



Red Hat Ceph Storage 5

개발자 가이드

Red Hat Ceph Storage에 다양한 애플리케이션 프로그래밍 인터페이스 사용

Red Hat Ceph Storage 5 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

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Developer_Guide.ent file |.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution-Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

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 RESTFUL API	8
1.1. 사전 요구 사항	8
1.2. CEPH API의 버전 관리	8
1.3. CEPH API에 대한 인증 및 권한 부여	9
1.4. CEPH API 모듈 활성화 및 보안	9
1.5. 질문 및 답변	11
1.5.1. 정보 얻기	11
1.5.1.1. 모든 클러스터 구성 옵션을 보려면 어떻게 해야 합니까?	12
curl 명령	12
Python	12
웹 브라우저	12
추가 리소스	13
1.5.1.2. 파र्ट별 클러스터 구성 옵션을 보려면 어떻게 해야 합니까?	13
curl 명령	13
Python	13
웹 브라우저	14
추가 리소스	14
1.5.1.3. OSD에 대한 모든 구성 옵션을 보려면 어떻게 해야 합니까?	14
curl 명령	14
Python	14
웹 브라우저	15
추가 리소스	15
1.5.1.4. CRUSH 규칙을 어떻게 볼 수 있습니까?	15
curl 명령	15
Python	16
웹 브라우저	16
추가 리소스	16
1.5.1.5. 모니터에 대한 정보를 어떻게 볼 수 있습니까?	16
curl 명령	16
Python	17
웹 브라우저	17
1.5.1.6. How can I view information about a Particular Monitor?	17
curl 명령	18
Python	18
웹 브라우저	19
1.5.1.7. OSD에 대한 정보를 어떻게 볼 수 있습니까?	19
curl 명령	19
Python	19
웹 브라우저	20
1.5.1.8. OSD에 대한 정보를 어떻게 볼 수 있습니까?	20
curl 명령	20
Python	21
웹 브라우저	21
1.5.1.9. OSD에서 일정을 설정할 수 있는 프로세스를 어떻게 확인할 수 있습니까?	21
curl 명령	21
Python	22
웹 브라우저	22
1.5.1.10. 풀에 대한 정보를 어떻게 볼 수 있습니까?	23
curl 명령	23
Python	23
웹 브라우저	23

1.5.1.11. Particular Pool에 대한 정보를 어떻게 볼 수 있습니까?	24
curl 명령	24
Python	24
웹 브라우저	25
1.5.1.12. 호스트에 대한 정보를 어떻게 볼 수 있습니까?	25
curl 명령	25
Python	25
웹 브라우저	26
1.5.1.13. Particular Host에 대한 정보를 어떻게 볼 수 있습니까?	26
curl 명령	26
Python	27
웹 브라우저	27
1.5.2. 설정 변경	27
1.5.2.1. OSD 구성 옵션을 변경하려면 어떻게 해야 합니까?	28
curl 명령	28
Python	28
1.5.2.2. OSD 상태를 어떻게 변경할 수 있습니까?	29
curl 명령	29
Python	30
1.5.2.3. OSD를 다시 스케일링하려면 어떻게 해야 합니까?	31
curl 명령	31
Python	32
1.5.2.4. 풀의 정보를 어떻게 변경할 수 있습니까?	33
curl 명령	33
Python	34
1.5.3. 클러스터 관리	35
1.5.3.1. OSD에서 예정된 프로세스를 실행하려면 어떻게 해야 합니까?	35
curl 명령	35
Python	36
1.5.3.2. 새 풀을 어떻게 생성할 수 있습니까?	37
curl 명령	37
Python	37
1.5.3.3. 풀을 어떻게 제거 할 수 있습니까?	38
curl 명령	39
Python	39
1.6. 추가 리소스	40
2장. CEPH OBJECT GATEWAY 관리 API	41
2.1. 사전 요구 사항	44
2.2. 관리 작업	44
2.3. 관리 인증 요청	45
2.4. 관리 사용자 생성	54
2.5. 사용자 정보 가져오기	57
2.6. 사용자 생성	61
2.7. 사용자 수정	70
2.8. 사용자 제거	78
2.9. 하위 사용자 생성	80
2.10. 하위 사용자 수정	85
2.11. 하위 사용자 제거	89
2.12. 사용자에게 기능 추가	91
2.13. 사용자의 기능 제거	94
2.14. 키 생성	97
2.15. 키 제거	102

2.16. 버킷 알람	104
2.16.1. 사전 요구 사항	105
2.16.2. 버킷 알람 개요	105
2.16.3. 영구 알람	105
2.16.4. 항목 생성	105
2.16.5. 주제 정보 얻기	110
2.16.6. 주제 나열	112
2.16.7. 주제 삭제	114
2.16.8. 이벤트 레코드	115
2.16.9. 지원되는 이벤트 유형	117
2.17. 버킷 정보 가져오기	118
2.18. 버킷 인덱스 확인	123
2.19. 버킷 제거	126
2.20. 버킷 연결	127
2.21. 버킷 연결 해제	132
2.22. 버킷 또는 오브젝트 정책 가져오기	133
2.23. 오브젝트 제거	135
2.24. 할당량	137
2.25. 사용자 할당량 가져오기	138
2.26. 사용자 할당량 설정	138
2.27. 버킷 할당량 가져오기	139
2.28. 버킷 할당량 설정	144
2.29. 사용 정보 받기	144
2.30. 사용 정보 제거	150
2.31. 표준 오류 응답	152
3장. CEPH OBJECT GATEWAY 및 S3 API	154
3.1. 사전 요구 사항	154
3.2. S3 제한 사항	154
3.3. S3 API를 사용하여 CEPH 오브젝트 게이트웨이 액세스	155
3.3.1. 사전 요구 사항	155
3.3.2. S3 인증	155
3.3.3. S3 서버 측 암호화	157
3.3.4. S3 액세스 제어 목록	159
3.3.5. S3를 사용하여 Ceph Object Gateway에 대한 액세스 준비	160
3.3.6. Ruby AWS S3를 사용하여 Ceph Object Gateway에 액세스	162
3.3.7. Ruby AWS SDK를 사용하여 Ceph Object Gateway에 액세스	169
3.3.8. PHP를 사용하여 Ceph Object Gateway에 액세스	177
3.3.9. 보안 토큰 서비스	184
3.3.9.1. 보안 토큰 서비스 애플리케이션 프로그래밍 인터페이스	184
3.3.9.2. 보안 토큰 서비스 구성	189
3.3.9.3. OpenID Connect 공급자의 사용자 생성	191
3.3.9.4. OpenID Connect 공급자의 지문 가져오기	193
3.3.9.5. Keystone과 함께 STS Lite를 구성 및 사용 (기술 프리뷰)	195
3.3.9.6. Keystone에서 STS Lite를 사용하는 제한 사항 (기술 프리뷰)	201
3.4. S3 버킷 작업	202
3.4.1. 사전 요구 사항	203
3.4.2. S3 버킷 알람 생성	204
3.4.3. S3 버킷 알람 가져오기	209
3.4.4. S3 버킷 알람 삭제	213
3.4.5. 버킷 호스트 이름 액세스	215
3.4.6. S3 목록 버킷	215
3.4.7. S3 버킷 오브젝트 목록을 반환합니다.	217

3.4.8. S3 새 버킷 생성	222
3.4.9. S3 put bucket website	224
3.4.10. S3 버킷 웹 사이트 가져오기	225
3.4.11. S3 delete bucket website	226
3.4.12. S3 버킷 삭제	226
3.4.13. S3 버킷 라이프사이클	227
3.4.14. S3 GET 버킷 라이프사이클	230
3.4.15. S3 버킷 라이프 사이클을 생성하거나 교체	230
3.4.16. S3 버킷 라이프사이클 삭제	231
3.4.17. S3 버킷 위치 가져오기	232
3.4.18. S3 버킷 버전 관리	233
3.4.19. S3, 버킷 버전 관리	233
3.4.20. S3 get bucket access control lists	235
3.4.21. S3, bucket Access Control Lists	237
3.4.22. S3 버킷 코드 가져오기	239
3.4.23. S3 put bucket cors	239
3.4.24. S3 bucket cors 삭제	240
3.4.25. S3 list bucket 오브젝트 버전	240
3.4.26. S3 헤드 버킷	244
3.4.27. S3 목록 다중 파트 업로드	245
3.4.28. S3 버킷 정책	251
3.4.29. S3 버킷에 대한 요청 결제 구성을 가져옵니다.	256
3.4.30. S3 버킷에 요청 결제 구성을 설정	257
3.4.31. 멀티 테넌트 버킷 작업	257
3.5. S3 오브젝트 작업	258
3.5.1. 사전 요구 사항	260
3.5.2. S3 버킷에서 오브젝트를 가져옵니다.	260
3.5.3. S3 개체에 대한 정보 가져오기	262
3.5.4. S3 put 오브젝트 잠금	264
3.5.5. S3 개체 잠금을 가져옵니다.	268
3.5.6. S3 put object legal hold	271
3.5.7. S3의 개체 법적 유지	272
3.5.8. S3 put 오브젝트 보존	273
3.5.9. S3 가져오기 오브젝트 보존	275
3.5.10. S3 put object tagging	277
3.5.11. S3 개체 태그 가져오기	278
3.5.12. S3 delete 오브젝트 태그	279
3.5.13. S3 버킷에 오브젝트 추가	280
3.5.14. S3 개체 삭제	281
3.5.15. S3 여러 오브젝트 삭제	282
3.5.16. S3에서 개체의 ACL(액세스 제어 목록)을 가져옵니다.	282
3.5.17. S3에서 개체의 ACL(액세스 제어 목록) 설정	285
3.5.18. S3 개체 복사	287
3.5.19. S3 HTML 양식을 사용하여 버킷에 오브젝트 추가	290
3.5.20. S3 요청 옵션 확인	290
3.5.21. S3에서 멀티 파트 업로드 시작	290
3.5.22. S3 멀티 파트 업로드에 일부를 추가	293
3.5.23. S3 다중 부분 업로드 목록	293
3.5.24. S3 업로드된 부분 조립	297
3.5.25. S3에서 다중 파트 업로드 복사	300
3.5.26. S3에서 다중 파트 업로드를 중단	302
3.5.27. S3 Hadoop 상호 운용성	302
3.5.28. 추가 리소스	302

3.6. S3 선택 작업 (기술 프리뷰)	302
3.6.1. 사전 요구 사항	303
3.6.2. S3 오브젝트에서 콘텐츠 선택	303
3.6.3. S3 지원 일부 기능	312
3.6.4. S3 별칭 프로그래밍 구성	315
3.6.5. S3 CSV 구문 분석 설명	316
3.7. 추가 리소스	317
4장. CEPH 오브젝트 게이트웨이 및 SWIFT API	318
4.1. 사전 요구 사항	319
4.2. SWIFT API 제한	319
4.3. SWIFT 사용자 만들기	319
4.4. SWIFT에서 사용자 인증	323
4.5. SWIFT 컨테이너 작업	324
4.5.1. 사전 요구 사항	324
4.5.2. Swift 컨테이너 작업	324
4.5.3. Swift에서 컨테이너의 ACL(액세스 제어 목록)을 업데이트합니다.	325
4.5.4. Swift 목록 컨테이너	326
4.5.5. Swift에서 컨테이너의 오브젝트 나열	328
4.5.6. Swift에서 컨테이너 만들기	332
4.5.7. Swift에서 컨테이너 삭제	334
4.5.8. Swift 추가 또는 컨테이너 메타데이터 업데이트	335
4.6. SWIFT 오브젝트 작업	336
4.6.1. 사전 요구 사항	336
4.6.2. Swift 오브젝트 작업	336
4.6.3. Swift에서 오브젝트를 가져옵니다.	336
4.6.4. Swift create 또는 update an object	338
4.6.5. Swift에서 오브젝트 삭제	340
4.6.6. Swift에서 오브젝트 복사	340
4.6.7. Swift에서 오브젝트 메타데이터 가져오기	343
4.6.8. Swift 추가 또는 오브젝트 메타데이터 업데이트	344
4.7. SWIFT 임시 URL 작업	344
4.7.1. Swift에서 임시 URL 오브젝트 가져오기	345
4.7.2. Swift POST 임시 URL 키	346
4.8. SWIFT 멀티 테넌시 컨테이너 작업	347
4.9. 추가 리소스	348
부록 A. CEPH RESTFUL API 사양	349
A.1. 사전 요구 사항	351
A.2. CEPH 요약	351
A.3. 인증	352
A.4. CEPH 파일 시스템	354
A.5. 스토리지 클러스터 구성	366
A.6. CRUSH 규칙	370
A.7. 코드 프로필 삭제	373
A.8. 기능 토글	376
A.9. GRAFANA	377
A.10. 스토리지 클러스터 상태	379
A.11. 호스트	381
A.12. ISCSI	389
A.13. 로그	395
A.14. CEPH MANAGER 모듈	396
A.15. CEPH MONITOR	400

A.16. CEPH OSD	401
A.17. CEPH 오브젝트 게이트웨이	417
A.18. 역할 조작을 위한 REST API	436
A.19. NFS GANESHA	443
A.20. CEPH ORCHESTRATOR	449
A.21. POOL	450
A.22. PROMETHEUS	455
A.23. RADOS 블록 장치	459
A.24. 성능 카운터	487
A.25. 역할	493
A.26. 서비스	498
A.27. 설정	502
A.28. CEPH 작업	506
A.29. TELEMETRY	507
A.30. CEPH 사용자	509
부록 B. S3 일반 요청 헤더	516
부록 C. S3 일반적인 응답 상태 코드	517
부록 D. S3 지원되지 않는 헤더 필드	519
부록 E. SWIFT 요청 헤더	520
부록 F. SWIFT 응답 헤더	521
부록 G. 보안 토큰 서비스 API 사용 예	522

1장. CEPH RESTFUL API

스토리지 관리자는 Red Hat Ceph Storage 대시 보드에서 제공하는 Ceph RESTful API 또는 간단히 Ceph API를 사용하여 Red Hat Ceph Storage 클러스터와 상호 작용할 수 있습니다. Ceph 모니터 및 OSD에 대한 정보를 해당 구성 옵션과 함께 표시할 수 있습니다. Ceph 풀을 생성하거나 편집할 수도 있습니다.

Ceph API는 다음 표준을 사용합니다.

- HTTP 1.1
- JSON
- MIME 및 HTTP 콘텐츠 협상
- JWT

이러한 표준은 OpenAPI 3.0을 준수하여 API 구문, 의미 체계, 콘텐츠 인코딩, 버전 관리, 인증 및 권한 부여를 조정합니다.

1.1. 사전 요구 사항

- 정상 실행 중인 Red Hat Ceph Storage 클러스터.
- Ceph Manager를 실행하는 노드에 액세스합니다.

1.2. CEPH API의 버전 관리

Ceph RESTful API의 주요 목표는 안정적인 인터페이스를 제공하는 것입니다. 안정적인 인터페이스를 달성하기 위해 Ceph API는 다음 원칙에 따라 빌드됩니다.

- 암시적 기본값을 피하기 위해 모든 엔드 포인트에 대한 필수 명시적인 기본 버전입니다.
- 엔드 포인트당 세부 제어 변경.
 - 특정 끝점에서 예상되는 버전은 HTTP 헤더에 설명되어 있습니다.

구문

```
Accept: application/vnd.ceph.api.vMAJOR.MINOR+json
```

예제

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

현재 Ceph API 서버가 특정 버전을 처리할 수 없는 경우 **415 - 지원되지 않는 미디어 유형** 응답이 반환됩니다.

- 의미 체계 버전 관리 사용.
 - 주요 변경 사항은 이전 버전과 호환되지 않습니다. 변경으로 인해 요청에 대한 비추가적인 변경 사항 및 특정 끝점의 응답 형식이 변경될 수 있습니다.
 - 일부 변경 사항은 이전 버전과 호환되며 변경될 수 있습니다. 변경 사항은 특정 엔드포인트에 대한 요청 또는 응답 형식에 대한 추가 변경 사항으로 구성됩니다.

1.3. CEPH API에 대한 인증 및 권한 부여

Ceph RESTful API에 대한 액세스는 두 개의 검사점을 거칩니다. 첫 번째는 요청이 유효한 사용자 및 기존 사용자를 대신하여 수행됨을 인증하는 것입니다. 두 번째는 이전에 인증된 사용자를 승인하는 것이며 대상 끝점에서 생성, 읽기, 업데이트 또는 삭제와 같은 특정 작업을 수행할 수 있습니다.

사용자가 Ceph API를 사용하기 전에 유효한 JSON 웹 토큰(JWT)이 필요합니다. **/api/auth** 엔드포인트를 사용하면 이 토큰을 검색할 수 있습니다.

예제

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

이 토큰은 **Authorization** HTTP 헤더에 배치하여 모든 API 요청과 함께 사용해야 합니다.

구문

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

추가 리소스

- 자세한 내용은 *Red Hat Ceph Storage 관리 가이드의 Ceph 사용자 관리* 장을 참조하십시오.

1.4. CEPH API 모듈 활성화 및 보안

Red Hat Ceph Storage Dashboard 모듈은 SSL 보안 연결을 통해 스토리지 클러스터에 대한 RESTful API 액세스를 제공합니다.



중요

SSL을 비활성화하면 사용자 이름과 암호는 암호화되지 않은 상태로 Red Hat Ceph Storage 대시보드로 전송됩니다.

사전 요구 사항

- Ceph Monitor 노드에 대한 루트 수준 액세스.
- 하나 이상의 **ceph-mgr** 데몬이 활성화되어 있는지 확인합니다.
- 방화벽을 사용하는 경우, SSL에 대해 TCP 포트 **8443**, 및 SSL 없이 TCP 포트 **8080** 이 활성화 **ceph-mgr** 데몬이 있는 노드에서 열려 있는지 확인합니다.

절차

1. Cephadm 셸에 로그인합니다.

예제

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

2. RESTful 플러그인을 활성화합니다.

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

3. SSL 인증서를 구성합니다.

- a. 조직의 CA(인증 기관)에서 인증서를 제공하는 경우 인증서 파일을 사용하여 설정합니다.

구문

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

예제

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

고유한 노드 기반 인증서를 설정하려면 명령에 *HOST_NAME* 을 추가합니다.

예제

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

- b. 또는 자체 서명된 인증서를 생성할 수 있습니다. 그러나 자체 서명된 인증서를 사용하면 HTTPS 프로토콜의 완전한 보안 이점을 제공하지 않습니다.

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



주의

대부분의 최신 웹 브라우저에서는 보안 연결을 설정하기 전에 확인해야 하는 자체 서명된 인증서에 대해 보고합니다.

4. 사용자를 생성하고 암호를 설정한 다음 역할을 설정합니다.

구문

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

예제

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

이 예제에서는 관리자 역할을 사용하여 **user1** 이라는 사용자를 생성합니다.

5. RESTful 플러그인 웹 페이지에 연결합니다. 웹 브라우저를 열고 다음 URL을 입력합니다.

구문

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

예제

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

자체 서명된 인증서를 사용한 경우 보안 예외를 확인합니다.

추가 리소스

- **ceph dashboard --help** 명령
- **https://HOST_NAME:8443/doc** 페이지는 *HOST_NAME* 은 **ceph-mgr** 인스턴스가 실행 중인 노드의 IP 주소 또는 이름입니다.
- [Red Hat Enterprise Linux 8 Security Hardening](#) 가이드.

1.5. 질문 및 답변

1.5.1. 정보 얻기

이 섹션에서는 Ceph API를 사용하여 스토리지 클러스터, Ceph 모니터, OSD, 풀 및 호스트에 대한 정보를 확인하는 방법을 설명합니다.

- 1.5.1.1절. "모든 클러스터 구성 옵션을 보려면 어떻게 해야 합니까?"
- 1.5.1.2절. "파트별 클러스터 구성 옵션을 보려면 어떻게 해야 합니까?"
- 1.5.1.3절. "OSD에 대한 모든 구성 옵션을 보려면 어떻게 해야 합니까?"
- 1.5.1.4절. "CRUSH 규칙을 어떻게 볼 수 있습니까?"
- 1.5.1.5절. "모니터에 대한 정보를 어떻게 볼 수 있습니까?"
- 1.5.1.6절. "How can I view information about a Particular Monitor?"
- 1.5.1.7절. "OSD에 대한 정보를 어떻게 볼 수 있습니까?"
- 1.5.1.8절. "OSD에 대한 정보를 어떻게 볼 수 있습니까?"
- 1.5.1.9절. "OSD에서 일정을 설정할 수 있는 프로세스를 어떻게 확인할 수 있습니까?"
- 1.5.1.10절. "풀에 대한 정보를 어떻게 볼 수 있습니까?"
- 1.5.1.11절. "Particular Pool에 대한 정보를 어떻게 볼 수 있습니까?"
- 1.5.1.12절. "호스트에 대한 정보를 어떻게 볼 수 있습니까?"
- 1.5.1.13절. "Particular Host에 대한 정보를 어떻게 볼 수 있습니까?"

1.5.1.1. 모든 클러스터 구성 옵션을 보려면 어떻게 해야 하나요?

이 섹션에서는 RESTful 플러그인을 사용하여 클러스터 구성 옵션 및 해당 값을 확인하는 방법을 설명합니다.

curl 명령

명령줄에서 다음을 사용합니다.

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

교체:

- 사용자 이름을 가진 **USER**
- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- **CEPH_MANAGER_PORT** 및 TCP 포트 번호. 기본 TCP 포트 번호는 8443입니다.

메시지가 표시되면 사용자 암호를 입력합니다.

자체 서명된 인증서를 사용한 경우 **--insecure** 옵션을 사용합니다.

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

Python

Python 인터프리터에서 다음을 입력합니다.

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

교체:

- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- 사용자 이름을 가진 **USER**
- 사용자 **암호** 가 있는 암호

자체 서명된 인증서를 사용한 경우 **verify=False** 옵션을 사용합니다.

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

웹 브라우저

웹 브라우저에서 다음을 입력합니다.

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


교체:

- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**

메시지가 표시되면 사용자 이름과 암호를 입력합니다.

추가 리소스

- Red Hat Ceph Storage 5 [구성 가이드](#)

1.5.1.2. 파트별 클러스터 구성 옵션을 보려면 어떻게 해야 하나요?

이 섹션에서는 특정 클러스터 옵션 및 해당 값을 보는 방법을 설명합니다.

curl 명령

명령줄에서 다음을 사용합니다.

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

교체:

- 사용자 이름을 가진 **USER**
- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- 보려는 구성 옵션을 사용한 **ARGUMENT**

메시지가 표시되면 사용자 암호를 입력합니다.

자체 서명된 인증서를 사용한 경우 **--insecure** 옵션을 사용합니다.

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

Python

Python 인터프리터에서 다음을 입력합니다.

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

교체:

- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- 보려는 구성 옵션을 사용한 **ARGUMENT**
- 사용자 이름을 가진 **USER**
- 사용자 **암호**가 있는 암호

자체 서명된 인증서를 사용한 경우 **verify=False** 옵션을 사용합니다.

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

웹 브라우저

웹 브라우저에서 다음을 입력합니다.

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

교체:

- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- 보려는 구성 옵션을 사용한 **ARGUMENT**

메시지가 표시되면 사용자 이름과 암호를 입력합니다.

추가 리소스

- Red Hat Ceph Storage 5 [구성 가이드](#)

1.5.1.3. OSD에 대한 모든 구성 옵션을 보려면 어떻게 해야 합니까?

이 섹션에서는 모든 구성 옵션 및 OSD의 값을 확인하는 방법에 대해 설명합니다.

curl 명령

명령줄에서 다음을 사용합니다.

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

교체:

- 사용자 이름을 가진 **USER**
- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**

메시지가 표시되면 사용자 암호를 입력합니다.

자체 서명된 인증서를 사용한 경우 **--insecure** 옵션을 사용합니다.

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

Python

Python 인터프리터에서 다음을 입력합니다.

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

교체:

- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- 사용자 이름을 가진 **USER**
- 사용자 **암호**가 있는 암호

자체 서명된 인증서를 사용한 경우 **verify=False** 옵션을 사용합니다.

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

웹 브라우저

웹 브라우저에서 다음을 입력합니다.

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

교체:

- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**

메시지가 표시되면 사용자 이름과 암호를 입력합니다.

추가 리소스

- Red Hat Ceph Storage 5 [구성 가이드](#)

1.5.1.4. CRUSH 규칙을 어떻게 볼 수 있습니까?

이 섹션에서는 CRUSH 규칙을 보는 방법에 대해 설명합니다.

curl 명령

명령줄에서 다음을 사용합니다.

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

교체:

- 사용자 이름을 가진 **USER**
- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**

메시지가 표시되면 사용자 암호를 입력합니다.

자체 서명된 인증서를 사용한 경우 **--insecure** 옵션을 사용합니다.

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

Python

Python 인터프리터에서 다음을 입력합니다.

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

교체:

- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- 사용자 이름을 가진 **USER**
- 사용자 **암호** 가 있는 암호

자체 서명된 인증서를 사용한 경우 **verify=False** 옵션을 사용합니다.

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

웹 브라우저

웹 브라우저에서 다음을 입력합니다.

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

교체:

- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**

메시지가 표시되면 사용자 이름과 암호를 입력합니다.

추가 리소스

- Red Hat Ceph Storage 5에 대한 *관리 가이드* 의 NetNamespace [규칙](#) 섹션.

1.5.1.5. 모니터에 대한 정보를 어떻게 볼 수 있습니까?

이 섹션에서는 다음과 같은 특정 모니터에 대한 정보를 보는 방법을 설명합니다.

- IP 주소
- 이름
- 쿼럼 상태

curl 명령

명령줄에서 다음을 사용합니다.

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

교체:

- 사용자 이름을 가진 **USER**
- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**

메시지가 표시되면 사용자 암호를 입력합니다.

자체 서명된 인증서를 사용한 경우 **--insecure** 옵션을 사용합니다.

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

Python

Python 인터프리터에서 다음을 입력합니다.

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

교체:

- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- 사용자 이름을 가진 **USER**
- 사용자 **암호** 가 있는 암호

자체 서명된 인증서를 사용한 경우 **verify=False** 옵션을 사용합니다.

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

웹 브라우저

웹 브라우저에서 다음을 입력합니다.

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

교체:

- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**

메시지가 표시되면 사용자 이름과 암호를 입력합니다.

1.5.1.6. How can I view information about a Particular Monitor?

이 섹션에서는 다음과 같은 특정 모니터에 대한 정보를 보는 방법을 설명합니다.

- IP 주소
- 이름
- 쿼럼 상태

curl 명령

명령줄에서 다음을 사용합니다.

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

교체:

- 사용자 이름을 가진 ***USER***
- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 ***CEPH_MANAGER***
- 모니터의 짧은 호스트 이름으로 ***NAME***

메시지가 표시되면 사용자 암호를 입력합니다.

자체 서명된 인증서를 사용한 경우 **--insecure** 옵션을 사용합니다.

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

Python

Python 인터프리터에서 다음을 입력합니다.

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

교체:

- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 ***CEPH_MANAGER***
- 모니터의 짧은 호스트 이름으로 ***NAME***
- 사용자 이름을 가진 ***USER***
- 사용자 **암호** 가 있는 암호

자체 서명된 인증서를 사용한 경우 **verify=False** 옵션을 사용합니다.

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

웹 브라우저

웹 브라우저에서 다음을 입력합니다.

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

교체:

- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- 모니터의 짧은 호스트 이름으로 **NAME**

메시지가 표시되면 사용자 이름과 암호를 입력합니다.

1.5.1.7. OSD에 대한 정보를 어떻게 볼 수 있습니까?

이 섹션에서는 다음과 같은 OSD에 대한 정보를 확인하는 방법을 설명합니다.

- IP 주소
- 해당 풀
- 유사성
- weight

curl 명령

명령줄에서 다음을 사용합니다.

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

교체:

- 사용자 이름을 가진 **USER**
- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**

메시지가 표시되면 사용자 암호를 입력합니다.

자체 서명된 인증서를 사용한 경우 **--insecure** 옵션을 사용합니다.

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

Python

Python 인터프리터에서 다음을 입력합니다.

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

교체:

- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- 사용자 이름을 가진 **USER**
- 사용자 **암호**가 있는 암호

자체 서명된 인증서를 사용한 경우 **verify=False** 옵션을 사용합니다.

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

웹 브라우저

웹 브라우저에서 다음을 입력합니다.

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

교체:

- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**

메시지가 표시되면 사용자 이름과 암호를 입력합니다.

1.5.1.8. OSD에 대한 정보를 어떻게 볼 수 있습니까?

이 섹션에서는 다음과 같은 특정 OSD에 대한 정보를 확인하는 방법을 설명합니다.

- IP 주소
- 해당 풀
- 유사성
- weight

curl 명령

명령줄에서 다음을 사용합니다.

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

교체:

- 사용자 이름을 가진 **USER**
- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- **osd** 필드에 나열된 OSD의 **ID**가 있는 ID

메시지가 표시되면 사용자 암호를 입력합니다.

자체 서명된 인증서를 사용한 경우 **--insecure** 옵션을 사용합니다.


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

Python

Python 인터프리터에서 다음을 입력합니다.

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

교체:

- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- **osd** 필드에 나열된 OSD의 **ID**가 있는 ID
- 사용자 이름을 가진 **USER**
- 사용자 **암호**가 있는 암호

자체 서명된 인증서를 사용한 경우 **verify=False** 옵션을 사용합니다.

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

웹 브라우저

웹 브라우저에서 다음을 입력합니다.

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

교체:

- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- **osd** 필드에 나열된 OSD의 **ID**가 있는 ID

메시지가 표시되면 사용자 이름과 암호를 입력합니다.

1.5.1.9. OSD에서 일정을 설정할 수 있는 프로세스를 어떻게 확인할 수 있습니까?

이 섹션에서는 RESTful 플러그인을 사용하여 스크립 또는 딥 스크립과 같이 어떤 프로세스를 OSD에 예약할 수 있는지 확인하는 방법을 설명합니다.

curl 명령

명령줄에서 다음을 사용합니다.

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

교체:

- 사용자 이름을 가진 **USER**
- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- **osd** 필드에 나열된 OSD의 **ID**가 있는 ID

메시지가 표시되면 사용자 암호를 입력합니다.

자체 서명된 인증서를 사용한 경우 **--insecure** 옵션을 사용합니다.

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

Python

Python 인터프리터에서 다음을 입력합니다.

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

교체:

- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- **osd** 필드에 나열된 OSD의 **ID**가 있는 ID
- 사용자 이름을 가진 **USER**
- 사용자 **암호**가 있는 암호

자체 서명된 인증서를 사용한 경우 **verify=False** 옵션을 사용합니다.

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

웹 브라우저

웹 브라우저에서 다음을 입력합니다.

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

교체:

- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- **osd** 필드에 나열된 OSD의 **ID**가 있는 ID

메시지가 표시되면 사용자 이름과 암호를 입력합니다.

1.5.1.10. 풀에 대한 정보를 어떻게 볼 수 있습니까?

이 섹션에서는 다음과 같은 풀에 대한 정보를 보는 방법을 설명합니다.

- 플래그
- 크기
- 배치 그룹 수

curl 명령

명령줄에서 다음을 사용합니다.

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

교체:

- 사용자 이름을 가진 **USER**
- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**

메시지가 표시되면 사용자 암호를 입력합니다.

자체 서명된 인증서를 사용한 경우 **--insecure** 옵션을 사용합니다.

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

Python

Python 인터프리터에서 다음을 입력합니다.

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

교체:

- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- 사용자 이름을 가진 **USER**
- 사용자 **암호** 가 있는 암호

자체 서명된 인증서를 사용한 경우 **verify=False** 옵션을 사용합니다.

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

웹 브라우저

웹 브라우저에서 다음을 입력합니다.

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

교체:

- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**

메시지가 표시되면 사용자 이름과 암호를 입력합니다.

1.5.1.11. Particular Pool에 대한 정보를 어떻게 볼 수 있습니까?

이 섹션에서는 다음과 같은 특정 풀에 대한 정보를 보는 방법을 설명합니다.

- 플래그
- 크기
- 배치 그룹 수

curl 명령

명령줄에서 다음을 사용합니다.

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

교체:

- 사용자 이름을 가진 **USER**
- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- **pool** 필드에 나열된 풀 **ID** 가 있는 ID

메시지가 표시되면 사용자 암호를 입력합니다.

자체 서명된 인증서를 사용한 경우 **--insecure** 옵션을 사용합니다.

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

Python

Python 인터프리터에서 다음을 입력합니다.

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

교체:

- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- **pool** 필드에 나열된 풀 **ID** 가 있는 ID
- 사용자 이름을 가진 **USER**

- 사용자 **암호**가 있는 암호

자체 서명된 인증서를 사용한 경우 **verify=False** 옵션을 사용합니다.

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

웹 브라우저

웹 브라우저에서 다음을 입력합니다.

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

교체:

- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- **pool** 필드에 나열된 풀 **ID**가 있는 ID

메시지가 표시되면 사용자 이름과 암호를 입력합니다.

1.5.12. 호스트에 대한 정보를 어떻게 볼 수 있습니까?

이 섹션에서는 다음과 같은 호스트에 대한 정보를 보는 방법을 설명합니다.

- 호스트 이름
- Ceph 데몬 및 ID
- Ceph 버전

curl 명령

명령줄에서 다음을 사용합니다.

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

교체:

- 사용자 이름을 가진 **USER**
- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**

메시지가 표시되면 사용자 암호를 입력합니다.

자체 서명된 인증서를 사용한 경우 **--insecure** 옵션을 사용합니다.

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

Python

Python 인터프리터에서 다음을 입력합니다.

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

교체:

- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- 사용자 이름을 가진 **USER**
- 사용자 **암호** 가 있는 암호

자체 서명된 인증서를 사용한 경우 **verify=False** 옵션을 사용합니다.

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

웹 브라우저

웹 브라우저에서 다음을 입력합니다.

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

교체:

- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**

메시지가 표시되면 사용자 이름과 암호를 입력합니다.

1.5.1.13. Particular Host에 대한 정보를 어떻게 볼 수 있습니까?

이 섹션에서는 다음과 같은 특정 호스트에 대한 정보를 보는 방법을 설명합니다.

- 호스트 이름
- Ceph 데몬 및 ID
- Ceph 버전

curl 명령

명령줄에서 다음을 사용합니다.

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

교체:

- 사용자 이름을 가진 **USER**
- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**

- **hostname** 필드에 나열된 호스트의 호스트 이름이 있는 **HOST_NAME**

메시지가 표시되면 사용자 암호를 입력합니다.

자체 서명된 인증서를 사용한 경우 **--insecure** 옵션을 사용합니다.

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

Python

Python 인터프리터에서 다음을 입력합니다.

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

교체:

- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- **hostname** 필드에 나열된 호스트의 호스트 이름이 있는 **HOST_NAME**
- 사용자 이름을 가진 **USER**
- 사용자 **암호**가 있는 암호

자체 서명된 인증서를 사용한 경우 **verify=False** 옵션을 사용합니다.

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

웹 브라우저

웹 브라우저에서 다음을 입력합니다.

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

교체:

- active **ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- **hostname** 필드에 나열된 호스트의 호스트 이름이 있는 **HOST_NAME**

메시지가 표시되면 사용자 이름과 암호를 입력합니다.

1.5.2. 설정 변경

이 섹션에서는 Ceph API를 사용하여 OSD 구성 옵션, OSD 상태 및 풀 관련 정보를 변경하는 방법을 설명합니다.

- 1.5.2.1절. "OSD 구성 옵션을 변경하려면 어떻게 해야 하나요?"
- 1.5.2.2절. "OSD 상태를 어떻게 변경할 수 있습니까?"
- 1.5.2.3절. "OSD를 다시 스케일링하려면 어떻게 해야 하나요?"
- 1.5.2.4절. "풀의 정보를 어떻게 변경할 수 있습니까?"

1.5.2.1. OSD 구성 옵션을 변경하려면 어떻게 해야 하나요?

이 섹션에서는 RESTful 플러그인을 사용하여 OSD 구성 옵션을 변경하는 방법을 설명합니다.

curl 명령

명령줄에서 다음을 사용합니다.

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

교체:

- modify 옵션이 있는
optionION,noup,nodown,noout,noin,nobackfill,norecover,noscrub,nodeep-scrub
- **true 또는 false를 가진 VALUE**
- **사용자 이름을 가진 USER**
- **active ceph-mgr 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 CEPH_MANAGER**

메시지가 표시되면 사용자 암호를 입력합니다.

자체 서명된 인증서를 사용한 경우 **--insecure** 옵션을 사용합니다.

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

Python

Python 인터프리터에서 다음을 입력합니다.

```
$ python
>> import requests
```



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

교체:

- **active ceph-mgr** 인스턴스가 있는 노드의 **IP** 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- **modify** 옵션이 있는 **optionION,noup,nodown,noout,noin,nobackfill,norecover,noscrub,nodeep-scrub**
- **True** 또는 **False**를 사용하여 **VALUE**
- 사용자 이름을 가진 **USER**
- 사용자 암호가 있는 암호

자체 서명된 인증서를 사용한 경우 **verify=False** 옵션을 사용합니다.

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

1.5.2.2. OSD 상태를 어떻게 변경할 수 있습니까?

이 섹션에서는 **RESTful** 플러그인을 사용하여 **OSD**의 상태를 변경하는 방법을 설명합니다.

curl 명령

명령줄에서 다음을 사용합니다.

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

교체:

- 변경할 상태와 함께 **STATE** (내 또는 **up**)
- **true** 또는 **false**를 가진 **VALUE**
- 사용자 이름을 가진 **USER**
- **active ceph-mgr** 인스턴스가 있는 노드의 **IP** 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- **osd** 필드에 나열된 **OSD**의 **ID**가 있는 **ID**

메시지가 표시되면 사용자 암호를 입력합니다.

자체 서명된 인증서를 사용한 경우 **--insecure** 옵션을 사용합니다.

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

Python

Python 인터프리터에서 다음을 입력합니다.

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

교체:

- **active ceph-mgr** 인스턴스가 있는 노드의 **IP** 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**

- **osd** 필드에 나열된 **OSD**의 **ID** 가 있는 **ID**
- 변경할 상태와 함께 **STATE** (내 또는 up)
- **True** 또는 **False**를 사용하여 **VALUE**
- 사용자 이름을 가진 **USER**
- 사용자 암호 가 있는 암호

자체 서명된 인증서를 사용한 경우 **verify=False** 옵션을 사용합니다.

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

1.5.2.3. OSD를 다시 스케일링하려면 어떻게 해야 할까요?

이 섹션에서는 **OSD**의 가중치를 변경하는 방법을 설명합니다.

curl 명령

명령줄에서 다음을 사용합니다.

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

교체:

- 새로운 가중치와 함께 **VALUE**
- 사용자 이름을 가진 **USER**

- **active ceph-mgr** 인스턴스가 있는 노드의 **IP** 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- **osd** 필드에 나열된 **OSD**의 **ID** 가 있는 **ID**

메시지가 표시되면 사용자 암호를 입력합니다.

자체 서명된 인증서를 사용한 경우 **--insecure** 옵션을 사용합니다.

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

Python

Python 인터프리터에서 다음을 입력합니다.

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

교체:

- **active ceph-mgr** 인스턴스가 있는 노드의 **IP** 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- **osd** 필드에 나열된 **OSD**의 **ID** 가 있는 **ID**
- 새로운 가중치와 함께 **VALUE**
- 사용자 이름을 가진 **USER**
- 사용자 암호 가 있는 암호

자체 서명된 인증서를 사용한 경우 **verify=False** 옵션을 사용합니다.

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

1.5.2.4. 풀의 정보를 어떻게 변경할 수 있습니까?

이 섹션에서는 **RESTful** 플러그인을 사용하여 특정 풀의 정보를 변경하는 방법에 대해 설명합니다.

curl 명령

명령줄에서 다음을 사용합니다.

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

교체:

- 헤더 를 변경할 수 있는 옵션이 있는 경우
- 옵션 의 새 값을 나타내는 값을 가져옵니다. **Gets the value of the new value of the option.**
- 사용자 이름을 가진 **USER**
- **active ceph-mgr** 인스턴스가 있는 노드의 **IP** 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- **pool** 필드에 나열된 풀 **ID** 가 있는 **ID**

메시지가 표시되면 사용자 암호를 입력합니다.

자체 서명된 인증서를 사용한 경우 **--insecure** 옵션을 사용합니다.

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

Python

Python 인터프리터에서 다음을 입력합니다.

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

교체:

- **active ceph-mgr** 인스턴스가 있는 노드의 **IP** 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- **pool** 필드에 나열된 풀 **ID** 가 있는 **ID**
- 헤더를 변경할 수 있는 옵션이 있는 경우
- 옵션의 새 값을 나타내는 값을 가져옵니다. **Gets the value of the new value of the option.**
- 사용자 이름을 가진 **USER**
- 사용자 암호가 있는 암호

자체 서명된 인증서를 사용한 경우 **verify=False** 옵션을 사용합니다.

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

1.5.3. 클러스터 관리

이 섹션에서는 **Ceph API**를 사용하여 **OSD**에서 스크럽 또는 딥 스크럽을 초기화하거나, 풀에서 풀을 생성하거나, 요청을 제거하거나, 요청을 생성하는 방법을 설명합니다.

-

1.5.3.1절. “OSD에서 예정된 프로세스를 실행하려면 어떻게 해야 하나요?”

-

1.5.3.2절. “새 풀을 어떻게 생성할 수 있습니까?”

-

1.5.3.3절. “풀을 어떻게 제거 할 수 있습니까?”

1.5.3.1. OSD에서 예정된 프로세스를 실행하려면 어떻게 해야 하나요?

이 섹션에서는 **RESTful API**를 사용하여 **OSD**에서 스크럽 또는 딥 스크럽과 같이 예약된 프로세스를 실행하는 방법을 설명합니다.

curl 명령

명령줄에서 다음을 사용합니다.

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

교체:

-

시작하려는 프로세스 (**crub**, **deep-scrub** 또는 **repair**)를 사용 합니다. 프로세스가 **OSD**에서 지원되는지 확인합니다. 자세한 내용은 **1.5.1.9절. “OSD에서 일정을 설정할 수 있는 프로세스를 어떻게 확인할 수 있습니까?”** 를 참조하십시오.

-

사용자 이름을 가진 **USER**

-

active ceph-mgr 인스턴스가 있는 노드의 **IP** 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**

- **osd 필드에 나열된 OSD의 ID 가 있는 ID**

메시지가 표시되면 사용자 암호를 입력합니다.

자체 서명된 인증서를 사용한 경우 **--insecure** 옵션을 사용합니다.

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

Python

Python 인터프리터에서 다음을 입력합니다.

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

교체:

- **active ceph-mgr** 인스턴스가 있는 노드의 IP 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- **osd 필드에 나열된 OSD의 ID 가 있는 ID**
- 시작하려는 프로세스 (**crub**, **deep-scrub** 또는 **repair**)를 사용 합니다. 프로세스가 **OSD**에서 지원되는지 확인합니다. 자세한 내용은 [1.5.1.9절. “OSD에서 일정을 설정할 수 있는 프로세스를 어떻게 확인할 수 있습니까?”](#) 를 참조하십시오.
- 사용자 이름을 가진 **USER**
- 사용자 암호 가 있는 암호

자체 서명된 인증서를 사용한 경우 **verify=False** 옵션을 사용합니다.


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

1.5.3.2. 새 풀을 어떻게 생성할 수 있습니까?

이 섹션에서는 **RESTful** 플러그인을 사용하여 새 풀을 생성하는 방법을 설명합니다.

curl 명령

명령줄에서 다음을 사용합니다.

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

교체:

- 새 풀의 이름으로 **NAME**
- 배치 그룹 수가 있는 **NUMA**
- 사용자 이름을 가진 **USER**
- **active ceph-mgr** 인스턴스가 있는 노드의 **IP** 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**

메시지가 표시되면 사용자 암호를 입력합니다.

자체 서명된 인증서를 사용한 경우 **--insecure** 옵션을 사용합니다.

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

Python

Python 인터프리터에서 다음을 입력합니다.

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

교체:

- **active ceph-mgr** 인스턴스가 있는 노드의 **IP** 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- 새 풀의 이름으로 **NAME**
- 배치 그룹 수가 있는 **NUMA**
- 사용자 이름을 가진 **USER**
- 사용자 암호가 있는 암호

자체 서명된 인증서를 사용한 경우 **verify=False** 옵션을 사용합니다.

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

1.5.3.3. 풀을 어떻게 제거 할 수 있습니까?

이 섹션에서는 **RESTful** 플러그인을 사용하여 풀을 제거하는 방법을 설명합니다.

이 요청은 기본적으로 금지되어 있습니다. 이를 허용하려면 **Ceph** 구성 가이드에 다음 매개 변수를 추가합니다.

```
mon_allow_pool_delete = true
```

curl 명령

명령줄에서 다음을 사용합니다.

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

교체:

- 사용자 이름을 가진 **USER**
- **active ceph-mgr** 인스턴스가 있는 노드의 **IP** 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**
- **pool** 필드에 나열된 풀 **ID** 가 있는 **ID**

메시지가 표시되면 사용자 암호를 입력합니다.

자체 서명된 인증서를 사용한 경우 **--insecure** 옵션을 사용합니다.

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

Python

Python 인터프리터에서 다음을 입력합니다.

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

교체:

- **active ceph-mgr** 인스턴스가 있는 노드의 **IP** 주소 또는 짧은 호스트 이름이 있는 **CEPH_MANAGER**

- **pool** 필드에 나열된 풀 ID 가 있는 ID
- 사용자 이름을 가진 **USER**
- 사용자 암호 가 있는 암호

자체 서명된 인증서를 사용한 경우 **verify=False** 옵션을 사용합니다.

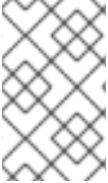
```
$ python
>> import requests
>> result = requests.delete('https://CEPH_MANAGER:8080/api/pool/ID', auth=("USER",
"PASSWORD"), verify=False)
>> print result.json()
```

1.6. 추가 리소스

- API에 대한 자세한 내용은 [부록 A. Ceph RESTful API 사양](#) 을 참조하십시오.
- [GitHub](#) 에서 [API Python 스크립트 테스트](#) 를 참조하십시오.

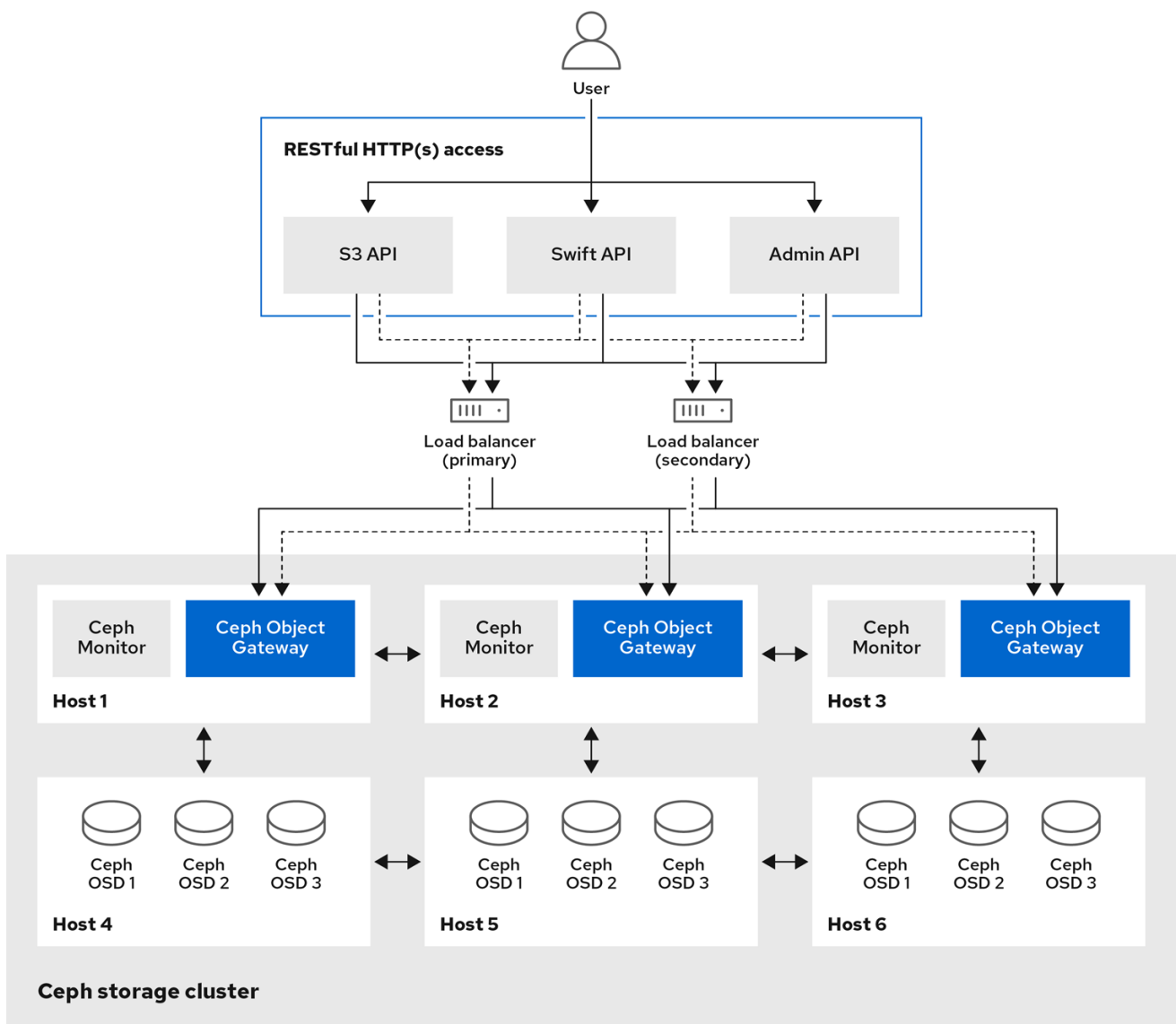
2장. CEPH OBJECT GATEWAY 관리 API

개발자는 **RESTful** 애플리케이션 프로그래밍 인터페이스(API)와 상호 작용하여 **Ceph** 개체 게이트웨이를 관리할 수 있습니다. **Ceph Object Gateway**를 사용하면 **RESTful API**에서 **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.



250_Ceph_0522

관리 **API**는 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
 - 정책
- 오브젝트 관리
 - 제거
 - 정책
- 할당량 관리
 - 사용자 가져오기
 - 사용자 설정

- **Bucket 가져오기**
- **Bucket 설정**
- **사용량 정보 얻기**
- **사용 정보 제거**
- **표준 오류 응답**

2.1. 사전 요구 사항

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

2.2. 관리 작업

관리 **API(Application Programming Interface)** 요청은 구성 가능한 **'admin'** 리소스 진입점으로 시작하는 **URI**에서 수행됩니다. 관리 **API**에 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 **JSON**으로 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
```


2.3. 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 AmazonS3Client와 같은 오픈 소스 S3 클라이언트를 사용합니다. 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 코드에는 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

폴고 **lib** 디렉토리로 이동하여 **JAVA_HOME** 디렉터리의 **/jre/lib/ext** 디렉터리 또는 사용자 지정 **classpath**에 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. **URI**를 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 **Amazon**의 **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 공백 없이 대문자여야 합니다. 공백을 비우지 않으면 인증이 실패합니다. 날짜는 **kafka**로 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 **URI**를 전달하는 것입니다. 예시적인 방법은 사용자 및 하위 사용자를 생성하는 데 **PUT** 을 사용하며, 사용자를 가져오기 위해 **GET**, 사용자 수정을 위한 **POST**, 사용자 삭제용 **DELETE** 를 사용합니다.

요청을 인스턴스화하면 **Date** 헤더와 **Authorization** 헤더를 추가합니다. **Amazon**의 **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 HMAC**를 **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 액세스 키, 시크릿 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",
            "Xac39eCAhITGcCAUreuwe1ZuH5oVQFa51lbEMVoT",
            "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();

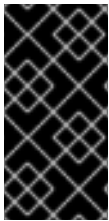
try {
    System.out.println("\nResponse Content is: "
        + EntityUtils.toString(entity, "UTF-8") + "\n");
    response.close();
} catch (IOException e){
    System.err.println ("Encountered an I/O exception.");
    e.printStackTrace();
}
}
}
}

```

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 **S3 인증 섹션**을 참조하십시오.
- **Amazon S3** 인증 절차에 대한 자세한 내용은 **Amazon Simple Storage Service** 설명서의 **REST 요청 서명 및 인증 섹션**을 참조하십시오.

2.4. 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": "gr1VEGIV7rxcP3xvXDFCo4UDwwl2YoNrmtRIIAty"
    }
  ],
  "swift_keys": [],
  "caps": [],
  "op_mask": "read, write, delete",
  "default_placement": "",
  "placement_tags": [],
  "bucket_quota": {
    "enabled": false,
    "max_size_kb": -1,
    "max_objects": -1
  },
  "user_quota": {
    "enabled": false,
    "max_size_kb": -1,
    "max_objects": -1
  },
  "temp_url_keys": []
}
```

2.

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": "gr1VEGIV7rxcP3xvXDFCo4UDwwl2YoNrmtRIlAty"
    }
  ],
  "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.

2.5. 사용자 정보 가져오기

사용자 정보를 가져옵니다.

capabilities

users=read

구문

```

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

```

요청 매개변수

uid

설명

정보가 요청된 사용자입니다.

유형

문자열

예제

foo_user

필수 항목

있음

응답 엔티티

user

설명

사용자 데이터 정보를 위한 컨테이너입니다.

유형

컨테이너

parent

해당 없음

user_id

설명

사용자 ID입니다.

유형

문자열

parent

user

display_name

설명

사용자의 이름을 표시합니다.

유형

문자열

parent

user

일시 중지됨

설명

사용자가 일시 중지되면 **True**입니다.

유형

부울

parent

user

max_buckets

설명

사용자가 소유할 최대 버킷 수입니다.

유형

정수

parent

user

subusers

설명

이 사용자 계정과 연결된 하위 사용자입니다.

유형

컨테이너

parent

user

keys

설명

이 사용자 계정과 연결된 **S3** 키입니다.

유형

컨테이너

parent

user

swift_keys

설명

이 사용자 계정과 연결된 **Swift** 키입니다.

유형

컨테이너

parent

user

caps

설명

사용자 기능.

유형

컨테이너

parent

user

성공하면 응답에 사용자 정보가 포함됩니다.

특수 오류 응답

없음.

2.6. 사용자 생성

새 사용자를 만듭니다. 기본적으로 **S3** 키 쌍은 자동으로 생성되고 응답으로 반환됩니다. **access-key** 또는 **secret-key** 만 제공되는 경우 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
```

요청 매개변수

uid

설명

생성할 사용자 ID입니다.

유형

문자열

예제

`foo_user`

필수 항목

있음

`display-name`

설명

생성할 사용자의 표시 이름입니다.

유형

문자열

예제

`foo_user`

필수 항목

있음

`email`

설명

사용자와 연결된 이메일 주소입니다.

유형

문자열

예제

`foo@bar.com`

필수 항목

없음

`key-type`

설명

생성할 키 유형은 **swift**, **s3**(기본값)입니다.

유형

문자열

예제

s3 [s3]

필수 항목

없음

액세스 키

설명

액세스 키를 지정합니다.

유형

문자열

예제

ABCD0EF12GHIJ2K34LMN

필수 항목

없음

secret-key

설명

시크릿 키를 지정합니다.

유형

문자열

예제

0AbCDEFg1h2i34JkIM5nop6QrSTUV+WxyzaBC7D8

필수 항목

없음

user-caps

설명

사용자 기능.

유형

문자열

예제

usage=read, write; users=read

필수 항목

없음

generate-key

설명

새 키 쌍을 생성하고 기존 인증 키에 를 추가합니다.

유형

부울

예제

True [True]

필수 항목

없음

max-buckets

설명

사용자가 소유할 수 있는 최대 버킷 수를 지정합니다.

유형

정수

예제

500 [1000]

필수 항목

없음

일시 중지됨

설명

사용자를 중지할지 여부 지정

유형

부울

예제

false [False]

필수 항목

없음

응답 엔티티

user

설명

사용자를 중지할지 여부 지정

유형

부울

parent

없음

user_id

설명

사용자 **ID**입니다.

유형

문자열

parent

user

display_name

설명

사용자의 이름을 표시합니다.

유형

문자열

parent

user

일시 중지됨

설명

사용자가 일시 중지되면 **True**입니다.

유형

부울

parent

user

max_buckets

설명

사용자가 소유할 최대 버킷 수입니다.

유형

정수

parent

user

subusers

설명

이 사용자 계정과 연결된 하위 사용자입니다.

유형

컨테이너

parent

user

keys

설명

이 사용자 계정과 연결된 **S3** 키입니다.

유형

컨테이너

parent

user

swift_keys

설명

이 사용자 계정과 연결된 **Swift** 키입니다.

유형

컨테이너

parent

user

caps

설명

사용자 기능.

유형

컨테이너

parent

성공하면 응답에 사용자 정보가 포함됩니다.

특수 오류 응답

UserExists

설명

기존 사용자 생성을 시도합니다.

코드

409 충돌

InvalidAccessKey

설명

잘못된 액세스 키가 지정되어 있습니다.

코드

400 잘못된 요청

InvalidKeyType

설명

잘못된 키 유형이 지정되어 있습니다.

코드

400 잘못된 요청**InvalidSecretKey****설명**

잘못된 보안 키가 지정되어 있습니다.

코드**400 잘못된 요청****KeyExists****설명**

제공된 액세스 키가 존재하며 다른 사용자에게 속합니다.

코드**409 충돌****EmailExists****설명**

제공된 이메일 주소가 있습니다.

코드**409 충돌****InvalidCap****설명**

잘못된 **admin** 기능을 부여하려고 합니다.

코드**400 잘못된 요청****추가 리소스**

-

하위 사용자를 [생성하려면 Red Hat Ceph Storage 개발자 가이드](#) 를 참조하십시오.

2.7. 사용자 수정

기존 사용자를 수정합니다.

capabilities

```
`users=write`
```

구문

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

요청 매개변수

uid

설명

생성할 사용자 **ID**입니다.

유형

문자열

예제

foo_user

필수 항목

있음

display-name

설명

생성할 사용자의 표시 이름입니다.

유형

문자열

예제

foo_user

필수 항목

있음

email

설명

사용자와 연결된 이메일 주소입니다.

유형

문자열

예제

foo@bar.com

필수 항목

없음

generate-key

설명

새 키 쌍을 생성하고 기존 인증 키에 를 추가합니다.

유형

부울

예제

true [False]

필수 항목

없음

액세스 키

설명

액세스 키를 지정합니다.

유형

문자열

예제

ABCD0EF12GHIJ2K34LMN

필수 항목

없음

secret-key

설명

시크릿 키를 지정합니다.

유형

문자열

예제

0AbCDEFg1h2i34JkIM5nop6QrSTUV+WxyzaBC7D8

필수 항목

없음

key-type

설명

생성할 키 유형은 **swift**, **s3**(기본값)입니다.

유형

문자열

예제

s3

필수 항목

없음

user-caps

설명

사용자 기능.

유형

문자열

예제

usage=read, write; users=read

필수 항목

없음

max-buckets

설명

사용자가 소유할 수 있는 최대 버킷 수를 지정합니다.

유형

정수

예제

500 [1000]

필수 항목

없음

일시 중지됨

설명

사용자를 중지할지 여부 지정

유형

부울

예제

false [False]

필수 항목

없음

응답 엔티티

user

설명

사용자를 중지할지 여부 지정

유형

부울

parent

없음

user_id

설명

사용자 **ID**입니다.

유형

문자열

parent

user

display_name

설명

사용자의 이름을 표시합니다.

유형

문자열

parent

user

일시 중지됨

설명

사용자가 일시 중지되면 **True**입니다.

유형

부울

parent

user

max_buckets

설명

사용자가 소유할 최대 버킷 수입니다.

유형

정수

parent

user

subusers

설명

이 사용자 계정과 연결된 하위 사용자입니다.

유형

컨테이너

parent

user

keys

설명

이 사용자 계정과 연결된 **S3** 키입니다.

유형

컨테이너

parent

user

swift_keys

설명

이 사용자 계정과 연결된 **Swift** 키입니다.

유형

컨테이너

parent

user

caps

설명

사용자 기능.

유형

컨테이너

parent

성공하면 응답에 사용자 정보가 포함됩니다.

특수 오류 응답

InvalidAccessKey

설명

잘못된 액세스 키가 지정되어 있습니다.

코드

400 잘못된 요청

InvalidKeyType

설명

잘못된 키 유형이 지정되어 있습니다.

코드

400 잘못된 요청

InvalidSecretKey

설명

잘못된 보안 키가 지정되어 있습니다.

코드

400 잘못된 요청

KeyExists

설명

제공된 액세스 키가 존재하며 다른 사용자에게 속합니다.

코드

409 충돌

EmailExists

설명

제공된 이메일 주소가 있습니다.

코드

409 충돌

InvalidCap

설명

잘못된 **admin** 기능을 부여하려고 합니다.

코드

400 잘못된 요청

추가 리소스

- 하위 사용자를 수정하려면 [Red Hat Ceph Storage 개발자 가이드](#) 를 참조하십시오.

2.8. 사용자 제거

기존 사용자를 제거합니다.

capabilities

```
`users=write`
```

구문

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

요청 매개변수

uid

설명

제거할 사용자 ID입니다.

유형

문자열

예제

foo_user

필수 항목

있음

purge-data

설명

지정된 경우 사용자에게 속한 버킷과 오브젝트도 제거됩니다.

유형

부울

예제

True

필수 항목

없음

응답 엔티티

없음.

특수 오류 응답

없음.

추가 리소스

- 하위 사용자를 제거하려면 [Red Hat Ceph Storage 개발자 가이드](#) 를 참조하십시오.

2.9. 하위 사용자 생성

Swift API를 사용하여 클라이언트에 주로 유용한 새 하위 사용자를 만듭니다.



참고

유효한 요청에는 **gen-subuser** 또는 **subuser**가 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
```

요청 매개변수

uid

설명

하위 사용자를 생성할 사용자 **ID**입니다.

유형

문자열

예제

foo_user

필수 항목

있음

subuser

설명

생성할 하위 사용자 **ID**를 지정합니다.

유형

문자열

예제

sub_foo

필수 항목

예(또는 **gen-subuser**)

gen-subuser

설명

생성할 하위 사용자 **ID**를 지정합니다.

유형

문자열

예제

sub_foo

필수 항목

예(또는 **gen-subuser**)

secret-key

설명

시크릿 키를 지정합니다.

유형

문자열

예제

0AbCDEfg1h2i34JkIM5nop6QrSTUV+WxyzaBC7D8

필수 항목

없음

key-type

설명

생성할 키 유형은 **swift**(기본값), **s3**입니다.

유형

문자열

예제

swift [swift]

필수 항목

없음

액세스

설명

하위 사용자에게 대한 액세스 권한을 설정합니다. 읽기, 쓰기, 읽기 쓰기, 전체 중 하나여야 합니다.

유형

문자열

예제

read

필수 항목

없음

generate-secret

설명

비밀 키를 생성합니다.

유형

부울

예제

true [False]

필수 항목

없음

응답 엔티티

subusers

설명

사용자 계정과 연결된 하위 사용자입니다.

유형

컨테이너

parent

해당 없음

권한

설명

사용자 계정에 대한 하위 사용자 액세스.

유형

문자열

parent

subusers

성공하면 응답에 하위 사용자 정보가 포함됩니다.

특수 오류 응답

SubuserExists

설명

지정된 하위 사용자가 있습니다.

코드

409 충돌

InvalidKeyType

설명

잘못된 키 유형이 지정되어 있습니다.

코드

400 잘못된 요청

InvalidSecretKey

설명

잘못된 보안 키가 지정되어 있습니다.

코드

400 잘못된 요청**InvalidAccess****설명**

잘못된 하위 사용자 액세스

코드**400 잘못된 요청****2.10. 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

요청 매개변수**uid****설명**

하위 사용자를 생성할 사용자 ID입니다.

유형

문자열

예제

foo_user

필수 항목

있음

subuser

설명

수정할 하위 사용자 ID입니다.

유형

문자열

예제

sub_foo

필수 항목

generate-secret

설명

하위 사용자에게 대한 새 비밀 키를 생성하여 기존 키를 교체합니다.

유형

부울

예제

true [False]

필수 항목

없음

Secret

설명

시크릿 키를 지정합니다.

유형

문자열

예제

0AbCDEFG1h2i34JkIM5nop6QrSTUV+WxyzaBC7D8

필수 항목

없음

key-type

설명

생성할 키 유형은 **swift**(기본값), **s3**입니다.

유형

문자열

예제

swift [swift]

필수 항목

없음

액세스

설명

하위 사용자에게 대한 액세스 권한을 설정합니다. 읽기, 쓰기, 읽기 쓰기, 전체 중 하나여야 합니다.

유형

문자열

예제

read

필수 항목

없음

응답 엔티티

subusers

설명

사용자 계정과 연결된 하위 사용자입니다.

유형

컨테이너

parent

해당 없음

id

설명

하위 사용자 ID

유형

문자열

parent

subusers

권한

설명

사용자 계정에 대한 하위 사용자 액세스.

유형

문자열

parent

subusers

성공하면 응답에 하위 사용자 정보가 포함됩니다.

특수 오류 응답

InvalidKeyType

설명

잘못된 키 유형이 지정되어 있습니다.

코드

400 잘못된 요청

InvalidSecretKey

설명

잘못된 보안 키가 지정되어 있습니다.

코드

400 잘못된 요청

InvalidAccess

설명

잘못된 하위 사용자 액세스

코드

400 잘못된 요청

2.11. 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
```

요청 매개변수

uid

설명

제거할 사용자 **ID**입니다.

유형

문자열

예제

foo_user

필수 항목

있음

subuser

설명

제거할 하위 사용자 **ID**입니다.

유형

문자열

예제

sub_foo

필수 항목

있음

purge-keys

설명

하위 사용자에게 속하는 키를 제거합니다.

유형

부울

예제

True [True]

필수 항목

없음

응답 엔티티

없음.

특수 오류 응답

없음.

2.12. 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
```

요청 매개변수

uid

설명

관리 기능을 추가할 사용자 **ID**입니다.

유형

문자열

예제

foo_user

필수 항목

있음

user-caps

설명

사용자에게 추가할 관리 기능.

유형

문자열

예제

usage=read, write

필수 항목

있음

응답 에티티

오버라이드

user

설명

사용자 데이터 정보를 위한 컨테이너입니다.

유형

컨테이너

parent

해당 없음

user_id

설명

사용자 ID

유형

문자열

parent**user****caps**

설명

사용자 기능,

유형

컨테이너

parent**user**

성공하면 응답에 사용자의 기능이 포함됩니다.

특수 오류 응답

InvalidCap

설명

잘못된 **admin** 기능을 부여하려고 합니다.

코드

400 잘못된 요청

2.13. 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
```

요청 매개변수

uid

설명

관리 기능을 제거할 수 있는 사용자 **ID**입니다.

유형

문자열

예제

foo_user

필수 항목

있음

user-caps

설명

사용자에서 제거할 관리 기능.

유형

문자열

예제

usage=read, write

필수 항목

있음

응답 엔티티**user**

설명

사용자 데이터 정보를 위한 컨테이너입니다.

유형

컨테이너

parent

해당 없음

user_id

설명

사용자 ID입니다.

유형

문자열

parent

user

caps

설명

사용자 기능.

유형

컨테이너

parent

user

성공하면 응답에 사용자의 기능이 포함됩니다.

특수 오류 응답

InvalidCap

설명

잘못된 관리자 기능을 제거하려고 합니다.

코드

400 잘못된 요청

NoSuchCap

설명

사용자에게 지정된 기능이 없습니다.

코드

404 찾을 수 없음

2.14. 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
```

요청 매개변수

uid

설명

새 키를 받을 사용자 ID입니다.

유형

문자열

예제

foo_user

필수 항목

있음

subuser

설명

새 키를 수신할 하위 사용자 **ID**입니다.

유형

문자열

예제

sub_foo

필수 항목

없음

key-type

설명

생성할 키 유형은 **swift**, **s3**(기본값)입니다.

유형

문자열

예제

s3 [s3]

필수 항목

없음

액세스 키

설명

액세스 키를 지정합니다.

유형

문자열

예제

AB01C2D3EF45G6H7IJ8K

필수 항목

없음

secret-key

설명

시크릿 키를 지정합니다.

유형

문자열

예제

0ab/CdeFGhij1klmnopqRSTUv1WxyZabcDEFgHij

필수 항목

없음

generate-key

설명

새 키 쌍을 생성하고 기존 인증 키에 를 추가합니다.

유형

부울

예제

true [true]

필수 항목

없음

응답 엔티티

keys

설명

이 사용자 계정과 연결된 생성된 유형의 키입니다.

유형

컨테이너

parent

해당 없음

user

설명

키와 연결된 사용자 계정입니다.

유형

문자열

parent

keys

액세스 키

설명

액세스 키입니다.

유형

문자열

parent

keys

secret-key

설명

시크릿 키입니다.

유형

문자열

parent

keys

특수 오류 응답

InvalidAccessKey

설명

잘못된 액세스 키가 지정되어 있습니다.

코드

400 잘못된 요청

InvalidKeyType

설명

잘못된 키 유형이 지정되어 있습니다.

코드

400 잘못된 요청

InvalidSecretKey

설명

잘못된 보안 키가 지정되어 있습니다.

코드

400 잘못된 요청

InvalidKeyType

설명

잘못된 키 유형이 지정되어 있습니다.

코드

400 잘못된 요청

KeyExists

설명

제공된 액세스 키가 존재하며 다른 사용자에게 속합니다.

코드

409 충돌

2.15. 키 제거

기존 키를 제거합니다.

capabilities

```
`users=write`
```

구문

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

요청 매개변수

액세스 키

설명

제거할 **S3** 키 쌍에 속하는 **S3** 액세스 키입니다.

유형

문자열

예제

AB01C2D3EF45G6H7IJ8K

필수 항목

있음

uid**설명**

키를 제거할 사용자입니다.

유형

문자열

예제

foo_user

필수 항목

없음

subuser**설명**

키를 제거할 하위 사용자입니다.

유형

문자열

예제

sub_foo

필수 항목

없음

key-type

설명

제거할 키 유형은 **swift**, **s3**입니다.



참고

swift 키를 제거해야 합니다.

유형

문자열

예제

swift

필수 항목

없음

특수 오류 응답

없음.

응답 엔티티

없음.

2.16. 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.

2.16.1. 사전 요구 사항



Ceph Object Gateway에서 버킷 알림을 생성합니다.

2.16.2. 버킷 알림 개요

버킷 알림은 버킷에서 특정 이벤트가 발생할 때 **Ceph Object Gateway**에서 정보를 보낼 수 있는 방법을 제공합니다. 버킷 알림을 **HTTP**, **AMQP0.9.1** 및 **Kafka** 엔드포인트로 보낼 수 있습니다. 특정 버킷의 이벤트 및 특정 항목에 대한 버킷 알림을 보내려면 알림 항목을 생성해야 합니다. 이벤트 유형의 하위 집합 또는 모든 이벤트 유형에 대해 기본적으로 버킷 알림을 만들 수 있습니다. 버킷 알림은 키 접두사 또는 접미사, 키와 일치하는 정규 표현식, 개체에 연결된 메타데이터 속성 또는 오브젝트 태그를 기반으로 이벤트를 필터링할 수 있습니다. 버킷 알림에는 버킷 알림 메커니즘에 대한 구성 및 제어 인터페이스를 제공하는 **REST API**가 있습니다.

2.16.3. 영구 알림

영구 알림을 사용하면 **Ceph Object Gateway**에서 주제에서 구성된 엔드포인트로 알림을 안정적이고 비동기적으로 제공할 수 있습니다. 정규 알림은 요청 중에 끝점에 대한 전달이 동기적으로 수행되기 때문에 안정적입니다. 영구 알림을 사용하면 **Ceph Object Gateway**가 끝점이 다운되거나 작업 중에 네트워크 문제가 발생하더라도 알림 보내기를 다시 시도합니다. 이 알림은 끝점에 성공적으로 전달되지 않는 경우 다시 시도됩니다. 알림은 알림을 받은 작업과 관련된 다른 모든 작업이 성공한 후에만 전송됩니다. 끝점이 더 긴 기간 동안 다운되면 알림 큐가 채워지고 이러한 엔드포인트에 대한 알림을 구성한 **S3** 작업이 실패합니다.

2.16.4. 항목 생성

버킷 알림을 생성하기 전에 주제를 생성할 수 있습니다. 주제는 **SNS(Simple Notification Service)** 엔티티이며 모든 주제 작업(즉, 생성, 삭제, 목록, 가져오기 및 **get**)은 **DASD** 작업입니다. 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 주제가 생성된 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]
[&Attributes.entry.9.key=persistent&Attributes.entry.9.value=true|false]
```

요청 매개 변수는 다음과 같습니다.

- **endpoint** : 알람을 보낼 끝점의 URL입니다.
- **OpaqueData**: 불투명 데이터가 주제 구성에 설정되고 해당 주제에서 트리거한 모든 알람에 추가됩니다.
- **persistent**: 이 엔드포인트에 대한 알람이 비동기적인지 여부를 나타냅니다. 기본값은 **false**입니다.
- **HTTP 끝점**:
 - **url**: **https://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:passwords@FQDN:PORT[/VHOST].**
 - 사용자 및 암호의 기본값은 **guest** 및 **guest**입니다.
 - 사용자 및 암호 세부 정보는 **HTTPS**를 통해 제공해야 합니다. 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: 메시지가 최종 목적지로 전달되기 전에 브로커에 보관될 수 있으므로 확인 완료는 필요하지 않습니다. 인증하는 세 가지 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 매개변수의 키와 값은 동일한 줄 또는 특정 순서로 상주할 필요는 없지만 동일한 인덱스를 사용해야 합니다. 특정 인덱스는 순차적이거나 특정 값에서 시작할 필요가 없습니다.



참고

topic-name 은 **AMQP** 주제를 위해 사용됩니다.

●

Kafka 끝점:

○

URL: `kafka://USER:passwords@FQDN:PORT.`

○

기본적으로 **use-ssl** 이 **false** 로 설정됩니다. **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 **CA**가 사용됩니다.
- 사용자 및 암호는 **HTTP[S]**를 통해서만 제공할 수 있습니다. 그렇지 않으면 주제 생성 요청이 거부됩니다.
- 사용자 및 암호는 **use-ssl** 과 함께만 제공될 수 있습니다. 그렇지 않으면 브로커에 대한 연결이 실패합니다.
- 포트 기본값은: **9092**입니다.
- **Kafka-ack-level:** 메시지가 최종 목적지로 전달되기 전에 브로커가 지속되기 전에 브로커가 지속할 수 있으므로 필요한 확인 작업을 종료하지 않습니다. 감사하는 두 가지 방법이 있습니다.
 - **None:** 브로커에 전송되는 경우 메시지가 배달되는 것으로 간주됩니다.
 - **broker:** 기본적으로 브로커가 승인한 경우 메시지가 배달되는 것으로 간주됩니다.

다음은 응답 형식의 예입니다.

예제

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



참고

응답의 **ARN(Amazon Resource Name)** 항목에는 **arn:aws:sns:ZONE_GROUP:TENANT:topic** 형식이 있습니다.

다음은 **AMQP0.9.1** 끝점의 예입니다.

예제

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

2.16.5. 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>
  <GetTopicResult>
    <Topic>
      <User></User>
      <Name></Name>
      <EndPoint>
        <EndpointAddress></EndpointAddress>
        <EndpointArgs></EndpointArgs>
        <EndpointTopic></EndpointTopic>
        <HasStoredSecret></HasStoredSecret>
        <Persistent></Persistent>
      </EndPoint>
      <TopicArn></TopicArn>
      <OpaqueData></OpaqueData>
    </Topic>
  </GetTopicResult>
  <ResponseMetadata>
    <RequestId></RequestId>
  </ResponseMetadata>
</GetTopicResponse>
```

태그 및 정의는 다음과 같습니다.

- **User:** 주제를 생성한 사용자의 이름입니다.
- **Name:** 주제의 이름입니다.

- - **JSON 형식의 끝점은 다음과 같습니다.**
 - **EndpointAddress:** 엔드포인트 URL입니다. 엔드포인트 URL에 사용자 및 암호 정보가 포함된 경우 HTTPS를 통해 요청을 수행해야 합니다. otherwise, 주제 get 요청이 거부됩니다.
 - **EndPointArgs:** 끝점 인수입니다.
 - **EndpointTopic:** 엔드포인트로 전송되는 주제 이름은 위의 예제 주제 이름과 다를 수 있습니다.
 - **HasStoredSecret:** 끝점 URL에 사용자 및 암호 정보가 포함된 경우 true 입니다.
 - **persistent:** 주제가 지속되면 true 입니다.
- **TopicArn:** 주제입니다.
- **OpaqueData:** 항목에 대한 불투명 데이터 집합입니다.

2.16.6. 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.

구문

POST
Action=ListTopics

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

```
<ListTopicsResponse xmlns="https://sns.amazonaws.com/doc/2020-03-31/">
  <ListTopicsResult>
    <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 않으면 주제 목록 요청이 거부됩니다.

2.16.7. 주제 삭제

삭제된 주제를 제거하면 작업이 작동하지 않으며 실패하지 않습니다.

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>
```

2.16.8. 이벤트 레코드

이벤트에는 **Ceph Object Gateway**에서 수행한 작업에 대한 정보가 있으며 **HTTP, HTTPS, Kafka** 또는 **AMQ0.9.1**과 같은 선택한 끝점에서 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": "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 ID입니다.
- **requestParameters.sourceIPAddress** - 이벤트를 트리거한 클라이언트의 IP 주소입니다. 이 필드는 지원되지 않습니다.
- **responseElements.x-amz-request-id:** 이벤트를 트리거한 요청 ID입니다.
- **responseElements.x_amz_id_2:** 이벤트가 트리거된 Ceph Object Gateway의 ID입니다. ID 형식은 RGWID-ZONE-ZONEGROUP 입니다.
- **s3.configurationId:** 이벤트를 생성한 알람 ID입니다.
- **s3.bucket.name:** 버킷의 이름입니다.

- **s3.bucket.ownerIdentity.principalId:** 버킷의 소유자입니다.
- **s3.bucket.arn:** 버킷의 *Amazon Resource Name (ARN)*입니다.
- **s3.bucket.id:** 버킷의 ID입니다.
- **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 16진수 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 구조](#)를 참조하십시오.

2.16.9. 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`**

2.17. 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
```

요청 매개변수

bucket

설명

정보를 반환하는 버킷입니다.

유형

문자열

예제

foo_bucket

필수 항목

없음

uid

설명

버킷 정보를 검색할 사용자입니다.

유형

문자열

예제

foo_user

필수 항목

없음

통계

설명

버킷 통계 반환.

유형

부울

예제

true [False]

필수 항목

없음

응답 엔티티**통계****설명**

버킷 정보당.

유형

컨테이너

parent

해당 없음

buckets**설명**

하나 이상의 버킷 컨테이너 목록을 포함합니다.

유형

컨테이너

parent

buckets

bucket

설명

단일 버킷 정보용 컨테이너입니다.

유형

컨테이너

parent

buckets

name**설명**

버킷의 이름입니다.

유형

문자열

parent

bucket

pool**설명**

버킷이 저장된 풀입니다.

유형

문자열

parent

bucket

id**설명**

고유한 버킷 ID입니다.

유형

문자열

parent

bucket

marker

설명

내부 버킷 태그.

유형

문자열

parent

bucket

소유자

설명

버킷 소유자의 사용자 **ID**입니다.

유형

문자열

parent

bucket

사용법

설명

스토리지 사용량 정보.

유형

컨테이너

parent

bucket**index****설명**

버킷 인덱스의 상태.

유형

문자열

parent**bucket**

성공하면 요청이 버킷 정보를 사용하여 버킷의 컨테이너를 반환합니다.

특수 오류 응답**IndexRepairFailed****설명**

버킷 인덱스 복구에 실패했습니다.

코드

409 충돌

2.18. 버킷 인덱스 확인

기존 버킷의 인덱스를 확인합니다.

**참고**

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

capabilities

buckets=write

구문

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

요청 매개변수

bucket

설명

정보를 반환하는 버킷입니다.

유형

문자열

예제

foo_bucket

필수 항목

있음

check-objects

설명

다중 파트 오브젝트 회계를 확인합니다.

유형

부울

예제

true [False]

필수 항목

없음

fix

설명

또한 확인할 때 버킷 인덱스를 수정합니다.

유형

부울

예제

false [False]

필수 항목

없음

응답 엔티티

index

설명

버킷 인덱스의 상태.

유형

문자열

특수 오류 응답

IndexRepairFailed

설명

버킷 인덱스 복구에 실패했습니다.

코드

409 충돌

2.19. 버킷 제거

기존 버킷을 제거합니다.

capabilities

```
`buckets=write`
```

구문

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

요청 매개변수

bucket

설명

제거할 버킷입니다.

유형

문자열

예제

foo_bucket

필수 항목

있음

purge-objects

설명

삭제하기 전에 버킷의 오브젝트를 제거합니다.

유형

부울

예제

true [False]

필수 항목

없음

응답 엔티티

없음.

특수 오류 응답

BucketNotCrashLoopBackOff

설명

비어 있지 않은 버킷을 삭제하려고 했습니다.

코드

409 충돌

ObjectRemovalFailed

설명

개체를 제거할 수 없습니다.

코드

409 충돌

2.20. 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`

요청 매개변수

bucket

설명

연결할 버킷입니다.

유형

문자열

예제

`foo_bucket`

필수 항목

있음

uid

설명

버킷을 연결할 사용자 **ID**입니다.

유형

문자열

예제

foo_user

필수 항목

있음

응답 엔티티

bucket

설명

단일 버킷 정보용 컨테이너입니다.

유형

컨테이너

parent

해당 없음

name

설명

버킷의 이름입니다.

유형

문자열

parent

bucket

pool

설명

버킷이 저장된 풀입니다.

유형

문자열

parent

bucket

id

설명

고유한 버킷 ID입니다.

유형

문자열

parent

bucket

marker

설명

내부 버킷 태그.

유형

문자열

parent

bucket

소유자

설명

버킷 소유자의 사용자 ID입니다.

유형

문자열

parent

bucket**사용법****설명**

스토리지 사용량 정보.

유형

컨테이너

parent**bucket****index****설명**

버킷 인덱스의 상태.

유형

문자열

parent**bucket****특수 오류 응답****BucketUnlinkFailed****설명**

지정된 사용자로부터 버킷을 연결할 수 없습니다.

코드

409 충돌

BucketLinkFailed**설명**

지정된 사용자에게 버킷을 연결할 수 없습니다.

코드

409 충돌

2.21. 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

요청 매개변수

bucket

설명

연결할 버킷입니다.

유형

문자열

예제

foo_bucket

필수 항목

있음

uid

설명

버킷을 연결할 사용자 ID입니다.

유형

문자열

예제

foo_user

필수 항목

있음

응답 엔티티

없음.

특수 오류 응답

BucketUnlinkFailed

설명

지정된 사용자로부터 버킷을 연결할 수 없습니다.

유형

409 충돌**2.22. 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`

요청 매개변수

bucket

설명

정책을 읽을 버킷입니다.

유형

문자열

예제

foo_bucket

필수 항목

있음

object

설명

정책을 읽을 오브젝트입니다.

유형

문자열

예제

foo.txt

필수 항목

없음

응답 엔티티

policy

설명

액세스 제어 정책.

유형

컨테이너

parent

해당 없음

성공하면 오브젝트 또는 버킷 정책을 반환합니다.

특수 오류 응답

IncompleteBody

설명

버킷 정책 요청 또는 버킷에 대해 버킷이 지정되지 않았으며 오브젝트 정책 요청에 대해 오브젝트가 지정되지 않았습니다.

코드

400 잘못된 요청

2.23. 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
```

요청 매개변수

bucket

설명

제거할 개체가 포함된 버킷입니다.

유형

문자열

예제

foo_bucket

필수 항목

있음

object

설명

제거할 오브젝트입니다.

유형

문자열

예제

foo.txt

필수 항목

있음

응답 엔티티

없음.

특수 오류 응답

NoSuchObject

설명

지정된 오브젝트가 존재하지 않습니다.

코드

404 찾을 수 없음**ObjectRemovalFailed**

설명

개체를 제거할 수 없습니다.

코드

409 충돌**2.24. 할당량**

관리 운영 **API**를 사용하면 사용자 및 사용자가 소유한 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.

2.25. 사용자 할당량 가져오기

할당량을 가져오려면 사용자에게 읽기 권한으로 설정된 사용자 기능이 있어야 합니다.

구문

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

2.26. 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 쓰기 작업에서 인코딩된 할당량 설정에 대한 **JSON** 표시가 포함되어야 합니다.

2.27. 버킷 할당량 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
```

요청 매개변수

bucket

설명

정보를 반환하는 버킷입니다.

유형

문자열

예제

foo_bucket

필수 항목

없음

uid

설명

버킷 정보를 검색할 사용자입니다.

유형

문자열

예제

foo_user

필수 항목

없음

통계

설명

버킷 통계 반환.

유형

부울

예제

true [False]

필수 항목

없음

응답 엔티티

통계

설명

버킷 정보당.

유형

컨테이너

parent

해당 없음

buckets

설명

하나 이상의 버킷 컨테이너 목록을 포함합니다.

유형

컨테이너

parent

해당 없음

bucket

설명

단일 버킷 정보용 컨테이너입니다.

유형

컨테이너

parent

buckets

name

설명

버킷의 이름입니다.

유형

문자열

parent

bucket

pool

설명

버킷이 저장된 풀입니다.

유형

문자열

parent

bucket

id

설명

고유한 버킷 ID입니다.

유형

문자열

parent

bucket

marker

설명

내부 버킷 태그.

유형

문자열

parent

bucket

소유자

설명

버킷 소유자의 사용자 ID입니다.

유형

문자열

parent

bucket

사용법

설명

스토리지 사용량 정보.

유형

컨테이너

parent

bucket

index

설명

버킷 인덱스의 상태.

유형

문자열

parent

bucket

성공하면 요청이 버킷 정보를 사용하여 버킷의 컨테이너를 반환합니다.

특수 오류 응답

IndexRepairFailed

설명

버킷 인덱스 복구에 실패했습니다.

코드

409 충돌

2.28. 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 쓰기 작업에서 인코딩된 할당량 설정에 대한 **JSON** 표시가 포함되어야 합니다.

2.29. 사용 정보 받기

대역폭 사용량 정보 요청.

capabilities

`usage=read`

구문

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

요청 매개변수

uid

설명

정보가 요청된 사용자입니다.

유형

문자열

필수 항목

있음

start

설명

데이터 요청이 시작된 시간 및 선택적으로 날짜입니다. 예를 들어 **2012-09-25 16:00:00** 과 같습니다.

유형

문자열

필수 항목

없음

end

설명

데이터 요청이 종료된 날짜 및 선택적으로 시간. 예를 들어 **2012-09-25 16:00:00** 과 같습니다.

유형

문자열

필수 항목

없음

show-entries**설명**

데이터 항목을 반환할지 여부를 지정합니다.

유형

부울

필수 항목

없음

show-summary**설명**

데이터 항목을 반환할지 여부를 지정합니다.

유형

부울

필수 항목

없음

응답 엔터티**사용법****설명**

사용 정보를 위한 컨테이너입니다.

유형

컨테이너

항목

설명

사용 항목 정보를 위한 컨테이너입니다.

유형

컨테이너

user

설명

사용자 데이터 정보를 위한 컨테이너입니다.

유형

컨테이너

소유자

설명

버킷을 소유하는 사용자의 이름입니다.

유형

문자열

bucket

설명

버킷 이름입니다.

유형

문자열

time

설명

첫 번째 관련 시간의 시작으로 반올림되는 데이터를 지정하는 데 시간이 더 낮습니다.

유형

문자열

epoch**설명**

1/1/1970 이후 지정된 시간(초)입니다.

유형

문자열

카테고리**설명**

통계 카테고리에 대한 컨테이너입니다.

유형

컨테이너

항목**설명**

통계 항목에 대한 컨테이너입니다.

유형

컨테이너

category**설명**

통계가 제공되는 요청 카테고리의 이름입니다.

유형

문자열

bytes_sent

설명

Ceph Object Gateway에서 보내는 바이트 수입니다.

유형

정수

bytes_received

설명

Ceph Object Gateway에서 수신한 바이트 수입니다.

유형

정수

ops

설명

작업 수입니다.

유형

정수

successful_ops

설명

성공한 작업 수입니다.

유형

정수

summary

설명

성공한 작업 수입니다.

유형

컨테이너

total

설명

통계 요약 합계에 대한 컨테이너입니다.

유형

컨테이너

성공하면 응답에 요청된 정보가 포함됩니다.

2.30. 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
```

요청 매개변수

uid

설명

정보가 요청된 사용자입니다.

유형

문자열

예제

foo_user

필수 항목

있음

start

설명

데이터 요청이 시작된 시간 및 선택적으로 날짜입니다. 예를 들어 **2012-09-25 16:00:00** 과 같습니다.

유형

문자열

예제

2012-09-25 16:00:00

필수 항목

없음

end

설명

데이터 요청이 종료된 날짜 및 선택적으로 시간. 예를 들어 **2012-09-25 16:00:00** 과 같습니다.

유형

문자열

예제

2012-09-25 16:00:00

필수 항목

없음

remove-all

설명

다중 사용자 데이터 제거를 승인하기 위해 **uid** 를 지정하지 않은 경우 필수 항목입니다.

유형

부울

예제

true [False]

필수 항목

없음

2.31. 표준 오류 응답

다음 목록은 표준 오류 응답 및 해당 설명을 나열합니다.

AccessDenied

설명

액세스 거부.

코드

403 forbidden

InternalError

설명

내부 서버 오류.

코드

500 내부 서버 오류

NoSuchUser

설명

사용자가 존재하지 않습니다.

코드

404 찾을 수 없음

NoSuchBucket

설명

버킷이 존재하지 않습니다.

코드

404 찾을 수 없음

NoSuchKey

설명

이러한 액세스 키가 없습니다.

코드

404 찾을 수 없음

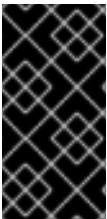
3장. CEPH OBJECT GATEWAY 및 S3 API

개발자는 **Amazon S3** 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 오브젝트를 관리할 수 있습니다.

3.1. 사전 요구 사항

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

3.2. 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 **0B**에서 최대 **5TB**까지 다양합니다. **PUT** 단일 업로드할 수 있는 가장 큰 오브젝트는 **5GB**입니다. **100MB**보다 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 **16,000**바이트로 제한됩니다.
- **Red Hat Ceph Storage** 클러스터가 **S3** 오브젝트 및 메타데이터를 저장하기 위해 생성하는 데이터 오버헤드 양: **200-300**바이트와 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 필드](#)에 대한 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드를 참조하십시오.

3.3. S3 API를 사용하여 CEPH 오브젝트 게이트웨이 액세스

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

3.3.1. 사전 요구 사항

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

3.3.2. S3 인증

Ceph Object Gateway에 대한 요청은 인증되거나 인증되지 않을 수 있습니다. **Ceph Object Gateway**는 인증되지 않은 요청이 익명 사용자가 전송된다고 가정합니다. **Ceph Object Gateway**는 카나리아 **ACL**을 지원합니다.

대부분의 사용 사례의 경우 클라이언트는 **Amazon SDK**의 **AmazonS3Client** 및 **Python Boto**와 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 Gateway**는 **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 ID에 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. 사전순으로 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.

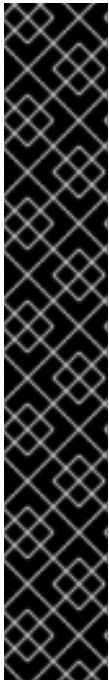
3.3.3. S3 서버 측 암호화

Ceph Object Gateway는 **S3 API**(애플리케이션 프로그래밍 인터페이스)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 Gateway**에서 **SSL**을 사용하지 않는 한 **Red Hat**은 클라이언트의 **S3** 암호화를 지원하지 않습니다. 그러나 테스트 목적으로 관리자는 **ceph config set client.rgw** 명령을 사용하여 **rgw_crypt_require_ssl** 구성 설정을 **false**로 설정한 다음 **Ceph Object Gateway** 인스턴스를 다시 시작하여 테스트 중에 **SSL**을 비활성화할 수 있습니다.

프로덕션 환경에서는 **SSL**을 통해 암호화된 요청을 보낼 수 없습니다. 이러한 경우 서버 측 암호화와 함께 **HTTP**를 사용하여 요청을 보냅니다.

서버 측 암호화를 사용하여 **HTTP**를 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 Gateway**의 키를 기억해야 합니다.

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

고객은 키 관리를 처리하고 **S3** 클라이언트가 키를 **Ceph Object Gateway**에 전달하므로 **Ceph Object Gateway**에는 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 Gateway**는 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 유일하게 테스트된 키 관리 구현은 **HashiCorp Vault**와 **OpenStack Barbican**입니다. 그러나 **OpenStack Barbican**은 기술 프리뷰이며 프로덕션 시스템에서 사용할 수 없습니다.

추가 리소스

- [Amazon SSE-C](#)
- [Amazon SSE-KMS](#)
- [서버 측 암호화 구성](#)
- [HashiCorp Vault](#)

3.3.4. 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 3.1. 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.
WRITE_ACP	grantee는 버킷 ACL을 작성할 수 있습니다.	grantee는 오브젝트 ACL에 쓸 수 있습니다.

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

3.3.5. S3를 사용하여 Ceph Object Gateway에 대한 액세스 준비

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

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 가이드](#)에서 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.

루트로서 **NetworkManager**를 중지합니다.

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

c.

루트로서 게이트웨이 서버의 **IP**를 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 **FQDN**으로 바꿉니다.

d.

하위 도메인 요청을 확인합니다.

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

FQDN_OF_GATEWAY_NODE 를 게이트웨이 노드의 **FQDN**으로 교체합니다.



주의

로컬 **DNS** 캐싱을 위해 게이트웨이 서버를 설정하는 것은 테스트 목적으로만 사용됩니다. 이 작업을 수행한 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. 이러

한 키는 **S3** 액세스 및 후속 버킷 관리 작업에 필요합니다.

3.3.6. Ruby AWS S3를 사용하여 Ceph Object Gateway에 액세스

S3 액세스에 대해 **aws-s3 gem**과 함께 **Ruby** 프로그래밍 언어를 사용할 수 있습니다. **Ruby AWS::S3**을 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** 및 **ruby-lib s**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 교체합니다.
MY_ACCESS_KEY 및 **MY_SECRET_KEY** 를 **Red Hat Ceph Storage Object Gateway** 구성 및
 관리 가이드에 언급된 **S3** 액세스를 위해 **radosgw** 사용자를 생성할 때 생성된 **access_key** 및
secret_key 로 교체합니다.

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
```


명령의 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. 다음으로, 빈 버킷을 삭제하기 전에 위의 **del_empty_bucket.rb** 파일을 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.

3.3.7. 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 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** 및 **ruby-lib s** 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@dev ~]$ 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 교체합니다.
MY_ACCESS_KEY 및 **MY_SECRET_KEY** 를 **Red Hat Ceph Storage Object Gateway** 구성 및
 관리 가이드에 언급된 **S3** 액세스를 위해 **radosgw** 사용자를 생성할 때 생성된 **access_key** 및
secret_key 로 교체합니다.

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@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.

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

10.

파일을 실행합니다.

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

명령의 출력이 **true** 이면 버킷 **my-new-bucket2** 가 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'

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

13.

파일을 실행합니다.

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

출력은 다음과 같아야 합니다.

```
my-new-bucket2 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'

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@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'

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@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'

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

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 버킷을 생성합니다(예: **my-new-bucket6**, **my-new-bucket7**). 다음으로, 빈 버킷을 삭제하기 전에 위의 **del_empty_bucket.rb** 파일을 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'

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@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'

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

28.

파일을 실행합니다.

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

그러면 **hello.txt** 오브젝트가 삭제됩니다.

3.3.8. 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 예**를 제공합니다.

중요

아래 예제는 **php v5.4.16** 및 **aws-sdk v2.8.24**에 대해 테스트됩니다. **php >= 5.5** 이상 이 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.

PHP용 aws-sdk의 **zip** 아카이브를 **다운로드하여** 압축을 풉니다.

3.

프로젝트 디렉터리를 생성합니다.

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

4.

*추출된 **aws** 디렉터를 프로젝트 디렉터리에 복사합니다. 예를 들어 다음과 같습니다.*

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

5.

연결 파일을 생성합니다.

```
[user@dev ~]$ 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 Object Gateway** 구성 및

관리 가이드에 언급된 **S3** 액세스를 위해 **radosgw** 사용자를 생성할 때 생성된 **access_key** 및 **secret_key** 로 교체합니다. **PATH_TO_AWS** 를 **php** 프로젝트 디렉터리에 복사한 추출된 **AWS** 디렉터리의 절대 경로로 교체합니다.

파일을 저장하고 편집기를 종료합니다.

7.

파일을 실행합니다.

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

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

8.

버킷을 만들기 위한 새 파일을 만듭니다.

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

10.

소유 버킷을 나열할 새 파일을 만듭니다.

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

출력은 다음과 유사해야 합니다.

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

12.

hello.txt 라는 소스 파일을 먼저 생성하여 오브젝트를 생성합니다.

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


13.

새 **php** 파일을 만듭니다.

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

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

15.

버킷의 콘텐츠를 나열할 새 파일을 만듭니다.

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

출력은 다음과 유사합니다.

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

17.

빈 버킷을 삭제하기 위해 새 파일을 생성합니다.

```
[user@dev ~]$ 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.

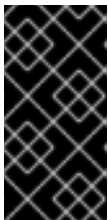
```
[user@dev ~]$ php -f del_empty_bucket.php | echo $?
```

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



참고

create_bucket.php 파일을 편집하여 빈 버킷을 생성합니다(예: **my-new-bucket4**, **my-new-bucket5**). 다음으로 빈 버킷을 삭제하기 전에 위의 **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@dev ~]$ 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@dev ~]$ php -f delete_object.php
```

그러면 **hello.txt** 오브젝트가 삭제됩니다.

3.3.9. 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 가이드의 Keystone과 함께 STS Lite 구성 섹션을 참조하십시오.](#)
- [STS Lite 및 Keystone의 제한 사항에 대한 자세한 내용은 Red Hat Ceph Storage 개발자 가이드의 Keystone과 함께 STS Lite 사용 제한 사항을 참조하십시오.](#)

3.3.9.1. 보안 토큰 서비스 애플리케이션 프로그래밍 인터페이스

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

AssumeRole

이 **API**는 교차 계정 액세스에 대한 임시 인증 정보 집합을 반환합니다. 이러한 임시 자격 증명을 사용하면 **Assume Role API**에 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 1에서 2048자의 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 2에서 1224자의 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 **MFA**가 필요한 경우 **MFA(Multi-factor authentication)** 장치에서 생성된 값입니다. **MFA** 장치가 필요하고 이 매개변수 값이 비어 있거나 만료된 경우 **AssumeRole** 호출은 "액세스 거부" 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 2~64자 길이입니다.,,,,.....
. 문자는 허용되지만 공백은 사용할 수 없습니다.

유형

문자열

필수 항목

있음

정책

설명

인라인 세션에서 사용할 **JSON** 형식의 **IAM(Identity and Access Management Policy)**입니다. 이 매개 변수의 값은 길이가 1에서 2048자의 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 4~2048자인 것입니다.

유형

문자열

필수 항목

없음

WebIdentityToken

설명

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

유형

문자열

필수 항목

없음

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 [보안 토큰 서비스 API](#) 섹션을 참조하십시오.
- **Amazon Web Services** 보안 토큰 서비스, [AssumeRole](#) 작업입니다.
- **Amazon Web Services** 보안 토큰 서비스, [AssumeRoleWithWebIdentity](#) 작업.

3.3.9.2. 보안 토큰 서비스 구성

rgw_sts_key, **rgw_s3_auth_use_sts** 옵션을 설정하여 **Ceph Object Gateway**와 함께 사용하도록 **Secure Token Service(STS)**를 구성합니다.



참고

S3 및 **STS API**는 동일한 네임스페이스에 공존하고 모두 **Ceph Object Gateway**의 동일한 끝점에서 액세스할 수 있습니다.

사전 요구 사항

- 실행 중인 **Red Hat Ceph Storage** 클러스터.
- 실행 중인 **Ceph** 오브젝트 게이트웨이.
- **Ceph Manager** 노드에 대한 루트 수준 액세스.

절차

1. **Ceph Object Gateway** 클라이언트에 다음 구성 옵션을 설정합니다.

구문

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

예제

```
[root@mgr ~]# ceph config set client.rgw rgw_sts_key  
9a2142613c504c42a23ba2b82147dc28  
[root@mgr ~]# ceph config set client.rgw rgw_s3_auth_use_sts true
```

추가 리소스

- **STS API**에 대한 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 [Secure Token Service 애플리케이션 프로그래밍 인터페이스](#) 섹션을 참조하십시오.
- **Ceph** 구성 데이터베이스 사용에 대한 자세한 내용은 **Red Hat Ceph Storage** 구성 가이드의 **Ceph 구성 기본** 장을 참조하십시오.

3.3.9.3. OpenID Connect 공급자의 사용자 생성

Ceph Object Gateway와 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) API를 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

위 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.

3.3.9.4. OpenID Connect 공급자의 지문 가져오기

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

사전 요구 사항

- **openssl 및 curl 패키지를 설치합니다.**

절차

1. **IDP의 URL에서 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 사이트의 **OpenID Connect ID** 공급자 문서를 보려면 루트 **CA Thumbprint**를 참조하십시오.
- **STS API**에 대한 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 **보안 토큰 서비스 애플리케이션 프로그래밍 인터페이스** 섹션을 참조하십시오.
- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 **보안 토큰 서비스 API** 섹션을 참조하십시오.

3.3.9.5. 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 5.0** 이상.
- 실행 중인 **Ceph** 오브젝트 게이트웨이.
- **Boto Python** 모듈, 버전 **3** 이상 설치.
- **Ceph Manager** 노드에 대한 루트 수준 액세스.

- **OpenStack 노드에 대한 사용자 수준 액세스.**

절차

1. **Ceph Object Gateway 클라이언트에 다음 구성 옵션을 설정합니다.**

구문

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

예제

```
[root@mgr ~]# ceph config set client.rgw rgw_sts_key
9a2142613c504c42a23ba2b82147dc28
[root@mgr ~]# ceph config set client.rgw rgw_s3_auth_use_sts true
```

2. **OpenStack 노드에서 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 |
+-----+
```

3.

생성된 자격 증명을 사용하여 **GetSessionToken API**를 사용하여 임시 보안 인증 정보 집합을 다시 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
)
```

4.

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'],
    )
```

5.

새 **S3Access** 역할을 생성하고 정책을 구성합니다.

a.

관리 **CAPS**가 있는 사용자를 할당합니다.

구문

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

예제

```
[root@mgr ~]# 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

```
[root@mgr ~]# radosgw-admin role create --role-name=S3Access --
path=/application_abc/component_xyz/ --assume-role-policy-doc="{\"Version\": \"2012-10-
17\", \"Statement\": [{\"Effect\": \"Allow\", \"Principal\": {\"AWS\": [
\"arn:aws:iam::user/TESTER\"]}, \"Action\": [\"sts:AssumeRole\"]}]}"
```

c.

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

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

d.

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

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

6.

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

예제

```
import boto3

access_key = 11BS02LGFB6AL6H1ADMW
secret_key = vzCEkuryfn060dfec4fgQPqFrncKElkh3ZcdOANY

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** 가이드의 **Test S3 Access** 섹션을 참조하십시오.
- 자세한 내용은 **Red Hat Ceph Storage Object Gateway** 가이드의 **Create a User** 섹션을 참조하십시오.

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

Keystone의 제한 사항은 STS(Secure Token Service) 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 5.0 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 overridden in ``add_auth``
        # later for real requests.
        self._region_name = region_name
        if service_name == 'sts': ❶
            self._service_name = 's3' ❷
        else: ❸
            self._service_name = service_name ❹
```

- 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 ❶
    else: ❷
        request.headers['X-Amz-Content-SHA256'] = self.payload(request)

```

추가 리소스



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

3.4. S3 버킷 작업

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

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

표 3.2. 버킷 작업

기능	상태	참고
Buckets 나열	지원됨	
Bucket을 생성	지원됨	다양한 카나리아 ACL 세트.
Bucket 웹 사이트 입력	지원됨	
Bucket 웹 사이트 가져오기	지원됨	
Bucket 웹 사이트 삭제	지원됨	
버킷 라이프 사이클	부분적으로 지원됨	expiration, NoncurrentVersionExpiration 및 AbortIncompleteMultipartUpload 지원

기능	상태	참고
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	
멀티 테넌트 Bucket 작업	지원됨	

3.4.1. 사전 요구 사항

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

3.4.2. 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 Hat은 **put,post,multipartUpload, copy** 와 같은 **ObjectCreate** 이벤트를 지원합니다. **Red Hat**은 **object_delete** 및 **s3_multi_object_delete** 와 같은 **ObjectRemove** 이벤트도 지원합니다.

요청 엔터티

NotificationConfiguration

설명

TopicConfiguration 엔터티 목록.

유형

컨테이너

필수 항목

있음

TopicConfiguration

설명

이벤트 엔티티 목록, 주제.

유형

컨테이너

필수 항목

있음

id

설명

알림의 이름입니다.

유형

문자열

필수 항목

있음

주제

설명

주제 **Amazon** 리소스 이름(ARN)



참고

주제를 사전에 만들어야 합니다.

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** 엔티티 목록입니다. 대부분의 경우 **3**개의 엔티티가 목록에 있을 수 있습니다(예: **Name** 은 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. 값에 키 접두사, 키 접미사 또는 키와 일치하는 정규식을 포함합니다.

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.

3.4.3. 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 엔티티 목록.

유형

컨테이너

필수 항목

있음

TopicConfiguration

설명

이벤트 엔티티 목록, 주제.

유형

컨테이너

필수 항목

있음

id

설명

알림의 이름입니다.

유형

문자열

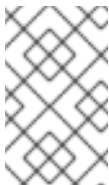
필수 항목

있음

주제

설명

주제 **Amazon** 리소스 이름(**ARN**)



참고

주제를 사전에 만들어야 합니다.

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.

3.4.4. 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 삭제하는 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.

3.4.5. 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.

3.4.6. 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

응답 엔티티

버킷

설명

버킷 목록의 컨테이너입니다.

유형

컨테이너

bucket

설명

버킷 정보용 컨테이너입니다.

유형

컨테이너

이름

설명

버킷 이름입니다.

유형

문자열

CreationDate

설명

버킷이 생성된 **UTC** 시간입니다.

유형

날짜

ListAllMyBucketsResult

설명

결과물의 컨테이너입니다.

유형

컨테이너

소유자

설명

버킷 소유자 ID 및 **DisplayName** 의 컨테이너입니다.

유형

컨테이너

ID

설명

버킷 소유자의 ID입니다.

유형

문자열

DisplayName

설명

버킷 소유자의 표시 이름입니다.

유형

문자열

3.4.7. S3 버킷 오브젝트 목록을 반환합니다.

버킷 개체 목록을 반환합니다.

구문

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

매개 변수

접두사

설명

지정된 접두사가 포함된 오브젝트만 반환합니다.

유형

문자열

delimiter

설명

접두사와 나머지 오브젝트 이름 간의 구분 기호입니다.

유형

문자열

marker

설명

반환된 오브젝트 목록의 시작 인덱스입니다.

유형

문자열

max-keys

설명

반환할 최대 키 수입니다. 기본값은 **1000**입니다.

유형

정수

HTTP 응답

200

상태 코드

OK

설명

검색된 버킷입니다.

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

버킷 응답 엔티티

ListBucketResult

설명

오브젝트 목록의 컨테이너입니다.

유형

엔티티

이름

설명

콘텐츠가 반환될 버킷의 이름입니다.

유형

문자열

접두사

설명

오브젝트 키의 접두사입니다.

유형

문자열

마커

설명

반환된 오브젝트 목록의 시작 인덱스입니다.

유형

문자열

MaxKeys

설명

반환된 최대 키 수입니다.

유형

정수

구분자

설명

설정된 경우 접두사가 동일한 오브젝트가 **CommonPrefixes** 목록에 표시됩니다.

유형

문자열

IsTruncated

설명

true 인 경우 버킷 콘텐츠 서브 세트만 반환됩니다.

유형

부울

CommonPrefixes

설명

여러 오브젝트에 동일한 접두사가 포함된 경우 이 목록에 표시됩니다.

유형

컨테이너

ListBucketResult에는 각 오브젝트가 콘텐츠 컨테이너 내에 있는 오브젝트가 포함되어 있습니다.

오브젝트 응답 엔티티

내용

설명

오브젝트의 컨테이너입니다.

유형

개체

키

설명

오브젝트의 키입니다.

유형

문자열

LastModified

설명

오브젝트의 마지막 업데이트 날짜 및 시간입니다.

유형

날짜

etag

설명

개체의 MD-5 해시입니다. **etag**는 엔티티 태그입니다.

유형

문자열

크기

설명

오브젝트의 크기입니다.

유형

정수

StorageClass

설명

항상 **STANDARD** 를 반환해야 합니다.

유형

문자열

3.4.8. S3 새 버킷 생성

새 버킷을 생성합니다. 버킷을 생성하려면 요청을 인증하기 위해 사용자 **ID**와 유효한 **AWS** 액세스 키 **ID**가 있어야 합니다. 버킷을 익명 사용자로 생성할 수 없습니다.

Constraints

일반적으로 버킷 이름은 도메인 이름 제약 조건을 따라야 합니다.

- 버킷 이름은 고유해야 합니다.
- 버킷 이름은 **IP** 주소로 포맷할 수 없습니다.
- 버킷 이름은 **3~63**자 사이일 수 있습니다.

- 버킷 이름에는 대문자 또는 밑줄이 없어야 합니다.
- 버킷 이름은 소문자 또는 숫자로 시작해야 합니다.
- 버킷 이름에는 대시(-)를 포함할 수 있습니다.
- 버킷 이름은 하나 이상의 라벨이어야 합니다. 인접한 레이블은 마침표(.)로 구분됩니다. 버킷 이름에는 소문자, 숫자 및 하이픈이 포함될 수 있습니다. 각 레이블은 소문자 또는 숫자로 시작하고 끝나야 합니다.



참고

rgw_relaxed_s3_bucket_names 가 **true** 로 설정된 경우 위의 제약조건이 완화됩니다. 버킷 이름은 여전히 고유해야 하며 IP 주소로 포맷할 수 없으며 문자, 숫자, 마침표, 대시, 최대 255자까지 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
```

매개 변수

x-amz-acl

설명

ACL을 연결할 수 있습니다.

유효한 값

private, public-read, public-read-write, authenticated-read

필수 항목

없음

HTTP 응답

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

409

상태 코드

BucketAlreadyExists

설명

버킷은 다른 사용자의 소유권에 이미 존재합니다.

3.4.9. S3 put bucket website

put bucket website API는 **website** 하위 리소스에 지정된 웹 사이트의 구성을 설정합니다. 버킷을 웹 사이트로 구성하기 위해 버킷에 **website** 하위 리소스를 추가할 수 있습니다.



참고

put 작업에는 **S3:PutBucketWebsite** 권한이 필요합니다. 기본적으로 버킷 소유자만 버킷에 연결된 웹 사이트를 구성할 수 있습니다.

구문

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

예제

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

추가 리소스

- 이 API 호출에 대한 자세한 내용은 [S3 API](#) 를 참조하십시오.

3.4.10. S3 버킷 웹 사이트 가져오기

get bucket website API는 **website** 하위 리소스에 지정된 웹 사이트의 구성을 검색합니다.



참고

가져오기 작업에는 **S3:GetBucketWebsite** 권한이 필요합니다. 기본적으로 버킷 소유자만 버킷 웹 사이트 구성을 읽을 수 있습니다.

구문

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

예제

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

추가 리소스

- 이 API 호출에 대한 자세한 내용은 [S3 API](#) 를 참조하십시오.

3.4.11. S3 delete bucket website

Delete bucket website API에서 버킷의 웹 사이트 구성을 제거합니다.

구문

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

예제

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

추가 리소스

- 이 API 호출에 대한 자세한 내용은 [S3 API](#)를 참조하십시오.

3.4.12. 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
```

HTTP 응답

204

상태 코드

콘텐츠 없음

설명

버킷이 제거되었습니다.

3.4.13. 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. 오브젝트가 있어야 하는 일 수 또는 만료 날짜가 소요되며, 이 시점에서 **Ceph Object Gateway**가 오브젝트를 삭제합니다. 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. 여러 가지 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>

```



```

    </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 0개 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 Developer](#)

Guide의 **S3 GET** 버킷 라이프사이클 섹션을 참조하십시오.

- 버킷 라이프사이클 생성에 대한 자세한 내용은 **Red Hat Ceph Storage Developer Guide**의 **S3** 생성 또는 버킷 라이프사이클 섹션을 참조하십시오.
- 버킷 라이프사이클 삭제에 대한 자세한 내용은 **Red Hat Ceph Storage Developer Guide**의 **S3** 삭제 버킷 라이프사이클을 참조하십시오.

3.4.14. 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

일반적인 요청 헤더에 대한 자세한 내용은 **부록 B**의 **S3** 일반 요청 헤더를 참조하십시오.

응답

응답에는 버킷 라이프사이클 및 해당 요소가 포함됩니다.

3.4.15. 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>
```

요청 헤더

content-md5

설명

base64로 인코딩된 메시지의 MD-5 해시

유효한 값

문자열 기본값 또는 제약 조건이 없습니다.**No defaults or constraints.**

필수 항목

없음

추가 리소스

- **Amazon S3 공통 요청 헤더**에 대한 자세한 내용은 **Red Hat Ceph Storage Developer Guide**의 부록 B에 있는 **S3 일반 요청 헤더** 섹션을 참조하십시오.
- **Amazon S3 버킷 라이프사이클**에 대한 자세한 내용은 **Red Hat Ceph Storage Developer Guide**의 **S3 버킷 라이프 사이클** 섹션을 참조하십시오.

3.4.16. 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 공통 요청 헤더에** 대한 자세한 내용은 **Red Hat Ceph Storage Developer Guide**의 부록 B에 있는 **S3 일반 요청 헤더** 섹션을 참조하십시오.
- **Amazon S3 공통 응답 상태 코드에** 대한 자세한 내용은 **Red Hat Ceph Storage Developer Guide**의 부록 C에 있는 **S3 일반 응답 상태 코드** 섹션을 참조하십시오.

3.4.17. 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.

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

구문

```
GET /BUCKET?location HTTP/1.1
```

```
Host: cname.domain.com
```

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET

응답 엔티티

LocationConstraint

설명

버킷이 있는 영역 그룹입니다. 기본 영역 그룹의 빈 문자열입니다.

유형

문자열

3.4.18. S3 버킷 버전 관리

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

아래와 같이 **bucket** 리소스에 **versioning** 하위 리소스를 추가합니다.

구문

GET /BUCKET?versioning HTTP/1.1

Host: cname.domain.com

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET

3.4.19. 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 ID가 부여됩니다. **suspended:** 버킷에 있는 개체에 대한 버전 관리를 비활성화합니다. 버킷에 추가된 모든 오브젝트는 버전 ID null을 수신합니다.

구문

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

예제

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

버킷 요청 엔티티**VersioningConfiguration****설명**

요청에 대한 컨테이너입니다.

유형

컨테이너

상태**설명**

버킷의 **versioning** 상태를 설정합니다. 유효한 값: **Suspended/Enabled**

유형

문자열

3.4.20. 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
```

응답 엔티티

AccessControlPolicy

설명

응답을 위한 컨테이너입니다.

유형

컨테이너

AccessControlList

설명

ACL 정보에 대한 컨테이너입니다.

유형

컨테이너

소유자

설명

버킷 소유자 **ID** 및 **DisplayName** 의 컨테이너입니다.

유형

컨테이너

ID

설명

버킷 소유자의 **ID**입니다.

유형

문자열

DisplayName

설명

버킷 소유자의 표시 이름입니다.

유형

문자열

부여

설명

부여자 및 권한 부여를 위한 컨테이너.

유형

컨테이너

부여

설명

권한 부여를 수신하는 사용자의 **DisplayName** 및 **ID** 에 대한 컨테이너입니다.

유형

컨테이너

권한

설명

Grantee 버킷에 부여된 권한입니다.

유형

문자열

3.4.21. 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
```

요청 엔티티

S3 목록 다중 파트 업로드

AccessControlList

설명

ACL 정보에 대한 컨테이너입니다.

유형

컨테이너

소유자

설명

버킷 소유자 **ID** 및 **DisplayName** 의 컨테이너입니다.

유형

컨테이너

ID

설명

버킷 소유자의 **ID**입니다.

유형

문자열

DisplayName

설명

버킷 소유자의 표시 이름입니다.

유형

문자열

부여

설명

부여자 및 권한 부여를 위한 컨테이너.

유형

컨테이너

부여

설명

권한 부여를 수신하는 사용자의 **DisplayName** 및 **ID** 에 대한 컨테이너입니다.

유형

컨테이너

권한

설명

Grantee 버킷에 부여된 권한입니다.

유형

문자열

3.4.22. 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
```

3.4.23. 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
```

3.4.24. 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
```

3.4.25. 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 에 대한 매개변수를 지정할 수 있지만 필요하지 않습니다.

매개 변수

접두사

설명

지정된 접두사가 포함된 키의 진행 중인 업로드를 반환합니다.

유형

문자열

delimiter

설명

접두사와 나머지 오브젝트 이름 간의 구분 기호입니다.

유형

문자열

key-marker

설명

업로드 목록의 시작 마커입니다.

유형

문자열

max-keys

설명

진행 중인 최대 업로드 수입니다. 기본값은 **1000**입니다.

유형

정수

version-id-marker

설명

목록을 시작할 개체 버전을 지정합니다.

유형

문자열

응답 엔티티

KeyMarker

설명

key-marker request 매개변수로 지정된 키 마커(있는 경우)입니다.

유형

문자열

NextKeyMarker

설명

IsTruncated 가 **true** 인 경우 후속 요청에 사용할 키 마커입니다.

유형

문자열

NextUploadIdMarker

설명

IsTruncated 가 **true** 인 경우 후속 요청에 사용할 업로드 ID 마커입니다.

유형

문자열

IsTruncated

설명

true 인 경우 버킷의 업로드 콘텐츠 서브 세트만 반환됩니다.

유형

부울

크기

설명

업로드된 부분의 크기입니다.

유형

정수

DisplayName

설명

소유자의 표시 이름입니다.

유형

문자열

ID

설명

소유자의 ID입니다.

유형

문자열

소유자

설명

오브젝트를 소유한 사용자의 ID 및 **DisplayName** 에 대한 컨테이너입니다.

유형

컨테이너

StorageClass

설명

결과 오브젝트를 저장하는 데 사용되는 방법입니다. **STANDARD** 또는

REDUCED_REDUNDANCY

유형

문자열

버전

설명

버전 정보의 컨테이너입니다.

유형

컨테이너

versionId

설명

오브젝트의 버전 ID입니다.

유형

문자열

versionIdMarker

설명

잘린 응답의 마지막 키 버전입니다.

유형

문자열

3.4.26. 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
```

3.4.27. 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 필요하지 않습니다.

매개 변수

접두사

설명

지정된 접두사가 포함된 키의 진행 중인 업로드를 반환합니다.

유형

문자열

delimiter

설명

접두사와 나머지 오브젝트 이름 간의 구분 기호입니다.

유형

문자열

key-marker

설명

업로드 목록의 시작 마커입니다.

유형

문자열

max-keys

설명

진행 중인 최대 업로드 수입니다. 기본값은 **1000**입니다.

유형

정수

max-uploads

설명

최대 멀티 파트 업로드 수입니다. 범위는 **1-1000**입니다. 기본값은 **1000**입니다.

유형

정수

version-id-marker

설명

key-marker 가 지정되지 않은 경우 무시됩니다. **ID** 를 사용하거나 다음에 따라 나열할 첫 번째 업로드의 **ID** 를 **lexicographical** 순서로 지정합니다.

유형

문자열

응답 엔티티

ListMultipartUploadsResult

설명

결과에 대한 컨테이너입니다.

유형

컨테이너

ListMultipartUploadsResult.Prefix

설명

접두사 요청 매개 변수로 지정된 접두사 (있는 경우).

유형

문자열

bucket

설명

버킷 콘텐츠를 수신할 버킷입니다.

유형

문자열

KeyMarker

설명

key-marker request 매개변수로 지정된 키 마커(있는 경우)입니다.

유형

문자열

UploadIdMarker

설명

upload-id-marker 요청 매개변수로 지정된 마커(있는 경우)입니다.

유형

문자열

NextKeyMarker

설명

IsTruncated 가 ***true*** 인 경우 후속 요청에 사용할 키 마커입니다.

유형

문자열

NextUploadIdMarker

설명

IsTruncated 가 ***true*** 인 경우 후속 요청에 사용할 업로드 ID 마커입니다.

유형

문자열

MaxUploads

설명

max-uploads 요청 매개변수로 지정된 최대 업로드입니다.

유형

정수

구분자

설명

설정된 경우 접두사가 동일한 오브젝트가 ***CommonPrefixes*** 목록에 표시됩니다.

유형

문자열

IsTruncated

설명

true 인 경우 버킷의 업로드 콘텐츠 서브 세트만 반환됩니다.

유형

부울

업로드**설명**

키, **UploadId**, **InitiatorOwner**, **StorageClass** 및 **Initiated** 요소의 컨테이너입니다.

유형

컨테이너

키**설명**

다중 부분 업로드가 완료되면 오브젝트의 키입니다.

유형

문자열

UploadId**설명**

다중 부분 업로드를 식별하는 ID 입니다.

유형

문자열

이니시에이터**설명**

업로드를 시작한 사용자의 ID 및 **DisplayName** 을 포함합니다.

유형

컨테이너

DisplayName**설명**

이니시에이터의 표시 이름입니다.

유형

문자열

ID

설명

이니시에이터의 **ID**입니다.

유형

문자열

소유자

설명

업로드된 오브젝트를 소유한 사용자의 **ID** 및 **DisplayName**에 대한 컨테이너입니다.

유형

컨테이너

StorageClass

설명

결과 오브젝트를 저장하는 데 사용되는 방법입니다. **STANDARD** 또는 **REDUCED_REDUNDANCY**

유형

문자열

개시됨

설명

사용자가 업로드를 시작한 날짜와 시간입니다.

유형

날짜

CommonPrefixes

설명

여러 오브젝트에 동일한 접두사가 포함된 경우 이 목록에 표시됩니다.

유형

컨테이너

CommonPrefixes.Prefix

설명

접두사 요청 매개 변수에 정의된 접두사 뒤에 있는 키의 하위 문자열입니다.

유형

문자열

3.4.28. 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***



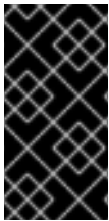
참고

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 Gateway**는 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 Gateway**는 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 Gateway는 다음과 같은 조건 키를 지원합니다.

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

- **aws:username**

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

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

Swift에 미치는 영향

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

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

3.4.29. 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
```

3.4.30. 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
```

요청 엔티티

payer

설명

다운로드 및 요청 요금을 누가 지불하고 요금을 요청하도록 지정합니다.

유형

enum

RequestPaymentConfiguration

설명

Payer 컨테이너입니다.

유형

컨테이너

3.4.31. 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 **Red Hat Ceph Storage Object Gateway** 가이드의 사용자 관리를 참조하십시오. https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/object_gateway_guide/#usr-mgmt-per-tenant-namespace-rgw

3.5. S3 오브젝트 작업

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

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

표 3.3. 오브젝트 작업

기능	상태
오브젝트 가져오기	지원됨
오브젝트 정보 가져오기	지원됨
오브젝트 잠금 해제	지원됨
오브젝트 잠금 가져오기	지원됨
오브젝트 법적 보유 확인	지원됨
오브젝트 법률 보관 받기	지원됨
오브젝트 보존 작업	지원됨
오브젝트 보존 가져오기	지원됨
오브젝트 태그 지정	지원됨
오브젝트 태그 지정 가져오기	지원됨
오브젝트 태그 삭제	지원됨
오브젝트 배치	지원됨
오브젝트 삭제	지원됨
여러 오브젝트 삭제	지원됨
오브젝트 ACL 가져오기	지원됨
오브젝트 ACL 추가	지원됨
오브젝트 복사	지원됨
오브젝트 후	지원됨

기능	상태
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

3.5.1. 사전 요구 사항

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

3.5.2. 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

요청 헤더

범위

설명

검색할 오브젝트의 범위입니다.

유효한 값

Range:bytes=beginbyte-endbyte

필수 항목

없음

if-modified-since

설명

타임스탬프 이후 변경된 경우에만 가져옵니다.**Gets only if modified since the timestamp.**

유효한 값

Timestamp

필수 항목

없음

if-match

설명

ETag 개체가 **ETag**와 일치하는 경우에만 가져옵니다.

유효한 값

엔터티 태그

필수 항목

없음

if-none-match

설명

ETag 개체가 **ETag**와 일치하는 경우에만 가져옵니다.

유효한 값

엔터티 태그

필수 항목

없음

응답 헤더

content-Range

설명

데이터 범위는 범위 헤더 필드가 요청에 지정된 경우에만 반환됩니다.

x-amz-version-id

설명

버전 ID 또는 **null**을 반환합니다.

3.5.3. 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

요청 헤더

범위

설명

검색할 오브젝트의 범위입니다.

유효한 값

Range:bytes=beginbyte-endbyte

필수 항목

없음

if-modified-since

설명

타임스탬프 이후 변경된 경우에만 가져옵니다. **Gets only if modified since the timestamp.**

유효한 값

Timestamp

필수 항목

없음

if-match

설명

ETag 개체가 **ETag**와 일치하는 경우에만 가져옵니다.

유효한 값

엔터티 태그

필수 항목

없음

if-none-match

설명

ETag 개체가 **ETag**와 일치하는 경우에만 가져옵니다.

유효한 값

엔터티 태그

필수 항목

없음

응답 헤더

x-amz-version-id

설명

버전 ID 또는 **null**을 반환합니다.

3.5.4. S3 put 오브젝트 잠금

put 오브젝트 잠금 **API**는 선택한 버킷에 잠금 구성을 배치합니다. 개체 잠금을 사용하면 **WORM(Write-once-read-many)** 모델을 사용하여 개체를 저장할 수 있습니다. 개체 잠금은 고정된 시간 또는 무기한에 대해 개체를 삭제하거나 덮어쓰지 않도록 합니다. 오브젝트 잠금 구성에 지정된 규칙은 기본적으로 선택한 버킷에 배치된 모든 새 오브젝트에 적용됩니다.



중요

버킷을 만들 때 오브젝트 잠금을 활성화합니다. 그렇지 않으면 작업이 실패합니다.

구문

```
PUT /BUCKET?object-lock HTTP/1.1
```

예제

```
PUT /testbucket?object-lock HTTP/1.1
```

요청 엔티티

ObjectLockConfiguration

설명

요청에 대한 컨테이너입니다.

유형

컨테이너

필수 항목

있음

ObjectLockEnabled

설명

이 버킷에 개체 잠금 구성이 활성화되어 있는지 여부를 나타냅니다.

유형

문자열

필수 항목

있음

Rule

설명

지정된 버킷에 대한 오브젝트 잠금 규칙입니다.

유형

컨테이너

필수 항목

없음

DefaultRetention

설명

지정된 버킷에 배치된 새 오브젝트에 적용되는 기본 보존 기간입니다.

유형

컨테이너

필수 항목

없음

mode

설명

기본 오브젝트 잠금 보존 모드입니다. 유효한 값: **GOVERNANCE/COMPLIANCE**.

유형

컨테이너

필수 항목

있음

days

설명

기본 보존 기간에 지정된 일 수입니다.

유형

정수

필수 항목

없음

년

설명

기본 보존 기간에 지정된 연도 수입니다.

유형

정수

필수 항목

없음

HTTP 응답

400

상태 코드

MalformedXML

설명

XML이 제대로 포맷되지 않았습니다.

409

상태 코드

InvalidBucketState

설명

버킷 오브젝트 잠금이 활성화되지 않습니다.

추가 리소스

- 이 **API** 호출에 대한 자세한 내용은 **S3 API** 를 참조하십시오.

3.5.5. S3 개체 잠금을 가져옵니다.

get 오브젝트 잠금 **API**는 버킷의 잠금 구성을 검색합니다.

구문

```
GET /BUCKET?object-lock HTTP/1.1
```

예제

```
GET /testbucket?object-lock HTTP/1.1
```

응답 엔티티

ObjectLockConfiguration

설명

요청에 대한 컨테이너입니다.

유형

컨테이너

필수 항목

있음

ObjectLockEnabled

설명

이 버킷에 개체 잠금 구성이 활성화되어 있는지 여부를 나타냅니다.

유형

문자열

필수 항목

있음

Rule

설명

지정된 버킷에 대해 개체 잠금 규칙이 위치합니다.

유형

컨테이너

필수 항목

없음

DefaultRetention

설명

지정된 버킷에 배치된 새 오브젝트에 적용되는 기본 보존 기간입니다.

유형

컨테이너

필수 항목

없음

mode

설명

기본 오브젝트 잠금 보존 모드입니다. 유효한 값: **GOVERNANCE/COMPLIANCE**.

유형

컨테이너

필수 항목

있음

days

설명

기본 보존 기간에 지정된 일 수입니다.

유형

정수

필수 항목

없음

년

설명

기본 보존 기간에 지정된 연도 수입니다.

유형

정수

필수 항목

없음

추가 리소스

•

이 API 호출에 대한 자세한 내용은 [S3 API](#) 를 참조하십시오.

3.5.6. S3 put object legal hold

put 오브젝트의 법적 보유 API는 선택한 오브젝트에 법적 보유 설정을 적용합니다. 법적인 보유를 갖 추면 오브젝트 버전을 덮어쓰거나 삭제할 수 없습니다. 법적 보관 기간에는 관련 보관 기간이 없으며 명시 적으로 제거할 때까지 그대로 유지됩니다.

구문

```
PUT /BUCKET/OBJECT?legal-hold&versionId= HTTP/1.1
```

예제

```
PUT /testbucket/testobject?legal-hold&versionId= HTTP/1.1
```

versionId 하위 리소스는 특정 버전의 오브젝트를 검색합니다.

요청 엔티티

LegalHold

설명

요청에 대한 컨테이너입니다.

유형

컨테이너

필수 항목

있음

상태**설명**

지정된 개체에 법적 보유가 있는지 여부를 나타냅니다. **Indicates whether the specified object has a legal hold in place. 유효한 값: ON/OFF**

유형**문자열****필수 항목****있음****추가 리소스**

- 이 API 호출에 대한 자세한 내용은 [S3 API](#) 를 참조하십시오.

3.5.7. S3의 개체 법적 유지

get object legal hold API는 오브젝트의 현재 법적 보유 상태를 검색합니다.

구문

```
GET /BUCKET/OBJECT?legal-hold&versionId= HTTP/1.1
```

예제

```
GET /testbucket/testobject?legal-hold&versionId= HTTP/1.1
```

versionId 하위 리소스는 특정 버전의 오브젝트를 검색합니다.

응답 엔티티

LegalHold

설명

요청에 대한 컨테이너입니다.

유형

컨테이너

필수 항목

있음

상태

설명

지정된 개체에 법적 보유가 있는지 여부를 나타냅니다. **Indicates whether the specified object has a legal hold in place.** 유효한 값: **ON/OFF**

유형

문자열

필수 항목

있음

추가 리소스

•

이 **API** 호출에 대한 자세한 내용은 **S3 API** 를 참조하십시오.

3.5.8. S3 put 오브젝트 보존

put 오브젝트 보존 **API**는 오브젝트 보존 구성을 배치합니다. 보존 기간은 고정된 기간 동안 오브젝트 버전을 보호합니다. 거버넌스 모드와 규정 준수 모드라는 두 가지 모드가 있습니다. 이 두 보존 모드는 오브젝트에 서로 다른 수준의 보호 모드를 적용합니다.



참고

이 기간 동안 오브젝트는 **write-once-read-many(WORM protected)**이며 덮어쓰거나 삭제할 수 없습니다.

구문

```
PUT /BUCKET/OBJECT?retention&versionId= HTTP/1.1
```

예제

```
PUT /testbucket/testobject?retention&versionId= HTTP/1.1
```

versionId 하위 리소스는 특정 버전의 오브젝트를 검색합니다.

요청 엔티티

보존

설명

요청에 대한 컨테이너입니다.

유형

컨테이너

필수 항목

있음

mode

설명

지정된 오브젝트의 보존 모드입니다. 유효한 값: **GOVERNANCE/COMPLIANCE**

유형

문자열

필수 항목

있음

RetainUntilDate

설명

보존 날짜. 형식: **2020-01-05T00:00:00.000Z**

유형

Timestamp

필수 항목

있음

추가 리소스

- 이 API 호출에 대한 자세한 내용은 [S3 API](#) 를 참조하십시오.

3.5.9. S3 가져오기 오브젝트 보존

get 오브젝트 보존 API는 오브젝트에서 오브젝트 보존 구성을 검색합니다.

구문

GET /BUCKET/OBJECT?retention&versionId= HTTP/1.1

예제

`GET /testbucket/testobject?retention&versionId= HTTP/1.1`

versionId 하위 리소스는 특정 버전의 오브젝트를 검색합니다.

응답 엔티티

보존

설명

요청에 대한 컨테이너입니다.

유형

컨테이너

필수 항목

있음

mode

설명

지정된 오브젝트의 보존 모드입니다. 유효한 값: **GOVERNANCE/COMPLIANCE**

유형

문자열

필수 항목

있음

RetainUntilDate

설명

보존 날짜. 형식: **2020-01-05T00:00:00.000Z**

유형

Timestamp

필수 항목

있음

추가 리소스

- 이 API 호출에 대한 자세한 내용은 [S3 API](#) 를 참조하십시오.

3.5.10. S3 put object tagging

put 오브젝트 태그 API는 태그와 오브젝트를 연결합니다. 태그는 키-값 쌍입니다. 다른 버전의 태그를 배치하려면 **versionId** 쿼리 매개 변수를 사용합니다. **s3:PutObjectTagging** 작업을 수행할 수 있는 권한이 있어야 합니다. 기본적으로 버킷 소유자는 이 권한을 가지며 다른 사용자에게 이 권한을 부여할 수 있습니다.

구문

```
PUT /BUCKET/OBJECT?tagging&versionId= HTTP/1.1
```

예제

```
PUT /testbucket/testobject?tagging&versionId= HTTP/1.1
```

요청 엔티티

태그

설명

요청에 대한 컨테이너입니다.

유형

컨테이너

필수 항목

있음

tagset

설명

태그 집합 컬렉션을 나타냅니다. **Represents a collection of tags.**

유형

문자열

필수 항목

있음

추가 리소스

-

이 **API** 호출에 대한 자세한 내용은 [S3 API](#) 를 참조하십시오.

3.5.11. S3 개체 태그 가져오기

get 오브젝트 태그 **API**는 오브젝트의 태그를 반환합니다. 기본적으로 **GET** 작업은 오브젝트의 현재 버전에 대한 정보를 반환합니다.



참고

버전이 지정된 버킷의 경우 버킷에 여러 버전의 오브젝트를 사용할 수 있습니다. 다른 버전의 태그를 검색하려면 요청에 **versionId** 쿼리 매개변수를 추가합니다.

구문

GET /BUCKET/OBJECT?tagging&versionId= HTTP/1.1

예제

```
GET /testbucket/testobject?tagging&versionId= HTTP/1.1
```

추가 리소스

- 이 API 호출에 대한 자세한 내용은 [S3 API](#) 를 참조하십시오.

3.5.12. S3 delete 오브젝트 태그

삭제 오브젝트 태그 API는 지정된 오브젝트에서 전체 태그 집합을 제거합니다. 이 작업을 사용하려면 **s3:DeleteObjectTagging** 작업을 수행할 수 있는 권한이 있어야 합니다.



참고

특정 오브젝트 버전의 태그를 삭제하려면 요청에 **versionId** 쿼리 매개변수를 추가합니다.

구문

```
DELETE /BUCKET/OBJECT?tagging&versionId= HTTP/1.1
```

예제

```
DELETE /testbucket/testobject?tagging&versionId= HTTP/1.1
```

추가 리소스

- 이 API 호출에 대한 자세한 내용은 [S3 API](#) 를 참조하십시오.

3.5.13. S3 버킷에 오브젝트 추가

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

구문

PUT /BUCKET/OBJECT HTTP/1.1

요청 헤더

content-md5

설명

base64로 인코딩된 메시지의 **MD-5** 해시입니다.

유효한 값

문자열입니다. 기본값 또는 제약 조건이 없습니다.

필수 항목

없음

content-type

설명

표준 **MIME** 유형입니다.

유효한 값

모든 **MIME** 유형. 기본값: **binary/octet-stream**.

필수 항목

없음

x-amz-meta-<...>*

설명

사용자 메타데이터. 오브젝트로 저장됩니다.

유효한 값

8kb 까지의 문자열입니다. 기본값 없음.

필수 항목

없음

x-amz-acl

설명

신뢰할 수 있는 **ACL**.

유효한 값

private, public-read, public-read-write, authenticated-read

필수 항목

없음

응답 헤더

x-amz-version-id

설명

버전 ID 또는 **null**을 반환합니다.

3.5.14. 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
```

3.5.15. S3 여러 오브젝트 삭제

이 **API** 호출은 버킷에서 여러 오브젝트를 삭제합니다.

구문

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

3.5.16. 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
```

응답 헤더

x-amz-version-id

설명

버전 ID 또는 **null**을 반환합니다.

응답 엔티티

AccessControlPolicy

설명

응답을 위한 컨테이너입니다.

유형

컨테이너

AccessControlList

설명

ACL 정보에 대한 컨테이너입니다.

유형

컨테이너

소유자

설명

버킷 소유자 **ID** 및 **DisplayName** 의 컨테이너입니다.

유형

컨테이너

ID**설명**

버킷 소유자의 **ID**입니다.

유형

문자열

DisplayName**설명**

버킷 소유자의 표시 이름입니다.

유형

문자열

부여**설명**

부여자 및 권한 부여를 위한 컨테이너.

유형

컨테이너

부여**설명**

권한 부여를 수신하는 사용자의 **DisplayName** 및 **ID** 에 대한 컨테이너입니다.

유형

컨테이너

권한

설명

Grantee 버킷에 부여된 권한입니다.

유형

문자열

3.5.17. S3에서 개체의 ACL(액세스 제어 목록) 설정

현재 오브젝트 버전에 대한 오브젝트 **ACL**을 설정합니다.

구문

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

요청 엔티티

AccessControlPolicy

설명

응답을 위한 컨테이너입니다.

유형

컨테이너

AccessControlList

설명

ACL 정보에 대한 컨테이너입니다.

유형

컨테이너

소유자

설명

버킷 소유자 **ID** 및 **DisplayName** 의 컨테이너입니다.

유형

컨테이너

ID**설명**

버킷 소유자의 **ID**입니다.

유형

문자열

DisplayName**설명**

버킷 소유자의 표시 이름입니다.

유형

문자열

부여**설명**

부여자 및 권한 부여를 위한 컨테이너.

유형

컨테이너

부여**설명**

권한 부여를 수신하는 사용자의 **DisplayName** 및 **ID** 에 대한 컨테이너입니다.

유형

컨테이너

권한**설명**

Grantee 버킷에 부여된 권한입니다.

유형**문자열****3.5.18. 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
```

요청 헤더**x-amz-copy-source****설명**

소스 버킷 이름 + 오브젝트 이름입니다.

유효한 값

BUCKET/OBJECT

필수 항목

있음

x-amz-acl**설명**

신뢰할 수 있는 **ACL**.

유효한 값

private, public-read, public-read-write, authenticated-read

필수 항목

없음

x-amz-copy-if-modified-since

설명

타임스탬프 이후 변경된 경우에만 복사합니다.

유효한 값

Timestamp

필수 항목

없음

x-amz-copy-if-unmodified-since

설명

타임스탬프 이후 수정되지 않은 경우에만 복사합니다.

유효한 값

Timestamp

필수 항목

없음

x-amz-copy-if-match

설명

오브젝트 **ETag**가 **ETag**와 일치하는 경우에만 복사합니다.

유효한 값

엔터티 태그

필수 항목

없음

x-amz-copy-if-none-match

설명

오브젝트 **ETag**가 **ETag**와 일치하는 경우에만 복사합니다.

유효한 값

엔티티 태그

필수 항목

없음

응답 엔티티

CopyObjectResult

설명

응답 요소에 대한 컨테이너입니다.

유형

컨테이너

LastModified

설명

소스 오브젝트의 마지막 수정 날짜입니다.

유형

날짜

etag

설명

새 오브젝트의 **ETag**입니다.

유형

문자열

3.5.19. 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
```

3.5.20. 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
```

3.5.21. 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
```

요청 헤더

content-md5

설명

base64로 인코딩된 메시지의 **MD-5** 해시입니다.

유효한 값

문자열입니다. 기본값 또는 제약 조건이 없습니다.

필수 항목

없음

content-type

설명

표준 **MIME** 유형입니다.

유효한 값

모든 **MIME** 유형. 기본값: **binary/octet-stream**

필수 항목

없음

x-amz-meta-<...>

설명

사용자 메타데이터. 오브젝트로 저장됩니다.

유효한 값

8kb 까지의 문자열입니다. 기본값 없음.

필수 항목

없음

x-amz-acl

설명

신뢰할 수 있는 **ACL**.

유효한 값

private, public-read, public-read-write, authenticated-read

필수 항목

없음

응답 엔티티***InitiatedMultipartUploadsResult*****설명**

결과에 대한 컨테이너입니다.

유형

컨테이너

bucket**설명**

오브젝트 콘텐츠를 수신할 버킷입니다.

유형

문자열

키**설명**

키 요청 매개변수로 지정한 키 (있는 경우)입니다.

유형

문자열

UploadId**설명**

다중 부분 업로드를 식별하는 **upload-id** 요청 매개변수로 지정된 **ID**입니다(있는 경우).

유형

문자열

3.5.22. 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.

HTTP 응답

404

상태 코드

NoSuchUpload

설명

지정된 **upload-id**는 이 오브젝트의 초기 업로드와 일치하지 않습니다.

3.5.23. 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
```

응답 엔티티

InitiatedMultipartUploadsResult

설명

결과에 대한 컨테이너입니다.

유형

컨테이너

bucket

설명

오브젝트 콘텐츠를 수신할 버킷입니다.

유형

문자열

키

설명

키 요청 매개변수로 지정한 키 (있는 경우)입니다.

유형

문자열

UploadId

설명

다중 부분 업로드를 식별하는 **upload-id** 요청 매개변수로 지정된 ID입니다(있는 경우).

유형

문자열

이니시에이터

설명

업로드를 시작한 사용자의 **ID** 및 **DisplayName** 을 포함합니다.

유형

컨테이너

ID

설명

이니시에이터의 **ID**입니다.

유형

문자열

DisplayName

설명

이니시에이터의 표시 이름입니다.

유형

문자열

소유자

설명

업로드된 오브젝트를 소유한 사용자의 **ID** 및 **DisplayName** 에 대한 컨테이너입니다.

유형

컨테이너

StorageClass

설명

결과 오브젝트를 저장하는 데 사용되는 방법입니다. **STANDARD** 또는

REDUCED_REDUNDANCY

유형

문자열

PartNumberMarker

설명

IsTruncated 가 **true** 인 경우 후속 요청에 사용할 부분 마커입니다. 목록을 전제로 했습니다.

유형

문자열

NextPartNumberMarker

설명

IsTruncated 가 **true** 인 경우 후속 요청에 사용할 다음 부분 마커입니다. 목록의 끝입니다.

유형

문자열

IsTruncated

설명

true 인 경우 오브젝트 업로드 콘텐츠의 하위 집합만 반환됩니다.

유형

부울

part

설명

키, **Part**, **InitiatorOwner**, **StorageClass** 및 **Initiated** 요소의 컨테이너입니다.

유형

컨테이너

PartNumber

설명

키, **Part**, **InitiatorOwner**, **StorageClass** 및 **Initiated** 요소의 컨테이너입니다.

유형

정수

etag

설명

파트의 엔터티 태그입니다.

유형

문자열

크기

설명

업로드된 부분의 크기입니다.

유형

정수

3.5.24. S3 업로드된 부분 조립

업로드된 부분을 어셈블하고 새 오브젝트를 생성하여 다중 부분 업로드를 완료합니다.

다중 파트 업로드를 완료하려면 **uploadId** 하위 리소스 및 업로드 **ID**를 지정합니다.

구문

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

요청 엔터티

CompleteMultipartUpload

설명

하나 이상의 부분으로 구성된 컨테이너입니다.

유형

컨테이너

필수 항목

있음

part

설명

PartNumber 및 **ETag** 에 대한 컨테이너입니다.

유형

컨테이너

필수 항목

있음

PartNumber

설명

부분의 식별자입니다.

유형

정수

필수 항목

있음

etag

설명

파트의 엔터티 태그입니다.

유형

문자열

필수 항목

있음

응답 엔티티

CompleteMultipartUploadResult

설명

응답을 위한 컨테이너입니다.

유형

컨테이너

위치

설명

새 오브젝트의 리소스 식별자(path)입니다.

유형

URI

bucket

설명

새 오브젝트가 포함된 버킷의 이름입니다.

유형

문자열

키

설명

오브젝트의 키입니다.

유형

문자열

etag

설명

새 오브젝트의 **entity** 태그입니다.

유형

문자열

3.5.25. 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
```

요청 헤더

x-amz-copy-source

설명

소스 버킷 이름 및 오브젝트 이름입니다.

유효한 값

BUCKET/OBJECT

필수 항목

있음

x-amz-copy-source-range

설명

소스 오브젝트에서 복사할 바이트 범위입니다.

유효한 값

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.

필수 항목

없음

응답 엔티티

CopyPartResult

설명

모든 응답 요소에 대한 컨테이너입니다.

유형

컨테이너

etag

설명

새 부분의 **ETag**를 반환합니다.

유형

문자열

LastModified

설명

파트가 마지막으로 수정된 날짜를 반환합니다.

유형

문자열

추가 리소스

- 이 기능에 대한 자세한 내용은 [Amazon S3 사이트](#)를 참조하십시오.

3.5.26. 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
```

3.5.27. S3 Hadoop 상호 운용성

HDFS(Haop Distributed File System) 액세스가 필요한 데이터 분석 애플리케이션의 경우 **Hadoop용 Apache S3A** 커넥터를 사용하여 **Ceph Object Gateway**에 액세스할 수 있습니다. **S3A** 커넥터는 데이터를 **Ceph Object Gateway**에 저장되는 동안 **HDFS** 파일 시스템과 **HDFS** 파일 시스템으로 애플리케이션에 의미 있는 의미를 읽고 쓰는 오픈 소스 툴입니다.

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

3.5.28. 추가 리소스

- 멀티 테넌시에 대한 자세한 내용은 [Red Hat Ceph Storage Object Gateway 가이드](#)를 참조하십시오.

3.6. S3 선택 작업 (기술 프리뷰)

개발자는 **Spark-SQL**과 같은 고급 분석 애플리케이션에 **S3 select API**를 사용하여 대기 시간과 처리량을 향상시킬 수 있습니다. 예를 들어 여러 기가바이트의 데이터가 있는 **CSV S3** 개체는 다음 쿼리를 사용하여 다른 열에 의해 필터링된 단일 열을 추출할 수 있습니다.

예제

```
select customerid from s3Object where age>30 and age<65;
```

현재 **S3** 오브젝트는 데이터를 필터링 및 추출하기 전에 **Ceph OSD**에서 **Ceph Object Gateway**를 통해 데이터를 검색해야 합니다. 개체가 클 때 성능이 향상되고 쿼리가 더 구체적입니다.

3.6.1. 사전 요구 사항

- 실행 중인 **Red Hat Ceph Storage** 클러스터.
- **RESTful** 클라이언트.
- 사용자 액세스 권한으로 생성된 **S3** 사용자.

3.6.2. S3 오브젝트에서 콘텐츠 선택

선택한 오브젝트 콘텐츠 **API**는 구조화된 쿼리 언어(**SQL**)를 통해 오브젝트 콘텐츠를 필터링합니다. 요청에서는 지정된 콘텐츠를 검색하려면 **data serialization** 형식을, 쉼표로 구분된 값(**CSV**)으로 지정해야 합니다. **Amazon Web Services(AWS) CLI(명령줄 인터페이스)** **Select object content uses the CSV format to parse object data into records and returns only the records specified in the query.**



참고

응답에 대한 데이터 직렬화 형식을 지정해야 합니다. 이 작업에 대한 **s3:GetObject** 권한이 있어야 합니다.

구문

```
POST /BUCKET/KEY?select&select-type=2 HTTP/1.1\r\n
```

예제

```
POST /testbucket/sample1csv?select&select-type=2 HTTP/1.1\r\n
```

요청 엔티티

bucket

설명

오브젝트 콘텐츠를 선택할 버킷입니다.

유형

문자열

필수 항목

있음

키

설명

오브젝트 키입니다.

길이 제한

최소 길이 1입니다.

유형

문자열

필수 항목

있음

SelectObjectContentRequest

설명

선택한 오브젝트 콘텐츠 요청 매개변수의 루트 수준 태그입니다.

유형

문자열

필수 항목

있음

expression

설명

개체를 쿼리하는 데 사용되는 식입니다.

유형

문자열

필수 항목

있음

ExpressionType

설명

제공된 표현식의 유형(예: **SQL**).

유형

문자열

유효한 값

SQL

필수 항목

있음

InputSerialization

설명

쿼리 중인 개체의 데이터 형식에 대해 설명합니다.**Describes the format of the data in the object that is being queried.**

유형

문자열

필수 항목

있음

OutputSerialization

설명

섬표 구분 기호 및 줄 바꿈으로 반환되는 데이터 형식입니다.

유형

문자열

필수 항목

있음

응답 엔티티

작업이 성공하면 서비스에서 **HTTP 200** 응답을 다시 보냅니다. 서비스는 **XML** 형식으로 반환됩니다.**Data is returned in XML format by the service:**

페이로드

설명

페이로드 매개변수에 대한 루트 수준 태그입니다.

유형

문자열

필수 항목

있음

레코드

설명

records 이벤트입니다.

유형

base64로 인코딩된 바이너리 데이터 오브젝트

필수 항목

없음

통계

설명

통계 이벤트입니다.

유형

long

필수 항목

없음

Ceph Object Gateway에서는 다음 응답을 지원합니다.

예제

```
{:event-type,records} {:content-type,application/octet-stream} :message-type,event}
```

구문

```
aws --endpoint-url http://localhost:80 s3api select-object-content
--bucket BUCKET_NAME
--expression-type 'SQL'
--input-serialization
'{"CSV": {"FieldDelimiter": ";", "QuoteCharacter": "\"", "RecordDelimiter": "\n",
"QuoteEscapeCharacter": "\\", "FileHeaderInfo": "USE"}, "CompressionType": "NONE"}'
--output-serialization '{"CSV": {}}'
--key OBJECT_NAME
--expression "select count(0) from s3object where int(_1)<10;" output.csv
```

예제

```
aws --endpoint-url http://localhost:80 s3api select-object-content
--bucket testbucket
--expression-type 'SQL'
--input-serialization
'{"CSV": {"FieldDelimiter": ";", "QuoteCharacter": "\"", "RecordDelimiter": "\n",
"QuoteEscapeCharacter": "\\", "FileHeaderInfo": "USE"}, "CompressionType": "NONE"}'
--output-serialization '{"CSV": {}}'
--key testobject
--expression "select count(0) from s3object where int(_1)<10;" output.csv
```

지원되는 기능

현재 **AWS s3 select** 명령의 일부만 지원됩니다.

기능	세부 정보	설명	예제
산술 연산자	$^ * \% / + - ()$		s3object에서 $(int(_1)+int(_2)) * int(_9)$ 를 선택합니다.
산술 연산자	% modulo		s3object에서 count(*)를 선택합니다. 여기서 $cast(_1 as int) \% 2 == 0$;
산술 연산자	power-of		s3object에서 $cast(2^10 as int)$ 를 선택합니다.

기능	세부 정보	설명	예제
Operator 비교	> < >= <= == !=		s3object에서 <code>_1,_2</code> 를 선택합니다. 여기서 <code>(int(_1)+int(_3))>int(_5)</code>
논리 연산자	및 없음		그렇지 않은 경우 s3object에서 <code>count(*)</code> 을 선택합니다(<code>int(1)>123</code> 및 <code>int(_5)<200</code>).
논리 연산자	null	표현식에서 null 표시를 위해 <code>true/false</code> 를 반환합니다.	
논리 연산자 및 null	null이 아님	표현식에서 null 표시를 위해 <code>true/false</code> 를 반환합니다.	
논리 연산자 및 null	알 수 없는 상태	null을 검토하고 NULL을 사용한 논리적 작업의 결과를 관찰합니다. 쿼리가 0 을 반환합니다.	s3object에서 null 및 (3>2)를 선택합니다.
NULL이 있는 산술 연산자	알 수 없는 상태	null을 검토하고 NULL을 사용하여 바이너리 작업의 결과를 관찰합니다. 쿼리가 0 을 반환합니다.	s3object에서 count(*)를 선택합니다. 여기서 (null+1) 및 (3>2).
NULL과 비교	알 수 없는 상태	null을 검토하고 NULL과 비교 작업의 결과를 관찰합니다. 쿼리가 0 을 반환합니다.	s3object에서 (null*1.5) != 3을 선택합니다.
누락된 열	알 수 없는 상태		s3object에서 count(*)를 선택합니다. 여기서 _1은 null입니다.
프로젝션 열	if 또는 그 후와 동일합니다.	케이스 선택	(1+1==(2+1)*3인 경우, 경우 'case_1' 4*3)==(12 그런 다음 'case_else' 끝, s3object의 age*2
논리 연산자		coalesce 첫 번째 null이 아닌 인수를 반환합니다.	s3object에서 coalesce(nullesce(nullif(5,5), nullif(1,1.0),age+12)를 선택합니다.
논리 연산자		두 인수가 모두 동일한 경우 nullif(1,1)=NULL nullif(null,1)=NULL nullif(2,1)=2 의 경우 NULL을 반환합니다.	s3object에서 nullif(cast(_1 int),cast(_2 as int))를 선택합니다.

기능	세부 정보	설명	예제
논리 연산자		{expression} in (.. {expression} ..)	s3object 에서 'ben' in(trim(_5),substring(_1,char_length(_1)-3,3),last_name) 에서 count(*) 를 선택합니다.
논리 연산자		{expression}과 {expression} 사이의 {expression}	stdin 에서 "x"와 trim(_1) 과 substring(_3,char_length(_3)-1,1) 사이에서 부분 문자열 (_3,char_length(_3),1)을 == ":";
논리 연산자		{expression} like {match-pattern}	s3object() 를 선택합니다. 여기서 first_name like '%de_' ; 선택 개수()는 s3object 에서 _1 을 "%a[r-s]"와 같이 선택합니다.
캐스팅 연산자			s3object 에서 cast(123)%2 를 선택합니다.
캐스팅 연산자			s3object 에서 cast(123.456을 float)%2 를 선택합니다.
캐스팅 연산자			s3object 에서 cast('BC0-9'),cast(substr('ab12cd',3,2) 를 int 로 선택합니다.
캐스팅 연산자			s3object 에서 cast(substring) (2007-01-01',12,10 에서 게시)를 선택합니다.
AWS가 아닌 Operator			s3object 에서 int(_1),int(1.2 + 3.4) 를 선택합니다.
AWS가 아닌 Operator			s3object 에서 float(1.2) 를 선택합니다.
AWS가 아닌 Operator			s3object 에서 timestamp('1999:10:10:23:44') 를 선택합니다.
집계 함수	일요일		s3object 에서 sum(int(_1)) 을 선택합니다.

기능	세부 정보	설명	예제
집계 함수	avg		s3object 에서 avg(cast(cast(_1 a float)) + cast(_2 as int)) 를 선택합니다.
집계 함수	분		s3object 에서 avg(cast(cast(_1 a float)) + cast(_2 as int)) 를 선택합니다.
집계 함수	max		s3object 에서 max(float(_1)),min(int(_5)) 을 선택합니다.
집계 함수	count		s3object 에서 count(*) 를 선택합니다. 여기서 (int(1)+int(_3))>int(_5)
타임 스탬프 함수	extract		s3object 에서 (*) 를 선택하여 extract('year',timestamp(_2)) > 1950 및 extract('year',timestamp(_1)) < 1960;
타임 스탬프 함수	DateAdd		s3object 에서 datediff('year',timestamp(_1),dateadd('day',366,timestamp(_1))) == 1;
타임 스탬프 함수	DateDiff		s3object 에서 datediff('month',timestamp(_1),timestamp(_2)) == 2 를 선택합니다.
타임 스탬프 함수	utcnow		s3object 에서 datediff('hours',utcnow(),dateadd('day',1,utcnow())) == 24 를 선택합니다.
문자열 기능	substring		s3개체 에서 count(0) 를 선택합니다. 여기서 int(substring(_1,1,4))>1950 및 int(substring(_1,1,4))<1960;
문자열 기능	trim		s3object 에서 trim(' foobar ') 을 선택합니다.
문자열 기능	trim		s3object 에서 trim(foobar ') 에서 트림(trailing)을 선택합니다.

기능	세부 정보	설명	예제
문자열 기능	trim		s3object에서 'foobar')에서 trim(leading)을 선택합니다.
문자열 기능	trim		s3objects에서 trim('12211foobar22211122')을 선택합니다.
문자열 기능	낮음 또는 상한		s3objects에서 trim('12211foobar22211122')을 선택합니다.
문자열 기능	char_length, character_length		s3object에서 char_length(_3)=3;을 선택합니다.
복잡한 쿼리			s3object에서 sum(cast(_1 as int)),max(cast(cast(_3) as int)), 부분 문자열 ('abcdefghijklm', (2-1)*3sum(cast(_1 as int))/sum(cast(_1 as int)) (count() + count(0))을 선택합니다.
별칭 지원			a3object where a3>100 및 a3<300;에서 a3object에서 a3으로 a1, int(_2)를 a2로 선택합니다.

추가 리소스



자세한 내용은 [Amazon의 S3 Select Object Content API](#)를 참조하십시오.

3.6.3. S3 지원 일부 기능

S3 select는 다음과 같은 기능을 지원합니다: **.Timestamp**

timestamp(string)

설명

문자열을 기본 유형의 타임스탬프로 변환합니다.

지원됨

현재 다음과 같이 변환합니다: **yyyy:mm:dd hh:mi:dd**

extract(date-part,timestamp)

설명

입력 타임스탬프에서 날짜 부분 추출에 따른 정수를 반환합니다.

지원됨

날짜 부분: **year,month,week,day**.

dateadd(date-part ,integer,timestamp)

설명

입력 타임스탬프 및 **date-part** 결과를 기반으로 한 타임스탬프를 반환합니다.

지원됨

date-part : **year,month,day**.

datediff(date-part,timestamp,timestamp)

설명

date-part에 따라 두 타임스탬프 사이의 차이로 계산된 정수를 반환합니다.

지원됨

Date-part : **year,month,day,hours**.

utcnow()

설명

현재 시간의 타임스탬프를 반환합니다.

집계

count()

설명

조건인 경우 조건과 일치하는 행 수에 따라 정수를 반환합니다.

`sum(expression)`**설명**

조건(있는 경우)과 일치하는 각 행에 대한 요약을 반환합니다.

`avg(expression)`**설명**

조건이 있는 경우 조건과 일치하는 각 행에서 평균 표현식을 반환합니다.

`max(expression)`**설명**

조건이 있는 경우 조건과 일치하는 모든 표현식에 대해 최대 결과를 반환합니다.

`min(expression)`**설명**

조건이 있는 경우 조건과 일치하는 모든 표현식에 대해 최소 결과를 반환합니다.

문자열**`substring(string,from,to)`****설명**

에서 및 입력에 따라 입력 문자열에서 추출을 반환합니다.

`Char_length`**설명**

문자열에서 문자 수를 반환합니다.**Returns a number of characters in string.**
characters_length도 동일한 작업을 수행합니다.

`Trim`**설명**

대상 문자열에서 선행 또는 후행 문자를 트리밍하고 기본값은 공백 문자입니다.

`upper/lower`**설명**

문자를 대문자 또는 소문자로 변환합니다.

NULL

NULL 은 산술 연산에 대한 값을 생성할 수 없는 경우 **NULL**이 누락되거나 알 수 없는 값입니다.*The NULL value is missing or unknown that is NULL can not produce a value on any arithmetic operations.* 동일한 산술 비교에 적용되며, **NULL** 에 대한 모든 비교는 알 수 없는 **NULL** 입니다.

표 3.4. NULL 사용 사례

A는 null입니다.	Result(NULL=UNKNOWN)
A 없음	NULL
a 또는 False	NULL
A 또는 True	True
A 또는 A	NULL
A 및 False	False
A 및 True	NULL
A 및 A	NULL

추가 리소스

- 자세한 내용은 [Amazon의 S3 Select Object Content API](#) 를 참조하십시오.

3.6.4. S3 별칭 프로그래밍 구성

별칭 프로그래밍 구문은 많은 열 또는 복잡한 쿼리가 포함된 개체를 더 잘 프로그래밍할 수 있으므로 **s3** 선택 언어의 필수 부분입니다.*Alias programming construct is an essential part of the s3 select language because it enables better programming with objects that contain many columns or complex queries.* 별칭 구문을 사용하여 문을 구문 분석할 때 별칭을 올바른 프로젝트션 열 및 쿼리 실행에 대한 참조로 교체하고 다른 식처럼 참조를 평가합니다. **alias**는 별칭을 두 번 이상 사용하는 경우 캐시의 결과가 사용되므로 동일한 결과가 계산되지 않고 동일한 결과가 반환됩니다. 현재 **Red Hat**은 열 별칭을 지원합니다.

예제

```
select int(_1) as a1, int(_2) as a2 , (a1+a2) as a3 from s3object where a3>100 and a3<300;")
```

3.6.5. S3 CSV 구문 분석 설명

기본값이 있는 입력 **serialization**을 사용하여 **CSV** 정의를 정의할 수 있습니다.

- **row-delimiter**에 `{\n}` 을 사용합니다.
- 따옴표로 `{"}` 를 사용합니다.
- 이스케이프 문자에는 `{\}` 를 사용합니다.

csv-header-info 가 구문 분석되고, 이는 스키마를 포함하는 입력 오브젝트의 첫 번째 행입니다. 현재 **output serialization** 및 **compression-type**은 지원되지 않습니다. **S3** 선택 엔진에는 **S3-objects**를 구문 분석하는 **CSV** 구문 분석기가 있습니다.

- 각 행은 **row-delimiter**로 끝납니다.
- 필드 분리기는 인접한 열을 구분합니다. *The field-separator separates the adjacent columns.*
- 연속 필드 구분자는 **NULL** 열을 정의합니다.
- **quote-character**는 **filed** 구분자인 **field-separator**를 덮어씁니다.
- 이스케이프 문자는 행 구분자를 제외한 특수 문자를 비활성화합니다.

다음은 **CSV** 구문 분석 규칙의 예입니다.

표 3.5. CSV 구문 분석

기능	설명	입력 (토큰)
NULL	연속 필드 구분자	„1,,2, => {null}{null}{1}{null}{2}{null}
QUOTE	인용 문자 덮어쓰기 필드 구분자입니다.	11,22,"a,b,c,d",last => {11}{22}{“a,b,c,d”}{last}
이스케이프	이스케이프 문자는 메타 문자를 덮어씁니다.	오브젝트 소유자 ID 및 DisplayName의 컨테이너
행 구분	폐쇄된 인용구가 없으며 행 구분 기호가 닫는 라인입니다.	11,22,a="str,44,55,66 = EgressIP {11}{22}{a="str,44,55,66}
CSV 헤더 정보	FileHeaderInfo tag	USE 값은 첫 번째 줄의 각 토큰이 column-name, IGNORE 값은 첫 번째 줄을 건너뛰는 것을 의미합니다.

추가 리소스

- 자세한 내용은 [Amazon의 S3 Select Object Content API](#) 를 참조하십시오.

3.7. 추가 리소스

- **Amazon S3 일반 요청 헤더**는 [부록 B. S3 일반 요청 헤더](#) 을 참조하십시오.
- **Amazon S3 일반적인 응답 상태 코드**는 [부록 C. S3 일반적인 응답 상태 코드](#) 를 참조하십시오.
- **지원되지 않는 헤더 필드**는 [부록 D. S3 지원되지 않는 헤더 필드](#) 를 참조하십시오.

4장. 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 4.1. 기능

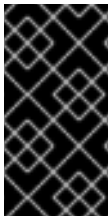
기능	상태	remarks
인증	지원됨	
계정 메타데이터 가져오기	지원됨	사용자 정의 메타데이터 없음
Swift ACL	지원됨	Swift ACL의 하위 집합 지원
컨테이너 나열	지원됨	
컨테이너의 오브젝트 나열	지원됨	
컨테이너 생성	지원됨	
컨테이너 삭제	지원됨	
컨테이너 메타데이터 가져오기	지원됨	
컨테이너 메타데이터 추가/업데이트	지원됨	
컨테이너 메타데이터 삭제	지원됨	
오브젝트 가져오기	지원됨	
오브젝트 생성/업데이트	지원됨	
대규모 오브젝트 생성	지원됨	
오브젝트 삭제	지원됨	
오브젝트 복사	지원됨	
오브젝트 메타데이터 가져오기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

4.1. 사전 요구 사항

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

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

4.3. SWIFT 사용자 만들기

Swift 인터페이스를 테스트하려면 **Swift** 하위 사용자를 만듭니다. **Swift** 사용자 생성은 2단계 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@host01 ~]# radosgw-admin subuser create --uid=testuser --subuser=testuser:swift --
access=full
{
  "user_id": "testuser",
  "display_name": "First User",
  "email": "",
  "suspended": 0,
  "max_buckets": 1000,
  "auid": 0,
  "subusers": [
    {
      "id": "testuser:swift",
      "permissions": "full-control"
```

```

    }
  ],
  "keys": [
    {
      "user": "testuser",
      "access_key": "O8JDE41XMI74O185EHKD",
      "secret_key": "i4Au2yxG5wtr1JK01mI8kjJPM93HNAoVWOSTdJd6"
    }
  ],
  "swift_keys": [
    {
      "user": "testuser:swift",
      "secret_key": "13TLtdEW7bCqgttQgPzxFxziu0AgabtOc6vM8DLA"
    }
  ],
  "caps": [],
  "op_mask": "read, write, delete",
  "default_placement": "",
  "placement_tags": [],
  "bucket_quota": {
    "enabled": false,
    "check_on_raw": false,
    "max_size": -1,
    "max_size_kb": 0,
    "max_objects": -1
  },
  "user_quota": {
    "enabled": false,
    "check_on_raw": false,
    "max_size": -1,
    "max_size_kb": 0,
    "max_objects": -1
  },
  "temp_url_keys": [],
  "type": "rgw"
}

```

2.

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@host01 ~]# radosgw-admin key create --subuser=testuser:swift --key-type=swift --gen-secret
{
  "user_id": "testuser",
  "display_name": "First User",
  "email": "",
  "suspended": 0,
  "max_buckets": 1000,
  "auid": 0,
  "subusers": [
    {
      "id": "testuser:swift",
      "permissions": "full-control"
    }
  ],
  "keys": [
    {
      "user": "testuser",
      "access_key": "O8JDE41XMI74O185EHKD",
      "secret_key": "i4Au2yxG5wtr1JK01mI8kjJPM93HNAoVWOSTdJd6"
    }
  ],
  "swift_keys": [
    {
      "user": "testuser:swift",
      "secret_key": "a4ioT4jEP653CDcdU8p4OuhruwABBRZmyNUbnSSt"
    }
  ],
  "caps": [],
  "op_mask": "read, write, delete",
  "default_placement": "",
  "placement_tags": [],
  "bucket_quota": {
    "enabled": false,
    "check_on_raw": false,
    "max_size": -1,
    "max_size_kb": 0,
    "max_objects": -1
  },
  "user_quota": {
    "enabled": false,
    "check_on_raw": false,
    "max_size": -1,
    "max_size_kb": 0,
    "max_objects": -1
  },
}
```

```

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

```

4.4. 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: UOICCC8TahFKIWuv9DB09TWHF0nDjpPEIha0kAa
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 헤더**는 [Red Hat Ceph Storage Developer Guide](#) 를 참조하십시오.
- **Swift 응답 헤더**는 [Red Hat Ceph Storage Developer Guide](#) 를 참조하십시오.

4.5. SWIFT 컨테이너 작업

개발자는 **Ceph Object Gateway**를 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.

4.5.1. 사전 요구 사항

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

4.5.2. Swift 컨테이너 작업

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



참고

Amazon S3 API는 'bucket'이라는 용어를 사용하여 데이터 컨테이너를 설명합니다. 다른 사람이 **Swift API**에서 '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 게이트웨이**의 **Swift 호환 API**는 '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 매 니페스트 파일을 덮어씁니다. 따라서 기존 개체를 덮어쓰면 데이터 손실로 이어집니다. **Consequently, an existing object is overwritten, which leads to data loss.**

4.5.3. 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
```

요청 헤더

X-Container-Read

설명

컨테이너에 대한 읽기 권한이 있는 사용자 ID입니다.

유형

사용자 ID의 쉼표로 구분된 문자열 값.

필수 항목

없음

X-Container-Write

설명

컨테이너에 대한 쓰기 권한이 있는 사용자 ID입니다.

유형

사용자 ID의 접두어로 구분된 문자열 값.

필수 항목

없음

4.5.4. 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
```

요청 매개변수

limit

설명

결과 수를 지정된 값으로 제한합니다.

유형

정수

유효한 값

해당 없음

필수 항목

있음

형식

설명

결과 수를 지정된 값으로 제한합니다.

유형

정수

유효한 값

JSON 또는 **xml**

필수 항목

없음

marker

설명

마커 값보다 큰 결과 목록을 반환합니다.

유형

문자열

유효한 값

해당 없음

필수 항목

없음

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

응답 엔티티

계정

설명

계정 정보 목록입니다.

유형

컨테이너

컨테이너

설명

컨테이너 목록.

유형

컨테이너

name

설명

컨테이너의 이름입니다.

유형

문자열

bytes

설명

컨테이너의 크기입니다.

유형

정수

4.5.5. Swift에서 컨테이너의 오브젝트 나열

컨테이너 내의 오브젝트를 나열하려면 **API** 버전, 계정 및 컨테이너 이름을 사용하여 **GET** 요청을 만듭니다. 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
```

요청 매개변수

형식

설명

결과 수를 지정된 값으로 제한합니다.

유형

정수

유효한 값

JSON 또는 xml

필수 항목

없음

접두사

설명

결과 집합을 지정된 접두사로 시작하는 오브젝트로 제한합니다.

유형

문자열

유효한 값

해당 없음

필수 항목

없음

marker

설명

마커 값보다 큰 결과 목록을 반환합니다.

유형

문자열

유효한 값

해당 없음

필수 항목

없음

limit

설명

결과 수를 지정된 값으로 제한합니다.

유형

정수

유효한 값

0 - 10,000

필수 항목

없음

delimiter

설명

접두사와 나머지 오브젝트 이름 간의 구분 기호입니다.

유형

문자열

유효한 값

해당 없음

필수 항목

없음

경로

설명

오브젝트의 *pseudo-hierarchical* 경로입니다.

유형

문자열

유효한 값

해당 없음

필수 항목

없음

응답 엔티티

컨테이너

설명

컨테이너입니다.

유형

컨테이너

object

설명

컨테이너 내의 오브젝트입니다.

유형

컨테이너

name

설명

컨테이너 내 오브젝트의 이름입니다.

유형

문자열

hash

설명

오브젝트 콘텐츠의 해시 코드입니다.

유형

문자열

last_modified

설명

오브젝트의 콘텐츠가 마지막으로 수정된 시간입니다.

유형

날짜

content_type

설명

오브젝트의 콘텐츠 유형입니다.

유형

문자열

4.5.6. 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 멍청합니다. 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
```

headers

X-Container-Read

설명

컨테이너에 대한 읽기 권한이 있는 사용자 ID입니다.

유형

사용자 ID의 쉼표로 구분된 문자열 값.

필수 항목

없음

X-Container-Write

설명

컨테이너에 대한 쓰기 권한이 있는 사용자 ID입니다.

유형

사용자 ID의 쉼표로 구분된 문자열 값.

필수 항목

없음

X-Container-Meta-KEY

설명

임의의 문자열 값을 사용하는 사용자 정의 메타데이터 키입니다.

유형

문자열

필수 항목

없음

X-Storage-Policy

설명

Ceph Object Gateway의 **placement_pools** 에서 스토리지 정책을 식별하는 키입니다. 사용 가능한 키에 대해 **radosgw-admin** 영역을 가져옵니다.

유형

문자열

필수 항목

없음

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

HTTP 응답

409

상태 코드

BucketAlreadyExists

설명

컨테이너는 이미 다른 사용자의 소유권에 있습니다.

4.5.7. 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
```

HTTP 응답

204

상태 코드

NoContent

설명

컨테이너가 제거되었습니다.

4.5.8. 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
```

요청 헤더

X-Container-Meta-KEY

설명

임의의 문자열 값을 사용하는 사용자 정의 메타데이터 키입니다.

유형

문자열

필수 항목

없음

4.6. SWIFT 오브젝트 작업

개발자는 **Ceph Object Gateway**를 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.

4.6.1. 사전 요구 사항

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

4.6.2. 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.

4.6.3. 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
```

요청 헤더

범위

설명

오브젝트 콘텐츠의 하위 집합을 검색하려면 바이트 범위를 지정할 수 있습니다.

유형

날짜

필수 항목

없음

if-Modified-Since

설명

소스 오브젝트의 **last_modified** 특성의 날짜와 시간 이후에 수정된 경우에만 복사합니다.

유형

날짜

필수 항목

없음

if-Unmodified-Since

설명

소스 오브젝트의 **last_modified** 특성의 날짜 및 시간 이후에 수정되지 않은 경우에만 복사합니다.

유형

날짜

필수 항목

없음

copy-If-Match

설명

요청에 있는 **ETag**가 소스 오브젝트의 **ETag**와 일치하는 경우에만 복사합니다.

유형

etag

필수 항목

없음

copy-If-None-Match

설명

요청에 있는 **ETag**가 소스 오브젝트의 **ETag**와 일치하지 않는 경우에만 복사합니다.

유형

etag

필수 항목

없음

응답 헤더

content-Range

설명

오브젝트 콘텐츠의 하위 집합 범위입니다. **range** 헤더 필드가 요청에 지정된 경우에만 반환됩니다.

4.6.4. 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 멍등이 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
```

요청 헤더

etag

설명

오브젝트 콘텐츠의 **MD5** 해시입니다. 권장 사항.

유형

문자열

유효한 값

해당 없음

필수 항목

없음

content-Type

설명

오브젝트 콘텐츠의 **MD5** 해시입니다.

유형

문자열

유효한 값

해당 없음

필수 항목

없음

transfer-authorization

설명

개체가 더 큰 집계 개체의 일부인지 여부를 나타냅니다. **Indicates whether the object is part of a larger aggregate object.**

유형

문자열

유효한 값

chunked

필수 항목

없음

4.6.5. 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
```

4.6.6. 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 이름**으로 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 멍등이 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
```

요청 헤더

X-Copy-From

설명

PUT 요청과 함께 사용하여 소스 컨테이너/오브젝트 경로를 정의합니다.

유형

문자열

필수 항목

PUT 를 사용하는 경우 .

대상

설명

대상 컨테이너/오브젝트 경로를 정의하는 **COPY** 요청과 함께 사용됩니다.

유형

문자열

필수 항목

예. **COPY** 를 사용하는 경우 .

if-Modified-Since

설명

소스 오브젝트의 **last_modified** 특성의 날짜와 시간 이후에 수정된 경우에만 복사합니다.

유형

날짜

필수 항목

없음

if-Unmodified-Since

설명

소스 오브젝트의 **last_modified** 특성의 날짜 및 시간 이후에 수정되지 않은 경우에만 복사합니다.

유형

날짜

필수 항목

없음

copy-If-Match

설명

요청에 있는 **ETag**가 소스 오브젝트의 **ETag**와 일치하는 경우에만 복사합니다.

유형

etag

필수 항목

없음

copy-If-None-Match

설명

요청에 있는 **ETag**가 소스 오브젝트의 **ETag**와 일치하지 않는 경우에만 복사합니다.

유형

etag

필수 항목

없음

4.6.7. Swift에서 오브젝트 메타데이터 가져오기

오브젝트의 메타데이터를 검색하려면 **API** 버전, 계정, 컨테이너 및 오브젝트 이름을 사용하여 **HEAD**를 요청합니다. 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
```

4.6.8. 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
```

요청 헤더

X-Object-Meta-KEY

설명

임의의 문자열 값을 사용하는 사용자 정의 메타 데이터 키입니다.

유형

문자열

필수 항목

없음

4.7. 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.

4.7.1. Swift에서 임시 URL 오브젝트 가져오기

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

- 요청 방법의 값, 인스턴스의 **"GET"**
- 만료 시간, **epoch** 이후의 초 형식으로, 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
```

4.7.2. 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
```

요청 헤더

X-Account-Meta-Temp-URL-Key

설명

임의의 문자열 값을 사용하는 사용자 정의 키입니다.

유형

문자열

필수 항목

있음

X-Account-Meta-Temp-URL-Key-2

설명

임의의 문자열 값을 사용하는 사용자 정의 키입니다.

유형

문자열

필수 항목

없음

4.8. 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 참조된 컨테이너 및 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")
```

■

4.9. 추가 리소스

- 멀티 테넌시에 대한 자세한 내용은 [Red Hat Ceph Storage Object Gateway 가이드](#) 를 참조하십시오.
- **Swift 요청 헤더**는 [부록 E. Swift 요청 헤더](#) 을 참조하십시오.
- **Swift 응답 헤더**는 [부록 F. Swift 응답 헤더](#) 을 참조하십시오.

부록 A. CEPH RESTFUL API 사양

스토리지 관리자는 **Ceph RESTful API** 엔드포인트를 통해 다양한 **Ceph** 하위 시스템에 액세스할 수 있습니다. 사용 가능한 **Ceph RESTful API** 방법에 대한 참조 가이드입니다.

사용 가능한 **Ceph API** 끝점:

- [A.2절. “Ceph 요약”](#)
- [A.3절. “인증”](#)
- [A.4절. “Ceph 파일 시스템”](#)
- [A.5절. “스토리지 클러스터 구성”](#)
- [A.6절. “CRUSH 규칙”](#)
- [A.7절. “코드 프로필 삭제”](#)
- [A.8절. “기능 토글”](#)
- [A.9절. “Grafana”](#)
- [A.10절. “스토리지 클러스터 상태”](#)
- [A.11절. “호스트”](#)
- [A.12절. “iSCSI”](#)

- [A.13절. “로그”](#)
- [A.14절. “Ceph Manager 모듈”](#)
- [A.15절. “Ceph Monitor”](#)
- [A.16절. “Ceph OSD”](#)
- [A.17절. “Ceph 오브젝트 게이트웨이”](#)
- [A.18절. “역할 조작을 위한 REST API”](#)
- [A.19절. “NFS Ganesha”](#)
- [A.20절. “Ceph Orchestrator”](#)
- [A.21절. “pool”](#)
- [A.22절. “Prometheus”](#)
- [A.23절. “RADOS 블록 장치”](#)
- [A.24절. “성능 카운터”](#)
- [A.25절. “역할”](#)
- [A.26절. “서비스”](#)

- **A.27절. “설정”**
- **A.28절. “Ceph 작업”**
- **A.29절. “telemetry”**
- **A.30절. “Ceph 사용자”**

A.1. 사전 요구 사항

- **RESTful API 사용 방법에 대한 이해.**
- **정상 실행 중인 Red Hat Ceph Storage 클러스터.**
- **Ceph Manager 대시보드 모듈이 활성화됩니다.**

A.2. CEPH 요약

Ceph RESTful API 요약 끝점을 사용하여 Ceph 요약 세부 정보를 표시하는 방법 참조입니다.

GET /api/summary

설명

Ceph 세부 정보 요약 표시.

예제

```
GET /api/summary HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**

- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스** 먼저 로그인하십시오.
- **403 forbidden - 인증되지 않은 액세스입니다.** 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 [CephRESTful API](#) 장을 참조하십시오.

A.3. 인증

Ceph RESTful API 인증 끝점을 사용하여 **Red Hat Ceph Storage** 세션을 시작하는 방법 참조입니다.

POST /api/auth

curl 예

```
curl -i -k --location -X POST 'https://192.168.0.44:8443/api/auth' -H 'Accept: application/vnd.ceph.api.v1.0+json' -H 'Content-Type: application/json' --data '{"password": "admin@123", "username": "admin"}'
```

예제

```
POST /api/auth HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "password": "STRING",
  "username": "STRING"
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/auth/check

설명

인증 토큰의 요구 사항을 확인합니다.

예제

```
POST /api/auth/check?token=STRING HTTP/1.1
Host: example.com
Content-Type: application/json
```

```
{
  "token": "STRING"
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.

- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/auth/logout

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 [CephRESTful API](#) 장을 참조하십시오.

A.4. CEPH 파일 시스템

Ceph RESTful API cephfs 엔드포인트를 사용하여 **Ceph File Systems(CephFS)**를 관리하는 방법 [참](#)

조.

GET /api/cephfs

예제

```
GET /api/cephfs HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다.** 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/cephfs/FS_ID

매개 변수

- **FS_ID** 를 Ceph 파일 시스템 식별자 문자열로 바꿉니다.

예제

```
GET /api/cephfs/FS_ID HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.

- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

DELETE /api/cephfs/FS_ID/client/CLIENT_ID

매개 변수

- **FS_ID** 를 Ceph 파일 시스템 식별자 문자열로 바꿉니다.
- **CLIENT_ID** 를 Ceph 클라이언트 ID 문자열로 교체합니다.

상태 코드

- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/cephfs/FS_ID/clients

매개 변수

- **FS_ID** 를 **Ceph** 파일 시스템 식별자 문자열로 바꿉니다.

예제

```
GET /api/cephfs/FS_ID/clients HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/cephfs/FS_ID/get_root_directory**설명**

ls_dir API 호출을 사용하여 가져올 수 없는 루트 디렉터리입니다.

매개 변수

- **FS_ID** 를 **Ceph** 파일 시스템 식별자 문자열로 바꿉니다.

예제

```
GET /api/cephfs/FS_ID/get_root_directory HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인

- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스** 먼저 로그인하십시오.
- **403 forbidden - 인증되지 않은 액세스입니다.** 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/cephfs/FS_ID/ls_dir**설명**

지정된 경로의 디렉터리를 나열합니다.

매개 변수

- **FS_ID** 를 Ceph 파일 시스템 식별자 문자열로 바꿉니다.
- **쿼리:**
 - **path** - 목록을 시작할 문자열 값입니다. 기본 경로는 지정되지 않은 경우 / 입니다.
 - **integer value that specifies the number of steps to go down the directory tree.** 디렉터리 트리 아래로 이동하는 단계 수를 지정하는 정수 값입니다.

예제

```
GET /api/cephfs/FS_ID/ls_dir HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**

- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스** 먼저 로그인하십시오.
- **403 forbidden - 인증되지 않은 액세스입니다.** 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/cephfs/FS_ID/mds_counters

매개 변수

- **FS_ID** 를 **Ceph** 파일 시스템 식별자 문자열로 바꿉니다.
- 쿼리:
 - **Count** - 정수 값입니다. **Seriality** - an integer value.

예제

```
GET /api/cephfs/FS_ID/mds_counters HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스** 먼저 로그인하십시오.
- **403 forbidden - 인증되지 않은 액세스입니다.** 권한을 확인해 주십시오.

- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/cephfs/FS_ID/quota

설명

지정된 경로에 **CephFS** 할당량을 표시합니다.

매개 변수

- **FS_ID** 를 **Ceph** 파일 시스템 식별자 문자열로 바꿉니다.
- 쿼리:
 - **path** - 디렉터리 경로를 지정하는 필수 문자열 값입니다.

예제

```
GET /api/cephfs/FS_ID/quota?path=STRING HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

PUT /api/cephfs/FS_ID/quota

설명

지정된 경로에 대한 할당량을 설정합니다.

매개 변수

- **FS_ID** 를 **Ceph** 파일 시스템 식별자 문자열로 바꿉니다.
- **max_bytes** - 바이트 제한을 정의하는 문자열 값입니다.
- **max_files** - 파일 제한을 정의하는 문자열 값입니다.
- **path** - 디렉터리 또는 파일의 경로를 정의하는 문자열 값입니다.

예제

```
PUT /api/cephfs/FS_ID/quota HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "max_bytes": "STRING",
  "max_files": "STRING",
  "path": "STRING"
}
```

상태 코드

- **200 OK** - 확인
- **202 허용** - 작업이 아직 실행되고 있으며 작업 대기열을 확인합니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.

- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

DELETE /api/cephfs/FS_ID/snapshot

설명

snapsnot를 제거합니다.

매개 변수

- **FS_ID**를 **Ceph** 파일 시스템 식별자 문자열로 바꿉니다.
- 쿼리:
 - **Name** - 스냅샷 이름을 지정하는 데 필요한 문자열 값입니다.
 - **path** - 디렉터리의 경로를 정의하는 필수 문자열 값입니다.

상태 코드

- **202 허용** - 작업이 아직 실행되고 있으며 작업 대기열을 확인합니다.
- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/cephfs/FS_ID/snapshot**설명**

스냅샷을 만듭니다.

매개 변수

- **FS_ID** 를 **Ceph** 파일 시스템 식별자 문자열로 바꿉니다.
- **Name** - 스냅샷 이름을 지정하는 문자열 값입니다. 이름을 지정하지 않으면 현재 시간을 사용하는 이름이 **RFC3339 UTC** 형식으로 생성됩니다.
- **path** - 디렉터리의 경로를 정의하는 문자열 값입니다.

예제

```
POST /api/cephfs/FS_ID/snapshot HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "name": "STRING",
  "path": "STRING"
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 아직 실행되고 있으며 작업 대기열을 확인합니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

시오.

DELETE /api/cephfs/FS_ID/tree

설명

디렉터리를 제거합니다.

매개 변수

- **FS_ID**를 Ceph 파일 시스템 식별자 문자열로 바꿉니다.
- 쿼리:
 - **path** - 디렉터리의 경로를 정의하는 필수 문자열 값입니다.

상태 코드

- **202 허용** - 작업이 아직 실행되고 있으며 작업 대기열을 확인합니다.
- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/cephfs/FS_ID/tree

설명

디렉터리를 생성합니다.

매개 변수

- **FS_ID** 를 **Ceph** 파일 시스템 식별자 문자열로 바꿉니다.
- **path** - 디렉터리의 경로를 정의하는 문자열 값입니다.

예제

```
POST /api/cephfs/FS_ID/tree HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "path": "STRING"
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 아직 실행되고 있으며 작업 대기열을 확인합니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 [CephRESTful API](#) 장을 참조하십시오.

시오.

A.5. 스토리지 클러스터 구성

Ceph RESTful API cluster_conf 끝점을 사용하여 **Red Hat Ceph Storage** 클러스터를 관리하는 방법 참조입니다.

GET /api/cluster_conf

예제

```
GET /api/cluster_conf HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.**
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

시오.

POST /api/cluster_conf

예제

```
POST /api/cluster_conf HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "name": "STRING",
  "value": "STRING"
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 아직 실행되고 있으며 작업 대기열을 확인합니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

PUT /api/cluster_conf

예제

```
PUT /api/cluster_conf HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "options": "STRING"
}
```

상태 코드

- **200 OK** - 확인
- **202 허용** - 작업이 아직 실행되고 있으며 작업 대기열을 확인합니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.

- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/cluster_conf/filter

설명

스토리지 클러스터 구성을 이름으로 표시합니다.

매개 변수

- 쿼리:
 - **Name** - 구성 옵션 이름의 문자열 값입니다.

예제

```
GET /api/cluster_conf/filter HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

DELETE /api/cluster_conf/NAME

매개 변수

- **NAME** 을 스토리지 클러스터 구성 이름으로 교체합니다.
- 쿼리:
 - **section** - 필요한 문자열 값입니다.

상태 코드

- **202 허용** - 작업이 아직 실행되고 있으며 작업 대기열을 확인합니다.
- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/cluster_conf/NAME

매개 변수

- **NAME** 을 스토리지 클러스터 구성 이름으로 교체합니다.

예제

```
GET /api/cluster_conf/NAME HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스** 먼저 로그인하십시오.
- **403 forbidden - 인증되지 않은 액세스**입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 [CephRESTful API](#) 장을 참조하십시오.

A.6. CRUSH 규칙

Ceph RESTful API `crush_rule` 엔드포인트를 사용하여 **CRUSH** 규칙을 관리하는 방법 참조입니다.

GET /api/crush_rule

설명

CRUSH 규칙 구성을 나열합니다.

예제

```
GET /api/crush_rule HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**

- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스** 먼저 로그인하십시오.
- **403 forbidden - 인증되지 않은 액세스입니다.** 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/crush_rule

예제

```
POST /api/crush_rule HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "device_class": "STRING",
  "failure_domain": "STRING",
  "name": "STRING",
  "root": "STRING"
}
```

상태 코드

- **201 생성 - 리소스가 생성되었습니다.**
- **202 허용 - 작업이 아직 실행되고 있으며 작업 대기열을 확인합니다.**
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스** 먼저 로그인하십시오.
- **403 forbidden - 인증되지 않은 액세스입니다.** 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

시오.

DELETE /api/crush_rule/NAME

매개 변수

- **NAME** 을 규칙 이름으로 교체합니다.

상태 코드

- **202 허용** - 작업이 아직 실행되고 있으며 작업 대기열을 확인합니다.
- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/crush_rule/NAME

매개 변수

- **NAME** 을 규칙 이름으로 교체합니다.

예제

```
GET /api/crush_rule/NAME HTTP/1.1
Host: example.com
```

상태 코드

- **202 허용** - 작업이 아직 실행되고 있으며 작업 대기열을 확인합니다.
- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 Red Hat Ceph Storage 개발자 가이드의 [CephRESTful API](#) 장을 참조하십시오.

A.7. 코드 프로필 삭제

Ceph RESTful API erasure_code_profile 엔드포인트를 사용하여 삭제 코딩을 위한 프로필을 관리하는 방법 참조입니다.

GET /api/erasure_code_profile

설명

삭제된 프로필 정보를 나열합니다.

예제

```
GET /api/erasure_code_profile HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/erasure_code_profile

예제

```
POST /api/erasure_code_profile HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "name": "STRING"
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 아직 실행되고 있으며 작업 대기열을 확인합니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
-

500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

DELETE /api/erasure_code_profile/NAME

매개 변수

- **NAME** 을 프로필 이름으로 교체합니다.

상태 코드

- **202 허용** - 작업이 아직 실행되고 있으며 작업 대기열을 확인합니다.
- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.**

GET /api/erasure_code_profile/NAME

매개 변수

- **NAME** 을 프로필 이름으로 교체합니다.

예제

```
GET /api/erasure_code_profile/NAME HTTP/1.1
Host: example.com
```

상태 코드

- **202 허용** - 작업이 아직 실행되고 있으며 작업 대기열을 확인합니다.
- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 [CephRESTful API](#) 장을 참조하십시오.

A.8. 기능 토글

Ceph RESTful API feature_toggles 엔드포인트를 사용하여 **CRUSH** 규칙을 관리하는 방법 참조입니다.

GET /api/feature_toggles

설명

Red Hat Ceph Storage의 기능을 나열합니다.

예제

```
GET /api/feature_toggles HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스** 먼저 로그인하십시오.
- **403 forbidden - 인증되지 않은 액세스입니다.** 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 [CephRESTful API](#) 장을 참조하십시오.

A.9. GRAFANA

Ceph RESTful API grafana 엔드포인트를 사용하여 **Grafana**를 관리하는 방법 참조입니다.

POST /api/grafana/dashboards

상태 코드

- **201 생성 - 리소스가 생성되었습니다.**
- **202 허용 - 작업이 실행 중입니다.** 작업 대기열을 확인하십시오.
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스** 먼저 로그인하십시오.

- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/grafana/url

설명

Grafana URL 인스턴스를 나열합니다.

예제

```
GET /api/grafana/url HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/grafana/validation/PARAMS

매개 변수

- **PARAMS** 를 문자열 값으로 바꿉니다.

예제

```
GET /api/grafana/validation/PARAMS HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 [CephRESTful API](#) 장을 참조하십시오.

A.10. 스토리지 클러스터 상태

Ceph RESTful API 상태 끝점을 사용하여 스토리지 클러스터 상태 세부 정보 및 상태를 표시하는 방법 참조입니다.

GET /api/health/full

예제

```
GET /api/health/full HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인

- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스** 먼저 로그인하십시오.
- **403 forbidden - 인증되지 않은 액세스입니다.** 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/health/minimal

설명

스토리지 클러스터의 최소 상태 보고서를 표시합니다.

예제

```
GET /api/health/minimal HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스** 먼저 로그인하십시오.
- **403 forbidden - 인증되지 않은 액세스입니다.** 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 **CephRESTful API** 장을 참조하십시오.

A.11. 호스트

Ceph RESTful API 호스트 끝점을 사용하여 노드 정보라고도 하는 호스트를 표시하는 방법에 대한 메서드 참조입니다.

GET /api/host

설명

호스트 사양을 나열합니다.

매개 변수

- 쿼리:
 - Source** - 호스트 소스의 문자열 값입니다.

예제

```
GET /api/host HTTP/1.1
Host: example.com
```

상태 코드

- 200 OK** - 확인
- 400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- 401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- 403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- 500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/host**예제**

```
POST /api/host HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "hostname": "STRING",
  "status": "STRING"
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

DELETE /api/host/HOST_NAME**매개 변수**

- **HOST_NAME** 을 노드 이름으로 교체합니다.

상태 코드

- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.

- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/host/HOST_NAME

설명

지정된 호스트에 대한 정보를 표시합니다.

매개 변수

- **HOST_NAME** 을 노드 이름으로 교체합니다.

예제

```
GET /api/host/HOST_NAME HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.

- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

PUT /api/host/HOST_NAME

설명

지정된 호스트의 정보를 업데이트합니다. 이 방법은 **Ceph Orchestrator**가 활성화된 경우에만 지원됩니다.

매개 변수

- **HOST_NAME** 을 노드 이름으로 교체합니다.
- **강제로 - 호스트를 유지 관리 모드로 전환합니다.**
- **labels** - 레이블 목록입니다.
- **유지 관리 - 유지 관리 모드를 입력하거나 종료합니다.**
- **update_labels** - 레이블을 업데이트합니다.

예제

```
PUT /api/host/HOST_NAME HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "force": true,
  "labels": [
    "STRING"
  ],
  "maintenance": true,
  "update_labels": true
}
```

상태 코드

- **200 OK - 확인**

- **202 허용 - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.**
- **400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.**
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.**
- **500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.**

GET /api/host/HOST_NAME/daemons

매개 변수

- **HOST_NAME** 을 노드 이름으로 교체합니다.

예제

```
GET /api/host/HOST_NAME/daemons HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.**
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.**
- **500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.**

GET /api/host/HOST_NAME/devices

매개 변수

- **HOST_NAME** 을 노드 이름으로 교체합니다.

예제

```
GET /api/host/HOST_NAME/devices HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/host/HOST_NAME/identify_device

설명

장치의 표시등을 지정된 시간(초)으로 전환하여 장치를 식별합니다.

매개 변수

- **HOST_NAME** 을 노드 이름으로 교체합니다.
- **장치** - /dev/dm-0 또는 **ABC1234DEF567-1R1234_ABC8DE0Q** 와 같은 장치 ID입니다.
- **duration** - 장치의 LED가 플래쉬해야 하는 시간(초)입니다.

예제

```
POST /api/host/HOST_NAME/identify_device HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "device": "STRING",
  "duration": "STRING"
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/host/HOST_NAME/inventory

설명

호스트 인벤토리를 표시합니다.

매개 변수

- **HOST_NAME** 을 노드 이름으로 교체합니다.
- 쿼리:
 -

새로 고침 - 비동기 새로 고침을 트리거하는 문자열 값입니다. **Refresh - A string value to trigger an asynchronous refresh.**

예제

```
GET /api/host/HOST_NAME/inventory HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.**
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.**
- **500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.**

GET /api/host/HOST_NAME/smart

매개 변수

- **HOST_NAME** 을 노드 이름으로 교체합니다.

예제

```
GET /api/host/HOST_NAME/smart HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.**

- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 [CephRESTful API](#) 장을 참조하십시오.

A.12. iSCSI

Ceph RESTful API iscsi 끝점을 사용하여 **iSCSI**를 관리하는 방법 참조입니다.

GET /api/iscsi/discoveryauth

설명

iSCSI 검색 인증 세부 정보를 확인합니다.

예제

```
GET /api/iscsi/discoveryauth HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.

- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

PUT /api/iscsi/discoveryauth**설명**

iSCSI 검색 인증을 설정합니다.

매개 변수

- **쿼리:**
 - **user** - 필수 사용자 이름 문자열입니다.
 - **암호** - 필수 암호 문자열입니다.
 - **mutual_user** - 필수 상호 사용자 이름 문자열입니다.
 - **mutual_password** - 필수 상호 암호 문자열입니다.

예제

```
PUT /api/iscsi/discoveryauth?
user=STRING&password=STRING&mutual_user=STRING&mutual_password=STRING
HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "mutual_password": "STRING",
  "mutual_user": "STRING",
  "password": "STRING",
  "user": "STRING"
}
```

상태 코드

- **200 OK - 확인**
- **202 허용 - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.**
- **400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.**
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.**
- **500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.**

GET /api/iscsi/target**예제**

```
GET /api/iscsi/target HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.**
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.**
- **500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.**

POST /api/iscsi/target**예제**

```
POST /api/iscsi/target HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "acl_enabled": "STRING",
  "auth": "STRING",
  "clients": "STRING",
  "disks": "STRING",
  "groups": "STRING",
  "portals": "STRING",
  "target_controls": "STRING",
  "target_iqn": "STRING"
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

DELETE /api/iscsi/target/TARGET_IQN**매개 변수**

- **ECDHE GET_IQN** 을 경로 문자열로 바꿉니다.

상태 코드

- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/iscsi/target/TARGET_IQN

매개 변수

- **ECDHE GET_IQN** 을 경로 문자열로 바꿉니다.

예제

```
GET /api/iscsi/target/TARGET_IQN HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.

- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

PUT /api/iscsi/target/TARGET_IQN

매개 변수

- **ECDHE GET_IQN** 을 경로 문자열로 바꿉니다.

예제

```
PUT /api/iscsi/target/TARGET_IQN HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "acl_enabled": "STRING",
  "auth": "STRING",
  "clients": "STRING",
  "disks": "STRING",
  "groups": "STRING",
  "new_target_iqn": "STRING",
  "portals": "STRING",
  "target_controls": "STRING"
}
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다.** 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 Red Hat Ceph Storage 개발자 가이드의 [CephRESTful API](#) 장을 참조하십시오.

A.13. 로그

로그 정보를 표시하는 **Ceph RESTful API** 로그 끝점을 사용하는 방법 참조입니다.

GET /api/logs/all

설명

모든 로그 구성을 확인합니다.

예제

```
GET /api/logs/all HTTP/1.1
Host: example.com
```

상태 코드

- 200 OK - 확인**
- 400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- 401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- 403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.**
- 500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 Red Hat Ceph Storage 개발자 가이드의 [CephRESTful API](#) 장을 참조하십시오.

A.14. CEPH MANAGER 모듈

Ceph RESTful API mgr/module 엔드포인트를 사용하여 **Ceph Manager** 모듈을 관리하는 방법 참조입니다.

GET /api/mgr/module

설명

관리 모듈 목록을 확인합니다.

예제

```
GET /api/mgr/module HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스** 먼저 로그인하십시오.
- **403 forbidden - 인증되지 않은 액세스입니다.** 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/mgr/module/MODULE_NAME

설명

영구 구성 설정 값을 검색합니다.

매개 변수

- **MODULE_NAME** 을 **Ceph Manager** 모듈 이름으로 교체합니다.

예제

-


```
GET /api/mgr/module/MODULE_NAME HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.**
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.**
- **500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.**

PUT /api/mgr/module/MODULE_NAME

설명

영구 구성 설정 값을 설정합니다.

매개 변수

- **MODULE_NAME** 을 **Ceph Manager** 모듈 이름으로 교체합니다.
- **config** - 모듈 옵션의 값.

예제

```
PUT /api/mgr/module/MODULE_NAME HTTP/1.1
Host: example.com
Content-Type: application/json
```

```
{
  "config": "STRING"
}
```

상태 코드

- **200 OK - 확인**
- **202 허용 - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.**
- **400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.**
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.**
- **500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.**

POST /api/mgr/module/MODULE_NAME/disable

설명

지정된 **Ceph Manager** 모듈을 비활성화합니다.

매개 변수

- **MODULE_NAME** 을 **Ceph Manager** 모듈 이름으로 교체합니다.

상태 코드

- **201 생성 - 리소스가 생성되었습니다.**
- **202 허용 - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.**
- **400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.**
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**

- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/mgr/module/MODULE_NAME/enable

설명

지정된 **Ceph Manager** 모듈을 활성화합니다.

매개 변수

- **MODULE_NAME** 을 **Ceph Manager** 모듈 이름으로 교체합니다.

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/mgr/module/MODULE_NAME/options

설명

지정된 **Ceph Manager** 모듈의 옵션을 봅니다.

매개 변수

- **MODULE_NAME** 을 **Ceph Manager** 모듈 이름으로 교체합니다.

예제

```
GET /api/mgr/module/MODULE_NAME/options HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 [CephRESTful API](#) 장을 참조하십시오.

A.15. CEPH MONITOR

Ceph RESTful API 모니터 끝점을 사용하여 **Ceph** 모니터에 대한 정보를 표시하는 방법 참조입니다.

GET /api/monitor

설명

Ceph 모니터 세부 정보를 확인합니다.

예제

```
GET /api/monitor HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 [CephRESTful API](#) 장을 참조하십시오.

A.16. CEPH OSD

Ceph RESTful API osd 엔드포인트를 사용하여 **Ceph OSD**를 관리하는 방법 참조입니다.

GET /api/osd**예제**

```
GET /api/osd HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인

- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/osd**예제**

```
POST /api/osd HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "data": "STRING",
  "method": "STRING",
  "tracking_id": "STRING"
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/osd/flags**설명**

Ceph OSD 플래그를 확인합니다.

예제

```
GET /api/osd/flags HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.**
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

PUT /api/osd/flags**설명**

전체 스토리지 클러스터의 Ceph OSD 플래그를 설정합니다.

매개 변수

- **recovery_deletes, sortbitwise, pglog_hardlimit** 플래그는 설정을 해제할 수 없습니다.
- **purged_snapshots** 플래그를 설정할 수 없습니다.

**중요**

성공적인 작업을 위해 다음 4개의 플래그를 포함해야 합니다.

예제

```
PUT /api/osd/flags HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "flags": [
    "STRING"
  ]
}
```

상태 코드

- **200 OK** - 확인
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/osd/flags/individual**설명**

개별 Ceph OSD 플래그를 확인합니다.

예제


```
GET /api/osd/flags/individual HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

PUT /api/osd/flags/individual

설명

Ceph OSD의 개별 하위 집합에 대한 **no out, noin, noup** 플래그를 업데이트합니다.

예제

```
PUT /api/osd/flags/individual HTTP/1.1
Host: example.com
Content-Type: application/json
```

```
{
  "flags": {
    "nodown": true,
    "noin": true,
    "noout": true,
    "noup": true
  },
  "ids": [
    1
  ]
}
```

상태 코드

- **200 OK - 확인**
- **202 허용 - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.**
- **400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.**
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.**
- **500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.**

GET /api/osd/safe_to_delete

매개 변수

- 쿼리:
 - **svc_ids - Ceph OSD 서비스 식별자의 필수 문자열입니다.**

예제

```
GET /api/osd/safe_to_delete?svc_ids=STRING HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.**
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**

- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/osd/safe_to_destroy

설명

Ceph OSD가 안전한지 확인합니다.

매개 변수

- 쿼리:
 - **ID - Ceph OSD 서비스 식별자의 필수 문자열입니다.**

예제

```
GET /api/osd/safe_to_destroy?ids=STRING HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

DELETE /api/osd/SVC_ID

매개 변수

- **SVC_ID** 를 **Ceph OSD** 서비스 식별자의 문자열 값으로 바꿉니다.
- 쿼리:
 - **preserve_id** - 문자열 값입니다.
 - **force** - 문자열 값입니다.

상태 코드

- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/osd/SVC_ID

설명

Ceph OSD에 대한 수집된 데이터를 반환합니다.

매개 변수

- **SVC_ID** 를 **Ceph OSD** 서비스 식별자의 문자열 값으로 바꿉니다.

예제

```
GET /api/osd/SVC_ID HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

PUT /api/osd/SVC_ID

매개 변수

- **SVC_ID** 를 Ceph OSD 서비스 식별자의 문자열 값으로 바꿉니다.

예제

```
PUT /api/osd/SVC_ID HTTP/1.1
Host: example.com
Content-Type: application/json
```

```
{
  "device_class": "STRING"
}
```

상태 코드

- **200 OK** - 확인
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.

- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스** 먼저 로그인하십시오.
- **403 forbidden - 인증되지 않은 액세스**입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/osd/SVC_ID/destroy

설명

Ceph OSD가 삭제됨으로 표시됩니다. 제거되기 전에 **Ceph OSD**를 축소해야 합니다. 이 작업은 **Ceph OSD ID**를 그대로 유지하지만 **Cephx** 키, 구성 키 데이터, 잠금 상자 키를 제거합니다.



주의

이 작업은 데이터를 영구적으로 잃을 수 없도록 렌더링합니다.

매개 변수

- **SVC_ID** 를 **Ceph OSD** 서비스 식별자의 문자열 값으로 바꿉니다.

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.

- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/osd/SVC_ID/devices

매개 변수

- **SVC_ID** 를 **Ceph OSD** 서비스 식별자의 문자열 값으로 바꿉니다.

예제

```
GET /api/osd/SVC_ID/devices HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/osd/SVC_ID/histogram

설명

Ceph OSD 히스토그램 데이터를 반환합니다.

매개 변수

- **SVC_ID** 를 **Ceph OSD** 서비스 식별자의 문자열 값으로 바꿉니다.

예제

```
GET /api/osd/SVC_ID/histogram HTTP/1.1
Host: example.com
```

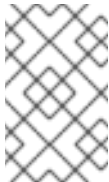
상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

PUT /api/osd/SVC_ID/mark

설명

Ceph OSD 를,아래쪽으로 표시하고 손실된 를 표시합니다.



참고

손실된 것으로 표시하기 전에 **Ceph OSD**를 축소 해야 합니다.

매개 변수

- **SVC_ID** 를 **Ceph OSD** 서비스 식별자의 문자열 값으로 바꿉니다.

예제


```
PUT /api/osd/SVC_ID/mark HTTP/1.1
Host: example.com
Content-Type: application/json
```

```
{
  "action": "STRING"
}
```

상태 코드

- **200 OK** - 확인
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/osd/SVC_ID/purge

설명

ArgoCD 맵에서 Ceph OSD를 제거합니다.



참고

제거 전에 Ceph OSD를 표시해야 합니다.

매개 변수

- **SVC_ID** 를 Ceph OSD 서비스 식별자의 문자열 값으로 바꿉니다.

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/osd/SVC_ID/reweight**설명**

Ceph OSD를 일시적으로 다시 스케일링합니다. **Ceph OSD**가 임을 표시하면 **OSD**의 가중치가 **0**으로 설정됩니다. **Ceph OSD**가 다시 으로 표시되면 **OSD**의 가중치가 **1**로 설정됩니다.

매개 변수

- **SVC_ID**를 **Ceph OSD** 서비스 식별자의 문자열 값으로 바꿉니다.

예제

```
POST /api/osd/SVC_ID/reweight HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "weight": "STRING"
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.

- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/osd/SVC_ID/scrub

매개 변수

- **SVC_ID** 를 Ceph OSD 서비스 식별자의 문자열 값으로 바꿉니다.
- 쿼리:
 - **답** - 부울 값(**true** 또는 **false**)입니다.

예제

```
POST /api/osd/SVC_ID/scrub HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "deep": true
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.

- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스** 먼저 로그인하십시오.
- **403 forbidden - 인증되지 않은 액세스**입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/osd/SVC_ID/smart

매개 변수

- **SVC_ID** 를 **Ceph OSD** 서비스 식별자의 문자열 값으로 바꿉니다.

예제

```
GET /api/osd/SVC_ID/smart HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스** 먼저 로그인하십시오.
- **403 forbidden - 인증되지 않은 액세스**입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

-

자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 **CephRESTful API** 장을 참조하십시오.

A.17. CEPH 오브젝트 게이트웨이

Ceph RESTful API rgw 엔드포인트를 사용하여 **Ceph Object Gateway**를 관리하는 방법 참조입니다.

GET /api/rgw/status

설명

Ceph Object Gateway 상태를 표시합니다.

예제

```
GET /api/rgw/status HTTP/1.1
Host: example.com
```

상태 코드

-

200 OK - 확인

-

400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.

-

401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.

-

403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.

-

500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/rgw/daemon

설명

Ceph Object Gateway 데몬을 표시합니다.

예제

```
GET /api/rgw/daemon HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.**
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.**
- **500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.**

```
GET /api/rgw/daemon/SVC_ID
```

매개 변수

- **SVC_ID** 를 문자열 값으로 서비스 식별자로 바꿉니다.

예제

```
GET /api/rgw/daemon/SVC_ID HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.**
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**

- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/rgw/site

매개 변수

- 쿼리:
 - 쿼리 - 문자열 값입니다.
 - **daemon_name** - 문자열 값으로 데몬의 이름입니다.

예제

```
GET /api/rgw/site HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

bucket Management

GET /api/rgw/bucket

매개 변수

- 쿼리:
 - **통계** - 버킷 통계의 부울 값입니다.
 - **daemon_name** - 문자열 값으로 데몬의 이름입니다.

예제

```
GET /api/rgw/bucket HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/rgw/bucket

예제

```
POST /api/rgw/bucket HTTP/1.1
Host: example.com
Content-Type: application/json
```

```
{
  "bucket": "STRING",
  "daemon_name": "STRING",
  "lock_enabled": "false",
```



```

    "lock_mode": "STRING",
    "lock_retention_period_days": "STRING",
    "lock_retention_period_years": "STRING",
    "placement_target": "STRING",
    "uid": "STRING",
    "zonegroup": "STRING"
  }

```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

DELETE /api/rgw/bucket/BUCKET

매개 변수

- **BUCKET** 을 문자열 값으로 버킷 이름으로 교체합니다.
- 쿼리:
 - **purge_objects** - 문자열 값입니다.
 - **daemon_name** - 문자열 값으로 데몬의 이름입니다.

상태 코드

- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/rgw/bucket/BUCKET

매개 변수

- **BUCKET** 을 문자열 값으로 버킷 이름으로 교체합니다.
- 쿼리:
 - **daemon_name** - 문자열 값으로 데몬의 이름입니다.

예제

```
GET /api/rgw/bucket/BUCKET HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.

- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

PUT /api/rgw/bucket/BUCKET

매개 변수

- **BUCKET** 을 문자열 값으로 버킷 이름으로 교체합니다.

예제

```
PUT /api/rgw/bucket/BUCKET HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "bucket_id": "STRING",
  "daemon_name": "STRING",
  "lock_mode": "STRING",
  "lock_retention_period_days": "STRING",
  "lock_retention_period_years": "STRING",
  "mfa_delete": "STRING",
  "mfa_token_pin": "STRING",
  "mfa_token_serial": "STRING",
  "uid": "STRING",
  "versioning_state": "STRING"
}
```

상태 코드

- **200 OK** - 확인
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
-

-

401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.

-

403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.

-

500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

사용자 관리

GET /api/rgw/user

설명

Ceph Object Gateway 사용자를 표시합니다.

매개 변수

-

쿼리:

-

daemon_name - 문자열 값으로 데몬의 이름입니다.

예제

```
GET /api/rgw/user HTTP/1.1
Host: example.com
```

상태 코드

-

200 OK - 확인

-

400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.

-

401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.

-

403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.

-

500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/rgw/user

예제

```
POST /api/rgw/user HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "access_key": "STRING",
  "daemon_name": "STRING",
  "display_name": "STRING",
  "email": "STRING",
  "generate_key": "STRING",
  "max_buckets": "STRING",
  "secret_key": "STRING",
  "suspended": "STRING",
  "uid": "STRING"
}
```

상태 코드

- **201 생성 - 리소스가 생성되었습니다.**
- **202 허용 - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.**
- **400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.**
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.**
- **500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.**

GET /api/rgw/user/get_emails

매개 변수

-

쿼리:

-

daemon_name - 문자열 값으로 데몬의 이름입니다.

예제

```
GET /api/rgw/user/get_emails HTTP/1.1
Host: example.com
```

상태 코드

-

200 OK - 확인

-

400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.

-

401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.

-

403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.

-

500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

DELETE /api/rgw/user/UID

매개 변수

-

UID 를 사용자 식별자로 바꿉니다.

-

쿼리:

-

daemon_name - 문자열 값으로 데몬의 이름입니다.

상태 코드

- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/rgw/user/UID

매개 변수

- **UID** 를 사용자 식별자로 바꿉니다.
- 쿼리:
 - **daemon_name** - 문자열 값으로 데몬의 이름입니다.
 - **통계** - 사용자 통계의 부울 값입니다.

예제

```
GET /api/rgw/user/UID HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인

- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

PUT /api/rgw/user/UID

매개 변수

- **UID** 를 사용자 식별자로 바꿉니다.

예제

```
PUT /api/rgw/user/UID HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "daemon_name": "STRING",
  "display_name": "STRING",
  "email": "STRING",
  "max_buckets": "STRING",
  "suspended": "STRING"
}
```

상태 코드

- **200 OK** - 확인
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.

- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

DELETE /api/rgw/user/UID/capability

매개 변수

- **UID** 를 사용자 식별자로 바꿉니다.
- 쿼리:
 - **daemon_name** - 문자열 값으로 데몬의 이름입니다.
 - **유형** - 필수 항목입니다. 문자열 값입니다.
 - **perm** - 필수 항목입니다. 문자열 값입니다.

상태 코드

- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.

- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/rgw/user/UID/capability

매개 변수

- **UID** 를 사용자 식별자로 바꿉니다.

예제

```
POST /api/rgw/user/UID/capability HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "daemon_name": "STRING",
  "perm": "STRING",
  "type": "STRING"
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

DELETE /api/rgw/user/UID/key

매개 변수

- **UID** 를 사용자 식별자로 바꿉니다.
- 쿼리:
 - **daemon_name** - 문자열 값으로 데몬의 이름입니다.
 - **KEY_TYPE** - 문자열 값입니다.
 - **subuser** - 문자열 값입니다.
 - **access_key** - 문자열 값입니다.

상태 코드

- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/rgw/user/UID/key

매개 변수

- **UID** 를 사용자 식별자로 바꿉니다.

예제

```
POST /api/rgw/user/UID/key HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "access_key": "STRING",
  "daemon_name": "STRING",
  "generate_key": "true",
  "key_type": "s3",
  "secret_key": "STRING",
  "subuser": "STRING"
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/rgw/user/UID/quota

매개 변수

- **UID** 를 사용자 식별자로 바꿉니다.

예제

```
GET /api/rgw/user/UID/quota HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

PUT /api/rgw/user/UID/quota

매개 변수

- **UID** 를 사용자 식별자로 바꿉니다.

예제

```
PUT /api/rgw/user/UID/quota HTTP/1.1
Host: example.com
Content-Type: application/json
```

```
{
  "daemon_name": "STRING",
  "enabled": "STRING",
  "max_objects": "STRING",
  "max_size_kb": 1,
  "quota_type": "STRING"
}
```

상태 코드

- **200 OK** - 확인

- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/rgw/user/UID/subuser

매개 변수

- **UID** 를 사용자 식별자로 바꿉니다.

예제

```
POST /api/rgw/user/UID/subuser HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "access": "STRING",
  "access_key": "STRING",
  "daemon_name": "STRING",
  "generate_secret": "true",
  "key_type": "s3",
  "secret_key": "STRING",
  "subuser": "STRING"
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.

- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

DELETE /api/rgw/user/UID/subuser/SUBUSER

매개 변수

- **UID** 를 사용자 식별자로 바꿉니다.
- **SUBUSER** 를 하위 사용자 이름으로 바꿉니다.
- 쿼리:
 - **purge_keys** - 키를 제거하지 않도록 **false** 로 설정합니다. 이는 **S3** 하위 사용자에게만 적용됩니다.
 - **daemon_name** - 문자열 값으로 데몬의 이름입니다.

상태 코드

- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.

- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 [CephRESTful API](#) 장을 참조하십시오.

A.18. 역할 조작을 위한 REST API

radosgw-admin 역할 명령 외에도 **REST API**를 사용하여 역할을 조작할 수 있습니다.

REST 관리 **API**를 호출하려면 **admin caps**가 있는 사용자를 생성합니다.

예제

```
[root@host01 ~]# radosgw-admin --uid TESTER --display-name "TestUser" --access_key TESTER --secret test123 user create
[root@host01 ~]# radosgw-admin caps add --uid="TESTER" --caps="roles=*"

```

- 역할을 생성합니다.

구문

POST "<hostname>?"

Action=CreateRole&RoleName=ROLE_NAME&Path=PATH_TO_FILE&AssumeRolePolicyDocument=TRUST_RELATIONSHIP_POLICY_DOCUMENT"

예제

POST "<hostname>?"

Action=CreateRole&RoleName=S3Access&Path=/application_abc/component_xyz/&AssumeRolePolicyDocument={"Version":"2022-06-17","Statement":[{"Effect":"Allow","Principal":{"AWS":["arn:aws:iam:::user/TESTER"]},"Action":["sts:AssumeRole"]}]}"

응답 예

```
<role>
  <id>8f41f4e0-7094-4dc0-ac20-074a881ccbc5</id>
  <name>S3Access</name>
  <path>/application_abc/component_xyz</path>
  <arn>arn:aws:iam:::role/application_abc/component_xyz/S3Access</arn>
  <create_date>2022-06-23T07:43:42.811Z</create_date>
  <max_session_duration>3600</max_session_duration>
  <assume_role_policy_document>{"Version":"2022-06-17","Statement":
[{"Effect":"Allow","Principal":{"AWS":["arn:aws:iam:::user/TESTER"]},"Action":
["sts:AssumeRole"]}]}</assume_role_policy_document>
</role>
```

•

역할을 가져옵니다.

구문

POST "<hostname>?Action=GetRole&RoleName=ROLE_NAME"

예제

```
POST "<hostname>?Action=GetRole&RoleName=S3Access"
```

응답 예

```
<role>
  <id>8f41f4e0-7094-4dc0-ac20-074a881ccbc5</id>
  <name>S3Access</name>
  <path>/application_abc/component_xyz</path>
  <arn>arn:aws:iam::role/application_abc/component_xyz/S3Access</arn>
  <create_date>2022-06-23T07:43:42.811Z</create_date>
  <max_session_duration>3600</max_session_duration>
  <assume_role_policy_document>{"Version":"2022-06-17","Statement":
[{"Effect":"Allow","Principal":{"AWS":["arn:aws:iam::user/TESTER"]},"Action":
["sts:AssumeRole"]}]}</assume_role_policy_document>
</role>
```

•

역할을 나열합니다.

구문

```
POST "<hostname>?
Action=GetRole&RoleName=ROLE_NAME&PathPrefix=PATH_PREFIX"
```

요청 예

■

POST "<hostname>?Action=ListRoles&RoleName=S3Access&PathPrefix=/application"

응답 예

```
<role>
  <id>8f41f4e0-7094-4dc0-ac20-074a881ccbc5</id>
  <name>S3Access</name>
  <path>/application_abc/component_xyz</path>
  <arn>arn:aws:iam::role/application_abc/component_xyz/S3Access</arn>
  <create_date>2022-06-23T07:43:42.811Z</create_date>
  <max_session_duration>3600</max_session_duration>
  <assume_role_policy_document>{"Version":"2022-06-17","Statement":
[{"Effect":"Allow","Principal":{"AWS":["arn:aws:iam::user/TESTER"]},"Action":
["sts:AssumeRole"]}]}</assume_role_policy_document>
</role>
```

•

가정 역할 정책 문서를 업데이트합니다.

구문

POST "<hostname>?
Action=UpdateAssumeRolePolicy&RoleName=ROLE_NAME&PolicyDocument=TRUST_RE
LATIONSHIP_POLICY_DOCUMENT"

예제

```
POST "<hostname>?
Action=UpdateAssumeRolePolicy&RoleName=S3Access&PolicyDocument=
{"Version":"2022-06-17","Statement":[{"Effect":"Allow","Principal":{"AWS":
["arn:aws:iam::user/TESTER2"]},"Action":["sts:AssumeRole"]}]}"
```

- 역할에 연결된 업데이트 정책:

구문

```
POST "<hostname>?
Action=PutRolePolicy&RoleName=ROLE_NAME&PolicyName=POLICY_NAME&PolicyDocu
ment=TRUST_RELATIONSHIP_POLICY_DOCUMENT"
```

예제

```
POST "<hostname>?
Action=PutRolePolicy&RoleName=S3Access&PolicyName=Policy1&PolicyDocument=
{"Version":"2022-06-17","Statement":[{"Effect":"Allow","Action":
["s3:CreateBucket"],"Resource":"arn:aws:s3:::example_bucket"}]}"
```

- 역할에 연결된 권한 정책 이름을 나열합니다.

구문

```
POST "<hostname>?Action=ListRolePolicies&RoleName=ROLE_NAME"
```

예제

```
POST "<hostname>?Action=ListRolePolicies&RoleName=S3Access"
```

```
<PolicyNames>
  <member>Policy1</member>
</PolicyNames>
```

- 역할에 연결된 권한 정책을 가져옵니다.

구문

```
POST "<hostname>?
Action=GetRolePolicy&RoleName=ROLE_NAME&PolicyName=POLICY_NAME"
```

예제

```
POST "<hostname>?Action=GetRolePolicy&RoleName=S3Access&PolicyName=Policy1"
```

```
<GetRolePolicyResult>
  <PolicyName>Policy1</PolicyName>
  <RoleName>S3Access</RoleName>
  <Permission_policy>{"Version":"2022-06-17","Statement":[{"Effect":"Allow","Action":
["s3:CreateBucket"],"Resource":"arn:aws:s3:::example_bucket"}]}</Permission_policy>
</GetRolePolicyResult>
```

- 역할에 연결된 정책을 삭제합니다.

구문

```
POST "<hostname>?
Action>DeleteRolePolicy&RoleName=ROLE_NAME&PolicyName=POLICY_NAME"
```

예제

```
POST "<hostname>?Action=DeleteRolePolicy&RoleName=S3Access&PolicyName=Policy1"
```

- 역할 제거합니다.



참고

권한 정책이 연결된 경우에만 역할을 삭제할 수 있습니다.

구문

```
POST "<hostname>?Action=DeleteRole&RoleName=ROLE_NAME"
```

예제

```
POST "<hostname>?Action=DeleteRole&RoleName=S3Access"
```

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage Object Gateway** 가이드의 [역할 관리](#) 섹션을 참조하십시오.

A.19. NFS GANESHA

Ceph RESTful API nfs-ganesha 엔드포인트를 사용하여 **Ceph NFS** 게이트웨이를 관리하는 방법 참조합니다.

GET /api/nfs-ganesha/daemon

설명

NFS Ganesha 데몬에 대한 정보를 확인합니다.

예제

```
GET /api/nfs-ganesha/daemon HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스** 먼저 로그인하십시오.
- **403 forbidden - 인증되지 않은 액세스입니다.** 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/nfs-ganesha/export

설명

모든 **NFS Ganesha** 내보내기를 확인합니다.

예제

```
GET /api/nfs-ganesha/export HTTP/1.1
Host: example.com
```

상태 코드

•

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스** 먼저 로그인하십시오.
- **403 forbidden - 인증되지 않은 액세스입니다.** 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/nfs-ganesha/export

설명

새 **NFS Ganesha** 내보내기를 생성합니다.

예제

```
POST /api/nfs-ganesha/export HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "access_type": "STRING",
  "clients": [
    {
      "access_type": "STRING",
      "addresses": [
        "STRING"
      ],
      "squash": "STRING"
    }
  ],
  "cluster_id": "STRING",
  "daemons": [
    "STRING"
  ],
  "fsal": {
    "filesystem": "STRING",
    "name": "STRING",
    "rgw_user_id": "STRING",
    "sec_label_xattr": "STRING",
    "user_id": "STRING"
  },
  "path": "STRING",
```



```

    "protocols": [
      1
    ],
    "pseudo": "STRING",
    "reload_daemons": true,
    "security_label": "STRING",
    "squash": "STRING",
    "tag": "STRING",
    "transports": [
      "STRING"
    ]
  }

```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

DELETE /api/nfs-ganesha/export/CLUSTER_ID/EXPORT_ID

설명

NFS Ganesha 내보내기를 삭제합니다.

매개 변수

- **CLUSTER_ID** 를 스토리지 클러스터 식별자 문자열로 교체합니다.
- **EXPORT_ID** 를 정수로 내보내기 식별자로 바꿉니다.

- - **쿼리:**
 - **reload_daemons - NFS Ganesha 데몬 구성을 다시 로드하는 부울 값입니다.**

상태 코드

- **202 허용 - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.**
- **204 콘텐츠 없음 - 리소스가 삭제되었습니다.**
- **400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.**
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.**
- **500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.**

GET /api/nfs-ganesha/export/CLUSTER_ID/EXPORT_ID

설명

NFS Ganesha 내보내기 정보를 확인합니다.

매개 변수

- **CLUSTER_ID** 를 스토리지 클러스터 식별자 문자열로 교체합니다.
- **EXPORT_ID** 를 정수로 내보내기 식별자로 바꿉니다.

예제

```
GET /api/nfs-ganesha/export/CLUSTER_ID/EXPORT_ID HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.**
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

PUT /api/nfs-ganesha/export/CLUSTER_ID/EXPORT_ID

설명

NFS Ganesha 내보내기 정보를 업데이트합니다.

매개 변수

- **CLUSTER_ID** 를 스토리지 클러스터 식별자 문자열로 교체합니다.
- **EXPORT_ID** 를 정수로 내보내기 식별자로 바꿉니다.

예제

```
PUT /api/nfs-ganesha/export/CLUSTER_ID/EXPORT_ID HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "access_type": "STRING",
  "clients": [
    {
      "access_type": "STRING",
      "addresses": [
        "STRING"
      ],
      "squash": "STRING"
    }
  ]
}
```

```

    }
  ],
  "daemons": [
    "STRING"
  ],
  "fsal": {
    "filesystem": "STRING",
    "name": "STRING",
    "rgw_user_id": "STRING",
    "sec_label_xattr": "STRING",
    "user_id": "STRING"
  },
  "path": "STRING",
  "protocols": [
    1
  ],
  "pseudo": "STRING",
  "reload_daemons": true,
  "security_label": "STRING",
  "squash": "STRING",
  "tag": "STRING",
  "transports": [
    "STRING"
  ]
}

```

상태 코드

- **200 OK - 확인**
- **202 허용 - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.**
- **400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.**
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.**
- **500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.**

GET /api/nfs-ganesha/status

설명

NFS Ganesha 관리 기능의 상태 정보를 확인합니다.

예제

```
GET /api/nfs-ganesha/status HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 [CephRESTful API](#) 장을 참조하십시오.
- 자세한 내용은 **Red Hat Ceph Storage Object Gateway** 가이드의 [네임스페이스를 NFS-Ganesha로 내보내기](#) 섹션을 참조하십시오.

A.20. CEPH ORCHESTRATOR

Ceph RESTful API 오케스트레이터 엔드포인트를 사용하여 **Ceph Orchestrator** 상태를 표시하는 방법 참조입니다.

GET /api/orchestrator/status

설명

Ceph Orchestrator 상태를 표시합니다.

예제

```
GET /api/orchestrator/status HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400** 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401** 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500** 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 [CephRESTful API](#) 장을 참조하십시오.

A.21. POOL

Ceph RESTful API 풀 끝점을 사용하여 스토리지 풀을 관리하는 방법 참조입니다.

GET /api/pool

설명

풀 목록을 표시합니다.

매개 변수

- 쿼리:
 - **attrs** - 풀 속성의 문자열 값입니다.
 - **통계** - 풀 통계의 부울 값입니다.

예제

```
GET /api/pool HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/pool**예제**

```
POST /api/pool HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "application_metadata": "STRING",
  "configuration": "STRING",
  "erasure_code_profile": "STRING",
  "flags": "STRING",
  "pg_num": 1,
  "pool": "STRING",
```

```

    "pool_type": "STRING",
    "rule_name": "STRING"
  }

```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

DELETE /api/pool/POOL_NAME**매개 변수**

- **POOL_NAME** 을 풀 이름으로 교체합니다.

상태 코드

- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.

- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/pool/POOL_NAME

매개 변수

- **POOL_NAME** 을 풀 이름으로 교체합니다.
- 쿼리:
 - **attrs** - 풀 속성의 문자열 값입니다.
 - **통계** - 풀 통계의 부울 값입니다.

예제

```
GET /api/pool/POOL_NAME HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

PUT /api/pool/POOL_NAME

매개 변수

- **POOL_NAME** 을 풀 이름으로 교체합니다.

예제

```
PUT /api/pool/POOL_NAME HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "application_metadata": "STRING",
  "configuration": "STRING",
  "flags": "STRING"
}
```

상태 코드

- **200 OK** - 확인
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/pool/POOL_NAME/configuration

매개 변수

- **POOL_NAME** 을 풀 이름으로 교체합니다.

예제

```
GET /api/pool/POOL_NAME/configuration HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 [CephRESTful API](#) 장을 참조하십시오.

A.22. PROMETHEUS

Ceph RESTful API prometheus 끝점을 사용하여 **Prometheus**를 관리하는 방법 참조입니다.

GET /api/prometheus

예제

```
GET /api/prometheus/rules HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인

- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스** 먼저 로그인하십시오.
- **403 forbidden - 인증되지 않은 액세스입니다.** 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/prometheus/rules

예제

```
GET /api/prometheus/rules HTTP/1.1  
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스** 먼저 로그인하십시오.
- **403 forbidden - 인증되지 않은 액세스입니다.** 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/prometheus/silence

상태 코드

- **201 생성 - 리소스가 생성되었습니다.**

- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

DELETE /api/prometheus/silence/S_ID

매개 변수

- **S_ID** 를 문자열 값으로 바꿉니다.

상태 코드

- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/prometheus/silences**예제**

```
GET /api/prometheus/silences HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/prometheus/authorizations**예제**

```
GET /api/prometheus/notifications HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
-

500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 [CephRESTful API](#) 장을 참조하십시오.

A.23. RADOS 블록 장치

Ceph RESTful API 블록 끝점을 사용하여 **RADOS** 블록 장치(**RBD**)를 관리하는 방법 참조입니다. 이 참조에는 다음과 같이 사용 가능한 모든 **RBD** 기능 엔드 포인트가 포함됩니다.

- [RBD 네임스페이스](#)
- [RBD 스냅샷](#)
- [RBD 방지](#)
- [RBD 미러링](#)
 - [RBD 미러링 요약](#)
 - [RBD Mirroring Pool Bootstrap](#)
 - [RBD 미러링 풀 모드](#)
 - [RBD 미러링 풀 피어](#)

RBD 이미지

GET /api/block/image

설명

RBD 이미지를 확인합니다.

매개 변수

- **쿼리:**
 - **pool_name** - 문자열 풀 이름입니다.

예제

```
GET /api/block/image HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/block/image

예제

```
POST /api/block/image HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "configuration": "STRING",
  "data_pool": "STRING",
  "features": "STRING",
  "name": "STRING",
```



```

    "namespace": "STRING",
    "obj_size": 1,
    "pool_name": "STRING",
    "size": 1,
    "stripe_count": 1,
    "stripe_unit": "STRING"
  }

```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/block/image/clone_format_version

설명

RBD 복제 형식 버전을 반환합니다.

예제

```

GET /api/block/image/clone_format_version HTTP/1.1
Host: example.com

```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.

- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/block/image/default_features

예제

```
GET /api/block/image/default_features HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/block/image/default_features

예제

```
GET /api/block/image/default_features HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인

- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

DELETE /api/block/image/IMAGE_SPEC

매개 변수

- **IMAGE_SPEC** 를 문자열 값으로 이미지 이름으로 교체합니다.

상태 코드

- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/block/image/IMAGE_SPEC

매개 변수

•

IMAGE_SPEC 를 문자열 값으로 이미지 이름으로 교체합니다.

예제

```
GET /api/block/image/IMAGE_SPEC HTTP/1.1
Host: example.com
```

상태 코드

•

200 OK - 확인

•

400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.

•

401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.

•

403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.

•

500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

PUT /api/block/image/IMAGE_SPEC

매개 변수

•

IMAGE_SPEC 를 문자열 값으로 이미지 이름으로 교체합니다.

예제

```
PUT /api/block/image/IMAGE_SPEC HTTP/1.1
Host: example.com
Content-Type: application/json
```

```
{
  "configuration": "STRING",
  "features": "STRING",
  "name": "STRING",
  "size": 1
}
```

상태 코드

- **200 OK - 확인**
- **202 허용 - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.**
- **400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.**
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.**
- **500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.**

POST /api/block/image/IMAGE_SPEC/copy

매개 변수

- **IMAGE_SPEC** 를 문자열 값으로 이미지 이름으로 교체합니다.

예제

```
POST /api/block/image/IMAGE_SPEC/copy HTTP/1.1
Host: example.com
Content-Type: application/json
```

```
{
  "configuration": "STRING",
  "data_pool": "STRING",
  "dest_image_name": "STRING",
  "dest_namespace": "STRING",
  "dest_pool_name": "STRING",
  "features": "STRING",
  "obj_size": 1,
  "snapshot_name": "STRING",
  "stripe_count": 1,
  "stripe_unit": "STRING"
}
```

상태 코드

- **201 생성 - 리소스가 생성되었습니다.**
- **202 허용 - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.**
- **400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.**
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.**
- **500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.**

POST /api/block/image/IMAGE_SPEC/flatten

매개 변수

- **IMAGE_SPEC** 를 문자열 값으로 이미지 이름으로 교체합니다.

상태 코드

- **201 생성 - 리소스가 생성되었습니다.**
- **202 허용 - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.**
- **400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.**
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.**
- **500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.**

시오.

POST /api/block/image/IMAGE_SPEC/move_trash

설명

이미지를 휴지통으로 이동합니다. 클론에 의해 적극적으로 사용 중인 이미지는 기저체로 이동할 수 있으며 나중에 삭제할 수 있습니다.

매개 변수

- **IMAGE_SPEC** 를 문자열 값으로 이미지 이름으로 교체합니다.

예제

```
POST /api/block/image/IMAGE_SPEC/move_trash HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "delay": 1
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

RBD 미러링

GET /api/block/mirroring/site_name

설명

RBD 미러링 사이트 이름을 표시합니다.

예제

```
GET /api/block/mirroring/site_name HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다.** 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

PUT /api/block/mirroring/site_name

예제

```
PUT /api/block/mirroring/site_name HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "site_name": "STRING"
}
```

상태 코드

- **200 OK - 확인**
- -

- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

RBD Mirroring Pool Bootstrap

POST /api/block/mirroring/pool/POOL_NAME/bootstrap/peer

매개 변수

- **POOL_NAME** 을 문자열로 풀 이름으로 교체합니다.

예제

```
POST /api/block/mirroring/pool/POOL_NAME/bootstrap/peer HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "direction": "STRING",
  "token": "STRING"
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.

- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/block/mirroring/pool/POOL_NAME/bootstrap/token

매개 변수

- **POOL_NAME** 을 문자열로 풀 이름으로 교체합니다.

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

RBD 미러링 풀 모드

GET /api/block/mirroring/pool/POOL_NAME

설명

RBD 미러링 요약을 표시합니다.

매개 변수

- **POOL_NAME** 을 문자열로 풀 이름으로 교체합니다.

예제

```
GET /api/block/mirroring/pool/POOL_NAME HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

PUT /api/block/mirroring/pool/POOL_NAME

매개 변수

- **POOL_NAME** 을 문자열로 풀 이름으로 교체합니다.

예제

```
PUT /api/block/mirroring/pool/POOL_NAME HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "mirror_mode": "STRING"
}
```

상태 코드

- **200 OK** - 확인
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

RBD 미러링 풀 피어**GET /api/block/mirroring/pool/POOL_NAME/peer**

매개 변수

- **POOL_NAME** 을 문자열로 풀 이름으로 교체합니다.

예제

```
GET /api/block/mirroring/pool/POOL_NAME/peer HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.

- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/block/mirroring/pool/POOL_NAME/peer

매개 변수

- **POOL_NAME** 을 문자열로 풀 이름으로 교체합니다.

예제

```
POST /api/block/mirroring/pool/POOL_NAME/peer HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "client_id": "STRING",
  "cluster_name": "STRING",
  "key": "STRING",
  "mon_host": "STRING"
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

DELETE /api/block/mirroring/pool/POOL_NAME/peer/PEER_UUID

매개 변수

- **POOL_NAME** 을 문자열로 풀 이름으로 교체합니다.
- **PEER_UUID** 를 문자열로 피어의 **UUID**로 바꿉니다.

상태 코드

- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/block/mirroring/pool/POOL_NAME/peer/PEER_UUID

매개 변수

- **POOL_NAME** 을 문자열로 풀 이름으로 교체합니다.
- **PEER_UUID** 를 문자열로 피어의 **UUID**로 바꿉니다.

예제

```
GET /api/block/mirroring/pool/POOL_NAME/peer/PEER_UUID HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

```
PUT /api/block/mirroring/pool/POOL_NAME/peer/PEER_UUID
```

매개 변수

- **POOL_NAME** 을 문자열로 풀 이름으로 교체합니다.
- **PEER_UUID** 를 문자열로 피어의 UUID로 바꿉니다.

예제

```
PUT /api/block/mirroring/pool/POOL_NAME/peer/PEER_UUID HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "client_id": "STRING",
  "cluster_name": "STRING",
  "key": "STRING",
  "mon_host": "STRING"
}
```

상태 코드

- **200 OK - 확인**
- **202 허용 - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.**
- **400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.**
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.**
- **500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.**

RBD 미러링 요약

GET /api/block/mirroring/summary

설명

RBD 미러링 요약을 표시합니다.

예제

```
GET /api/block/mirroring/summary HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.**
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**

- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

RBD 네임스페이스

GET /api/block/pool/POOL_NAME/namespace

매개 변수

- **POOL_NAME** 을 문자열로 풀 이름으로 교체합니다.

예제

```
GET /api/block/pool/POOL_NAME/namespace HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/block/pool/POOL_NAME/namespace

매개 변수

- **POOL_NAME** 을 문자열로 풀 이름으로 교체합니다.

예제

```
POST /api/block/pool/POOL_NAME/namespace HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "namespace": "STRING"
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

DELETE /api/block/pool/POOL_NAME/namespace/NAMESPACE

매개 변수

- **POOL_NAME** 을 문자열로 풀 이름으로 교체합니다.
- **NAMESPACE** 를 문자열로 대체합니다.

상태 코드

- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.

- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

RBD 스냅샷

POST /api/block/image/IMAGE_SPEC/snap

매개 변수

- **IMAGE_SPEC** 를 문자열 값으로 이미지 이름으로 교체합니다.

예제

```
POST /api/block/image/IMAGE_SPEC/snap HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "snapshot_name": "STRING"
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.

- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

DELETE /api/block/image/IMAGE_SPEC/snap/SNAPSHOT_NAME

매개 변수

- **IMAGE_SPEC** 를 문자열 값으로 이미지 이름으로 교체합니다.
- **SNAPSHOT_NAME** 을 스냅샷의 이름으로 문자열 값으로 바꿉니다.

상태 코드

- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

PUT /api/block/image/IMAGE_SPEC/snap/SNAPSHOT_NAME

매개 변수

- **IMAGE_SPEC** 를 문자열 값으로 이미지 이름으로 교체합니다.
- **SNAPSHOT_NAME** 을 스냅샷의 이름으로 문자열 값으로 바꿉니다.

예제

```
PUT /api/block/image/IMAGE_SPEC/snap/SNAPSHOT_NAME HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "is_protected": true,
  "new_snap_name": "STRING"
}
```

상태 코드

- **200 OK** - 확인
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/block/image/IMAGE_SPEC/snap/SNAPSHOT_NAME/clone

설명

스냅샷을 이미지에 복제합니다.

매개 변수

- **IMAGE_SPEC** 를 문자열 값으로 이미지 이름으로 교체합니다.
- **SNAPSHOT_NAME** 을 스냅샷의 이름으로 문자열 값으로 바꿉니다.

예제

```
POST /api/block/image/IMAGE_SPEC/snap/SNAPSHOT_NAME/clone HTTP/1.1
Host: example.com
Content-Type: application/json
```

```
{
  "child_image_name": "STRING",
  "child_namespace": "STRING",
  "child_pool_name": "STRING",
  "configuration": "STRING",
  "data_pool": "STRING",
  "features": "STRING",
  "obj_size": 1,
  "stripe_count": 1,
  "stripe_unit": "STRING"
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

```
POST /api/block/image/IMAGE_SPEC/snap/SNAPSHOT_NAME/rollback
```

매개 변수

- **IMAGE_SPEC** 를 문자열 값으로 이미지 이름으로 교체합니다.
- **SNAPSHOT_NAME** 을 스냅샷의 이름으로 문자열 값으로 바꿉니다.

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

RBD Trash**GET /api/block/image/trash**

설명

풀 이름별로 모든 **RBD** 항목 또는 **RBD** 휴지통 세부 정보를 표시합니다.

매개 변수

- 쿼리:
 - **pool_name** - 문자열 값으로 풀의 이름입니다.

예제

```
GET /api/block/image/trash HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/block/image/trash/purge

설명

만료된 모든 이미지를 휴지통에서 제거합니다.

매개 변수

- 쿼리:
 - **pool_name** - 문자열 값으로 풀의 이름입니다.

예제

```
POST /api/block/image/trash/purge HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "pool_name": "STRING"
}
```


상태 코드

- **201 생성 - 리소스가 생성되었습니다.**
- **202 허용 - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.**
- **400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.**
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.**
- **500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.**

DELETE /api/block/image/trash/IMAGE_ID_SPEC

설명

휴지통에서 이미지를 삭제합니다. 이미지 지연 시간이 만료되지 않은 경우 강제로 사용하지 않는 한 삭제할 수 없습니다. 복제본에 의해 적극적으로 사용되거나 스냅샷이 있는 이미지는 삭제할 수 없습니다.

매개 변수

- **IMAGE_ID_SPEC** 를 문자열 값으로 교체하십시오.
- 쿼리:
 - **force - bin**에서 이미지를 강제로 삭제할 수 있는 부울 값입니다.

상태 코드

- **202 허용 - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.**

- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/block/image/trash/IMAGE_ID_SPEC/restore

설명

휴지통에서 이미지를 복원합니다.

매개 변수

- **IMAGE_ID_SPEC** 를 문자열 값으로 교체하십시오.

예제

```
POST /api/block/image/trash/IMAGE_ID_SPEC/restore HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "new_image_name": "STRING"
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.

- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스** 먼저 로그인하십시오.
- **403 forbidden - 인증되지 않은 액세스입니다.** 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 [CephRESTful API](#) 장을 참조하십시오.

A.24. 성능 카운터

Ceph RESTful API perf_counters 엔드포인트를 사용하여 다양한 **Ceph** 성능 카운터를 표시하는 방법 참조입니다. 이 참조에는 다음과 같은 사용 가능한 모든 성능 카운터 끝점이 포함됩니다.

- [Ceph Metadata Server \(MDS\)](#)
- [Ceph Manager](#)
- [Ceph Monitor](#)
- [Ceph OSD](#)
- [Ceph 오브젝트 게이트웨이](#)
- [Ceph RADOS 블록 장치\(RBD\) 미러링](#)

-

ProcessMU Runner

GET /api/perf_counters

설명

성능 카운터를 표시합니다.

예제

```
GET /api/perf_counters HTTP/1.1
Host: example.com
```

상태 코드

-

200 OK - 확인

-

400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.

-

401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.

-

403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.

-

500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

Ceph 메타데이터 서버

GET /api/perf_counters/mds/SERVICE_ID

매개 변수

-

SERVICE_ID 를 문자열로 바꿉니다.

예제

```
GET /api/perf_counters/mds/SERVICE_ID HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다.** 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

Ceph Manager

GET /api/perf_counters/mgr/SERVICE_ID

매개 변수

- **SERVICE_ID** 를 문자열로 바꿉니다.

예제

```
GET /api/perf_counters/mgr/SERVICE_ID HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다.** 권한을 확인해 주십시오.
-

500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

Ceph Monitor

GET /api/perf_counters/mon/SERVICE_ID

매개 변수

- **SERVICE_ID** 를 문자열로 바꿉니다.

예제

```
GET /api/perf_counters/mon/SERVICE_ID HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.**
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.**
- **500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.**

Ceph OSD

GET /api/perf_counters/osd/SERVICE_ID

매개 변수

- **SERVICE_ID** 를 문자열로 바꿉니다.

예제

```
GET /api/perf_counters/osd/SERVICE_ID HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다.** 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

Ceph RADOS 블록 장치(RBD) 미러링

```
GET /api/perf_counters/rbd-mirror/SERVICE_ID
```

매개 변수

- **SERVICE_ID** 를 문자열로 바꿉니다.

예제

```
GET /api/perf_counters/rbd-mirror/SERVICE_ID HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**

- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

Ceph 오브젝트 게이트웨이

GET /api/perf_counters/rgw/SERVICE_ID

매개 변수

- **SERVICE_ID** 를 문자열로 바꿉니다.

예제

```
GET /api/perf_counters/rgw/SERVICE_ID HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

ProcessMU Runner

GET /api/perf_counters/tcmu-runner/SERVICE_ID

매개 변수

- **SERVICE_ID** 를 문자열로 바꿉니다.

예제

```
GET /api/perf_counters/tcmu-runner/SERVICE_ID HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 [CephRESTful API](#) 장을 참조하십시오.

A.25. 역할

Ceph RESTful API 역할 끝점을 사용하여 **Ceph**에서 다양한 사용자 역할을 관리하는 방법 참조입니다.

GET /api/role

설명

역할 목록을 표시합니다.

예제

```
GET /api/role HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/role**예제**

```
POST /api/role HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "description": "STRING",
  "name": "STRING",
  "scopes_permissions": "STRING"
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.

- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

DELETE /api/role/NAME

매개 변수

- **NAME** 을 문자열로 바꿉니다.

상태 코드

- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/role/NAME

매개 변수

- **NAME** 을 문자열로 바꿉니다.

예제

```
GET /api/role/NAME HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

PUT /api/role/NAME

매개 변수

- **NAME** 을 문자열로 바꿉니다.

예제

```
PUT /api/role/NAME HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "description": "STRING",
  "scopes_permissions": "STRING"
}
```

상태 코드

- **200 OK** - 확인

- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/role/NAME/clone

매개 변수

- **NAME** 을 문자열로 바꿉니다.

예제

```
POST /api/role/NAME/clone HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "new_name": "STRING"
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.

- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 [CephRESTful API](#) 장을 참조하십시오.

A.26. 서비스

Ceph RESTful API 서비스 엔드포인트를 사용하여 다양한 **Ceph** 서비스를 관리하는 방법 참조입니다.

GET /api/service

매개 변수

- 쿼리:
 - **service_name** - 서비스 이름을 문자열로 나타냅니다.

예제

```
GET /api/service HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.

- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/service

매개 변수

- **service_spec** - JSON 파일로 서비스 사양입니다.
- **service_name** - 서비스의 이름입니다.

예제

```
POST /api/service HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "service_name": "STRING",
  "service_spec": "STRING"
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

시오.

GET /api/service/known_types

설명

알려진 서비스 유형 목록을 표시합니다.

예제

```
GET /api/service/known_types HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

DELETE /api/service/SERVICE_NAME

매개 변수

- **SERVICE_NAME** 을 문자열로 서비스 이름으로 교체합니다.

상태 코드

- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.

- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스** 먼저 로그인하십시오.
- **403 forbidden - 인증되지 않은 액세스입니다.** 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/service/SERVICE_NAME

매개 변수

- **SERVICE_NAME** 을 문자열로 서비스 이름으로 교체합니다.

예제

```
GET /api/service/SERVICE_NAME HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스** 먼저 로그인하십시오.
- **403 forbidden - 인증되지 않은 액세스입니다.** 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/service/SERVICE_NAME/daemons

매개 변수

- **`SERVICE_NAME`** 을 문자열로 서비스 이름으로 교체합니다.

예제

```
GET /api/service/SERVICE_NAME/daemons HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 [CephRESTful API](#) 장을 참조하십시오.

A.27. 설정

Ceph RESTful API 설정 끝점을 사용하여 다양한 **Ceph** 설정을 관리하는 방법 참조입니다.

GET /api/settings

설명

사용 가능한 옵션 목록 표시

매개 변수

- 쿼리:
 - **names** - 쉼표로 구분된 옵션 이름 목록입니다.

예제

```
GET /api/settings HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

PUT /api/settings

상태 코드

- **200 OK** - 확인
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.

- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

DELETE /api/settings/NAME

매개 변수

- **NAME** 을 문자열 이름으로 바꿉니다.

상태 코드

- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/settings/NAME

설명

지정된 옵션을 표시합니다.

매개 변수

- **NAME** 을 문자열 이름으로 바꿉니다.

예제

```
GET /api/settings/NAME HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

PUT /api/settings/NAME**매개 변수**

- **NAME** 을 문자열 이름으로 바꿉니다.

예제

```
PUT /api/settings/NAME HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "value": "STRING"
}
```

상태 코드

- **200 OK** - 확인

- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 [CephRESTful API](#) 장을 참조하십시오.

A.28. CEPH 작업

Ceph RESTful API 작업 끝점을 사용하여 **Ceph** 작업을 표시하는 방법 참조입니다.

GET /api/task

설명

Ceph 작업을 표시합니다.

매개 변수

- 쿼리:
 - **name** - 작업의 이름입니다.

예제

```
GET /api/task HTTP/1.1
Host: example.com
```

■

상태 코드

- **200 OK** - 확인
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 [CephRESTful API](#) 장을 참조하십시오.

A.29. TELEMETRY

Ceph RESTful API Telemetry 엔드포인트를 사용하여 원격 분석 **Ceph Manager** 모듈의 데이터를 관리하는 방법 참조입니다.

PUT /api/telemetry

설명

Telemetry 모듈을 통해 수집된 데이터 전송을 활성화하거나 비활성화합니다.

매개 변수

- **Enable** - 부울 값을 지정합니다.
- **license_name** - 과 같은 문자열 값 (예: **sharing-1-0**)입니다. 사용자가 원격 분석 데이터를 공유하는 라이선스를 인식하고 수락하는지 확인합니다.

예제

```
PUT /api/telemetry HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "enable": true,
  "license_name": "STRING"
}
```

상태 코드

- **200 OK - 확인**
- **202 허용 - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.**
- **400 잘못된 요청 - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.**
- **401 승인되지 않음 - 인증되지 않은 액세스 먼저 로그인하십시오.**
- **403 forbidden - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.**
- **500 내부 서버 오류 - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.**

GET /api/telemetry/report**설명**

Ceph 및 장치에 대한 보고서 데이터를 표시합니다.

예제

```
GET /api/telemetry/report HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**

- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스** 먼저 로그인하십시오.
- **403 forbidden - 인증되지 않은 액세스입니다.** 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 [CephRESTful API](#) 장을 참조하십시오.

A.30. CEPH 사용자

Ceph RESTful API 사용자 엔드포인트를 사용하여 **Ceph** 사용자 세부 정보를 표시하고 **Ceph** 사용자 암호를 관리하는 방법 참조입니다.

GET /api/user

설명

사용자 목록을 표시합니다.

예제

```
GET /api/user HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.

- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/user**예제**

```
POST /api/user HTTP/1.1
Host: example.com
Content-Type: application/json
```

```
{
  "email": "STRING",
  "enabled": true,
  "name": "STRING",
  "password": "STRING",
  "pwdExpirationDate": "STRING",
  "pwdUpdateRequired": true,
  "roles": "STRING",
  "username": "STRING"
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

시오.

DELETE /api/user/USER_NAME

매개 변수

- **USER_NAME** 을 사용자 이름으로 문자열로 바꿉니다.

상태 코드

- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **204 콘텐츠 없음** - 리소스가 삭제되었습니다.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

GET /api/user/USER_NAME

매개 변수

- **USER_NAME** 을 사용자 이름으로 문자열로 바꿉니다.

예제

```
GET /api/user/USER_NAME HTTP/1.1
Host: example.com
```

상태 코드

- **200 OK - 확인**
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스** 먼저 로그인하십시오.
- **403 forbidden - 인증되지 않은 액세스**입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

PUT /api/user/USER_NAME

매개 변수

- **USER_NAME** 을 사용자 이름으로 문자열로 바꿉니다.

예제

```
PUT /api/user/USER_NAME HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "email": "STRING",
  "enabled": "STRING",
  "name": "STRING",
  "password": "STRING",
  "pwdExpirationDate": "STRING",
  "pwdUpdateRequired": true,
  "roles": "STRING"
}
```

상태 코드

- **200 OK - 확인**
- **202 허용 - 작업이 실행 중입니다.** 작업 대기열을 확인하십시오.

- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스** 먼저 로그인하십시오.
- **403 forbidden - 인증되지 않은 액세스입니다.** 권한을 확인해 주십시오.
- **500 내부 서버 오류 - 예상되지 않은 오류.** 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/user/USER_NAME/change_password

매개 변수

- **USER_NAME** 을 사용자 이름으로 문자열로 바꿉니다.

예제

```
POST /api/user/USER_NAME/change_password HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "new_password": "STRING",
  "old_password": "STRING"
}
```

상태 코드

- **201 생성 - 리소스가 생성되었습니다.**
- **202 허용 - 작업이 실행 중입니다.** 작업 대기열을 확인하십시오.
- **400 잘못된 요청 - 작동 예외.** 자세한 내용은 응답 본문을 확인하십시오.
- **401 승인되지 않음 - 인증되지 않은 액세스** 먼저 로그인하십시오.

- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

POST /api/user/validate_password

설명

암호가 암호 정책을 충족하는지 확인합니다.

매개 변수

- **password** - 유효성을 검사할 암호입니다.
- **username** - 선택 사항입니다. 사용자의 이름입니다.
- **old_password** - Optional. 이전 암호입니다.

예제

```
POST /api/user/validate_password HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "old_password": "STRING",
  "password": "STRING",
  "username": "STRING"
}
```

상태 코드

- **201 생성** - 리소스가 생성되었습니다.
- **202 허용** - 작업이 실행 중입니다. 작업 대기열을 확인하십시오.
- **400 잘못된 요청** - 작동 예외. 자세한 내용은 응답 본문을 확인하십시오.

- **401 승인되지 않음** - 인증되지 않은 액세스 먼저 로그인하십시오.
- **403 forbidden** - 인증되지 않은 액세스입니다. 권한을 확인해 주십시오.
- **500 내부 서버 오류** - 예상되지 않은 오류. 스택 추적에 대한 응답 본문을 확인하십시오.

추가 리소스

- 자세한 내용은 **Red Hat Ceph Storage** 개발자 가이드의 [CephRESTful API](#) 장을 참조하십시오.

부록 B. S3 일반 요청 헤더

다음 표에는 유효한 공통 요청 헤더와 설명이 나열되어 있습니다.

표 B.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 C. S3 일반적인 응답 상태 코드

다음 표에는 유효한 공통 **HTTP** 응답 상태 및 해당 코드가 나열되어 있습니다.

표 C.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	InternalServerError

부록 D. S3 지원되지 않는 헤더 필드

표 D.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 E. SWIFT 요청 헤더

표 E.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 F. 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 F.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 G. 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 TESTER2는 역할에 연결된 권한 정책에 따라 TESTER1이 소유한 S3 리소스에 액세스하는 것처럼 TESTER1에서 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=""
)

oidc_response = iam_client.create_open_id_connect_provider(
    Url=<URL of the OpenID Connect Provider>,
    ClientIDList=[
        <Client id registered with the IDP>
    ],
    ThumbprintList=[
        <IDP THUMBPRINT>
    ]
)

policy_document = "{ \"Version\": \"2012-10-17\", \"Statement\": [ { \"Effect\": \"Allow\", \"Principal\": { \"Federated\": \"arn:aws:iam::oidc-provider:localhost:8080/auth/realms/demo\" }, \"Action\": [ \"sts:AssumeRoleWithWebIdentity\" ], \"Condition\": { \"StringEquals\": { \"localhost:8080/auth/realms/demo:app_id\": \"customer-portal\" } } } ] }"
role_response = iam_client.create_role(
    AssumeRolePolicyDocument=policy_document,
    Path='/',

```

```

    RoleName='S3Access',
)

role_policy = "{ \"Version\": \"2012-10-17\", \"Statement\": [ { \"Effect\": \"Allow\", \"Action\": \"s3:*\", \"Resource\": \"arn:aws:s3:::*\" } ] }"

response = iam_client.put_role_policy(
    RoleName='S3Access',
    PolicyName='Policy1',
    PolicyDocument=role_policy
)

sts_client = boto3.client('sts',
    aws_access_key_id=ACCESS_KEY_OF_TESTER2,
    aws_secret_access_key=SECRET_KEY_OF_TESTER2,
    endpoint_url=<STS URL>,
    region_name="",
)

response = 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.