



# Red Hat Ceph Storage 3

## Ubuntu のオブジェクトゲートウェイガイド

Ubuntu での Ceph Storage Object Gateway のインストール、設定、および管理



# Red Hat Ceph Storage 3 Ubuntu のオブジェクトゲートウェイガイド

---

Ubuntu での Ceph Storage Object Gateway のインストール、設定、および管理

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.

## 法律上の通知

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Object\_Gateway\_Guide\_for\_Ubuntu.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<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

Java<sup>®</sup> is a registered trademark of Oracle and/or its affiliates.

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

MySQL<sup>®</sup> is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js<sup>®</sup> 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<sup>®</sup> 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.

## 概要

このドキュメントでは、AMD64 および Intel 64 アーキテクチャーで実行されている Ubuntu 14.04 で Ceph Storage Object Gateway をインストール、設定、および管理する手順を説明します。

## 目次

第1章 概要 .....	5
第2章 設定 .....	6
2.1. CIVETWEB フロントエンド	6
2.2. CIVETWEB ポートの変更	6
2.3. CIVETWEB での SSL の使用	8
2.4. CIVETWEB 設定オプション	8
2.5. BEAST フロントエンドの使用	9
2.6. BEAST 設定オプション	10
2.7. DNS へのワイルドカードの追加	11
2.8. ログインおよびデバッグ出力の調整	12
2.9. S3 API サーバー側の暗号化	12
2.10. ゲートウェイのテスト	13
2.10.1. S3 ユーザーの作成	13
2.10.2. Swift ユーザーの作成	15
2.10.3. S3 アクセスのテスト	17
2.10.4. Swift アクセスのテスト	18
2.11. HAPROXY/KEEPALIVED の設定	19
2.11.1. HAProxy/keepalived の前提条件	19
2.11.2. HAProxy ノードの準備	19
2.11.3. keepalived のインストールと設定	20
2.11.4. HAProxy のインストールおよび設定	23
2.11.5. HAProxy 設定のテスト	24
2.12. 静的 WEB ホスティング用のゲートウェイの設定	25
2.12.1. 静的 Web ホストの前提条件	25
2.12.2. 静的 Web ホストの要件	25
2.12.3. 静的 Web ホストゲートウェイの設定	25
2.12.4. 静的 Web ホスティング DNS 設定	26
2.12.5. 静的 Web ホストサイトの作成	27
2.13. NFS-GANESHA への名前空間のエクスポート	28
第3章 管理 .....	34
3.1. 管理データストレージ	34
3.2. ストレージポリシーの作成	35
3.3. インデックスのないバケットの作成	37
3.4. バケットシャーディングの設定	39
3.4.1. バケットシャーディングの制限	39
3.4.2. 簡易設定でのバケットインデックスシャードの設定	39
3.4.3. マルチサイト設定でのバケットインデックスのシャード化の設定	40
3.4.4. バケットインデックスの動的再シャーディング	42
3.4.5. バケットインデックスの手動再シャーディング	44
3.4.6. 再シャーディングを行った後に古いインスタンスの消去	45
3.5. 圧縮の有効化	45
3.6. ユーザー管理	47
3.6.1. マルチテナンシー	48
3.6.2. ユーザーの作成	48
3.6.3. サブユーザーの作成	49
3.6.4. ユーザー情報の取得	50
3.6.5. ユーザー情報の変更	50
3.6.6. ユーザーの有効化および一時停止	51
3.6.7. ユーザーの削除	51

3.6.8. サブユーザーの削除	51
3.6.9. ユーザーの名前変更	52
3.6.10. キーの作成	54
3.6.11. アクセスキーの追加および削除	55
3.6.12. 管理機能の追加と削除	56
3.7. クォータ管理	56
3.7.1. ユーザークォータの設定	57
3.7.2. ユーザークォータの有効化および無効化	57
3.7.3. バケットクォータの設定	57
3.7.4. バケットクォータの有効化および無効化	57
3.7.5. クォータ設定の取得	58
3.7.6. クォータウォーターを更新	58
3.7.7. ユーザークォータ使用統計の取得	58
3.7.8. クォータキャッシュ	58
3.7.9. グローバルクォータの読み取りおよび作成	58
3.8. 用途	59
3.8.1. 使用方法の表示	59
3.8.2. トリムの使用方法	59
3.8.3. 孤立したオブジェクトの検索	60
3.9. バケット管理	60
3.9.1. バケットの移動	60
3.9.1.1. 前提条件	61
3.9.1.2. テナントのないユーザー間でのバケットの移動	61
3.9.1.3. テナントユーザー間でのバケットの移動	62
3.9.1.4. バケットをテナントのないユーザーからテナントユーザーに移動する	63
3.9.2. バケットの名前変更	64
3.9.3. 関連情報	65
3.10. CEPH OBJECT GATEWAY のガベージコレクションの最適化	66
3.10.1. ガベージコレクションキューの表示	66
3.10.2. 削除が多いワークロードのガベージコレクションの調整	66
<b>第4章 設定リファレンス</b>	<b>68</b>
4.1. 一般設定	68
4.2. プールについて	71
4.3. SWIFT 設定	73
4.4. ログ設定	73
4.5. KEYSTONE 設定	75
4.6. LDAP 設定	75
<b>第5章 マルチサイト</b>	<b>77</b>
5.1. 要件および前提条件	78
5.2. POOLS	78
5.3. OBJECT GATEWAY のインストール	79
5.4. マルチサイトレルムの確立	80
5.4.1. レルムの作成	80
5.4.2. マスターゾーングループの作成	80
5.4.3. マスターゾーンの作成	81
5.4.4. デフォルトのゾーングループおよびゾーンの削除	82
5.4.5. システムユーザーの作成	83
5.4.6. 期間の更新	83
5.4.7. Ceph 設定ファイルを更新します。	83
5.4.8. ゲートウェイの開始	83
5.5. セカンダリーゾーンの確立	84

---

5.5.1. レルムのプル	84
5.5.2. 期間のプル	84
5.5.3. セカンダリーゾーンの作成	85
5.5.4. 期間の更新	86
5.5.5. Ceph 設定ファイルを更新します。	86
5.5.6. ゲートウェイの開始	86
5.6. フェイルオーバーおよび障害復旧	86
5.7. シングルサイトシステムからマルチサイトへの移行	87
5.8. マルチサイトのコマンドラインの使用方法	88
5.8.1. レルム	88
5.8.1.1. レルムの作成	89
5.8.1.2. レルムのデフォルトの設定	89
5.8.1.3. レルムの削除	89
5.8.1.4. レルムの取得	89
5.8.1.5. レルムの設定	90
5.8.1.6. レルムの一覧表示	90
5.8.1.7. レルム期間の一覧表示	90
5.8.1.8. レルムのプル	90
5.8.1.9. レルムの名前変更	90
5.8.2. ゾーングループ	91
5.8.2.1. ゾーングループの作成	91
5.8.2.2. ゾーングループをデフォルトにする	91
5.8.2.3. ゾーングループへのゾーンの追加	92
5.8.2.4. ゾーングループからのゾーンの削除	92
5.8.2.5. ゾーングループの名前変更	92
5.8.2.6. ゾーングループの削除	92
5.8.2.7. ゾーングループの一覧表示	93
5.8.2.8. ゾーングループの取得	93
5.8.2.9. ゾーングループの設定	94
5.8.2.10. ゾーングループマップの設定	95
5.8.3. ゾーン	96
5.8.3.1. ゾーン作成	96
5.8.3.2. ゾーン削除	97
5.8.3.3. ゾーン変更	98
5.8.3.4. ゾーン一覧	98
5.8.3.5. ゾーン取得	98
5.8.3.6. ゾーン設定	99
5.8.3.7. ゾーンの名前変更	99
5.9. ゾーングループとゾーン設定	99
5.10. マルチサイトでのバケットの手動再シャーディング	100
5.11. レプリケーションなしでの複数ゾーンの設定	101
5.12. 同じストレージクラスターに複数のレルムの設定	104



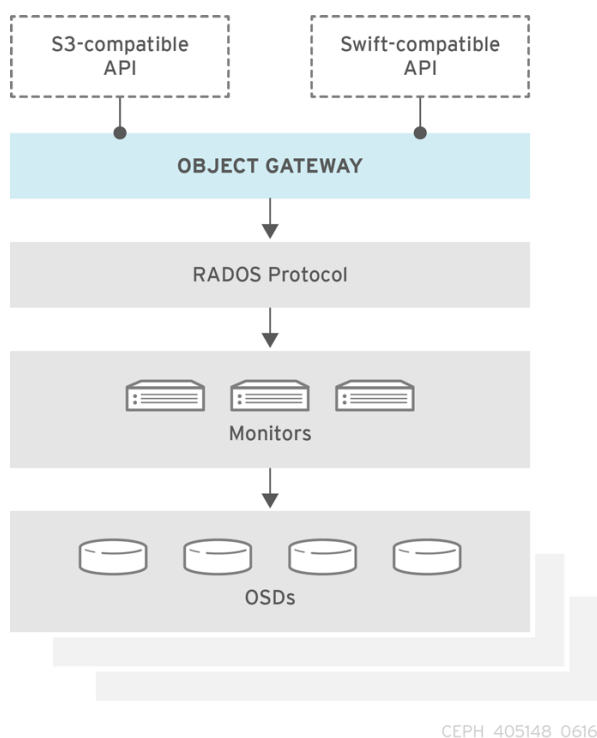


## 第1章 概要

Ceph Object Gateway は RADOS ゲートウェイ (RGW) としても知られている、**librados** 上に構築されたオブジェクトストレージインターフェイスで、RESTful ゲートウェイを Ceph ストレージクラスターに提供します。Ceph Object Gateway は以下の 2 つのインターフェイスをサポートします。

1. **S3 準拠**: Amazon S3 RESTful API の大規模なサブセットと互換性のあるインターフェイスでオブジェクトストレージ機能を提供します。
2. **Swift 準拠**: OpenStack Swift API の大規模なサブセットと互換性のあるインターフェイスでオブジェクトストレージ機能を提供します。

Ceph Object Gateway は、Ceph Storage Cluster と対話するためのサーバーです。OpenStack Swift および Amazon S3 と互換性のあるインターフェイスを提供するため、Ceph Object Gateway には独自のユーザー管理があります。Ceph Object Gateway は、Ceph ブロックデバイスクライアントからのデータを保存するために使用される同じ Ceph ストレージクラスターにデータを保存できますが、これには別個のプールが使用され、別の CRUSH 階層も使用される可能性があります。S3 および Swift API は共通の名前空間を共有するため、1 つの API でデータを作成し、これを別の API で取得することができます。



### 警告

RGW で使用されるプールで RADOS スナップショットを使用しないでください。使用すると、望ましくないデータ不整合が発生する可能性があります。

## 第2章 設定

### 2.1. CIVETWEB フロントエンド

デフォルトでは、Ceph Object Gateway は、CivetWeb Web サーバーを使用して HTTP 経由で RESTful インターフェイスを公開します。CivetWeb は、C/C++ 組み込み可能な Web サーバーです。

#### 関連情報

- [GitHub の Civetweb](#)

### 2.2. CIVETWEB ポートの変更

Ansible を使用して Ceph Object Gateway をインストールすると、ポート **8080** で実行するように CivetWeb が設定されます。Ansible は、Ceph 設定ファイルに以下のような行を追加することでこれを行います。

```
rgw frontends = civetweb port=192.168.122.199:8080 num_threads=100
```



#### 重要

Ceph 設定ファイルに **rgw frontends = civetweb** 行が含まれていない場合、Ceph Object Gateway はポート **7480** をリッスンします。**rgw frontends = civetweb** 行が含まれているがポートが指定されていない場合、Ceph Object Gateway はポート **80** をリッスンします。



#### 重要

Ansible は、Ceph Object Gateway がポート **8080** をリッスンするように設定し、Red Hat Ceph Storage 3 をインストールするためのサポート対象の方法として **ceph-ansible** を使用するため、ポート **8080** は Red Hat Ceph Storage 3 ドキュメントのデフォルトのポートとみなされます。

#### 前提条件

- Red Hat Ceph Storage 3.3 クラスタが実行されている。
- Ceph Object Gateway ノードがある。

#### 手順

1. ゲートウェイノードで、**/etc/ceph/** ディレクトリーで Ceph 設定ファイルを開きます。
2. 次の例のような RGW クライアントセクションを見つけます。

```
[client.rgw.gateway-node1]
host = gateway-node1
keyring = /var/lib/ceph/radosgw/ceph-rgw.gateway-node1/keyring
log file = /var/log/ceph/ceph-rgw.gateway-node1.log
rgw frontends = civetweb port=192.168.122.199:8080 num_threads=100
```

**[client.rgw.gateway-node1]** の見出しは、Ceph Storage Cluster クライアントを設定する時に Ceph 設定ファイルのこの部分を特定します。クライアントタイプは **rgw** で識別される Ceph Object Gateway で、ノードの名前は **gateway-node1** です。

3. デフォルトの Ansible 設定ポート **8080** を **80** に変更するには、**rgw frontends** 行を編集します。

```
rgw frontends = civetweb port=192.168.122.199:80 num_threads=100
```

**rgw\_frontends** キーと値のペアの **port=port-number** の間に空白がないことを確認します。

ポートを変更する他のゲートウェイノードでこの手順を繰り返します。

4. 各ゲートウェイノードから Ceph Object Gateway サービスを再起動して、新規ポートの設定を有効にします。

```
$ sudo systemctl restart ceph-radosgw.target
```

5. 各ゲートウェイノードのファイアウォールで、設定したポートが開いていることを確認します。

```
$ sudo iptables --list
```

6. ポートが開いていない場合は、ポートを追加します。

```
$ sudo iptables -I INPUT 1 -i iface -p tcp -s IP_address/netmask --dport 80 -j ACCEPT
```

**iface**、**IP\_address**、および **netmask** を、Ceph Object Gateway ノードに関連する値に置き換えます。

## 例

```
$ sudo iptables -I INPUT 1 -i eth0 -p tcp -s 192.168.122.199/255.255.255.0 --dport 80 -j ACCEPT
```

7. 変更を永続化して、Ceph Object Gateway ノードの再起動時に有効になるようにします。

```
$ sudo apt-get install iptables-persistent
```

8. ターミナル UI で、現在の **IPv4** iptables ルールを **/etc/iptables/rules.v4** に保存し、現在の **IPv6** iptables ルールを **/etc/iptables/rules.v6** に保存するプロンプトに対して **yes** を選択します。

9. オプション: **iptables-persistent** のインストール後に新しい **IPv4** iptables ルールを追加する場合は、それをルールファイルに追加します。その場合は、**root** ユーザーで以下のコマンドを実行してください。

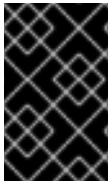
```
$ iptables-save > /etc/iptables/rules.v4
```

## 関連情報

- 詳細については、[CivetWeb での SSL の使用](#) を参照してください。
- 詳細については、[Civetweb 設定オプション](#) を参照してください。

## 2.3. CIVETWEB での SSL の使用

Red Hat Ceph Storage 1 では、Ceph Object Gateway に対する Civetweb SSL サポートは HAProxy および keepalived に依存していました。Red Hat Ceph Storage 2 以降のリリースでは、Civetweb は OpenSSL ライブラリーを使用して Transport Layer Security (TLS) を提供できます。

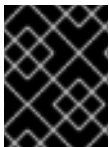


### 重要

実稼働デプロイメントでは HAProxy および keepalived を使用して HAProxy で SSL 接続を終了する **必要があります**。小規模から中規模の **テストおよび実稼働前** デプロイメントに **のみ**、Civetweb で SSL を使用することが推奨されます。

Civetweb で SSL を使用するには、ゲートウェイノードのホスト名と一致する認証局 (CA) から証明書を取得します。Red Hat は、**subject alternate name** フィールドがあり、S3-style サブドメインで使用するワイルドカードを持つ CA から証明書を取得することを推奨します。

Civetweb では、鍵、サーバー証明書、およびその他の認証局または中間証明書が 1 つの **.pem** ファイルに必要です。



### 重要

**.pem** ファイルには秘密鍵が含まれます。**.pem** ファイルを不正アクセスから保護します。

SSL にポートを設定するには、ポート番号を **rgw\_frontends** に追加し、ポート番号に **s** を追加して、セキュアなポートであることを示します。さらに、**.pem** ファイルへのパスを含む **ssl\_certificate** を追加します。以下に例を示します。

```
[client.rgw.{hostname}]
rgw_frontends = "civetweb port=443s ssl_certificate=/etc/ceph/private/server.pem"
```

## 2.4. CIVETWEB 設定オプション

以下の Civetweb 設定オプションは、RADOS Gateway の Ceph 設定ファイルの組み込み Web サーバーへ渡すことができます。それぞれのオプションにはデフォルト値があり、値が指定されていない場合は、デフォルト値が空になります。

オプション	説明	デフォルト
<b>access_log_file</b>	アクセスログ用のファイルへのパス。完全パスまたは現在の作業ディレクトリーに関連しているかのいずれか。存在しない場合 (デフォルト)、アクセスはログに記録されません。	空
<b>error_log_file</b>	エラーログ用のファイルへのパス。完全パスまたは現在の作業ディレクトリーに関連しているかのいずれか。存在しない場合 (デフォルト)、エラーはログに記録されません。	空

オプション	説明	デフォルト
<b>num_threads</b>	ワーカースレッドの数。Civetweb は、別のスレッドで各着信接続を処理します。そのため、このオプションの値は、事実上、Civetweb が処理できる同時 HTTP 接続の数になります。	512
<b>request_timeout_ms</b>	ネットワーク読み取り操作およびネットワーク書き込み操作のタイムアウト (ミリ秒単位)。クライアントが長時間実行される接続を維持する場合は、この値を増やすか、keep-alive メッセージを (より適切に) 使用します。	30000



### 重要

**num\_threads** を設定すると、**rgw\_thread\_pool\_size** が上書きされます。したがって、両方を同じ値に設定するか、**rgw\_thread\_pool\_size** のみを設定して **num\_threads** を設定しないでください。デフォルトでは、両方の変数が **ceph-ansible** によって **512** に設定されています。

このオプションの一部が設定された **/etc/ceph/ceph.conf** ファイルの例を以下に示します。

...

```
[client.rgw.node1]
rgw frontends = civetweb request_timeout_ms=30000
error_log_file=/var/log/radosgw/civetweb.error.log
access_log_file=/var/log/radosgw/civetweb.access.log
```

## 2.5. BEAST フロントエンドの使用

Ceph Object Gateway は、CivetWeb および Beast 埋め込み HTTP サーバーをフロントエンドとして提供します。Beast フロントエンドは、HTTP 解析に **Boost.Beast** ライブラリーを使用し、非同期ネットワーク I/O に **Boost.Asio** ライブラリーを使用します。CivetWeb はデフォルトのフロントエンドであるため、Beast フロントエンドを使用するには、Red Hat Ceph Storage 設定ファイルの **rgw\_frontends** パラメーターで指定します。

### 前提条件

- 稼働中の Red Hat Ceph Storage クラスタがある。
- Ceph Object Gateway がインストールされている。

### 手順

- 管理サーバーの **/etc/ceph/ceph.conf** 設定ファイルを変更します。
  - [client.rgw.<gateway-node>]** というタイトルのセクションを追加し、**<gateway-node>** を Ceph Object Gateway ノードの短いノード名に置き換えます。
  - hostname -s** を使用して、ホストの短縮名を取得します。

- c. たとえば、ゲートウェイノード名が **gateway-node1** の場合、`/etc/ceph/ceph.conf` ファイルの **[global]** セクションの後に次のようなセクションを追加します。

```
[client.rgw.gateway-node1]
rgw frontends = beast endpoint=192.168.0.100:80
```

2. 更新された設定ファイルを Ceph Object Gateway ノードおよび他の Ceph ノードにコピーします。

```
# scp /etc/ceph/ceph.conf <ceph-node>:/etc/ceph
```

3. Ceph Object Gateway を再起動して、Beast フロントエンドを有効にします。

```
# systemctl restart ceph-radosgw.target
```

4. 設定されたポートがノードのファイアウォールで開いていることを確認します。ポートが開かない場合は、ポートを追加し、ファイアウォール設定を再読み込みします。たとえば、Ceph Object Gateway ノードで、次を実行します。

```
# firewall-cmd --list-all
# firewall-cmd --zone=public --add-port 80/tcp --permanent
# firewall-cmd --reload
```

## 関連情報

- 詳細については、[Beast 設定オプション](#) を参照してください。

## 2.6. BEAST 設定オプション

以下の Beast 設定オプションは、RADOS Gateway の Ceph 設定ファイルの組み込み Web サーバーに渡すことができます。それぞれのオプションにはデフォルト値があります。値の指定がない場合は、デフォルト値が空になります。

オプション	説明	デフォルト
<b>endpoint</b> および <b>ssl_endpoint</b>	<b>address[:port]</b> 形式のリスニングアドレスを設定します。アドレスはドット付き 10 進数形式の IPv4 アドレス文字列、または 16 進数表記の IPv6 アドレスが角括弧で囲まれている IPv6 アドレスです。任意のポートのデフォルトは、 <b>endpoint</b> が <b>80</b> になり、 <b>ssl_endpoint</b> が <b>443</b> になります。 <b>endpoint=[::1]</b> <b>endpoint=192.168.0.100:8000</b> のように複数回指定できます。	空
<b>ssl_certificate</b>	SSL 対応のエンドポイントに使用する SSL 証明書ファイルへのパス。	空
<b>ssl_private_key</b>	SSL 対応のエンドポイントに使用される秘密鍵ファイルへのオプションのパス。 <b>ssl_certificate</b> で指定されているファイルがない場合は、秘密鍵として使用されます。	空

## SSL を使用する Beast オプションのある `/etc/ceph/ceph.conf` ファイルの例:

```
...

[client.rgw.node1]
rgw frontends = beast ssl_endpoint=192.168.0.100:443 ssl_certificate=<path to SSL certificate>
```

### 関連情報

- 詳細は、[Beast フロントエンド](#) を参照してください。

## 2.7. DNS へのワイルドカードの追加

S3 スタイルのサブドメイン (例: **bucket-name.domain-name.com**) で Ceph を使用するには、**ceph-radosgw** デーモンがドメイン名を解決するために使用する DNS サーバーの DNS レコードにワイルドカードを追加します。

**dnsmasq** の場合は、ホスト名の先頭にドット (.) を付けた以下のアドレス設定を追加します。

```
address=/{hostname-or-fqdn}/{host-ip-address}
```

以下に例を示します。

```
address=/.gateway-node1/192.168.122.75
```

**bind** の場合は、ワイルドカードを DNS レコードに追加します。以下に例を示します。

```
$TTL 604800
@ IN SOA gateway-node1. root.gateway-node1. (
        2      ; Serial
        604800 ; Refresh
        86400  ; Retry
        2419200 ; Expire
        604800 ) ; Negative Cache TTL
;
@ IN NS gateway-node1.
@ IN A 192.168.122.113
* IN CNAME @
```

DNS サーバーを再起動して、サブドメインを使用してサーバーに ping し、**ceph-radosgw** デーモンがサブドメイン要求を処理できるようにします。

```
ping mybucket.{hostname}
```

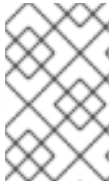
以下に例を示します。

```
ping mybucket.gateway-node1
```

DNS サーバーがローカルマシンにある場合は、ローカルマシンのネームサーバーエントリーを追加して `/etc/resolv.conf` を変更しないといけない場合があります。

最後に、`rgw_dns_name = {hostname}` 設定を使用して、Ceph 設定ファイルの適切な `[client.rgw.{instance}]` セクションに DNS サーバーのホスト名またはアドレスを指定します。以下に例を示します。

```
[client.rgw.rgw1]
...
rgw_dns_name = {hostname}
```



### 注記

ベストプラクティスとして、管理ノードや **ceph-ansible** などの一元的な場所で Ceph 設定ファイルを変更し、必要に応じて設定ファイルを再配布し、クラスター全体で一貫性を確保します。

最後に、Ceph Object Gateway を再起動して DNS 設定を有効にします。

## 2.8. ロギングおよびデバッグ出力の調整

設定手順を完了したら、ログの出力を確認して、ニーズを満たしていることを確認してください。設定に問題が発生した場合には、Ceph 設定ファイルの `[global]` セクションでメッセージおよびデバッグメッセージを増やし、ゲートウェイを再起動して設定の問題のトラブルシューティングを行うことができます。以下に例を示します。

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

実行時に設定を変更することも可能です。以下に例を示します。

```
# ceph tell osd.0 injectargs --debug_civetweb 10/20
```

Ceph ログファイルは、デフォルトで `/var/log/ceph` に保存されています。

ロギングとデバッグの一般的な詳細については、Red Hat Ceph Storage 3 の [設定ガイド](#) の [ロギング設定リファレンス](#) の章を参照してください。Ceph Object Gateway に固有のロギングの詳細については、このガイドの [ロギング設定リファレンス](#) の章の [Ceph Object Gateway](#) セクションを参照してください。

## 2.9. S3 API サーバー側の暗号化

Ceph Object Gateway は、S3 API のアップロードされたオブジェクトのサーバー側の暗号化をサポートしています。サーバー側の暗号化とは、S3 クライアントが暗号化されていない形式で HTTP 経由でデータを送信し、Ceph Object Gateway はそのデータを暗号化した形式で Ceph Storage Cluster に保存することを意味します。



### 注記

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





## 重要

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

暗号化キーの管理には、以下の2つのオプションがあります。

### お客様提供のキー

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

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

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

### キー管理サービス

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

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



## 重要

現在、テスト済みの唯一のキー管理実装は OpenStack Barbican を使用します。ただし、OpenStack Barbican の使用はまだ完全にはサポートされていません。本番環境で使用する唯一の方法は、サポートの例外を取得することです。詳細については、[テクニカルサポート](#) にお問い合わせください。

## 2.10. ゲートウェイのテスト

REST インターフェイスを使用するには、まず S3 インターフェイス用に初期 Ceph Object Gateway ユーザーを作成します。次に、Swift インターフェイスのサブユーザーを作成します。作成されたユーザーがゲートウェイにアクセスできるかどうかを確認する必要があります。

### 2.10.1. S3 ユーザーの作成

ゲートウェイをテストするには、S3 ユーザーを作成し、ユーザーアクセスを付与します。`man radosgw-admin` コマンドは、追加のコマンドオプションに関する情報を提供します。



## 注記

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

### 前提条件

- **root** または **sudo** アクセス
- Ceph Object Gateway がインストールされていること

## 手順

1. S3 ユーザーを作成します。

```
radosgw-admin user create --uid=name --display-name="First User"
```

**name** を、S3 ユーザーの名前に置き換えます。以下に例を示します。

```
[root@master-zone]# radosgw-admin user create --uid="testuser" --display-name="First User"
{
  "user_id": "testuser",
  "display_name": "First User",
  "email": "",
  "suspended": 0,
  "max_buckets": 1000,
  "auid": 0,
  "subusers": [],
  "keys": [
    {
      "user": "testuser",
      "access_key": "CEP28KDIQXBKU4M15PDC",
      "secret_key": "MARoio8HFc8JxhEilES3dKFVj8tV3NOOYymihTLO"
    }
  ],
  "swift_keys": [],
  "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. 出力を確認し、**access\_key** および **secret\_key** の値に JSON エスケープ文字 (\) が含まれていないことを確認します。これらの値はアクセスの検証に必要ですが、値に JSON エスケープ文字が含まれる場合、特定のクライアントは処理できません。この問題を修正するには、以下の

アクションのいずれかを実行します。

- JSON エスケープ文字を削除します。
- 文字列を引用符でカプセル化します。
- キーを再生成し、JSON エスケープ文字が含まれていないことを確認します。
- キーおよびシークレットを手動で指定します。

正引きスラッシュ / は有効な文字であるため、削除しないでください。

## 2.10.2. Swift ユーザーの作成

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



### 注記

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

### 前提条件

- **root** または **sudo** アクセス
- Ceph Object Gateway がインストールされていること

### 手順

1. Swift ユーザーを作成します。

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

**name** を Swift ユーザーの名前に置き換えます。次に例を示します。

```
[root@master-zone]# 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": "13TLtdEW7bCqgttQgPzxFzxiu0AgabtOc6vM8DLA"
  }
],
"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"
}

```

- シークレットキーを作成します。

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

**name** を Swift ユーザーの名前に置き換えます。次に例を示します。

```

[root@master-zone]# 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"
}

```

### 2.10.3. S3 アクセスのテスト

S3 アクセスを検証するには、Python テストスクリプトを作成し、実行する必要があります。S3 アクセステストスクリプトは **radosgw** に接続し、新規バケットを作成し、すべてのバケットを一覧表示します。**aws\_access\_key\_id** および **aws\_secret\_access\_key** の値は、**radosgw\_admin** コマンドで返される **access\_key** および **secret\_key** の値から取得されます。

次の手順を実行します。

1. 共通リポジトリを有効にします。

```
# subscription-manager repos --enable=rhel-7-server-rh-common-rpms
```

2. **python-boto** パッケージをインストールします。

```
sudo yum install python-boto
```

3. Python スクリプトを作成します。

```
vi s3test.py
```

4. ファイルに以下のコンテンツを追加します。

```
import boto
import boto.s3.connection

access_key = $access
secret_key = $secret

boto.config.add_section('s3')

conn = boto.connect_s3(
    aws_access_key_id = access_key,
    aws_secret_access_key = secret_key,
    host = 's3.<zone>.hostname',
    port = <port>,
    is_secure=False,
    calling_format = boto.s3.connection.OrdinaryCallingFormat(),
)

bucket = conn.create_bucket('my-new-bucket')
for bucket in conn.get_all_buckets():
    print "{name}\t{created}".format(
        name = bucket.name,
        created = bucket.creation_date,
    )
```

- a. **<zone>** は、ゲートウェイサービスを設定したホストのゾーン名に置き換えます。つまり、**gateway host** です。**host** の設定が DNS で解決されていることを確認します。**<port>** は、ゲートウェイのポート番号に置き換えます。
- b. **\$access** と **\$secret** を、[S3 ユーザーの作成](#) セクションの **access\_key** と **secret\_key** の値に置き換えます。

5. スクリプトを実行します。

```
python s3test.py
```

出力は以下のようになります。

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

#### 2.10.4. Swift アクセスのテスト

Swift アクセスは、**swift** コマンドラインクライアントを使用して検証できます。**man swift** コマンドは、利用可能なコマンドラインオプションの詳細を提供します。

**swift** クライアントをインストールするには、以下のコマンドを実行します。

```
sudo yum install python-setuptools
sudo easy_install pip
sudo pip install --upgrade setuptools
sudo pip install --upgrade python-swiftclient
```

swift アクセスをテストするには、以下のコマンドを実行します。

```
swift -A http://{IP ADDRESS}:{port}/auth/1.0 -U testuser:swift -K '{swift_secret_key}' list
```

{IP ADDRESS} を、ゲートウェイサーバーのパブリック IP アドレスに置き換え、{swift\_secret\_key} を、swift ユーザーに対して実行した **radosgw-admin key create** コマンドの出力にある値に置き換えます。{port} を、Civetweb で使用しているポート番号に置き換えてください (例: デフォルトは **8080** です)。ポートを置き換えない場合、デフォルトはポート **80** になります。

以下に例を示します。

```
swift -A http://10.19.143.116:8080/auth/1.0 -U testuser:swift -K '244+fz2gSqoHwR3lYtSblyomyPHf3i7rgSJrF/IA' list
```

出力は以下のようになります。

```
my-new-bucket
```

## 2.11. HAPROXY/KEEPALIVED の設定

Ceph Object Gateway では、Object Gateway の多数のインスタンスを1つのゾーンに割り当てることができます。これにより、負荷が増大するとスケールアウトすることができます。つまり、同じゾーングループおよびゾーンとなりますが、HAProxy/**keepalived** を使用するためにフェデレーションされたアーキテクチャーは必要ありません。各オブジェクトゲートウェイは独自の IP アドレスを持つため、HAProxy および **keepalived** を使用して Ceph Object Gateway サーバー全体で負荷を分散できます。

HAProxy および **keepalived** のもう1つのユースケースは、HAProxy サーバーで HTTPS を終了することです。Red Hat Ceph Storage (RHCS) 1.3.x は Civetweb を使用しますが、RHCS 1.3.x の実装は HTTPS をサポートしません。HAProxy サーバーを使用して HAProxy サーバーで HTTPS を終了でき、HAProxy サーバーと Civetweb ゲートウェイインスタンスの間で HTTP を使用できます。

### 2.11.1. HAProxy/keepalived の前提条件

Ceph Object Gateway で HA プロキシを設定するには、以下が必要です。

- 稼働中の Ceph クラスタ
- ポート **80** で実行されるように設定している同じゾーン内の少なくとも2つの Ceph Object Gateway サーバー。簡単なインストール手順に従うと、ゲートウェイインスタンスはデフォルトで同じゾーングループおよびゾーンに置かれます。フェデレーションアーキテクチャーを使用している場合は、インスタンスが同じゾーングループとゾーンにあることを確認してください。
- HAProxy および **keepalived** の場合は少なくとも2台のサーバー。



#### 注記

本セクションでは、少なくとも2つの Ceph Object Gateway サーバーが実行されていることを前提とし、ポート **80** でテストスクリプトを実行する際に、このいずれかのサーバーからの有効な応答を取得していることを前提としています。

HAProxy および **keepalived** の詳細な説明は、[ロードバランサーの管理](#) を参照してください。

### 2.11.2. HAProxy ノードの準備

以下の設定は、**haproxy** と **haproxy2** という名前の 2 つの HAProxy ノードと、**rgw1** と **rgw2** という名前の 2 台の Ceph Object Gateway サーバーを前提としています。希望の命名規則を使用できます。少なくとも 2 つの HAProxy ノードで以下の手順を実行します。

1. RHEL 7.x をインストールします。

2. ノードを登録します。

```
[root@haproxy]# subscription-manager register
```

3. RHEL サーバーリポジトリを有効にします。

```
[root@haproxy]# subscription-manager repos --enable=rhel-7-server-rpms
```

4. サーバーを更新します。

```
[root@haproxy]# yum update -y
```

5. 必要に応じて管理ツール (**wget**、**vim** など) をインストールします。

6. ポート **80** を開きます。

```
[root@haproxy]# firewall-cmd --zone=public --add-port 80/tcp --permanent
[root@haproxy]# firewall-cmd --reload
```

7. HTTPS の場合は、ポート **443** を開きます。

```
[root@haproxy]# firewall-cmd --zone=public --add-port 443/tcp --permanent
[root@haproxy]# firewall-cmd --reload
```

### 2.11.3. keepalived のインストールと設定

少なくとも 2 つの HAProxy ノードで以下の手順を実行します。

#### 前提条件

- 少なくとも 2 つの HAProxy ノード
- 少なくとも 2 つの Object Gateway ノード

#### 手順

1. **keepalived** をインストールします。

```
[root@haproxy]# yum install -y keepalived
```

2. 両方の HAProxy ノードで **keepalived** を設定します。

```
[root@haproxy]# vim /etc/keepalived/keepalived.conf
```

設定ファイルには、**haproxy** プロセスを確認するスクリプトがあります。

```
vrp_script chk_haproxy {
```



```

script "killall -0 haproxy" # check the haproxy process
interval 2 # every 2 seconds
weight 2 # add 2 points if OK
}

```

次に、マスターロードバランサーとバックアップロードバランサーのインスタンスは、ネットワークインターフェイスとして **eno1** を使用します。また、仮想 IP アドレス (**192.168.1.20**) も割り当てます。

### マスターロードバランサーノード

```

vrrp_instance RGW {
    state MASTER # might not be necessary. This is on the Master LB node.
    @main interface eno1
    priority 100
    advert_int 1
    interface eno1
    virtual_router_id 50
    @main unicast_src_ip 10.8.128.43 80
    unicast_peer {
        10.8.128.53
    }
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.1.20
    }
    track_script {
        chk_haproxy
    }
}
virtual_server 192.168.1.20 80 eno1 { #populate correct interface
    delay_loop 6
    lb_algo wlc
    lb_kind dr
    persistence_timeout 600
    protocol TCP
    real_server 10.8.128.43 80 { # ip address of rgw2 on physical interface, haproxy listens
here, rgw listens to localhost:8080 or similar
        weight 100
        TCP_CHECK { # perhaps change these to a HTTP/SSL GET?
            connect_timeout 3
        }
    }
    real_server 10.8.128.53 80 { # ip address of rgw3 on physical interface, haproxy listens
here, rgw listens to localhost:8080 or similar
        weight 100
        TCP_CHECK { # perhaps change these to a HTTP/SSL GET?
            connect_timeout 3
        }
    }
}
}

```

## バックアップロードバランサーノード

```

vrrp_instance RGW {
    state BACKUP # might not be necessary?
    priority 99
    advert_int 1
    interface eno1
    virtual_router_id 50
    unicast_src_ip 10.8.128.53 80
    unicast_peer {
        10.8.128.43
    }
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.1.20
    }
    track_script {
        chk_haproxy
    }
}
virtual_server 192.168.1.20 80 eno1 { #populate correct interface
    delay_loop 6
    lb_algo wlc
    lb_kind dr
    persistence_timeout 600
    protocol TCP
    real_server 10.8.128.43 80 { # ip address of rgw2 on physical interface, haproxy listens
here, rgw listens to localhost:8080 or similar
        weight 100
        TCP_CHECK { # perhaps change these to a HTTP/SSL GET?
            connect_timeout 3
        }
    }
    real_server 10.8.128.53 80 { # ip address of rgw3 on physical interface, haproxy listens
here, rgw listens to localhost:8080 or similar
        weight 100
        TCP_CHECK { # perhaps change these to a HTTP/SSL GET?
            connect_timeout 3
        }
    }
}
}

```

3. **keepalived** サービスを有効にして開始します。

```

[root@haproxy]# systemctl enable keepalived
[root@haproxy]# systemctl start keepalived

```

### 関連情報

- **keepalived** の設定に関する詳しい説明は、[Keepalived を使用した初期ロードバランサーの設定](#)を参照してください。

## 2.11.4. HAProxy のインストールおよび設定

少なくとも 2 つの HAProxy ノードで以下の手順を実行します。

1. **haproxy** をインストールします。

```
[root@haproxy]# yum install haproxy
```

2. SELinux および HTTP に対して **haproxy** を設定します。

```
[root@haproxy]# vim /etc/firewalld/services/haproxy-http.xml
```

以下の行を追加します。

```
<?xml version="1.0" encoding="utf-8"?>
<service>
<short>HAProxy-HTTP</short>
<description>HAProxy load-balancer</description>
<port protocol="tcp" port="80"/>
</service>
```

**root** として、正しい SELinux コンテキストとファイルパーミッションを **haproxy-http.xml** ファイルに割り当てます。

```
[root@haproxy]# cd /etc/firewalld/services
[root@haproxy]# restorecon haproxy-http.xml
[root@haproxy]# chmod 640 haproxy-http.xml
```

3. HTTPS を使用する場合は、SELinux および HTTPS に対して **haproxy** を設定します。

```
[root@haproxy]# vim /etc/firewalld/services/haproxy-https.xml
```

以下の行を追加します。

```
<?xml version="1.0" encoding="utf-8"?>
<service>
<short>HAProxy-HTTPS</short>
<description>HAProxy load-balancer</description>
<port protocol="tcp" port="443"/>
</service>
```

**root** で、正しい SELinux コンテキストとファイルパーミッションを **haproxy-https.xml** ファイルに割り当てます。

```
# cd /etc/firewalld/services
# restorecon haproxy-https.xml
# chmod 640 haproxy-https.xml
```

4. HTTPS を使用する場合は、SSL のキーを生成します。証明書がない場合は、自己署名証明書を使用できます。キーを生成するには、Red Hat Enterprise Linux 7 のシステム管理者のガイドの [新しい鍵および証明書の生成](#) セクションを参照してください。最後に、証明書と鍵を PEM ファイルに格納します。

```
[root@haproxy]# cat example.com.crt example.com.key > example.com.pem
[root@haproxy]# cp example.com.pem /etc/ssl/private/
```

## 5. haproxy を設定します。

```
[root@haproxy]# vim /etc/haproxy/haproxy.cfg
```

**global** および **defaults** は変更しない可能性があります。 **defaults** セクションの後に、 **frontend** セクションおよび **backend** セクションを設定する必要があります。以下に例を示します。

```
frontend http_web *:80
    mode http
    default_backend rgw

frontend rgw-https
    bind *:443 ssl crt /etc/ssl/private/example.com.pem
    default_backend rgw

backend rgw
    balance roundrobin
    mode http
    server rgw1 10.0.0.71:80 check
    server rgw2 10.0.0.80:80 check
```

HAProxy 設定の詳細な説明は、 [HAProxy 設定](#) を参照してください。

## 6. haproxy の有効化/起動

```
[root@haproxy]# systemctl enable haproxy
[root@haproxy]# systemctl start haproxy
```

### 2.11.5. HAProxy 設定のテスト

HAProxy ノードで、 **keepalived** 設定からの仮想 IP アドレスが表示されることを確認します。

```
[root@haproxy]# ip addr show
```

calamari ノードで、ロードバランサー設定経由でゲートウェイノードにアクセスできるかどうかを確認します。以下に例を示します。

```
[root@haproxy]# wget haproxy
```

上記のコマンドの結果は以下のコマンドと同じになるはずですが。

```
[root@haproxy]# wget rgw1
```

以下の内容を含む **index.html** ファイルを返す場合は、次のコマンドを実行します。

```
<?xml version="1.0" encoding="UTF-8"?>
<ListAllMyBucketsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Owner>
```

```

<ID>anonymous</ID>
<DisplayName></DisplayName>
</Owner>
<Buckets>
</Buckets>
</ListAllMyBucketsResult>

```

その後、設定が正常に機能します。

## 2.12. 静的 WEB ホスティング用のゲートウェイの設定

従来の Web ホストでは、各 Web サイトに Web サーバーをを設定する必要があります。これは、コンテンツが動的に変更されない場合、リソースは効率的に使用されない可能性があります。Ceph Object Gateway は、S3 バケットで静的 Web サイトをホストできます。つまり、PHP、サブレット、データベース、nodejs などのサーバー側のサービスを使用しないサイトです。このアプローチは、サイトごとに Web サーバーを備えた仮想マシンをセットアップするよりも、はるかに経済的です。

### 2.12.1. 静的 Web ホストの前提条件

静的 Web ホストには、1 つ以上の実行中の Ceph Storage Cluster が必要です。また、静的 Web サイト用に 2 つ以上の Ceph Object Gateway インスタンスが必要です。Red Hat は、各ゾーンに HAProxy/keepalived によって負荷分散された複数のゲートウェイインスタンスがあることを前提としています。



#### 注記

Red Hat は、Ceph Object Gateway インスタンスを使用した標準の S3/Swift API と静的 Web ホストの両方を同時にデプロイすることはサポート **されません**。

### 2.12.2. 静的 Web ホストの要件

静的 Web ホスティング機能は独自の API を使用するため、S3 バケットで静的 Web サイトを使用するようにゲートウェイを設定するには、以下が必要です。

1. S3 静的 Web ホストでは、Ceph Object Gateway インスタンスが使用され、標準の S3/Swift API のユースケースに使用されるインスタンスと区別されます。
2. S3 静的 Web サイトをホストするゲートウェイインスタンスは、標準の S3/Swift API ゲートウェイインスタンスとは別の重複しないドメイン名を持っている必要があります。
3. S3 静的 Web サイトをホストするゲートウェイインスタンスは、標準の S3/Swift API ゲートウェイインスタンスとは別のパブリック向け IP アドレスを使用する必要があります。
4. S3 静的 Web サイトをホストするゲートウェイインスタンスは負荷分散し、必要に応じて HAProxy/keepalived を使用して SSL を終了します。

### 2.12.3. 静的 Web ホストゲートウェイの設定

静的 Web ホスト用のゲートウェイを有効にするには、Ceph 設定ファイルを編集して以下の設定を追加します。

```

[client.rgw.<STATIC-SITE-HOSTNAME>]
...
rgw_enable_static_website = true

```

```
rgw_enable_apis = s3, s3website
rgw_dns_name = objects-zonegroup.domain.com
rgw_dns_s3website_name = objects-website-zonegroup.domain.com
rgw_resolve_cname = true
...
```

**rgw\_enable\_static\_website** 設定は **true** にする必要があります。**rgw\_enable\_apis** 設定は **s3website** API を有効にする必要があります。**rgw\_dns\_name** 設定および **rgw\_dns\_s3website\_name** 設定は、完全修飾ドメインを提供する必要があります。サイトが標準的な名前の拡張子を使用する場合は、**rgw\_resolve\_cname** を **true** に設定します。



### 重要

**rgw\_dns\_name** および **rgw\_dns\_s3website\_name** の完全修飾ドメイン名は重複しないでください。

## 2.12.4. 静的 Web ホスティング DNS 設定

以下は、想定される DNS 設定の例です。最初の 2 行は、標準の S3 インターフェイスを使用してゲートウェイインスタンスのドメインを指定し、それぞれ IPv4 アドレスおよび IPv6 アドレスを指しています。3 行目は、正規名の拡張子を使用して S3 バケットのワイルドカード CNAME 設定を提供します。4 番目と 5 番目の行は、S3 の Web サイトインターフェイスを使用してゲートウェイインスタンスのドメインを指定し、それぞれ IPv4 アドレスおよび IPv6 アドレスを参照します。

```
objects-zonegroup.domain.com. IN A 192.0.2.10
objects-zonegroup.domain.com. IN AAAA 2001:DB8::192:0:2:10
*.objects-zonegroup.domain.com. IN CNAME objects-zonegroup.domain.com.
objects-website-zonegroup.domain.com. IN A 192.0.2.20
objects-website-zonegroup.domain.com. IN AAAA 2001:DB8::192:0:2:20
```



### 注記

最初の 2 行にある IP アドレスは、4 番目と 5 行目の IP アドレスとは異なります。

マルチサイト設定で Ceph Object Gateway を使用している場合は、ルーティングソリューションを使用してトラフィックをクライアントに最も近いゲートウェイにルーティングすることを検討してください。

Amazon Web Service (AWS) では、ホスト名に一致するように静的 Web ホストバケットが必要です。Ceph は DNS を設定するいくつかの方法を提供し、**プロキシに適合する証明書がある場合に HTTPS は機能します。**

### サブドメインのバケットのホスト名

AWS 形式の S3 サブドメインを使用するには、DNS エントリーでワイルドカードを使用し、要求を任意のバケットにリダイレクトできます。DNS エントリーは以下のようになります。

```
*.objects-website-zonegroup.domain.com. IN CNAME objects-website-zonegroup.domain.com.
```

以下の方法でバケット名にアクセスします。

```
http://bucket1.objects-website-zonegroup.domain.com
```

バケット名は **bucket1** です。

## 一致しないバケットのホスト名

Ceph は、リクエストにバケット名を含めずにドメイン名をバケットへのマッピングをサポートします。これは Ceph Object Gateway に固有のもので、ドメイン名を使用してバケットにアクセスするには、ドメイン名をバケット名にマップします。DNS エントリーは以下のようになります。

```
www.example.com. IN CNAME bucket2.objects-website-zonegroup.domain.com.
```

バケット名は **bucket2** です。

以下の方法でバケットにアクセスします。

```
http://www.example.com
```

## CNAME を使用した長いバケットのホスト名

AWS は通常、ドメイン名に一致するバケット名を必要とします。CNAME を使用して静的 Web ホストに DNS を設定するには、DNS エントリーは以下のようになります。

```
www.example.com. IN CNAME www.example.com.objects-website-zonegroup.domain.com.
```

以下の方法でバケットにアクセスします。

```
http://www.example.com
```

## CNAME のない長いバケットのホスト名

DNS 名には、**SOA**、**NS**、**MX**、**TXT** などの他の非 CNAME レコードが含まれている場合、DNS レコードはドメイン名を IP アドレスに直接マップする必要があります。以下に例を示します。

```
www.example.com. IN A 192.0.2.20  
www.example.com. IN AAAA 2001:DB8::192:0:2:20
```

以下の方法でバケットにアクセスします。

```
http://www.example.com
```

### 2.12.5. 静的 Web ホストサイトの作成

静的 Web サイトを作成するには、以下の手順を実行します。

1. S3 バケットを作成します。バケット名は、Web サイトのドメイン名と同じである場合があります。たとえば、**mysite.com** のバケット名は **mysite.com** になります。これは AWS に必要ですが、Ceph には必要ありません。詳細は [DNS 設定](#) を参照してください。
2. 静的 Web コンテンツをバケットにアップロードします。コンテンツには、HTML、CSS、クライアント側の JavaScript、イメージ、音声/ビデオコンテンツなどのダウンロード可能なファイルが含まれる場合があります。Web サイトには **index.html** ファイルが必要で、**error.html** ファイルも利用できます。
3. Web サイトの内容を確認します。この時点で、バケットの作成者のみがコンテンツにアクセスできます。
4. ファイルにパーミッションを設定し、それらが一般に公開されるようにします。

## 2.13. NFS-GANESHA への名前空間のエクスポート

Red Hat Ceph Storage 3 では、Ceph Object Gateway は、実稼働システムに NFS バージョン 3 および NFS バージョン 4.1 を使用して、S3 オブジェクト名前空間をエクスポートする機能を提供します。



### 注記

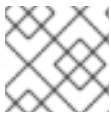
NFS Ganesha 機能は一般的な使用ではなく、S3 クラウドへの移行のみを目的としています。

この実装は、UNIX 形式のパス名を S3 バケットおよびオブジェクトにマッピングする Amazon Web Services (AWS) 階層名前空間の規則に準拠しています。アタッチされた名前空間のトップレベルは、存在する場合は NFSv4 疑似 root に従属し、Ceph Object Gateway S3 バケットで設定されます。バケットは、NFS ディレクトリーとして表されます。バケット内のオブジェクトは、S3 規則に従って、NFS ファイルおよびディレクトリー階層として表示されます。ファイルおよびディレクトリーを作成する操作はサポートされています。



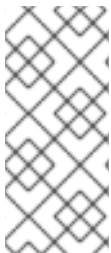
### 注記

ハードリンクまたはソフトリンクの作成または削除は、サポートされていません。バケットまたはディレクトリーでの名前変更操作の実行は NFS 経由ではサポートされていませんが、ファイルでの名前変更はディレクトリー内およびディレクトリー間、およびファイルシステムと NFS マウント間でサポートされています。ファイルの名前変更操作は、NFS 上で行う場合、ターゲットディレクトリーを変更し、通常は完全な **readdir** を強制的に更新するため、コストが高くなります。



### 注記

NFS マウントを使用したファイルの編集はサポートされていません。



### 注記

Ceph Object Gateway では、アプリケーションがオフセット 0 からファイルの終わりまで順番に書き込む必要があります。順不同で書き込もうとすると、アップロード操作が失敗します。この問題を回避するには、ファイルを NFS 領域にコピーする際に、**cp**、**cat**、**rsync** などのユーティリティーを使用します。常に **sync** オプションでマウントします。

NFS を使用する Ceph Object Gateway は、Gateway サーバーのインプロセスライブラリーパッケージと、NFS-Ganesha NFS サーバーの File System Abstraction Layer (FSAL) 名前空間ドライバーに基づいています。ランタイム時に、NFS を使用する Ceph Object Gateway デーモンのインスタンスは、Civetweb HTTP サービスがない場合でも、完全な Ceph Object Gateway デーモンと NFS-Ganesha インスタンスを単一のプロセスで結合します。この機能を使用するには、NFS-Ganesha バージョン 2.3.2 以降をデプロイします。

NFS-Ganesha (**nfs-ganesha-rgw**) インスタンスを含むホストで [作業を開始する前に](#) および [NFS-Ganesha インスタンス](#) のステップを実行します。

### 複数の NFS ゲートウェイの実行

各 NFS-Ganesha インスタンスは完全なゲートウェイエンドポイントとして機能しますが、HTTP サービスをエクスポートするように NFS-Ganesha インスタンスを設定できないという制限が現時点であります。通常のゲートウェイインスタンスと同様に、任意の数の NFS-Ganesha インスタンスを起動し、



クラスターから同じまたは異なるリソースをエクスポートできます。これにより、NFS-Ganesha インスタンスのクラスターリングが可能になります。ただし、これは高可用性を意味するものではありません。

通常のゲートウェイインスタンスと NFS-Ganesha インスタンスが同じデータリソースと重複している場合、標準の S3 API から、およびエクスポートされた NFS-Ganesha インスタンスを介してアクセスできます。NFS-Ganesha インスタンスを同じホスト上の Ceph Object Gateway インスタンスと同じ場所に配置できます。

### 作業を開始する前に

1. NFS-Ganesha を実行する前に、NFS-Ganesha を実行するホストで実行中のカーネル NFS サービスインスタンスをすべて無効にします。NFS-Ganesha は、別の NFS インスタンスが実行している場合は起動しません。
2. **rpcbind** サービスが実行していることを確認します。

```
# systemctl start rpcbind
```



#### 注記

**rpcbind** を提供する **rpcbind** パッケージは通常、デフォルトでインストールされます。そうでない場合は、最初にパッケージをインストールします。

NFS の **rpcbind** の使用方法の詳細は、Red Hat Enterprise Linux 7 のストレージ管理ガイドの [必要なサービス](#) セクションを参照してください。

3. **nfs-service** サービスが実行中である場合は、これを停止して無効にします。

```
# systemctl stop nfs-server.service  
# systemctl disable nfs-server.service
```

### NFS-Ganesha インスタンスの設定

1. **nfs-ganesha-rgw** パッケージをインストールします。

```
$ sudo apt-get install nfs-ganesha-rgw
```

2. Ceph Monitor ノードから NFS-Ganesha ホストの **/etc/ceph/** ディレクトリーに Ceph 設定ファイルをコピーし、必要に応じて編集します。

```
# scp <mon-host>:/etc/ceph/ceph.conf <nfs-ganesha-rgw-host>:/etc/ceph
```



## 注記

Ceph 設定ファイルには、有効な `[client.rgw.{instance-name}]` セクションと、`rgw_data`、`keyring`、`rgw_frontends` など、必要なさまざまな Gateway 設定変数が含まれている必要があります。有効な S3 バケットの命名要件に準拠していない Swift コンテナをエクスポートする場合は、Ceph 設定ファイルの `[client.rgw]` セクションで `rgw_relaxed_s3_bucket_names` を `true` に設定します。たとえば、Swift コンテナ名にアンダースコアが含まれる場合、これは有効な S3 バケット名ではなく、`rgw_relaxed_s3_bucket_names` が `true` に設定されていない限り同期されません。NFS 外にオブジェクトおよびバケットを追加すると、これらのオブジェクトは、デフォルトで `rgw_nfs_namespace_expire_secs` によって設定された時間に NFS 名前空間に表示されます。デフォルトでは約 5 分です。Ceph 設定ファイルの `rgw_nfs_namespace_expire_secs` のデフォルト値を上書きして、更新レートを変更します。

3. NFS-Ganesha 設定ファイルを開きます。

```
# vim /etc/ganesha/ganesha.conf
```

4. **FSAL** (File System Abstraction Layer) ブロックを使用して **EXPORT** セクションを設定します。ID、S3 ユーザー ID、S3 アクセスキー、およびシークレットを指定します。NFSv4 の場合は、以下ようになります。

```
EXPORT
{
    Export_ID={numeric-id};
    Path = "/";
    Pseudo = "/";
    Access_Type = RW;
    SecType = "sys";
    NFS_Protocols = 4;
    Transport_Protocols = TCP;
    Squash = No_Root_Squash;

    FSAL {
        Name = RGW;
        User_Id = {s3-user-id};
        Access_Key_Id = "{s3-access-key}";
        Secret_Access_Key = "{s3-secret}";
    }
}
```

**Path** オプションは、エクスポートを見つける場所を Ganesha に指示します。VFS FSAL の場合は、これはサーバーの名前空間内の場所になります。他の FSAL の場合は、その FSAL の名前空間が管理するファイルシステム内の場所になる可能性があります。たとえば、CephFSAL を使用して CephFS ボリューム全体をエクスポートする場合、**Path** は / になります。

**Pseudo** オプションは、Ganesha に対して NFS v4 の擬似ファイルシステムの名前空間内にエクスポートを配置するよう指示します。NFS v4 は、エクスポートの実際の場所に対応しない疑似名前空間を構築する可能性があるサーバーを指定し、その疑似ファイルシステムの一部は NFS サーバーのレーム内にのみ存在し、物理ディレクトリーに対応しない可能性があります。さらに、NFS v4 サーバーはすべてのエクスポートを 1 つの名前空間に配置します。単一のエクスポートを疑似ファイルシステムの root としてエクスポートすることは可能ですが、複数のエクスポートを疑似ファイルシステムに配置する方がはるかに一般的です。従来の VFS では、多

くの場合、**Pseudo** の場所は **Path** の場所と同じです。/ を **Path** として使用して CephFS エクスポート例に戻る場合、複数のエクスポートが必要な場合は、エクスポートに **Pseudo** オプションが他にない可能性があります。たとえば、`/ceph` です。

NFSv3 に対応する **EXPORT** ブロックには、**NFS\_Protocols** 設定でバージョン 3 を含める必要があります。さらに、NFSv3 は、UDP トランスポートをサポートする最後のメジャーバージョンになります。標準の初期バージョンには UDP が含まれていましたが、RFC 7530 ではその使用が禁止されています。UDP を有効にするには、**Transport\_Protocols** 設定に追加します。以下に例を示します。

```
EXPORT {
...
    NFS_Protocols = 3,4;
    Transport_Protocols = UDP,TCP;
...
}
```

**SecType = sys;** を設定することで、クライアントは Kerberos 認証なしで接続できます。

**Squash = No\_Root\_Squash;** を設定すると、ユーザーは NFS マウント内のディレクトリー所有権を変更できます。

従来の OS ネイティブ NFS 4.1 クライアントを使用する NFS クライアントは通常、移行先サーバーの **pseudofs** root で定義されるエクスポートされたファイルシステムのフェデレーションされた名前空間を表示します。これらの任意の数を Ceph Object Gateway エクスポートに指定できます。

各エクスポートには、**name**、**User\_Id**、**Access\_Key**、および **Secret\_Access\_Key** という独自のタプルがあり、指定されたユーザーが確認できるオブジェクトの名前空間のプロキシーを作成します。

**ganesha.conf** のエクスポートには、**NFSV4** ブロックを含めることもできます。Red Hat Ceph Storage では、**idmapper** プログラムを設定する代わりに、**Allow\_Numeric\_Owners** パラメーターおよび **Only\_Numeric\_Owners** パラメーターサポートされます。

```
NFSV4 {
    Allow_Numeric_Owners = true;
    Only_Numeric_Owners = true;
}
```

##### 5. **NFS\_CORE\_PARAM** ブロックを設定します。

```
NFS_CORE_PARAM{
    mount_path_pseudo = true;
}
```

**mount\_path\_pseudo** 設定設定は、**true** に設定すると、NFS v3 および NFS v4.x のマウントが同じサーバー側パスを使用してエクスポートに到達させます。

```
mount -o vers=3 <IP ADDRESS>:/export /mnt
mount -o vers=4 <IP ADDRESS>:/export /mnt
```

```
Path      Pseudo    Tag      Mechanism  Mount
/export/test1 /export/test1 test1    v3 Pseudo  mount -o vers=3 server:/export/test1
```

```

/export/test1 /export/test1 test1 v3 Tag mount -o vers=3 server:test1
/export/test1 /export/test1 test1 v4 Pseudo mount -o vers=4 server:/export/test1
/ /export/ceph1 ceph1 v3 Pseudo mount -o vers=3 server:/export/ceph1
/ /export/ceph1 ceph1 v3 Tag mount -o vers=3 server:ceph1
/ /export/ceph1 ceph1 v4 Pseudo mount -o vers=4 server:/export/ceph1
/ /export/ceph2 ceph2 v3 Pseudo mount -o vers=3 server:/export/ceph2
/ /export/ceph2 ceph2 v3 Tag mount -o vers=3 server:ceph2
/ /export/ceph2 ceph2 v4 Pseudo mount -o vers=4

```

**mount\_path\_pseudo** の設定設定を **false** に設定すると、NFS v3 は **Path** オプションを使用し、NFS v4.x マウントは **Pseudo** オプションを使用します。

```

Path      Pseudo      Tag      Mechanism  Mount
/export/test1 /export/test1 test1 v3 Path mount -o vers=3 server:/export/test1
/export/test1 /export/test1 test1 v3 Tag mount -o vers=3 server:test1
/export/test1 /export/test1 test1 v4 Pseudo mount -o vers=4 server:/export/test1
/ /export/ceph1 ceph1 v3 Path mount -o vers=3 server:/
/ /export/ceph1 ceph1 v3 Tag mount -o vers=3 server:ceph1
/ /export/ceph1 ceph1 v4 Pseudo mount -o vers=4 server:/export/ceph1
/ /export/ceph2 ceph2 v3 Path not accessible
/ /export/ceph2 ceph2 v3 Tag mount -o vers=3 server:ceph2
/ /export/ceph2 ceph2 v4 Pseudo mount -o vers=4 server:/export/ceph2

```

6. **RGW** セクションを設定します。インスタンスの名前を指定し、Ceph 設定ファイルへのパスを指定し、任意の初期化引数を指定します。

```

RGW {
    name = "client.rgw.{instance-name}";
    ceph_conf = "/etc/ceph/ceph.conf";
    init_args = "--{arg}={arg-value}";
}

```

7. **/etc/ganesha/ganesha.conf** 設定ファイルを保存します。
8. **nfs-ganesha** サービスを有効にして開始します。

```

# systemctl enable nfs-ganesha
# systemctl start nfs-ganesha

```

9. 擬似ディレクトリーが非常に大きい場合には、**ceph.conf** ファイルの設定可能なパラメーター **rgw\_nfs\_s3\_fast\_attrs** を **true** に設定して、名前をイミュータブルかつ加速します。

```

rgw_nfs_s3_fast_attrs= true

```

10. 各ゲートウェイノードから Ceph Object Gateway サービスを再起動します。

```

# systemctl restart ceph-radosgw.target

```

## NFSv4 クライアントの設定

名前空間にアクセスするには、設定された NFS-Ganesha エクスポートをローカルの POSIX 名前空間で必要な場所にマウントします。前述のように、この実装には固有の制限がいくつかあります。

- NFS 4.1 以降のプロトコルフレーバーのみがサポートされます。

- 書き込み順序を設定するには、**sync** マウントオプションを使用します。

NFS-Ganesha エクスポートをマウントするには、クライアントホストの **/etc/fstab** ファイルに以下のエントリーを追加します。

```
<ganesha-host-name>:/ <mount-point> nfs noauto,soft,nfsvers=4.1,sync,proto=tcp 0 0
```

NFS-Ganesha ホスト名とクライアントのマウントポイントへのパスを指定します。



### 注記

NFS-Ganesha エクスポートを正常にマウントするには、クライアントに **/sbin/mount.nfs** ファイルが存在する必要があります。**nfs-tools** パッケージはこのファイルを提供します。多くの場合、パッケージはデフォルトでインストールされています。ただし、**nfs-tools** パッケージがクライアントにインストールされていることを確認し、インストールされていない場合はインストールします。

NFS の詳細は、Red Hat Enterprise Linux 7 のストレージ管理ガイドの [ネットワークファイルシステム \(NFS\)](#) の章を参照してください。

## NFSv3 クライアントの設定

マウントオプションとして **nfsvers=3** および **noacl** を指定して、Linux クライアントが NFSv3 でマウントされるように設定できます。UDP をトランスポートとして使用するには、**proto=udp** をマウントオプションに追加します。ただし、TCP が推奨されるプロトコルです。

```
<ganesha-host-name>:/ <mount-point> nfs noauto,noacl,soft,nfsvers=3,sync,proto=tcp 0 0
```



### 注記

NFS Ganesha **EXPORT** ブロックの **Protocols** 設定をバージョン 3 に設定し、マウントがバージョン 3 を UDP で使用する場合は **Transports** 設定を UDP に設定します。

NFSv3 はクライアントの OPEN および CLOSE 操作をファイルサーバーに通信しないため、RGW NFS はこれらの操作を使用して、ファイルのアップロードトランザクションの開始と終了をマークすることはできません。代わりに、RGW NFS は、オフセット 0 でファイルに最初の書き込みが送信されたときに新しいアップロードを開始しようとし、ファイルへの新しい書き込みが一定期間 (デフォルトでは 10 秒) 見られなかったときにアップロードを終了します。この値を変更するには、Ceph 設定ファイルの RGW セクションに **rgw\_nfs\_write\_completion\_interval\_s** の値を設定します。

## 第3章 管理

管理者は、**radosgw-admin** コマンドラインインターフェイスを使用して Ceph Object Gateway を管理できます。

- [管理データストレージ](#)
- [ストレージポリシー](#)
- [インデックスのないバケット](#)
- [バケットシャード化](#)
- [圧縮](#)
- [ユーザー管理](#)
- [クォータ管理](#)
- [用途](#)

### 3.1. 管理データストレージ

Ceph Object Gateway は、インスタンスのゾーン設定で定義された一連のプールに管理データを保存します。たとえば、後続のセクションで説明したバケット、ユーザー、ユーザークォータおよび使用状況の統計は、Ceph Storage Cluster のプールに保存されます。デフォルトでは、Ceph Object Gateway は以下のプールを作成し、それらをデフォルトゾーンにマッピングします。

- **.rgw**
- **.rgw.control**
- **.rgw.gc**
- **.log**
- **.intent-log**
- **.usage**
- **.users**
- **.users.email**
- **.users.swift**
- **.users.uid**

CRUSH ルールセットと配置グループの数を設定できるように、これらのプールを手動で作成することを検討してください。一般的な設定では、Ceph Object Gateway の管理データを格納するプールは、管理データに 10 個のプールがあるため、多くの場合、同じ CRUSH ルールセットを使用し、使用する配置グループの数を少なくします。詳細は、Red Hat Ceph Storage 3 の場合は、[プール](#) および [ストレージ戦略](#) ガイドを参照してください。

また、配置グループの計算の詳細については、[Ceph Placement Groups\(PGs\)per Pool Calculator](#) も参照してください。**mon\_pg\_warn\_max\_per\_osd** 設定は、プールに過剰な配置グループを割り当てると警告します (つまりデフォルトでは 300)。この値は、ニーズやハードウェアの能力に合わせて調整する

ことができ、**n** は OSD あたりの PG の最大数です。

```
mon_pg_warn_max_per_osd = n
```

## 3.2. ストレージポリシーの作成

Ceph Object Gateway は配置ターゲットを特定し、配置ターゲットに関連付けられたプールにバケットおよびオブジェクトを保存することで、クライアントバケットとオブジェクトデータを保存します。配置ターゲットを設定しておらず、インスタンスのゾーン設定内のプールにマッピングすると、Ceph Object Gateway はデフォルトのターゲットとプールを使用します (例: **default\_placement**)。

ストレージポリシーは、Ceph Object Gateway クライアントに対し、ストレージストラテジーにアクセスする手段を提供します。つまり、SSD、SAS ドライブ、SATA ドライブなどの特定のタイプのストレージをターゲットに設定する機能があります。持続性、レプリケーション、イレイジャーコーディングなどを確保するための特定の手法。詳細は、Red Hat Ceph Storage 3 の [ストレージ戦略](#) を参照してください。

ストレージポリシーを作成するには、以下の手順に従います。

1. 必要なストレージストラテジーを使用して、新しいプールの **.rgw.buckets.special** を作成します。たとえば、イレイジャーコーディング、特定の CRUSH ルールセット、レプリカ数、および **pg\_num** 数および **pgp\_num** 数でカスタマイズしたプールなどです。
2. ゾーングループの設定を取得して、これをファイルに保存します (例: **zonegroup.json**)。

### Syntax

```
[root@master-zone]# radosgw-admin zonegroup --rgw-zonegroup=<zonegroup_name> get > zonegroup.json
```

### 例

```
[root@master-zone]# radosgw-admin zonegroup --rgw-zonegroup=default get > zonegroup.json
```

3. **zonegroup.json** ファイルの **placement\_target** の下に、特別な **special-placement** を追加します。

```
{
  "name": "default",
  "api_name": "",
  "is_master": "true",
  "endpoints": [],
  "hostnames": [],
  "master_zone": "",
  "zones": [{
    "name": "default",
    "endpoints": [],
    "log_meta": "false",
    "log_data": "false",
    "bucket_index_max_shards": 5
  }],
  "placement_targets": [{
    "name": "default-placement",
```

```
"tags": []
}, {
  "name": "special-placement",
  "tags": []
}],
"default_placement": "default-placement"
}
```

4. 変更された **zonegroup.json** ファイルでゾーングループを設定します。

```
[root@master-zone]# radosgw-admin zonegroup set < zonegroup.json
```

5. ゾーン設定を取得して、これをファイル (例: **zone.json**) に保存します。

```
[root@master-zone]# radosgw-admin zone get > zone.json
```

6. ゾーンファイルを編集し、**placement\_pool** に新しい配置ポリシーキーを追加します。

```
{
  "domain_root": ".rgw",
  "control_pool": ".rgw.control",
  "gc_pool": ".rgw.gc",
  "log_pool": ".log",
  "intent_log_pool": ".intent-log",
  "usage_log_pool": ".usage",
  "user_keys_pool": ".users",
  "user_email_pool": ".users.email",
  "user_swift_pool": ".users.swift",
  "user_uid_pool": ".users.uid",
  "system_key": {
    "access_key": "",
    "secret_key": ""
  },
  "placement_pools": [{
    "key": "default-placement",
    "val": {
      "index_pool": ".rgw.buckets.index",
      "data_pool": ".rgw.buckets",
      "data_extra_pool": ".rgw.buckets.extra"
    }
  }, {
    "key": "special-placement",
    "val": {
      "index_pool": ".rgw.buckets.index",
      "data_pool": ".rgw.buckets.special",
      "data_extra_pool": ".rgw.buckets.extra"
    }
  }
]}
}
```

7. 新しいゾーン設定を設定します。

```
[root@master-zone]# radosgw-admin zone set < zone.json
```



- ゾーングループのマップを更新します。

```
[root@master-zone]# radosgw-admin period update --commit
```

**special-placement** エントリは **placement\_target** として一覧表示されます。

要求の実行時にストレージポリシーを指定するには、以下を実行します。

以下に例を示します。

```
$ curl -i http://10.0.0.1/swift/v1/TestContainer/file.txt -X PUT -H "X-Storage-Policy: special-placement"
-H "X-Auth-Token: AUTH_rgwtxxxxxx"
```

### 3.3. インデックスのないバケットの作成

作成されたバケットがバケットインデックスを使用せずに、オブジェクトのインデックスを格納する、つまりインデックスレスバケットを配置先として設定することができます。データのレプリケーションや一覧表示を使用しない配置ターゲットは、インデックスレスバケットを実装することができます。

インデックスレスバケットは、配置ターゲットが特定のバケット内のオブジェクトを追跡しないメカニズムです。これにより、オブジェクト書き込みが発生するたびに発生するリソース競合が削除され、Ceph Object Gateway が Ceph Storage クラスターに必要なラウンドトリップの数を減らします。これにより、同時操作や、小規模のオブジェクト書き込みパフォーマンスに正当な影響を与える可能性があります。

配置ターゲットをインデックスレスとして指定するには、以下の手順に従います。

- zone.json** の設定を取得します。

```
$ radosgw-admin zone get --rgw-zone=<zone> > zone.json
```

- 新しい配置対象を追加するか、既存の配置対象を変更して **"index\_type": 1** を持つようにすることで、**zone.json** を変更します。たとえば以下のようになります。

```
"placement_pools": [
  {
    "key": "default-placement",
    "val": {
      "index_pool": "default.rgw.buckets.index",
      "data_pool": "default.rgw.buckets.data",
      "data_extra_pool": "default.rgw.buckets.non-ec",
      "index_type": 1,
      "compression": ""
    }
  },
  {
    "key": "indexless",
    "val": {
      "index_pool": "default.rgw.buckets.index",
      "data_pool": "default.rgw.buckets.data",
      "data_extra_pool": "default.rgw.buckets.non-ec",
      "index_type": 1
    }
  }
]
```

```

    }
  }
],

```

3. **zone.json** の設定を設定します。

```
$ radosgw-admin zone set --rgw-zone=<zone> --infile zone.json
```

4. 新しい配置ターゲットを作成している場合は、**zonegroup** が新しい配置ターゲットを参照していることを確認します。

```
$ radosgw-admin zonegroup get --rgw-zonegroup=<zonegroup> > zonegroup.json
```

5. ゾーングループの **default\_placement** を設定します。

```
$ radosgw-admin zonegroup placement default --placement-id indexless
```

6. 必要に応じて **zonegroup.json** を変更します。以下に例を示します。

```

"placement_targets": [
  {
    "name": "default-placement",
    "tags": []
  },
  { "name": "indexless",
    "tags": []
  }
],
"default_placement": "default-placement",

```

```
$ radosgw-admin zonegroup set --rgw-zonegroup=<zonegroup> < zonegroup.json
```

7. クラスタがマルチサイト設定にある場合は、期間を更新し、コミットします。

```
$ radosgw-admin period update --commit
```

この例では、**"indexless"** ターゲットで作成されたバケットはインデックスレスバケットです。



### 重要

バケットインデックスはバケットの正しい状態を反映せず、これらのバケットを一覧表示してもオブジェクトの一覧を正しく返しません。これは複数の機能に影響します。具体的には、バケットインデックスが変更情報の保存に使用されていないため、これらのバケットはマルチゾーン環境では同期されません。この機能にはバケットインデックスが必要になるため、インデックスレスバケットで S3 オブジェクトのバージョン管理を使用することは推奨されません。



### 注記

インデックスレスバケットを使用すると、単一バケットのオブジェクトの最大数の上限が削除されます。



## 注記

インデックスレスバケットのオブジェクトは NFS から一覧表示できない

### 3.4. バケットシャーディングの設定

Ceph Object Gateway は、バケットインデックスデータをインデックスプール (**index\_pool**) に格納します。デフォルトは **.rgw.buckets.index** です。バケットあたりの最大オブジェクト数のクォータを設定せずに、クライアントが数十万から数百万のオブジェクトを1つのバケットに入れると、インデックスプールのパフォーマンスが著しく低下します。

**バケットインデックスのシャーディング** は、バケットあたりのオブジェクト数が多い場合のパフォーマンスのボトルネックを防ぐのに役立ちます。

新規バケットのバケットインデックスシャーディングを設定したり、既存のバケットでバケットインデックスを変更したりすることができます。

バケットインデックスのシャード化を設定するには、以下を実行します。

- 単純な設定の新規バケットの場合は、**rgw\_override\_bucket\_index\_max\_shards** オプションを使用します。「[簡易設定でのバケットインデックスシャードの設定](#)」を参照してください。
- マルチサイト設定の新規バケットの場合は、**bucket\_index\_max\_shards** オプションを使用します。「[マルチサイト設定でのバケットインデックスのシャード化の設定](#)」を参照してください。

バケットを再シャード化するには、以下を実行します。

- 動的な場合:「[バケットインデックスの動的再シャーディング](#)」を参照してください。
- 手動で、「[バケットインデックスの手動再シャーディング](#)」を参照してください。
- マルチサイト設定で [マルチサイトを使用した手動によるバケットのリシャード化](#) を参照してください。

#### 3.4.1. バケットシャーディングの制限



## 重要

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

- **シャードが必要になる前に1つのバケット内のオブジェクトの最大数:** Red Hat は、バケットインデックスのシャードごとに最大 102,400 個のオブジェクトを推奨しています。シャーディングを最大限に活用するには、Ceph Object Gateway バケットインデックスプールの十分な数の OSD を提供します。
- **シャード化の使用時の最大オブジェクト数:** 以前のテストに基づき、現在サポートされているバケットインデックスシャードの数は 65521 です。Red Hat の品質保証は、バケットシャーディングで完全なスケーラビリティテストを実施していません。

#### 3.4.2. 簡易設定でのバケットインデックスシャードの設定

すべての新規バケットでバケットインデックスシャードを有効にし、設定するには、**rgw\_override\_bucket\_index\_max\_shards** パラメーターを使用します。パラメーターを次のように設定します。

- バケットインデックスシャード化を無効にする場合は **0**。これがデフォルト値になります。
- **0** より大きい値を有効にすると、バケットシャード化が有効になり、シャードの最大数が設定されます。

#### 前提条件

- [バケットシャーディングの制限](#) を読んでいる。

#### 手順

1. 推奨されるシャード数を計算します。これを行うには、以下の式を使用します。

```
number of objects expected in a bucket / 100,000
```

シャードの最大数は 65521 であることに注意してください。

2. Ceph 設定ファイルに **rgw\_override\_bucket\_index\_max\_shards** を追加します。

```
rgw_override_bucket_index_max_shards = value
```

**value** を、直前の手順で計算したシャードの推奨数に置き換えます。以下に例を示します。

```
rgw_override_bucket_index_max_shards = 10
```

- Ceph Object Gateway のすべてのインスタンスにバケットインデックスシャードを設定するには、**[global]** セクションに **rgw\_override\_bucket\_index\_max\_shards** を追加します。
  - Ceph Object Gateway の特定のインスタンスに対してのみバケットインデックスのシャーディングを設定するには、インスタンスの下に **rgw\_override\_bucket\_index\_max\_shards** を追加します。
3. Ceph Object Gateway を再起動します。

```
$ sudo service radosgw restart id=rgw.hostname
```

**hostname** を、Ceph Object Gateway が実行されているノードの短いホスト名に置き換えます。

#### 関連情報

- [バケットインデックスの動的再シャーディング](#)
- [バケットインデックスの手動再シャーディング](#)

### 3.4.3. マルチサイト設定でのバケットインデックスのシャード化の設定

マルチサイト設定では、各ゾーンに異なる **index\_pool** を設定して、フェイルオーバーを管理できます。1つのゾーングループのゾーンに一貫したシャード数を設定するには、そのゾーングループの設定に **bucket\_index\_max\_shards** を設定します。パラメーターを次のように設定します。

- バケットインデックスシャード化を無効にする場合は **0**。これがデフォルト値になります。
- **0** より大きい値を有効にすると、バケットシャード化が有効になり、シャードの最大数が設定されます。



### 注記

SSD ベースの OSD の CRUSH ルールセットにインデックスプール (該当する場合は各ゾーン) をマッピングすることも、バケットインデックスのパフォーマンスに役立つ可能性があります。

### 前提条件

- [バケットシャーディングの制限](#) を読んでいます。

### 手順

1. 推奨されるシャード数を計算します。これを行うには、以下の式を使用します。

```
number of objects expected in a bucket / 100,000
```

シャードの最大数は 65521 であることに注意してください。

2. ゾーングループ設定を **zonegroup.json** ファイルに展開します。

```
$ radosgw-admin zonegroup get > zonegroup.json
```

3. **zonegroup.json** ファイルで、名前付きゾーンごとに **bucket\_index\_max\_shards** を設定します。

```
bucket_index_max_shards = value
```

**value** を、直前の手順で計算したシャードの推奨数に置き換えます。以下に例を示します。

```
bucket_index_max_shards = 10
```

4. ゾーングループをリセットします。

```
$ radosgw-admin zonegroup set < zonegroup.json
```

5. 期間を更新します。

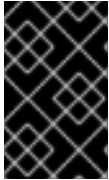
```
$ radosgw-admin period update --commit
```

### 関連情報

- [マルチサイトでのバケットの手動再シャーディング](#)

### 3.4.4. バケットインデックスの動的再シャーディング

動的バケットの再シャーディングのプロセスは、すべての Ceph Object Gateway バケットを定期的にチェックし、再シャーディングを必要とするバケットを検出します。バケットが `rgw_max_objs_per_shard` パラメーターで指定された値よりも大きい場合、Ceph Object Gateway はバックグラウンドでバケットを動的に再シャードします。`rgw_max_objs_per_shard` のデフォルト値は、シャードごとに 100k オブジェクトです。



#### 重要

現在、Red Hat は、マルチサイト設定で動的バケットの再シャーディングをサポートしていません。このような設定でシャードバケットのインデックスを再作成するには、[マルチサイトを使用した手動によるバケットのリシャード化](#) を参照してください。

#### 前提条件

- [バケットシャーディングの制限](#) を読んでいます。

#### 手順

- バケットインデックスの動的再シャーディングを有効にするには、以下を実行します。
  1. Ceph 設定ファイルの `rgw_dynamic_resharding` 設定を `true` (デフォルト値) に設定します。
  2. **任意です。** 必要に応じて、Ceph 設定ファイルの以下のパラメーターを変更します。
    - `rgw_reshard_num_logs`: 再シャードログのシャードの数。デフォルト値は **16** です。
    - `rgw_reshard_bucket_lock_duration`: リシャード中にバケットのロックの期間。デフォルト値は **120** 秒です。
    - `rgw_dynamic_resharding`: 動的リシャードを有効または無効にします。デフォルト値は `true` です。
    - `rgw_max_objs_per_shard`: シャードごとのオブジェクトの最大数。デフォルト値は、シャードごとに **100000** オブジェクトです。
    - `rgw_reshard_thread_interval`: 再シャード処理のラウンド間の最大時間。デフォルト値は **600** 秒です。
- バケットを再シャーディングキューに追加するには、以下を実行します。

```
radosgw-admin reshard add --bucket BUCKET_NAME --num-shards NUMBER
```

以下を置き換えます。

- 再シャードするバケットの名前を含む `BUCKET_NAME`
- 新しいシャード数の `NUMBER`

以下に例を示します。

```
$ radosgw-admin reshard add --bucket data --num-shards 10
```

- 再シャーディングキューを一覧表示するには、以下を実行します。

```
$ radosgw-admin reshard list
```

- バケットの再シャーディングステータスを確認するには、以下のコマンドを実行します。

```
radosgw-admin reshard status --bucket BUCKET_NAME
```

以下を置き換えます。

- 再シャードするバケットの名前を持つ **BUCKET\_NAME**

以下に例を示します。

```
$ radosgw-admin reshard status --bucket data
```



#### 注記

**radosgw-admin reshard status** コマンドは、次のステータス識別子のいずれかを表示します。

- **not-resharding**
- **in-progress**
- **done**

- 再シャーディングキューのエントリーを即座に処理するには、以下を実行します。

```
$ radosgw-admin reshard process
```

- 保留中のバケットの再シャーディングをキャンセルするには、以下を実行します。

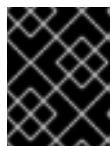
```
radosgw-admin reshard cancel --bucket BUCKET_NAME
```

以下を置き換えます。

- 保留中のバケットの名前を持つ **BUCKET\_NAME**。

以下に例を示します。

```
$ radosgw-admin reshard cancel --bucket data
```



#### 重要

保留中の再シャード操作のみをキャンセルできます。継続中のリシャード操作をキャンセルしないでください。

- Red Hat Ceph Storage 3.1 およびそれ以前のバージョンを使用している場合は、[再シャーディング後の古いインスタンスのクリーニング](#)のセクションで説明されているように、古いバケットエントリーを削除します。

- [バケットインデックスの手動再シャーディング](#)
- [簡易設定でのバケットインデックスシャードの設定](#)

### 3.4.5. バケットインデックスの手動再シャーディング

バケットのサイズが初期設定に対して最適化されている場合は、**radosgw-admin bucket reshard** コマンドを使用してバケットインデックスプールを再シャードします。このコマンドは、以下のようになります。

- 指定されたバケットのバケットインデックスオブジェクトの新しいセットを作成します。
- これらのバケットインデックスオブジェクトにオブジェクトエントリーを分散します。
- 新規バケットインスタンスを作成します。
- 新規インデックス操作すべてが新規バケットインデックスを通過するように、新しいバケットインスタンスをバケットとリンクします。
- 古いバケット ID および新しいバケット ID をコマンド出力に出力します。



#### 重要

この手順は、簡単な設定でのみ使用してください。マルチサイト設定でバケットを再シャーディングするには、[マルチサイトでのバケットの手動再シャーディング](#) を参照してください。

#### 前提条件

- [バケットシャーディングの制限](#) を読んでいます。

#### 手順

1. 元のバケットインデックスをバックアップします。

```
radosgw-admin bi list --bucket=BUCKET > BUCKET.list.backup
```

以下を置き換えます。

- **BUCKET** を、再シャードするバケットの名前に

たとえば、**data** という名前のバケットの場合は、以下を入力します。

```
$ radosgw-admin bi list --bucket=data > data.list.backup
```

2. バケットインデックスを再シャード化します。

```
radosgw-admin bucket reshard --bucket=BUCKET --num-shards=NUMBER
```

以下を置き換えます。

- **BUCKET** を、再シャードするバケットの名前に
- **NUMBER** を、新しいシャード数に



たとえば、**data** という名前のバケットおよび必要なシャード数が **100** の場合は、以下を入力します。

```
$ radosgw-admin bucket reshard --bucket=data --num-shards=100
```

- Red Hat Ceph Storage 3.1 およびそれ以前のバージョンを使用している場合は、[再シャーディング後の古いインスタンスのクリーニング](#) のセクションで説明されているように、古いバケットエントリを削除します。

## 関連情報

- [バケットインデックスの動的再シャーディング](#)
- [簡易設定でのバケットインデックスシャードの設定](#)
- [マルチサイト設定でのバケットインデックスシャーディングの設定](#)
- [マルチサイトでのバケットの手動再シャーディング](#)

### 3.4.6. 再シャーディングを行った後に古いインスタンスの消去

Red Hat Ceph Storage 3.1 以前のバージョンでは、再シャーディングプロセスでは、バケットエントリーの古いインスタンスは自動的にクリーンアップされません。これらの古いインスタンスは、手動でクリーンアップされない場合にクラスターのパフォーマンスに影響を与える可能性があります。



#### 重要

この手順は、マルチサイトクラスターではない単純な設定にのみ使用するようになっています。

## 前提条件

- Ceph Object Gateway がインストールされている。

## 手順

- 古いインスタンスを一覧表示します。

```
$ radosgw-admin reshard stale-instances list
```

- 古いインスタンスを削除します。

```
$ radosgw-admin reshard stale-instances rm
```

## 3.5. 圧縮の有効化

Ceph Object Gateway は、Ceph の圧縮プラグインを使用してアップロードしたオブジェクトのサーバー側の圧縮をサポートします。これには、以下が含まれます。

- **zlib**: サポート対象。
- **snappy**: テクノロジープレビュー。

- **zstd**: テクノロジープレビュー。



## 注記

圧縮プラグイン **snappy** および **zstd** はテクノロジープレビュー機能であり、Red Hat が品質保証テストを完了していないため、完全にはサポートされていません。

## 設定

ゾーンの配置ターゲットで圧縮を有効にするには、**--compression=<type>** オプションを **radosgw-admin zone placement modify** コマンドに指定します。圧縮 **type** は、新しいオブジェクトデータの書き込み時に使用する圧縮プラグインの名前を指します。

圧縮される各オブジェクトは圧縮タイプを保存します。設定を変更しても、既存の圧縮オブジェクトを展開します。また、Ceph Object Gateway は強制的に既存オブジェクトを再圧縮する訳ではありません。

この圧縮設定は、この配置ターゲットを使用してバケットにアップロードされるすべての新規オブジェクトに適用されます。

ゾーンの配置ターゲットで圧縮を無効にするには、**--compression=<type>** オプションを **radosgw-admin zone placement modify** コマンドに指定して、空の文字列または **none** を指定します。

以下に例を示します。

```
$ radosgw-admin zone placement modify --rgw-zone=default --placement-id=default-placement --compression=zlib
{
...
  "placement_pools": [
    {
      "key": "default-placement",
      "val": {
        "index_pool": "default.rgw.buckets.index",
        "data_pool": "default.rgw.buckets.data",
        "data_extra_pool": "default.rgw.buckets.non-ec",
        "index_type": 0,
        "compression": "zlib"
      }
    }
  ],
...
}
```

圧縮の有効化または無効化後に Ceph Object Gateway インスタンスを再起動して、変更を反映します。



## 注記

Ceph Object Gateway は **デフォルト** ゾーンとプールのセットを作成します。実稼働デプロイメントの場合は、**実稼働用 Ceph Object Gateway** の **レルムの作成** セクションを参照してください。 [Multisite](#) も参照してください。

## 統計

既存のコマンドおよび API は、圧縮されていないデータに基づいてオブジェクトおよびバケットサイズを引き続き報告しますが、**radosgw-admin bucket stats** コマンドには指定されたバケットの圧縮統計が含まれます。

```
$ radosgw-admin bucket stats --bucket=<name>
{
  ...
  "usage": {
    "rgw.main": {
      "size": 1075028,
      "size_actual": 1331200,
      "size_utilized": 592035,
      "size_kb": 1050,
      "size_kb_actual": 1300,
      "size_kb_utilized": 579,
      "num_objects": 104
    }
  },
  ...
}
```

**size\_utilized** フィールドおよび **size\_kb\_utilized** フィールドは、それぞれ圧縮データの合計サイズをバイト単位で表します。

### 3.6. ユーザー管理

Ceph Object Storage ユーザー管理とは、Ceph Storage Cluster のクライアントアプリケーションとしての Ceph Object Gateway ではなく、Ceph Object Storage サービスのクライアントアプリケーションであるユーザーを指します。クライアントアプリケーションが Ceph Object Gateway サービスと対話できるようにするには、ユーザー、アクセスキー、およびシークレットを作成する必要があります。

ユーザータイプが2つあります。

- **User:** user という用語は、S3 インターフェイスのユーザーを反映しています。
- **Subuser:** subuser という用語は、Swift インターフェイスのユーザーを反映しています。サブユーザーがユーザーに関連付けられています。

ユーザーとサブユーザーを作成、変更、表示、一時停止、および削除できます。



#### 重要

マルチサイトデプロイメントでユーザーを管理する場合は、常にマスターゾーングループのマスターゾーン内の Ceph Object Gateway ノードで **radosgw-admin** コマンドを実行して、ユーザーがマルチサイトクラスター全体で同期するようにします。マルチサイトクラスター上のユーザーをセカンダリーゾーンまたはセカンダリーゾーングループから作成、変更、または削除しないでください。このドキュメントでは、マスターゾーングループのマスターゾーンにあるホストのコマンドライン規則として **[root@master-zone]#** を使用しています。

ユーザーおよびサブユーザー ID の作成に加え、ユーザーの表示名およびメールアドレスを追加することができます。キーおよびシークレットを指定するか、キーおよびシークレットを自動的に生成できます。キーを生成または指定した際には、ユーザー ID が S3 キータイプに対応し、サブユーザー ID が Swift キータイプに対応することに注意してください。Swift キーには、アクセスレベルの **read**、**write**、**readwrite**、および **full** もあります。

ユーザー管理コマンドライン構文は、通常、パターン **user <command> <user-id>** に従います。ここで、**<user-id>** は、**--uid=** オプションの後にユーザー ID (S3) が続くか、**--subuser=** オプションの後にユーザー名 (Swift) が続きます。以下に例を示します。

```
[root@master-zone]# radosgw-admin user <create|modify|info|rm|suspend|enable|check|stats> <--uid={id}|--subuser={name}> [other-options]
```

実行するコマンドによっては、追加のオプションが必要になる場合があります。

### 3.6.1. マルチテナンシー

Red Hat Ceph Storage 2 以降では、Ceph Object Gateway は S3 および Swift API の両方に対するマルチテナンシーをサポートします。この場合、各ユーザーとバケットはテナント下に置かれます。マルチテナンシーは、複数のテナントが共通のバケット名 (例: test、main など) を使用している場合に、名前空間のクラッシュを防ぎます。

各ユーザーとバケットはテナントの下にあります。下位互換性のために、空の名前を持つレガシーテナントが追加されます。テナントを具体的に指定せずにバケットを参照する場合は常に、Swift API はレガシーテナントを想定します。既存のユーザーもレガシーテナントに保存されるため、以前のリリースと同様にバケットとオブジェクトにアクセスします。

このようなテナントの場合、テナント自体には何の操作もありません。ユーザーが管理されている場合には、必要に応じて表示および非表示になります。明示的なテナントを持つユーザーを作成、変更、および削除するには、追加のオプション **--tenant** を指定するか、**radosgw-admin** コマンドのパラメーターで構文 "**<tenant>\$<user>**" を使用します。

S3 用のユーザー **testx\$tester** を作成するには、以下を実行します。

```
[root@master-zone]# radosgw-admin --tenant testx --uid tester \
    --display-name "Test User" --access_key TESTER \
    --secret test123 user create
```

Swift のユーザー **testx\$tester** を作成するには、以下のいずれかを実行します。

```
[root@master-zone]# radosgw-admin --tenant testx --uid tester \
    --display-name "Test User" --subuser tester:swift \
    --key-type swift --access full subuser create

[root@master-zone]# radosgw-admin key create --subuser 'testx$tester:swift' \
    --key-type swift --secret test123
```



#### 注記

明示的なテナントを持つサブユーザーは、シェルで引用する必要がありました。

### 3.6.2. ユーザーの作成

**user create** コマンドを使用して S3-interface ユーザーを作成します。ユーザー ID と表示名を指定する必要があります。メールアドレスを指定することもできます。key または secret を指定しないと、**radosgw-admin** によって自動的に生成されます。ただし、生成されたキー/シークレットのペアを使用しない場合は、キーやシークレットを指定できます。

```
[root@master-zone]# radosgw-admin user create --uid=<id> \
```

```
[--key-type=<type>] [--gen-access-key|--access-key=<key>]\
[--gen-secret | --secret=<key>] \
[--email=<email>] --display-name=<name>
```

以下に例を示します。

```
[root@master-zone]# radosgw-admin user create --uid=janedoe --display-name="Jane Doe" --
email=jane@example.com
```

```
{ "user_id": "janedoe",
  "display_name": "Jane Doe",
  "email": "jane@example.com",
  "suspended": 0,
  "max_buckets": 1000,
  "auid": 0,
  "subusers": [],
  "keys": [
    { "user": "janedoe",
      "access_key": "11BS02LGFB6AL6H1ADMW",
      "secret_key": "vzCEkuryfn060dfee4fgQPqFrncKEIkh3ZcdOANY"}],
  "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": []}
```



### 重要

キーの出力を確認します。**radosgw-admin** が JSON エスケープ (\) 文字を生成することがあり、一部のクライアントは JSON エスケープ文字の処理方法を知りません。対処法には、JSON エスケープ文字 (\) の削除、文字列の引用符でのカプセル化、キーの再生成、JSON エスケープ文字が含まれていないことの確認、またはキーとシークレットの手動指定が含まれます。

### 3.6.3. サブユーザーの作成

サブユーザー (Swift インターフェイス) を作成するには、ユーザー ID (**--uid={username}**)、サブユーザー ID、およびサブユーザーのアクセスレベルを指定する必要があります。key または secret を指定しないと、**radosgw-admin** によって自動的に生成されます。ただし、生成されたキー/シークレットのペアを使用しない場合は、キーやシークレットを指定できます。



### 注記

アクセス制御ポリシーも含まれるため、**full** は **readwrite** ではありません。

```
[root@master-zone]# radosgw-admin subuser create --uid={uid} --subuser={uid} --access=[ read |
write | readwrite | full ]
```

以下に例を示します。

```
[root@master-zone]# radosgw-admin subuser create --uid=janedoe --subuser=janedoe:swift --
access=full
```

```
{ "user_id": "janedoe",
  "display_name": "Jane Doe",
  "email": "jane@example.com",
  "suspended": 0,
  "max_buckets": 1000,
  "auid": 0,
  "subusers": [
    { "id": "janedoe:swift",
      "permissions": "full-control"}],
  "keys": [
    { "user": "janedoe",
      "access_key": "11BS02LGFB6AL6H1ADMW",
      "secret_key": "vzCEkuryfn060dfee4fgQPqFrncKEIkh3ZcdOANY"}],
  "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": []}
```

### 3.6.4. ユーザー情報の取得

ユーザーに関する情報を取得するには、**ユーザー情報** とユーザー ID (**--uid={username}**) を指定する必要があります。

```
# radosgw-admin user info --uid=janedoe
```

### 3.6.5. ユーザー情報の変更

ユーザーに関する情報を変更するには、ユーザー ID (**--uid={username}**) と変更する属性を指定する必要があります。変更は通常、キーとシークレット、電子メールアドレス、表示名、およびアクセスレベルに対して行われます。以下に例を示します。

```
[root@master-zone]# radosgw-admin user modify --uid=janedoe / --display-name="Jane E. Doe"
```

サブユーザーの値を変更するには、**subuser modify** とサブユーザー ID を指定します。以下に例を示します。

```
[root@master-zone]# radosgw-admin subuser modify --subuser=janedoe:swift / --access=full
```

### 3.6.6. ユーザーの有効化および一時停止

ユーザーを作成すると、ユーザーはデフォルトで有効になります。ただし、ユーザー特権を一時停止して、後で再度有効にすることができます。ユーザーを一時停止するには、**user suspend** とユーザー ID を指定します。

```
[root@master-zone]# radosgw-admin user suspend --uid=johndoe
```

一時停止ユーザーを再度有効にするには、**user enable** とユーザー ID を指定します。

```
[root@master-zone]# radosgw-admin user enable --uid=johndoe
```



#### 注記

ユーザーを無効にすると、サブユーザーが無効になります。

### 3.6.7. ユーザーの削除

ユーザーを削除すると、ユーザーとサブユーザーはシステムから削除されます。ただし、必要に応じてサブユーザーのみを削除できます。ユーザー (およびサブユーザー) を削除するには、**user rm** とユーザー ID を指定します。

```
[root@master-zone]# radosgw-admin user rm --uid=<uid> [--purge-keys] [--purge-data]
```

以下に例を示します。

```
[root@master-zone]# radosgw-admin user rm --uid=johndoe --purge-data
```

サブユーザーのみを削除するには、**subuser rm** およびサブユーザー名を指定します。

```
[root@master-zone]# radosgw-admin subuser rm --subuser=johndoe:swift --purge-keys
```

オプションには以下が含まれます。

- **データのパージ: --purge-data** オプションは、UID に関連付けられたすべてのデータをパージします。
- **Purge Keys: --purge-keys** オプションは、UID に関連付けられたすべてのキーをパージします。

### 3.6.8. サブユーザーの削除

サブユーザーを削除すると、Swift インターフェイスへのアクセスが削除されます。ユーザーはシステムに残ります。Ceph Object Gateway: サブユーザーを削除するには、**subuser rm** およびサブユーザー ID を指定します。

```
[root@master-zone]# radosgw-admin subuser rm --subuser=johndoe:test
```

オプションには以下が含まれます。

- **Purge Keys: --purge-keys** オプションは、UID に関連付けられたすべてのキーをパーズします。

### 3.6.9. ユーザーの名前変更

ユーザーの名前を変更するには、**radosgw-admin user rename** コマンドを使用します。このコマンドにかかる時間は、ユーザーが持つバケットおよびオブジェクトの数によって異なります。この数字が大きい場合、Red Hat は、**screen** パッケージが提供する **Screen** ユーティリティーでコマンドを使用することを推奨します。

#### 前提条件

- 稼働中の Ceph クラスタ。
- **root** または **sudo** アクセス
- インストールされた Ceph Object Gateway

#### 手順

1. ユーザーの名前を変更します。

```
radosgw-admin user rename --uid=current-user-name --new-uid=new-user-name
```

たとえば、名前 **user1** を **user2** に変更するには、以下を実行します。

```
# radosgw-admin user rename --uid=user1 --new-uid=user2
{
  "user_id": "user2",
  "display_name": "user 2",
  "email": "",
  "suspended": 0,
  "max_buckets": 1000,
  "auid": 0,
  "subusers": [],
  "keys": [
    {
      "user": "user2",
      "access_key": "59EKHI6AI9F8WOW8JQZJ",
      "secret_key": "XH0uY3rKCUcuL73X0ftjXbZqUbK0cavD11rD8MsA"
    }
  ],
  "swift_keys": [],
  "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"
}

```

ユーザーがテナント内にある場合は、`tenant$user-name` 形式を使用します。

```

radosgw-admin user rename --uid=tenant$current-user-name --new-uid=tenant$new-user-name

```

たとえば、`test` テナント内の `user1` の名前を `user2` に変更するには、以下を実行します。

```

# radosgw-admin user rename --uid=test$user1 --new-uid=test$user2

```

```

1000 objects processed in tvtester1. Next marker 80_tVtester1_99
2000 objects processed in tvtester1. Next marker 64_tVtester1_44
3000 objects processed in tvtester1. Next marker 48_tVtester1_28
4000 objects processed in tvtester1. Next marker 2_tVtester1_74
5000 objects processed in tvtester1. Next marker 14_tVtester1_53
6000 objects processed in tvtester1. Next marker 87_tVtester1_61
7000 objects processed in tvtester1. Next marker 6_tVtester1_57
8000 objects processed in tvtester1. Next marker 52_tVtester1_91
9000 objects processed in tvtester1. Next marker 34_tVtester1_74
9900 objects processed in tvtester1. Next marker 9_tVtester1_95
1000 objects processed in tvtester2. Next marker 82_tVtester2_93
2000 objects processed in tvtester2. Next marker 64_tVtester2_9
3000 objects processed in tvtester2. Next marker 48_tVtester2_22
4000 objects processed in tvtester2. Next marker 32_tVtester2_42
5000 objects processed in tvtester2. Next marker 16_tVtester2_36
6000 objects processed in tvtester2. Next marker 89_tVtester2_46
7000 objects processed in tvtester2. Next marker 70_tVtester2_78
8000 objects processed in tvtester2. Next marker 51_tVtester2_41
9000 objects processed in tvtester2. Next marker 33_tVtester2_32
9900 objects processed in tvtester2. Next marker 9_tVtester2_83
{
  "user_id": "test$user2",
  "display_name": "User 2",
  "email": "",
  "suspended": 0,
  "max_buckets": 1000,
  "auid": 0,
  "subusers": [],
  "keys": [
    {
      "user": "test$user2",
      "access_key": "user2",
      "secret_key": "123456789"
    }
  ],
}

```

```

"swift_keys": [],
"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. ユーザーの名前が正常に変更されたことを確認します。

```
radosgw-admin user info --uid=new-user-name
```

以下に例を示します。

```
# radosgw-admin user info --uid=user2
```

ユーザーがテナント内にある場合は、**tenant\$user-name** 形式を使用します。

```
radosgw-admin user info --uid=tenant$new-user-name
```

```
# radosgw-admin user info --uid=test$user2
```

## 関連情報

- man ページの **screen(1)**

### 3.6.10. キーの作成

ユーザーのキーを作成するには、**key create** を指定する必要があります。ユーザーには、ユーザー ID と **s3** キータイプを指定します。サブユーザーのキーを作成するには、サブユーザー ID と **swift** キータイプを指定する必要があります。以下に例を示します。

```
[root@master-zone]# radosgw-admin key create --subuser=johndoe:swift --key-type=swift --gen-secret
```

```
{ "user_id": "johndoe",
  "rados_uid": 0,
  "display_name": "John Doe",
```

```
"email": "john@example.com",
"suspended": 0,
"subusers": [
  { "id": "johndoe:swift",
    "permissions": "full-control"}],
"keys": [
  { "user": "johndoe",
    "access_key": "QFAMEDSJP5DEKJO0DDXY",
    "secret_key": "iaSFLDVvDdQt6IkNzHyW4fPLZugBAI1g17LO0+87"}],
"swift_keys": [
  { "user": "johndoe:swift",
    "secret_key": "E9T2rUZNu2gxUjcwUBO8n\Ev4KX6V\GprEuH4qhu1"}]}
```

### 3.6.11. アクセスキーの追加および削除

ユーザーおよびサブユーザーには、S3 インターフェイスおよび Swift インターフェイスを使用するためのアクセスキーが必要です。ユーザーまたはサブユーザーを作成し、アクセスキーおよびシークレットを指定しない場合、キーおよびシークレットが自動的に生成されます。キーを作成し、アクセスキーやシークレットを指定または生成することができます。アクセスキーおよびシークレットを削除することもできます。オプションには以下が含まれます。

- **--secret=<key>** は、秘密鍵を指定します (例: 手動で生成)。
- **--gen-access-key** は、ランダムなアクセスキーを生成します (デフォルトでは S3 ユーザー用)。
- **--gen-secret** は、ランダムな秘密鍵を生成します。
- **--key-type=<type>** は、キータイプを指定します。オプションは swift、s3 です。

キーを追加するには、ユーザーを指定します。

```
[root@master-zone]# radosgw-admin key create --uid=johndoe --key-type=s3 --gen-access-key --gen-secret
```

鍵とシークレットを指定することもできます。

アクセスキーを削除するには、ユーザーとキーを指定する必要があります。

1. 特定ユーザーのアクセスキーを検索します。

```
[root@master-zone]# radosgw-admin user info --uid=<testid>
```

アクセスキーは、出力の **"access\_key"** 値になります。以下に例を示します。

```
$ radosgw-admin user info --uid=johndoe
{
  "user_id": "johndoe",
  ...
  "keys": [
    {
      "user": "johndoe",
      "access_key": "0555b35654ad1656d804",
      "secret_key":
      "h7GhxuBLTrlhVUyxSPUKUV8r/2E14ngqJxD7iBdBYLhwluN30JaT3Q=="
    }
  ]
}
```

```
    }
  ],
  ...
}
```

2. 前の手順のユーザー ID とアクセスキーを指定して、アクセスキーを削除します。

```
[root@master-zone]# radosgw-admin key rm --uid=<user_id> --access-key <access_key>
```

以下に例を示します。

```
[root@master-zone]# radosgw-admin key rm --uid=johndoe --access-key
0555b35654ad1656d804
```

### 3.6.12. 管理機能の追加と削除

Ceph Storage Cluster は、ユーザーが REST API を介して管理機能を実行できるようにする管理 API を提供します。デフォルトでは、ユーザーはこの API にアクセスできません。ユーザーが管理機能を実行できるようにするには、ユーザーに管理機能を提供します。

ユーザーに管理機能を追加するには:

```
[root@master-zone]# radosgw-admin caps add --uid={uid} --caps={caps}
```

ユーザー、バケット、メタデータ、および使用状況 (使用率) に、読み取り、書き込み、またはすべての機能を追加できます。以下に例を示します。

```
--caps="[users|buckets|metadata|usage|zone]=[*|read|write|read, write]"
```

以下に例を示します。

```
[root@master-zone]# radosgw-admin caps add --uid=johndoe --caps="users=*"
```

ユーザーから管理機能を削除するには:

```
[root@master-zone]# radosgw-admin caps rm --uid=johndoe --caps={caps}
```

## 3.7. クォータ管理

Ceph Object Gateway を使用すると、ユーザーが所有するユーザーおよびバケットにクォータを設定することができます。クォータには、バケットのオブジェクトの最大数と、メガバイト単位のストレージの最大サイズが含まれます。

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

- **Quota Scope: --quota-scope** オプションは、クォータのスコープを設定します。オプションは **bucket** と **user** です。バケットクォータは、ユーザーが所有するバケットに適用されます。ユーザークォータはユーザーに適用されます。



### 重要

多数のオブジェクトを含むバケットは、深刻なパフォーマンスの問題を引き起こす可能性があります。1つのバケット内のオブジェクトの推奨最大数は100,000です。この数を増やすには、バケットインデックスシャーディングを設定します。詳細は、「[バケットシャーディングの設定](#)」を参照してください。

## 3.7.1. ユーザークォータの設定

クォータを有効にする前に、まずクォータパラメーターを設定する必要があります。以下に例を示します。

```
[root@master-zone]# radosgw-admin quota set --quota-scope=user --uid=<uid> [--max-objects=<num objects>] [--max-size=<max size>]
```

以下に例を示します。

```
radosgw-admin quota set --quota-scope=user --uid=johndoe --max-objects=1024 --max-size=1024
```

num オブジェクトおよび/または最大サイズの負の値は、特定のクォータ属性チェックが無効になっていることを意味します。

## 3.7.2. ユーザークォータの有効化および無効化

ユーザークォータを設定したら、これを有効にすることができます。以下に例を示します。

```
[root@master-zone]# radosgw-admin quota enable --quota-scope=user --uid=<uid>
```

有効なユーザークォータを無効にすることができます。以下に例を示します。

```
[root@master-zone]# radosgw-admin quota disable --quota-scope=user --uid=<uid>
```

## 3.7.3. バケットクォータの設定

バケットクォータは、指定された **uid** が所有するバケットに適用されます。ユーザーからは独立しています。

```
[root@master-zone]# radosgw-admin quota set --uid=<uid> --quota-scope=bucket [--max-objects=<num objects>] [--max-size=<max size>]
```

num オブジェクトおよび/または最大サイズの負の値は、特定のクォータ属性チェックが無効になっていることを意味します。

## 3.7.4. バケットクォータの有効化および無効化

バケットクォータを設定したら、それを有効にできます。以下に例を示します。

```
[root@master-zone]# radosgw-admin quota enable --quota-scope=bucket --uid=<uid>
```

有効なバケットクォータを無効にするには:

```
[root@master-zone]# radosgw-admin quota disable --quota-scope=bucket --uid=<uid>
```

### 3.7.5. クォータ設定の取得

ユーザー情報 API を使用して、各ユーザーのクォータ設定にアクセスすることができます。CLI インターフェイスを使用してユーザークォータ設定情報を読み取るには、以下を実行します。

```
# radosgw-admin user info --uid=<uid>
```

### 3.7.6. クォータウォーターを更新

クォータ統計は非同期で更新されます。すべてのユーザーおよびすべてのバケットのクォータ統計を手動で更新して、最新のクォータ統計を取得できます。

```
[root@master-zone]# radosgw-admin user stats --uid=<uid> --sync-stats
```

### 3.7.7. ユーザークォータ使用統計の取得

ユーザーが消費したクォータの量を確認するには、次の手順を実行します。

```
# radosgw-admin user stats --uid=<uid>
```



#### 注記

最新のデータを受け取るには、**--sync-stats** オプションを指定して **radosgw-admin user stats** を実行する必要があります。

### 3.7.8. クォータキャッシュ

クォータ統計は、各 Ceph Gateway インスタンスに対してキャッシュされます。複数のインスタンスがある場合、インスタンスごとにクォータのビューが異なるため、キャッシュによってクォータが完全に適用されないようにすることができます。これを制御するオプションは、**rgw bucket quota ttl**、**rgw user quota bucket sync interval**、および **rgw user quota sync interval** です。これらの値が高いほど、クォータ操作は効率的ですが、複数のインスタンスが同期しなくなります。これらの値が低いほど、複数のインスタンスは完全に近い形で適用されます。3 つすべてが 0 の場合には、クォータキャッシュは実質的に無効になり、複数のインスタンスで完全にクォータが適用されます。これらのオプションの詳細は、[4章 設定リファレンス](#) を参照してください。

### 3.7.9. グローバルクォータの読み取りおよび作成

ゾーングループマップでクォータ設定を読み書きできます。ゾーングループマップを取得するには、次のコマンドを実行します。

```
[root@master-zone]# radosgw-admin global quota get
```

グローバルクォータ設定は、**quota set**、**quota enable**、および **quota disable** コマンドに対応する **global quota** で操作できます。次に例を示します。

```
[root@master-zone]# radosgw-admin global quota set --quota-scope bucket --max-objects 1024
[root@master-zone]# radosgw-admin global quota enable --quota-scope bucket
```



### 注記

レulumと期間が存在するマルチサイト設定では、グローバルクォータへの変更は、**period update --commit** を使用してコミットする必要があります。期間が表示されていない場合、変更を有効にするには、Ceph Object Gateway を再起動する必要があります。

## 3.8. 用途

Ceph Object Gateway は、各ユーザーの使用状況をログに記録します。ユーザーの使用状況を日付の範囲内でも追跡できます。

オプションには以下が含まれます。

- **開始日: --start-date** オプションを使用すると、特定の開始日から使用統計をフィルターリングできます (形式: **yyyy-mm-dd[HH:MM:SS]**)。
- **終了日: --end-date** オプションを使用すると、特定の日付までの使用をフィルターリングできます (形式: **yyyy-mm-dd[HH:MM:SS]**)。
- **ログエントリ: --show-log-entries** オプションを使用すると、ログエントリを使用統計に含めるかどうかを指定できます (オプション: **true | false**)。



### 注記

分と秒で時間を指定できますが、1時間分解能で保存されます。

### 3.8.1. 使用方法の表示

使用状況の統計を表示するには、**usage show** を指定します。特定のユーザーの使用状況を表示するには、ユーザー ID を指定する必要があります。開始日、終了日、およびログエントリを表示するかどうかを指定することもできます。

```
# radosgw-admin usage show \
    --uid=johndoe --start-date=2012-03-01 \
    --end-date=2012-04-01
```

ユーザー ID を省略することで、すべてのユーザーの使用状況情報の概要も表示できます。

```
# radosgw-admin usage show --show-log-entries=false
```

### 3.8.2. トリムの使用方法

頻繁に使用すると、使用状況のログがストレージスペースを占有し始める可能性があります。すべてのユーザーおよび特定ユーザーの使用状況ログをトリミングできます。トリム操作の日付範囲を指定することもできます。

```
[root@master-zone]# radosgw-admin usage trim --start-date=2010-01-01 \
    --end-date=2010-12-31
```

```
[root@master-zone]# radosgw-admin usage trim --uid=johndoe
[root@master-zone]# radosgw-admin usage trim --uid=johndoe --end-date=2013-12-31
```

### 3.8.3. 孤立したオブジェクトの検索

通常、正常なストレージクラスターでは、リークオブジェクトは発生しないはずですが、場合によっては、リークオブジェクトが発生する可能性があります。たとえば、操作の途中で RADOS ゲートウェイがダウンした場合、一部の RADOS オブジェクトが孤立する可能性があります。また、未知のバグにより、これらの孤立したオブジェクトが発生する可能性があります。**radosgw-admin** コマンドは、これらの孤立したオブジェクトを検索してクリーンアップするためのツールを提供します。**--pool** オプションを使用すると、リークのある RADOS オブジェクトをスキャンするプールを指定できます。**--num-shards** オプションを使用すると、一時的なスキャンデータを保持するために使用するシャードの数を指定できます。

1. 新しいログプールを作成します。

例

```
# rados mkpool .log
```

2. 孤立したオブジェクトを検索します。

Syntax

```
# radosgw-admin orphans find --pool=<data_pool> --job-id=<job_name> [--num-shards=
<num_shards>] [--orphan-stale-secs=<seconds>]
```

例

```
# radosgw-admin orphans find --pool=.rgw.buckets --job-id=abc123
```

3. 検索データをクリーンアップします。

Syntax

```
# radosgw-admin orphans finish --job-id=<job_name>
```

例

```
# radosgw-admin orphans finish --job-id=abc123
```

## 3.9. バケット管理

ストレージ管理者は、Ceph Object Gateway を使用する場合は、バケットをユーザー間で移動して名前を変更することで、バケットを管理できます。

### 3.9.1. バケットの移動

**radosgw-admin bucket** ユーティリティーは、ユーザー間でバケットを移行する機能を提供します。これを実行するには、バケットを新規ユーザーにリンクし、バケットの所有権を新規ユーザーに変更します。



バケットを移動できます。

- テナントのない2人のユーザー間
- 2人のテナントユーザー間
- テナントのないユーザーとテナントユーザーとの間

### 3.9.1.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスタがある。
- Ceph Object Gateway がインストールされている。
- バケット
- さまざまなテナントユーザーとテナントのないユーザー

### 3.9.1.2. テナントのないユーザー間でのバケットの移動

**radosgw-admin bucket chown** コマンドは、バケットとそれに含まれるすべてのオブジェクトの所有権をあるユーザーから別のユーザーに変更する機能を提供します。これを行うには、バケットを現在のユーザーからリンク解除し、新しいユーザーにリンクして、バケットの所有権を新しいユーザーに変更します。

#### 手順

1. バケットを新規ユーザーにリンクします。

```
radosgw-admin bucket link --uid=user --bucket=bucket
```

以下を置き換えます。

- **user** を、バケットをリンクするユーザーのユーザー名に
- **bucket** を、名前を持つバケットに

たとえば、**data** バケットを **user2** という名前のユーザーにリンクするには、以下を実行します。

```
# radosgw-admin bucket link --uid=user2 --bucket=data
```

2. バケットが **user2** に正常にリンクされていることを確認します。

```
# radosgw-admin bucket list --uid=user2
[
  "data"
]
```

3. バケットの所有権を新規ユーザーに変更します。

```
radosgw-admin bucket chown --uid=user --bucket=bucket
```

以下を置き換えます。

- **user** を、バケットの所有権を変更するユーザーのユーザー名に
- **bucket** を、名前を持つバケットに

たとえば、**データ** バケットの所有権を **user2** に変更するには、以下を実行します。

```
# radosgw-admin bucket chown --uid=user2 --bucket=data
```

4. 次のコマンドの出力で **owner** 行を確認して、**data** バケットの所有権が正常に変更されたことを確認します。

```
# radosgw-admin bucket list --bucket=data
```

### 3.9.1.3. テナントユーザー間でのバケットの移動

バケットは、あるテナントユーザーと別のテナントユーザーの間を移動できます。

#### 手順

1. バケットを新規ユーザーにリンクします。

```
radosgw-admin bucket link --bucket=current-tenant/bucket --uid=new-tenant$user
```

置き換え:

- **current-tenant** を、バケットのテナントの名前に
- **bucket** リンクするバケットの名前に
- **new-tenant** を、新規ユーザーがあるテナントの名前に置き換えます。
- **user** を、新しいユーザーのユーザー名に

たとえば、**data** バケットを、**test** テナントから、**test2** テナントの **user2** という名前のユーザーにリンクします。

```
# radosgw-admin bucket link --bucket=test/data --uid=test2$user2
```

2. バケットが **user2** に正常にリンクされていることを確認します。

```
# radosgw-admin bucket list --uid=test$user2
[
  "data"
]
```

3. バケットの所有権を新規ユーザーに変更します。

```
radosgw-admin bucket chown --bucket=new-tenant/bucket --uid=new-tenant$user
```

以下を置き換えます。

- **bucket** リンクするバケットの名前に
- **new-tenant** を、新規ユーザーがあるテナントの名前に置き換えます。

- **user** を、新しいユーザーのユーザー名に

たとえば、**data** バケットの所有権を **test2** テナント内の **user2** に変更するには、次のようにします。

```
# radosgw-admin bucket chown --bucket='test2/data' --uid='test$user2'
```

4. 次のコマンドの出力で **owner** 行を確認して、**data** バケットの所有権が正常に変更されたことを確認します。

```
# radosgw-admin bucket list --bucket=test2/data
```

### 3.9.1.4. バケットをテナントのないユーザーからテナントユーザーに移動する

バケットをテナントのないユーザーからテナントユーザーに移動できます。

#### 手順

1. 任意です。まだ複数のテナントがない場合は、**rgw\_keystone\_implicit\_tenants** を有効にして、外部テナントから Ceph Object Gateway にアクセスすることでテナントを作成できます。Ceph 設定ファイル (デフォルトでは `/etc/ceph/ceph.conf`) を開き、編集します。**rgw\_keystone\_implicit\_tenants** オプションを有効にします。

```
rgw_keystone_implicit_tenants = true
```

**s3cmd** コマンドまたは **swift** コマンドのいずれかを使用して、一時テナントから Ceph Object Gateway にアクセスします。

```
# swift list
```

または、**s3cmd** を使用します。

```
# s3cmd ls
```

外部テナントからの最初のアクセスにより、同等の Ceph Object Gateway ユーザーが作成されます。

2. バケットをテナントされたユーザーに移動します。

```
radosgw-admin bucket link --bucket=/bucket --uid='tenant$user'
```

置き換え:

- **bucket** を、名前を持つバケットに
- **tenant** を、新規ユーザーがあるテナントの名前に
- **user** を、新しいユーザーのユーザー名に

たとえば、**data** バケットを **test** テナント内の **tenanted-user** に移動するには、以下を実行します。

```
# radosgw-admin bucket link --bucket=/data --uid='test$tenanted-user'
```

3. **data** バケットが **tenanted-user** に正常にリンクされていることを確認します。

```
# radosgw-admin bucket list --uid='test$tenanted-user'
[
  "data"
]
```

4. バケットの所有権を新規ユーザーに変更します。

```
radosgw-admin bucket chown --bucket='tenant/bucket name' --uid='tenant$user'
```

置き換え:

- **bucket** を、名前を持つバケットに
- **tenant** を、新規ユーザーがあるテナントの名前に
- **user** を、新しいユーザーのユーザー名に

たとえば、**data** バケットの所有権を、**test** テナント内にある **tenanted-user** に変更するには、以下を実行します。

```
# radosgw-admin bucket chown --bucket='test/data' --uid='test$tenanted-user'
```

5. 次のコマンドの出力で **owner** 行を確認して、**data** バケットの所有権が正常に変更されたことを確認します。

```
# radosgw-admin bucket list --bucket=test/data
```

### 3.9.2. バケットの名前変更

バケットの名前を変更できます。

#### 前提条件

- 稼働中の Red Hat Ceph Storage クラスタがある。
- Ceph Object Gateway がインストールされている。
- バケットがある。

#### 手順

1. バケットを一覧表示します。

```
radosgw-admin bucket list
```

たとえば、出力からのバケットに注意してください。

```
# radosgw-admin bucket list
[
  "34150b2e9174475db8e191c188e920f6/swcontainer",
  "s3bucket1",
]
```

```
"34150b2e9174475db8e191c188e920f6/swimpfalse",
"c278edd68cfb4705bb3e07837c7ad1a8/ec2container",
"c278edd68cfb4705bb3e07837c7ad1a8/demoten1",
"c278edd68cfb4705bb3e07837c7ad1a8/demo-ct",
"c278edd68cfb4705bb3e07837c7ad1a8/demopostup",
"34150b2e9174475db8e191c188e920f6/postimpfalse",
"c278edd68cfb4705bb3e07837c7ad1a8/demoten2",
"c278edd68cfb4705bb3e07837c7ad1a8/postupsw"
```

```
]
```

2. バケットの名前を変更します。

```
radosgw-admin bucket link --bucket=original-name --bucket-new-name=new-name --
uid=user-ID
```

たとえば、**s3bucket1** バケットの名前を **s3newb** に変更するには、以下を実行します。

```
# radosgw-admin bucket link --bucket=s3bucket1 --bucket-new-name=s3newb --uid=testuser
```

バケットがテナント内部にある場合は、テナントも指定します。

```
radosgw-admin bucket link --bucket=tenant/original-name --bucket-new-name=new-name
--uid=tenant$user-ID
```

以下に例を示します。

```
# radosgw-admin bucket link --bucket=test/s3bucket1 --bucket-new-name=s3newb --
uid=test$testuser
```

3. バケットの名前が変更されたことを確認します。

```
radosgw-admin bucket list
```

たとえば、**s3newb** という名前のバケットが存在するようになりました。

```
# radosgw-admin bucket list
[
  "34150b2e9174475db8e191c188e920f6/swcontainer",
  "34150b2e9174475db8e191c188e920f6/swimpfalse",
  "c278edd68cfb4705bb3e07837c7ad1a8/ec2container",
  "s3newb",
  "c278edd68cfb4705bb3e07837c7ad1a8/demoten1",
  "c278edd68cfb4705bb3e07837c7ad1a8/demo-ct",
  "c278edd68cfb4705bb3e07837c7ad1a8/demopostup",
  "34150b2e9174475db8e191c188e920f6/postimpfalse",
  "c278edd68cfb4705bb3e07837c7ad1a8/demoten2",
  "c278edd68cfb4705bb3e07837c7ad1a8/postupsw"
```

```
]
```

### 3.9.3. 関連情報

- [ユーザーを認証するためのキーストーンの使用](#) を参照してください。

- 詳細については、[開発者ガイド](#) を参照してください。

## 3.10. CEPH OBJECT GATEWAY のガベージコレクションの最適化

新規データオブジェクトがストレージクラスターに書き込まれると、Ceph Object Gateway はこれらの新規オブジェクトにすぐにストレージを割り当てます。ストレージクラスターのデータオブジェクトを削除または上書きした後に、Ceph Object Gateway はバケットインデックスからこれらのオブジェクトを削除します。その後しばらくして、Ceph Object Gateway は、ストレージクラスターにオブジェクトを格納するために使用されたスペースをパーズします。ストレージクラスターから削除されたオブジェクトデータをパーズするプロセスは、ガベージコレクションまたは GC として知られています。

通常、ガベージコレクションの操作はバックグラウンドで実行されます。これらの操作は、継続的に実行するように設定することも、アクティビティーが少なくワークロードが軽い期間でのみ実行するように設定することもできます。デフォルトでは、Ceph Object Gateway は GC 操作を継続的に実行します。GC 操作は Ceph Object Gateway 操作の通常の部分であるため、ガベージコレクションの対象となる削除済みオブジェクトはほとんどの場合存在します。

### 3.10.1. ガベージコレクションキューの表示

削除および上書きされたオブジェクトをストレージクラスターからパーズする前に、**radosgw-admin** を使用して、ガベージコレクションを待機しているオブジェクトを表示します。

#### 前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph Object Gateway へのルートレベルのアクセス。

#### 手順

1. ガベージコレクションを待っているオブジェクトのキューを表示するには、以下を実行します。

#### 例

```
[root@rgw ~] radosgw-admin gc list
```



#### 注記

有効期限が切れていないエントリーを含む、キュー内のすべてのエントリーを表示するには、**--include-all** オプションを使用します。

### 3.10.2. 削除が多いワークロードのガベージコレクションの調整

一部のワークロードは、一時的または永続的にガベージコレクション (GC) のアクティビティーの回数を上回る場合があります。これは、多くのオブジェクトが短期間保存されてから削除される、削除の多いワークロードに特に当てはまります。これらのタイプのワークロードでは、他の操作と比較してガベージコレクション操作の優先度を上げることが検討してください。Ceph Object Gateway ガベージコレクションに関するその他の質問については、Red Hat サポートにお問い合わせください。

#### 前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。

- ストレージクラスター内のすべてのノードへの root レベルのアクセス。

## 手順

1. `/etc/ceph/ceph.conf` を開いて編集します。
2. `rgw_gc_max_concurrent_io` の値を 20 に設定し、`rgw_gc_max_trim_chunk` の値を 64 に設定します。

```
rgw_gc_max_concurrent_io = 20
rgw_gc_max_trim_chunk = 64
```

3. ファイルを保存してから閉じます。
4. Ceph Object Gateway を再起動して、変更した設定が有効になるようにします。
5. GC アクティビティー中にストレージクラスターを監視して、値の増加がパフォーマンスに悪影響を与えないことを確認します。



### 重要

実行中のクラスターの `rgw_gc_max_objs` オプションの値を変更しないでください。この値は、RGW ノードをデプロイする前にのみ変更する必要があります。

## 関連情報

- [Ceph RGW: GC の調整オプション](#)
- [RGW 一般設定](#)
- [設定リファレンス](#)

## 第4章 設定リファレンス

次の設定は、**[client.rgw.<instance\_name>]** セクションの下の Ceph 設定ファイル (通常は **ceph.conf**) に追加できます。設定にはデフォルト値が含まれる場合があります。Ceph 設定ファイル内で各設定を指定しない場合、デフォルト値は自動的に設定されます。

**[client.rgw.<instance\_name>]** セクションで設定された設定変数は、コマンドで **instance\_name** が指定されていない **rgw** または **radosgw-admin** コマンドには適用されません。したがって、すべての Ceph Object Gateway インスタンスまたはすべての **radosgw-admin** コマンドに適用されることを意図した変数を **[global]** セクションまたは **[client]** セクションに配置して、**instance\_name** 指定を回避できます。

### 4.1. 一般設定

名前	説明	型	デフォルト
<b>rgw_data</b>	Ceph Object Gateway のデータファイルの場所を設定します。	文字列	<b>/var/lib/ceph/radosgw/\$cluster-\$id</b>
<b>rgw_enable_apis</b>	指定された API を有効にします。	文字列	<b>s3、swift、swift_auth、admin</b> すべての API
<b>rgw_cache_enabled</b>	Ceph Object Gateway キャッシュを有効にするかどうか。	ブール値	<b>true</b>
<b>rgw_cache_lru_size</b>	Ceph Object Gateway キャッシュのエントリ数。	整数	<b>10000</b>
<b>rgw_socket_path</b>	ドメインソケットのソケットパス。 <b>FastCgiExternalServer</b> はこのソケットを使用します。ソケットパスを指定しない場合、Ceph Object Gateway は外部サーバーとしては実行しません。ここで指定するパスは、 <b>rgw.conf</b> ファイルで指定されたパスと同じである必要があります。	文字列	該当なし
<b>rgw_host</b>	Ceph Object Gateway インスタンスのホスト。IP アドレスまたはホスト名を指定できます。	文字列	<b>0.0.0.0</b>
<b>rgw_port</b>	インスタンスが要求をリッスンするポート。指定されていない場合、Ceph Object Gateway は外部の FastCGI を実行します。	文字列	なし
<b>rgw_dns_name</b>	提供されるドメインの DNS 名。ゾーングループ内での <b>hostnames</b> の設定も参照してください。	文字列	なし



名前	説明	型	デフォルト
<b>rgw_script_uri</b>	リクエストで設定されていない場合は <b>SCRIPT_URI</b> の代替値。	文字列	なし
<b>rgw_request_uri</b>	リクエストで設定されていない場合は <b>REQUEST_URI</b> の代替値。	文字列	なし
<b>rgw_print_continue</b>	稼働中の場合は <b>100-continue</b> を有効にします。	ブール値	<b>true</b>
<b>rgw_remote_addr_param</b>	リモートアドレスパラメーター。たとえば、リモートアドレスを含む HTTP フィールド、またはリバースプロキシが動作している場合は <b>X-Forwarded-For</b> アドレス。	文字列	<b>REMOTE_ADDR</b>
<b>rgw_op_thread_timeout</b>	オープンスレッドのタイムアウト (秒単位)。	Integer	<b>600</b>
<b>rgw_op_thread_suicide_timeout</b>	Ceph Object Gateway プロセスが終了するまでの <b>timeout</b> 時間 (秒単位)。0 に設定すると無効です。	整数	<b>0</b>
<b>rgw_thread_pool_size</b>	スレッドプールのサイズ。この変数は、設定されている場合、 <b>num_threads</b> によって上書きされます。詳細については、 <a href="#">Civetweb 設定オプション</a> を参照してください。	Integer	<b>512</b>
<b>rgw_num_control_objects</b>	異なる <b>rgw</b> インスタンス間でキャッシュの同期に使用される通知オブジェクトの数。	整数	<b>8</b>
<b>rgw_init_timeout</b>	Ceph Object Gateway が初期化時に起動するまでの時間 (秒数)。	整数	<b>30</b>
<b>rgw_mime_types_file</b>	MIME タイプのパスおよび場所。オブジェクトタイプの Swift の自動検出に使用されます。	文字列	<b>/etc/mime.types</b>
<b>rgw_gc_max_objs</b>	1つのガベージコレクションの処理サイクルで、ガベージコレクションによって処理される可能性のあるオブジェクトの最大数。	整数	<b>32</b>
<b>rgw_gc_obj_min_wait</b>	オブジェクトが削除され、ガベージコレクション処理によって処理されるまでの最小待機時間。	整数	<b>2 * 3600</b>
<b>rgw_gc_processor_max_time</b>	2つの連続するガベージコレクション処理サイクルで、1つが開始された後に2つ目が開始されるまでの最大時間。	整数	<b>3600</b>

名前	説明	型	デフォルト
<code>rgw_gc_processor_period</code>	ガベージコレクション処理のサイクル時間。	整数	<b>3600</b>
<code>rgw_s3_success_create_obj_status</code>	<code>create-obj</code> の代替成功ステータス応答。	整数	<b>0</b>
<code>rgw_resolve_cname</code>	<code>rgw</code> が要求ホスト名フィールドの DNS CNAME レコードを使用すべきかどうか (ホスト名が <code>rgw_dns 名</code> と等しくない場合)。	ブール値	<b>false</b>
<code>rgw_object_stripe_size</code>	Ceph Object Gateway オブジェクトのオブジェクトストライプのサイズ。	整数	<b>4 &lt;&lt; 20</b>
<code>rgw_extended_http_attrs</code>	オブジェクトに設定できる属性の新規セットを追加します。これらの追加属性は、オブジェクトを配置する際に HTTP ヘッダーフィールドを介して設定できます。設定された場合、オブジェクトで GET/HEAD を実行すると、これらの属性は HTTP フィールドとして返されます。	文字列	なし。例: "content_foo、 content_bar"
<code>rgw_exit_timeout_secs</code>	無条件に終了する前にプロセスを待機する秒数。	整数	<b>120</b>
<code>rgw_get_obj_window_size</code>	単一オブジェクト要求のウィンドウサイズ (バイト単位)。	整数	<b>16 &lt;&lt; 20</b>
<code>rgw_get_obj_max_req_size</code>	Ceph Storage Cluster に送信される単一の get 操作の最大リクエストサイズ。	整数	<b>4 &lt;&lt; 20</b>
<code>rgw_relaxed_s3_bucket_names</code>	ゾーングループバケットの緩和された S3 バケット名ルールを有効にします。	ブール値	<b>false</b>
<code>rgw_list_buckets_max_chunk</code>	ユーザーバケットを一覧表示する際に、単一の操作で取得するバケットの最大数。	整数	<b>1000</b>
<code>rgw_override_bucket_index_max_shards</code>	バケットインデックスオブジェクトのシャードの数。値が <b>0</b> の場合は、シャード化がないことを示します。Red Hat では、バケットの一覧のコストを増やすため、大きすぎる値 ( <b>1000</b> など) を設定することは推奨されません。  この変数は、 <code>[client]</code> セクションまたは <code>[global]</code> セクションで設定して、 <code>radosgw-admin</code> コマンドに自動的に適用されるようにする必要があります。	整数	<b>0</b>
<code>rgw_num_zone_opstate_shards</code>	ゾーングループ間コピーの進行状況情報を保持するためのシャードの最大数。	整数	<b>128</b>

名前	説明	型	デフォルト
<b>rgw_opstate_ratelimit_sec</b>	1回のアップロードでの opstate 更新間の最小時間。0 レート制限を無効にします。	整数	<b>30</b>
<b>rgw_curl_wait_timeout_ms</b>	特定の <b>curl</b> 呼び出しのタイムアウト (ミリ秒単位)。	整数	<b>1000</b>
<b>rgw_copy_obj_progress</b>	長いコピー操作中にオブジェクトの進行状況を出力できるようにします。	ブール値	<b>true</b>
<b>rgw_copy_obj_progress_every_bytes</b>	コピー進行状況出力間の最小バイト。	整数	<b>1024 * 1024</b>
<b>rgw_admin_entry</b>	管理要求 URL のエントリーポイント。	文字列	<b>admin</b>
<b>rgw_content_length_compat</b>	CONTENT_LENGTH と HTTP_CONTENT_LENGTH セットの両方で FCGI 要求の互換性処理を有効にします。	ブール値	<b>false</b>
<b>rgw_bucket_default_quota_max_objects</b>	バケットごとのデフォルトのオブジェクトの最大数。他のクォータが指定されていない場合、この値は新規ユーザーに設定されます。既存ユーザーには影響しません。  この変数は、 <b>[client]</b> セクションまたは <b>[global]</b> セクションで設定して、 <b>radosgw-admin</b> コマンドに自動的に適用されるようにする必要があります。	整数	<b>-1</b>
<b>rgw_bucket_quota_ttl</b>	キャッシュされたクォータ情報が信頼される秒単位の時間。このタイムアウト後、クォータ情報がクラスターから再フェッチされます。	Integer	<b>600</b>
<b>rgw_user_quota_bucket_sync_interval</b>	クラスターに同期する前に、バケットクォータ情報が累積される時間 (秒単位)。この間に、他の RGW インスタンスは、このインスタンスでの操作によるバケットクォータ統計の変更を認識しません。	Integer	<b>180</b>
<b>rgw_user_quota_sync_interval</b>	クラスターに同期する前に、ユーザークォータ情報が累積される時間 (秒単位)。この間に、他の RGW インスタンスは、このインスタンスでの操作によるユーザークォータ統計の変更を認識しません。	Integer	<b>3600 * 24</b>

## 4.2. プールについて

Ceph ゾーンは、一連の Ceph Storage Cluster プールにマッピングします。

手動で作成されたプールと生成されたプールの比較

Ceph Object Gateway のユーザーキーに書き込み機能が含まれる場合、ゲートウェイにはプールを自動的に作成する機能があります。これは、使用開始に便利です。ただし、Ceph Object Storage Cluster は、Ceph 設定ファイルに設定されていない限り、配置グループのデフォルト値を使用します。また、Ceph はデフォルトの CRUSH 階層を使用します。これらの設定は、実稼働システムに適していません。

実稼働システムを設定するには、Red Hat Ceph Storage 3 の [実稼働環境での Ceph Object Gateway](#) を参照してください。ストレージ戦略は、[実稼働用 Ceph Object Gateway ガイドの ストレージ戦略の開発](#) セクションを参照してください。

Ceph Object Gateway のデフォルトゾーンのデフォルトプールには以下が含まれます。

- **.rgw.root**
- **.default.rgw.control**
- **.default.rgw.gc**
- **.default.log**
- **.default.intent-log**
- **.default.usage**
- **.default.users**
- **.default.users.email**
- **.default.users.swift**
- **.default.users.uid**

Ceph Object Gateway は、ゾーンごとにプールを作成します。プールを手動で作成する場合は、ゾーン名を先頭に追加します。システムプールには、システム制御、ガベージコレクション、ロギング、ユーザー情報、使用法などに関連するオブジェクトが格納されます。慣例により、これらのプール名には、プール名の前にゾーン名が付加されます。

- **.<zone-name>.rgw.control**: コントロールプール。
- **.<zone-name>.rgw.gc**: 削除するオブジェクトのハッシュバケットを含むガベージコレクションプール。
- **.<zone-name>.log**: ログプールには、すべてのバケット/コンテナのログおよび create、read、update、および delete などのオブジェクトアクションが含まれます。
- **.<zone-name>.intent-log**: インテントログプールには、リクエストが失敗した場合に元に戻す/やり直しを容易にするオブジェクト更新リクエストのコピーが含まれます。
- **.<zone-name>.users.uid**: ユーザー ID プールには、一意のユーザー ID のマップが含まれます。
- **.<zone-name>.users.keys**: キープールには、各ユーザー ID のアクセスキーおよびシークレットキーが含まれます。
- **.<zone-name>.users.email**: メールプールには、ユーザー ID に関連付けられたメールアドレスが含まれます。

- `.<zone-name>.users.swift`: Swift プールには、ユーザー ID の Swift サブユーザー情報が含まれます。
- `.<zone-name>.usage`: 使用状況プールには、ユーザーごとの使用状況ログが含まれています。

Ceph Object Gateways は、配置プールにバケットインデックス (`index_pool`) およびバケットデータ (`data_pool`) のデータを保存します。これらは重複する可能性があります。つまり、インデックスとデータに同じプールを使用できます。デフォルト配置のインデックスプールは `{zone-name}.rgw.buckets.index` であり、デフォルト配置のデータプールのインデックスプールは `{zone-name}.rgw.buckets` です。

名前	説明	型	デフォルト
<code>rgw_zonegroup_root_pool</code>	すべてのゾーングループ固有の情報を保存するプール。	文字列	<code>.rgw.root</code>
<code>rgw_zone_root_pool</code>	ゾーン固有の情報を保存するプール。	文字列	<code>.rgw.root</code>

### 4.3. SWIFT 設定

名前	説明	型	デフォルト
<code>rgw_enforce_swift_acl</code>	Swift Access Control List (ACL) 設定を強制します。	ブール値	<code>true</code>
<code>rgw_swift_token_expiration</code>	Swift トークンの有効期限が切れるまでの時間 (秒単位)。	整数	<code>24 * 3600</code>
<code>rgw_swift_url</code>	Ceph Object Gateway Swift API の URL。	文字列	なし
<code>rgw_swift_url_prefix</code>	Swift API の URL 接頭辞 (例: <a href="http://fqdn.com/swift">http://fqdn.com/swift</a> )。	<code>swift</code>	該当なし
<code>rgw_swift_auth_url</code>	v1 認証トークンを確認するためのデフォルト URL (内部の Swift 認証を使用しない場合)。	文字列	なし
<code>rgw_swift_auth_entry</code>	Swift 認証 URL のエントリーポイント。	文字列	<code>auth</code>

### 4.4. ログ設定

名前	説明	型	デフォルト
<code>rgw_log_nonexistent_bucket</code>	Ceph Object Gateway が、存在しないバケットの要求をログに記録できるようにします。	ブール値	<code>false</code>

名前	説明	型	デフォルト
<code>rgw_log_object_name</code>	オブジェクト名のロギング形式。フォーマット指定子の詳細は、man ページの <code>date</code> を参照してください。	Date	<code>%Y-%m-%d-%H-%i-%n</code>
<code>rgw_log_object_name_utc</code>	ログに記録されたオブジェクト名に UTC 時刻が含まれているかどうか。 <b>false</b> の場合はローカルタイムを使用します。	ブール値	<b>false</b>
<code>rgw_usage_max_shards</code>	使用状況ログのシャードの最大数。	整数	<b>32</b>
<code>rgw_usage_max_user_shards</code>	1人のユーザーの使用状況ログに使用されるシャードの最大数。	整数	<b>1</b>
<code>rgw_enable_ops_log</code>	成功した Ceph Object Gateway 操作ごとにロギングを有効にします。	ブール値	<b>false</b>
<code>rgw_enable_usage_log</code>	使用状況ログを有効にします。	ブール値	<b>false</b>
<code>rgw_ops_log_rados</code>	操作ログを Ceph Storage Cluster のバックエンドに書き込む必要があるかどうか。	ブール値	<b>true</b>
<code>rgw_ops_log_socket_path</code>	操作ログを書き込む Unix ドメインソケット。	文字列	なし
<code>rgw_ops_log_data_backlog</code>	Unix ドメインソケットに書き込まれた操作ログの最大データバックログデータサイズ。	整数	<b>5 &lt;&lt; 20</b>
<code>rgw_usage_log_flush_threshold</code>	同期的にフラッシュする前に、使用状況ログでダーティーマージされたエントリーの数。	整数	1024
<code>rgw_usage_log_tick_interval</code>	保留中の使用量のログデータを <b>n</b> 秒ごとにフラッシュします。	整数	<b>30</b>
<code>rgw_intent_log_object_name</code>	インテントログオブジェクト名のロギング形式。フォーマット指定子の詳細は、man ページの <code>date</code> を参照してください。	Date	<code>%Y-%m-%d-%i-%n</code>
<code>rgw_intent_log_object_name_utc</code>	インテントログオブジェクト名に UTC 時刻が含まれているかどうか。 <b>false</b> の場合はローカルタイムを使用します。	ブール値	<b>false</b>
<code>rgw_data_log_window</code>	データログエントリーウィンドウ (秒単位)。	整数	<b>30</b>

名前	説明	型	デフォルト
<b>rgw_data_log_changes_size</b>	データ変更ログのために保持するインメモリーエントリーの数。	整数	<b>1000</b>
<b>rgw_data_log_num_shards</b>	データ変更ログを保持するシャード (オブジェクト) の数。	整数	<b>128</b>
<b>rgw_data_log_obj_prefix</b>	データログのオブジェクト名の接頭辞。	文字列	<b>data_log</b>
<b>rgw_replica_log_obj_prefix</b>	レプリカログのオブジェクト名の接頭辞。	文字列	<b>replica log</b>
<b>rgw_md_log_max_shards</b>	メタデータログのシャードの最大数。	整数	<b>64</b>

## 4.5. KEYSTONE 設定

名前	説明	型	デフォルト
<b>rgw_keystone_url</b>	Keystone サーバーの URL。	文字列	なし
<b>rgw_keystone_admin_token</b>	Keystone 管理トークン (共有シークレット)	文字列	なし
<b>rgw_keystone_accepted_roles</b>	要求を提供するのに必要なロール。	文字列	<b>Member, admin</b>
<b>rgw_keystone_token_cache_size</b>	各 Keystone トークンキャッシュのエントリーの最大数。	整数	<b>10000</b>
<b>rgw_keystone_revocation_interval</b>	トークン失効チェックの間隔 (秒単位)。	整数	<b>15 * 60</b>

## 4.6. LDAP 設定

名前	説明	型	例
<b>rgw_ldap_uri</b>	URI 形式の LDAP サーバーのスペース区切り一覧。	文字列	<b>ldaps://&lt;ldap.your.domain&gt;</b>
<b>rgw_ldap_searchdn</b>	ベースドメインとも呼ばれる LDAP 検索ドメイン名。	文字列	<b>cn=users,cn=accounts,dc=example,dc=com</b>

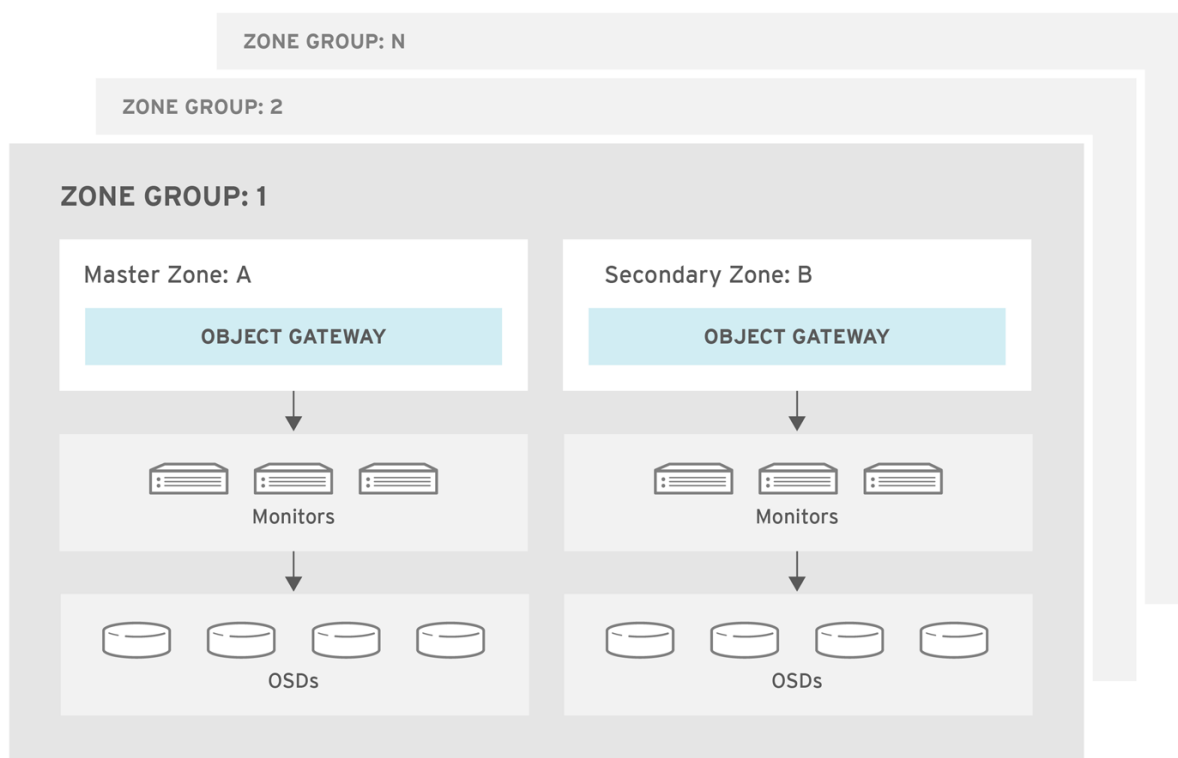
名前	説明	型	例
<b>rgw_ldap_binddn</b>	ゲートウェイはこの LDAP エントリー (ユーザー一致) でバインドします。	文字列	<b>uid=admin,cn=users,dc=example,dc=com</b>
<b>rgw_ldap_secret</b>	<b>rgw_ldap_binddn</b> の認証情報を含むファイル	文字列	<b>/etc/openldap/secret</b>
<b>rgw_ldap_dnattr</b>	Ceph Object Gateway のユーザー名を含む LDAP 属性 (binddn 形式)。	文字列	<b>uid</b>



## 第5章 マルチサイト

単一ゾーン設定は通常、1つのゾーンと1つ以上の **ceph-radosgw** インスタンスを含む1つのゾーングループで設定され、インスタンス間でゲートウェイクライアント要求の負荷を分散できます。単一ゾーン設定では、通常複数のゲートウェイインスタンスが単一の Ceph Storage Cluster を参照します。ただし、Red Hat は、Ceph Object Gateway の複数のマルチサイト設定オプションをサポートしています。

- マルチゾーン:** より高度な設定は、1つのゾーングループと複数のゾーンで設定され、各ゾーンには1つ以上の **ceph-radosgw** インスタンスがあります。各ゾーンは、独自の Ceph Storage Cluster でサポートされます。ゾーングループ内の複数のゾーンは、ゾーンの1つで重大な障害が発生した場合に、ゾーングループに障害復旧を提供します。Red Hat Ceph Storage 2 以降のリリースでは、各ゾーンがアクティブであり、書き込み操作を受け取る場合があります。障害復旧に加え、複数のアクティブなゾーンがコンテンツ配信ネットワークの基盤としても機能する場合があります。レプリケーションなしで複数のゾーンを設定するには、「[レプリケーションなしでの複数ゾーンの設定](#)」を参照してください。
- Multi-zone-group:** 以前はリージョンと呼ばれていた Ceph Object Gateway は、複数のゾーングループをサポートすることもでき、各ゾーングループには1つ以上のゾーンがあります。同じレルム内のゾーングループに保管されるオブジェクトは、グローバル名前空間を共有し、ゾーングループとゾーン全体で一意的なオブジェクト ID を確保します。
- 複数のレルム:** Red Hat Ceph Storage 2 以降のリリースでは、Ceph Object Gateway はレルムの概念をサポートします。これは、単一のゾーングループまたは複数のゾーングループであり、レルムのグローバルに一意的な名前空間です。複数のレルムは、多数の設定と名前空間をサポートする機能を提供します。



REALM: A

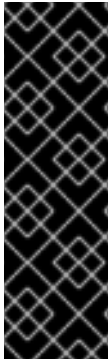
CEPH\_405148\_0616

## 5.1. 要件および前提条件

マルチサイト設定には、少なくとも 2 つの Ceph Storage Cluster が必要です。さらに、2 つ以上の Ceph Object Gateway インスタンス (Ceph Storage Cluster ごとに 1 つずつ) が必要です。

本ガイドでは、地理的に別々の場所にある Ceph Storage Cluster が 2 つ以上あることを前提としていますが、この設定は同じ物理サイトで機能することができます。本ガイドでは、**rgw1**、**rgw2**、**rgw3**、および **rgw4** という名前の 4 つの Ceph Object Gateway サーバーをそれぞれ前提としています。

マルチサイト設定では、**マスターゾーングループとマスターゾーンが必要です**。さらに、**各ゾーングループにはマスターゾーンが必要です**。ゾーングループには、1 つ以上のセカンダリーゾーンまたはマスター以外のゾーンがあります。



### 重要

レルムのマスターゾーングループ内のマスターゾーンは、(**radosgw-admin** CLI によって作成された) ユーザー、クォータ、バケットを含むレルムのメタデータのマスターコピーを保存するルールを果たします。このメタデータは、セカンダリーゾーンおよびセカンダリーゾーングループに自動的に同期されます。**radosgw-admin** CLI で実行されるメタデータ操作は、セカンダリーゾーングループおよびゾーンに確実に同期されるように、マスターゾーングループのマスターゾーン内のホストで **実行する必要があります**。現在、セカンダリーゾーンとゾーングループでメタデータ操作を実行することは **可能ですが、同期されず** ため、メタデータが断片化されるため、**推奨されません**。

次の例では、**rgw1** ホストがマスターゾーングループのマスターゾーンとして機能します。**rgw2** ホストは、マスターゾーングループのセカンダリーゾーンとして機能します。**rgw3** ホストは、セカンダリーゾーングループのマスターゾーンとして機能します。**rgw4** ホストは、セカンダリーゾーングループのセカンダリーゾーンとして機能します。

## 5.2. POOLS

Red Hat は、[Ceph 配置グループのプールごとの計算機](#) を使用して、**ceph-radosgw** デーモンが作成するプールに適した配置グループの数を計算することを推奨します。Ceph 設定ファイルで、計算値をデフォルトとして設定します。以下に例を示します。

```
osd pool default pg num = 50
osd pool default pgp num = 50
```



### 注記

ストレージクラスターの Ceph 設定ファイルにこの変更を行います。その後、ゲートウェイインスタンスがプールを作成する際にそれらのデフォルトを使用するように、設定にランタイムの変更を加えます。

または、プールを手動で作成します。プールの作成に関する詳細は、[ストラテジー戦略ガイドのプール](#) の章を参照してください。

ゾーンに固有のプール名は、命名規則 **{zone-name}.pool-name** に従います。たとえば、**us-east** という名前のゾーンには以下のプールがあります。

- **.rgw.root**
- **us-east.rgw.control**

- `us-east.rgw.data.root`
- `us-east.rgw.gc`
- `us-east.rgw.log`
- `us-east.rgw.intent-log`
- `us-east.rgw.usage`
- `us-east.rgw.users.keys`
- `us-east.rgw.users.email`
- `us-east.rgw.users.swift`
- `us-east.rgw.users.uid`
- `us-east.rgw.buckets.index`
- `us-east.rgw.buckets.data`

### 5.3. OBJECT GATEWAY のインストール

Ceph Object Gateway のインストール方法の詳細は、Red Hat Ceph Storage 3 [Installation Guide for Ubuntu](#) を参照してください。

すべての Ceph Object Gateway ノードは、**Red Hat Ceph Storage のインストール要件**セクションにリストされているタスクに従う必要があります。

Ansible は、Ceph Storage Cluster で使用する Ceph Object Gateway をインストールおよび設定することができます。マルチサイトおよびマルチサイトグループのデプロイメントの場合、ゾーンごとに Ansible 設定が必要です。

Ansible で Ceph Object Gateway をインストールする場合、Ansible Playbook が初期設定を処理します。Ansible を使用して Ceph Object Gateway をインストールするには、ホストを `/etc/ansible/hosts` ファイルに追加します。**[rgws]** セクションの下に Ceph Object Gateway ホストを追加して、Ansible に対するそのロールを識別します。ホストに連続する命名がある場合は、以下のように範囲を使用します。以下に例を示します。

```
[rgws]
<rgw-host-name-1>
<rgw-host-name-2>
<rgw-host-name[3..10]>
```

ホストを追加したら、Ansible Playbook を再実行できます。



#### 注記

Ansible は、ゲートウェイが実行していることを確認するため、デフォルトのゾーンとプールは手動で削除する必要がある場合があります。本ガイドでは、これらの手順を説明します。

非同期更新で既存のマルチサイトクラスターを更新する場合は、更新のインストール指示に従います。次に、ゲートウェイインスタンスを再起動します。



## 注記

規定されているインスタンスの再起動の順番はありません。Red Hat は、最初にマスターゾーングループおよびマスターゾーンを再起動し、次にセカンダリーゾーングループとセカンダリーゾーンを再起動することを推奨しています。

## 5.4. マルチサイトレルムの確立

クラスター内のすべてのゲートウェイには設定があります。マルチサイトレルムでは、このようなゲートウェイが異なるゾーングループおよびゾーンに存在する可能性があります。それでも、レルム内で連携する必要があります。マルチサイトレルムでは、すべてのゲートウェイインスタンスは、マスターゾーングループおよびマスターゾーン内のホスト上の **ceph-radosgw** デーモンから設定を取得する **必要があります**。

したがって、マルチサイトクラスター作成の最初の手順では、レルム、マスターゾーングループ、およびマスターゾーンを確立します。マルチサイト設定でゲートウェイを設定するには、レルム設定、マスターゾーングループ、およびマスターゾーンを保持する **ceph-radosgw** インスタンスを選択します。

### 5.4.1. レルムの作成

レルムには、ゾーングループとゾーンのマルチサイト設定が含まれ、レルム内でグローバルに一意の名前空間を適用するルールも果たします。

マスターゾーングループおよびゾーンで提供できるように識別されたホストでコマンドラインインターフェイスを開いて、マルチサイト設定用に新しいレルムを作成します。次に、以下のコマンドを実行します。

```
[root@master-zone]# radosgw-admin realm create --rgw-realm={realm-name} [--default]
```

以下に例を示します。

```
[root@master-zone]# radosgw-admin realm create --rgw-realm=movies --default
```

クラスターに単一のレルムがある場合は、**--default** フラグを指定します。**--default** が指定されている場合、**radosgw-admin** はデフォルトでこのレルムを使用します。**--default** が指定されていない場合に、ローングループおよびゾーンを追加するには、**--rgw-realm** フラグまたは **--realm-id** フラグのいずれかを指定して、ゾーングループおよびゾーンを追加するときにレルムを識別する必要があります。

レルムの作成後、**radosgw-admin** はレルム設定を返します。以下に例を示します。

```
{
  "id": "0956b174-fe14-4f97-8b50-bb7ec5e1cf62",
  "name": "movies",
  "current_period": "1950b710-3e63-4c41-a19e-46a715000980",
  "epoch": 1
}
```



## 注記

Ceph はレルムに一意的 ID を生成します。これにより、必要に応じてレルムの名前を変更することができます。

### 5.4.2. マスターゾーングループの作成

レルムには、レルムのマスターゾーングループとして機能するゾーングループが少なくとも1つ必要です。

マスターゾーングループおよびゾーンで提供するように識別されたホストでコマンドラインインターフェイスを開いて、マルチサイト設定用に新しいマスターゾーングループを作成します。次に、以下のコマンドを実行します。

```
[root@master-zone]# radosgw-admin zonegroup create --rgw-zonegroup={name} --endpoints={url} [--rgw-realm={realm-name}]--realm-id={realm-id}] --master --default
```

以下に例を示します。

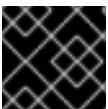
```
[root@master-zone]# radosgw-admin zonegroup create --rgw-zonegroup=us --endpoints=http://rgw1:80 --rgw-realm=movies --master --default
```

レルムにゾーングループが1つしかない場合は、**--default** フラグを指定します。**--default** が指定されている場合、**radosgw-admin** はデフォルトでこのゾーングループを使用します。**--default** が指定されていない場合に、ゾーンを追加または変更するときにゾーングループを識別するには、**--rgw-zonegroup** フラグまたは **--zonegroup-id** フラグのいずれかが必要になります。

マスターゾーングループの作成後、**radosgw-admin** はゾーングループの設定を返します。以下に例を示します。

```
{
  "id": "f1a233f5-c354-4107-b36c-df66126475a6",
  "name": "us",
  "api_name": "us",
  "is_master": "true",
  "endpoints": [
    "http://rgw1:80"
  ],
  "hostnames": [],
  "hostnames_s3webzone": [],
  "master_zone": "",
  "zones": [],
  "placement_targets": [],
  "default_placement": "",
  "realm_id": "0956b174-fe14-4f97-8b50-bb7ec5e1cf62"
}
```

### 5.4.3. マスターゾーンの作成



#### 重要

ゾーン内の Ceph Object Gateway ノードでゾーンを作成する必要があります。

マスターゾーングループおよびゾーンで提供するように識別されたホストでコマンドラインインターフェイスを開いて、マルチサイト設定用のマスターゾーンを作成します。次に、以下のコマンドを実行します。

```
[root@master-zone]# radosgw-admin zone create
--rgw-zonegroup={zone-group-name} \
--rgw-zone={zone-name} \
```

```
--master --default \  
--endpoints={http://fqdn:port}[,{http://fqdn:port}]
```

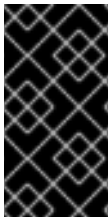
以下に例を示します。

```
[root@master-zone]# radosgw-admin zone create --rgw-zonegroup=us \  
--rgw-zone=us-east \  
--master --default \  
--endpoints={http://fqdn:port}[,{http://fqdn:port}]
```



### 注記

**--access-key** および **--secret** を指定しません。これらの設定は、次のセクションでユーザーが作成されると、ゾーンに追加されます。



### 重要

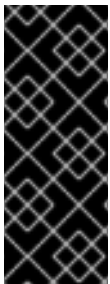
次の手順は、まだデータを保存していない、新しくインストールされたシステムを使用したマルチサイト設定を想定しています。**default** ゾーンとそのプールをすでにデータの保存に使用している場合は、削除しないでください。削除すると、データが削除されて回復できなくなります。

#### 5.4.4. デフォルトのゾングループおよびゾーンの削除

**default** ゾーンが存在する場合は削除します。最初にデフォルトのゾングループから削除してください。

```
[root@master-zone]# radosgw-admin zonegroup remove --rgw-zonegroup=default --rgw-  
zone=default  
[root@master-zone]# radosgw-admin period update --commit  
[root@master-zone]# radosgw-admin zone delete --rgw-zone=default  
[root@master-zone]# radosgw-admin period update --commit  
[root@master-zone]# radosgw-admin zonegroup delete --rgw-zonegroup=default  
[root@master-zone]# radosgw-admin period update --commit
```

最後に、Ceph Storage Cluster の **default** プールが存在する場合は削除します。



### 重要

以下の手順は、現在データを保存していない、新しくインストールされたシステムを使用するマルチサイト設定を想定しています。データを保存するために **default** ゾングループを使用している場合は、このゾングループを削除しないでください。

**default** ゾーンおよびゾングループの古いデータにアクセスするには、**radosgw-admin** コマンドで **--rgw-zone default** および **--rgw-zonegroup default** を使用します。

```
# rados rmpool default.rgw.control default.rgw.control --yes-i-really-really-mean-it  
# rados rmpool default.rgw.data.root default.rgw.data.root --yes-i-really-really-mean-it  
# rados rmpool default.rgw.gc default.rgw.gc --yes-i-really-really-mean-it  
# rados rmpool default.rgw.log default.rgw.log --yes-i-really-really-mean-it  
# rados rmpool default.rgw.users.uid default.rgw.users.uid --yes-i-really-really-mean-it
```

### 5.4.5. システムユーザーの作成

**ceph-radosgw** デーモンは、レームおよび期間情報をプルする前に認証する必要があります。マスターゾーンで、システムユーザーを作成し、デーモン間の認証を容易にします。

```
[root@master-zone]# radosgw-admin user create --uid="{user-name}" --display-name="{Display Name}" --system
```

以下に例を示します。

```
[root@master-zone]# radosgw-admin user create --uid="synchronization-user" --display-name="Synchronization User" --system
```

セカンダリーゾーンではマスターゾーンでの認証が必要になるため、**access\_key** と **secret\_key** をメモしておきます。

最後に、システムユーザーをマスターゾーンに追加します。

```
[root@master-zone]# radosgw-admin zone modify --rgw-zone=us-east --access-key={access-key} --secret={secret}
[root@master-zone]# radosgw-admin period update --commit
```

### 5.4.6. 期間の更新

マスターゾーン設定の更新後に、期間を更新します。

```
# radosgw-admin period update --commit
```



#### 注記

期間を更新するとエポックが変更され、他のゾーンが更新された設定を確実に受信できるようになります。

### 5.4.7. Ceph 設定ファイルを更新します。

**rgw\_zone** 設定オプションとマスターゾーンの名前をインスタンスエントリーに追加して、マスターゾーンホスト上の Ceph 設定ファイルを更新します。

```
[client.rgw.{instance-name}]
...
rgw_zone={zone-name}
```

以下に例を示します。

```
[client.rgw.rgw1]
host = rgw1
rgw frontends = "civetweb port=80"
rgw_zone=us-east
```

### 5.4.8. ゲートウェイの開始

Object Gateway ホストで、Ceph Object Gateway サービスを開始して有効にします。

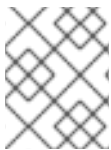
```
$ sudo systemctl start ceph-radosgw@rgw.`hostname` -s`
$ sudo systemctl enable ceph-radosgw@rgw.`hostname` -s`
```

サービスがすでに実行中の場合は、サービスを開始して有効にするのではなく、サービスを再起動します。

```
$ sudo systemctl restart ceph-radosgw@rgw.`hostname` -s`
```

## 5.5. セカンダリーゾーンの確立

ゾーングループ内のゾーンは、すべてのデータを複製して、各ゾーンが同じデータを持つようにします。セカンダリーゾーンを作成するときは、セカンダリーゾーンにサービスを提供するように識別されたホストで **すべての radosgw-admin zone** 操作を実行します。



### 注記

ゾーンを追加するには、セカンダリーゾーンを追加する手順と同じ手順に従います。別のゾーン名を使用します。



### 重要

マスターゾーングループのマスターゾーン内のホストで、ユーザーの作成やクォータなどのメタデータ操作を実行する必要があります。マスターゾーンおよびセカンダリーゾーンは、RESTful API からバケット操作を受信できますが、セカンダリーゾーンはバケット操作をマスターゾーンにリダイレクトします。マスターゾーンがダウンしている場合、バケット操作は失敗します。**radosgw-admin** CLI を使用してバケットを作成する場合は、マスターゾーングループのマスターゾーン内のホストでバケットを実行する必要があります。そうしないと、バケットは他のゾーングループおよびゾーンに同期されません。

### 5.5.1. レルムのプル

マスターゾーングループ内のマスターゾーンの URL パス、アクセスキー、およびシークレットを使用して、レルムをホストにプルします。デフォルト以外のレルムをプルするには、**--rgw-realm** 設定または **--realm-id** 設定オプションを使用してレルムを指定します。

```
# radosgw-admin realm pull --url={url-to-master-zone-gateway} --access-key={access-key} --secret={secret}
```

このレルムがデフォルトのレルムまたは唯一のレルムである場合は、そのレルムをデフォルトのレルムにします。

```
# radosgw-admin realm default --rgw-realm={realm-name}
```

### 5.5.2. 期間のプル

マスターゾーングループ内のマスターゾーンの URL パス、アクセスキー、およびシークレットを使用して、期間をホストにプルします。デフォルト以外のレルムから期間をプルするには、**--rgw-realm** または **--realm-id** 設定オプションを使用してレルムを指定します。

```
# radosgw-admin period pull --url={url-to-master-zone-gateway} --access-key={access-key} --secret={secret}
```





### 注記

期間をプルすると、レルムのゾーングループとゾーン設定の最新バージョンが取得されます。

## 5.5.3. セカンダリーゾーンの作成



### 重要

ゾーン内の Ceph Object Gateway ノードでゾーンを作成する必要があります。

セカンダリーゾーンを提供するために識別されたホストでコマンドラインインターフェイスを開いて、マルチサイト設定用のセカンダリーゾーンを作成します。ゾーングループ ID、新しいゾーン名、およびゾーンのエンドポイントを指定します。 **--master** フラグまたは **--default** フラグを使用 **しないでください**。Red Hat Ceph Storage 2 では、すべてのゾーンがデフォルトでアクティブ/アクティブ設定で実行されます。つまり、ゲートウェイクライアントは任意のゾーンにデータを書き込むことができ、そのゾーンはそのデータをゾーングループ内の他のすべてのゾーンに複製します。セカンダリーゾーンが書き込み操作を受け入れない場合は、 **--read-only** フラグを指定して、マスターゾーンとセカンダリーゾーンの間でアクティブ-パッシブ設定を作成します。さらに、マスターゾーングループのマスターゾーンに格納されている、生成されたシステムユーザーの **access\_key** および **secret\_key** を指定します。以下のコマンドを実行します。

```
[root@second-zone]# radosgw-admin zone create \
    --rgw-zonegroup={zone-group-name} \
    --rgw-zone={zone-name} --endpoints={url} \
    --access-key={system-key} --secret={secret} \
    --endpoints=http://{fqdn}:80 \
    [--read-only]
```

以下に例を示します。

```
[root@second-zone]# radosgw-admin zone create --rgw-zonegroup=us \
    --rgw-zone=us-west \
    --access-key={system-key} --secret={secret} \
    --endpoints=http://rgw2:80
```



### 重要

以下の手順は、データを保存していない新たにインストールしたシステムを使用するマルチサイト設定を想定しています。 **default** ゾーンとそのプールをすでにデータの保存に使用している場合は、 **削除しないでください**。削除すると、データが失われ、回復できなくなります。

必要に応じてデフォルトゾーンを削除します。

```
[root@second-zone]# radosgw-admin zone delete --rgw-zone=default
```

最後に、必要に応じて Ceph Storage Cluster のデフォルトプールを削除します。

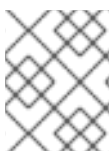
```
# rados rmpool default.rgw.control default.rgw.control --yes-i-really-really-mean-it
# rados rmpool default.rgw.data.root default.rgw.data.root --yes-i-really-really-mean-it
```

```
# rados rmpool default.rgw.gc default.rgw.gc --yes-i-really-really-mean-it
# rados rmpool default.rgw.log default.rgw.log --yes-i-really-really-mean-it
# rados rmpool default.rgw.users.uid default.rgw.users.uid --yes-i-really-really-mean-it
```

#### 5.5.4. 期間の更新

マスターゾーン設定の更新後に、期間を更新します。

```
# radosgw-admin period update --commit
```



#### 注記

期間を更新するとエポックが変更され、他のゾーンが更新された設定を確実に受信できるようになります。

#### 5.5.5. Ceph 設定ファイルを更新します。

インスタンスエントリーに **rgw\_zone** 設定オプションとセカンダリーゾーンの名前を追加して、セカンダリーゾーンホストの Ceph 設定ファイルを更新します。

```
[client.rgw.{instance-name}]
...
rgw_zone={zone-name}
```

以下に例を示します。

```
[client.rgw.rgw2]
host = rgw2
rgw frontends = "civetweb port=80"
rgw_zone=us-west
```

#### 5.5.6. ゲートウェイの開始

Object Gateway ホストで、Ceph Object Gateway サービスを開始して有効にします。

```
$ sudo systemctl start ceph-radosgw@rgw.`hostname` -s`
$ sudo systemctl enable ceph-radosgw@rgw.`hostname` -s`
```

サービスがすでに実行中の場合は、サービスを開始して有効にするのではなく、サービスを再起動します。

```
$ sudo systemctl restart ceph-radosgw@rgw.`hostname` -s`
```

### 5.6. フェイルオーバーおよび障害復旧

マスターゾーンに障害が発生した場合は、障害復旧のためにセカンダリーゾーンにフェイルオーバーします。

1. セカンダリーゾーンをマスターおよびデフォルトゾーンにします。以下に例を示します。

```
# radosgw-admin zone modify --rgw-zone={zone-name} --master --default
```

デフォルトでは、Ceph Object Gateway はアクティブ/アクティブ設定で実行されます。クラスターが active-passive 設定で実行されるように設定されている場合、セカンダリーゾーンは読み取り専用ゾーンになります。ゾーンが書き込み操作を受け取れるように **--read-only** ステータスを削除します。以下に例を示します。

```
# radosgw-admin zone modify --rgw-zone={zone-name} --master --default
```

2. 期間を更新して、変更を反映します。

```
# radosgw-admin period update --commit
```

3. 最後に、Ceph Object Gateway を再起動します。

```
$ sudo systemctl restart ceph-radosgw@rgw.`hostname -s`
```

以前のマスターゾーンが復旧する場合は、操作を元に戻す。

1. 復元されたゾーンから、現在のマスターゾーンからレームをプルします。

```
# radosgw-admin realm pull --url={url-to-master-zone-gateway} \  
--access-key={access-key} --secret={secret}
```

2. 復旧したゾーンをマスターおよびデフォルトゾーンにします。

```
# radosgw-admin zone modify --rgw-zone={zone-name} --master --default
```

3. 期間を更新して、変更を反映します。

```
# radosgw-admin period update --commit
```

4. 次に、復旧されたゾーンで Ceph Object Gateway を再起動します。

```
$ sudo systemctl restart ceph-radosgw@rgw.`hostname -s`
```

5. セカンダリーゾーンを読み取り専用設定を使用する必要がある場合は、セカンダリーゾーンを更新します。

```
# radosgw-admin zone modify --rgw-zone={zone-name} --read-only
```

6. 期間を更新して、変更を反映します。

```
# radosgw-admin period update --commit
```

7. 最後に、セカンダリーゾーンで Ceph Object Gateway を再起動します。

```
$ sudo systemctl restart ceph-radosgw@rgw.`hostname -s`
```

## 5.7. シングルサイトシステムからマルチサイトへの移行

**default** ゾーングループとゾーンを持つシングルサイトシステムからマルチサイトシステムに移行するには、次の手順を使用します。

1. レルムを作成します。<name> を、レルム名に置き換えます。

```
[root@master-zone]# radosgw-admin realm create --rgw-realm=<name> --default
```

2. デフォルトゾーンとゾーングループの名前を変更します。<name> を、ゾーングループまたはゾーン名に置き換えます。

```
[root@master-zone]# radosgw-admin zonegroup rename --rgw-zonegroup default --
zonegroup-new-name=<name>
[root@master-zone]# radosgw-admin zone rename --rgw-zone default --zone-new-name us-
east-1 --rgw-zonegroup=<name>
```

3. マスターゾーングループを設定します。<name> を、レルムまたはゾーングループ名に置き換えます。<fqdn> を、ゾーングループの完全修飾ドメイン名に置き換えます。

```
[root@master-zone]# radosgw-admin zonegroup modify --rgw-realm=<name> --rgw-
zonegroup=<name> --endpoints http://<fqdn>:80 --master --default
```

4. マスターゾーンを設定します。<name> を、レルム、ゾーングループ、またはゾーン名に置き換えます。<fqdn> を、ゾーングループの完全修飾ドメイン名に置き換えます。

```
[root@master-zone]# radosgw-admin zone modify --rgw-realm=<name> --rgw-zonegroup=
<name> \
    --rgw-zone=<name> --endpoints http://<fqdn>:80 \
    --access-key=<access-key> --secret=<secret-key> \
    --master --default
```

5. システムユーザーを作成します。<user-id> を、ユーザー名に置き換えます。<display-name> を、表示名に置き換えます。スペースが含まれる場合があります。

```
[root@master-zone]# radosgw-admin user create --uid=<user-id> \
    --display-name="<display-name>" \
    --access-key=<access-key> --secret=<secret-key> \ --system
```

6. 更新された設定をコミットします。

```
# radosgw-admin period update --commit
```

7. 最後に、Ceph Object Gateway を再起動します。

```
$ sudo systemctl restart ceph-radosgw@rgw.`hostname -s`
```

この手順を完了したら、[セカンダリーゾーンの確立](#)に進み、マスターゾーングループにセカンダリーゾーンを作成します。

## 5.8. マルチサイトのコマンドラインの使用方法

### 5.8.1. レルム

レルムは、1つ以上のゾーンが含まれる1つ以上のゾーングループと、バケットが含まれるゾーンで設定されるグローバル固有の名前空間を表します。この名前空間にはオブジェクトが含まれます。レルムにより、Ceph Object Gateway は同じハードウェアで複数の名前空間および設定をサポートできるよう

になります。

レルムには期間の概念が含まれます。それぞれの期間は、ゾーングループとゾーン設定の状態を時間で表しています。ゾーングループまたはゾーンに変更を加えるたびに、期間を更新してコミットします。

デフォルトでは、Ceph Object Gateway バージョン 2 は、バージョン 1.3 以前のリリースとの後方互換性のためのレルムを作成しません。ただし、ベストプラクティスとして、Red Hat は新規クラスタのレルムを作成することを推奨します。

### 5.8.1.1. レルムの作成

レルムを作成するには、**realm create** を実行してレルム名を指定します。レルムがデフォルトの場合は、**--default** を指定します。

```
[root@master-zone]# radosgw-admin realm create --rgw-realm={realm-name} [--default]
```

以下に例を示します。

```
[root@master-zone]# radosgw-admin realm create --rgw-realm=movies --default
```

**--default** を指定すると、**--rgw-realm** とレルム名が明示的に指定されていない限り、各 **radosgw-admin** 呼び出しでレルムが暗黙的に呼び出されます。

### 5.8.1.2. レルムのデフォルトの設定

レルム一覧にある1つのレルムはデフォルトのレルムである必要があります。デフォルトレルムは1つのみです。レルムが1つだけあり、そのレルムが作成時にデフォルトレルムとして指定されていない場合は、デフォルトのレルムにします。または、デフォルトであるレルムを変更するには、以下のコマンドを実行します。

```
[root@master-zone]# radosgw-admin realm default --rgw-realm=movies
```



#### 注記

レルムがデフォルトの場合、コマンドラインでは **--rgw-realm=<realm-name>** を引数と想定します。

### 5.8.1.3. レルムの削除

レルムを削除するには、**realm delete** を実行し、レルム名を指定します。

```
[root@master-zone]# radosgw-admin realm delete --rgw-realm={realm-name}
```

以下に例を示します。

```
[root@master-zone]# radosgw-admin realm delete --rgw-realm=movies
```

### 5.8.1.4. レルムの取得

レルムを取得するには、**realm get** を実行してレルム名を指定します。

```
# radosgw-admin realm get --rgw-realm=<name>
```

以下に例を示します。

```
# radosgw-admin realm get --rgw-realm=movies [> filename.json]
```

CLI は、レルムプロパティを使用して JSON オブジェクトを echo します。

```
{
  "id": "0a68d52e-a19c-4e8e-b012-a8f831cb3ebc",
  "name": "movies",
  "current_period": "b0c5bbef-4337-4edd-8184-5aeab2ec413b",
  "epoch": 1
}
```

> と出力ファイル名を使用して、JSON オブジェクトをファイルに出力します。

### 5.8.1.5. レルムの設定

レルムを設定するには、**realm set** を実行し、レルム名を指定し、**--infile=** を入力ファイル名で指定します。

```
[root@master-zone]# radosgw-admin realm set --rgw-realm=<name> --infile=<infilename>
```

以下に例を示します。

```
[root@master-zone]# radosgw-admin realm set --rgw-realm=movies --infile=filename.json
```

### 5.8.1.6. レルムの一覧表示

レルムを一覧表示するには、**realm list** を実行します。

```
# radosgw-admin realm list
```

### 5.8.1.7. レルム期間の一覧表示

レルムの期間を一覧表示するには、**realm list-periods** を実行します。

```
# radosgw-admin realm list-periods
```

### 5.8.1.8. レルムのプル

マスターゾーングループとマスターゾーンを含むノードからセカンダリーゾーングループまたはゾーンを含むノードにレルムをプルするには、レルム設定を受け取るノードで **realm pull** を実行します。

```
# radosgw-admin realm pull --url={url-to-master-zone-gateway} --access-key={access-key} --secret={secret}
```

### 5.8.1.9. レルムの名前変更

レルムは期間の一部ではありません。そのため、レルムの名前変更はローカルでのみ適用され、**realm pull** でプルされません。複数のゾーンを持つレルムの名前を変更する場合は、**各ゾーンでこのコマンドを実行します**。レルムの名前を変更するには、以下のコマンドを実行します。

```
# radosgw-admin realm rename --rgw-realm=<current-name> --realm-new-name=<new-realm-name>
```



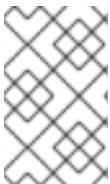
### 注記

**realm set** を使用して **name** パラメーターを変更しないでください。これにより、内部名のみが変更されます。**--rgw-realm** を指定すると、古いレルム名が使用されます。

## 5.8.2. ゾーングループ

Ceph Object Gateway は、ゾーングループの概念を使用したマルチサイトデプロイメントおよびグローバル名前空間をサポートします。以前は Red Hat Ceph Storage 1.3 でリージョンと呼ばれていたゾーングループは、1つ以上のゾーン内の1つ以上の Ceph Object Gateway インスタンスの地理的な場所を定義します。

ゾーングループの設定は、設定のすべてが Ceph 設定ファイルになるわけではないため、一般的な設定手順とは異なります。ゾーングループの一覧を表示し、ゾーングループ設定を取得し、ゾーングループ設定を設定できます。



### 注記

**radosgw-admin zonegroup** 操作は、レルム内の任意のホストで実行できます。これは、期間を更新するステップによって変更がクラスター全体に伝播されるためです。ただし、**radosgw-admin zone** 操作は、ゾーン内のホストで実行する **必要があります**。

### 5.8.2.1. ゾーングループの作成

ゾーングループの作成は、ゾーングループ名の指定から始まります。ゾーンの作成では、**--rgw-realm=<realm-name>** が指定されていない限り、デフォルトのレルムで実行されていることを前提としています。ゾーングループがデフォルトのゾーングループの場合は、**--default** フラグを指定します。ゾーングループがマスターゾーングループの場合は、**--master** フラグを指定します。以下に例を示します。

```
# radosgw-admin zonegroup create --rgw-zonegroup=<name> [--rgw-realm=<name>][--master] [--default]
```



### 注記

**zonegroup modify --rgw-zonegroup=<zonegroup-name>** を使用して、既存のゾーングループの設定を変更します。

### 5.8.2.2. ゾーングループをデフォルトにする

ゾーングループ一覧内の1つのゾーングループは、デフォルトのゾーングループである必要があります。デフォルトのゾーングループは1つのみです。ゾーングループが1つだけあり、そのゾーンは作成時にデフォルトのゾーングループとして指定されていない場合は、デフォルトのゾーングループにします。デフォルトであるゾーングループを変更する場合は、次のコマンドを実行します。

```
# radosgw-admin zonegroup default --rgw-zonegroup=comedy
```



## 注記

ゾーングループがデフォルトの場合、コマンドラインは `--rgw-zonegroup=<zonegroup-name>` を引数として想定します。

次に、期間を更新します。

```
# radosgw-admin period update --commit
```

### 5.8.2.3. ゾーングループへのゾーンの追加

ゾーングループにゾーンを追加するには、ゾーンに追加するホストでこの手順を実行する**必要があります**。ゾーンをゾーングループに追加するには、次のコマンドを実行します。

```
# radosgw-admin zonegroup add --rgw-zonegroup=<name> --rgw-zone=<name>
```

次に、期間を更新します。

```
# radosgw-admin period update --commit
```

### 5.8.2.4. ゾーングループからのゾーンの削除

ゾーングループからゾーンを削除するには、次のコマンドを実行します。

```
# radosgw-admin zonegroup remove --rgw-zonegroup=<name> --rgw-zone=<name>
```

次に、期間を更新します。

```
# radosgw-admin period update --commit
```

### 5.8.2.5. ゾーングループの名前変更

ゾーングループの名前を変更するには、次のコマンドを実行します。

```
# radosgw-admin zonegroup rename --rgw-zonegroup=<name> --zonegroup-new-name=<name>
```

次に、期間を更新します。

```
# radosgw-admin period update --commit
```

### 5.8.2.6. ゾーングループの削除

ゾーングループを削除するには、次のコマンドを実行します。

```
# radosgw-admin zonegroup delete --rgw-zonegroup=<name>
```

次に、期間を更新します。

```
# radosgw-admin period update --commit
```



### 5.8.2.7. ゾーングループの一覧表示

Ceph クラスタには、ゾーングループの一覧が含まれます。ゾーングループを一覧表示するには、次のコマンドを実行します。

```
# radosgw-admin zonegroup list
```

**radosgw-admin** は、JSON 形式のゾーングループの一覧を返します。

```
{
  "default_info": "90b28698-e7c3-462c-a42d-4aa780d24eda",
  "zonegroups": [
    "us"
  ]
}
```

### 5.8.2.8. ゾーングループの取得

ゾーングループの設定を表示するには、次のコマンドを実行します。

```
# radosgw-admin zonegroup get [--rgw-zonegroup=<zonegroup>]
```

ゾーングループの設定は以下のようになります。

```
{
  "id": "90b28698-e7c3-462c-a42d-4aa780d24eda",
  "name": "us",
  "api_name": "us",
  "is_master": "true",
  "endpoints": [
    "http://rgw1:80"
  ],
  "hostnames": [],
  "hostnames_s3website": [],
  "master_zone": "9248cab2-afe7-43d8-a661-a40bf316665e",
  "zones": [
    {
      "id": "9248cab2-afe7-43d8-a661-a40bf316665e",
      "name": "us-east",
      "endpoints": [
        "http://rgw1"
      ],
      "log_meta": "true",
      "log_data": "true",
      "bucket_index_max_shards": 0,
      "read_only": "false"
    },
    {
      "id": "d1024e59-7d28-49d1-8222-af101965a939",
      "name": "us-west",
      "endpoints": [
        "http://rgw2:80"
      ],
      "log_meta": "false",

```

```

    "log_data": "true",
    "bucket_index_max_shards": 0,
    "read_only": "false"
  }
],
"placement_targets": [
  {
    "name": "default-placement",
    "tags": []
  }
],
"default_placement": "default-placement",
"realm_id": "ae031368-8715-4e27-9a99-0c9468852cfe"
}

```

### 5.8.2.9. ゾーングループの設定

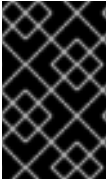
ゾーングループの定義は、少なくとも必要な設定を指定して JSON オブジェクトの作成で設定されます。

1. **name**: ゾーングループの名前。必須。
2. **api\_name**: ゾーングループの API 名。任意です。
3. **is\_master**: ゾーングループがマスターゾーングループであるかどうかを指定します。必須。注記: マスターゾーングループを1つだけ指定できます。
4. **endpoints**: ゾーングループ内のエンドポイントの一覧。たとえば、複数のドメイン名を使用して、同じゾーングループを参照できます。忘れずに前方スラッシュ (\) エスケープしてください。各エンドポイントにポート (**fqdn:port**) を指定することもできます。任意です。
5. **hostnames**: ゾーングループのホスト名の一覧。たとえば、複数のドメイン名を使用して、同じゾーングループを参照できます。任意です。**rgw dns name** 設定は、このリストに自動的に含まれます。この設定を変更したら、ゲートウェイデーモンを再起動する必要があります。
6. **master\_zone**: ゾーングループのマスターゾーン。任意です。指定がない場合は、デフォルトゾーンを使用します。注記: ゾーングループごとにマスターゾーンを1つだけ指定できます。
7. **zones**: ゾーングループ内のゾーンの一覧。各ゾーンには、名前(必須)、エンドポイントの一覧(任意)、およびゲートウェイがメタデータおよびデータ操作をログに記録するかどうか(デフォルトでは false)があります。
8. **placement\_targets**: 配置ターゲットの一覧(任意)。各配置ターゲットには、配置ターゲットの名前(必須)とタグのリスト(任意)が含まれているため、タグを持つユーザーのみが配置ターゲットを使用できます(つまり、ユーザー情報のユーザーの **placement\_tags** フィールド)。
9. **default\_placement**: オブジェクトインデックスおよびオブジェクトデータのデフォルトの配置ターゲット。デフォルトでは **default-placement** に設定されます。また、ユーザーごとのデフォルトの配置を、ユーザー情報で設定することもできます。

ゾーングループを設定するには、必須フィールドで設定される JSON オブジェクトを作成し、オブジェクトをファイル(たとえば、**zonegroup.json**)に保存します。次に、次のコマンドを実行します。

```
# radosgw-admin zonegroup set --infile zonegroup.json
```

ここで、**zonegroup.json** は作成した JSON ファイルです。



## 重要

**default** ゾーングループの **is\_master** 設定は **true** です。新しいゾーングループを作成してそれをマスターゾーングループにしたい場合は、**default** ゾーングループ **is\_master** 設定を **false** に設定するか、**default** ゾーングループを削除する必要があります。

最後に、期間を更新します。

```
# radosgw-admin period update --commit
```

### 5.8.2.10. ゾーングループマップの設定

ゾーングループマップの設定は、1つ以上のゾーングループで設定される JSON オブジェクトの作成と、クラスタの **master\_zonegroup** の設定で設定されます。ゾーングループマップの各ゾーングループは、キーと値のペアで設定されます。**key** 設定は、個々のゾーングループ設定の **名前** 設定と同等であり、**val** は、個々のゾーングループ設定で設定される JSON オブジェクトです。

**is\_master** が **true** と同等のゾーングループを1つだけ持つ可能性があり、ゾーングループマップの最後に **master\_zonegroup** として指定する必要があります。以下の JSON オブジェクトは、デフォルトゾーングループマップの例です。

```
{
  "zonegroups": [
    {
      "key": "90b28698-e7c3-462c-a42d-4aa780d24eda",
      "val": {
        "id": "90b28698-e7c3-462c-a42d-4aa780d24eda",
        "name": "us",
        "api_name": "us",
        "is_master": "true",
        "endpoints": [
          "http://rgw1:80"
        ],
        "hostnames": [],
        "hostnames_s3website": [],
        "master_zone": "9248cab2-afe7-43d8-a661-a40bf316665e",
        "zones": [
          {
            "id": "9248cab2-afe7-43d8-a661-a40bf316665e",
            "name": "us-east",
            "endpoints": [
              "http://rgw1"
            ],
            "log_meta": "true",
            "log_data": "true",
            "bucket_index_max_shards": 0,
            "read_only": "false"
          },
          {
            "id": "d1024e59-7d28-49d1-8222-af101965a939",
            "name": "us-west",
            "endpoints": [
              "http://rgw2:80"
            ],
            "log_meta": "false",
```

```

        "log_data": "true",
        "bucket_index_max_shards": 0,
        "read_only": "false"
    }
],
"placement_targets": [
    {
        "name": "default-placement",
        "tags": []
    }
],
"default_placement": "default-placement",
"realm_id": "ae031368-8715-4e27-9a99-0c9468852cfe"
}
}
],
"master_zonegroup": "90b28698-e7c3-462c-a42d-4aa780d24eda",
"bucket_quota": {
    "enabled": false,
    "max_size_kb": -1,
    "max_objects": -1
},
"user_quota": {
    "enabled": false,
    "max_size_kb": -1,
    "max_objects": -1
}
}
}

```

ゾーングループマップを設定するには、次のコマンドを実行します。

```
# radosgw-admin zonegroup-map set --infile zonegroupmap.json
```

ここで、**zonegroupmap.json** は作成した JSON ファイルです。ゾーングループマップで指定したゾーンが作成されていることを確認します。最後に、期間を更新します。

```
# radosgw-admin period update --commit
```

### 5.8.3. ゾーン

Ceph Object Gateway はゾーンの概念をサポートします。ゾーンは、1つ以上の Ceph Object Gateway インスタンスで設定される論理グループを定義します。

ゾーンの設定は、Ceph 設定ファイル内で終了するすべての設定ではないため、一般的な設定手順とは異なります。ゾーンを一覧表示して、ゾーン設定を取得し、ゾーン設定を設定できます。



#### 重要

**radosgw-admin zone** 操作はすべて、ゾーン内で動作するまたはこれから動作するホストで実行する **必要があります**。

#### 5.8.3.1. ゾーンを作成

ゾーンを作成するには、ゾーン名を指定します。マスターゾーンの場合は、**--master** オプションを指定します。ゾーングループ内の1つのゾーンのみがマスターゾーンになることができます。ゾーングループにゾーンを追加するには、**--rgw-zonegroup** オプションをゾーングループ名で指定します。



### 重要

ゾーン内の Ceph Object Gateway ノードでゾーンを作成する必要があります。

```
[root@zone] radosgw-admin zone create --rgw-zone=<name> \  
  [--zonegroup=<zonegroup-name>] \  
  [--endpoints=<endpoint:port>[,<endpoint:port>] \  
  [--master] [--default] \  
  --access-key $SYSTEM_ACCESS_KEY --secret $SYSTEM_SECRET_KEY
```

次に、期間を更新します。

```
# radosgw-admin period update --commit
```

### 5.8.3.2. ゾーンの削除

ゾーンを削除するには、最初にゾーングループからこれを削除します。

```
# radosgw-admin zonegroup remove --zonegroup=<name> \  
  --zone=<name>
```

次に、期間を更新します。

```
# radosgw-admin period update --commit
```

次に、ゾーンを削除します。



### 重要

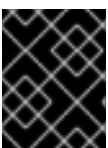
この手順では、ゾーン内のホストで **MUST** を実行する **必要があります**。

以下のコマンドを実行します。

```
[root@zone]# radosgw-admin zone delete --rgw-zone<name>
```

最後に、期間を更新します。

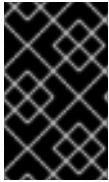
```
# radosgw-admin period update --commit
```



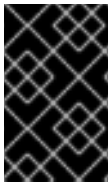
### 重要

ゾーングループから先にゾーンを削除せずに、ゾーンを削除しないでください。それ以外の場合には、期間の更新に失敗します。

削除したゾーンのプールが他に使用されていない場合は、プールを削除することを検討してください。以下の例の **<del-zone>** を、削除したゾーン名に置き換えます。

**重要**

Ceph がゾーンプールを削除すると、それによってリカバリー不可能な方法でその中のデータが削除されます。Ceph クライアントにプールの内容が必要なくなった場合にのみ、ゾーンプールを削除します。

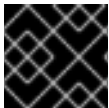
**重要**

マルチレルムクラスターでは、**.rgw.root** プールをゾーンプールと共に削除すると、クラスターのレルム情報のすべてが削除されます。**.rgw.root** プールを削除する前に、**.rgw.root** に他のアクティブなレルムが含まれていないことを確認します。

```
# rados rm_pool <del-zone>.rgw.control <del-zone>.rgw.control --yes-i-really-really-mean-it
# rados rm_pool <del-zone>.rgw.data.root <del-zone>.rgw.data.root --yes-i-really-really-mean-it
# rados rm_pool <del-zone>.rgw.gc <del-zone>.rgw.gc --yes-i-really-really-mean-it
# rados rm_pool <del-zone>.rgw.log <del-zone>.rgw.log --yes-i-really-really-mean-it
# rados rm_pool <del-zone>.rgw.users.uid <del-zone>.rgw.users.uid --yes-i-really-really-mean-it
```

### 5.8.3.3. ゾーンの変更

ゾーンを変更するには、ゾーン名と、変更するパラメーターを指定します。

**重要**

ゾーンは、ゾーン内にある Ceph Object Gateway ノードで変更する必要があります。

```
[root@zone]# radosgw-admin zone modify [options]
```

**--access-key=<key> --secret/--secret-key=<key> --master --default --endpoints=<list>**

次に、期間を更新します。

```
# radosgw-admin period update --commit
```

### 5.8.3.4. ゾーンの一覧

**root** でクラスター内のゾーンを一覧表示するには、以下を実行します。

```
$ sudo radosgw-admin zone list
```

### 5.8.3.5. ゾーンの取得

**root** でゾーンの設定を取得するには、次のコマンドを実行します。

```
$ sudo radosgw-admin zone get [--rgw-zone=<zone>]
```

**default** ゾーンは以下ようになります。

```
{ "domain_root": ".rgw",
  "control_pool": ".rgw.control",
  "gc_pool": ".rgw.gc",
```

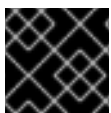
```

"log_pool": ".log",
"intent_log_pool": ".intent-log",
"usage_log_pool": ".usage",
"user_keys_pool": ".users",
"user_email_pool": ".users.email",
"user_swift_pool": ".users.swift",
"user_uid_pool": ".users.uid",
"system_key": { "access_key": "", "secret_key": ""},
"placement_pools": [
  { "key": "default-placement",
    "val": { "index_pool": ".rgw.buckets.index",
            "data_pool": ".rgw.buckets"}
  }
]
}

```

### 5.8.3.6. ゾーンの設定

ゾーンの設定には、一連の Ceph Object Gateway プールを指定する必要があります。一貫性を保つために、ゾーン名と同じプールの接頭辞を使用することが推奨されます。プールに関する詳細は、`Pools_` を参照してください。



#### 重要

ゾーン内の Ceph Object Gateway ノードでゾーンを設定する必要があります。

ゾーンを設定するには、プールで設定される JSON オブジェクトを作成し、オブジェクトをファイル (例: `zone.json`) に保存します。続いて以下のコマンドを実行して、`{zone-name}` をゾーンの名前に置き換えます。

```
[root@zone]$ sudo radosgw-admin zone set --rgw-zone={zone-name} --infile zone.json
```

ここで、`zone.json` は作成した JSON ファイルです。

次に、`root` でピリオドを更新します。

```
$ sudo radosgw-admin period update --commit
```

### 5.8.3.7. ゾーンの名前変更

ゾーンの名前を変更するには、ゾーン名および新しいゾーン名を指定します。ゾーン内のホストで以下を実行します。

```
[root@zone]# radosgw-admin zone rename --rgw-zone=<name> --zone-new-name=<name>
```

次に、期間を更新します。

```
# radosgw-admin period update --commit
```

## 5.9. ゾーングループとゾーン設定

デフォルトのゾーングループおよびゾーンを設定する場合、プール名にはゾーン名が含まれます。以下に例を示します。

- **default.rgw.control**

デフォルトを変更するには、各 **[client.rgw.{instance-name}]** インスタンスの配下にある Ceph 設定ファイルに以下の設定を追加します。

名前	説明	型	デフォルト
<b>rgw_zone</b>	ゲートウェイインスタンスのゾーンの名前。	文字列	なし
<b>rgw_zonegroup</b>	ゲートウェイインスタンスのゾーングループの名前。	文字列	なし
<b>rgw_zonegroup_root_pool</b>	ゾーングループの root プール。	文字列	<b>.rgw.root</b>
<b>rgw_zone_root_pool</b>	ゾーンの root プール。	文字列	<b>.rgw.root</b>
<b>rgw_default_zone_group_info_oid</b>	デフォルトのゾーングループを保存する OID。この設定を変更することは推奨していません。	文字列	<b>default.zonegroup</b>
<b>rgw_num_zone_operation_shards</b>	ゾーン間のグループ Synchronization の進行を維持するためのシャードの最大数。	整数	<b>128</b>

## 5.10. マルチサイトでのバケットの手動再シャーディング

{storage-product} は、マルチサイトクラスターの動的バケットリシャーディングをサポートしていません。次の手順を使用して、マルチサイトクラスター内のバケットを手動でリシャーディングできます。

### 注記

手動再シャーディングは、特に手動の再シャーディングを保証する巨大バケットの場合に、非常にコストのかかるプロセスです。すべてのセカンダリーゾーンは、すべてのオブジェクトを削除し、マスターゾーンからそれらを再同期します。

### 前提条件

- すべての Object Gateway インスタンスを停止します。

### 手順

1. マスターゾーングループのマスターゾーン内のノードで、以下のコマンドを実行します。

```
# radosgw-admin bucket sync disable --bucket=BUCKET_NAME
```

すべてのゾーンの **sync status** が、データの同期が最新であることを報告するのを待ちます。

2. すべてのゾーンですべての **ceph-radosgw** デーモンを停止します。



3. マスターゾーングループのマスターゾーン内のノードで、バケットを再シャーディングします。以下に例を示します。

```
# radosgw-admin bucket reshard --bucket=BUCKET_NAME --num-shards=NEW_SHARDS_NUMBER
```

4. 各セカンダリーゾーンで、以下を実行します。

```
# radosgw-admin bucket rm --purge-objects --bucket=BUCKET_NAME
```

5. すべてのゾーンですべての **ceph-radosgw** デーモンを再起動します。

6. マスターゾーングループのマスターゾーン内のノードで、以下のコマンドを実行します。

```
# radosgw-admin bucket sync enable --bucket=BUCKET_NAME
```

メタデータの同期プロセスでは、更新されたバケットエントリーポイントとバケットインスタンスのメタデータを取得します。データ同期プロセスは完全な同期を実行します。

## 関連情報

- [マルチサイト設定でのバケットインデックスシャーディングの設定](#)

## 5.11. レプリケーションなしでの複数ゾーンの設定

互いをレプリケートしない複数のゾーンを設定できます。たとえば、会社内の各チームに専用のゾーンを作成できます。

### 前提条件

- Ceph Object Gateway がインストールされている Ceph Storage Cluster。

### 手順

1. レルムを作成します。

```
radosgw-admin realm create --rgw-realm=realm-name [--default]
```

以下に例を示します。

```
[root@master-zone]# radosgw-admin realm create --rgw-realm=movies --default
{
  "id": "0956b174-fe14-4f97-8b50-bb7ec5e1cf62",
  "name": "movies",
  "current_period": "1950b710-3e63-4c41-a19e-46a715000980",
  "epoch": 1
}
```

2. ゾーングループを作成します。

```
radosgw-admin zonegroup create --rgw-zonegroup=zone-group-name --endpoints=url [--rgw-realm=realm-name|--realm-id=realm-id] --master --default
```

以下に例を示します。

```
[root@master-zone]# radosgw-admin zonegroup create --rgw-zonegroup=us --
endpoints=http://rgw1:80 --rgw-realm=movies --master --default
{
  "id": "f1a233f5-c354-4107-b36c-df66126475a6",
  "name": "us",
  "api_name": "us",
  "is_master": "true",
  "endpoints": [
    "http://rgw1:80"
  ],
  "hostnames": [],
  "hostnames_s3webzone": [],
  "master_zone": "",
  "zones": [],
  "placement_targets": [],
  "default_placement": "",
  "realm_id": "0956b174-fe14-4f97-8b50-bb7ec5e1cf62"
}
```

3. ユースケースに応じて、1つ以上のゾーンを作成します。

```
radosgw-admin zone create
  --rgw-zonegroup=zone-group-name \
  --rgw-zone=zone-name \
  --master --default \
  --endpoints=http://fqdn:port[,http://fqdn:port]
```

以下に例を示します。

```
[root@master-zone]# radosgw-admin zone create --rgw-zonegroup=us \
  --rgw-zone=us-east \
  --master --default \
  --endpoints=http://rgw1:80
```

4. ゾーングループの設定が含まれる JSON ファイルを取得します。

```
radosgw-admin zonegroup get --rgw-zonegroup=zone-group-name > zonegroup.json
```

以下に例を示します。

```
[root@master-zone]# radosgw-admin zonegroup get --rgw-zonegroup=us > zonegroup.json
```

5. ファイルで、**log\_meta** パラメーター、**log\_data** パラメーター、および **sync\_from\_all** パラメーターを **false** に設定します。

```
{
  "id": "72f3a886-4c70-420b-bc39-7687f072997d",
  "name": "default",
  "api_name": "",
  "is_master": "true",
  "endpoints": [],
  "hostnames": [],
```

```

"hostnames_s3website": [],
"master_zone": "a5e44ecd-7aae-4e39-b743-3a709acb60c5",
"zones": [
  {
    "id": "975558e0-44d8-4866-a435-96d3e71041db",
    "name": "testzone",
    "endpoints": [],
    "log_meta": "false",
    "log_data": "false",
    "bucket_index_max_shards": 0,
    "read_only": "false",
    "tier_type": "",
    "sync_from_all": "false",
    "sync_from": []
  },
  {
    "id": "a5e44ecd-7aae-4e39-b743-3a709acb60c5",
    "name": "default",
    "endpoints": [],
    "log_meta": "false",
    "log_data": "false",
    "bucket_index_max_shards": 0,
    "read_only": "false",
    "tier_type": "",
    "sync_from_all": "false",
    "sync_from": []
  }
],
"placement_targets": [
  {
    "name": "default-placement",
    "tags": []
  }
],
"default_placement": "default-placement",
"realm_id": "2d988e7d-917e-46e7-bb18-79350f6a5155"
}

```

6. 更新された JSON ファイルを使用します。

```
radosgw-admin zonegroup set --rgw-zonegroup=zone-group-name --infile=zonegroup.json
```

以下に例を示します。

```
[root@master-zone]# radosgw-admin zonegroup set --rgw-zonegroup=us --
infile=zonegroup.json
```

7. 期間を更新します。

```
# radosgw-admin period update --commit
```

## 関連情報

- [レルム](#)

- [ゾーングループ](#)
- [ゾーン](#)
- [Ubuntu インストールガイド](#)

## 5.12. 同じストレージクラスターに複数のレルムの設定

このセクションでは、同じストレージクラスターに複数のレルムを設定する方法を説明します。これは、マルチサイトの高度なユースケースです。同一のストレージクラスター内に複数のレルムを設定することで、ローカルの RGW クライアントのトラフィックを処理するためのローカルレルムと、セカンダリーサイトに複製されるデータ用のレプリケートされたレルムを使用することができます。



### 注記

Red Hat では、各レルムに独自の Ceph Object Gateway があることを推奨しています。

### 前提条件

- ストレージクラスター内の各データセンターのアクセスキーおよびシークレットキー。
- ストレージクラスター内の 2 つの実行中の {storage-product} データセンター。
- 各データセンターには独自のローカルレルムがあります。両方のサイトでレプリケートするレルムを共有する。
- Ceph Object Gateway ノード上で、{storage-product} インストールガイドの [Red Hat Ceph Storage のインストール要件](#) に記載のタスクを実行する。
- 各 Ceph Object Gateway ノードについて、{storage-product} インストールガイドの [Ceph Object Gateway のインストール](#) セクションに記載のステップ 1 から 7 を実施する。

### 手順

1. 同期ユーザーを作成します。

#### Syntax

```
radosgw-admin user create --uid="SYNCHRONIZATION_USER" --display-name="Synchronization User" --system
```

2. ストレージクラスターの最初のデータセンターにローカルレルムを 1 つ作成します。

#### Syntax

```
radosgw-admin realm create --rgw-realm=REALM_NAME --default
```

#### 例

```
[user@rgw1]$ radosgw-admin realm create --rgw-realm=ldc1 --default
```

3. 最初のデータセンター上に、1 つのローカルマスターゾーングループを作成します。

## Syntax

```
radosgw-admin zonegroup create --rgw-zonegroup=ZONE_GROUP_NAME --
endpoints=http://RGW_NODE_NAME:80 --rgw-realm=REALM_NAME --master --default
```

## 例

```
[user@rgw1]$ radosgw-admin zonegroup create --rgw-zonegroup=ldc1zg --
endpoints=http://rgw1:80 --rgw-realm=ldc1 --master --default
```

- 最初のデータセンターに1つのローカルゾーンを作成します。

## Syntax

```
radosgw-admin zone create --rgw-zonegroup=ZONE_GROUP_NAME --rgw-
zone=ZONE_NAME --master --default --endpoints=HTTP_FQDN[,HTTP_FQDN]
```

## 例

```
[user@rgw1]$ radosgw-admin zone create --rgw-zonegroup=ldc1zg --rgw-zone=ldc1z --
master --default --endpoints=http://rgw.example.com
```

- 期間をコミットします。

## 例

```
[user@rgw1]$ radosgw-admin period update --commit
```

- `ceph.conf` を、`rgw_realm` 名、`rgw_zonegroup` 名、および `rgw_zone` 名で更新します。

## Syntax

```
rgw_realm = REALM_NAME
rgw_zonegroup = ZONE_GROUP_NAME
rgw_zone = ZONE_NAME
```

## 例

```
rgw_realm = ldc1
rgw_zonegroup = ldc1zg
rgw_zone = ldc1z
```

- RGW デーモンを再起動します。

## Syntax

```
systemctl restart ceph-radosgw@rgw.$(hostname -s).rgw0.service
```

- ストレージクラスターの2番目のデータセンターに、ローカルレルムを1つ作成します。

## Syntax

```
radosgw-admin realm create --rgw-realm=REALM_NAME --default
```

### 例

```
[user@rgw2]$ radosgw-admin realm create --rgw-realm=ldc2 --default
```

- 2 番目のデータセンターに、1つのローカルマスターゾーングループを作成します。

### Syntax

```
radosgw-admin zonegroup create --rgw-zonegroup=ZONE_GROUP_NAME --  
endpoints=http://RGW_NODE_NAME:80 --rgw-realm=REALM_NAME --master --default
```

### 例

```
[user@rgw2]$ radosgw-admin zonegroup create --rgw-zonegroup=ldc2zg --  
endpoints=http://rgw2:80 --rgw-realm=ldc2 --master --default
```

- 2 番目のデータセンターに1つのローカルゾーンを作成します。

### Syntax

```
radosgw-admin zone create --rgw-zonegroup=ZONE_GROUP_NAME --rgw-  
zone=ZONE_NAME --master --default --endpoints=HTTP_FQDN[, HTTP_FQDN]
```

### 例

```
[user@rgw2]$ radosgw-admin zone create --rgw-zonegroup=ldc2zg --rgw-zone=ldc2z --  
master --default --endpoints=http://rgw.example.com
```

- 期間をコミットします。

### 例

```
[user@rgw2]$ radosgw-admin period update --commit
```

- `ceph.conf` を、`rgw_realm` 名、`rgw_zonegroup` 名、および `rgw_zone` 名で更新します。

### Syntax

```
rgw_realm = REALM_NAME  
rgw_zonegroup = ZONE_GROUP_NAME  
rgw_zone = ZONE_NAME
```

### 例

```
rgw_realm = ldc2  
rgw_zonegroup = ldc2zg  
rgw_zone = ldc2z
```

- RGW デーモンを再起動します。

## Syntax

```
systemctl restart ceph-radosgw@rgw.${hostname -s}.rgw0.service
```

- レプリケーション/同期ユーザーを作成します。

## Syntax

```
radosgw-admin user create --uid="r_REPLICATION_SYNCHRONIZATION_USER" --display-name="Replication-Synchronization User" --system
```

- ストレージクラスターの最初のデータセンターにレプリケートされたレルムを作成します。

## Syntax

```
radosgw-admin realm create --rgw-realm=REPLICATED_REALM_1
```

## 例

```
[user@rgw1] radosgw-admin realm create --rgw-realm=rdc1
```

- 最初のデータセンターのマスターゾーングループを作成します。

## Syntax

```
radosgw-admin zonegroup create --rgw-zonegroup=RGW_ZONE_GROUP --endpoints=http://_RGW_NODE_NAME:80 --rgw-realm=_RGW_REALM_NAME --master --default
```

## 例

```
[user@rgw1] radosgw-admin zonegroup create --rgw-zonegroup=rdc1zg --endpoints=http://rgw1:80 --rgw-realm=rdc1 --master --default
```

- 最初のデータセンターにマスターゾーンを作成します。

## Syntax

```
radosgw-admin zone create --rgw-zonegroup=RGW_ZONE_GROUP --rgw-zone=MASTER_RGW_NODE_NAME --master --default --endpoints=HTTP_FQDN[,HTTP_FQDN]
```

## 例

```
[user@rgw1] radosgw-admin zone create --rgw-zonegroup=rdc1zg --rgw-zone=rdc1z --master --default --endpoints=http://rgw.example.com
```

- 期間をコミットします。

## Syntax

```
radosgw-admin period update --commit
```

- 最初のデータセンターの **rgw\_realm** 名、**rgw\_zonegroup** 名、および **rgw\_zone** 名で **ceph.conf** を更新します。

### Syntax

```
rgw_realm = REALM_NAME  
rgw_zonegroup = ZONE_GROUP_NAME  
rgw_zone = ZONE_NAME
```

### 例

```
rgw_realm = rdc1  
rgw_zonegroup = rdc1zg  
rgw_zone = rdc1z
```

- RGW デーモンを再起動します。

### Syntax

```
systemctl restart ceph-radosgw@rgw.$(hostname -s).rgw0.service
```

- 2 番目のデータセンターでレプリケートされたレルムをプルします。

### Syntax

```
radosgw-admin realm pull --url=https://tower-osd1.cephtips.com --access-  
key=ACCESS_KEY --secret-key=SECRET_KEY
```

### 例

```
radosgw-admin realm pull --url=https://tower-osd1.cephtips.com --access-  
key=3QV0D6ZMMCJZMSCXJ2QJ --secret-  
key=VpvQWcsfI9OPzUCpR4kynDLAbqa1OIKqRB6WEnH8
```

- 最初のデータセンターから期間をプルします。

### Syntax

```
radosgw-admin period pull --url=https://tower-osd1.cephtips.com --access-  
key=ACCESS_KEY --secret-key=SECRET_KEY
```

### 例

```
radosgw-admin period pull --url=https://tower-osd1.cephtips.com --access-  
key=3QV0D6ZMMCJZMSCXJ2QJ --secret-  
key=VpvQWcsfI9OPzUCpR4kynDLAbqa1OIKqRB6WEnH8
```

- 2 番目のデータセンターにセカンダリーゾーンを作成します。



## Syntax

```
radosgw-admin zone create --rgw-zone=RGW_ZONE --rgw-  
zonegroup=RGW_ZONE_GROUP --endpoints=https://tower-osd4.cephtips.com --  
access-key=_ACCESS_KEY --secret-key=SECRET_KEY
```

## 例

```
[user@rgw2] radosgw-admin zone create --rgw-zone=rdc2z --rgw-zonegroup=rdc1zg --  
endpoints=https://tower-osd4.cephtips.com --access-key=3QV0D6ZMMCJZMSCXJ2QJ --  
secret-key=VpvQWcsfI9OPzUCpR4kynDLAbqa1OIKqRB6WEnH8
```

24. 期間をコミットします。

## Syntax

```
radosgw-admin period update --commit
```

25. 2番目のデータセンターの **rgw\_realm** 名、**rgw\_zonegroup** 名、および **rgw\_zone** 名を持つ **ceph.conf** を更新します。

## Syntax

```
rgw_realm = REALM_NAME  
rgw_zonegroup = ZONE_GROUP_NAME  
rgw_zone = ZONE_NAME
```

## 例

```
rgw realm = rdc1  
rgw zonegroup = rdc1zg  
rgw zone = rdc2z
```

26. RGW デーモンを再起動します。

## Syntax

```
systemctl restart ceph-radosgw@rgw.$(hostname -s).rgw0.service
```

27. 2番目のデータセンターのエンドポイントに **root** としてログインします。
28. マスターレルムで同期のステータスを確認します。

## Syntax

```
radosgw-admin sync status
```

## 例

```
[root@tower-osd4 ceph-ansible]# radosgw-admin sync status  
realm 59762f08-470c-46de-b2b1-d92c50986e67 (ldc2)  
zonegroup 7cf8daf8-d279-4d5c-b73e-c7fd2af65197 (ldc2zg)
```

```
zone 034ae8d3-ae0c-4e35-8760-134782cb4196 (ldc2z)
metadata sync no sync (zone is master)
```

29. 最初のデータセンターのエンドポイントに **root** としてログインします。
30. レプリケーション同期レルムの同期ステータスを確認します。

## Syntax

```
radosgw-admin sync status --rgw-realm RGW_REALM_NAME
```

以下に例を示します。

```
[root@tower-osd4 ceph-ansible]# [root@tower-osd4 ceph-ansible]# radosgw-admin sync
status --rgw-realm rdc1
  realm 73c7b801-3736-4a89-aaf8-e23c96e6e29d (rdc1)
  zonegroup d67cc9c9-690a-4076-89b8-e8127d868398 (rdc1zg)
  zone 67584789-375b-4d61-8f12-d1cf71998b38 (rdc2z)
metadata sync syncing
  full sync: 0/64 shards
  incremental sync: 64/64 shards
  metadata is caught up with master
data sync source: 705ff9b0-68d5-4475-9017-452107cec9a0 (rdc1z)
  syncing
  full sync: 0/128 shards
  incremental sync: 128/128 shards
  data is caught up with source
  realm 73c7b801-3736-4a89-aaf8-e23c96e6e29d (rdc1)
  zonegroup d67cc9c9-690a-4076-89b8-e8127d868398 (rdc1zg)
  zone 67584789-375b-4d61-8f12-d1cf71998b38 (rdc2z)
metadata sync syncing
  full sync: 0/64 shards
  incremental sync: 64/64 shards
  metadata is caught up with master
data sync source: 705ff9b0-68d5-4475-9017-452107cec9a0 (rdc1z)
  syncing
  full sync: 0/128 shards
  incremental sync: 128/128 shards
  data is caught up with source
```

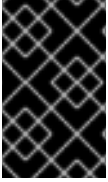
31. ローカルサイトにデータを保存およびアクセスするには、ローカルレルムのユーザーを作成します。

## Syntax

```
radosgw-admin user create --uid="LOCAL_USER" --display-name="Local user" --rgw-
realm=_REALM_NAME --rgw-zonegroup=ZONE_GROUP_NAME --rgw-
zone=ZONE_NAME
```

## 例

```
[user@rgw2] #radosgw-admin user create --uid="local-user" --display-name="Local user" --
rgw-realm=ldc1 --rgw-zonegroup=ldc1zg --rgw-zone=ldc1z
```



## 重要

デフォルトでは、マルチサイト設定にユーザーが追加されます。ユーザーがローカルゾーン内のデータにアクセスするには、**radosgw-admin** コマンドに **--rgw-realm** 引数が必要です。