key	시크릿 키를 지정합니다.	문자열	**0ab/CdeFGhij1klmnopqRSTUv1WxyZabcDEFgHij**	없음
generate-key	새 키 쌍을 생성하고 기존 인증 키에 를 추가합니다.	부울	true [**true**]	없음

표 1.24. 응답 엔티티

이름	설명	유형	parent
keys	이 사용자 계정과 연결된 생성된 유형의 키입니다.	컨테이너	해당 없음
user	키와 연결된 사용자 계정입니다.	문자열	**keys**
액세스 키	액세스 키입니다.	문자열	**keys**
secret-key	시크릿 키	문자열	**keys**

표 1.25. 특수 오류 응답

이름	설명	코드
InvalidAccessKey	잘못된 액세스 키가 지정되어 있습니다.	400 잘못된 요청
InvalidKeyType	잘못된 키 유형이 지정되어 있습니다.	400 잘못된 요청
InvalidSecretKey	잘못된 보안 키가 지정되어 있습니다.	400 잘못된 요청
InvalidKeyType	잘못된 키 유형이 지정되어 있습니다.	400 잘못된 요청
KeyExists	제공된 액세스 키가 존재하며 다른 사용자에게 속합니다.	409 충돌

1.15. 키 제거

기존 키를 제거합니다.

capabilities

> **`users=write`**

구문

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

표 1.26. 요청 매개변수

이름	설명	유형	예제	필수 항목
액세스 키	제거할 S3 키 쌍에 속하는 S3 액세스 키입니다.	문자열	**AB01C2D3EF45 G6H7IJ8K**	있음
uid	키를 제거할 사용자입니다.	문자열	**foo_user**	없음
subuser	키를 제거할 하위 사용자입니다.	문자열	**sub_foo**	없음
key-type	제거할 키 유형은 swift, s3입니다. **참고** swift 키를 제거해야 합니다.	문자열	**swift**	없음

특수 오류 응답

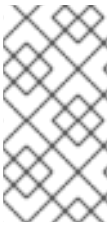
없음.

응답 엔티티

없음.

1.16. 버킷 알림

스토리지 관리자는 이러한 **API**를 사용하여 버킷 알림 메커니즘에 대한 구성 및 제어 인터페이스를 제공할 수 있습니다. **API** 주제는 특정 끝점의 정의를 포함하는 오브젝트라는 이름입니다. 버킷 알림은 주제를 특정 버킷과 연결합니다. *S3 버킷 작업* 섹션에서는 버킷 알림에 대한 자세한 내용을 제공합니다.



참고

모든 주제 작업에서 매개 변수는 인코딩된 **URL**이며 **application/x-www-form-urlencoded** 콘텐츠 유형을 사용하여 메시지 본문에 전송됩니다.

참고

주제 업데이트를 적용하려면 주제와 이미 연결된 모든 버킷 알림을 다시 생성해야 합니다.

1.16.1. 사전 요구 사항

- **Ceph Object Gateway**에서 버킷 알림을 생성합니다.

1.16.2. 항목 생성

버킷 알림을 생성하기 전에 주제를 생성할 수 있습니다. 주제는 **SNS(Simple Notification Service)** 엔티티 및 모든 주제 작업, 즉 , 생성,삭제,목록 및 가져오기 작업을 의미합니다. 항목에 버킷 알림이 생성될 때 사용되는 끝점 매개 변수가 있어야 합니다. 요청이 성공하면 응답에 버킷 알림 요청에서 이 주제를 참조하는 데 사용할 수 있는 **topic Amazon Resource Name(ARN)**이 포함됩니다.

참고

topic_arn 은 버킷 알림 구성을 제공하며 주제를 만든 후 생성됩니다.

사전 요구 사항

- 실행 중인 **Red Hat Ceph Storage** 클러스터.
- 루트 수준 액세스.
- **Ceph** 개체 게이트웨이 설치.
- 사용자 액세스 키 및 시크릿 키입니다.
- 엔드포인트 매개변수입니다.

절차

1. 다음 요청 형식을 사용하여 주제를 생성합니다.

구문

```
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]
```

요청 매개 변수는 다음과 같습니다.

- **endpoint :** 알림을 보낼 끝점의 **URL**입니다.
-

OpaqueData: 불투명 데이터가 주제 구성에 설정되고 해당 주제에서 트리거한 모든 알림에 추가됩니다.

- **HTTP 끝점:**
  - **URL: http[s]://*FQDN*[: *PORT*]**
  - 포트 기본값은 다음과 같습니다 **:** 그에 따라 **HTTP[S]**에 **80/443**을 사용합니다.
  - **verify-ssl:** 클라이언트에서 서버 인증서를 검증할지 여부를 나타냅니다. 기본값은 **true**입니다.**By default, this is true.**
- **AMQP0.9.1 끝점:**
  - **URL: amqp://[*USER* : *PASSWORD* @] *FQDN* [: *PORT*][/*VHOST*].**
  - 사용자 및 암호의 기본값은 **guest** 및 **guest** 입니다.
  - 사용자 및 암호는 **HTTPS**에서만 제공할 수 있습니다. 그렇지 않으면 주제 생성 요청이 거부됩니다.
  - 포트 기본값은**: 5672**입니다.
  - **vhost** 기본값은 "/"입니다.
  - **AMQP-exchange:** 교환이 있어야 하며 주제를 기반으로 메시지를 라우팅할 수 있어야 합니다. **AMQP0.9.1**의 필수 매개변수입니다. 동일한 엔드포인트를 가리키는 다른 주제는 동일한 교환기를 사용해야 합니다.
  - **AMQP-ack-level:** 최종 대상으로 메시지가 전달되기 전에 브로커가 지속되기 전에 브로커가 지속되므로 결국 인정을 종료해야 합니다. 세 가지 승인 방법이 있습니다.

- None: 브로커에 전송되는 경우 메시지가 배달되는 것으로 간주됩니다.
- broker: 기본적으로 브로커가 승인한 경우 메시지가 배달되는 것으로 간주됩니다.
- 라우팅 가능: 브로커가 소비자로 라우팅할 수 있는 경우 전달되는 메시지가 표시됩니다.

참고

특정 매개 변수의 키와 값은 동일한 줄 또는 특정 순서로 상주할 필요는 없지만 동일한 인덱스를 사용해야 합니다. 특성 인덱싱은 순차적이거나 특정 값에서 시작할 필요가 없습니다.

참고

topic-name 은 AMQP 주제를 위해 사용됩니다.

- Kafka 끝점:
  - URL: kafka://[*USER*: *PASSWORD* @] *FQDN*[: *PORT*].
  - 기본적으로 use-ssl 이 false 로 설정된 경우. use-ssl 이 true 로 설정되면 브로커 연결에 보안 연결이 사용됩니다.
  - ca-location 이 제공되고 보안 연결이 사용되는 경우 브로커를 인증하는 기본 CA 대신 지정된 CA가 사용됩니다.
  - 사용자 및 암호는 HTTP[S]를 통해서만 제공할 수 있습니다. 그렇지 않으면topic 생성 요청이 거부됩니다.
  - 사용자 및 암호는 use-ssl 과 함께 제공될 수 있지만 그렇지 않은 경우 브로커에 대한 연결이 실패합니다.

- 포트 기본값은: **9092**입니다.

- **Kafka-ack-level:** 메시지가 최종 목적지로 전달되기 전에 브로커에서 지속될 수 있기 때문에 승인이 필요하지 않습니다. 두 가지 승인 방법이 있습니다.

  - **None:** 브로커에 전송되는 경우 메시지가 배달되는 것으로 간주됩니다.

  - **broker:** 기본적으로 브로커가 승인한 경우 메시지가 배달되는 것으로 간주됩니다.

2. 다음 형식으로 응답을 생성합니다.

구문

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

참고

응답의 **ARMN(Amazon Resource Name)**의 주제는 **arn:aws:sns:*ZONE_GROUP*:*TENANT : topic***

다음은 **AMQP0.9.1** 끝점의 예입니다.

구문

```
"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. 주제 정보 얻기

특정 항목에 대한 정보를 반환합니다. 제공된 경우 끝점 정보가 포함될 수 있습니다.

사전 요구 사항

- 실행 중인 **Red Hat Ceph Storage** 클러스터.
- 루트 수준 액세스.
- **Ceph** 개체 게이트웨이 설치.
- 사용자 액세스 키 및 시크릿 키입니다.
- 엔드포인트 매개변수입니다.

절차

1. 다음 요청 형식으로 주제 정보를 가져옵니다.

구문

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

응답 형식의 예는 다음과 같습니다.**Here is an example of the response format:**

```
<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>
```

태그 및 해당 정의**:**

- **User:** 주제를 생성한 사용자의 이름입니다**.**
- **Name:** 주제의 이름입니다**.**
- **EndpointAddress:** 엔드포인트 **URL**입니다**.** 엔드포인트 **URL**에 사용자 및 암호 정보가 포함된 경우 **HTTPS**를 통해 요청해야 합니다**.** 그렇지 않은 경우 주제 가져오기 요청이 거부됩니다**.**
- **EndPointArgs:** 끝점 인수입니다**.**
- **EndpointTopic:** 엔드포인트로 전송할 주제 이름은 위의 주제 이름과 다를 수 있습니다**.**

- **TopicArn:** 주제입니다.

1.16.4. 주제 나열

사용자가 정의한 주제를 나열합니다.

사전 요구 사항

- 실행 중인 **Red Hat Ceph Storage** 클러스터.
- 루트 수준 액세스.
- **Ceph** 개체 게이트웨이 설치.
- 사용자 액세스 키 및 시크릿 키입니다.
- 엔드포인트 매개변수입니다.

절차

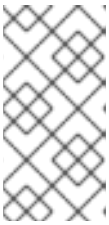
1. 다음 요청 형식으로 주제 정보를 나열합니다.

```
POST
Action=ListTopics
```

응답 형식의 예는 다음과 같습니다.**Here is an example of the response format:**

```
<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>
```



참고

끝점 **URL**에 사용자 및 암호 정보가 포함된 경우 모든 항목에서 **HTTPS**를 통해 요청을 수행해야 합니다. 그렇지 않은 경우 주제 목록 요청이 거부됩니다.

1.16.5. 주제 삭제

삭제된 주제를 제거하면 작업이 없고 오류가 발생합니다.

사전 요구 사항

- 실행 중인 **Red Hat Ceph Storage** 클러스터.
- 루트 수준 액세스.
- **Ceph** 개체 게이트웨이 설치.
- 사용자 액세스 키 및 시크릿 키입니다.
- 엔드포인트 매개변수입니다.

절차

1. 요청 형식을 사용하여 주제를 삭제합니다.

구문

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

응답 형식의 예는 다음과 같습니다.Here is an example of the response format:

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

1.16.6. 이벤트 레코드

이벤트에는 **Ceph Object Gateway**에서 수행한 작업에 대한 정보가 있으며, 선택한 엔드포인트(예: **HTTP, HTTPS, Kafka** 또는 **AMQ0.9.1**)를 통해 페이로드로 전송됩니다. 이벤트 레코드는 **JSON** 형식으로 되어 있습니다.

예제

```
{"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"
    }
]}
```


다음은 이벤트 레코드 키와 해당 정의입니다.

- **awsRegion: Zonegroup.**
- **eventTime:** 이벤트가 트리거된 시기를 나타내는 타임 스탬프입니다.
- **event name :** 이벤트의 유형입니다.
- **userIdentity.principalId:** 이벤트를 트리거한 사용자의 **ID**입니다.
- **requestParameters.sourceIPAddress -** 이벤트를 트리거한 클라이언트의 **IP** 주소입니다. 이 필드는 지원되지 않습니다.

- **responseElements.x-amz-request-id:** 이벤트를 트리거한 요청 **ID**입니다.
- **responseElements.x_amz_id_2:** 이벤트가 트리거된 **Ceph Object Gateway**의 **ID**입니다. **ID** 형식은 ***RGWID-ZONE-ZONEGROUP*** 입니다.
- **s3.configurationId:** 이벤트를 생성한 알림 **ID**입니다.
- **s3.bucket.name:** 버킷의 이름입니다.
- **s3.bucket.ownerIdentity.principalId:** 버킷의 소유자입니다.
- **s3.bucket.arn:** 버킷의 **Amazon Resource Name (ARN)**입니다.
- **s3.bucket.id:** 버킷의 **ID**입니다.
- **s3.object.key:** 오브젝트 키입니다.
- **s3.object.size:** 오브젝트의 크기입니다.
- **s3.object.eTag:** 오브젝트 **etag.**
- **s3.object.version:** 버전이 지정된 버킷의 오브젝트 버전입니다.
- **s3.object.sequencer: Monotonically increasing 16**진수 개체별 변경 식별자입니다.
- **s3.object.metadata: x-amz-meta** 로 전송되는 오브젝트에 설정된 모든 메타데이터입니다.

- **s3.object.tags:** 오브젝트에 설정된 모든 태그입니다.
- **s3.eventId:** 이벤트의 고유 ID입니다.
- **s3.opaqueData: Opaque** 데이터는 주제 구성에 설정되고 해당 주제에서 트리거한 모든 알림에 추가됩니다.

추가 리소스

- 자세한 내용은 *이벤트 메시지 구조를* 참조하십시오.
- 자세한 내용은 ***Red Hat Ceph Storage** 개발자 가이드* 의 *지원 이벤트 유형* 섹션을 참조하십시오.

1.16.7. 지원되는 이벤트 유형

다음과 같은 이벤트 유형이 지원됩니다.

- **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. 추가 리소스

- 자세한 내용은 ***Creating bucket notifications*** 섹션을 참조하십시오.

1.17. 버킷 정보 가져오기

*기존 버킷의 하위 집합에 대한 정보를 가져옵니다. 버킷 없이 **uid** 를 지정하면 사용자에게 속한 모든 버킷이 반환됩니다. 버킷 만 지정하면 특정 버킷에 대한 정보가 검색됩니다.*

capabilities

> `***buckets=read***`

구문

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

*표 **1.27.** 요청 매개변수*

이름	설명	유형	예제	필수 항목
bucket	정보를 반환하는 버킷입니다.	문자열	**foo_bucket**	없음
uid	버킷 정보를 검색할 사용자입니다.	문자열	**foo_user**	없음

이름	설명	유형	예제	필수 항목
통계	버킷 통계 반환.	부울	true [False]	없음

표 1.28. 응답 엔티티

이름	설명	유형	parent
통계	버킷 정보당.	컨테이너	해당 없음
buckets	하나 이상의 버킷 컨테이너 목록을 포함합니다.	컨테이너	**bucket**
단일 버킷 정보용 컨테이너입니다.	컨테이너	**buckets**	**name**
버킷의 이름입니다.	문자열	**bucket**	**pool**
버킷이 저장된 풀입니다.	문자열	**bucket**	**id**
고유한 버킷 ID입니다.	문자열	**bucket**	**marker**
내부 버킷 태그.	문자열	**bucket**	**소유자**
버킷 소유자의 사용자 ID입니다.	문자열	**bucket**	**사용법**
스토리지 사용량 정보.	컨테이너	**bucket**	**index**

요청이 성공하면 원하는 버킷 정보가 포함된 버킷 컨테이너를 반환합니다.

표 1.29. 특수 오류 응답

이름	설명	코드
IndexRepairFailed	버킷 인덱스 복구에 실패했습니다.	409 충돌

1.18. 버킷 인덱스 확인

기존 버킷의 인덱스를 확인합니다.

참고

***check-objects** 가 있는 다중 파트 오브젝트 계정을 확인하려면수정 사항을 **True**로 설정해야 합니다.*

capabilities

buckets=write

구문

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

표 1.30. 요청 매개변수

이름	설명	유형	예제	필수 항목
bucket	정보를 반환하는 버킷입니다.	문자열	**foo_bucket**	있음
check-objects	다중 파트 오브젝트 회계를 확인합니다.	부울	true [False]	없음
fix	또한 확인할 때 버킷 인덱스를 수정합니다.	부울	false [False]	없음

표 1.31. 응답 엔티티

이름	설명	유형
index	버킷 인덱스의 상태.	문자열

표 1.32. 특수 오류 응답

이름	설명	코드
IndexRepairFailed	버킷 인덱스 복구에 실패했습니다.	409 충돌

1.19. 버킷 제거

기존 버킷을 제거합니다.

capabilities

> *`buckets=write`*

구문

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

표 1.33. 요청 매개변수

이름	설명	유형	예제	필수 항목
bucket	제거할 버킷입니다.	문자열	**foo_bucket**	있음
purge-objects	삭제하기 전에 버킷 오브젝트를 제거합니다.	부울	true [False]	없음

응답 엔티티

없음.

표 1.34. 특수 오류 응답

이름	설명	코드
BucketNotCrashLoopBackOff	비어 있지 않은 버킷을 삭제하려고 했습니다.	409 충돌
ObjectRemovalFailed	개체를 제거할 수 없습니다.	409 충돌

1.20. 버킷 연결

버킷을 지정된 사용자에게 연결하고 이전 사용자의 버킷을 연결합니다.

capabilities

> `***buckets=write***`

구문

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

표 1.35. 요청 매개변수

이름	설명	유형	예제	필수 항목
bucket	연결할 버킷입니다.	문자열	**foo_bucket**	있음
uid	버킷을 연결할 사용자 ID입니다.	문자열	**foo_user**	있음

표 1.36. 응답 엔티티

이름	설명	유형	parent
bucket	단일 버킷 정보용 컨테이너입니다.	컨테이너	해당 없음
name	버킷의 이름입니다.	문자열	**bucket**
pool	버킷이 저장된 풀입니다.	문자열	**bucket**
id	고유한 버킷 ID입니다.	문자열	**bucket**
marker	내부 버킷 태그.	문자열	**bucket**
소유자	버킷 소유자의 사용자 ID입니다.	문자열	**bucket**
사용법	스토리지 사용량 정보.	컨테이너	**bucket**
index	버킷 인덱스의 상태.	문자열	**bucket**

표 1.37. 특수 오류 응답

이름	설명	코드
BucketUnlinkFailed	지정된 사용자로부터 버킷을 연결할 수 없습니다.	409 충돌
BucketLinkFailed	지정된 사용자에게 버킷을 연결할 수 없습니다.	409 충돌

1.21. 버킷 연결 해제

지정된 사용자의 버킷을 분리합니다. 버킷 소유권을 변경하는 데 주로 유용합니다.

capabilities

> `*buckets=write*`

구문

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

표 1.38. 요청 매개변수

이름	설명	유형	예제	필수 항목
bucket	연결할 버킷입니다.	문자열	**foo_bucket**	있음
uid	버킷을 언로드할 사용자 ID입니다.	문자열	**foo_user**	있음

응답 엔티티

없음.

표 1.39. 특수 오류 응답

이름	설명	코드
BucketUnlinkFailed	지정된 사용자로부터 버킷을 연결할 수 없습니다.	409 충돌

1.22. 버킷 또는 오브젝트 정책 가져오기

오브젝트 또는 버킷 정책을 읽습니다.

capabilities

```
`buckets=read`
```

구문

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

표 1.40. 요청 매개변수

이름	설명	유형	예제	필수 항목
bucket	정책을 읽을 버킷입니다.	문자열	**foo_bucket**	있음
object	정책을 읽을 오브젝트입니다.	문자열	**foo.txt**	없음

표 1.41. 응답 엔티티

이름	설명	유형	parent
policy	액세스 제어 정책.	컨테이너	해당 없음

성공하면 오브젝트 또는 버킷 정책을 반환합니다.

표 1.42. 특수 오류 응답

이름	설명	코드
IncompleteBody	버킷 정책 요청 또는 버킷에 대해 버킷이 지정되지 않았으며 오브젝트 정책 요청에 대해 오브젝트가 지정되지 않았습니다.	400 잘못된 요청

1.23. 오브젝트 제거

기존 개체를 제거합니다.Removes an existing object.

참고

소유자가 일시 중지되지 않도록 해야 합니다.

capabilities

> **`buckets=write`**

구문

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

표 1.43. 요청 매개변수

이름	설명	유형	예제	필수 항목
bucket	제거할 개체가 포함된 버킷입니다.	문자열	**foo_bucket**	있음
object	제거할 요소입니다.The object to remove.	문자열	**foo.txt**	있음

응답 엔티티

없음.

표 1.44. 특수 오류 응답

이름	설명	코드
NoSuchObject	지정된 오브젝트가 존재하지 않습니다.	404 찾을 수 없음

이름	설명	코드
ObjectRemovalFailed	개체를 제거할 수 없습니다.	409 충돌

1.24. 할당량

*관리 운영 **API**를 사용하면 사용자가 소유한 사용자 및 버킷에 할당량을 설정할 수 있습니다. 쿼터에는 버킷의 최대 오브젝트 수와 최대 스토리지 크기(**MB**)가 포함됩니다.*

*할당량을 보려면 **users=read** 기능이 있어야 합니다. 할당량을 설정, 수정 또는 비활성화하려면 사용자에게 **users=write** 기능이 있어야 합니다.*

할당량에 유효한 매개변수는 다음과 같습니다.

- *버킷: 버킷 옵션을 사용하면 사용자가 소유한 버킷에 할당량을 지정할 수 있습니다.*
- ***Maximum Objects: max-objects** 설정을 사용하면 최대 오브젝트 수를 지정할 수 있습니다. 음수 값은 이 설정을 비활성화합니다.*
- *최대 크기: **max-size** 옵션을 사용하면 최대 바이트 수에 대한 할당량을 지정할 수 있습니다. 음수 값은 이 설정을 비활성화합니다.*
- *할당량 범위: **quota-scope** 옵션은 할당량 범위를 설정합니다. 옵션은 버킷 과 사용자입니다.*

1.25. 사용자 할당량 가져오기

할당량을 가져오려면 사용자에게 읽기 권한으로 설정된 사용자 기능이 있어야 합니다.

구문

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

1.26. 사용자 할당량 설정

할당량을 설정하려면 사용자에게 쓰기 권한으로 설정된 사용자 기능이 있어야 합니다.

구문

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

*콘텐츠에 해당 읽기 작업에서 인코딩된 할당량 설정에 대한 **JSON** 표시가 포함되어야 합니다.*

1.27. 버킷 할당량 가져오기

버킷 할당량을 가져오려면 사용자에게 읽기 권한으로 사용자 기능을 설정해야 합니다.

구문

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

1.28. 버킷 할당량 설정

할당량을 설정하려면 사용자에게 쓰기 권한으로 설정된 사용자 기능이 있어야 합니다.

구문

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

*콘텐츠에 해당 읽기 작업에서 인코딩된 할당량 설정에 대한 **JSON** 표시가 포함되어야 합니다.*

1.29. 개별 버킷에 대한 할당량 설정

할당량을 설정하려면 사용자에게 쓰기 권한으로 버킷 기능을 설정해야 합니다.

구문

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

*콘텐츠에는 할당량 설정의 **JSON** 표현이 포함되어야 합니다.*

1.30. 사용 정보 받기

대역폭 사용량 정보 요청.

capabilities

```
`usage=read`
```

구문

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

표 1.45. 요청 매개변수

이름	설명	유형	필수 항목
uid	정보가 요청된 사용자입니다.	문자열.	있음
start	요청된 데이터의 시작 시간을 지정하는 날짜 및 (선택 사항) 시간입니다. E.g., **2012-09-25 16:00:00**	문자열	없음
end	요청된 데이터의 종료 시간을 지정하는 날짜 및 (선택 사항) 시간(선택 사항)입니다. E.g., **2012-09-25 16:00:00**	문자열	없음
show-entries	데이터 항목을 반환할지 여부를 지정합니다.	부울	없음
show-summary	데이터 요약을 반환할지 여부를 지정합니다.	부울	없음

표 1.46. 응답 엔티티

이름	설명	유형
사용법	사용 정보를 위한 컨테이너입니다.	컨테이너
항목	사용 항목 정보를 위한 컨테이너입니다.	컨테이너
user	사용자 데이터 정보를 위한 컨테이너입니다.	컨테이너
소유자	버킷을 소유하는 사용자의 이름입니다.	문자열
bucket	버킷 이름입니다.	문자열
time	데이터가 지정되는 시간(첫 번째 관련 시간)으로 반올림됩니다.	문자열
epoch	**1/1/1970** 이후 지정된 시간(초)입니다.	문자열
카테고리	통계 카테고리에 대한 컨테이너입니다.	컨테이너
항목	통계 항목에 대한 컨테이너입니다.	컨테이너
category	통계가 제공되는 요청 카테고리의 이름입니다.	문자열
bytes_sent	Ceph Object Gateway에서 보내는 바이트 수입니다.	정수

이름	설명	유형
bytes_received	Ceph Object Gateway에서 수신한 바이트 수입니다.	정수
ops	작업 수입니다.	정수
successful_ops	성공한 작업 수입니다.	정수
summary	통계 요약을 위한 컨테이너입니다.	컨테이너
total	통계 요약 합계에 대한 컨테이너입니다.	컨테이너

성공하면 응답에 요청된 정보가 포함됩니다.

1.31. 사용 정보 제거

사용 정보를 제거합니다. 날짜가 지정되지 않은 경우 모든 사용 정보를 제거합니다.

capabilities

> `***usage=write***`

구문

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

*표 **1.47.** 요청 매개변수*

이름	설명	유형	예제	필수 항목
uid	정보가 요청된 사용자입니다.	문자열	**foo_user**	없음

이름	설명	유형	예제	필수 항목
start	요청된 데이터의 시작 시간을 지정하는 날짜 및 (선택 사항) 시간입니다.	문자열	**2012-09-25 16:00:00**	없음
end	요청된 데이터의 종료 시간을 지정하는 날짜 및 (선택 사항) 시간(선택 사항)입니다.	문자열	**2012-09-25 16:00:00**	없음
remove-all	다중 사용자 데이터 제거를 승인하기 위해 **uid** 를 지정하지 않은 경우 필수 항목입니다.	부울	true [False]	없음

1.32. 표준 오류 응답

다음 표에서는 표준 오류 응답 및 해당 설명을 자세히 설명합니다.

이름	설명	코드
AccessDenied	액세스 거부.	403 forbidden
InternalError	내부 서버 오류.	500 내부 서버 오류
NoSuchUser	사용자가 존재하지 않습니다.	404 찾을 수 없음
NoSuchBucket	버킷이 존재하지 않습니다.	404 찾을 수 없음
NoSuchKey	이러한 액세스 키가 없습니다.	404 찾을 수 없음

2장. CEPH OBJECT GATEWAY 및 S3 API

개발자는 Amazon S3 데이터 액세스 모델과 호환되는 RESTful API(애플리케이션 프로그래밍 인터페이스)를 사용할 수 있습니다. Ceph Object Gateway를 통해 Red Hat Ceph Storage 클러스터에 저장된 버킷과 개체를 관리할 수 있습니다.

2.1. 사전 요구 사항

- 실행 중인 Red Hat Ceph Storage 클러스터.
- RESTful 클라이언트.

2.2. S3 제한 사항

중요

다음 제한 사항을 주의해서 사용해야 합니다. 하드웨어 선택과 관련된 영향이 있으므로 Red Hat 계정 팀과 항상 이러한 요구 사항을 논의해야 합니다.

- Amazon S3 사용 시 최대 오브젝트 크기: 개별 Amazon S3 오브젝트의 크기는 최소 0B에서 최대 5TB까지 다양합니다. PUT 단일에 업로드할 수 있는 가장 큰 오브젝트는 5GB입니다. 100MB보다 큰 개체의 경우 Multipart Upload 기능을 사용하는 것이 좋습니다.
- Amazon S3 사용 시 최대 메타데이터 크기: 오브젝트에 적용할 수 있는 전체 사용자 메타데이터 크기에 정의된 제한이 없지만 단일 HTTP 요청은 16,000바이트로 제한됩니다.
- Red Hat Ceph Storage 클러스터가 S3 오브젝트 및 메타데이터를 저장하기 위해 생성하는 데이터 오버헤드 양: 200-300바이트와 오브젝트 이름의 길이입니다. 버전화된 오브젝트는 버전 수에 비례하는 추가 공간을 사용합니다. 또한 일시적인 오버헤드는 다중 부분 업로드 및 기타 트랜잭션 업데이트 중에 생성되지만 이러한 오버헤드는 가비지 수집 중에 복구됩니다.

추가 리소스

- 지원되지 않는 헤더 필드에 대한 자세한 내용은 Red Hat Ceph Storage 개발자 가이드 를 참조하십시오.

2.3. S3 API를 사용하여 CEPH 오브젝트 게이트웨이 액세스

개발자는 Amazon S3 API를 사용하기 전에 Ceph Object Gateway 및 STS(Secure Token Service)에 대한 액세스를 구성해야 합니다.

2.3.1. 사전 요구 사항

- 실행 중인 Red Hat Ceph Storage 클러스터.
- 실행 중인 Ceph 오브젝트 게이트웨이.
- RESTful 클라이언트.

2.3.2. S3 인증

Ceph Object Gateway에 대한 요청은 인증되거나 인증되지 않을 수 있습니다. Ceph Object Gateway는 인증되지 않은 요청이 익명 사용자가 전송된다고 가정합니다. Ceph Object Gateway는 카나리아 ACL을 지원합니다.

대부분의 사용 사례의 경우 클라이언트는 Amazon SDK의 AmazonS3Client 및 Python Boto와 같은 기존 오픈 소스 라이브러리를 사용합니다. 오픈 소스 라이브러리를 사용하면 액세스 키와 시크릿 키를 전달하면 라이브러리에서 요청 헤더와 인증 서명을 빌드합니다. 그러나 요청을 작성하고 서명할 수도 있습니다.

요청을 인증하려면 Ceph Object Gateway 서버로 전송하기 전에 요청에 대한 액세스 키 및 기본 64 인코딩 해시 기반 메시지 인증 코드(HMAC)를 포함해야 합니다. Ceph Object Gateway는 S3 호환 인증 방법을 사용합니다.

예제

```
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
```

위 예에서 **ACCESS_KEY** 를 액세스 키 **ID** 뒤에 콜론(:)의 값으로 바꿉니다. **HASH_OF_HEADER_AND_SECRET** 을 정식화된 헤더 문자열 해시 및 액세스 키 **ID**에 해당하는 시크릿으로 교체합니다.

헤더 문자열 및 시크릿의 해시 생성

헤더 문자열 및 시크릿의 해시를 생성하려면 다음을 수행합니다.

1. 헤더 문자열의 값을 가져옵니다.**Gets the value of the header string.**
2. 요청 헤더 문자열을 표준 형식으로 정규화합니다.
3. **SHA-1** 해시 알고리즘을 사용하여 **HMAC**를 생성합니다.
4. **hmac** 결과를 **base-64**로 인코딩합니다.

헤더 정규화

헤더를 표준 형식으로 정규화하려면 다음을 수행합니다.

1. 모든 **content-** 헤더를 가져옵니다.
2. **Content- type** 및 **content- md5** 를 제외한 모든 **content-** 헤더를 제거합니다.
3. **content-** 헤더 이름이 소문자인지 확인합니다.
4. 사전순 으로 콘텐츠 헤더를 정렬합니다.
5. 날짜 헤더가 있는지 확인 및 지정된 날짜가 오프셋이 아닌 **kafka**를 사용하는지 확인합니다.

6. *x-amz-* 로 시작하는 모든 헤더를 가져옵니다.

7. *x-amz-* 헤더가 모두 소문자인지 확인합니다.

8. *X -amz-* 헤더를 사전순으로 정렬합니다.

9. 동일한 필드 이름의 여러 인스턴스를 단일 필드로 결합하고 필드 값을 쉼표로 구분합니다.

10. 헤더 값의 공백과 줄 바꿈을 하나의 공백으로 바꿉니다.

11. 콜론 앞과 뒤에 공백을 제거합니다.

12. 각 헤더 뒤에 새 행을 추가합니다.

13. 헤더를 요청 헤더에 다시 병합합니다.

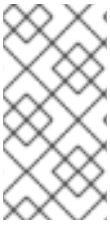
HASH_OF_HEADER_AND_SECRET 을 *base-64*로 인코딩된 *HMAC* 문자열로 바꿉니다.

추가 리소스

- 자세한 내용은 *Amazon Simple Storage Service* 설명서의 *REST* 요청 서명 및 인증 섹션을 참조하십시오.

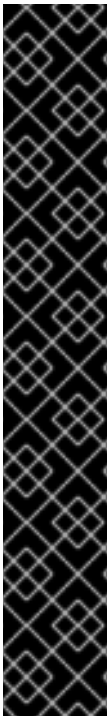
2.3.3. S3 서버 측 암호화

*Ceph Object Gateway*는 *S3* 애플리케이션 프로그래밍 인터페이스(*API*)에 대해 업로드된 오브젝트의 서버 측 암호화를 지원합니다. 서버 측 암호화는 *S3* 클라이언트가 암호화되지 않은 형식으로 *HTTP*를 통해 데이터를 전송하며, *Ceph Object Gateway*는 암호화된 형식으로 해당 데이터를 *Red Hat Ceph Storage* 클러스터에 저장합니다.



참고

Red Hat은 Static Large Object(SLO) 또는 Dynamic Large Object(DLO)의 S3 개체 암호화를 지원하지 않습니다.



중요

암호화를 사용하려면 클라이언트 요청이 SSL 연결을 통해 요청을 보내야 합니다. Ceph Object Gateway에서 SSL을 사용하지 않는 한 Red Hat은 클라이언트의 S3 암호화를 지원하지 않습니다. 그러나 관리자는 런타임에 rgw_crypt_require_ssl 구성 설정을 false 로 설정하고, Ceph 구성 파일에서 false 로 설정하고, 게이트웨이 인스턴스를 다시 시작하거나, Ceph 구성 파일에서 false 로 설정하고, Ceph Object Gateway를 위해 Ansible 플레이북을 재생하여 테스트 중에 SSL을 비활성화할 수 있습니다.

프로덕션 환경에서는 SSL을 통해 암호화된 요청을 보낼 수 없습니다. 이러한 경우 서버 측 암호화와 함께 HTTP를 사용하여 요청을 보냅니다.

서버 측 암호화를 사용하여 HTTP를 구성하는 방법에 대한 자세한 내용은 아래의 추가 리소스 섹션을 참조하십시오.

암호화 키 관리를 위한 두 가지 옵션이 있습니다.

고객 제공 키

고객 제공 키를 사용하는 경우 S3 클라이언트는 암호화된 데이터를 읽거나 쓰기 위해 각 요청과 함께 암호화 키를 전달합니다. 이러한 키를 관리하는 것은 고객의 책임입니다. 고객은 각 오브젝트를 암호화하는 데 사용된 Ceph Object Gateway의 키를 기억해야 합니다.

Ceph Object Gateway는 Amazon SSE-C 사양에 따라 S3 API에서 고객 제공 키 동작을 구현합니다.

고객은 키 관리를 처리하고 S3 클라이언트가 키를 Ceph Object Gateway에 전달하므로 Ceph Object Gateway에는 이 암호화 모드를 지원하기 위한 특별한 구성이 필요하지 않습니다.

키 관리 서비스

키 관리 서비스를 사용하는 경우 보안 키 관리 서비스는 키를 저장하고 Ceph Object Gateway는 데이터를 암호화하거나 해독하는 요청을 처리하기 위해 필요에 따라 검색합니다.

Ceph Object Gateway는 Amazon SSE-KMS 사양에 따라 S3 API에서 키 관리 서비스 동작을 구현합니다.

중요

현재 테스트된 유일한 키 관리 구현은 HashiCorp Vault 및 OpenStack Barbican입니다. 그러나 OpenStack Barbican은 기술 프리뷰이며 프로덕션 시스템에서 사용할 수 없습니다.

추가 리소스

- *Amazon SSE-C*
- *Amazon SSE-KMS*
- *서버 측 암호화 구성*
- *HashiCorp Vault*

2.3.4. S3 액세스 제어 목록

Ceph Object Gateway는 S3 호환 ACL(Access Control List) 기능을 지원합니다. ACL은 사용자가 버킷 또는 오브젝트에서 수행할 수 있는 작업을 지정하는 액세스 권한 부여 목록입니다. 각 권한 부여는 오브젝트에 적용되는 버킷에 적용할 때 다른 의미를 갖습니다.

표 2.1. 사용자 작업

권한	bucket	개체
READ	grantee는 버킷의 오브젝트를 나열할 수 있습니다.	권한 부여자는 오브젝트를 읽을 수 있습니다.
쓰기	grantee는 버킷에 오브젝트를 작성하거나 삭제할 수 있습니다.	해당 없음
READ_ACP	grantee는 버킷 ACL을 읽을 수 있습니다.	grantee는 오브젝트 ACL을 읽을 수 있습니다.

권한	bucket	개체
WRITE_ACP	grantee는 버킷 ACL을 작성할 수 있습니다.	grantee는 오브젝트 ACL에 쓸 수 있습니다.
FULL_CONTROL	grantee는 버킷의 오브젝트에 대한 전체 권한을 갖습니다.	grantee는 오브젝트 ACL을 읽거나 쓸 수 있습니다.

2.3.5. S3를 사용하여 Ceph Object Gateway에 대한 액세스 준비

게이트웨이 서버에 액세스하기 전에 Ceph Object Gateway 노드에서 몇 가지 사전 요구 사항을 따라야 합니다.

주의

포트 80 을 사용하도록 Ceph 구성 파일을 수정하지 말고 Civetweb 이 기본 Ansible 구성 포트 8080 을 사용하도록 합니다.

사전 요구 사항

- *Ceph Object Gateway 소프트웨어 설치.*
- *Ceph Object Gateway 노드에 대한 루트 수준 액세스.*

절차

1. *루트 로서 방화벽에서 포트 8080 을 엽니다.*

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

2. *오브젝트 게이트웨이 구성 및 관리 가이드에서 언급한 대로 게이트웨이에 사용하는 DNS 서버에 와일드카드를 추가합니다.*

로컬 DNS 캐싱에 대한 게이트웨이 노드를 설정할 수도 있습니다. 이렇게 하려면 다음 단계를 실행합니다.

a.

root 로 dnsmasq 를 설치하고 설정합니다.

```
[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 및 FQDN_OF_GATEWAY_NODE 를 게이트웨이 노드의 IP 주소 및 FQDN으로 바꿉니다.

b.

루트 로서 NetworkManager를 중지합니다.

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

c.

루트 로서 게이트웨이 서버의 IP를 이름 서버로 설정합니다.

```
[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 및 FQDN_OF_GATEWAY_NODE 를 게이트웨이 노드의 IP 주소 및 FQDN으로 바꿉니다.

d.

하위 도메인 요청을 확인합니다.

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

FQDN_OF_GATEWAY_NODE 를 게이트웨이 노드의 FQDN으로 교체합니다.

주의

로컬 DNS 캐싱을 위해 게이트웨이 서버를 설정하는 것은 테스트 목적으로만 사용됩니다. 이 작업을 수행 한 후에는 외부 네트워크에 액세스할 수 없습니다. Red Hat Ceph Storage 클러스터 및 게이트웨이 노드에 적절한 DNS 서버를 사용하는 것이 좋습니다.

3. *Object Gateway Configuration and Administration Guide 에 언급된 대로 S3 액세스에 대한 radosgw 사용자를 신중하게 만들고 생성된 access_key 및 secret_key 를 복사합니다. 이러한 키는 S3 액세스 및 후속 버킷 관리 작업에 필요합니다.*

2.3.6. Ruby AWS S3를 사용하여 Ceph Object Gateway에 액세스

S3 액세스에 대해 aws-s3 gem과 함께 Ruby 프로그래밍 언어를 사용할 수 있습니다. Ruby AWS::S3을 사용하여 Ceph Object Gateway 서버에 액세스하는 데 사용되는 노드에서 아래에 언급된 단계를 실행합니다.

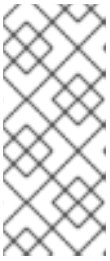
사전 요구 사항

- *Ceph Object Gateway에 대한 사용자 수준 액세스.*
- *Ceph Object Gateway에 액세스하는 노드에 대한 루트 수준 액세스입니다.*
- *인터넷 액세스.*

절차

1. *ruby 패키지를 설치합니다.*

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



참고

*위의 명령은 **ruby**를 설치하고 **ruby gems** 및 **ruby-libs** 와 같은 필수 종속 항목을 설치합니다. 임의의 경우 명령은 모든 종속 항목을 설치하지 않는 경우 별도로 설치합니다.*

2. ***aws-s3 Ruby** 패키지를 설치합니다.*

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

3. *프로젝트 디렉터리를 생성합니다.*

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

4. *연결 파일을 생성합니다.*

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

5. *다음 내용을 **conn.rb** 파일에 붙여넣습니다.*

구문

```
#!/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** 를 **Ceph Object Gateway** 노드의 **FQDN**으로 바꿉니다.*

MY_ACCESS_KEY 및 *MY_SECRET_KEY* 를 *Red Hat Ceph Storage 오브젝트 게이트웨이 구성 및 관리 가이드*에서 언급한 대로 *S3* 액세스를 위해 *radosgw* 사용자를 생성할 때 생성된 *access_key* 및 *secret_key* 로 바꿉니다.

예제

```
#!/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'
)
```

파일을 저장하고 편집기를 종료합니다.

6. *파일을 실행 가능으로 설정합니다.*

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

7. *파일을 실행합니다.*

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

파일에 값을 올바르게 제공한 경우 명령의 출력은 0 이 됩니다.

8. *버킷을 만들기 위한 새 파일을 만듭니다.*

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

다음 내용을 파일에 붙여넣습니다.

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

load 'conn.rb'

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

파일을 저장하고 편집기를 종료합니다.

9. *파일을 실행 가능으로 설정합니다.*

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

10. *파일을 실행합니다.*

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

*명령 출력이 **true** 이면 버킷 **my-new-bucket1** 이 성공적으로 생성되었습니다.*

11. *소유 버킷을 나열할 새 파일을 만듭니다.*

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

다음 내용을 파일에 붙여넣습니다.

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

load 'conn.rb'

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

파일을 저장하고 편집기를 종료합니다.

12. *파일을 실행 가능으로 설정합니다.*

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

13. *파일을 실행합니다.*

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

출력은 다음과 같아야 합니다.

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

14. *오브젝트를 생성하기 위한 새 파일을 생성합니다.*

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

다음 내용을 파일에 붙여넣습니다.

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

load 'conn.rb'

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

파일을 저장하고 편집기를 종료합니다.

15. *파일을 실행 가능으로 설정합니다.*

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

16. *파일을 실행합니다.*

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

*이렇게 하면 **Hello World!!** 문자열이 포함된 **hello.txt** 파일이 생성됩니다.*

17. *버킷의 콘텐츠를 나열할 새 파일을 만듭니다.*

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

다음 내용을 파일에 붙여넣습니다.

```
#!/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
```

파일을 저장하고 편집기를 종료합니다.

18. *파일을 실행 가능하게 만듭니다.*

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

19. *파일을 실행합니다.*

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

출력은 다음과 같이 표시됩니다.

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

20. *빈 버킷을 삭제하기 위해 새 파일을 생성합니다.*

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

다음 내용을 파일에 붙여넣습니다.

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

load 'conn.rb'
```

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

파일을 저장하고 편집기를 종료합니다.

21. *파일을 실행 가능으로 설정합니다.*

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

22. *파일을 실행합니다.*

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

*버킷이 성공적으로 삭제되면 명령은 **0** 을 출력으로 반환합니다.*

참고

__create_bucket.rb__ 파일을 편집하여 빈 버킷(예: __my-new-bucket4,my-new-bucket5__)을 생성합니다. 그런 다음 빈 버킷을 삭제하기 전에 위의 __del_empty_bucket.rb__ 파일을 적절하게 편집합니다.

23. *비어 있지 않은 버킷을 삭제하기 위해 새 파일을 생성합니다.*

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

다음 내용을 파일에 붙여넣습니다.

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

load 'conn.rb'

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

파일을 저장하고 편집기를 종료합니다.

24. *파일을 실행 가능으로 설정합니다.*

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

25. 파일을 실행합니다.

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

버킷이 성공적으로 삭제되면 명령은 **0** 을 출력으로 반환합니다.

26. 오브젝트를 삭제하기 위한 새 파일을 생성합니다.

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

다음 내용을 파일에 붙여넣습니다.

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

load 'conn.rb'

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

파일을 저장하고 편집기를 종료합니다.

27. 파일을 실행 가능으로 설정합니다.

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

28. 파일을 실행합니다.

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

그러면 **hello.txt** 오브젝트가 삭제됩니다.

2.3.7. Ruby AWS SDK를 사용하여 Ceph Object Gateway에 액세스

S3 액세스를 위해 **aws-sdk gem**과 함께 **Ruby** 프로그래밍 언어를 사용할 수 있습니다. **Ruby AWS::SDK** 를 사용하여 **Ceph Object Gateway** 서버에 액세스하는 데 사용되는 노드에서 아래에 언급된

단계를 실행합니다.

사전 요구 사항

- ***Ceph Object Gateway**에 대한 사용자 수준 액세스.*
- ***Ceph Object Gateway**에 액세스하는 노드에 대한 루트 수준 액세스입니다.*
- *인터넷 액세스.*

절차

1. ***ruby** 패키지를 설치합니다.*

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

참고

*위의 명령은 **ruby**를 설치하고 **ruby gems** 및 **ruby-libs** 와 같은 필수 종속 항목을 설치합니다. 임의의 경우 명령은 모든 종속 항목을 설치하지 않는 경우 별도로 설치합니다.*

2. ***aws-sdk Ruby** 패키지를 설치합니다.*

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

3. *프로젝트 디렉터리를 생성합니다.*

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

4. *연결 파일을 생성합니다.*

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

-

5. *다음 내용을 **conn.rb** 파일에 붙여넣습니다.*

구문

```
#!/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** 를 **Ceph Object Gateway** 노드의 **FQDN**으로 바꿉니다. **MY_ACCESS_KEY** 및 **MY_SECRET_KEY** 를 **Red Hat Ceph Storage** 오브젝트 게이트웨이 구성 및 관리 가이드에서 언급한 대로 **S3** 액세스를 위해 **radosgw** 사용자를 생성할 때 생성된 **access_key** 및 **secret_key** 로 바꿉니다.*

예제

```
#!/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'
)
```

파일을 저장하고 편집기를 종료합니다.

6. *파일을 실행 가능으로 설정합니다.*

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

7. *파일을 실행합니다.*

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

*파일에 값을 올바르게 제공한 경우 명령의 출력은 **0** 이 됩니다.*

8. *버킷을 만들기 위한 새 파일을 만듭니다.*

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

다음 내용을 파일에 붙여넣습니다.

구문

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

load 'conn.rb'

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

파일을 저장하고 편집기를 종료합니다.

9. *파일을 실행 가능으로 설정합니다.*

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

10. *파일을 실행합니다.*

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

*명령의 출력이 **true** 이면 버킷 **my-new-bucket2** 가 성공적으로 생성되었습니다.*

11. *소유 버킷을 나열할 새 파일을 만듭니다.*

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

다음 내용을 파일에 붙여넣습니다.

```
#!/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
```

파일을 저장하고 편집기를 종료합니다.

12. *파일을 실행 가능으로 설정합니다.*

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

13. *파일을 실행합니다.*

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

출력은 다음과 같아야 합니다.

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

14. *오브젝트를 생성하기 위한 새 파일을 생성합니다.*

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

다음 내용을 파일에 붙여넣습니다.

```
#!/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'
)
```

파일을 저장하고 편집기를 종료합니다.

15. 파일을 실행 가능으로 설정합니다.

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

16. 파일을 실행합니다.

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

이렇게 하면 **Hello World!!** 문자열이 포함된 **hello.txt** 파일이 생성됩니다.

17. 버킷의 콘텐츠를 나열할 새 파일을 만듭니다.

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

다음 내용을 파일에 붙여넣습니다.

```
#!/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
```

파일을 저장하고 편집기를 종료합니다.

18. *파일을 실행 가능하게 만듭니다.*

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

19. *파일을 실행합니다.*

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

출력은 다음과 같이 표시됩니다.

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

20. *빈 버킷을 삭제하기 위해 새 파일을 생성합니다.*

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

다음 내용을 파일에 붙여넣습니다.

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

load 'conn.rb'

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

파일을 저장하고 편집기를 종료합니다.

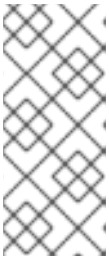
21. *파일을 실행 가능으로 설정합니다.*

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

22. *파일을 실행합니다.*

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

*버킷이 성공적으로 삭제되면 명령은 **0** 을 출력으로 반환합니다.*



참고

*`create_bucket.rb` 파일을 편집하여 빈 버킷을 생성합니다(예: **my-new-bucket6,my-new-bucket7**). 그런 다음 빈 버킷을 삭제하기 전에 위의 `del_empty_bucket.rb` 파일을 적절하게 편집합니다.*

23. *비어 있지 않은 버킷을 삭제하기 위해 새 파일을 생성합니다.*

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

다음 내용을 파일에 붙여넣습니다.

```
#!/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')
```

파일을 저장하고 편집기를 종료합니다.

24. *파일을 실행 가능으로 설정합니다.*

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

25. *파일을 실행합니다.*

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

버킷이 성공적으로 삭제되면 명령은 **0** 을 출력으로 반환합니다.

26. 오브젝트를 삭제하기 위한 새 파일을 생성합니다.

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

다음 내용을 파일에 붙여넣습니다.

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

load 'conn.rb'

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

파일을 저장하고 편집기를 종료합니다.

27. 파일을 실행 가능으로 설정합니다.

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

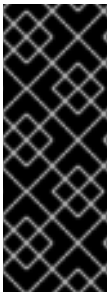
28. 파일을 실행합니다.

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

그러면 **hello.txt** 오브젝트가 삭제됩니다.

2.3.8. PHP를 사용하여 Ceph Object Gateway에 액세스

S3 액세스에 **PHP** 스크립트를 사용할 수 있습니다. 이 절차에서는 버킷 또는 오브젝트 삭제와 같은 다양한 작업을 수행하는 몇 가지 **PHP** 스크립트 예를 제공합니다.



중요

아래 예제는 **php v5.4.16** 및 **aws-sdk v2.8.24** 에 대해 테스트됩니다. **php >= 5.5** 이상이 필요하므로 최신 버전의 **aws-sdk** 를 사용하지 마십시오.**php 5.5** 는 **RHEL 7** 의 기본 리포지토리에서 사용할 수 없습니다. **php 5.5** 를 사용하려면 **epel** 및 기타 타사 리포지토리를 활성화해야 합니다. 또한 **php 5.5** 및 최신 버전의 **aws-sdk** 구성 옵션은 다릅니다.

사전 요구 사항

- 개발 워크스테이션에 대한 루트 수준 액세스.
- 인터넷 액세스.

절차

1. **php** 패키지를 설치합니다.

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

2. **PHP**용 **aws-sdk** 의 **zip** 아카이브를 다운로드하여 압축을 풉니다.
3. 프로젝트 디렉터리를 생성합니다.

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

4. 추출된 **aws** 디렉터리를 프로젝트 디렉터리에 복사합니다. 예를 들어 다음과 같습니다.

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

5. 연결 파일을 생성합니다.

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

6. **conn.php** 파일에 다음 내용을 붙여 넣습니다.

구문

```
<?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 를 게이트웨이 노드의 **FQDN**으로 교체합니다. **MY_ACCESS_KEY** 및 **MY_SECRET_KEY** 를 **Red Hat Ceph Storage** 오브젝트 게이트웨이 구성 및 관리 가이드에서 언급한 대로 **S3** 액세스용 **radosgw** 사용자를 생성할 때 생성된 **access_key** 및 **secret_key** 로 바꿉니다. **PATH_TO_AWS** 를 **php** 프로젝트 디렉터리에 복사한 추출된 **aws** 디렉터리의 절대 경로로 교체합니다.

파일을 저장하고 편집기를 종료합니다.

7. 파일을 실행합니다.

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

파일에 값을 올바르게 제공한 경우 명령의 출력은 **0** 이 됩니다.

8. 버킷을 만들기 위한 새 파일을 만듭니다.

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

다음 내용을 새 파일에 붙여넣습니다.

구문

```
<?php

include 'conn.php';

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

?>
```

파일을 저장하고 편집기를 종료합니다.

9. 파일을 실행합니다.

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

10. 소유 버킷을 나열할 새 파일을 만듭니다.

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

다음 내용을 파일에 붙여넣습니다.

구문

```
<?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";
```

```
}

?>
```

파일을 저장하고 편집기를 종료합니다.

11. *파일을 실행합니다.*

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

출력은 다음과 유사해야 합니다.

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

12. ***hello.txt** 라는 소스 파일을 먼저 생성하여 오브젝트를 생성합니다.*

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

13. *새 **php** 파일을 만듭니다.*

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

다음 내용을 파일에 붙여넣습니다.

구문

```
<?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);

?>
```

파일을 저장하고 편집기를 종료합니다.

14. *파일을 실행합니다.*

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

*이렇게 하면 **hello.txt** 개체가 **my-new-bucket3** 버킷에 생성됩니다.*

15. *버킷의 콘텐츠를 나열할 새 파일을 만듭니다.*

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

다음 내용을 파일에 붙여넣습니다.

구문

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

파일을 저장하고 편집기를 종료합니다.

16. *파일을 실행합니다.*

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

출력은 다음과 유사합니다.

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

17. *빈 버킷을 삭제하기 위해 새 파일을 생성합니다.*

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

다음 내용을 파일에 붙여넣습니다.

구문

```
<?php

include 'conn.php';

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

파일을 저장하고 편집기를 종료합니다.

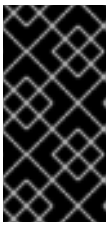
18. *파일을 실행합니다.*

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

*버킷이 성공적으로 삭제되면 명령은 **0** 을 출력으로 반환합니다.*

참고

***create_bucket.php** 파일을 편집하여 빈 버킷(예: **my-new-bucket4,my-new-bucket5**)을 생성합니다. 그런 다음 빈 버킷을 삭제하기 전에 위에서 언급한 **del_empty_bucket.php** 파일을 편집합니다.*



중요

*비어 있지 않은 버킷 삭제는 현재 **PHP 2** 및 **aws-sdk** 버전에서 지원되지 않습니다.*

19. *오브젝트를 삭제하기 위한 새 파일을 생성합니다.*

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

다음 내용을 파일에 붙여넣습니다.

구문

```
<?php

include 'conn.php';

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

파일을 저장하고 편집기를 종료합니다.

20. *파일을 실행합니다.*

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

그러면 **hello.txt** 오브젝트가 삭제됩니다.

2.3.9. AWS CLI를 사용하여 Ceph Object Gateway에 액세스

S3 액세스를 위해 **AWS CLI**를 사용할 수 있습니다. 이 절차에서는 **AWS CLI** 및 **MFA-Delete** 활성화된 버킷에서 오브젝트 삭제와 같은 다양한 작업을 수행하는 몇 가지 예제 명령을 제공합니다.

사전 요구 사항

- **Ceph Object Gateway**에 대한 사용자 수준 액세스.
- 개발 워크스테이션에 대한 루트 수준 액세스.
- **Multi-factor authentication (MFA) TOTP** 토큰은 **radosgw-admin mfa create**를 사용하여 생성되었습니다.

절차

1. **awscli** 패키지를 설치합니다.

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

2. **AWS CLI**를 사용하여 **Ceph Object Storage**에 액세스하도록 **awscli** 를 구성합니다.

구문

```
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** 을 이 프로필을 식별하는 데 사용할 이름으로 교체합니다.*
***MY_ACCESS_KEY** 및 **MY_SECRET_KEY** 를 **Red Hat Ceph Storage** 오브젝트 게이트웨이 구성 및 관리 가이드에서 언급한 대로 **S3** 액세스용 **radosgw** 사용자를 생성할 때 생성된 **access_key** 및 **secret_key** 로 바꿉니다.*

예제

```
[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** 노드의 **FQDN**을 가리키는 별칭을 생성합니다.*

구문

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

***FQDN_OF_GATEWAY_NODE** 를 **Ceph Object Gateway** 노드의 **FQDN**으로 바꿉니다.*

예제

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

4. *새 버킷을 생성합니다.*

구문

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

***MY_PROFILE_NAME** 을 이 프로필을 사용하도록 생성한 이름으로 교체합니다.*
***BUCKET_NAME** 을 새 버킷의 이름으로 바꿉니다.*

예제

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

5. *보유 버킷을 나열합니다.*

구문

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

***MY_PROFILE_NAME** 을 이 프로필을 사용하도록 생성한 이름으로 교체합니다.*

예제

```
[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-Delete에 대한 버킷을 구성합니다.**

구문

```
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** 을 이 프로필을 사용하도록 생성한 이름으로 교체합니다.
- **BUCKET_NAME** 을 새 버킷 이름으로 교체합니다.
- **TOTP_SERIAL** 을 문자열로 바꾸고 **TOTP_PIN** 을 **MFA** 인증 장치에 표시된 현재 핀으로 바꿉니다.
- **TOTP_SERIAL** 은 **S3**에 대해 **radosgw** 사용자를 만들 때 지정된 문자열입니다.
- **MFA TOTP** 토큰 생성에 대한 자세한 내용은 **Red Hat Ceph Storage Object Gateway** 구성 및 관리 가이드 의 새로운 다단계 인증 **TOTP** 토큰 생성 섹션을 참조하십시오.
- **oathtool**을 사용하여 **MFA** 유기 생성에 대한 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드 의 **oathtool**을 사용하여 다단계 인증 생성 섹션을 참조하십시오.

예제

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

7. *버킷 버전 관리 상태의* ***MFA-Delete*** *상태를 확인합니다.*

구문

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

MY_PROFILE_NAME *을 이 프로필을 사용하도록 생성한 이름으로 교체합니다.* ***BUCKET_NAME*** *을 새 버킷 이름으로 교체합니다.*

예제

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

8. ***MFA-Delete*** *활성화된 버킷에 오브젝트를 추가합니다.*

구문

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

- ***MY_PROFILE_NAME*** *을 이 프로필을 사용하도록 생성한 이름으로 교체합니다.*
- ***BUCKET_NAME*** *을 새 버킷 이름으로 교체합니다.*
- ***OBJECT_KEY*** *를 버킷의 오브젝트를 고유하게 식별하는 이름으로 교체합니다.*
- ***LOCAL_FILE*** *을 업로드할 로컬 파일 이름으로 바꿉니다.*

예제

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

9. ***특정 오브젝트의 오브젝트 버전을 나열합니다.***

구문

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

- *MY_PROFILE_NAME 을 이 프로필을 사용하도록 생성한 이름으로 교체합니다.*

- *BUCKET_NAME 을 새 버킷 이름으로 교체합니다.*

- *OBJECT_KEY 를 버킷의 오브젝트를 고유하게 식별하기 위해 지정된 이름으로 교체합니다.*

예제

```
[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 활성화된 버킷에서 오브젝트를 삭제합니다.*

구문

```
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** 을 이 프로필을 사용하도록 생성한 이름으로 교체합니다.*
- ***BUCKET_NAME** 을 삭제할 오브젝트가 포함된 버킷 이름으로 바꿉니다.*
- ***OBJECT_KEY** 를 버킷에서 오브젝트를 고유하게 식별하는 이름으로 교체합니다.*
- ***VERSION_ID** 를 삭제하려는 특정 버전의 버전**ID**로 바꿉니다.*
- ***TOTP_SERIAL** 을 **TOTP** 토큰의 **ID**를 나타내는 문자열로 바꾸고 **TOTP_PIN** 을 **MFA** 인증 장치에 표시되는 현재 핀으로 바꿉니다.*

예제

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

***MFA** 토큰이 포함되어 있지 않으면 아래에 표시된 오류와 함께 요청이 실패합니다.*

예제

```
[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. ***MFA-Delete** 활성화된 버킷에서 오브젝트가 삭제되었는지 확인하기 위해 오브젝트 버전을 나열합니다.*

구문

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

- ***MY_PROFILE_NAME** 을 이 프로필을 사용하도록 생성한 이름으로 교체합니다.*
- ***BUCKET_NAME** 을 버킷 이름으로 교체합니다.*
- ***OBJECT_KEY** 를 버킷에서 오브젝트를 고유하게 식별하는 이름으로 교체합니다.*

예제

```
[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 명령을 사용하여 다단계 인증에 대한 시작 단계 생성

다중 인증 (MFA)을 설정하려면 시간 기반 1 회 암호 (TOTP) 생성기 및 백엔드 MFA 시스템에서 사용할 시크릿 비트를 생성해야 합니다. oathtool 을 사용하여 16진수 및 선택적으로 qrencode 를 생성하여 토큰을 MFA 장치로 가져올 QR 코드를 생성할 수 있습니다.

사전 요구 사항

- *Linux 시스템입니다.*
- *명령줄 쉘에 액세스합니다.*
- *Linux 시스템에 대한 루트 또는 sudo 액세스.*

절차

1. *oathtool 패키지를 설치합니다.*

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

2. *qrencode 패키지를 설치합니다.*

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

3. *urandom Linux 장치 파일에서 30자유를 생성하여 쉘 변수 SEED 에 저장합니다.*

예제

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

4. *SEED 변수에서 echo를 실행하여 shim을 출력합니다.*

예제

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

5. ***SEED*** *를* ***oathtool*** *명령에 제공합니다.*

구문

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

예제

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

823182
```

참고

MFA *장치의 인증자 애플리케이션에 토큰을 추가하려면* ***base32*** *시크릿이 필요합니다.* ***QR*** *코드를 사용하여 토큰을 인증기 애플리케이션으로 가져오거나* ***base32*** *시크릿을 수동으로 추가할 수 있습니다.*

6. *선택 사항:* ***QR*** *코드 이미지 파일을 생성하여 인증기에 토큰을 추가합니다.*

구문

구분

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

TOTP_SERIAL 을 (TOTP) 토큰의 **ID**를 나타내는 문자열로 바꾸고 **BASE32_SECRET** 을 **oathtool**에서 생성한 **Base32** 시크릿으로 바꿉니다.

예제

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

7. 생성된 **QR** 코드 이미지 파일을 스캔하여 **MFA** 장치의 인증 애플리케이션에 토큰을 추가합니다.

8. **radowgw-admin** 명령을 사용하여 사용자를 위한 다단계 인증 **TOTP** 토큰을 만듭니다.

추가 리소스

- **MFA** *TOTP 토큰 생성에 대한 자세한 내용은* ***Red Hat Ceph Storage Object Gateway*** *구성 및 관리 가이드 의 새로운 다단계 인증* **TOTP** 토큰 생성 섹션을 참조하십시오.

2.3.11. 보안 토큰 서비스

Amazon Web Services의 **STS(Secure Token Service)**는 사용자 인증을 위해 일련의 임시 보안 자격 증명을 반환합니다. **Ceph Object Gateway**는 **STS API**(애플리케이션 프로그래밍 인터페이스) 서브 세트를 구현하여 **ID** 및 액세스 관리(**IAM**)에 대한 임시 자격 증명을 제공합니다. 이러한 임시 자격 증명을 사용하면 **Ceph Object Gateway**의 **STS** 엔진을 사용하여 **S3** 호출을 인증합니다. **STS API**에 전달된 매개변수인 **IAM** 정책을 사용하여 임시 인증 정보를 추가로 제한할 수 있습니다.

추가 리소스

- *Amazon Web Services 보안 토큰 서비스 시작 페이지.*

- *STS Lite 및 Keystone에 대한 자세한 내용은 Red Hat Ceph Storage 개발자 가이드 의 Keystone과 함께 STS Lite 구성 섹션을 참조하십시오.*

- *STS Lite 및 Keystone의 제한 사항에 대한 자세한 내용은 Red Hat Ceph Storage 개발자 가이드 의 Keystone과 함께 STS Lite를 사용하는 제한 사항 섹션을 참조하십시오.*

2.3.11.1. 보안 토큰 서비스 애플리케이션 프로그래밍 인터페이스

Ceph Object Gateway는 다음 STS(Secure Token Service) API(애플리케이션 프로그래밍 인터페이스)를 구현합니다.

AssumeRole

이 API는 교차 계정 액세스에 대한 임시 인증 정보 집합을 반환합니다. 이러한 임시 자격 증명을 사용하면 Assume Role API에 연결된 역할 및 정책에 연결된 권한 정책을 모두 사용할 수 있습니다. RoleArn 및 RoleSessionName 요청 매개변수는 필수이지만 다른 요청 매개변수는 선택 사항입니다.

RoleArn

설명

Amazon Resource Name(ARN)에 길이가 20~ 2048자인 것으로 가정할 역할입니다.

유형

문자열

필수 항목

있음

RoleSessionName

설명

위임할 역할 세션 이름을 식별합니다. 역할 세션 이름은 역할을 가정할 때 다른 주체 또는 다른 이유로 세션을 고유하게 식별할 수 있습니다. 이 매개변수의 값은 2~64자 길이입니다. =, ,, ., @, and - characters는 허용되지만 공백은 허용되지 않습니다.

유형

문자열

필수 항목

있음

정책

설명

*인라인 세션에서 사용할 **JSON** 형식의 **IAM(Identity and Access Management Policy)**입니다. 이 매개 변수의 값은 길이가 **1**에서 **2048**자의 값입니다.*

유형

문자열

필수 항목

없음

DurationSeconds

설명

*세션 기간(초)입니다. 최소 값 **900** 초에서 최대 **43200** 초로 설정됩니다. 기본값은 **3600** 초입니다.*

유형

정수

필수 항목

없음

ExternalId

설명

*다른 계정에 대한 역할을 가정할 때 사용할 수 있는 경우 고유한 외부 식별자를 제공하십시오. 이 매개 변수의 값은 **2**에서 **1224**자의 길이입니다.*

유형

문자열

필수 항목

없음

serialNumber

설명

연결된 다중 요소 인증(MFA) 장치에서 사용자의 식별 번호입니다. 매개 변수의 값은 하드웨어 장치 또는 가상 장치의 일련 번호일 수 있으며 길이가 9~256자일 수 있습니다.

유형

문자열

필수 항목

없음

TokenCode

설명

신뢰 정책에 MFA가 필요한 경우 multi-factor authentication (MFA) 장치에서 생성된 값입니다. MFA 장치가 필요하고 이 매개 변수의 값이 비어 있거나 만료된 경우 AssumeRole 호출에서 "access denied" 오류 메시지를 반환합니다. 이 매개변수의 값은 고정 길이는 6자입니다.

유형

문자열

필수 항목

없음

AssumeRoleWithWebIdentity

이 API는 OpenID Connect 또는 OAuth 2.0 ID 공급자와 같은 애플리케이션에서 인증한 사용자의 임시 인증 정보를 반환합니다. RoleArn 및 RoleSessionName 요청 매개변수는 필수이지만 다른 요청 매개변수는 선택 사항입니다.

RoleArn

설명

Amazon Resource Name(ARN)에 길이가 20~ 2048자인 것으로 가정할 역할입니다.

유형

문자열

필수 항목

있음

RoleSessionName

설명

*위임할 역할 세션 이름을 식별합니다. 역할 세션 이름은 역할을 가정할 때 다른 주체 또는 다른 이유로 세션을 고유하게 식별할 수 있습니다. 이 매개변수의 값은 **2~64**자 길이입니다. **=, ,, ., @, and - characters**는 허용되지만 공백은 허용되지 않습니다.*

유형

문자열

필수 항목

있음

정책

설명

*인라인 세션에서 사용할 **JSON** 형식의 **IAM(Identity and Access Management Policy)**입니다. 이 매개 변수의 값은 길이가 **1**에서 **2048**자의 값입니다.*

유형

문자열

필수 항목

없음

DurationSeconds

설명

*세션 기간(초)입니다. 최소 값 **900** 초에서 최대 **43200** 초로 설정됩니다. 기본값은 **3600** 초입니다.*

유형

정수

필수 항목

없음

providerID

설명

***ID** 공급자의 도메인 이름의 정규화된 호스트 구성 요소입니다. 이 매개변수의 값은 **OAuth 2.0** 액세스 토큰에만 유효하며 길이가 **4~2048**자인 것입니다.*

유형

문자열

필수 항목

없음

WebIdentityToken

설명

***ID** 공급자에서 제공하는 **OpenID Connect ID** 토큰 또는 **OAuth 2.0** 액세스 토큰입니다. 이 매개변수의 값은 길이가 **4~2048**자입니다.*

유형

문자열

필수 항목

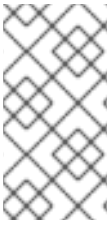
없음

추가 리소스

- *자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드 의 보안 토큰 서비스 **API** 섹션을 참조하십시오.*
- ***Amazon Web Services** 보안 토큰 서비스, **AssumeRole** 작업입니다.*
- ***Amazon Web Services** 보안 토큰 서비스, **AssumeRoleWithWebIdentity** 작업.*

2.3.11.2. 보안 토큰 서비스 구성

Ceph Ansible을 사용하여 Ceph Object Gateway와 함께 사용하도록 STS(Secure Token Service)를 구성합니다.



참고

S3 및 STS API는 동일한 네임스페이스에 공존하고 모두 Ceph Object Gateway의 동일한 끝점에서 액세스할 수 있습니다.

사전 요구 사항

- *Ceph Ansible 관리 노드.*
- *실행 중인 Red Hat Ceph Storage 클러스터.*
- *실행 중인 Ceph 오브젝트 게이트웨이.*

절차

1. *group_vars/rgws.yml 파일을 편집하도록 를 엽니다.*

   a. *다음 줄을 추가합니다.*

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

   교체:

   - *세션 토큰을 암호화하는 데 사용되는 키가 있는 STS_KEY.*

2. *group_vars/rgws.yml 파일에 변경 사항을 저장합니다.*
3. *적절한 Ceph Ansible 플레이북을 다시 실행합니다.*

a. *베어 메탈 배포:*

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

b. *컨테이너 배포:*

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

추가 리소스

- ***STS API**에 대한 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드 의 보안 토큰 서비스 애플리케이션 프로그래밍 인터페이스 섹션을 참조하십시오.*

2.3.11.3. OpenID Connect 공급자의 사용자 생성

***Ceph Object Gateway**와 **OpenID Connect** 공급자 간에 신뢰를 구축하기 위해 사용자 엔티티와 역할 신뢰 정책을 생성합니다.*

사전 요구 사항

- ***Ceph Object Gateway** 노드에 대한 사용자 수준 액세스.*

절차

1. *새 **Ceph** 사용자를 생성합니다.*

구문

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

예제

-

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

2. ***Ceph** 사용자 기능을 구성합니다.*

구문

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

예제

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

3. ***STS(Secure Token Service) API**를 사용하여 역할 신뢰 정책에 조건을 추가합니다.*

구문

```
"{\"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\"\}\}\}\]\}"
```

중요

위의 구문 예제의 **app_id** 는 들어오는 토큰의 **AUD_FIELD** 필드와 일치해야 합니다.

추가 리소스

- **Amazon** 웹 사이트 의 **OpenID Connect ID** 공급자 문서를 보려면 루트 **CA Thumbprint** 를 참조하십시오.
- **STS API**에 대한 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드 의 보안 토큰 서비스 애플리케이션 프로그래밍 인터페이스 섹션을 참조하십시오.
- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드 의 보안 토큰 서비스 **API** 섹션을 참조하십시오.

2.3.11.4. OpenID Connect 공급자의 지문 가져오기

OpenID Connect 공급자(**IDP**) 구성 문서를 가져오려면 다음을 수행합니다.

사전 요구 사항

- **openssl** 및 **curl** 패키지를 설치합니다.

절차

1. **IDP**의 **URL**에서 구성 문서를 가져옵니다.

구문

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

예제

```
[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** 인증서를 가져옵니다.*

구문

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

예제

```
[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. *이전 명령에서 **"x5c"** 응답 결과를 복사하여 **certificate.crt** 파일에 붙여넣습니다. 시작 부분에 **--BEGIN CERTIFICATE--** 를 포함하고 끝에 **--END CERTIFICATE--** 를 포함합니다.*

4. 인증서 지문을 가져옵니다.

구문

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

예제

```
[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
```

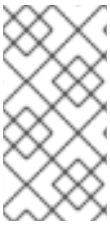
5. **SHA1** 지문에서 모든 콜론을 제거하고 **IAM** 요청에서 **IDP** 엔터티를 생성하기 위한 입력으로 사용합니다.

추가 리소스

- **Amazon** 웹 사이트 의 **OpenID Connect ID** 공급자 문서를 보려면 루트 **CA Thumbprint** 를 참조하십시오.
- **STS API**에 대한 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드 의 보안 토큰 서비스 애플리케이션 프로그래밍 인터페이스 섹션을 참조하십시오.
- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드 의 보안 토큰 서비스 **API** 섹션을 참조하십시오.

2.3.11.5. Keystone과 함께 STS Lite를 구성 및 사용 (기술 프리뷰)

Amazon STS(Secure Token Service) 및 **S3 API**는 동일한 네임스페이스에 공존합니다. **STS** 옵션은 **Keystone** 옵션과 함께 구성할 수 있습니다.



참고

***S3** 및 **STS API**는 모두 **Ceph Object Gateway**에서 동일한 끝점을 사용하여 액세스할 수 있습니다.*

사전 요구 사항

- ***Red Hat Ceph Storage 3.2** 이상.*
- *실행 중인 **Ceph** 오브젝트 게이트웨이.*
- ***Boto Python** 모듈, 버전 **3** 이상 설치.*

절차

1. *다음 옵션을 사용하여 **group_vars/rgws.yml** 파일을 열고 편집합니다.*

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

교체:

- *세션 토큰을 암호화하는 데 사용되는 키가 있는 **STS_KEY.***

2. *적절한 **Ceph Ansible** 플레이북을 다시 실행합니다.*

   a. *베어 메탈 배포:*

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

   b. *컨테이너 배포:*

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

3. ***EC2** 자격 증명을 생성합니다.*

예제

```
[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. *생성된 자격 증명을 사용하여 **GetSessionToken API**를 사용하여 임시 보안 자격 증명 집합을 가져옵니다.*

예제

```
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. *임시 인증 정보를 얻는 데는 **S3**를 호출하는 데 사용할 수 있습니다.*

예제

```
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. *새 **S3Access** 역할을 생성하고 정책을 구성합니다.*

   a. *관리 **CAPS**가 있는 사용자를 할당합니다.*

   구문

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

   예제

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

b. ***S3Access** 역할을 생성합니다.*

구문

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

예제

```
[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** 역할에 권한 정책을 연결합니다.*

구문

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

예제

```
[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. *이제 다른 사용자가 **gwadmin** 사용자의 역할을 가정할 수 있습니다. 예를 들어 **gwuser** 사용자는 **gwadmin** 사용자의 권한을 가정할 수 있습니다.*

e. *가정 사용자의 **access_key** 및 **secret_key** 값을 기록해 둡니다.*

예제

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

7. ***AssumeRole API** 호출을 사용하여 **assuming** 사용자의 **access_key** 및 **secret_key** 값을 제공합니다.*

예제

```
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
)
```

중요

AssumeRole API에는 S3Access 역할이 필요합니다.

추가 리소스

- **Boto Python 모듈 설치에 대한 자세한 내용은 Red Hat Ceph Storage Object Gateway 가이드 의 Test S3 Access 섹션을 참조하십시오.**

- 자세한 내용은 **Red Hat Ceph Storage Object Gateway** 가이드 의 **Create a User** 섹션을 참조하십시오.

2.3.11.6. Keystone에서 STS Lite를 사용하는 제한 사항 (기술 프리뷰)

Keystone의 제한 사항은 STS 요청을 지원하지 않는다는 것입니다. 페이로드 해시는 요청에 포함되지 않습니다. 이러한 두 가지 제한 사항을 해결하려면 Boto 인증 코드를 수정해야 합니다.

사전 요구 사항

- 실행 중인 **Red Hat Ceph Storage** 클러스터, 버전 **3.2** 이상.

- 실행 중인 **Ceph** 오브젝트 게이트웨이.

- **Boto Python** 모듈, 버전 **3** 이상 설치.

절차

1. **Boto의 auth.py** 파일을 열고 편집합니다.

   a. 코드 블록에 다음 네 줄을 추가합니다.

```
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.

다음 두 줄을 코드 블록에 추가합니다.

```
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)
```

추가 리소스

- Boto Python 모듈 설치에 대한 자세한 내용은 Red Hat Ceph Storage Object Gateway 가이드 의 Test S3 Access 섹션을 참조하십시오.

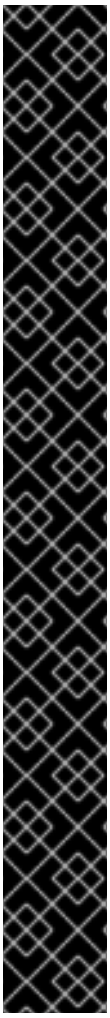
2.3.12. STS의 속성 기반 액세스 제어(ABAC)에 대한 세션 태그

세션 태그는 사용자를 연결하는 동안 전달될 수 있는 키-값 쌍입니다. 보안 토큰 서비스(STS)에서 반환되는 세션 또는 임시 인증 정보에서 aws:PrincipalTag 로 전달됩니다. 이러한 기본 태그는 역할에 가정되는 웹 토큰 및 태그의 일부로 제공되는 세션 태그로 구성됩니다.

참고

현재 세션 태그는 **AssumeRoleWithWebIdentity** 로 전달되는 웹 토큰의 일부로만 지원됩니다.

태그를 항상 다음 네임스페이스에서 지정해야 합니다. **https://aws.amazon.com/tags.**



중요

페더레이션 사용자에 의해 전달되는 웹 토큰에 세션 태그가 포함된 경우 신뢰 정책에 **sts:TagSession** 권한이 있어야 합니다. 그렇지 않으면 **AssumeRoleWithWebIdentity** 작업이 실패합니다.

sts:TagSession 를 사용한 신뢰 정책의 예:

```
{
      "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"}}
      }]
   }
```

속성

다음은 세션 태그의 속성입니다.

- 세션 태그는 다중 값으로 설정할 수 있습니다.

참고

다중 값 세션 태그는 **AWS(Amazon Web Service)**에서 지원되지 않습니다.

- *Keycloak은 최대 50개의 세션 태그가 있는 OpenID Connect ID 공급자(IDP)로 설정할 수 있습니다.*

- *허용되는 키의 최대 크기는 128자입니다.*

- *허용되는 값의 최대 크기는 256자입니다.*

- *태그 또는 값은 aws: 로 시작할 수 없습니다.*

추가 리소스

- *보안 토큰 서비스에 대한 자세한 내용은 Red Hat Ceph Storage 개발자 가이드 의 보안 토큰 서비스 섹션을 참조하십시오.*

2.3.12.1. 태그 키

다음은 역할 신뢰 정책 또는 역할 권한 정책에 사용할 수 있는 태그 키입니다.

aws:RequestTag

설명

요청에 전달된 키-값 쌍을 역할의 신뢰 정책의 키-값 쌍과 비교합니다.

AssumeRoleWithWebIdentity 의 경우 역할 신뢰 정책에서 세션 태그를 aws:RequestTag 로 사용할 수 있습니다. 이러한 세션 태그는 웹 토큰에서 Keycloak에서 전달됩니다. 결과적으로 페더레이션 사용자는 역할을 가정할 수 있습니다.

aws:PrincipalTag

설명

주체에 연결된 키-값 쌍을 정책의 키-값 쌍과 비교합니다.

AssumeRoleWithWebIdentity 의 경우 세션 태그는 사용자가 인증되면 임시 자격 증명에 주체 태그로 표시됩니다. 이러한 세션 태그는 웹 토큰에서 Keycloak에서 전달됩니다. 역할 권한 정책에서 aws:PrincipalTag 로 사용할 수 있습니다.

iam:ResourceTag

설명

리소스에 연결된 키-값 쌍을 정책의 키-값 쌍과 비교합니다.

***AssumeRoleWithWebIdentity** 의 경우 역할에 연결된 태그가 신뢰 정책의 역할과 비교되어 사용자가 역할을 가정할 수 있습니다.*

참고

***Ceph Object Gateway**는 이제 역할에 대한 태그 지정, 나열, 태그 해제 작업을 지원하기 위한 **RESTful API**를 지원합니다.*

AWS:TagKeys

설명

요청의 태그를 정책의 태그와 비교합니다.

***AssumeRoleWithWebIdentity** 의 경우 사용자가 역할을 가정하기 전에 역할 신뢰 정책 또는 권한 정책에서 태그 키를 확인하는 데 사용됩니다.*

s3:ResourceTag

설명

*버킷 또는 오브젝트인 **S3** 리소스에 있는 태그를 역할의 권한 정책의 태그와 비교합니다.*

***Ceph Object Gateway**에서 **S3** 작업을 승인하는 데 사용할 수 있습니다. 그러나 이는 **AWS**에서 허용되지 않습니다.*

*오브젝트 또는 버킷에 연결된 태그를 참조하는 데 사용되는 키입니다. 태그는 동일한에 사용 가능한 **RESTful API**를 사용하여 오브젝트 또는 버킷에 연결할 수 있습니다.*

2.3.12.2. S3 리소스 태그

*다음 목록은 특정 작업 승인을 위해 지원되는 **S3** 리소스 태그 유형을 보여줍니다.*

태그 유형: 오브젝트 태그

작업

GetObject,GetObjectTags,DeleteObjectTags,DeleteObject Tags , DeleteObjectTags ,InitMultipart, 'ListMultipart , 'ListMultipart,GetAttrs, GetObjectRetention , GetObjectRetention,GetObjectRetention, GetObjectLegalHold,GetObjectLegalHold

태그 유형: Bucket 태그

작업

, GetBucket Tags,GetBucketTags,DeleteBucketTags , GetBucketTags ,GetBucketReplication,GetBucket Versioning,SetBucketVersioning,GetBucketWebsite, SetBucketWebsite ,SetBucketWebsite, DeleteBucketWebsite,StatBucket,ListBucket Logging ,GetBucketLogging,GetBucketLogging ,DeleteBucket Location, DeleteBucket ,GetLC,DeleteLC,GetCORS ,GetRequestPayment, SetRequestPayment FlexVolume Policy,GetBucketPolicy,DeleteBucketPolicy, DeleteBucketPolicyLock,GetBucketObjectLock,GetBucketPolicyStatus, GetBucketPublicAccessBlock,GetBucketPublicAccessBlock,DeleteBucketPublicAccessBlock

태그 유형: 버킷 ACL에 대한 버킷 태그, 오브젝트 ACL의 오브젝트 태그

작업

GetACLs , ACLs

태그 유형: 소스 오브젝트의 오브젝트 태그, 대상 버킷의 버킷 태그

작업

FlexVolumeObject,CopyObject

2.4. S3 버킷 작업

개발자는 Ceph Object Gateway를 통해 Amazon S3 애플리케이션 프로그래밍 인터페이스(API)를 사용하여 버킷 작업을 수행할 수 있습니다.

다음 표에는 함수의 지원 상태와 함께 버킷에 대한 Amazon S3 기능 작업이 나열되어 있습니다.

표 2.2. 버킷 작업

기능	상태	참고
Buckets 나열	지원됨	

기능	상태	참고
Bucket을 생성	지원됨	다양한 카나리아 ACL 세트.
버킷 라이프 사이클	부분적으로 지원됨	**expiration**,**NoncurrentVersionExpiration** 및 **AbortIncompleteMultipartUpload** 지원
Bucket Lifecycle을 입력합니다.	부분적으로 지원됨	**expiration**,**NoncurrentVersionExpiration** 및 **AbortIncompleteMultipartUpload** 지원
Bucket Lifecycle 삭제	지원됨	
Bucket 오브젝트를 가져옵니다.	지원됨	
버킷 위치	지원됨	
Bucket 버전 가져오기	지원됨	
Bucket 버전 입력	지원됨	
Bucket 삭제	지원됨	
Bucket ACL 받기	지원됨	다양한 컨디네딩 ACL 세트
Bucket ACL 설정	지원됨	다양한 컨디네딩 ACL 세트
Bucket cors를 가져옵니다.	지원됨	
Bucket cors	지원됨	
Bucket cors 삭제	지원됨	
Bucket 오브젝트 버전 나열	지원됨	
head Bucket	지원됨	
Bucket Multipart Upload 나열	지원됨	
버킷 정책	부분적으로 지원됨	
Bucket 요청 결제 받기	지원됨	
Bucket 요청 결제를 입력하십시오.	지원됨	

기능	상태	참고
멀티 테넌트 Bucket 작업	지원됨	

2.4.1. 사전 요구 사항

- *실행 중인 **Red Hat Ceph Storage** 클러스터.*
- ***RESTful** 클라이언트.*

2.4.2. S3 버킷 알림 생성

*버킷 수준에서 버킷 알림을 생성합니다. 알림 구성에는 **Red Hat Ceph Storage Object Gateway S3** 이벤트, **ObjectCreated** 및 **ObjectRemoved** 가 있습니다. 이를 게시해야 하며 버킷 알림을 보낼 대상입니다. 버킷 알림은 **S3** 작업입니다.*

***s3:objectCreate** 및 **s3:objectRemove** 이벤트에 대한 버킷 알림을 생성하려면 **PUT**을 사용합니다.*

예제

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

중요

***Red Hat**은 다음과 같은 **ObjectCreate** 이벤트(예: **put,post,multipartUpload, copy**)를 지원합니다. **Red Hat**은 **object_delete** 및 **s3_multi_object_delete** 와 같은 **ObjectRemove** 이벤트도 지원합니다.*

요청 엔티티

NotificationConfiguration

설명

***TopicConfiguration** 엔터티 목록.*

유형

컨테이너

필수 항목

있음

TopicConfiguration

설명

ID, 항목 및 이벤트 엔터티 목록입니다.

유형

컨테이너

필수 항목

있음

id

설명

알림의 이름입니다.

유형

문자열

필수 항목

있음

주제

설명

*주제 **Amazon** 리소스 이름(**ARN**)*

참고

주제를 사전에 작성해야합니다.

유형

문자열

필수 항목

있음

이벤트

설명

지원되는 이벤트 목록입니다. 여러 이벤트 엔티티를 사용할 수 있습니다. 생략하면 모든 이벤트가 처리됩니다.

유형

문자열

필수 항목

없음

filter

설명

***S3Key,S3Metadata** 및 **S3Tags** 엔티티입니다.*

유형

컨테이너

필수 항목

없음

S3Key

설명

*개체 키를 기반으로 필터링하기 위한 **FilterRule** 엔터티 목록입니다. 대부분의 경우 **3**개 엔터티가 목록에 있을 수 있습니다. 예를 들어 이름은 접두사,접미사 또는 **regex** 입니다. 목록의 모든 필터 규칙이 필터가 일치해야 합니다.*

유형

컨테이너

필수 항목

없음

S3Metadata

설명

*개체 메타데이터를 기반으로 필터링하기 위한 **FilterRule** 엔터티 목록입니다. 목록의 모든 필터 규칙이 오브젝트에 정의된 메타데이터와 일치해야 합니다. 그러나 필터에 나열되지 않은 다른 메타데이터 항목이 있는 경우에도 오브젝트가 계속 일치합니다.*

유형

컨테이너

필수 항목

없음

S3Tags

설명

*오브젝트 태그를 기반으로 필터링하기 위한 **FilterRule** 엔터티 목록입니다. 목록의 모든 필터 규칙이 오브젝트에 정의된 태그와 일치해야 합니다. 그러나 필터에 나열되지 않은 다른 태그가 있는 경우에도 오브젝트가 계속 일치합니다.*

유형

컨테이너

필수 항목

없음

S3Key.FilterRule

설명

*이름 및 값 엔터티. **name**은 접두사,접미사 또는 **regex** 입니다. **Value** 에는 키 접두사, 키 접미사 또는 키와 일치하는 정규식이 포함됩니다.*

유형

컨테이너

필수 항목

있음

S3Metadata.FilterRule

설명

*이름 및 값 엔터티. **name**은 메타데이터 속성의 이름입니다(예: **x-amz-meta-xxx**). 값은 이 속성에 필요한 값입니다.*

유형

컨테이너

필수 항목

있음

S3Tags.FilterRule

설명

*이름 및 값 엔터티. **name**은 태그 키이며 값은 태그 값입니다.*

유형

컨테이너

필수 항목

있음

HTTP 응답

400

상태 코드

MalformedXML

설명

XML이 제대로 포맷되지 않았습니다.

400

상태 코드

InvalidArgument

설명

누락된 Id 또는 누락된 주제 ARN 또는 유효하지 않은 이벤트입니다.

404

상태 코드

NoSuchBucket

설명

버킷이 존재하지 않습니다.

404

상태 코드

NoSuchKey

설명

주제가 존재하지 않습니다.

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

2.4.3. S3 버킷 알림 가져오기

특정 알림을 받거나 버킷에 구성된 모든 알림을 나열합니다.

구문

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

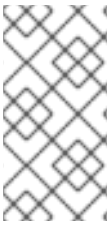
예제

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

응답의 예

```
<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>
```



참고

notification *하위 리소스는 버킷 알림 구성 또는 빈* ***NotificationConfiguration*** *요소를 반환합니다. 호출자는 버킷 소유자여야 합니다.*

요청 엔티티

notification-id

설명

알림의 이름입니다. ***ID****가 제공되지 않은 경우 모든 알림이 나열됩니다.*

유형

문자열

NotificationConfiguration

설명

TopicConfiguration *엔터티 목록.*

유형

컨테이너

필수 항목

있음

TopicConfiguration

설명

ID, 항목 *및 이벤트 엔터티 목록입니다.*

유형

컨테이너

필수 항목

있음

id

설명

알림의 이름입니다.

유형

문자열

필수 항목

있음

주제

설명

*주제 **Amazon** 리소스 이름(**ARN**)*

참고

주제를 사전에 작성해야합니다.

유형

문자열

필수 항목

있음

이벤트

설명

처리 이벤트. 여러 이벤트 엔티티가 존재할 수 있습니다.

유형

문자열

필수 항목

있음

filter

설명

지정된 구성의 필터입니다.

유형

컨테이너

필수 항목

없음

HTTP 응답

404

상태 코드

NoSuchBucket

설명

버킷이 존재하지 않습니다.

404

상태 코드

NoSuchKey

설명

제공된 경우 알림이 존재하지 않습니다.

2.4.4. S3 버킷 알림 삭제

버킷에서 특정 또는 모든 알림을 삭제합니다.

참고

*알림 삭제는 **S3** 알림 **API**의 확장입니다. 버킷에 정의된 알림은 버킷이 삭제될 때 삭제됩니다. 알 수 없는 알림(예: 이중 삭제)을 삭제하 는 것은 오류로 간주되지 않습니다.*

*특정 또는 모든 알림을 삭제하려면 **DELETE**를 사용합니다.*

구문

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

예제

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

요청 엔티티

notification-id

설명

*알림의 이름입니다. 알림 **ID**가 제공되지 않으면 버킷에 대한 모든 알림이 삭제됩니다.*

유형

문자열

***HTTP** 응답*

404

상태 코드

NoSuchBucket

설명

버킷이 존재하지 않습니다.

2.4.5. 버킷 호스트 이름 액세스

*버킷에 액세스하는 두 가지 모드가 있습니다. 첫 번째 및 기본 방법은 버킷을 **URI**에서 최상위 디렉터리로 식별합니다.*

예제

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

두 번째 방법은 가상 버킷 호스트 이름을 통해 버킷을 식별합니다.

예제

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

작은 정보

*두 번째 방법에는 비용이 많이 드는 도메인 인증 및 **DNS** 와일드카드가 필요하기 때문에 **Red Hat**은 첫 번째 방법을 선호합니다.*

2.4.6. S3 목록 버킷

***GET /** 는 사용자가 요청한 버킷 목록을 반환합니다. **GET /** 는 인증된 사용자가 생성한 버킷만 반환합니다. 익명의 요청을 할 수 없습니다.*

구문

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

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

*표 **2.3.** 응답 엔티티*

이름	유형	설명
버킷	컨테이너	버킷 목록의 컨테이너입니다.
bucket	컨테이너	버킷 정보용 컨테이너입니다.
이름	문자열	버킷 이름입니다.
CreationDate	날짜	버킷이 생성된 UTC 시간입니다.
ListAllMyBucketsResult	컨테이너	결과의 컨테이너입니다.
소유자	컨테이너	버킷 소유자 **ID** 및 **DisplayName** 의 컨테이너입니다.
ID	문자열	버킷 소유자의 ID입니다.
DisplayName	문자열	버킷 소유자의 표시 이름입니다.

2.4.7. S3 버킷 오브젝트 목록을 반환합니다.

버킷 개체 목록을 반환합니다.

구문

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

표 2.4. 매개 변수

이름	유형	설명
접두사	문자열	지정된 접두사가 포함된 오브젝트만 반환합니다.
delimiter	문자열	접두사와 나머지 오브젝트 이름 간의 구분 기호입니다.
marker	문자열	반환된 오브젝트 목록의 시작 인덱스입니다.
max-keys	정수	반환할 최대 키 수입니다. 기본값은 1000입니다.

표 2.5. HTTP 응답

HTTP 상태	상태 코드	설명
200	OK	검색된 버킷

***GET /BUCKET** 은 다음 필드가 있는 버킷에 대한 컨테이너를 반환합니다.*

표 2.6. 버킷 응답 엔티티

이름	유형	설명
ListBucketResult	엔터티	오브젝트 목록의 컨테이너입니다.
이름	문자열	콘텐츠가 반환될 버킷의 이름입니다.
접두사	문자열	오브젝트 키의 접두사입니다.
마커	문자열	반환된 오브젝트 목록의 시작 인덱스입니다.
MaxKeys	정수	반환된 최대 키 수입니다.
구분자	문자열	설정한 경우 접두사가 동일한 오브젝트가 **CommonPrefixes** 목록에 표시됩니다.
IsTruncated	부울	**true** 인 경우 버킷 콘텐츠 서브 세트만 반환됩니다.
CommonPrefixes	컨테이너	여러 오브젝트에 동일한 접두사가 포함된 경우 이 목록에 표시됩니다.

ListBucketResult 에는 각 오브젝트가 콘텐츠 컨테이너 내에 있는 오브젝트가 포함되어 있습니다.

표 2.7. 오브젝트 응답 엔티티

이름	유형	설명
내용	개체	오브젝트의 컨테이너입니다.
키	문자열	오브젝트의 키입니다.
LastModified	날짜	오브젝트의 마지막 업데이트 날짜/시간입니다.
etag	문자열	오브젝트의 MD-5 해시. (entity 태그)
크기	정수	오브젝트의 크기입니다.
StorageClass	문자열	항상 **STANDARD** 를 반환해야 합니다.

2.4.8. S3 새 버킷 생성

새 버킷을 생성합니다. 버킷을 생성하려면 요청을 인증하기 위해 사용자 ID와 유효한 AWS 액세스 키 ID가 있어야 합니다. 버킷을 익명 사용자로 생성할 수 없습니다.

Constraints

일반적으로 버킷 이름은 도메인 이름 제약 조건을 따라야 합니다.

- *버킷 이름은 고유해야 합니다.*
- *버킷 이름은 소문자로 시작하고 끝나야 합니다.*
- *버킷 이름에는 대시(-)를 포함할 수 있습니다.*

구문

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

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

표 2.8. 매개 변수

이름	설명	유효한 값	필수 항목
x-amz-acl	ACL을 연결할 수 있습니다.	**private**, **public-read**,**public-read-write**, **authenticated-read**	없음

HTTP 응답

버킷 이름이 제약 조건 내에서 고유하고 사용하지 않는 경우 작업이 성공합니다. 동일한 이름의 버킷이 이미 존재하고 사용자가 버킷 소유자인 경우 작업이 성공합니다. 버킷 이름이 이미 사용 중인 경우 작업이 실패합니다.

HTTP 상태	상태 코드	설명
409	BucketAlreadyExists	버킷은 다른 사용자의 소유권에 이미 존재합니다.

2.4.9. S3 버킷 삭제

버킷을 삭제합니다. 성공적인 버킷 제거 후 버킷 이름을 재사용할 수 있습니다.

구문

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

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

표 2.9. HTTP 응답

HTTP 상태	상태 코드	설명
204	콘텐츠 없음	버킷이 제거되었습니다.

HTTP 상태	상태 코드	설명

2.4.10. S3 버킷 라이프사이클

버킷 라이프사이클 구성을 사용하여 수명 주기 전체에 효율적으로 저장되도록 개체를 관리할 수 있습니다. **Ceph Object Gateway**의 **S3 API**는 **AWS** 버킷 라이프사이클 작업의 하위 집합을 지원합니다.

- **expiration:** 버킷 내에서 오브젝트의 수명을 정의합니다. **Ceph Object Gateway**가 오브젝트를 삭제하는 데 걸리는 시간(주로) 또는 만료일입니다. 버킷에서 버전 관리를 활성화하지 않으면 **Ceph Object Gateway**가 오브젝트를 영구적으로 삭제합니다. 버킷에서 버전 관리를 활성화하면 **Ceph Object Gateway**가 현재 버전에 대한 삭제 마커를 생성한 다음 현재 버전을 삭제합니다.

- **NoncurrentVersionExpiration:** 버킷 내에서 유효하지 않은 오브젝트 버전의 수명을 정의합니다. 이 기능을 사용하려면 버킷에서 버전 관리를 활성화해야 합니다. 현재 개체가 활성 상태가 되는 일 수가 소요되며, 이 시점에서 **Ceph Object Gateway**가 유효하지 않은 오브젝트를 삭제합니다.

- **AbortIncompleteMultipartUpload:** 이는 불완전한 다중 파트 업로드가 중단되기 전에 활성화되어야 하는 일 수를 정의합니다.

라이프사이클 구성에는 **< Rule>** 요소를 사용하는 하나 이상의 규칙이 포함됩니다.

예제

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

*라이프사이클 규칙은 라이프사이클 규칙에서 지정하는 **< Filter >** 요소를 기반으로 버킷의 모든 개체 또는 하위 집합에 적용할 수 있습니다. 다양한 방법으로 필터를 지정할 수 있습니다.*

- *키 접두사*
- *오브젝트 태그*
- *키 접두사 및 하나 이상의 오브젝트 태그 둘 다*

키 접두사

*키 이름 접두사를 기반으로 오브젝트의 하위 집합에 라이프사이클 규칙을 적용할 수 있습니다. 예를 들어 **< keypre/ >**를 지정하면 **keypre/**:로 시작하는 오브젝트에 적용됩니다.*

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

다른 키 접두사를 사용하여 오브젝트에 다른 라이프사이클 규칙을 적용할 수도 있습니다.

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

오브젝트 태그

***< Key >** 및 **< Value >** 요소를 사용하여 특정 태그가 있는 오브젝트에만 라이프사이클 규칙을 적용할 수 있습니다.*

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

접두사 및 하나 이상의 태그 둘 다

라이프사이클 규칙에서는 키 접두사와 하나 이상의 태그를 기반으로 필터를 지정할 수 있습니다. ***<And>****요소에 묶을 필요가 있습니다. 하나의 필터는 접두사가 하나뿐이고* ***0****개 이상의 태그를 포함할 수 있습니다.*

```
<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>
```

추가 리소스

- *버킷 라이프사이클 가져오기에 대한 자세한 내용은* ***Red Hat Ceph Storage*** *개발자 가이드를 참조하십시오.*
- *버킷 라이프사이클 생성에 대한 자세한 내용은* ***Red Hat Ceph Storage*** *개발자 가이드 를 참조하십시오.*
-

버킷 라이프사이클 삭제에 대한 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드 를 참조하십시오.

2.4.11. S3 GET 버킷 라이프사이클

버킷 라이프사이클을 가져오려면 **GET** 을 사용하여 대상 버킷을 지정합니다.

구문

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

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

요청 헤더

자세한 내용은 **Common Request Headers** 를 참조하십시오.

응답

응답에는 버킷 라이프사이클 및 해당 요소가 포함됩니다.

2.4.12. S3 버킷 라이프 사이클을 생성하거나 교체

버킷 라이프사이클을 생성하거나 교체하려면 **PUT** 을 사용하여 대상 버킷 및 라이프사이클 구성을 지정합니다. **Ceph Object Gateway**는 **S3** 라이프사이클 기능의 하위 집합만 지원합니다.

구문

```
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>
```

표 2.10. 요청 헤더

이름	설명	유효한 값	필수 항목
content-md5	base64로 인코딩된 메시지의 MD-5 해시입니다.	문자열입니다. 기본값 또는 제약 조건이 없습니다.	없음

추가 리소스

- *일반적인 Amazon S3 요청 헤더에 대한 자세한 내용은 Red Hat Ceph Storage 개발자 가이드 를 참조하십시오.*
- *Amazon S3 버킷 라이프사이클에 대한 자세한 내용은 Red Hat Ceph Storage 개발자 가이드 를 참조하십시오.*

2.4.13. S3 버킷 라이프사이클 삭제

버킷 라이프사이클을 삭제하려면 DELETE 를 사용하고 대상 버킷을 지정합니다.

구문

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

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

요청 헤더

요청에는 특별한 요소가 포함되어 있지 않습니다.

응답

응답에서 일반적인 응답 상태를 반환합니다.

추가 리소스

- ***Amazon S3** 일반 요청 헤더의 부록 **A** 를 참조하십시오.*
- ***Amazon S3** 일반적인 응답 상태 코드의 부록 **B** 를 참조하십시오.*

2.4.14. S3 버킷 위치 가져오기

*버킷의 영역 그룹을 검색합니다. 이를 호출하려면 사용자가 버킷 소유자여야 합니다. **PUT** 요청 중에 **LocationConstraint** 를 제공하여 버킷을 영역 그룹에 제한할 수 있습니다.*

***location subresource**를 다음과 같이 버킷 리소스에 추가합니다.*

구문

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

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

*표 **2.11.** 응답 엔티티*

이름	유형	설명
LocationConstraint	문자열	버킷이 있는 영역 그룹, 기본 영역 그룹의 빈 문자열

2.4.15. S3 버킷 버전 관리

버킷의 버전 관리 상태를 검색합니다. 이를 호출하려면 사용자가 버킷 소유자여야 합니다.

다음과 같이 **versioning** 하위 리소스를 버킷 리소스에 추가합니다.

구문

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

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

2.4.16. S3 버킷 버전 관리

이 하위 리소스는 기존 버킷의 버전 관리 상태를 설정합니다. 사용자는 버전 관리 상태를 설정하려면 버킷 소유자여야 합니다. 버전 관리 상태가 버킷에 설정되지 않은 경우 버전 관리 상태가 없습니다. **GET** 버전 관리 요청을 수행해도 버전 관리 상태 값을 반환하지 않습니다.

버킷 버전 관리 상태 설정:

Enabled : 버킷에 있는 개체에 대한 버전 관리를 활성화합니다. 버킷에 추가된 모든 오브젝트에는 고유한 버전 **ID**가 부여됩니다. **suspended :** 버킷에 있는 오브젝트에 대한 버전 관리를 비활성화합니다. 버킷에 추가된 모든 오브젝트는 버전 **ID null**을 수신합니다.

구문

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

표 **2.12.** 버킷 요청 엔티티

이름	유형	설명
VersioningConfiguration	컨테이너	요청에 대한 컨테이너입니다.
상태	문자열	버킷의 versioning 상태를 설정합니다. 유효한 값: Suspended/Enabled

2.4.17. S3 get bucket access control lists

*버킷 액세스 제어 목록을 검색합니다. 사용자는 버킷 소유자이거나 버킷에 대한 **READ_ACP** 권한이 부여되어야 합니다.*

*다음과 같이 **acl** 하위 리소스를 버킷 요청에 추가합니다.*

구문

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

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

*표 **2.13.** 응답 엔티티*

이름	유형	설명
AccessControlPolicy	컨테이너	응답을 위한 컨테이너입니다.
AccessControlList	컨테이너	ACL 정보에 대한 컨테이너입니다.
소유자	컨테이너	버킷 소유자 **ID** 및 **DisplayName** 의 컨테이너입니다.
ID	문자열	버킷 소유자의 ID입니다.
DisplayName	문자열	버킷 소유자의 표시 이름입니다.

이름	유형	설명
부여	컨테이너	**부여자** 및 **권한** 부여를 위한 컨테이너.
부여	컨테이너	권한 부여를 수신하는 사용자의 **DisplayName** 및 **ID** 에 대한 컨테이너입니다.
권한	문자열	**Grantee** 버킷에 부여된 권한입니다.

2.4.18. S3, bucket Access Control Lists

*액세스 제어를 기존 버킷에 설정합니다. 버킷 소유자이거나 버킷에 대한 **WRITE_ACP** 권한이 부여되어야 합니다.*

*다음과 같이 **acl** 하위 리소스를 버킷 요청에 추가합니다.*

구문

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

*표 **2.14.** 요청 엔티티*

이름	유형	설명
AccessControlPolicy	컨테이너	요청에 대한 컨테이너입니다.
AccessControlList	컨테이너	ACL 정보에 대한 컨테이너입니다.
소유자	컨테이너	버킷 소유자 **ID** 및 **DisplayName** 의 컨테이너입니다.
ID	문자열	버킷 소유자의 ID입니다.
DisplayName	문자열	버킷 소유자의 표시 이름입니다.

이름	유형	설명
부여	컨테이너	**부여자** 및 **권한** 부여를 위한 컨테이너.
부여	컨테이너	권한 부여를 수신하는 사용자의 **DisplayName** 및 **ID** 에 대한 컨테이너입니다.
권한	문자열	**Grantee** 버킷에 부여된 권한입니다.

2.4.19. S3 버킷 코드 가져오기

*버킷에 설정된 **cors** 구성 정보를 검색합니다. 사용자는 버킷 소유자이거나 버킷에 대한 **READ_ACP** 권한이 부여되어야 합니다.*

*다음과 같이 **cors** 하위 리소스를 버킷 요청에 추가합니다.*

구문

```
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

*버킷의 **cors** 구성을 설정합니다. 사용자는 버킷 소유자이거나 버킷에 대한 **READ_ACP** 권한이 부여되어야 합니다.*

*다음과 같이 **cors** 하위 리소스를 버킷 요청에 추가합니다.*

구문

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

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

2.4.21. S3 bucket cors 삭제

버킷에 설정된 **cors** 구성 정보를 삭제합니다. 사용자는 버킷 소유자이거나 버킷에 대한 **READ_ACP** 권한이 부여되어야 합니다.

다음과 같이 **cors** 하위 리소스를 버킷 요청에 추가합니다.

구문

```
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 오브젝트 버전

버킷 내 모든 개체 버전에 대한 메타데이터 목록을 반환합니다. 버킷에 대한 **READ** 액세스 권한이 필요합니다.

다음과 같이 **versions** 하위 리소스를 버킷 요청에 추가합니다.

구문

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

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

GET /BUCKET?versions 에 대한 매개변수를 지정할 수 있지만 이 매개변수는 필요하지 않습니다.

표 2.15. 매개 변수

이름	유형	설명
접두사	문자열	지정된 접두사가 포함된 키가 있는 진행 중인 업로드를 반환합니다.
delimiter	문자열	접두사와 나머지 오브젝트 이름 간의 구분 기호입니다.
key-marker	문자열	업로드 목록의 시작 마커입니다.
max-keys	정수	진행 중인 최대 업로드 수입니다. 기본값은 1000입니다.
version-id-marker	문자열	목록을 시작할 개체 버전을 지정합니다.

표 2.16. 응답 엔티티

이름	유형	설명
KeyMarker	문자열	**key-marker** 요청 매개변수로 지정된 키 마커(있는 경우)입니다.
NextKeyMarker	문자열	**IsTruncated** 가 **true** 인 경우 후속 요청에 사용할 키 마커입니다.
NextUploadIdMarker	문자열	**IsTruncated** 가 **true** 인 경우 후속 요청에 사용할 업로드 ID 마커입니다.
IsTruncated	부울	**true** 인 경우 버킷의 업로드 콘텐츠 서브 세트만 반환됩니다.
크기	정수	업로드된 부분의 크기입니다.
DisplayName	문자열	소유자의 표시 이름입니다.
ID	문자열	소유자 ID입니다.

이름	유형	설명
소유자	컨테이너	오브젝트를 소유한 사용자의 **ID** 및 **DisplayName** 에 대한 컨테이너입니다.
StorageClass	문자열	결과 오브젝트를 저장하는 데 사용되는 방법입니다. **STANDARD** 또는 **REDUCED_REDUNDANCY**
버전	컨테이너	버전 정보의 컨테이너입니다.
versionId	문자열	오브젝트의 버전 ID입니다.
versionIdMarker	문자열	잘린 응답의 마지막 키 버전입니다.

2.4.23. S3 헤드 버킷

*버킷에서 **HEAD**를 호출하여 해당 버킷이 존재하는지와 호출자에 액세스 권한이 있는지 확인합니다. 버킷이 있고 호출자에 사용 권한이 있으면 **200 OK** 를 반환합니다. 버킷이 없는 경우 **404 Not Found**; 버킷이 존재하지만 호출자에 액세스 권한이 없는 경우 **403 Forbidden** 입니다.*

구문

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

2.4.24. S3 목록 다중 파트 업로드

***GET /?uploads** 는 현재 진행 중인 다중 파트 업로드 목록을 반환합니다. 즉, 애플리케이션이 멀티 파트 업로드를 시작하지만 서비스는 아직 모든 업로드를 완료하지 않았습니다.*

구문

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

GET /BUCKET?uploads 에 대한 매개변수를 지정할 수 있지만 해당 매개변수는 필요하지 않습니다.

표 2.17. 매개 변수

이름	유형	설명
접두사	문자열	지정된 접두사가 포함된 키가 있는 진행 중인 업로드를 반환합니다.
delimiter	문자열	접두사와 나머지 오브젝트 이름 간의 구분 기호입니다.
key-marker	문자열	업로드 목록의 시작 마커입니다.
max-keys	정수	진행 중인 최대 업로드 수입니다. 기본값은 1000입니다.
max-uploads	정수	최대 멀티 파트 업로드 수입니다. 1-1000의 범위입니다. 기본값은 1000입니다.
version-id-marker	문자열	**key-marker** 가 지정되지 않은 경우 무시됩니다.**ID** 의 앞부분에 따라 사전순으로 나열할 첫 번째 업로드의 **ID** 를 지정합니다.

표 2.18. 응답 엔티티

이름	유형	설명
ListMultipartUploadsResult	컨테이너	결과에 대한 컨테이너입니다.
ListMultipartUploadsResult.Prefix	문자열	접두사 요청 매개변수로 지정된 **접두사** 입니다(있는 경우).
bucket	문자열	버킷 콘텐츠를 수신할 버킷입니다.
KeyMarker	문자열	**key-marker** 요청 매개변수로 지정된 키 마커(있는 경우)입니다.
UploadIdMarker	문자열	**upload-id-marker** 요청 매개 변수로 지정된 마커입니다(있는 경우).
NextKeyMarker	문자열	**IsTruncated** 가 **true** 인 경우 후속 요청에 사용할 키 마커입니다.

이름	유형	설명
NextUploadIdMarker	문자열	**IsTruncated** 가 **true** 인 경우 후속 요청에 사용할 업로드 ID 마커입니다.
MaxUploads	정수	**max-uploads** 요청 매개변수로 지정된 최대 업로드입니다.
구분자	문자열	설정한 경우 접두사가 동일한 오브젝트가 **CommonPrefixes** 목록에 표시됩니다.
IsTruncated	부울	**true** 인 경우 버킷의 업로드 콘텐츠 서브 세트만 반환됩니다.
업로드	컨테이너	**키**,**UploadId**,**InitiatorOwner**,**StorageClass** 및 **Initiated** 요소의 컨테이너입니다.
키	문자열	다중 부분 업로드가 완료되면 오브젝트의 키입니다.
UploadId	문자열	다중 부분 업로드를 식별하는 **ID** 입니다.
이니시에이터	컨테이너	업로드를 시작한 사용자의 **ID** 및 **DisplayName** 을 포함합니다.
DisplayName	문자열	이니시에이터의 표시 이름입니다.
ID	문자열	이니시에이터의 ID입니다.
소유자	컨테이너	업로드된 오브젝트를 소유한 사용자의 **ID** 및 **DisplayName** 에 대한 컨테이너입니다.
StorageClass	문자열	결과 오브젝트를 저장하는 데 사용되는 방법입니다. **STANDARD** 또는 **REDUCED_REDUNDANCY**
개시됨	날짜	사용자가 업로드를 시작한 날짜와 시간입니다.
CommonPrefixes	컨테이너	여러 오브젝트에 동일한 접두사가 포함된 경우 이 목록에 표시됩니다.
CommonPrefixes.Prefix	문자열	접두사 요청 매개 변수에 정의된 접두사 뒤에 있는 키의 하위 문자열입니다.

2.4.25. S3 버킷 정책

Ceph Object Gateway는 버킷에 적용된 **Amazon S3** 정책 언어의 하위 집합을 지원합니다.

생성 및 제거

Ceph Object Gateway는 **radosgw-admin CLI** 툴을 사용하지 않고 표준 **S3** 작업을 통해 **S3 Bucket** 정책을 관리합니다.

관리자는 **s3cmd** 명령을 사용하여 정책을 설정하거나 삭제할 수 있습니다.

예제

```
$ 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
```

제한 사항

Ceph Object Gateway는 다음 **S3** 작업만 지원합니다.

- **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*

참고

Ceph Object Gateway는 사용자, 그룹 또는 역할에 대한 정책 설정을 지원하지 않습니다.

Ceph Object Gateway는 Amazon 12자리 계정 ID 대신 RGW '테넌트' 식별자를 사용합니다. AWS(Amazon Web Service) S3와 Ceph Object Gateway S3 간에 정책을 사용하려는 Ceph Object Gateway 관리자는 사용자를 생성할 때 Amazon 계정 ID를 테넌트 ID로 사용해야 합니다.

AWS S3에서는 모든 테넌트가 단일 네임스페이스를 공유합니다. 대조적으로 Ceph Object Gateway는 모든 테넌트에 버킷의 자체 네임스페이스를 제공합니다. 현재 다른 테넌트에 속하는 버킷에 액세스하려고 하는 Ceph Object Gateway 클라이언트는 S3 요청에서 tenant:bucket 으로 주소를 지정해야 합니다.

AWS에서 버킷 정책은 다른 계정에 대한 액세스 권한을 부여할 수 있으며 해당 계정 소유자가 사용자 권한을 가진 개별 사용자에게 액세스 권한을 부여할 수 있습니다. Ceph Object Gateway는 아직 사용자, 역할 및 그룹 권한을 지원하지 않으므로 계정 소유자가 개별 사용자에게 직접 액세스 권한을 부여해야 합니다.

중요

버킷에 대한 전체 계정 액세스 권한을 부여하면 해당 계정의 모든 사용자에게 액세스 권한이 부여됩니다.

버킷 정책은 문자열 보간을 지원하지 않습니다.

***Ceph Object Gateway**는 다음과 같은 조건 키를 지원합니다.*

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

***Ceph Object Gateway ON**은 **ListBucket** 작업에 대해 다음 조건 키를만 지원합니다.*

- ***s3:prefix***

- ***s3:delimiter***

- ***s3:max-keys***

***Swift*에 미치는 영향**

***Ceph Object Gateway*는 *Swift API*에서 버킷 정책을 설정하는 기능을 제공하지 않습니다. 그러나 *S3 API govern Swift* 및 *S3* 작업으로 설정된 버킷 정책은 다음과 같습니다.**

***Ceph Object Gateway*는 정책에 지정된 보안 주체와 비교하여 *Swift* 자격 증명과 일치합니다.**

2.4.26. S3 버킷에 대한 요청 결제 구성을 가져옵니다.

***requestPayment* 하위 리소스를 사용하여 버킷의 요청 결제 구성을 반환합니다. 사용자는 버킷 소유자이거나 버킷에 대한 *READ_ACP* 권한이 부여되어야 합니다.**

다음과 같이 ***requestPayment* 하위 리소스를 버킷 요청에 추가합니다.**

구문

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

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

2.4.27. S3 버킷에 요청 결제 구성을 설정

***requestPayment* 하위 리소스를 사용하여 버킷의 요청 결제 구성을 설정합니다. 기본적으로 버킷 소유자는 버킷에서 다운로드에 대해 비용을 지불합니다. 이 구성 매개변수를 사용하면 버킷 소유자가 다운로드를 요청하는 사람이 버킷에서 요청 및 데이터 다운로드를 위해 부과되도록 지정할 수 있습니다.**

다음과 같이 ***requestPayment* 하위 리소스를 버킷 요청에 추가합니다.**

구문

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

*표 **2.19.** 요청 엔티티*

이름	유형	설명
payer	enum	다운로드 및 요청 요금을 누가 지불하고 요금을 요청하도록 지정합니다.
RequestPaymentConfiguration	컨테이너	**Payer** 컨테이너입니다.

2.4.28. 멀티 테넌트 버킷 작업

*클라이언트 애플리케이션이 버킷에 액세스하면 항상 특정 사용자의 자격 증명으로 작동합니다. **Red Hat Ceph Storage** 클러스터에서는 모든 사용자가 테넌트에 속합니다. 결과적으로 모든 버킷 작업에는 테넌트가 명시적으로 지정되지 않은 경우 컨텍스트에 암시적 테넌트가 있습니다. 따라서 참조된 버킷과 사용자가 동일한 테넌트에 속하는 경우 멀티 테넌시는 이전 릴리스와 완전히 이전 버전과 호환됩니다.*

명시적 테넌트를 지정하는 데 사용되는 확장 기능은 사용된 프로토콜 및 인증 시스템에 따라 다릅니다.

*다음 예에서 콜론 문자는 테넌트와 버킷을 구분합니다. 따라서 샘플 **URL**은 다음과 같습니다.*

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

*반면 간단한 **Python** 예제에서는 버킷 방법 자체에서 테넌트와 버킷을 분리합니다.*

예제

```
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")
```

참고

호스트 이름에 콜론이나 버킷 이름에 유효하지 않은 다른 구분 기호가 포함될 수 없으므로 멀티 테넌시를 사용하는 **S3** 스타일 하위 도메인을 사용할 수 없습니다. 기간을 사용하면 모호한 구문이 생성됩니다. 따라서 **bucket-in-URL-path** 형식을 멀티 테넌시와 함께 사용해야 합니다.

추가 리소스

- 자세한 내용은 멀티 텐트를 참조하십시오.

2.4.29. 추가 리소스

- 버킷 웹 사이트 설정에 대한 자세한 내용은 **Red Hat Ceph Storage Object Gateway 구성 및 관리 가이드** 를 참조하십시오.

2.5. S3 오브젝트 작업

개발자는 **Ceph Object Gateway**를 통해 **Amazon S3** 애플리케이션 프로그래밍 인터페이스**(API)**를 사용하여 오브젝트 작업을 수행할 수 있습니다.

다음 표에는 함수의 지원 상태와 함께 오브젝트에 대한 **Amazon S3** 기능 작업이 나열되어 있습니다.

표 **2.20.** 오브젝트 작업

오브젝트 가져오기	지원됨	
오브젝트 정보 가져오기	지원됨	
오브젝트 배치	지원됨	
오브젝트 삭제	지원됨	
여러 오브젝트 삭제	지원됨	

오브젝트 가져오기	지원됨	
오브젝트 ACL 가져오기	지원됨	
오브젝트 ACL 추가	지원됨	
오브젝트 복사	지원됨	
오브젝트 후	지원됨	
옵션 오브젝트	지원됨	
Multipart Upload 시작	지원됨	
Multipart Upload에 부분 추가	지원됨	
Multipart 업로드 부분 나열	지원됨	
assemble Multipart Upload	지원됨	
Multipart Upload 복사	지원됨	
Multipart Upload 중단	지원됨	
Multi-Tenancy	지원됨	

2.5.1. 사전 요구 사항

- *실행 중인 **Red Hat Ceph Storage** 클러스터.*
- ***RESTful** 클라이언트.*

2.5.2. S3 버킷에서 오브젝트를 가져옵니다.

버킷에서 오브젝트를 검색합니다.

구문

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

***versionId** 하위 리소스를 추가하여 특정 버전의 오브젝트를 검색합니다.*

구문

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

표 2.21. 요청 헤더

이름	설명	유효한 값	필수 항목
범위	검색할 오브젝트의 범위입니다.	범위: bytes=beginbyte-endbyte	없음
if-modified-since	타임스탬프 이후 변경된 경우에만 가져옵니다.Gets only if modified since the timestamp.	Timestamp	없음
if-unmodified-since	타임스탬프 이후 수정되지 않은 경우에만 가져옵니다.	Timestamp	없음
if-match	ETag 개체가 ETag와 일치하는 경우에만 가져옵니다.	엔터티 태그	없음
if-none-match	ETag 개체가 ETag와 일치하는 경우에만 가져옵니다.	엔터티 태그	없음

표 2.22. 응답 헤더

이름	설명
content-Range	데이터 범위는 요청에 range 헤더 필드가 지정된 경우에만 반환됩니다.
x-amz-version-id	버전 ID 또는 null을 반환합니다.

2.5.3. S3 개체에 대한 정보 가져오기

*개체에 대한 정보를 반환합니다. 이 요청은 **Get Object** 요청과 동일한 헤더 정보를 반환하지만 개체 데이터 페이로드가 아닌 메타데이터만 포함합니다.*

현재 버전의 오브젝트를 검색합니다.

구문

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

***versionId** 하위 리소스를 추가하여 특정 버전에 대한 정보를 검색합니다.*

구문

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

*표 **2.23.** 요청 헤더*

이름	설명	유효한 값	필수 항목
범위	검색할 오브젝트의 범위입니다.	범위: bytes=beginbyte-endbyte	없음
if-modified-since	타임스탬프 이후 변경된 경우에만 가져옵니다.Gets only if modified since the timestamp.	Timestamp	없음
if-unmodified-since	타임스탬프 이후 수정되지 않은 경우에만 가져옵니다.	Timestamp	없음
if-match	ETag 개체가 ETag와 일치하는 경우에만 가져옵니다.	엔터티 태그	없음

이름	설명	유효한 값	필수 항목
if-none-match	ETag 개체가 ETag와 일치하는 경우에만 가져옵니다.	엔터티 태그	없음

표 2.24. 응답 헤더

이름	설명
x-amz-version-id	버전 ID 또는 null을 반환합니다.

2.5.4. S3 버킷에 오브젝트 추가

버킷에 오브젝트를 추가합니다. 이 작업을 수행하려면 버킷에 대한 쓰기 권한이 있어야 합니다.

구문

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

표 2.25. 요청 헤더

이름	설명	유효한 값	필수 항목
content-md5	base64로 인코딩된 메시지의 MD-5 해시입니다.	문자열입니다. 기본값 또는 제약 조건이 없습니다.	없음
content-type	표준 MIME 유형입니다.	모든 MIME 유형. 기본값: **binary/octet-stream**	없음
x-amz-meta-<...>	사용자 메타데이터. 오브젝트로 저장됩니다.	8kb 까지의 문자열입니다. 기본값 없음.	없음
x-amz-acl	신뢰할 수 있는 ACL.	**private**, **public-read**, **public-read-write**, **authenticated-read**	없음

표 2.26. 응답 헤더

이름	설명
x-amz-version-id	버전 ID 또는 null을 반환합니다.

2.5.5. S3 개체 삭제

*개체를 제거합니다. 포함된 버킷에 **WRITE** 권한이 설정되어 있어야 합니다.*

개체를 삭제합니다. 오브젝트 버전 관리가 설정되어 있으면 마커를 생성합니다.

구문

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

*버전 관리 시 오브젝트를 삭제하려면 **versionId** 하위 리소스 및 삭제할 오브젝트의 버전을 지정해야 합니다.*

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

2.5.6. S3 여러 오브젝트 삭제

*이 **API** 호출은 버킷에서 여러 오브젝트를 삭제합니다.*

구문

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

2.5.7. S3에서 개체의 ACL(액세스 제어 목록)을 가져옵니다.

*현재 오브젝트 버전에 대한 **ACL**을 반환합니다.*

구문

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

***versionId** 하위 리소스를 추가하여 특정 버전에 대한 **ACL**을 검색합니다.*

구문

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

*표 **2.27.** 응답 헤더*

이름	설명
x-amz-version-id	버전 ID 또는 null을 반환합니다.

*표 **2.28.** 응답 엔티티*

이름	유형	설명
AccessControlPolicy	컨테이너	응답을 위한 컨테이너입니다.
AccessControlList	컨테이너	ACL 정보에 대한 컨테이너입니다.
소유자	컨테이너	오브젝트 소유자 **ID** 및 **DisplayName** 의 컨테이너입니다.
ID	문자열	오브젝트 소유자의 ID입니다.

이름	유형	설명
DisplayName	문자열	오브젝트 소유자의 표시 이름입니다.
부여	컨테이너	**부여자** 및 **권한** 부여를 위한 컨테이너.
부여	컨테이너	권한 부여를 수신하는 사용자의 **DisplayName** 및 **ID** 에 대한 컨테이너입니다.
권한	문자열	**Grantee** 오브젝트에 지정된 권한입니다.

2.5.8. S3에서 개체의 ACL(액세스 제어 목록) 설정

현재 오브젝트 버전에 대한 오브젝트 ACL을 설정합니다.

구문

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

표 2.29. 요청 엔티티

이름	유형	설명
AccessControlPolicy	컨테이너	응답을 위한 컨테이너입니다.
AccessControlList	컨테이너	ACL 정보에 대한 컨테이너입니다.
소유자	컨테이너	오브젝트 소유자 **ID** 및 **DisplayName** 의 컨테이너입니다.
ID	문자열	오브젝트 소유자의 ID입니다.
DisplayName	문자열	오브젝트 소유자의 표시 이름입니다.
부여	컨테이너	**부여자** 및 **권한** 부여를 위한 컨테이너.

이름	유형	설명
부여	컨테이너	권한 부여를 수신하는 사용자의 **DisplayName** 및 **ID** 에 대한 컨테이너입니다.
권한	문자열	**Grantee** 오브젝트에 지정된 권한입니다.

2.5.9. S3 개체 복사

*개체를 복사하려면 **PUT** 을 사용하고 대상 버킷과 개체 이름을 지정합니다.*

구문

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

*표 **2.30.** 요청 헤더*

이름	설명	유효한 값	필수 항목
x-amz-copy-source	소스 버킷 이름 + 오브젝트 이름입니다.	*BUCKET/OBJECT*	있음
x-amz-acl	신뢰할 수 있는 ACL.	**private**, **public-read**, **public-read-write**, **authenticated-read**	없음
x-amz-copy-if-modified-since	타임스탬프 이후 변경된 경우에만 복사합니다.	Timestamp	없음
x-amz-copy-if-unmodified-since	타임스탬프 이후 수정되지 않은 경우에만 복사합니다.	Timestamp	없음
x-amz-copy-if-match	오브젝트 ETag가 ETag와 일치하는 경우에만 복사합니다.	엔터티 태그	없음

이름	설명	유효한 값	필수 항목
x-amz-copy-if-none-match	ETag 개체가 일치하지 않는 경우에만 복사합니다.	엔터티 태그	없음

표 2.31. 응답 엔티티

이름	유형	설명
CopyObjectResult	컨테이너	응답 요소에 대한 컨테이너입니다.
LastModified	날짜	소스 오브젝트의 마지막 수정 날짜입니다.
etag	문자열	새 오브젝트의 ETag입니다.

추가 리소스

- *<추가 리소스 1>*
- *<additional resource 2>*

2.5.10. S3 HTML 양식을 사용하여 버킷에 오브젝트 추가

HTML 양식을 사용하여 버킷에 오브젝트를 추가합니다. 이 작업을 수행하려면 버킷에 대한 쓰기 권한이 있어야 합니다.

구문

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

2.5.11. S3 요청 옵션 확인

실제 요청이 특정 원본, HTTP 메서드 및 헤더를 사용하여 보낼 수 있는지 여부를 결정하는 사전 요청입니다.

구문

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

2.5.12. S3에서 멀티 파트 업로드 시작

*다중 파트 업로드 프로세스를 시작합니다. 추가 부품을 추가할 때 지정할 수 있는 **UploadId** 를 반환하고, 부품을 나열하며, 멀티 파트 업로드를 완료 또는 포기할 수 있습니다.*

구문

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

표 2.32. 요청 헤더

이름	설명	유효한 값	필수 항목
content-md5	base64로 인코딩된 메시지의 MD-5 해시입니다.	문자열입니다. 기본값 또는 제약 조건이 없습니다.	없음
content-type	표준 MIME 유형입니다.	모든 MIME 유형. 기본값: **binary/octet-stream**	없음
x-amz-meta-<...>	사용자 메타데이터. 오브젝트로 저장됩니다.	8kb 까지의 문자열입니다. 기본값 없음.	없음
x-amz-acl	신뢰할 수 있는 ACL.	**private**, **public-read**, **public-read-write**, **authenticated-read**	없음

표 2.33. 응답 엔티티

이름	유형	설명
InitiatedMultipartUploadsResult	컨테이너	결과에 대한 컨테이너입니다.

이름	유형	설명
bucket	문자열	오브젝트 콘텐츠를 수신할 버킷입니다.
키	문자열	키 요청 매개변수로 지정된 **키** (있는 경우)입니다.
UploadId	문자열	다중 부분 업로드를 식별하는 **upload-id** 요청 매개변수로 지정된 ID입니다(있는 경우).

2.5.13. S3 멀티 파트 업로드에 일부를 추가

다중 부분 업로드에 부분을 추가합니다.

*여러 부분 업로드에 부분을 추가하려면 **uploadId** 하위 리소스 및 업로드 **ID**를 지정합니다.*

구문

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

*다음 **HTTP** 응답이 반환될 수 있습니다.*

*표 **2.34. HTTP** 응답*

HTTP 상태	상태 코드	설명
404	NoSuchUpload	지정된 upload-id가 이 오브젝트의 초기 업로드와 일치하지 않음

2.5.14. S3 다중 부분 업로드 목록

*다중 파트 업로드의 일부를 나열하려면 **uploadId** 하위 리소스 및 업로드 **ID**를 지정합니다.*

구문

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

표 2.35. 응답 엔티티

이름	유형	설명
InitiatedMultipartUploadsResult	컨테이너	결과에 대한 컨테이너입니다.
bucket	문자열	오브젝트 콘텐츠를 수신할 버킷입니다.
키	문자열	키 요청 매개변수로 지정된 **키** (있는 경우)입니다.
UploadId	문자열	다중 부분 업로드를 식별하는 **upload-id** 요청 매개변수로 지정된 ID입니다(있는 경우).
이니시에이터	컨테이너	업로드를 시작한 사용자의 **ID** 및 **DisplayName** 을 포함합니다.
ID	문자열	이니시에이터의 ID입니다.
DisplayName	문자열	이니시에이터의 표시 이름입니다.
소유자	컨테이너	업로드된 오브젝트를 소유한 사용자의 **ID** 및 **DisplayName** 에 대한 컨테이너입니다.
StorageClass	문자열	결과 오브젝트를 저장하는 데 사용되는 방법입니다. **STANDARD** 또는 **REDUCED_REDUNDANCY**
PartNumberMarker	문자열	**IsTruncated** 가 **true** 인 경우 후속 요청에 사용할 부분 마커입니다. 목록을 전제로 했습니다.
NextPartNumberMarker	문자열	**IsTruncated** 가 **true** 인 경우 후속 요청에 사용할 다음 부분 마커입니다. 목록의 끝입니다.
MaxParts	정수	**max-parts** 요청 매개 변수에 지정된 대로 응답에 허용되는 최대 부분입니다.
IsTruncated	부울	**true** 인 경우 오브젝트 업로드 콘텐츠의 하위 집합만 반환됩니다.

이름	유형	설명
part	컨테이너	**키**,**Part**,**InitiatorOwner**,**StorageClass** 및 **Initiated** 요소의 컨테이너입니다.
PartNumber	정수	부분의 식별 번호.
etag	문자열	파트의 엔터티 태그입니다.
크기	정수	업로드된 부분의 크기입니다.

2.5.15. S3 업로드된 부분 조립

업로드된 부품을 어셈블하고 새 오브젝트를 생성하여 다중 부분 업로드를 완료합니다.

다중 파트 업로드를 완료하려면 uploadId 하위 리소스 및 업로드 ID를 지정합니다.

구문

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

표 2.36. 요청 엔티티

이름	유형	설명	필수 항목
CompleteMultipartUpload	컨테이너	하나 이상의 부분으로 구성된 컨테이너입니다.	있음
part	컨테이너	**PartNumber** 및 **ETag** 에 대한 컨테이너입니다.	있음
PartNumber	정수	부분의 식별자입니다.	있음
etag	문자열	파트의 엔터티 태그입니다.	있음

표 2.37. 응답 엔티티

이름	유형	설명
CompleteMultipartUploadResult	컨테이너	응답을 위한 컨테이너입니다.
위치	URI	새 오브젝트의 리소스 식별자(path)입니다.
bucket	문자열	새 오브젝트가 포함된 버킷의 이름입니다.
키	문자열	오브젝트의 키입니다.
etag	문자열	새 오브젝트의 entity 태그입니다.

2.5.16. S3에서 다중 파트 업로드 복사

기존 오브젝트에서 데이터 소스로 데이터를 복사하여 일부를 업로드합니다.

*다중 부분 업로드 사본을 수행하려면 **uploadId** 하위 리소스 및 업로드 **ID**를 지정합니다.*

구문

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

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

*표 **2.38.** 요청 헤더*

이름	설명	유효한 값	필수 항목
x-amz-copy-source	소스 버킷 이름 및 오브젝트 이름입니다.	*BUCKET/OBJECT*	있음
x-amz-copy-source-range	소스 오브젝트에서 복사할 바이트 범위입니다.	range: **bytes=first-last** - 첫 번째와 마지막은 복사할 제로 기반 바이트 오프셋입니다. 예를 들어, **bytes=0-9** 는 소스의 처음 10 바이트를 복사하려는 것을 나타냅니다.	없음

*표 **2.39.** 응답 엔티티*

이름	유형	설명
CopyPartResult	컨테이너	모든 응답 요소에 대한 컨테이너입니다.
etag	문자열	새 부분의 ETag를 반환합니다.
LastModified	문자열	파트가 마지막으로 수정된 날짜를 반환합니다.

.Additional Resources

- *이 기능에 대한 자세한 내용은 Amazon S3 사이트를 참조하십시오.*

2.5.17. S3에서 다중 파트 업로드를 중단

다중 파트 업로드를 중지합니다.

다중 파트 업로드를 중단하려면 uploadId 하위 리소스 및 업로드 ID를 지정합니다.

구문

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

2.5.18. S3 Hadoop 상호 운용성

HDFS(Haop Distributed File System) 액세스가 필요한 데이터 분석 애플리케이션의 경우 Hadoop용 Apache S3A 커넥터를 사용하여 Ceph Object Gateway에 액세스할 수 있습니다. S3A 커넥터는 데이터가 Ceph Object Gateway에 저장되는 동안 HDFS 파일 시스템의 HDFS 파일 시스템으로 애플리케이션에 대한 읽기 및 쓰기 의미 체계로 S3 호환 개체 스토리지를 제공하는 오픈 소스 도구입니다.

Ceph Object Gateway는 Hadoop 2.7.3과 함께 제공되는 S3A 커넥터와 완벽하게 호환됩니다.

2.5.19. 추가 리소스

- 멀티 테넌시에 대한 자세한 내용은 **Red Hat Ceph Storage Object Gateway** 구성 및 관리 가이드 를 참조하십시오.

2.6. 추가 리소스

- **Amazon S3** 일반 요청 헤더의 부록 **A** 를 참조하십시오.
- **Amazon S3** 일반적인 응답 상태 코드의 부록 **B** 를 참조하십시오.
- 지원되지 않는 헤더 필드는 부록 **C** 를 참조하십시오.

3장. CEPH 오브젝트 게이트웨이 및 SWIFT API

*개발자는 **Swift API** 데이터 액세스 모델과 호환되는 **RESTful API**(애플리케이션 프로그래밍 인터페이스)를 사용할 수 있습니다. **Ceph Object Gateway**를 통해 **Red Hat Ceph Storage** 클러스터에 저장된 버킷과 개체를 관리할 수 있습니다.*

*다음 표에서는 현재 **Swift** 기능 기능의 지원 상태를 설명합니다.*

*표 **3.1.** 기능*

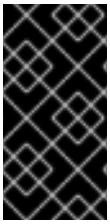
기능	상태	remarks
인증	지원됨	
계정 메타데이터 가져오기	지원됨	사용자 정의 메타데이터 없음
Swift ACL	지원됨	Swift ACL의 하위 집합 지원
컨테이너 나열	지원됨	
컨테이너의 오브젝트 나열	지원됨	
컨테이너 생성	지원됨	
컨테이너 삭제	지원됨	
컨테이너 메타데이터 가져오기	지원됨	
컨테이너 메타데이터 추가/업데이트	지원됨	
컨테이너 메타데이터 삭제	지원됨	
오브젝트 가져오기	지원됨	
오브젝트 생성/업데이트	지원됨	
대규모 오브젝트 생성	지원됨	
오브젝트 삭제	지원됨	
오브젝트 복사	지원됨	
오브젝트 메타데이터 가져오기	지원됨	
오브젝트 메타데이터 추가/업데이트	지원됨	

기능	상태	remarks
임시 URL 작업	지원됨	
CORS	지원되지 않음	
오브젝트 만료	지원됨	
오브젝트 버전 관리	지원되지 않음	
고정 웹 사이트	지원되지 않음	

3.1. 사전 요구 사항

- *실행 중인 **Red Hat Ceph Storage** 클러스터.*
- ***RESTful** 클라이언트.*

3.2. SWIFT API 제한



중요

*다음 제한 사항을 주의해서 사용해야 합니다. 하드웨어 선택과 관련된 영향이 있으므로 **Red Hat** 계정 팀과 항상 이러한 요구 사항을 논의해야 합니다.*

- ***Swift API**를 사용할 때 최대 오브젝트 크기: **5GB***
- ***Swift API**를 사용할 때 최대 메타데이터 크기: 오브젝트에 적용할 수 있는 전체 사용자 메타데이터 크기에 정의된 제한이 없지만 단일 **HTTP** 요청은 **16,000**바이트로 제한됩니다.*

3.3. SWIFT 사용자 만들기

*Swift 인터페이스를 테스트하려면 **Swift** 하위 사용자를 만듭니다. **Swift** 사용자를 만드는 것은 두 개의 단계 프로세스입니다. 첫 번째 단계는 사용자를 생성하는 것입니다. 두 번째 단계는 비밀 키를 생성하는 것입니다.*

참고

다중 사이트 배포에서 **master** 영역 그룹의 마스터 영역에 항상 호스트에 사용자를 생성합니다.

사전 요구 사항

- **Ceph** 개체 게이트웨이 설치.
- **Ceph Object Gateway** 노드에 대한 루트 수준 액세스.

절차

1. **Swift** 사용자를 만듭니다.

구문

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

NAME 을 **Swift** 사용자 이름으로 바꿉니다. 예를 들면 다음과 같습니다.

예제

```
[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. *시크릿 키를 생성합니다.*

구문

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

***NAME* 을 *Swift* 사용자 이름으로 바꿉니다. 예를 들면 다음과 같습니다.**

예제

```
[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. SWIFT에서 사용자 인증

사용자를 인증하려면 헤더에 **X-Auth-User** 및 **X-Auth-Key** 가 포함된 요청을 만듭니다.

구문

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

응답의 예

```
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
```

참고

인증 중에 **X-Storage-Url** 값을 사용하여 **GET** 요청을 실행하여 **Ceph**와 호환되는 서비스에 대한 데이터를 검색할 수 있습니다.

추가 리소스

- Swift 요청 헤더는 Red Hat Ceph Storage Developer Guide 를 참조하십시오.

- Swift 응답 헤더는 Red Hat Ceph Storage Developer Guide 를 참조하십시오.

3.5. SWIFT 컨테이너 작업

개발자는 Ceph Object Gateway를 통해 Swift API(애플리케이션 프로그래밍 인터페이스)를 사용하여 컨테이너 작업을 수행할 수 있습니다. 컨테이너를 나열, 생성, 업데이트 및 삭제할 수 있습니다. 컨테이너의 메타데이터를 추가하거나 업데이트할 수도 있습니다.

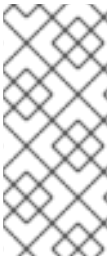
3.5.1. 사전 요구 사항

- 실행 중인 Red Hat Ceph Storage 클러스터.

- RESTful 클라이언트.

3.5.2. Swift 컨테이너 작업

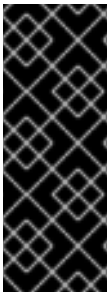
컨테이너는 데이터 오브젝트를 저장하는 메커니즘입니다. 계정에는 여러 컨테이너가 있을 수 있지만 컨테이너 이름은 고유해야 합니다. 이 API를 사용하면 클라이언트는 컨테이너를 생성하고, 액세스 제어 및 메타데이터를 설정하고, 컨테이너의 콘텐츠를 검색하고, 컨테이너를 삭제할 수 있습니다. 이 API는 특정 사용자 계정의 정보와 관련된 요청을 하므로 컨테이너의 액세스 제어가 의도적으로 공개적으로 액세스할 수 있는 경우를 제외하고 이 API의 모든 요청을 인증해야 합니다. 즉 익명의 요청을 허용합니다.



참고

Amazon S3 API는 'bucket'이라는 용어를 사용하여 데이터 컨테이너를 설명합니다. 다른 사람이 Swift API에서 'bucket'을 참조하는 것을 들으면 'bucket'이라는 용어는 'container.' 용어와 동등한 것으로 해석될 수 있습니다.

오브젝트 스토리지의 한 가지 기능은 계층적 경로 또는 디렉터리를 지원하지 않는다는 것입니다. 대신 각 컨테이너에 오브젝트가 포함될 수 있는 하나 이상의 컨테이너로 구성된 하나의 수준을 지원합니다. RADOS 게이트웨이의 Swift 호환 API는 'pseudo-hierarchical containers'의 개념을 지원합니다. 이는 오브젝트 이름 지정을 사용하여 컨테이너 또는 디렉터리 계층 구조를 실제로 스토리지 시스템에서 구현하지 않고 구현하는 수단입니다. pseudo-hierarchical 이름으로 오브젝트의 이름을 지정할 수 있습니다(예: photo/buildings/empire-state.jpg)이지만 컨테이너 이름에는 슬래시(/) 문자를 포함할 수 없습니다.



중요

대용량 오브젝트를 버전 **Swift** 컨테이너에 업로드할 때 **python-swiftclient** 유틸리티와 함께 **--leave-segments** 옵션을 사용합니다. **--leave-segments** 를 사용하지 않는 경우 매니페스트 파일을 덮어씁니다. 따라서 기존 개체를 덮어쓰면 데이터 손실로 이어집니다.**Consequently, an existing object is overwritten, which leads to data loss.**

3.5.3. Swift에서 컨테이너의 ACL(액세스 제어 목록)을 업데이트합니다.

사용자가 컨테이너를 생성할 때 기본적으로 사용자는 컨테이너에 대한 읽기 및 쓰기 액세스 권한을 갖습니다. 다른 사용자가 컨테이너의 콘텐츠를 읽거나 컨테이너에 쓸 수 있도록 하려면 사용자를 특별히 활성화해야 합니다. 모든 사용자가 컨테이너에서 읽거나 쓸 수 있도록 **X-Container-Read** 또는 **X-Container-Write** 설정에서 * 를 지정할 수도 있습니다. 설정 * 는 컨테이너를 공개합니다. 즉, 익명 사용자가 컨테이너에서 읽거나 쓸 수 있습니다.

구문

```
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
```

표 3.2. 요청 헤더

이름	설명	유형	필수 항목
X-Container-Read	컨테이너에 대한 읽기 권한이 있는 사용자 ID입니다.	사용자 ID의 쉼표로 구분된 문자열 값.	없음
X-Container-Write	컨테이너에 대한 쓰기 권한이 있는 사용자 ID입니다.	사용자 ID의 쉼표로 구분된 문자열 값.	없음

3.5.4. Swift 목록 컨테이너

API 버전을 지정하는 **GET** 요청이며 계정은 특정 사용자 계정에 대한 컨테이너 목록을 반환합니다. 요청이 특정 사용자의 컨테이너를 반환하므로 요청에는 인증 토큰이 필요합니다. 이 요청은 익명으로 만들 수 없습니다.

구문

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

표 3.3. 요청 매개변수

이름	설명	유형	필수 항목	유효한 값
limit	결과 수를 지정된 값으로 제한합니다.	정수	없음	해당 없음
형식	결과의 형식을 정의합니다.	문자열	없음	**JSON** 또는 **xml**
marker	마커 값보다 큰 결과 목록을 반환합니다.	문자열	없음	해당 없음

*응답에는 컨테이너 목록이 포함되어 있거나 **HTTP 204** 응답 코드를 사용하여 반환합니다.*

표 3.4. 응답 엔티티

이름	설명	유형
계정	계정 정보 목록입니다.	컨테이너
컨테이너	컨테이너 목록.	컨테이너
name	컨테이너의 이름입니다.	문자열
bytes	컨테이너의 크기입니다.	정수

3.5.5. Swift에서 컨테이너의 오브젝트 나열

*컨테이너 내의 오브젝트를 나열하려면 **API** 버전, 계정 및 컨테이너 이름을 사용하여 **GET** 요청을 만듭*

*니다. 쿼리 매개변수를 지정하여 전체 목록을 필터링하거나 매개 변수를 종료하여 컨테이너에 저장된 처음 **10,000**개 오브젝트 이름 목록을 반환할 수 있습니다.*

구문

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

*표 **3.5.** 매개 변수*

이름	설명	유형	유효한 값	필수 항목
형식	결과의 형식을 정의합니다.	문자열	**JSON** 또는 **xml**	없음
접두사	결과 집합을 지정된 접두사로 시작하는 오브젝트로 제한합니다.	문자열	해당 없음	없음
marker	마커 값보다 큰 결과 목록을 반환합니다.	문자열	해당 없음	없음
limit	결과 수를 지정된 값으로 제한합니다.	정수	0 - 10,000	없음
delimiter	접두사와 나머지 오브젝트 이름 간의 구분 기호입니다.	문자열	해당 없음	없음
경로	오브젝트의 pseudo-hierarchical 경로입니다.	문자열	해당 없음	없음

*표 **3.6.** 응답 엔티티*

이름	설명	유형
컨테이너	컨테이너입니다.	컨테이너
object	컨테이너 내의 오브젝트입니다.	컨테이너
name	컨테이너 내 오브젝트의 이름입니다.	문자열

이름	설명	유형
hash	오브젝트 콘텐츠의 해시 코드입니다.	문자열
last_modified	오브젝트의 콘텐츠가 마지막으로 수정된 시간입니다.	날짜
content_type	오브젝트의 콘텐츠 유형입니다.	문자열

3.5.6. Swift에서 컨테이너 만들기

*새 컨테이너를 생성하려면 **API** 버전, 계정 및 새 컨테이너의 이름을 사용하여 **PUT** 요청을 만듭니다. 컨테이너 이름은 고유해야 하며 **forward-slash(/)** 문자를 포함하지 않아야 하며 **256**바이트 미만이어야 합니다. 요청에 액세스 제어 헤더 및 메타데이터 헤더를 포함할 수 있습니다. 배치 풀 세트의 키를 식별하는 스토리지 정책을 포함할 수도 있습니다. 예를 들어, **execute radosgw-admin zone get to see a list of available keys under placement_pools.** 스토리지 정책을 사용하면 컨테이너에 대한 특수 풀 세트(예: **SSD** 기반 스토리지)를 지정할 수 있습니다. 이 작업은 멱등입니다. 이미 존재하는 컨테이너 생성을 요청하는 경우 **HTTP 202** 반환 코드와 함께 반환되지만 다른 컨테이너를 생성하지 않습니다.*

구문

```
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
```

*표 **3.7. headers***

이름	설명	유형	필수 항목
X-Container-Read	컨테이너에 대한 읽기 권한이 있는 사용자 ID입니다.	사용자 ID의 쉼표로 구분된 문자열 값.	없음

이름	설명	유형	필수 항목
X-Container-Write	컨테이너에 대한 쓰기 권한이 있는 사용자 ID입니다.	사용자 ID의 쉼표로 구분된 문자열 값.	없음
X-Container-Meta-*KEY*	임의의 문자열 값을 사용하는 사용자 정의 메타 데이터 키입니다.	문자열	없음
X-Storage-Policy	Ceph Object Gateway의 **placement_pools** 에서 스토리지 정책을 식별하는 키입니다. 사용 가능한 키에 대해 **radosgw-admin 영역을 가져옵니다**.	문자열	없음

동일한 이름의 컨테이너가 이미 있고 사용자가 컨테이너 소유자인 경우 작업이 성공합니다. 그렇지 않으면 작업이 실패합니다.

표 3.8. HTTP 응답

이름	설명	상태 코드
409	컨테이너는 이미 다른 사용자의 소유권에 있습니다.	**BucketAlreadyExists**

3.5.7. Swift에서 컨테이너 삭제

컨테이너를 삭제하려면 API 버전, 계정 및 컨테이너 이름을 사용하여 DELETE 요청을 만듭니다. 컨테이너가 비어 있어야 합니다. 컨테이너가 비어 있는지 확인하려면 컨테이너에 대해 HEAD 요청을 실행합니다. 컨테이너를 성공적으로 제거하면 컨테이너 이름을 재사용할 수 있습니다.

구문

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

표 3.9. HTTP 응답

이름	설명	상태 코드
204	컨테이너가 제거되었습니다.	**NoContent**

3.5.8. Swift 추가 또는 컨테이너 메타데이터 업데이트

컨테이너에 메타데이터를 추가하려면 API 버전, 계정 및 컨테이너 이름을 사용하여 POST 요청을 만듭니다. 메타데이터를 추가하거나 업데이트하려면 컨테이너에 쓰기 권한이 있어야 합니다.

구문

```
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
```

표 3.10. 요청 헤더

이름	설명	유형	필수 항목
X-Container-Meta-*KEY*	임의의 문자열 값을 사용하는 사용자 정의 메타 데이터 키입니다.	문자열	없음

3.6. SWIFT 오브젝트 작업

개발자는 Ceph Object Gateway를 통해 Swift API(애플리케이션 프로그래밍 인터페이스)를 사용하여 오브젝트 작업을 수행할 수 있습니다. 오브젝트를 나열, 생성, 업데이트 및 삭제할 수 있습니다. 오브젝트의 메타데이터를 추가하거나 업데이트할 수도 있습니다.

3.6.1. 사전 요구 사항

- *실행 중인 **Red Hat Ceph Storage** 클러스터.*
- ***RESTful** 클라이언트.*

3.6.2. Swift 오브젝트 작업

*오브젝트는 데이터 및 메타데이터를 저장하는 컨테이너입니다. 컨테이너에는 많은 오브젝트가 있을 수 있지만 오브젝트 이름은 고유해야 합니다. 이 **API**를 사용하면 클라이언트는 오브젝트를 생성하고, 액세스 제어 및 메타데이터를 설정하고, 오브젝트의 데이터 및 메타데이터를 검색하고, 오브젝트를 삭제할 수 있습니다. 이 **API**는 특정 사용자 계정의 정보와 관련된 요청을 하므로 이 **API**의 모든 요청을 인증해야 합니다. 컨테이너 또는 오브젝트의 액세스 제어가 의도적으로 공개적으로 액세스할 수 있는 한, 즉 익명의 요청을 허용합니다.*

3.6.3. Swift에서 오브젝트를 가져옵니다.

*오브젝트를 검색하려면 **API** 버전, 계정, 컨테이너 및 오브젝트 이름을 사용하여 **GET** 요청을 만듭니다. 컨테이너 내에서 오브젝트를 검색하려면 컨테이너에 대한 읽기 권한이 있어야 합니다.*

구문

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

*표 **3.11.** 요청 헤더*

이름	설명	유형	필수 항목
범위	오브젝트 콘텐츠의 하위 집합을 검색하려면 바이트 범위를 지정할 수 있습니다.	날짜	없음
if-Modified-Since	소스 오브젝트의 **last_modified** 속성의 날짜/시간 이후 변경된 경우에만 복사합니다.	날짜	없음
if-Unmodified-Since	소스 오브젝트의 **last_modified** 속성의 날짜/시간 이후 수정되지 않은 경우에만 복사합니다.	날짜	없음

이름	설명	유형	필수 항목
copy-If-Match	요청에 있는 ETag가 소스 오브젝트의 ETag와 일치하는 경우에만 복사합니다.	etag.	없음
copy-If-None-Match	요청에 있는 ETag가 소스 오브젝트의 ETag와 일치하지 않는 경우에만 복사합니다.	etag.	없음

표 3.12. 응답 헤더

이름	설명
content-Range	오브젝트 콘텐츠의 하위 집합 범위입니다. range 헤더 필드가 요청에 지정된 경우에만 반환됩니다.

3.6.4. Swift create 또는 update an object

*새 오브젝트를 생성하려면 **API** 버전, 계정, 컨테이너 이름 및 새 오브젝트의 이름을 사용하여 **PUT** 요청을 만듭니다. 오브젝트를 생성하거나 업데이트하려면 컨테이너에 쓰기 권한이 있어야 합니다. 오브젝트 이름은 컨테이너 내에서 고유해야 합니다. **PUT** 요청은 멱등이 아니므로 고유한 이름을 사용하지 않으면 요청이 오브젝트를 업데이트합니다. 그러나 오브젝트 이름에 **pseudo-hierarchical** 구문을 사용하여 다른 의사 구조 디렉터리에 있는 경우 동일한 이름의 다른 오브젝트와 구별할 수 있습니다. 요청에 액세스 제어 헤더 및 메타데이터 헤더를 포함할 수 있습니다.*

구문

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

표 3.13. 요청 헤더

이름	설명	유형	필수 항목	유효한 값
etag	오브젝트 콘텐츠의 MD5 해시입니다. 권장 사항.	문자열	없음	해당 없음

이름	설명	유형	필수 항목	유효한 값
content-Type	오브젝트에 포함된 콘텐츠의 유형입니다.	문자열	없음	해당 없음
transfer-authorization	개체가 더 큰 집계 개체의 일부인지 여부를 나타냅니다.Indicates whether the object is part of a larger aggregate object.	문자열	없음	**chunked**

3.6.5. Swift에서 오브젝트 삭제

*오브젝트를 삭제하려면 **API** 버전, 계정, 컨테이너 및 오브젝트 이름을 사용하여 **DELETE** 요청을 만듭니다. 컨테이너 내에서 오브젝트를 삭제하려면 컨테이너에 대한 쓰기 권한이 있어야 합니다. 개체를 성공적으로 삭제하면 개체 이름을 다시 사용할 수 있습니다.*

구문

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

3.6.6. Swift에서 오브젝트 복사

*오브젝트를 복사하면 오브젝트의 서버 측 복사본을 만들 수 있으므로 이를 다운로드하여 다른 컨테이너로 업로드할 필요가 없습니다. 한 오브젝트의 콘텐츠를 다른 오브젝트에 복사하려면 **PUT** 요청 또는 **COPY** 요청을 **API** 버전, 계정, 컨테이너 이름으로 만들 수 있습니다.*

__PUT__ 요청의 경우 요청에 대상 컨테이너 및 오브젝트 이름을 사용하고 요청 헤더에 소스 컨테이너와 오브젝트를 사용합니다.

*복사 요청의 경우 요청에서 소스 컨테이너와 오브젝트를 사용하고 요청 헤더에서 대상 컨테이너와 오브젝트를 사용합니다. 컨테이너에 오브젝트를 복사하려면 쓰기 권한이 있어야 합니다. 대상 오브젝트 이름은 컨테이너 내에서 고유해야 합니다. 요청은 멱등이 아니므로 고유한 이름을 사용하지 않는 경우 요청은 대상 오브젝트를 업데이트합니다. 오브젝트 이름에 **pseudo-hierarchical** 구문을 사용하여 대상 오브젝트를 다른 **pseudo-hierarchical** 디렉터리에 있는 경우 동일한 이름의 소스 오브젝트와 구분할 수 있습니다. 요청에 액세스 제어 헤더 및 메타데이터 헤더를 포함할 수 있습니다.*

구문

```
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
```

또는 다음을 수행합니다.

구문

```
COPY /AP_VERSION/ACCOUNT/TENANT:SOURCE_CONTAINER/SOURCE_OBJECT HTTP/1.1
Destination: TENANT:DEST_CONTAINER/DEST_OBJECT
```

표 3.14. 요청 헤더

이름	설명	유형	필수 항목
X-Copy-From	**PUT** 요청과 함께 사용하여 소스 컨테이너/오브젝트 경로를 정의합니다.	문자열	예, **PUT**를 사용하는 경우
대상	대상 컨테이너/오브젝트 경로를 정의하는 **COPY** 요청과 함께 사용됩니다.	문자열	예, **COPY**를 사용하는 경우
if-Modified-Since	소스 오브젝트의 **last_modified** 속성의 날짜/시간 이후 변경된 경우에만 복사합니다.	날짜	없음
if-Unmodified-Since	소스 오브젝트의 **last_modified** 속성의 날짜/시간 이후 수정되지 않은 경우에만 복사합니다.	날짜	없음

이름	설명	유형	필수 항목
copy-If-Match	요청에 있는 ETag가 소스 오브젝트의 ETag와 일치하는 경우에만 복사합니다.	etag.	없음
copy-If-None-Match	요청에 있는 ETag가 소스 오브젝트의 ETag와 일치하지 않는 경우에만 복사합니다.	etag.	없음

3.6.7. Swift에서 오브젝트 메타데이터 가져오기

*오브젝트의 메타데이터를 검색하려면 **API** 버전, 계정, 컨테이너 및 오브젝트 이름을 사용하여 **HEAD** 요청을 만듭니다. 컨테이너 내의 오브젝트에서 메타데이터를 검색하려면 컨테이너에 대한 읽기 권한이 있어야 합니다. 이 요청은 오브젝트 자체에 대한 요청과 동일한 헤더 정보를 반환하지만 오브젝트 데이터를 반환하지 않습니다.*

구문

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

3.6.8. Swift 추가 또는 오브젝트 메타데이터 업데이트

*오브젝트에 메타데이터를 추가하려면 **API** 버전, 계정, 컨테이너 및 오브젝트 이름을 사용하여 **POST** 요청을 만듭니다. 메타데이터를 추가하거나 업데이트하려면 상위 컨테이너에 쓰기 권한이 있어야 합니다.*

구문

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

*표 **3.15.** 요청 헤더*

이름	설명	유형	필수 항목
X-Object-Meta-*KEY*	임의의 문자열 값을 사용하는 사용자 정의 메타 데이터 키입니다.	문자열	없음

3.7. SWIFT 임시 URL 작업

*임시 액세스를 허용하기 위해 **radosgw** 의 **swift** 엔드포인트에서 임시 **URL** 기능을 지원합니다. 예를 들어 **GET** 요청은 자격 증명을 공유할 필요 없이 오브젝트에 대한 요청입니다.*

*이 기능의 경우 처음에 **X-Account-Meta-Temp-URL-Key** 및 **X-Account-Meta-Temp-URL-Key-2** 값을 설정해야 합니다. **Temp URL** 기능은 이러한 비밀 키에 대해 **HMAC-SHA1** 서명을 사용합니다.*

3.7.1. Swift에서 임시 URL 오브젝트 가져오기

*임시 **URL**은 다음 요소를 포함하는 암호화 **HMAC-SHA1** 서명을 사용합니다.*

- *요청 방법의 값, 인스턴스의 **"GET"***
- ***epoch** 이후의 만료 시간 (초 형식), 즉 **Unix** 시간*
- ***"v1"** 이후의 요청 경로*

*위의 항목은 새 줄 사이에 추가되는 줄로 정규화되고 **HMAC**는 이전에 게시된 **Temp URL** 키 중 하나에 대해 **SHA-1** 해싱 알고리즘을 사용하여 생성됩니다.*

*위를 보여주는 샘플 **python** 스크립트는 다음과 같습니다.*

예제

```
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
```

출력 예

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

3.7.2. Swift POST 임시 URL 키

*필수 키가 있는 **swift** 계정에 대한 **POST** 요청은 계정에 임시 **URL** 액세스를 제공할 수 있는 계정에 대한 시크릿 임시 **URL** 키를 설정합니다. 최대 두 개의 키가 지원되며 서명은 두 키 모두에 대해 확인되므로 임시 **URL**을 무효화하지 않고 키를 순환할 수 있습니다.*

구문

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

*표 **3.16.** 요청 헤더*

이름	설명	유형	필수 항목
X-Account-Meta-Temp-URL-Key	임의의 문자열 값을 사용하는 사용자 정의 키입니다.	문자열	있음
X-Account-Meta-Temp-URL-Key-2	임의의 문자열 값을 사용하는 사용자 정의 키입니다.	문자열	없음

3.8. SWIFT 멀티 테넌시 컨테이너 작업

*클라이언트 애플리케이션이 컨테이너에 액세스할 때 항상 특정 사용자의 인증 정보를 사용하여 작동합니다. **Red Hat Ceph Storage** 클러스터에서는 모든 사용자가 테넌트에 속합니다. 결과적으로 모든 컨테이너 작업에는 테넌트가 명시적으로 지정되지 않은 경우 컨텍스트에 암시적 테넌트가 있습니다. 따라서 참조되는 컨테이너와 참조하는 사용자가 동일한 테넌트에 속하는 경우 멀티 테넌시는 이전 릴리스와 완전히 이전 버전과 호환됩니다.*

명시적 테넌트를 지정하는 데 사용되는 확장 기능은 사용된 프로토콜 및 인증 시스템에 따라 다릅니다.

*콜론 문자는 테넌트와 컨테이너를 구분하므로 샘플 **URL**은 다음과 같습니다.*

예제

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

*반대로 **create_container()** 메서드에서 컨테이너 메서드 자체에서 테넌트와 컨테이너를 분리합니다.*

예제

```
create_container("tenant:container")
```

3.9. 추가 리소스

- 멀티 테넌시에 대한 자세한 내용은 **Red Hat Ceph Storage Object Gateway** 구성 및 관리 가이드 를 참조하십시오.

- **Swift** 요청 헤더에 대한 부록 **D** 를 참조하십시오.

- **Swift** 응답 헤더에 대한 부록 **E** 를 참조하십시오.

부록 A. S3 일반 요청 헤더

다음 표에는 유효한 공통 요청 헤더와 설명이 나열되어 있습니다.

표 A.1. 요청 헤더

요청 헤더	설명
CONTENT_LENGTH	요청 본문의 길이입니다.
DATE	요청 시간 및 날짜(UTC)
호스트	호스트 서버의 이름입니다.
권한 부여	권한 부여 토큰.

부록 B. S3 일반적인 응답 상태 코드

다음 표에는 유효한 공통 HTTP 응답 상태 및 해당 코드가 나열되어 있습니다.

표 B.1. 응답 상태

HTTP 상태	응답 코드
100	계속
200	성공
201	Created
202	승인됨
204	NoContent
206	부분 콘텐츠
304	NotModified
400	InvalidArgument
400	InvalidDigest
400	BadDigest
400	InvalidBucketName
400	InvalidObjectName
400	UnresolvableGrantByEmailAddress
400	InvalidPart
400	InvalidPartOrder
400	RequestTimeout
400	EntityTooLarge
403	AccessDenied
403	UserSuspended

HTTP 상태	응답 코드
403	RequestTimeTooSkewed
404	NoSuchKey
404	NoSuchBucket
404	NoSuchUpload
405	MethodNotAllowed
408	RequestTimeout
409	BucketAlreadyExists
409	BucketNotEmpty
411	MissingContentLength
412	PreconditionFailed
416	InvalidRange
422	UnprocessableEntity
500	InternalError

부록 C. S3 지원되지 않는 헤더 필드

표 C.1. 지원되지 않는 헤더 필드

이름	유형
x-amz-security-token	요청
서버	응답
x-amz-delete-marker	응답
x-amz-id-2	응답
x-amz-request-id	응답
x-amz-version-id	응답

부록 D. SWIFT 요청 헤더

표 D.1. 요청 헤더

이름	설명	유형	필수 항목
X-Auth-User	인증할 주요 Ceph Object Gateway 사용자 이름입니다.	문자열	있음
X-Auth-Key	Ceph Object Gateway 사용자 이름과 연결된 키입니다.	문자열	있음

부록 E. SWIFT 응답 헤더

*서버의 응답에는 **X-Auth-Token** 값이 포함되어야 합니다. 응답에는 **API** 문서 전체에서 다른 요청에 지정된 **API_VERSION/ACCOUNT** 접두사를 제공하는 **X-Storage-Url** 도 포함될 수 있습니다.*

표 E.1. 응답 헤더

이름	설명	유형
X-Storage-Token	요청에 지정된 **X-Auth-User** 에 대한 인증 토큰입니다.	문자열
X-Storage-Url	사용자의 URL 및 ***API_VERSION/ACCOUNT*** 경로입니다.	문자열

부록 F. 보안 토큰 서비스 API 사용 예

이러한 예는 Python의 boto3 모듈을 사용하여 STS(Secure Token Service)의 Ceph Object Gateway 구현과 상호 작용하는 것입니다. 이 예에서 TESTER2 는 역할에 연결된 권한 정책에 따라 TESTER1 이 소유한 S3 리소스에 액세스하는 것처럼 TESTER1 에서 생성한 역할을 가정합니다.

AssumeRole 예제에서는 역할을 만들고 역할을 할당한 다음 임시 자격 증명을 사용하여 S3 리소스에 대한 액세스 및 임시 자격 증명을 가져오는 역할을 가정합니다.

AssumeRoleWithWebIdentity 예제에서는 Keycloak인 외부 애플리케이션을 사용하여 사용자를 인증합니다. OpenID Connect ID 공급자는 역할의 권한 정책에 따라 임시 자격 증명을 가져오고 S3 리소스에 액세스하는 역할을 가정합니다.

AssumeRole 예

```
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 Example

```
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()
```

추가 리소스

- *Python의 boto 모듈을 사용하는 방법에 대한 자세한 내용은 Red Hat Ceph Storage Object Gateway Configuration and Administration Guide 의 Test S3 Access 섹션을 참조하십시오.*

부록 G. STS에서 특성 기반 액세스 제어에 세션 태그를 사용하는 예

다음 목록에는 STS의 특성 기반 액세스 제어(ABAC)에 대한 세션 태그 사용 예가 나와 있습니다.

웹 토큰의 Keycloak에서 전달하는 세션 태그의 예

```
{
    "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:RequestTag의 예

```
{
        "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:PrincipalTag의 예

```
{
    "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:ResourceTag예

```
{
    "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)

위의 작업이 작동하려면 ***'Department=Engineering'*** *태그를 역할에 연결해야 합니다.*

aws:TagKeys의 예

```
{
    "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** 는 요청의 모든 태그 키가 정책의 태그 키 서브 세트인지 여부를 테스트합니다. 따라서 조건은 요청에 전달된 태그 키를 제한합니다.*

***s3:ResourceTag**의 예*

```
{
  "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)
  }
}
```

(1)

*위의 작업이 수행되려면 이 정책을 적용하려는 버킷 또는 오브젝트에 **'Department=Engineering'** 태그를 첨부해야 합니다.*

aws:RequestTag 와 iam:ResourceTag의 예

```
{
        "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)
        }]
    }
```

(1)

이는 들어오는 요청의 태그와 역할에 연결된 태그를 일치시켜 역할을 가정하기 위한 것입니다. ***AWS:RequestTag*** *는* ***JSON*** *웹 토큰(JWT)에서 들어오는 태그이며* ***iam:ResourceTag*** *는 가정되는 역할에 연결된 태그입니다.*

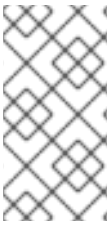
s3:ResourceTag가 있는 aws:PrincipalTag 의 예

```
{
    "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)
    }
}
```

(1)

부록 H. 세션 태그의 사용 예시 샘플 코드

다음은 역할, 버킷 또는 오브젝트를 태그하고 역할 신뢰 및 역할 권한 정책에서 태그 키를 사용하는 샘플 코드입니다.



참고

이 예제에서는 **tag department =Engineering** 태그가 **Keycloak의JWT(JSON Web Token)** 액세스 토큰으로 전달되는 것으로 가정합니다.

```
# -*- 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)
```