Red Hat Ceph Storage 4

Object Gateway 設定および管理ガイド

Ceph Storage Object Gateway の設定および管理

Last Updated: 2023-08-26

Red Hat Ceph Storage 4 Object Gateway 設定および管理ガイド

Ceph Storage Object Gateway の設定および管理

法律上の通知

概要

本書では、Ceph Storage Object Gateway を設定および管理する手順について説明します。 Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、弊社の CTO、Chris Wright のメッセージ を参照してください。

目次

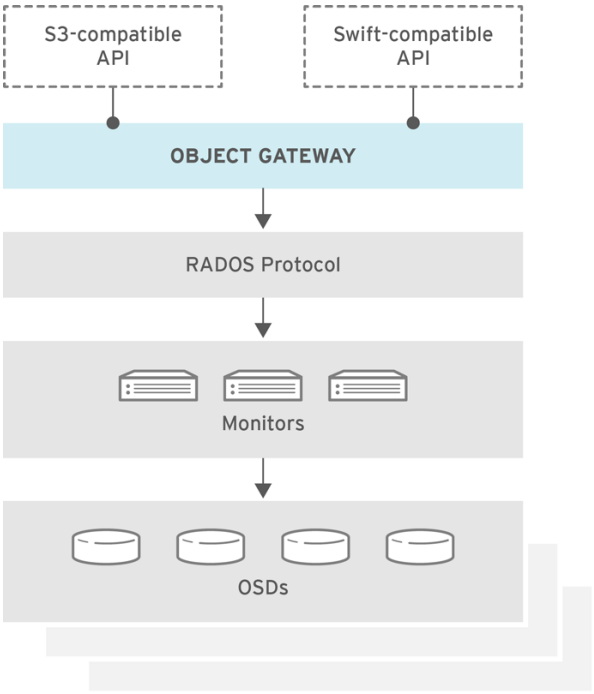
第1章 概要

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. BEAST および CIVETWEB フロントエンドの WEB サーバー

Ceph Object Gateway は、フロントエンドとして Beast および CivetWeb を提供します。どちらも C/C++ 埋め込み Web サーバーです。

Beast

Red Hat Ceph Storage 4 以降、Beast はデフォルトのフロントエンド Web サーバーです。Red Hat Ceph Storage 3 からアップグレードすると、**rgw_frontends** パラメーターが自動的に Beast に変更になります。Beast は **Boost.Beast** C++ ライブラリーを使用して HTTP を解析し、 **Boost.Asio** を使用して非同期ネットワーク I/O を行います。

CivetWeb

Red Hat Ceph Storage 3 では、CivetWeb がデフォルトのフロントエンドですが、それに応じて **rgw_frontends** オプションを設定して Beast を使用することもできます。CivetWeb は、Mongoose プロジェクトのフォークである HTTP ライブラリーです。

関連情報

- Boost C++ ライブラリー
- GitHub の Civetweb

2.2. 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。Red Hat Ceph Storage バージョン 3.x では、CivetWeb がデフォルトのフロントエンドとなり、Beast フロントエンドは Red Hat Ceph Storage 設定ファイルの **rgw_frontends** で指定する必要があります。Red Hat Ceph Storage バージョン 4.0 の時点で、Beast フロントエンドはデフォルトであり、Red Hat Ceph Storage 3.x からのアップグレードにより、**rgw_frontends** パラメーターが自動的に Beast に変更になります。

関連情報

- **Beast 設定オプション**

2.3. 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

オプション	説明	デフォルト
endpoint および **ssl_endpoint**	**address[:port]** 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、**endpoint** が **8080** になり、**ssl_endpoint** が **443** になります。**endpoint=[::1] endpoint=192.168.0.100:8000** のように複数回指定できます。	空
ssl_certificate	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。ファイルが複数の項目を含む PEM ファイルである場合、順番は重要です。ファイルは、RGW サーバーキーで始まり、次に中間証明書、最後に C A 証明書となる必要があります。	空
ssl_private_key	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。**ssl_certificate** 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
tcp_nodelay	一部の環境でのパフォーマンスの最適化。	空
request_timeout_ms	Beast フロントエンドの明示的な要求タイムアウトを設定します。より大きな要求タイムアウトを設定すると、ゲートウェイは低速のクライアント (たとえば、待ち時間の長いネットワークを介して接続されたクライアント) に対してより耐性を持つようになります。	65

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>
```

注記

デフォルトでは、Beast フロントエンドは、サーバーによって処理されるすべての要求を記録するアクセスログラインを RADOS Gateway ログファイルに書き込みます。

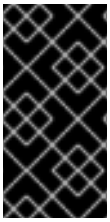
関連情報

- 詳細は、Beast フロントエンド を参照してください。

2.4. 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 4 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 4 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

- 稼働中の Red Hat Ceph Storage 4.1 クラスターがある。
- 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. 以下の例のような Ceph Object Gateway (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。

```
# 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。

```
# firewall-cmd --list-all
```

6. 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。

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

関連情報

- CivetWeb での SSL の使用
- Civetweb 設定オプション

2.5. CIVETWEB での SSL の使用

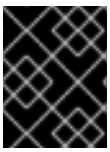
Red Hat Ceph Storage 1 では、Ceph Object Gateway に対する Civetweb SSL サポートは HAProxy および keepalived に依存していました。Red Hat Ceph Storage 2 以降のリリースでは、Civetweb は OpenSSL ライブラリーを使用して Transport Layer Security (TLS) を提供できます。

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"
```

2.6. 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
access_log_file	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
error_log_file	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
num_threads	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。	50
request_timeout_ms	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

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
```

CivetWeb および Beast フロントエンドはどちらも、サーバーによって処理されるすべての要求のアクセスログ行記録を RADOS ゲートウェイログファイルに書き込みます。

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.rgw0]
...
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 civetweb = 20
```

RGW デバッグログの場合は、Ceph 設定ファイルの **[client.rgw.{instance}]** セクションに以下のパラメーターを追加します。

```
[client.rgw.rgw1.rgw0]
...
debug rgw = 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。

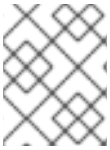
```
# ceph tell osd.0 injectargs --debug_civetweb 10/20
```

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。

ロギングおよびデバッグに関する一般的な情報は、**Red Hat Ceph Storage 設定ガイド**の **Ceph デバッグおよびログ設定** セクションを参照してください。

2.9. S3 サーバー側の暗号化

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



注記

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

重要

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

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

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

暗号化キーの管理には、以下の 2 つのオプションがあります。

お客様提供のキー

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

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

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

キー管理サービス

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

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

重要

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

関連情報

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

2.10. サーバー側の暗号化要求

実稼働環境では、クライアントはプロキシーを介して Ceph Object Gateway に接続されることがよくあります。このプロキシーは、複数の Ceph Object Gateway に接続するため、ロードバランサーと呼ばれます。クライアントが Ceph Object Gateway に要求を送信する場合、ロードバランサーはこれらの要求を複数の Ceph Object Gateway にルーティングし、ワークロードを配布します。

このような設定では、ロードバランサーでも、ロードバランサーと複数の Ceph Object Gateways の間でも、SSL の終了が発生する可能性があります。通信は HTTP のみを使用して行われます。サーバー側の暗号化要求を受け入れるように Ceph Object Gateway を設定するには、Configuring server-side encryption を参照してください。

2.11. サーバー側の暗号化の設定

ストレージ管理者は、SSL 経由で暗号化された要求を送信できない場合に、HTTP を使用して Ceph Object Gateway に要求を送信するようにサーバー側の暗号化を設定できます。

以下の手順では、HAProxy をプロキシーおよびロードバランサーとして使用します。

前提条件

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

手順

1. **haproxy.cfg** ファイルを編集します。

   例

```
frontend http_web
    bind *: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 10.0.0.71:8080 check
    server  rgw2 10.0.0.80:8080 check
```

2. **http** フロントエンドへのアクセスを許可する行をコメントアウトし、代わりに **https** フロントエンドを使用するように HAProxy に指示する手順を追加します。

   例

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

frontend rgw-https
  bind *:443 ssl crt /etc/ssl/private/example.com.pem
  http-request set-header X-Forwarded-Proto https if { ssl_fc }
  http-request set-header X-Forwarded-Proto https
# here we set the incoming HTTPS port on the load balancer (eg : 443)
  http-request set-header X-Forwarded-Port 443
  default_backend rgw

backend rgw
    balance roundrobin
    mode http
    server  rgw1 10.0.0.71:8080 check
    server  rgw2 10.0.0.80:8080 check
```

3. クラスターのすべてのノードで、以下のパラメーターを Ceph 設定ファイルの **[global]** セクションに追加します。

```
  rgw_trust_forwarded_https=true
```

4. HAProxy を有効にして起動します。

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

5. Ansible の実行時に **rgw_trust_forwarded_https=true** が Ceph 設定ファイルから削除されないようにするには、ceph-ansible **all.yml** ファイルを編集し、**ceph_conf_overrides** / **global** セクションの **rgw_trust_forwarded_https** を **true** に設定します。

```
ceph_conf_overrides:
  global:
    rgw_trust_forwarded_https: true
```

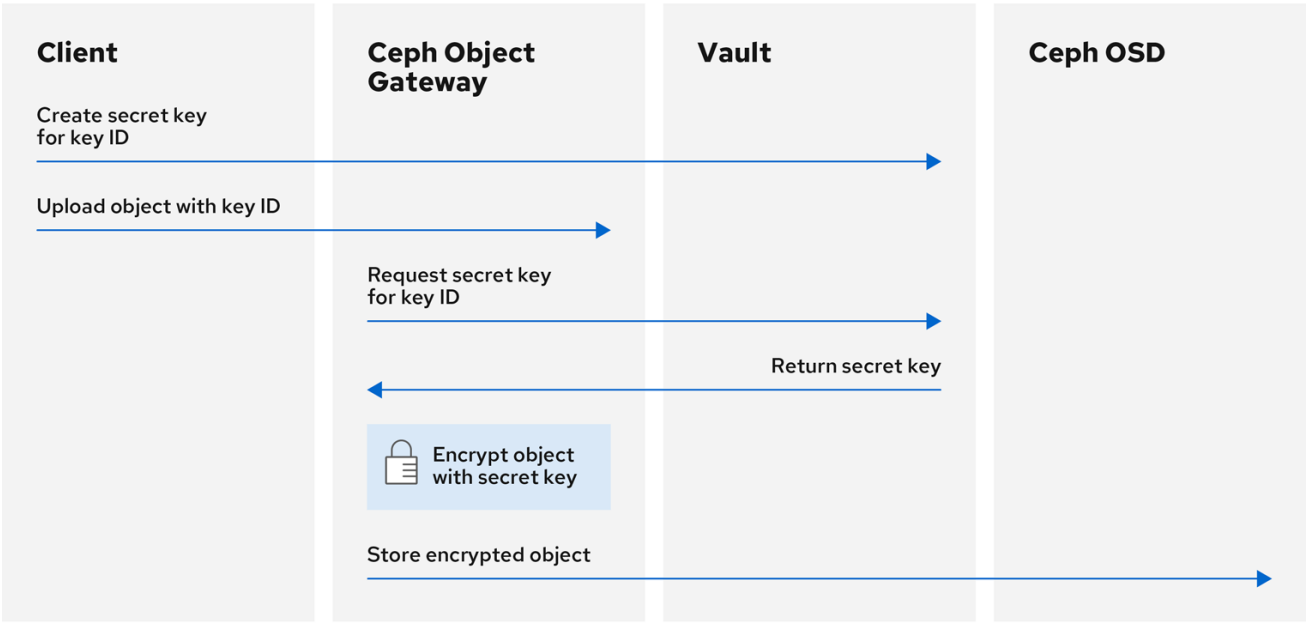
6. 変更が完了したら、ceph-ansible Playbook を実行して、すべての Ceph ノードの設定を更新します。

関連情報

- HAProxy のインストールおよび設定
- Ceph クライアントロールのインストール
- Red Hat ストレージクラスターのインストール

2.12. HASHICORP VAULT

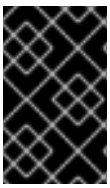
ストレージ管理者は、Ceph Object Gateway で使用するために、キー、パスワード、および証明書を HAshiCorp Vault に安全に保存できます。HashiCorp Vault は、Ceph Object Gateway で使用されるサーバー側の暗号化にセキュアなキー管理サービスを提供します。



基本的なワークフロー:

1. クライアントは、オブジェクトのキー ID に基づいて Vault からシークレットキーの作成を要求します。
2. クライアントはオブジェクトのキー ID を持つオブジェクトを Ceph Object Gateway にアップロードします。
3. 次に、Ceph Object Gateway は Vault から新規に作成されたシークレットキーを要求します。

4. Vault は、Ceph Object Gateway に秘密鍵を返して要求に応えます。
5. これで、Ceph Object Gateway は、新しい秘密鍵を使用してオブジェクトを暗号化できます。
6. 暗号化が完了すると、オブジェクトが Ceph OSD に保存されます。



重要

Red Hat は、弊社のテクノロジーパートナーと協力して、本書をお客様にサービスとして提供します。ただし、Red Hat はこの製品のサポートを提供しません。この製品の技術的なサポートが必要な場合は、Hashicorp にサポートを依頼してください。

2.12.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph Object Gateway ソフトウェアのインストール。
- HashiCorp Vault ソフトウェアのインストール。

2.12.2. Vault のシークレットエンジン

HashiCorp Vault は、データを生成、保存、または暗号化するためのシークレットエンジンを複数提供します。アプリケーションプログラミングインターフェイス (API) は、データに対するアクションを求めるデータ呼び出しをシークレットエンジンに送信し、シークレットエンジンはそのアクション要求の結果を返します。

Ceph Object Gateway は、HashiCorp Vault シークレットエンジンの 2 つをサポートします。

- キー/値のバージョン 2
- Transit

キー/値のバージョン 2

Key/Value シークレットエンジンは、ディスク上の Vault 内にランダムのシークレットを保存します。**kv** エンジンのバージョン 2 では、設定可能な数のバージョン数をキーに指定できます。デフォルトのバージョン数は 10 です。バージョンを削除しても元のデータは削除されませんが、データが削除されたことになり、削除されたバージョンを元に戻すことができます。鍵名は文字列にする必要があります。また、コマンドラインインターフェイスの使用時に、エンジンは文字列以外の値を文字列に変換します。文字列以外の値を保持するには、JSON ファイルを指定するか、HTTP アプリケーションプログラミングインターフェイス (API) を使用します。

注記

アクセス制御リスト (ACL) ポリシーの場合、キー/値のシークレットエンジンは **作成** 機能と **更新** 機能の違いを認識します。

Transit

Transit シークレットエンジンは、移動データで暗号化機能を実行します。Transit シークレットエンジンはハッシュを生成したり、ランダムバイトのソースとなったり、データの署名や検証を行うことができます。Vault は、Transit シークレットエンジンの使用時にデータを保存しません。Transit シークレットエンジンは、複数の目的で同じ鍵を使用できるようにすることで鍵導出をサポートします。また、transit シークレットエンジンは鍵バージョン管理をサポートします。Transit シークレットエンジンは、以下のキータイプをサポートします。

aes128-gcm96

128 ビットの AES キーと 96 ビットのノンスを備えた AES-GCM。暗号化、復号、鍵導出、および収束暗号化をサポートします。

aes256-gcm96

256 ビットの AES キーと 96 ビットのノンスを備えた AES-GCM。暗号化、復号、鍵導出、および収束暗号化 (デフォルト) をサポートします。

chacha20-poly1305

256 ビットキーの ChaCha20-Poly1305。暗号化、復号、鍵導出、および収束暗号化をサポートします。

ed25519

Ed25519。署名、署名の検証、および鍵導出をサポートします。

ecdsa-p256

曲線 P-256 を使用した ECDSA。署名および署名の検証をサポートします。

ecdsa-p384

曲線 P-384 を使用した ECDSA。署名および署名の検証をサポートします。

ecdsa-p521

曲線 P-521 を使用した ECDSA。署名および署名の検証をサポートします。

rsa-2048

2048 ビット RSA 鍵、暗号化、復号、署名、および署名の検証をサポートします。

rsa-3072

3072 ビット RSA 鍵。暗号化、復号、署名、および署名の検証をサポートします。

rsa-4096

4096 ビットの RSA 鍵。暗号化、復号、署名、および署名の検証をサポートします。

関連情報

- 詳細は、Vault のプロジェクトサイトにある KV Secrets Engine ドキュメントを参照してください。
- 詳細は、Vault のプロジェクトサイトにある Transport Secrets Engine ドキュメントを参照してください。

2.12.3. Vault の認証

HashiCorp Vault は、複数のタイプの認証メカニズムをサポートします。Ceph Object Gateway は現在、Vault エージェントとトークン認証メソッドをサポートしています。Ceph Object Gateway は **rgw_crypt_vault_auth** オプションおよび **rgw_crypt_vault_addr** オプションを使用して HashiCorp Vault を使用するように設定します。

トークン

トークン認証方法により、ユーザーはトークンを使用して認証できます。新規トークンの作成、トークンごとのシークレットの取り消し、およびその他の多くのトークン操作が可能です。トークンストアを使用すると、他の認証方法をバイパスできます。トークン認証方法を使用する場合は、**rgw_crypt_vault_token_file** オプションも使用する必要があります。トークンファイルは、Ceph Object Gateway でのみ読み取り可能です。また、特定のパスからのキーリングの取得を可能にする制限されたポリシーを持つ Vault トークンを使用する必要があります。

警告

Red Hat は、実稼働環境でのトークン認証の使用を推奨していません。

Vault エージェント

Vault エージェントは、クライアントノードで実行するデーモンで、トークンの更新と共にクライアント側のキャッシュを提供します。Vault エージェントは、通常 Ceph Object Gateway ノードで実行されます。

関連情報

- 詳細は、Vault のプロジェクトサイトにある Token Auth Method ドキュメントを参照してください。
- 詳細は、Vault のプロジェクトサイトにある Vault Agent ドキュメントを参照してください。

2.12.4. Vault の namespace

HashiCorp Vault をエンタープライズサービスとして使用すると、組織内のチームが使用可能な分離された名前空間の一元管理が提供されます。これらの分離された名前空間環境は **テナント** と呼ばれ、組織内のチームがこれらの **テナント** を使用して、ポリシー、シークレット、ID を他のチームから分離することができます。Vault の名前空間機能は、単一インフラストラクチャー内からセキュアなマルチテナンシーをサポートします。

関連情報

- 詳細は、Vault のプロジェクトサイトにある Vault Enterprise Namespaces ドキュメントを参照してください。

2.12.5. Vault を使用するために Ceph Object Gateway の設定

Ceph Object Gateway が HashiCorp Vault を使用するように設定するには、暗号化キーストアとして設定する必要があります。現在、Ceph Object Gateway は 2 つの異なるシークレットエンジンと、2 つの異なる認証方法をサポートしています。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph Object Gateway ソフトウェアのインストール。
- Ceph Object Gateway ノードへのルートレベルのアクセス。

手順

1. Ceph 設定ファイル (デフォルトでは **/etc/ceph/ceph.conf**) を編集するためにを開き、Vault を暗号化キーストアとして有効にします。

```
rgw_crypt_s3_kms_backend = vault
```

2. **[client.radosgw.INSTANCE_NAME]** セクションで、Token または Vault エージェントのいずれかの Vault 認証メソッドを選択します。

   a. **Token** を使用している場合は、以下の行を追加します。

   ```
   rgw_crypt_vault_auth = token
   rgw_crypt_vault_token_file = /etc/ceph/vault.token
   rgw_crypt_vault_addr = http://VAULT_SERVER:8200
   ```

   b. **Vault エージェント** を使用している場合は、以下の行を追加します。

   ```
   rgw_crypt_vault_auth = agent
   rgw_crypt_vault_addr = http://VAULT_SERVER:8100
   ```

3. **[client.radosgw.INSTANCE_NAME]** セクションで、Key/Value または Transit のいずれかの Vault シークレットエンジンを選択します。

   a. **Key/Value** を使用している場合は、以下の行を追加します。

   ```
   rgw_crypt_vault_secret_engine = kv
   ```

   b. **Transit** を使用している場合は、以下の行を追加します。

   ```
   rgw_crypt_vault_secret_engine = transit
   ```

4. 必要に応じて、**[client.radosgw.INSTANCE_NAME]** セクションで、暗号化キーを取得する Vault 名前空間を設定できます。

   ```
   rgw_crypt_vault_namespace = NAME_OF_THE_NAMESPACE
   ```

5. パス接頭辞を設定し、Ceph Object Gateway が Vault から暗号化キーを取得する場所を制限します。

   例

   ```
   rgw_crypt_vault_prefix = /v1/secret/data
   ```

   a. エクスポート可能な Transit キーの場合、以下のように接頭辞パスを設定します。

   ```
   rgw_crypt_vault_prefix = /v1/transit/export/encryption-key
   ```

   Vault サーバーのドメイン名が **vault-server** である場合は、Ceph Object Gateway は以下の URL から暗号化されたトランジションキーを取得します。

   例

   ```
   http://vault-server:8200/v1/transit/export/encryption-key
   ```

6. 変更を Ceph 設定ファイルに保存します。

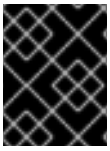
関連情報

詳細は、**Red Hat Ceph Storage Object Gateway 設定および管理ガイド** の **Vault 暗号化キー**

- 詳細は、**Red Hat Ceph Storage Object Gateway 設定および管理ガイド**の Vault 向けシークレットエンジン セクションを参照してください。
- 詳細は、**Red Hat Ceph Storage Object Gateway 設定および管理ガイド**の Vault 向け認証 セクションを参照してください。

2.12.6. kv エンジンを使用したキーの作成

HashiCorp Vault Key/Value シークレットエンジン (**kv**) を設定し、Ceph Object Gateway で使用するためのキーを作成できるようにします。シークレットは、**kv** シークレットエンジンのキーと値のペアとして保存されます。



重要

サーバー側の暗号化のキーは 256 ビットの長さで、**base64** を使用してエンコードする必要があります。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- HashiCorp Vault ソフトウェアのインストール。
- HashiCorp Vault ノードへの root レベルのアクセス。

手順

1. キー/値 バージョン 2 シークレットエンジンを有効にします。

```
[root@vault ~]# vault secrets enable kv-v2
```

2. 新しいキーを作成します。

構文

```
vault kv put secret/PROJECT_NAME/BUCKET_NAME key=$(openssl rand -base64 32)
```

例

```
[root@vault ~]# vault kv put secret/myproject/mybucketkey key=$(openssl rand -base64 32)

====== Metadata ======
Key              Value
---              -----
created_time     2020-02-21T17:01:09.095824999Z
deletion_time    n/a
destroyed        false
version          1
```

2.12.7. 転送エンジンを使用した鍵の作成

HashiCorp Vault 遷移シークレットエンジン (**transit**) を設定して、Ceph Object Gateway で使用するキーを作成できるようにします。Ceph Object Gateway でサーバー側の暗号化に使用するためには、Transit シークレットエンジンで作成した鍵がエクスポート可能である必要があります。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- HashiCorp Vault ソフトウェアのインストール。
- HashiCorp Vault ノードへの root レベルのアクセス。

手順

1. Transit シークレットエンジンを有効にします。

```
[root@vault ~]# vault secrets enable transit
```

2. 新しいエクスポート可能なキーを作成します。

構文

```
vault write -f transit/keys/BUCKET_NAME exportable=true
```

例

```
[root@vault ~]# vault write -f transit/keys/mybucketkey exportable=true
```

注記

デフォルトでは、上記のコマンドは **aes256-gcm96** タイプキーを作成します。

3. キーの作成を確認します。

構文

```
vault read transit/export/encryption-key/BUCKET_NAME/VERSION_NUMBER
```

例

```
[root@vault ~]# vault read transit/export/encryption-key/mybucketkey/1

Key     Value
---     -----
keys    map[1:-gbTI9lNpqv/V/2lDcmH2Nq1xKn6FPDWarCmFM2aNsQ=]
name    mybucketkey
type    aes256-gcm96
```

注記

キーバージョンを含む完全なキーパスを指定する必要があります。

2.12.8. AWS および Vault を使用したオブジェクトのアップロード

Ceph Object Gateway にオブジェクトをアップロードする際に、Ceph Object Gateway は Vault からキーを取得し、そのオブジェクトをバケットに暗号化して保存します。オブジェクトのダウンロード要求が行われると、Ceph Object Gateway は Vault から対応するキーを自動的に取得し、オブジェクトを復号します。

注記

URL は、**rgw_crypt_vault_addr** オプションと、**rgw_crypt_vault_prefix** オプションで設定されたパス接頭辞を使用して構築されます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph Object Gateway ソフトウェアのインストール。
- HashiCorp Vault ソフトウェアのインストール。
- Ceph Object Gateway クライアントノードへのアクセス。
- Amazon Web Services (AWS) へのアクセス。

手順

1. AWS コマンドラインクライアントを使用して、オブジェクトをアップロードします。

   例

   ```
   [user@client ~]$ aws --endpoint=http://radosgw:8000 s3 cp plaintext.txt
   s3://mybucket/encrypted.txt --sse=aws:kms --sse-kms-key-id myproject/mybucketkey
   ```

注記

例で使用するキーフェッチ URL は **http://vault-server:8200/v1/secret/data/myproject/mybucketkey** です。

2.12.9. 関連情報

- 詳細は、Vault のプロジェクトサイトにある Install Vault ドキュメントを参照してください。

2.13. 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。

2.13.1. 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"
}
```

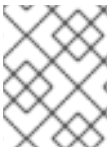
2. 出力を確認し、**access_key** および **secret_key** の値に JSON エスケープ文字 (\) が含まれていないことを確認します。これらの値はアクセスの検証に必要ですが、値に JSON エスケープ文字が含まれる場合、特定のクライアントは処理できません。この問題を修正するには、以下のアクションのいずれかを実行します。

   - 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。

2.13.2. 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

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

手順

1. Swift ユーザーを作成します。

   構文

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

   NAME を Swift ユーザー名に置き換えます。以下に例を示します。

   例

   ```
   [root@rgw]# radosgw-admin subuser create --uid=testuser --subuser=testuser:swift --
   access=full
   {
       "user_id": "testuser",
       "display_name": "First User",
       "email": "",
   ```

```
    "suspended": 0,
    "max_buckets": 1000,
    "auid": 0,
    "subusers": [
        {
            "id": "testuser:swift",
            "permissions": "full-control"
        }
    ],
    "keys": [
        {
            "user": "testuser",
            "access_key": "O8JDE41XMI74O185EHKD",
            "secret_key": "i4Au2yxG5wtr1JK01mI8kjJPM93HNAoVWOSTdJd6"
        }
    ],
    "swift_keys": [
        {
            "user": "testuser:swift",
            "secret_key": "13TLtdEW7bCqgttQgPzxFxziu0AgabtOc6vM8DLA"
        }
    ],
    "caps": [],
    "op_mask": "read, write, delete",
    "default_placement": "",
    "placement_tags": [],
    "bucket_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
    "user_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
    "temp_url_keys": [],
    "type": "rgw"
}
```


2. シークレットキーを作成します。

   構文

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

   NAME を Swift ユーザー名に置き換えます。以下に例を示します。

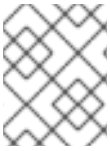
   例

   ```
   [root@rgw]# radosgw-admin key create --subuser=testuser:swift --key-type=swift --gen-
   ```

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

2.13.3. 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。



注記

出力にはメタデータを維持するための追加の json フィールドが含まれるため、システムユーザーはゾーン全体に対する root 権限を持っている必要があります。

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。
- S3 ユーザーが作成されました。

手順

1. Red Hat Enterprise Linux 7 の共通リポジトリーと RedHat Enterprise Linux 8 の高可用性リポジトリーを有効にします。

   Red Hat Enterprise Linux 7

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

   Red Hat Enterprise Linux 8

   ```
   # subscription-manager repos --enable=rhel-8-for-x86_64-highavailability-rpms
   ```

2. **python-boto** パッケージをインストールします。

   Red Hat Enterprise Linux 7

   ```
   # yum install python-boto
   ```

   Red Hat Enterprise Linux 8

   ```
   # dnf install python3-boto3
   ```

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。

   Red Hat Enterprise Linux 7

   ```
   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,
)
```

Red Hat Enterprise Linux 8

```
import boto3

endpoint = "" # enter the endpoint URL along with the port "http://URL:_PORT_"

access_key = 'ACCESS'
secret_key = 'SECRET'

s3 = boto3.client(
        's3',
        endpoint_url=endpoint,
        aws_access_key_id=access_key,
        aws_secret_access_key=secret_key
        )

s3.create_bucket(Bucket='my-new-bucket')

response = s3.list_buckets()
for bucket in response['Buckets']:
    print("{name}\t{created}".format(
  name = bucket['Name'],
  created = bucket['CreationDate']
))
```

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** および **SECRET** は、Red Hat Ceph Storage Object Gateway 設定および管理ガイドの S3 ユーザーの作成 セクションの **access_key** および **secret_key** の値に置き換えます。

5. スクリプトを実行します。

Red Hat Enterprise Linux 7

```
python s3test.py
```

Red Hat Enterprise Linux 8

```
python3 s3test.py
```

出力例:

```
my-new-bucket 2021-08-16T17:09:10.000Z
```

2.13.4. 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+fz2gSqoHwR3lYtSbIyomyPHf3i7rgSJrF/IA' list
```

出力は以下のようになります。

```
my-new-bucket
```

2.14. 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 Ceph Object Gateway インスタンスには独自の IP アドレスがあるため、HAProxy および **keepalived** を使用して、Ceph Object Gateway サーバー全体で負荷のバランスを取ることができます。

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。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。

注記

本セクションでは、Red Hat Enterprise Linux 7 向けの HAProxy および **keepalived** の設定を説明します。

Red Hat Enterprise Linux 8 の場合は、**keepalived** パッケージおよび **haproxy** パッケージをインストールして、ロードバランサーをインストールします。How do we need any additional subscription for Load Balancing on Red Hat Enterprise Linux 8? を参照してください。詳細については、ナレッジベースアーティクルを参照してください。

2.14.1. HAProxy/keepalived の前提条件

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

- 稼働中の Ceph クラスター
- 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。

2.14.2. 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. Red Hat Enterprise Linux 7 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ーを有効にします。

```
[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
```

8. 必要なポートに接続します。

```
[root@haproxy]# semanage port -m -t http_cache_port_t -p tcp 8081
```

2.14.3. 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。

```
vrrp_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。

2.14.4. 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
    bind *: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
```

2.14.5. 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。

```
[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。

2.15. 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、サーブレット、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。

2.15.1. 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。

HAProxy/keepalived の詳細は、HAProxy/keepalived の設定 を参照してください。

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**。

2.15.2. 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。

2.15.3. 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**。

2.15.4. 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、**プロキシー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 に固有のものです。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
```

2.15.5. 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。

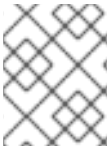
4. ファイルにパーミッションを設定し、それらが一般に公開されるようにします。

2.16. NFS-GANESHA への名前空間のエクスポート

Red Hat Ceph Storage 3 以降では、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。



注記

Red Hat Ceph Storage は、バージョン付けされたバケットの NFS エクスポートをサポートしません。

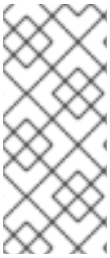
この実装は、UNIX 形式のパス名を S3 バケットおよびオブジェクトにマッピングする Amazon Web Services (AWS) 階層名前空間の規則に準拠しています。アタッチされた名前空間のトップレベルは、存在する場合は NFSv4 疑似 root に従属し、Ceph Object Gateway S3 バケットで設定されます。バケットは、NFS ディレクトリーとして表されます。バケット内のオブジェクトは、S3 規則に従って、NFS ファイルおよびディレクトリー階層として表示されます。ファイルおよびディレクトリーを作成する操作はサポートされています。

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. **root** で Red Hat Ceph Storage Tools リポジトリーを有効にします。

   Red Hat Enterprise Linux 7

   ```
   # subscription-manager repos --enable=rhel-7-server-rhceph-4-tools-rpms
   ```

   Red Hat Enterprise Linux 8

   ```
   # subscription-manager repos --enable=rhceph-4-tools-for-rhel-8-x86_64-rpms
   ```

3. **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。

4. **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。

   ```
   # yum 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 コンテナー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 コンテナー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、以下のようになります。

```
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 サーバーのレルム内にのみ存在し、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_Numberic_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。

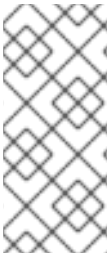
- NFS 4.1 以降のプロトコルフレーバーのみがサポートされます。

- 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
- バケット管理
- バケットライフサイクル
- Ceph Object Gateway データレイアウト
- Object Gateway データレイアウトパラメーター
- STS での属性ベースのアクセス制御 (ABAC) のセッションタグ
- Ceph Object Gateway のガベージコレクションの最適化
- Ceph Object Gateway のデータオブジェクトストレージの最適化
- Ceph Object Gateway およびマルチファクター認証
- Ansible を使用した Ceph Object Gateway の削除

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.root**
- **.default.rgw.control**
- **.default.rgw.meta**
- **.default.rgw.log**
- **.default.rgw.buckets.index**
- **.default.rgw.buckets.data**

- **.default.rgw.buckets.non-ec**

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 4 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方法。詳細は、Red Hat Ceph Storage 4 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**)。

   構文

   ```
   [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": 11
 }],
 "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
```

8. 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 エントリーは **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。

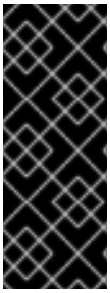
```
$ curl -i http://10.0.0.1/swift/v1/TestContainer/file.txt -X PUT -H "X-Storage-Policy: special-placement"
-H "X-Auth-Token: AUTH_rgwtxxxxxx"
```

3.3. インデックスレスバケットの作成

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。

警告

配置ポリシーにインデックスがないバケットで操作を実行すると、Ceph Object Gateway デーモンがクラッシュすることが確認されています。したがって、Red Hat はこの配置ポリシーを使用することを推奨していません。



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、Red Hat は、インデックスレスバケットで S3 オブジェクト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 から一覧表示できません。

前提条件

- 実行中で正常な Red Hat Ceph Storage クラスター
- Ceph Object Gateway ソフトウェアのインストール。
- Ceph Object Gateway ノードへのルートレベルのアクセス。

手順

1. 新しい配置ターゲットをゾーングループに追加します。

   例

   ```
   [root@rgw ~]# radosgw-admin zonegroup placement add --rgw-zonegroup="default" \
     --placement-id="indexless-placement"
   ```

2. 新しい配置ターゲットをゾーンに追加します。

   例

   ```
   [root@rgw ~]# radosgw-admin zone placement add --rgw-zone="default" \
     --placement-id="indexless-placement" \
     --data-pool="default.rgw.buckets.data" \
     --index-pool="default.rgw.buckets.index" \
     --data_extra_pool="default.rgw.buckets.non-ec" \
     --placement-index-type="indexless"
   ```

3. ゾーングループのデフォルト配置を **indexless-placement** に設定します。

   例

   ```
   [root@rgw ~]# radosgw-admin zonegroup placement default --placement-id "indexless-placement"
   ```

   この例では、**indexless-placement** 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。

4. クラスター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。

   例

   ```
   [root@rgw ~]# radosgw-admin period update --commit
   ```

5. Ceph Object Gateway デーモンを再起動して、変更を適用します。

例

```
[root@rgw ~]# systemctl restart ceph-radosgw.target
```

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オータを設定せずに、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。Red Hat Ceph Storage 4.1 以降、バケットインデックスシャードのデフォルト数である **bucket_index_max_shards** が 1 から 11 に変更されました。この変更により、小さなバケットへの書き込みスループットが増加し、動的な再シャーディングの開始を遅らせることができます。この変更は、新規バケットおよびデプロイメントにのみ影響します。

Red Hat では、計算されたシャード数に最も近い素数をシャード数とすることを推奨しています。素数であるバケットインデックスシャードは、シャード間でバケットインデックスエントリーを均等に分配する上で、より効果的に機能する傾向があります。たとえば、7001 個のバケットインデックスシャードは素数であるため、7000 個のシャードよりも優れています。

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。
- 手動の場合:「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手動の場合:**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。

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。

シャード化が必要になる前の 1 つのバケット内のオブジェクトの最大数

Red Hat では、バケットインデックスシャードあたり最大 102,400 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。

注記

Ceph OSD は現在、インデックス化されたストレージのキー範囲が 200,000 を超えると警告します。結果として、シャードあたりのオブジェクト数が 200,000 に近づくと、そのような警告が表示されます。設定によっては、値は大きくなる可能性があり、調整可能です。

シャード化を使用する場合の最大オブジェクト数

動的バケットの再シャーディングのデフォルトのバケットインデックスシャード数は 1999 です。この値は最大 65521 シャードまで変更できます。値が 1999 のバケットインデックスシャードは、バケット内の合計オブジェクト数が 204697600 で、値が 65521 のシャードは、オブジェクト数が 6709350400 であることが分かります。

注記

以前のテストに基づいて、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最大数は 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ーテストを実施していません。

重要

バケットインデックスシャードの数が 1999 を超える場合には、通常の S3 クライアントはバケットの内容を一覧表示できない可能性があります。カスタムクライアントは、順序付けられていないリストを要求して任意の数のシャードにスケーリングできます。

3.4.2. バケットのライフサイクルの並列スレッド処理

Red Hat Ceph Storage 4.1 の新機能により、バケットライフサイクルの並列スレッド処理が可能になりました。この並列化は、Ceph Object Gateway インスタンスの数に応じてスケーリングされ、順序が適切なインデックスシャードの列挙を数列に置き換えます。デフォルトのロックタイムアウトは 60 秒から 90 秒に拡張されました。各 Ceph Object Gateway インスタンスに対して並行して実行できるように、ライフサイクルワーカースレッドを調整するように、新たな調整可能なオプションが追加されました。

rgw_lc_max_worker

このオプションは、並行して実行するライフサイクルワーカースレッドの数を指定し、バケットおよびインデックスシャードを同時に処理します。**rgw_lc_max_worker** オプションのデフォルト値は **3** です。

rgw_lc_max_wp_worker

このオプションは、各ライフサイクルワーカーのワークプールのスレッド数を指定します。このオプションを使用すると、各バケットの処理を加速することができます。**rgw_lc_max_wp_worker** オプションのデフォルト値は **3** です。

関連情報

- 詳細は、**Red Hat Ceph Storage 設定ガイド**の Ceph 設定ファイル セクションを参照してください。

3.4.3. 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

- バケットシャード化の制限 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 = 12
```

   - 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。

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

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**

3.4.4. 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。

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ディングを無効にするには **0**、**bucket_index_max_shards** のデフォルト値は **11** です。
- **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

- バケットシャード化の制限 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. 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 = 12
```

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
```

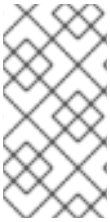
関連情報

- Manually Resharding Buckets with Multisite

3.4.5. バケットインデックスの動的再シャーディング

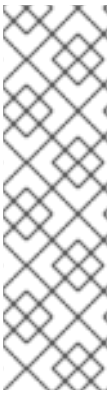
動的バケットの再シャーディングのプロセスは、すべての Ceph Object Gateway バケットを定期的に

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。



注記

デフォルト値は、回転するディスクに保存されているバケットインデックスのエクスペリエンスに基づいています。フラッシュメディアでバケットインデックスを使用する最新のセットアップでは、バケットインデックスシャードあたりの最大オブジェクト数の値が高くなる場合があります。



注記

アップグレードされたシングルサイト設定では、ゾーンやゾーングループに変更を加えることなく、バケットインデックスの動的再シャーディングが期待どおりに機能します。シングルサイト設定は、以下のいずれかです。

- レルムのないデフォルトのゾーン設定。
- レルムが 1 つ以上あるデフォルト以外の設定。
- 複数レルムのシングルサイト設定。

前提条件

- バケットシャード化の制限 を読んでいる。

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 **360** 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 --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

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
```

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

以下に例を示します。

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

- 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
```

以下を置き換えます。

- **bucket** を、保留中のバケットの名前に置き換えます。

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。

関連情報

- **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

3.4.6. 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シャードバケットを再シャードするには、**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

- バケットシャード化の制限 を読んでいる。

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
```

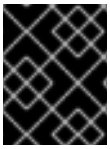
3. Red Hat Ceph Storage 3.1 およびそれ以前のバージョンを使用している場合は、**再シャーディング後の古いインスタンスのクリーニング** のセクションで説明されているように、古いバケットエントリーを削除します。

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ド化の設定**
- **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**

3.4.7. 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してください。

前提条件

- Ceph Object Gateway がインストールされている。

手順

1. 古いインスタンスを一覧表示します。

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

2. 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。

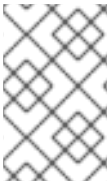
この圧縮設定は、この配置ターゲットを使用してバケットにアップロードされるすべての新規オブジェクトに適用されます。

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。

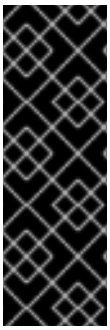
3.6. ユーザー管理

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。

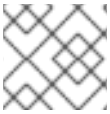
S3 用のユーザー **testx$tester** を作成するには、以下を実行します。

```
[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。

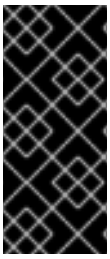
```
[root@master-zone]# radosgw-admin user create --uid=<id> \
```

```
[--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": []}
```

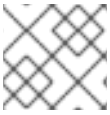


重要

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、**user info** ユーザー ID (**--uid={username}**) を指定します。

```
[root@master-zone]# radosgw-admin user info --uid=janedoe
```

テナントユーザーに関する情報を取得するには、ユーザー ID とテナントの名前の両方を指定します。

```
[root@master-zone]# radosgw-admin user info --uid=janedoe --tenant=test
```

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
```

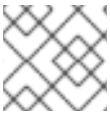
3.6.6. ユーザーの有効化および一時停止

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ジします。

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 クラスター。
- **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、ユーザー名とテナントの両方を指定します。

構文

```
radosgw-admin user rename --uid user-name --new-uid new-user-name --tenant tenant
```

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。

例

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

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。

   構文

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

   以下に例を示します。

   例

   ```
   # 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": "iaSFLDVvDdQt6lkNzHyW4fPLZugBAI1g17LO0+87"}],
  "swift_keys": [
   { "user": "johndoe:swift",
     "secret_key": "E9T2rUZNu2gxUjcwUBO8n\/Ev4KX6\/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)。
- **--geen-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)。
- **--geen-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/2EI4ngqJxD7iBdBYLhwluN30JaT3Q=="
        }
    ],
    ...
}
```

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. 管理機能の追加および削除

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、以下を実行します。

```
[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=*"
```

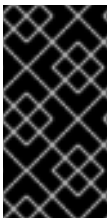
ユーザーから管理機能を削除するには、以下のコマンドを実行します。

```
[root@master-zone]# radosgw-admin caps remove --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。

構文

```
radosgw-admin quota set --uid=USER_ID --quota-scope=bucket --bucket=BUCKET_NAME [--max-objects=NUMBER_OF_OBJECTS] [--max-size=MAXIMUM_SIZE_IN_BYTES]
```

-

NUMBER_OF_OBJECTS、MAXIMUM_SIZE_IN_BYTES、またはその両方が負の値の場合、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、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=bucket --uid=<uid>
```

有効なバケット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=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>
```

テナントユーザーのクォータ設定を取得するには、ユーザー ID とテナントの名前を指定します。

```
+ radosgw-admin user info --uid=_user-id_ --tenant=_tenant_
```

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

レルム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リングできます (**形式: yyyy-mm-dd[HH:MM:SS]**)。
- **ログエントリー: --show-log-entries** オプションを使用すると、ログエントリー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ー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
```

3.9. 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。また、ストレージクラスターの存続期間中に発生する可能性のある孤立したオブジェクトやリークオブジェクトを Ceph Object Gateway 内で見つけることができます。

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 クラスターがある。
- 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$tuser2'
```

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 クラスターがある。
- 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. 孤立したオブジェクトやリークオブジェクトを見つける

正常なストレージクラスターには孤立したオブジェクトやリークオブジェクトがありませんが、場合によっては、孤立したオブジェクトや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、Ceph Object Gateway が操作の途中でダウンした場合、一部のオブジェクトが孤立する原因となる可能性があります。また、検出されないバグでも、孤立したオブジェクトが発生する可能性があります。

Red Hat Ceph Storage 4.1 以降、ストレージ管理者は Ceph Object Gateway オブジェクトが RADOS オブジェクトにマップする方法を確認できます。**radosgw-admin** コマンドは、これらの潜在的な孤立オブジェクトまたはリークオブジェクトの一覧を検索して生成するための新しいツールを提供します。**radoslist** サブコマンドを使用すると、バケット内またはストレージクラスター内のすべてのバケットに保存されているオブジェクトが表示されます。**rgw-orphan-list** スクリプトは、プール内の孤立したオブジェクトを表示します。

警告

rgw-orphan-list コマンドは、まだ実験段階にあります。**rados rm** command コマンドを使用して削除する前に、一覧表示されているオブジェクトを慎重かつ注意して評価してください。

重要

radoslist サブコマンドは、非推奨の **orphans find** サブコマンドおよび **orphans finish** サブコマンドを置き換えます。

前提条件

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

手順

1. バケット内でデータを保持するオブジェクトの一覧を生成するには、以下を実行します。

   構文

   ```
   radosgw-admin bucket radoslist --bucket BUCKET_NAME
   ```

   例

   ```
   [root@rgw ~]# radosgw-admin bucket radoslist --bucket mybucket
   ```

   注記

   BUCKET_NAME を省略すると、すべてのバケット内のすべてのオブジェクトが表示されます。

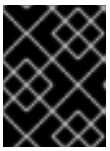
2. プールの孤立一覧を生成するには、以下を実行します。

   ```
   [root@rgw ~]# rgw-orphan-list
   ```

   例

   ```
   Available pools:
       .rgw.root
       default.rgw.control
       default.rgw.meta
       default.rgw.log
       default.rgw.buckets.index
       default.rgw.buckets.data
       rbd
       default.rgw.buckets.non-ec
       ma.rgw.control
       ma.rgw.meta
       ma.rgw.log
       ma.rgw.buckets.index
       ma.rgw.buckets.data
       ma.rgw.buckets.non-ec
   Which pool do you want to search for orphans?
   ```

   プール名を入力して、孤立を検索します。

   重要

   メタデータプールではなく、**rgw-orphan-list** コマンドを使用する場合は、データプールを指定する必要があります。

3. リスト内の孤立オブジェクトを確認します。

4. 孤立したオブジェクトを削除するには、以下を実行します。

構文

```
rados -p POOL_NAME rm OBJECT_NAME
```

例

```
[root@rgw ~]# rados -p default.rgw.buckets.data rm myobject
```

警告

正しいオブジェクトを削除していることを確認してください。**rados rm** コマンドを実行すると、ストレージクラスターからデータが削除されます。

関連情報

- 従来の **radosgw-admin orphans find** サブコマンドに関する詳細は、Red Hat Ceph Storage 3 Object Gateway Administration Guideの Finding Orphan Objects セクションを参照してください。

3.9.4. バケットインデックスエントリーの管理

radosgw-admin bucket check サブコマンドを使用して、Red Hat Ceph Storage クラスターで Ceph Object Gateway のバケットインデックスエントリーを管理できます。

マルチパートアップロードオブジェクトの一部に関連する各バケットインデックスエントリーは、対応する **.meta** インデックスエントリーと照合されます。特定のマルチパートアップロードのすべての部分に **.meta** エントリーが必要です。ピースに対応する **.meta** エントリーが見つからない場合、出力のセクションに孤立したエントリーが一覧表示されます。

バケットの統計はバケットインデックスヘッダーに保存されます。このフェーズでは、これらのヘッダーをロードし、バケットインデックスのすべてのプレーンオブジェクトエントリーを繰り返し処理し、統計を再計算します。次に、それぞれ existing_header と calculated_header というラベルの付いたセクションに実際の統計と計算した統計を表示して、比較できるようにします。

バケットチェック サブコマンドで **--fix** オプションを使用すると、孤立したエントリーがバケットインデックスから削除され、ヘッダー内の既存の統計が計算された統計で上書きされます。これにより、バージョン管理で使用される複数のエントリーを含むすべてのエントリーが出力の セクションに一覧表示されます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- 実行中の Ceph Object Gateway。
- バケットが作成されている。

手順

1. 特定のバケットのバケットインデックスを確認します。

   構文

   ```
   radosgw-admin bucket check --bucket=BUCKET_NAME
   ```

   例

   ```
   [root@rgw ~]# radosgw-admin bucket check --bucket=mybucket
   ```

2. 孤立したオブジェクトの削除など、バケットインデックスの不整合を修正します。

   構文

   ```
   radosgw-admin bucket check --fix --bucket=BUCKET_NAME
   ```

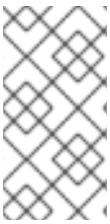
   例

   ```
   [root@rgw ~]# radosgw-admin bucket check --fix --bucket=mybucket
   ```

3.9.5. バケット通知

バケット通知により、バケットで特定のイベントが発生した場合に、Ceph Object Gateway から情報を送る方法が提供されます。バケット通知は HTTP、AMQP0.9.1、および Kafka エンドポイントに送信できます。

特定バケットおよび特定のトピック上のイベントのバケット通知を送信するために、通知エントリーを作成する必要があります。バケット通知は、イベントタイプのサブセットに作成することも、デフォルトですべてのイベントタイプに対して作成できます。バケット通知は、キーの接頭辞または接尾辞、キーに一致する正規表現、オブジェクトに割り当てられたメタデータ属性、またはオブジェクトタグに基づいてイベントをフィルタリングできます。バケット通知には、バケット通知メカニズムの設定および制御インターフェイスを提供する REST API があります。



注記

バケット通知 API はデフォルトで有効にされます。**rgw_enable_apis** 設定パラメーターが明示的に設定されている場合は、**s3** および **pubsub** が含まれていることを確認します。これを確認するには、**ceph config get mon.* rgw_enable_apis** コマンドを実行します。

関連情報

- バケット通知 REST API についての詳細は、Red Hat Ceph Storage 開発者ガイドを参照してください。

3.9.6. バケット通知の作成

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

前提条件

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

重要

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

手順

1. s3 バケットを作成します。
2. **http**、**amqp**、または **kafka** プロトコルに SNS トピックを作成します。
3. **s3:objectCreate** および **s3:objectRemove** イベントの s3 バケット通知を作成します。

   例

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

4. バケットに s3 オブジェクトを作成します。
5. レシーバー **http**、**rabbitmq**、または **kafka** でのオブジェクト作成イベントを確認します。
6. オブジェクトを削除します。
7. レシーバー **http**、**rabbitmq**、または **kafka** でオブジェクトの削除イベントを確認します。

3.9.7. 関連情報

- 詳細は、**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。
- 詳細は、**Red Hat Ceph Storage 開発者ガイド**を参照してください。

3.10. バケットライフサイクル

ストレージ管理者では、バケットのライフサイクル設定を使用してオブジェクトを管理し、そのオブジェクトが有効期間中効果的に保存されるようにすることができます。たとえば、オブジェクトを、ユースケースに基づいて、コストの低いストレージクラス、アーカイブ、または削除にできます。

注記

radosgw-admin lc reshard コマンドは Red Hat Ceph Storage 3.3 で非推奨となり、Red Hat Ceph Storage 4 以降のリリースではサポートされません。

3.10.1. ライフサイクル管理ポリシーの作成

radosgw-admin コマンドを使用する代わりに、標準の S3 操作を使用してバケットのライフサイクルポリシー設定を管理できます。RADOS Gateway は、バケットに適用される Amazon S3 API ポリシー言語のサブセットのみをサポートします。ライフサイクル設定には、バケットオブジェクトのセットに定義される 1 つまたは複数のルールが含まれます。

前提条件

- 稼働中の Red Hat Storage クラスター
- Ceph Object Gateway のインストール
- Ceph Object Gateway ノードへのルートレベルのアクセス。
- S3 バケットが作成されている。
- ユーザーアクセスで作成された S3 ユーザー。
- **AWS CLI** パッケージがインストールされた Ceph Object Gateway クライアントへのアクセス。

手順

1. ライフサイクル設定用の JSON ファイルを作成します。

   例

   ```
   [user@client ~]$ vi lifecycle.json
   ```

2. ファイルに特定のライフサイクル設定ルールを追加します。

   例

   ```
   {
    "Rules": [
           {
        "Filter": {
         "Prefix": "images/"
        },
        "Status": "Enabled",
        "Expiration": {
         "Days": 1
        },
        "ID": "ImageExpiration"
   ```

```
    }
   ]
}
```

ライフサイクル設定の例では、1日以上経過した images ディレクトリーのオブジェクトの有効期限が切れます。

3. バケットにライフサイクル設定を設定します。

構文

```
aws --endpoint-url=RADOSGW_ENDPOINT_URL:PORT s3api put-bucket-lifecycle-configuration --bucket BUCKET_NAME --lifecycle-configuration file://PATH_TO_LIFECYCLE_CONFIGURATION_FILE/LIFECYCLE_CONFIGURATION_FILE.json
```

例

```
[user@client ~]$ aws --endpoint-url=http://host01:80 s3api put-bucket-lifecycle-configuration --bucket testbucket --lifecycle-configuration file://lifecycle.json
```

この例では、**lifecycle.json** ファイルが現在のディレクトリーに存在します。

検証

- バケットのライフサイクル設定を取得します。

構文

```
aws --endpoint-url=RADOSGW_ENDPOINT_URL:PORT s3api get-bucket-lifecycle-configuration --bucket BUCKET_NAME
```

例

```
[user@client ~]$ aws --endpoint-url=http://host01:80 s3api get-bucket-lifecycle-configuration --bucket testbucket
{
 "Rules": [
        {
      "Expiration": {
       "Days": 1
      },
      "ID": "ImageExpiration",
      "Filter": {
       "Prefix": "images/"
      },
      "Status": "Enabled"
     }
    ]
}
```

- オプション:Ceph Object Gateway ノードから、Cephadm シェルにログインし、バケットのライフサイクル設定を取得します。

構文

```
radosgw-admin lc get --bucket=BUCKET_NAME
```

例

```
[root@rgw ~]# radosgw-admin lc get --bucket=testbucket
{
 "prefix_map": {
  "images/": {
   "status": true,
   "dm_expiration": false,
   "expiration": 1,
   "noncur_expiration": 0,
   "mp_expiration": 0,
   "transitions": {},
   "noncur_transitions": {}
  }
 },
 "rule_map": [
        {
  "id": "ImageExpiration",
  "rule": {
   "id": "ImageExpiration",
   "prefix": "",
   "status": "Enabled",
   "expiration": {
    "days": "1",
    "date": ""
   },
   "mp_expiration": {
    "days": "",
    "date": ""
   },
   "filter": {
    "prefix": "images/",
    "obj_tags": {
     "tagset": {}
    }
   },
   "transitions": {},
   "noncur_transitions": {},
   "dm_expiration": false
  }
 }
  ]
}
```

関連情報

- 詳細は、**Red Hat Ceph Storage 開発者ガイド**の **S3 バケットライフサイクル** セクションを参照してください。

- **AWS CLI** を使用したライフサイクル設定の管理に関する詳細は、**Amazon Simple Storage Service** ドキュメントの Setting lifecycle configuration on a bucket セクションを参照してください。

3.10.2. ライフサイクル管理ポリシーの削除

s3api delete-bucket-lifecycle コマンドを使用すると、指定されたバケットのライフサイクル管理ポリシーを削除できます。

前提条件

- 稼働中の Red Hat Storage クラスター
- Ceph Object Gateway のインストール
- Ceph Object Gateway ノードへのルートレベルのアクセス。
- S3 バケットが作成されている。
- ユーザーアクセスで作成された S3 ユーザー。
- **AWS CLI** パッケージがインストールされた Ceph Object Gateway クライアントへのアクセス。

手順

- ライフサイクル設定を削除します。

  構文

  ```
  aws --endpoint-url=RADOSGW_ENDPOINT_URL:PORT s3api delete-bucket-lifecycle --bucket BUCKET_NAME
  ```

  例

  ```
  [user@client ~]$ aws --endpoint-url=http://host01:80 s3api delete-bucket-lifecycle --bucket testbucket
  ```

検証

- バケットのライフサイクル設定を取得します。

  構文

  ```
  aws --endpoint-url=RADOSGW_ENDPOINT_URL:PORT s3api get-bucket-lifecycle-configuration --bucket BUCKET_NAME
  ```

  例

  ```
  [user@client ~]# aws --endpoint-url=http://host01:80  s3api get-bucket-lifecycle-configuration --bucket testbucket
  ```

- オプション:Ceph Object Gateway ノードから、バケットのライフサイクル設定を取得します。

構文

```
radosgw-admin lc get --bucket=BUCKET_NAME
```

例

```
[root@rgw ~]# radosgw-admin lc get --bucket=testbucket
```

注記

バケットのライフサイクルポリシーが存在しない場合、このコマンドは情報を返しません。

関連情報

- 詳細は、**Red Hat Ceph Storage 開発者ガイド**の S3 バケットライフサイクル セクションを参照してください。

3.10.3. ライフサイクル管理ポリシーの更新

s3cmd put-bucket-lifecycle-configuration コマンドを使用すると、ライフサイクル管理ポリシーを更新できます。

注記

put-bucket-lifecycle-configuration は、既存のバケットのライフサイクル設定を上書きします。現在のライフサイクルポリシー設定を保持する場合は、ライフサイクル設定ファイルに追加する必要があります。

前提条件

- 稼働中の Red Hat Storage クラスター
- Ceph Object Gateway のインストール
- Ceph Object Gateway ノードへのルートレベルのアクセス。
- S3 バケットが作成されている。
- ユーザーアクセスで作成された S3 ユーザー。
- **AWS CLI** パッケージがインストールされた Ceph Object Gateway クライアントへのアクセス。

手順

1. ライフサイクル設定用の JSON ファイルを作成します。

   例

   ```
   [user@client ~]$ vi lifecycle.json
   ```

2. ファイルに特定のライフサイクル設定ルールを追加します。

例

```
{
 "Rules": [
        {
      "Filter": {
       "Prefix": "images/"
      },
      "Status": "Enabled",
      "Expiration": {
       "Days": 1
      },
      "ID": "ImageExpiration"
     },
  {
   "Filter": {
    "Prefix": "docs/"
   },
   "Status": "Enabled",
   "Expiration": {
    "Days": 30
   },
   "ID": "DocsExpiration"
  }
 ]
}
```

3. バケットでライフサイクル設定を更新します。

構文

```
aws --endpoint-url=RADOSGW_ENDPOINT_URL:PORT s3api put-bucket-lifecycle-configuration --bucket BUCKET_NAME --lifecycle-configuration file://PATH_TO_LIFECYCLE_CONFIGURATION_FILE/LIFECYCLE_CONFIGURATION_FILE.json
```

例

```
[user@client ~]$ aws --endpoint-url=http://host01:80 s3api put-bucket-lifecycle-configuration --bucket testbucket --lifecycle-configuration file://lifecycle.json
```

検証

- バケットのライフサイクル設定を取得します。

構文

```
aws --endpointurl=RADOSGW_ENDPOINT_URL:PORT s3api get-bucket-lifecycle-configuration --bucket BUCKET_NAME
```

例

```
[user@client ~]$ aws -endpoint-url=http://host01:80 s3api get-bucket-lifecycle-configuration -
```

```
-bucket testbucket

{
    "Rules": [
        {
            "Expiration": {
                "Days": 30
            },
            "ID": "DocsExpiration",
            "Filter": {
                "Prefix": "docs/"
            },
            "Status": "Enabled"
        },
        {
            "Expiration": {
                "Days": 1
            },
            "ID": "ImageExpiration",
            "Filter": {
                "Prefix": "images/"
            },
            "Status": "Enabled"
        }
    ]
}
```

- オプション:Ceph Object Gateway ノードから、Cephadm シェルにログインし、バケットのライフサイクル設定を取得します。

  構文

  ```
  radosgw-admin lc get --bucket=BUCKET_NAME
  ```

  例

  ```
  [root@rgw ~]# radosgw-admin lc get --bucket=testbucket
  {
   "prefix_map": {
         "docs/": {
    "status": true,
    "dm_expiration": false,
    "expiration": 1,
    "noncur_expiration": 0,
    "mp_expiration": 0,
    "transitions": {},
    "noncur_transitions": {}
   },
   "images/": {
    "status": true,
    "dm_expiration": false,
    "expiration": 1,
    "noncur_expiration": 0,
    "mp_expiration": 0,
    "transitions": {},
  ```

```
  "noncur_transitions": {}
 }
},
"rule_map": [
      {
       "id": "DocsExpiration",
    "rule": {
     "id": "DocsExpiration",
     "prefix": "",
     "status": "Enabled",
     "expiration": {
      "days": "30",
      "date": ""
     },
          "noncur_expiration": {
              "days": "",
              "date": ""
          },
     "mp_expiration": {
      "days": "",
      "date": ""
     },
     "filter": {
      "prefix": "docs/",
      "obj_tags": {
       "tagset": {}
      }
     },
     "transitions": {},
     "noncur_transitions": {},
     "dm_expiration": false
    }
   },
   {
 "id": "ImageExpiration",
 "rule": {
  "id": "ImageExpiration",
  "prefix": "",
  "status": "Enabled",
  "expiration": {
   "days": "1",
   "date": ""
  },
  "mp_expiration": {
   "days": "",
   "date": ""
  },
  "filter": {
   "prefix": "images/",
   "obj_tags": {
    "tagset": {}
   }
  },
  "transitions": {},
  "noncur_transitions": {},
  "dm_expiration": false
```

```
    }
  }
   ]
}
```

関連情報

- Amazon S3 バケットライフサイクルに関する詳細は、**Red Hat Ceph Storage 開発者ガイド**の Amazon S3 バケットライフサイクル を参照してください。

3.10.4. バケットライフサイクルのモニタリング

ライフサイクル処理を監視することや、**radosgw-admin lc list** および **radosgw-admin lc process** コマンドを使用してバケットのライフサイクルを手動で処理することができます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph Object Gateway ノードへのルートレベルのアクセス。
- ライフサイクル設定ポリシーが適用される S3 バケットが作成されている。

手順

1. バケットのライフサイクルの進捗を一覧表示します。

   例

```
[root@rgw ~]# radosgw-admin lc list

[
  {
       “bucket”: “:testbucket:8b63d584-9ea1-4cf3-8443-a6a15beca943.54187.1”,
       “started”: “Thu, 01 Jan 1970 00:00:00 GMT”,
       “status” : “UNINITIAL”
  },
  {
       “bucket”: “:testbucket1:8b635499-9e41-4cf3-8443-a6a15345943.54187.2”,
       “started”: “Thu, 01 Jan 1970 00:00:00 GMT”,
       “status” : “UNINITIAL”
  }
]
```

   バケットのライフサイクル処理ステータスは、以下のいずれかになります。

   - UNINITIAL - プロセスはまだ実行されていません。
   - PROCESSING - プロセスは現在実行中です。
   - COMPLETE - プロセスが完了しました。

2. オプション: バケットのライフサイクルポリシーを手動で処理できます。

   a. 単一バケットのライフサイクルポリシーを処理します。

構文

```
radosgw-admin lc process --bucket=BUCKET_NAME
```

例

```
[root@rgw ~]# radosgw-admin lc process --bucket=testbucket1
```

b. すべてのバケットのライフサイクルポリシーを即座に処理します。

例

```
[root@rgw ~]# radosgw-admin lc process
```

検証

- バケットのライフサイクルポリシーを一覧表示します。

```
[root@rgw ~]# radosgw-admin lc list
[
    {
        “bucket”: “:testbucket:8b63d584-9ea1-4cf3-8443-a6a15beca943.54187.1”,
        “started”: “Thu, 17 Mar 2022 21:48:50 GMT”,
        “status” : “COMPLETE”
    }
    {
        “bucket”: “:testbucket1:8b635499-9e41-4cf3-8443-a6a15345943.54187.2”,
        “started”: “Thu, 17 Mar 2022 20:38:50 GMT”,
        “status” : “COMPLETE”
    }
]
```

関連情報

- 詳細は、**Red Hat Ceph Storage 開発者ガイド**の S3 バケットライフサイクル セクションを参照してください。

3.10.5. ライフサイクルの有効期間の設定

rgw_lifecycle_work_time パラメーターを設定すると、毎日ライフサイクル管理プロセスが実行される時間を設定できます。デフォルトでは、ライフサイクルの処理は 1 日 1 回午前 0 時に行われます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph Object Gateway のインストール
- Ceph Object Gateway ノードへのルートレベルのアクセス。

手順

1. ライフサイクルの有効期限を設定します。

構文

```
ceph config set client.rgw rgw_lifecycle_work_time %D:%D-%D:%D
```

%d:%d-%d:%d を **start_hour:start_minute-end_hour:end_minute** に置き換えます。

例

```
[root@rgw ~]# ceph config set client.rgw rgw_lifecycle_work_time 06:00-08:00
```

検証

- ライフサイクル満了の作業時間を取得します。

  例

```
[root@rgw ~]# ceph config get client.rgw rgw_lifecycle_work_time

06:00-08:00
```

関連情報

- 詳細は、**Red Hat Ceph Storage 開発者ガイド**の **S3 バケットライフサイクル** セクションを参照してください。

3.10.6. ストレージクラスター内での S3 バケットライフサイクルの移行

バケットライフサイクル設定を使用してオブジェクトを管理し、オブジェクトのライフタイム全体でオブジェクトを効果的に保存できます。オブジェクトライフサイクルの移行ルールを使用すると、オブジェクトのライフタイム全体でオブジェクトを管理し、効果的に保存できます。オブジェクトを、コストの低いストレージクラス、アーカイブ、または削除にできます。

以下についてストレージクラスを作成できます。

- I/O 機密ワークロード用の SSD や NVMe などの高速メディア
- アーカイブを行うため、SAS や SATA などの処理が遅いメディア。

ホットストレージクラスとコールドストレージクラスとの間で、データの移動をスケジュールできます。指定された時間後にこの移行をスケジュールして、オブジェクトの期限が切れ、永続的に削除されます。たとえば、オブジェクトを作成/からストレージクラスの作成後、またはストレージクラスを 1 年にアーカイブした後に、オブジェクトを 30 日間移行することができます。これは、移行ルールを使用して実行できます。このルールは、あるストレージクラスから別のストレージクラスに移行するオブジェクトに適用されます。ライフサイクル設定には、**<Rule>** 要素を使用した 1 つ以上のルールが含まれます。

関連情報

- バケットライフサイクル の詳細は、**Red Hat Ceph Storage 開発者ガイド**を参照してください。

3.10.7. あるストレージクラスから別のストレージクラスへのオブジェクトの移行

オブジェクトライフサイクルの移行ルールにより、オブジェクトをあるストレージクラスから別のストレージクラスに移行することができます。

前提条件

- Ceph Object Gateway ソフトウェアのインストール。
- Ceph Object Gateway ノードへのルートレベルのアクセス。
- ユーザーアクセスで作成された S3 ユーザー。

手順

1. 新しいデータプールを作成します。

   構文

   ```
   ceph osd pool create POOL_NAME
   ```

   例

   ```
   [root@rgw ~]# ceph osd pool create test.hot.data
   ```

2. 新規ストレージクラスを追加します。

   構文

   ```
   radosgw-admin zonegroup placement add  --rgw-zonegroup default --placement-id
   PLACEMENT_TARGET --storage-class STORAGE_CLASS
   ```

   例

   ```
   [root@rgw ~]# radosgw-admin zonegroup placement add  --rgw-zonegroup default --
   placement-id default-placement --storage-class hot.test
   {
           "key": "default-placement",
           "val": {
               "name": "default-placement",
               "tags": [],
               "storage_classes": [
                   "STANDARD",
                   "hot.test"
               ]
           }
       }
   ```

3. 新規ストレージクラスのゾーン配置情報を提供します。

   構文

   ```
   radosgw-admin zone placement add --rgw-zone default --placement-id
   PLACEMENT_TARGET --storage-class STORAGE_CLASS --data-pool DATA_POOL
   ```

例

```
[root@rgw ~]# radosgw-admin zone placement add --rgw-zone default --placement-id
default-placement --storage-class hot.test --data-pool test.hot.data
{
        "key": "default-placement",
        "val": {
            "index_pool": "test_zone.rgw.buckets.index",
            "storage_classes": {
                "STANDARD": {
                    "data_pool": "test.hot.data"
                },
                "hot.test": {
                    "data_pool": "test.hot.data",
                }
            },
            "data_extra_pool": "",
            "index_type": 0
        }
```

注記

一度書き込みでコールドまたはアーカイブデータストレージプールを作成する際には、**compression_type** を設定することを検討してください。

4. データプールの **rgw** アプリケーションを有効にします。

構文

```
ceph osd pool application enable POOL_NAME rgw
```

例

```
[root@rgw ~] ceph osd pool application enable test.hot.data rgw
enabled application 'rgw' on pool 'test.hot.data'
```

5. すべての **rgw** デーモンを再起動します。

6. バケットを作成します。

例

```
[root@rgw ~]# aws s3api create-bucket --bucket testbucket10 --create-bucket-configuration
LocationConstraint=default:default-placement --endpoint-url http://1x.7x.2xx.1xx:80
```

7. オブジェクトを追加します。

例

```
[root@rgw ~]# aws --endpoint=http://1x.7x.2xx.1xx:80 s3api put-object --bucket testbucket10
--key compliance-upload --body /root/test2.txt
```

8. 2 番目のデータプールを作成します。

構文

```
ceph osd pool create POOL_NAME
```

例

```
[root@rgw ~]# ceph osd pool create test.cold.data
```

9. 新規ストレージクラスを追加します。

構文

```
radosgw-admin zonegroup placement add  --rgw-zonegroup default --placement-id
PLACEMENT_TARGET --storage-class STORAGE_CLASS
```

例

```
[root@rgw ~]# radosgw-admin zonegroup placement add  --rgw-zonegroup default --
placement-id default-placement --storage-class cold.test
{
        "key": "default-placement",
        "val": {
            "name": "default-placement",
            "tags": [],
            "storage_classes": [
                "STANDARD",
                "cold.test"
            ]
        }
    }
```

10. 新規ストレージクラスのゾーン配置情報を提供します。

構文

```
radosgw-admin zone placement add --rgw-zone default --placement-id
PLACEMENT_TARGET --storage-class STORAGE_CLASS --data-pool DATA_POOL
```

例

```
[root@rgw ~]# radosgw-admin zone placement add --rgw-zone default --placement-id
default-placement --storage-class cold.test --data-pool test.cold.data
```

11. データプールの **rgw** アプリケーションを有効にします。

構文

```
ceph osd pool application enable POOL_NAME rgw
```

例

```
[root@rgw ~] ceph osd pool application enable test.cold.data rgw
enabled application 'rgw' on pool 'test.cold.data'
```

12. すべての **rgw** デーモンを再起動します。

13. ゾーングループ設定を表示するには、次のコマンドを実行します。

構文

```
radosgw-admin zonegroup get
{
    "id": "3019de59-ddde-4c5c-b532-7cdd29de09a1",
    "name": "default",
    "api_name": "default",
    "is_master": "true",
    "endpoints": [],
    "hostnames": [],
    "hostnames_s3website": [],
    "master_zone": "adacbe1b-02b4-41b8-b11d-0d505b442ed4",
    "zones": [
        {
            "id": "adacbe1b-02b4-41b8-b11d-0d505b442ed4",
            "name": "default",
            "endpoints": [],
            "log_meta": "false",
            "log_data": "false",
            "bucket_index_max_shards": 11,
            "read_only": "false",
            "tier_type": "",
            "sync_from_all": "true",
            "sync_from": [],
            "redirect_zone": ""
        }
    ],
    "placement_targets": [
        {
            "name": "default-placement",
            "tags": [],
            "storage_classes": [
                "hot.test",
                "cold.test",
                "STANDARD"
            ]
        }
    ],
    "default_placement": "default-placement",
    "realm_id": "",
    "sync_policy": {
        "groups": []
    }
}
```

14. ゾーン設定を表示するには、以下を実行します。

構文

```
radosgw-admin zone get
{
    "id": "adacbe1b-02b4-41b8-b11d-0d505b442ed4",
    "name": "default",
    "domain_root": "default.rgw.meta:root",
    "control_pool": "default.rgw.control",
    "gc_pool": "default.rgw.log:gc",
    "lc_pool": "default.rgw.log:lc",
    "log_pool": "default.rgw.log",
    "intent_log_pool": "default.rgw.log:intent",
    "usage_log_pool": "default.rgw.log:usage",
    "roles_pool": "default.rgw.meta:roles",
    "reshard_pool": "default.rgw.log:reshard",
    "user_keys_pool": "default.rgw.meta:users.keys",
    "user_email_pool": "default.rgw.meta:users.email",
    "user_swift_pool": "default.rgw.meta:users.swift",
    "user_uid_pool": "default.rgw.meta:users.uid",
    "otp_pool": "default.rgw.otp",
    "system_key": {
        "access_key": "",
        "secret_key": ""
    },
    "placement_pools": [
        {
            "key": "default-placement",
            "val": {
                "index_pool": "default.rgw.buckets.index",
                "storage_classes": {
                    "cold.test": {
                        "data_pool": "test.cold.data"
                    },
                    "hot.test": {
                        "data_pool": "test.hot.data"
                    },
                    "STANDARD": {
                        "data_pool": "default.rgw.buckets.data"
                    }
                },
                "data_extra_pool": "default.rgw.buckets.non-ec",
                "index_type": 0
            }
        }
    ],
    "realm_id": "",
    "notif_pool": "default.rgw.log:notif"
}
```

15. バケットを作成します。

例

```
[root@rgw ~]# aws s3api create-bucket --bucket testbucket10 --create-bucket-configuration
LocationConstraint=default:default-placement --endpoint-url http://1x.7x.2xx.1xx:80
```

16. ライフサイクル設定用の JSON ファイルを作成します。

例

```
[root@rgw ~]# vi lifecycle.json
```

17. ファイルに特定のライフサイクル設定ルールを追加します。

例

```
{
    "Rules": [
        {
            "Filter": {
                "Prefix": ""
            },
            "Status": "Enabled",
            "Transitions": [
                {
                    "Days": 5,
                    "StorageClass": "hot.test"
                },
 {
                    "Days": 20,
                    "StorageClass": "cold.test"
                }
            ],
            "Expiration": {
                "Days": 365
            },
            "ID": "double transition and expiration"
        }
    ]
}
```

ライフサイクル設定の例では、デフォルトの **STANDARD** ストレージクラスから 5 日後に **hot.test** ストレージクラスに移行し、20 日後に再び **cold.test** ストレージクラスに移行し、最後に **cold.test** ストレージクラスで 365 日後に期限切れとなるオブジェクトを示しています。

18. バケットにライフサイクル設定を設定します。

例

```
[root@rgw ~] aws s3api put-bucket-lifecycle-configuration --bucket testbucket20 --lifecycle-configuration file://lifecycle.json
```

19. バケットでライフサイクル設定を取得します。

例

```
[root@rgw ~]aws s3api get-bucket-lifecycle-configuration --bucket testbucket20
{
    "Rules": [
        {
            "Expiration": {
                "Days": 365
```

```
            },
            "ID": "double transition and expiration",
            "Prefix": "",
            "Status": "Enabled",
            "Transitions": [
                {
                    "Days": 20,
                    "StorageClass": "cold.test"
                },
                {
                    "Days": 5,
                    "StorageClass": "hot.test"
                }
            ]
        }
    ]
}
```

関連情報

- バケットライフサイクル の詳細は、**Red Hat Ceph Storage 開発者ガイド**を参照してください。

3.11. CEPH OBJECT GATEWAY データレイアウト

RADOS は拡張属性 (**xattrs**) とオブジェクトマップ (OMAP) を持つプールとオブジェクトしか認識しませんが、概念的には、Ceph Object Gateway はそのデータを 3 つの異なる種類に編成します。

- metadata
- バケットインデックス
- data

メタデータ

メタデータには 3 つのセクションがあります。

- **user**: ユーザー情報を保持します。
- **bucket**: バケット名とバケットインスタンス ID の間のマッピングを保持します。
- **bucket.instance**: バケットインスタンス情報を保持します。

以下のコマンドを使用して、メタデータエントリーを表示できます。

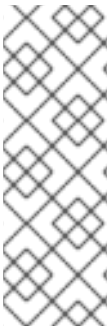
構文

```
radosgw-admin metadata get bucket:BUCKET_NAME
radosgw-admin metadata get bucket.instance:BUCKET:BUCKET_ID
radosgw-admin metadata get user:USER
radosgw-admin metadata set user:USER
```

例

```
[root@host01 ~]# radosgw-admin metadata list
[root@host01 ~]# radosgw-admin metadata list bucket
[root@host01 ~]# radosgw-admin metadata list bucket.instance
[root@host01 ~]# radosgw-admin metadata list user
```

すべてのメタデータエントリーは単一の RADOS オブジェクトに保存されます。



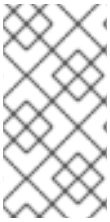
注記

Ceph Object Gateway オブジェクトは、いくつかの RADOS オブジェクトで設定される場合があります。最初のオブジェクトは、マニフェスト、アクセス制御リスト (ACL)、コンテンツタイプ、ETag、およびユーザー定義のメタデータなどのメタデータなどの Head です。メタデータは **xattrs** に保存されます。また、効率性とアトミック性を確保するために、最大 512 KB のオブジェクトデータが含まれる場合もあります。マニフェストには、各オブジェクトが RADOS オブジェクトでどのように配置されているかが記述されています。

バケットインデックス

これは別の種類のメタデータであり、個別に保持されます。バケットインデックスは RADOS オブジェクトのキーと値マップを保持します。デフォルトでは、バケットごとに 1 つの RADOS オブジェクトですが、複数の RADOS オブジェクトにわたってマップをシャードすることができます。

マップ自体は、各 RADOS オブジェクトに関連付けられた OMAP に保持されます。各 OMAP のキーはオブジェクトの名前であり、値にはそのオブジェクトの基本的なメタデータ (バケットを一覧表示するときに表示されるメタデータ) が保持されます。各 OMAP はヘッダーを保持し、オブジェクトの数、合計サイズなど、そのヘッダーにバケットアカウンティングメタデータを保持します。



注記

OMAP は、拡張属性が POSIX ファイルに関連付けられているのと同様に、オブジェクトに関連付けられたキー値ストアです。オブジェクトの OMAP はオブジェクトのストレージに物理的に配置されていませんが、その正確な実装は不可視であり、Ceph Object Gateway には重要ではありません。

データ

オブジェクトデータは、各 Ceph Object Gateway オブジェクトの 1 つ以上の RADOS オブジェクトに保持されます。

3.11.1. オブジェクトルックアップパス

オブジェクトにアクセスする場合、REST API は 3 つのパラメーターを持つ Ceph Object Gateway に送られます。

- S3 のアクセスキーまたは Swift のアカウント名を持つアカウント情報
- バケットまたはコンテナー名
- オブジェクト名またはキー

現在、Ceph Object Gateway はアカウント情報のみを使用してユーザー ID とアクセス制御を検索します。バケット名とオブジェクトキーのみを使用してプールのオブジェクトに対応します。

アカウント情報

Ceph Object Gateway のユーザー ID は文字列であり、通常はユーザー認証情報からの実際のユーザー名であり、ハッシュまたはマップされた識別子ではありません。

ユーザーのデータにアクセスする場合、ユーザーレコードは namespace が **users.uid** の **default.rgw.meta** プールのオブジェクト **USER_ID** から読み込まれます。

バケット名

これらは、**root** namespace で **default.rgw.meta** プールに表示されます。バケット ID として機能するマーカーを取得するために、バケットレコードがロードされます。

オブジェクト名

オブジェクトは **default.rgw.buckets.data** プールにあります。オブジェクト名は **MARKER_KEY** です。たとえば、**default.7593.4_image.png** の場合、マーカーは **default.7593.4** で、キーは **image.png** です。これらの連結された名前は解析されず、RADOS のみに渡されます。したがって、セパレーターの選択は重要ではなく、あいまいさを生じさせることはありません。同じ理由で、キーなどのオブジェクト名ではスラッシュを使用できます。

3.11.1.1. 複数のデータプール

複数のデータプールを作成して、異なるユーザーのバケットがデフォルトで異なる RADOS プールに作成されるようにすることで、必要なスケーリングを実現できます。これらのプールのレイアウトと命名は、**ポリシー** 設定によって制御されます。

3.11.2. バケットおよびオブジェクトの一覧

特定のユーザーに属するバケットは、namespace が **users.uid** の **default.rgw.meta** プール内の **foo.buckets** など、**USER_ID.buckets** という名前のオブジェクトの OMAP にリストされます。これらのオブジェクトには、バケットの一覧表示時、バケットの内容の更新時、およびクォータなどのバケット統計の更新および取得時にアクセスされます。これらのリストは、**.rgw** プール内のバケットと一貫性が保たれています。

注記

これらの OMAP エントリーの値は、ユーザーに表示されるエンコードされたクラス **cls_user_bucket_entry** およびネストされたクラス **cls_user_bucket** を参照してください。

指定のバケットに属するオブジェクトはバケットインデックスに一覧表示されます。インデックスオブジェクトのデフォルトの命名は、**default.rgw.buckets.index** プールの **.dir.MARKER** です。

関連情報

- 詳細は、**Red Hat Ceph Storage Object Gateway ガイド**の **バケットシャーディングの設定** のセクションを参照してください。

3.12. OBJECT GATEWAY データレイアウトパラメーター

これは、Ceph Object Gateway のデータレイアウトパラメーターの一覧です。

既知のプール:

.rgw.root

オブジェクトごとに 1 つの、指定されていない地域、ゾーン、およびグローバル情報レコード。

ZONE.rgw.control

notify.**N**

ZONE.rgw.meta

さまざまな種類のメタデータを持つ複数の namespace

Namespace: root

BUCKET .bucket.meta.**BUCKET**:**MARKER** # see put_bucket_instance_info()
テナントはバケットを明確にするために使用されますが、バケットインスタンスには使用されません。

例

```
.bucket.meta.prodtx:test%25star:default.84099.6
.bucket.meta.testcont:default.4126.1
.bucket.meta.prodtx:testcont:default.84099.4
prodtx/testcont
prodtx/test%25star
testcont
```

namespace: users.uid

USER オブジェクト内のユーザーごとの情報 (RGWUserInfo) と、**USER .buckets** オブジェクトの omap 内のバケットのユーザーごとのリストの両方が含まれます。空でない場合には、**USER** にそのテナントが含まれる場合があります。

例

```
prodtx$prodt
test2.buckets
prodtx$prodt.buckets
test2
```

namespace: users.email

Unimportant

namespace: users.keys

47UA98JSTJZ9YAN3OS3O
これにより、Ceph Object Gateway は認証中にアクセスキーでユーザーを検索できます。

namespace: users.swift

test:tester

ZONE.rgw.buckets.index

オブジェクトの名前は **.dir.MARKER** で、それぞれにバケットインデックスが含まれます。インデックスがシャード化されている場合、各シャードはマーカーの後にシャードインデックスを追加します。

ZONE.rgw.buckets.data

default.7593.4__shadow_.488urDFerTYXavx4yAd-Op8mxehnvTI_1 MARKER_KEY
マーカーの例は、**default.16004.1** または **default.7593.4** です。現在の形式は **ZONE です。INSTANCE_ID。BUCKET_ID** ですが、一度生成されたマーカーは再度解析されないため、今後その形式が自由に変更される可能性があります。

関連情報

- 詳細は、**Red Hat Ceph Storage Object Gateway ガイド** の Ceph Object Gateway データレイアウト セクションを参照してください。

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

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

注記

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

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

重要

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

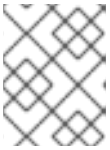
sts:TagSession を使用した信頼ポリシーの例:

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

プロパティー

セッションタグのプロパティーは次のとおりです。

- セッションタグは複数値にすることができます。



注記

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

- Keycloak は、最大 50 個のセッションタグを持つ OpenID Connect ID プロバイダー (IDP) として設定できます。
- 許可されるキーの最大サイズは 128 文字です。

- 許可される値の最大サイズは 256 文字です。
- タグまたは値は **aws:** で開始できません。

関連情報

- セキュアトークンサービスの詳細は、セキュアトークンサービス を参照してください。
- セッションタグの使用例は、STS で属性ベースのアクセス制御にセッションタグを使用する例 を参照してください。
- セッションタグの使用法を示すサンプルコードについては、セッションタグの使用法を示すサンプルコード を参照してください。

3.13.1. タグキー

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

aws:RequestTag

説明

リクエストで渡されたキーと値のペアを、ロールの信頼ポリシーのキーと値のペアと比較します。
AssumeRoleWithWebIdentity の場合、セッションタグはロール信頼ポリシーで **aws:RequestTag** として使用できます。これらのセッションタグは、Web トークンで Keycloak によって渡されます。その結果、フェデレーションユーザーはロールを引き受けることができます。

aws:PrincipalTag

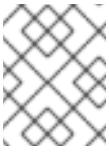
説明

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

iam:ResourceTag

説明

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



注記

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

aws:TagKeys

説明

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

s3:ResourceTag

説明

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

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

3.13.2. S3 リソースタグ

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

タグタイプ: オブジェクトタグ

操作

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

タグタイプ: バケットタグ

操作

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

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

操作

GetACLs、**PutACLs**

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

操作

PutObject、**CopyObject**

3.14. 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ジします。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ジするプロセスは、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。

3.14.1. 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ジする前に、**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。

3.14.2. 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を検討してください。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. 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。
4. 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

3.14.3. ガベージコレクションされたオブジェクト数の表示

パフォーマンスダンプで **gc_retire_object** パラメーターを使用して、Ceph Object Gateway の再起動以降に収集されたオブジェクトのガベージ数を表示できます。

このカウンターを監視して、その Ceph Object Gateway の特定の時間枠でのオブジェクト数の差分 (例: 週、日、または時間ごとにガベージコレクションされるオブジェクトの平均数) を判断できます。これは、Ceph Object Gateway ごとにガベージコレクションが効率的に進行するかどうかを判断するために使用できます。

ソケットファイルは **/var/run/ceph** ディレクトリーにあります。

前提条件

- Ceph Object Gateway がインストールされている実行中の Red Hat Ceph Storage クラスター。
- 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。
- Ceph Monitor にインストールされている **ceph-common** パッケージ。

手順

- **gc_retire_object** カウンターに **grep** を使用して、パフォーマンスカウンターデータにアクセスします。

  構文

```
ceph --admin-daemon PATH_TO_SOCKET_FILE perf dump | grep gc_retire_object
```

例

```
[root@mon ~]# ceph --admin-daemon /var/run/ceph/ceph-client.rgw.f27-h25-000-6048r.rgw0.104.93991732704816.asok perf dump | grep gc_retire_object
```

関連情報

- 詳細は、Red Hat ナレッジベースの記事 Ceph RGW - GC Tuning Optionsを参照してください。
- 詳細は、**Red Hat Ceph Storage Object Gateway 設定および管理ガイド** の **一般設定** セクションを参照してください。
- 詳細は、**Red Hat Ceph Storage Object Gateway 設定および管理ガイド** の **設定リファレンス** セクションを参照してください。
- 詳細は、**Red Hat Ceph Storage 管理ガイド** の **Ceph Object Gateway メトリクス** のセクションを参照してください。

3.15. CEPH OBJECT GATEWAY のデータオブジェクトストレージの最適化

バケットライフサイクルの設定は、データオブジェクトストレージを最適化して効率性を高め、データの存続期間を通じて効果的なストレージを提供します。

Ceph Object Gateway の S3 API は、現在 AWS バケットライフサイクルの設定アクションのサブセットをサポートします。

- 有効期限
- NoncurrentVersionExpiration
- AbortIncompleteMultipartUpload

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。

3.15.1. バケットライフサイクルの並列スレッド処理

Ceph Object Gateway では、複数の Ceph Object Gateway インスタンス間でバケットライフサイクルの並列スレッド処理が可能になりました。並行して実行されるスレッドの数を増やすと、Ceph Object Gateway は大きなワークロードをより効率的に処理できるようになります。さらに、Ceph Object Gateway は、順序どおりの番号付けを使用する代わりに、インデックスシャードの列挙に番号付けされたシーケンスを使用するようになりました。

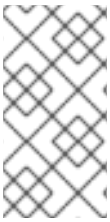
3.15.2. バケットライフサイクルの最適化

Ceph 設定ファイルにある 2 つのオプションは、バケットのライフサイクル処理の効率性に影響を与えます。

- **rgw_lc_max_worker** は、並行して実行するライフサイクルワーカースレッドの数を指定します。これにより、バケットシャードとインデックスシャードの両方を同時に処理できます。このオプションのデフォルト値は 3 です。
- **rgw_lc_max_wp_worker** は、各ライフサイクルワーカーのワーカースレッドプールのスレッド数を指定します。このオプションを使用すると、各バケットの処理を加速することができます。このオプションのデフォルト値は 3 です。

バケット数が多いワークロード (たとえば、バケット数が数千のワークロード) の場合は、**rgw_lc_max_worker** オプションの値を増やすことを検討してください。

バケットの数は少なく、各バケットのオブジェクトの数が多い (数十万など) ワークロードの場合は、**rgw_lc_max_wp_worker** オプションの値を増やすことを検討してください。



注記

これらのオプションのいずれかの値を増やす前に、現在のストレージクラスターのパフォーマンスと Ceph Object Gateway 使用率を検証してください。Red Hat では、これらのオプションのいずれかに対して、10 以上の値を割り当てることは推奨していません。

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. 並行して実行するスレッドの数を増やすには、**rgw_lc_max_worker** の値を 3 から 9 までの値に設定します。

   構文

   ```
   rgw_lc_max_worker = VALUE
   ```

   例

   ```
   rgw_lc_max_worker = 7
   ```

3. 各スレッドのワークプールのスレッド数を増やすには、**rgw_lc_max_wp_worker** の値を 3 から 9 までの値に設定します。

   構文

   ```
   rgw_lc_max_wp_worker = VALUE
   ```

   例

   ```
   rgw_lc_max_wp_worker = 7
   ```

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. ストレージクラスターを監視して、値を増やしてもパフォーマンスに悪影響がないことを確認します。

関連情報

- S3 API およびバケットのライフサイクル操作に関する詳細は、S3 API バケットライフサイクルを参照してください。
- バケットのライフサイクルおよび並列スレッド処理に関する詳細は、Bucket life cycle parallel processing を参照してください。
- Ceph Object Gateway のライフサイクルについての詳細は、Red Hat サポート にお問い合わせください。

3.15.3. 関連情報

- バケットライフサイクルの設定アクションに関する詳細は、S3 API bucket life cycle を参照してください。

3.16. CEPH OBJECT GATEWAY およびマルチファクター認証

ストレージ管理者は、Ceph Object Gateway ユーザーの時間ベースのワンタイムパスワード (TOTP) トークンを管理できます。

3.16.1. マルチファクター認証

バケットがオブジェクトのバージョン管理用に設定されている場合、必要に応じて、削除要求にマルチファクター認証 (MFA) を要求するように設定することができます。MFA を使用すると、時間ベースのワンタイムパスワード (TOTP) トークンは鍵として **x-amz-mfa** ヘッダーに渡されます。トークンは、Google Authenticator などの仮想 MFA デバイス、または Gemalto が提供するようなハードウェア MFA デバイスで生成されます。

radosgw-admin を使用して、時間ベースのワンタイムパスワードトークンをユーザーに割り当てます。シークレットシードおよびシリアル ID を設定する必要があります。**radosgw-admin** を使用して、トークンの一覧表示、削除、および再同期を行うこともできます。

重要

マルチサイト環境では、ゾーンごとに異なるトークンを使用することが推奨されます。これは、MFA ID がユーザーのメタデータに設定されている一方で、実際の MFA ワンタイムパスワード設定はローカルゾーンの OSD に存在するためです。

表3.1 用語

用語	詳細
TOTP	時間ベースのワンタイムパスワード
トークンのシリアル	TOTP トークンの ID を表す文字列。

用語	詳細
トークンシード	TOTP の計算に使用されるシークレットが表示されます。16 進数または base32 にすることができます。
TOTP 秒	TOTP の生成に使用される時間解決。
TOTP ウィンドウ	トークンの検証時に現在のトークンの前後にチェックされる TOTP トークンの数。
TOTP ピン	特定の時点で TOTP トークンの有効な値。

- 詳細は、The Ceph Object Gateway and multi-factor authenticationを参照してください。

3.16.2. マルチファクター認証の作成

マルチファクター認証 (MFA) を設定するには、ワンタイムパスワードジェネレーターおよびバックエンド MFA システムが使用するシークレットシードを作成する必要があります。

前提条件

- Linux システム。
- コマンドラインシェルへのアクセス。

手順

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

   例

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

2. **SEED** 変数で echo を実行して、シードを出力します。

   例

   ```
   [user@host ~]$ echo $SEED
   492dedb20cf51d1405ef6a1316017e
   ```

   同じシードを使用するように、ワンタイムパスワードジェネレーターおよびバックエンドの MFA システムを設定します。

関連情報

- 詳細は、ナレッジベースのソリューション Unable to create RGW MFA token for bucketを参照してください。

- 詳細は、The Ceph Object Gateway and multi-factor authenticationを参照してください。

3.16.3. 新しいマルチファクター認証 TOTP トークンの作成

新たなマルチファクター認証 (MFA) 時間ベースのワンタイムパスワード (TOTP) トークンを作成します。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph Object Gateway がインストールされている。
- Ceph Monitor ノード上での root アクセスがある。
- ワンタイムパスワードジェネレーターおよび Ceph Object Gateway MFA のシークレットシードが生成されている。

手順

1. 新しい MFA TOTP トークンを作成します。

   構文

   ```
   radosgw-admin mfa create --uid=USERID --totp-serial=SERIAL --totp-seed=SEED --totp-seed-type=SEED_TYPE --totp-seconds=TOTP_SECONDS --totp-window=TOTP_WINDOW
   ```

   USERID にはユーザー名を設定し、**SERIAL** には TOTP トークンの ID を表す文字列を設定し、**SEED** には TOTP の計算に使用する 16 進数または base32 値を設定します。以下の設定は任意です。**SEED_TYPE** を **hex** または **base32** に設定し、**TOTP_SECONDS** をタイムアウト (秒単位) に設定するか、**TOTP_WINDOW** を設定して、トークンの検証時に現在のトークンの前後にチェックします。

   例

   ```
   [root@mon ~]# radosgw-admin mfa create --uid=johndoe --totp-serial=MFAtest --totp-seed=492dedb20cf51d1405ef6a1316017e
   ```

関連情報

- 詳細は、Creating a seed for multi-factor authenticationを参照してください。
- 詳細は、Resynchronizing a multi-factor authenitcation token を参照してください。

3.16.4. マルチファクター認証 TOTP トークンのテスト

マルチファクター認証 (MFA) のタイムベースワンタイムパスワード (TOTP) トークンをテストします。

前提条件

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

- Ceph Monitor ノード上での root アクセスがある。
- MFA TOTP トークンは、**radosgw-admin mfa create** を使用して作成されました。

手順

1. TOTP トークンの PIN をテストして、TOTP が正しく機能することを確認します。

   構文

   ```
   radosgw-admin mfa check --uid=USERID --totp-serial=SERIAL --totp-pin=PIN
   ```

   USERID には MFA が設定されているユーザー名を設定し、**SERIAL** には TOTP トークンの ID を表す文字列を設定し、**PIN** にはワンタイムパスワードジェネレーターからの最新の PIN を設定します。

   例

   ```
   [root@mon ~] # radosgw-admin mfa check  --uid=johndoe --totp-serial=MFAtest --totp-pin=870305
   ok
   ```

   PIN の初回テスト時の場合には、失敗する場合があります。失敗する場合は、トークンを再同期します。**Red Hat Ceph Storage Object Gateway Configuration and Administration Guide**の Resynchronizing a multi-factor authentication token を参照してください。

関連情報

- 詳細は、Creating a seed for multi-factor authenticationを参照してください。
- 詳細は、Resynchronizing a multi-factor authenitcation token を参照してください。

3.16.5. マルチファクター認証 TOTP トークンの再同期

マルチファクター認証 (MFA) のタイムベースのワンタイムパスワードトークンを再同期します。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph Object Gateway がインストールされている。
- Ceph Monitor ノード上での root アクセスがある。
- MFA TOTP トークンは、**radosgw-admin mfa create** を使用して作成されました。

手順

1. タイムスキューまたはチェックが失敗した場合に、マルチファクター認証 TOTP トークンを再同期します。
   これには、前のピンと現在のピンの 2 回連続して渡す必要があります。

   構文

```
radosgw-admin mfa resync --uid=USERID --totp-serial=SERIAL --totp-pin=PREVIOUS_PIN --totp=pin=CURRENT_PIN
```

USERID には MFA が設定されているユーザー名を設定し、**SERIAL** には TOTP トークンの ID を表す文字列を設定し、**PREVIOUS_PIN** にはユーザーの以前の PIN を設定し、**CURRENT_PIN** にはユーザーの現在の PIN を設定します。

例

```
radosgw-admin mfa resync --uid=johndoe --totp-serial=MFAtest --totp-pin=802017 --totp-pin=439996
```

2. 新しい PIN をテストしてトークンが正常に再同期されたことを確認します。

構文

```
radosgw-admin mfa check --uid=USERID --totp-serial=SERIAL --totp-pin=PIN
```

USERID には MFA が設定されているユーザー名を設定し、**SERIAL**には TOTP トークンの ID を表す文字列を設定し、**PIN** にはユーザーの PIN を設定します。

例

```
[root@mon ~]# radosgw-admin mfa check  --uid=johndoe --totp-serial=MFAtest --totp-pin=870305
ok
```

関連情報

- 詳細は、Creating a new multi-factor authentication TOTP tokenを参照してください。

3.16.6. マルチファクター認証 TOTP トークンの一覧表示

特定のユーザーが持っているすべてのマルチファクター認証 (MFA) 時間ベースのワンタイムパスワード (TOTP) トークンを一覧表示します。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph Object Gateway がインストールされている。
- Ceph Monitor ノード上での root アクセスがある。
- MFA TOTP トークンは、**radosgw-admin mfa create** を使用して作成されました。

手順

1. MFA TOTP トークンを一覧表示します。

構文

```
radosgw-admin mfa list --uid=USERID
```

-

USERID に、MFA が設定されているユーザー名を設定します。

例

```
[root@mon ~]# radosgw-admin mfa list --uid=johndoe
{
    "entries": [
        {
            "type": 2,
            "id": "MFAtest",
            "seed": "492dedb20cf51d1405ef6a1316017e",
            "seed_type": "hex",
            "time_ofs": 0,
            "step_size": 30,
            "window": 2
        }
    ]
}
```

関連情報

- 詳細は、Creating a new multi-factor authentication TOTP tokenを参照してください。

3.16.7. マルチファクター認証 TOTP トークンの表示

シリアルを指定して、特定のマルチファクター認証 (MFA) 時間ベースのワンタイムパスワード (TOTP) トークンを表示します。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph Object Gateway がインストールされている。
- Ceph Monitor ノード上での root アクセスがある。
- MFA TOTP トークンは、**radosgw-admin mfa create** を使用して作成されました。

手順

1. MFA TOTP トークンを表示します。

   構文

   ```
   radosgw-admin mfa get --uid=USERID --totp-serial=SERIAL
   ```

   USERID には MFA が設定されているユーザー名に設定し、**SERIAL** には TOTP トークンの ID を表す文字列に設定します。

関連情報

- 詳細は、Creating a new multi-factor authentication TOTP tokenを参照してください。

3.16.8. マルチファクター認証 TOTP トークンの削除

マルチファクター認証 (MFA) のタイムベースワンタイムパスワード (TOTP) トークンを削除します。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ceph Object Gateway がインストールされている。
- Ceph Monitor ノード上での root アクセスがある。
- MFA TOTP トークンは、**radosgw-admin mfa create** を使用して作成されました。

手順

1. MFA TOTP トークンを削除します。

   構文

   ```
   radosgw-admin mfa remove --uid=USERID --totp-serial=SERIAL
   ```

   USERID には MFA が設定されているユーザー名に設定し、**SERIAL** には TOTP トークンの ID を表す文字列に設定します。

   例

   ```
   [root@mon ~]# radosgw-admin mfa remove --uid=johndoe --totp-serial=MFAtest
   ```

2. MFA TOTP トークンが削除されたことを確認します。

   構文

   ```
   radosgw-admin mfa get --uid=_USERID_ --totp-serial=_SERIAL_
   ```

   USERID には MFA が設定されているユーザー名に設定し、**SERIAL** には TOTP トークンの ID を表す文字列に設定します。

   例

   ```
   [root@mon ~]# radosgw-admin mfa get --uid=johndoe --totp-serial=MFAtest
   MFA serial id not found
   ```

関連情報

- 詳細は、The Ceph Object Gateway and multi-factor authenticationを参照してください。

3.17. ANSIBLE を使用した CEPH OBJECT GATEWAY の削除

Ansible を使用して Ceph Object Gateway を削除するには、Playbook の **shrink-rgw.yml** を使用します。

前提条件

- Ansible 管理ノード。
- Ansible によりデプロイされた実行中の Red Hat Ceph Storage クラスター

手順

1. **/usr/share/ceph-ansible/** ディレクトリーに移動します。

```
[user@admin ~]$ cd /usr/share/ceph-ansible
```

2. **ベアメタル** および **コンテナー** のデプロイメントに、Ansible Playbook の **shrink-mgr.yml** を実行します。

構文

```
ansible-playbook infrastructure-playbooks/shrink-rgw.yml -e
rgw_to_kill=HOSTNAME.rgw_INSTANCE_NAME_ -u ANSIBLE_USER_NAME -i hosts
```

以下を置き換えます。

- **HOSTNAME** は、Ceph Object Gateway ノードの短縮ホスト名に置き換えます。Playbook を実行するたびに、Ceph Object Gateway を 1 つだけ削除できます。
- **ANSIBLE_USER_NAME** は、Ansible ユーザーの名前に置き換えてください。

例

```
[user@admin ceph-ansible]$ ansible-playbook infrastructure-playbooks/shrink-rgw.yml -e
rgw_to_kill=node03.rgw0 -u admin -i hosts
```

3. ストレージクラスター内のすべての Ceph 設定ファイルから Ceph Object Gateway エントリーを削除します。
4. Ceph Object Gateway が正常に削除されていることを確認します。

```
[root@mon ~]# ceph -s
```

関連情報

- Red Hat Ceph Storage のインストールに関する詳細は、Red Hat Ceph Storage インストールガイド を参照してください。
- Ansible インベントリー設定の詳細は、**{storage_product} インストールガイド** の Ansible のインベントリーの場所の設定 セクションを参照してください。

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

Name	説明	型	デフォルト
rgw_data	Ceph Object Gateway のデータファイルの場所を設定します。	文字列	**/var/lib/ceph/radosgw/$cluster-$id**
rgw_enable_apis	指定された API を有効にします。	String	**s3, s3website, swift, swift_auth, admin, sts, iam, pubsub**
rgw_cache_enabled	Ceph Object Gateway キャッシュを有効にするかどうか。	ブール値	**true**
rgw_cache_lru_size	Ceph Object Gateway キャッシュのエントリー数。	整数	**10000**
rgw_socket_path	ドメインソケットのソケットパス。**FastCgiExternalServer** 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、**rgw.conf** 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
rgw_host	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	**0.0.0.0**
rgw_port	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

Name	説明	型	デフォルト
rgw_dns_name	提供されるドメインの DNS 名。ゾーングループ内での **hostnames** の設定も参照してください。	文字列	なし
rgw_script_uri	リクエストで設定されていない場合は **SCRIPT_URI** の代替値。	文字列	なし
rgw_request_uri	リクエストで設定されていない場合は **REQUEST_URI** の代替値。	文字列	なし
rgw_print_continue	稼働中の場合は **100-continue** を有効にします。	ブール値	**true**
rgw_remote_addr_param	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ーが動作している場合は **X-Forwarded-For** アドレス。	文字列	**REMOTE_ADDR**
rgw_op_thread_timeout	オープンスレッドのタイムアウト (秒単位)。	整数	600
rgw_op_thread_suicide_timeout	Ceph Object Gateway プロセスが終了するまでの **timeout** 時間 (秒単位)。**0** に設定すると無効です。	整数	**0**
rgw_thread_pool_size	スレッドプールのサイズ。	Integer	512 スレッド
rgw_num_control_oids	異なる **rgw** 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	**8**
rgw_init_timeout	Ceph Object Gateway が初期化時に起動するまでの時間 (秒数)。	整数	**30**
rgw_mime_types_file	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	**/etc/mime.types**
rgw_gc_max_objs	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	**32**

Name	説明	型	デフォルト
rgw_gc_obj_min_wait	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	**2 * 3600**
rgw_gc_processor_max_time	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	**3600**
rgw_gc_processor_period	ガベージコレクション処理のサイクル時間。	整数	**3600**
rgw_s3 success_create_obj_status	**create-obj** の代替成功ステータス応答。	整数	**0**
rgw_resolve_cname	**rgw** が要求ホスト名フィールドの DNS CNAME レコードを使用するべきかどうか (ホスト名が **rgw_dns 名** と等しくない場合)。	ブール値	**false**
rgw_object_stripe_size	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	**4 << 20**
rgw_extended_http_attrs	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"
rgw_exit_timeout_secs	無条件に終了する前にプロセスを待機する秒数。	整数	**120**
rgw_get_obj_window_size	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	**16 << 20**
rgw_get_obj_max_req_size	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	**4 << 20**
rgw_relaxed_s3_bucket_names	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	**false**
rgw_list buckets_max_chunk	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	**1000**

Name	説明	型	デフォルト
rgw_override_bucket_index_max_shards	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 **0** 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 (**1000** など) を設定することは推奨されません。 この変数は、**[client]** セクションまたは **[global]** セクションで設定して、**radosgw-admin** 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	**0**
rgw_curl_wait_timeout_ms	特定の **curl** 呼び出しのタイムアウト (ミリ秒単位)。	整数	**1000**
rgw_copy_obj_progress	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	**true**
rgw_copy_obj_progress_every_bytes	コピー進行状況出力間の最小バイト。	整数	**1024 * 1024**
rgw_admin_entry	管理要求 URL のエントリーポイント。	文字列	**admin**
rgw_content_length_compat	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	**false**
rgw_bucket_default_quota_max_objects	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、**[client]** セクションまたは **[global]** セクションで設定して、**radosgw-admin** 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	**-1**
rgw_bucket_quota_ttl	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。	整数	600

Name	説明	型	デフォルト
rgw_user_quota_bucket_sync_interval	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。	整数	180
rgw_user_quota_sync_interval	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。	整数	3600 * 24
log_meta	ゲートウェイがメタデータ操作をログに記録するかどうかを決定するゾーンパラメーター。	ブール値	**false**
log_data	ゲートウェイがデータ操作をログに記録するかどうかを決定するゾーンパラメーター。	ブール値	**false**
sync_from_all	ゾーンがすべてのゾーングループピアから同期するかどうかを設定または設定解除するための **radosgw-admin** コマンド。	ブール値	**false**

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 4 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**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.meta**

- **.default.rgw.log**
- **.default.rgw.buckets.index**
- **.default.rgw.buckets.data**
- **.default.rgw.buckets.non-ec**

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。システムプールは、システム制御、ロギング、ユーザー情報などに関連するオブジェクトを格納します。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>.log**: ログプールには、すべてのバケット/コンテナーのログおよび 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>.rgw.buckets.index**: このプールは、そのプールのインデックスを保存します。
- **.<zone-name>.rgw.buckets.data**: このプールはバケットのデータを格納します。
- **.<zone-name>.rgw.meta**: メタデータプールは **user_keys** およびその他の重要なメタデータを保存します。
- **.<zone-name>.meta: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>.meta:users.keys**: keys プールには、各ユーザー ID のアクセスキーと秘密鍵が含まれます。
- **.<zone-name>.meta:users.email:** email プールには、ユーザー ID に関連するメールアドレスが含まれます。
- **.<zone-name>.meta: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。

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。

Name	説明	型	デフォルト
rgw_zonegroup_root_pool	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	**.rgw.root**
rgw_zone_root_pool	ゾーン固有の情報を保存するプール。	文字列	**.rgw.root**

4.3. ライフサイクル設定

ストレージ管理者は、Ceph Object Gateway にさまざまなバケットライフサイクルオプションを設定できます。このようなオプションにはデフォルト値が含まれます。各オプションを指定しない場合、デフォルト値は自動的に設定されます。

このオプションに特定の値を設定するには、**ceph config set client.rgw OPTION VALUE** コマンドを使用して設定データベースを更新します。

Name	説明	型	デフォルト
rgw_lc_debug_interval	開発者専用。有効期限のルールを日単位から秒単位に拡大し、ライフサイクルルールをデバッグすることができます。Red Hat は、実稼働クラスターではこのオプションを使用しないことを推奨します。	Integer	**-1**
rgw_lc_lock_max_time	Ceph Object Gateway によって内部で使用されるタイムアウト値。	Integer	**90**
rgw_lc_max_objs	RADOS Gateway 内部ライフサイクル作業キューのシャーディングを制御します。意図的な再シャーディングワークフローの一部としてのみ設定する必要があります。Red Hat では、クラスターのセットアップ後に Red Hat サポートに連絡せずにこの設定を変更することは推奨していません。	Integer	**32**
rgw_lc_max_rules	バケットごとに 1 つのライフサイクル設定ドキュメントに含めるライフサイクルルールの数。Amazon Web Service (AWS) の制限は 1000 ルールです。	Integer	**1000**
rgw_lc_max_worker	バケットおよびインデックスシャードを同時に処理する、並行して実行するライフサイクルワーカースレッドの数。Red Hat は、Red Hat サポートに問い合わせることなく 10 を超える値を設定することは推奨していません。	Integer	**3**
rgw_lc_max_wp_worker	各ライフサイクルワーカースレッドが並行して処理できるバケットの数。Red Hat は、Red Hat サポートに問い合わせることなく 10 を超える値を設定することは推奨していません。	Integer	**3**
rgw_lc_thread_delay	複数のポイントでシャードの処理に注入できる遅延 (ミリ秒単位)。デフォルト値は 0 です。値を 10 から 100 ミリ秒に設定すると、RADOS Gateway インスタンスの CPU 使用率が低下し、飽和状態が観察される場合に、インジェストに対するライフサイクルスレッドのワークロード容量の比率が減少します。	Integer	**0**

4.4. SWIFT 設定

Name	説明	型	デフォルト
rgw_enforce_swift_acls	Swift Access Control List (ACL) 設定を強制します。	ブール値	**true**
rgw_swift_token_expiration	Swift トークンの有効期限が切れるまでの時間 (秒単位)。	整数	**24 * 3600**
rgw_swift_url	Ceph Object Gateway Swift API の URL。	文字列	なし
rgw_swift_url_prefix	Swift API の URL 接頭辞 (例: **http://fqdn.com/swift**)。	**swift**	該当なし
rgw_swift_auth_url	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
rgw_swift_auth_entry	Swift 認証 URL のエントリーポイント。	文字列	**auth**

4.5. ログ設定

Name	説明	型	デフォルト
rgw_log_nonexistent_bucket	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	**false**
rgw_log_object_name	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 date を参照してください。	Date	**%Y-%m-%d-%H-%i-%n**
rgw_log_object_name_utc	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。**false** の場合はローカルタイムを使用します。	ブール値	**false**
rgw_usage_max_shards	使用状況ログのシャードの最大数。	整数	**32**
rgw_usage_max_user_shards	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	**1**
rgw_enable_ops_log	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	**false**
rgw_enable_usage_log	使用状況ログを有効にします。	ブール値	**false**
rgw_ops_log_rados	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	**true**

Name	説明	型	デフォルト
rgw_ops_log_socket_path	操作ログを書き込む Unix ドメインソケット。	文字列	なし
rgw_ops_log_data-backlog	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	**5 << 20**
rgw_usage_log_flush_threshold	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
rgw_usage_log_tick_interval	保留中の使用量のログデータを **n** 秒ごとにフラッシュします。	整数	**30**
rgw_intent_log_object_name	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 date を参照してください。	Date	**%Y-%m-%d-%i-%n**
rgw_intent_log_object_name_utc	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。**false** の場合はローカルタイムを使用します。	ブール値	**false**
rgw_data_log_window	データログエントリーウィンドウ (秒単位)。	整数	**30**
rgw_data_log_changes_size	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	**1000**
rgw_data_log_num_shards	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	**128**
rgw_data_log_obj_prefix	データログのオブジェクト名の接頭辞。	文字列	**data_log**
rgw_replica_log_obj_prefix	レプリカログのオブジェクト名の接頭辞。	文字列	**replica log**
rgw_md_log_max_shards	メタデータログのシャードの最大数。	整数	**64**

4.6. KEYSTONE 設定

Name	説明	型	デフォルト
rgw_keystone_url	Keystone サーバーの URL。	文字列	なし
rgw_keystone_admin_token	Keystone 管理トークン (共有シークレット)	文字列	なし

Name	説明	型	デフォルト
rgw_keystone_accepted_roles	要求を提供するのに必要なロール。	文字列	**Member, admin**
rgw_keystone_token_cache_size	各 Keystone トークンキャッシュのエントリーの最大数。	整数	**10000**
rgw_keystone_revocation_interval	トークン失効チェックの間隔 (秒単位)。	整数	**15 * 60**

4.7. LDAP 設定

Name	説明	型	例
rgw_ldap_uri	URI 形式の LDAP サーバーのスペース区切り一覧。	文字列	**ldaps://<ldap.your.domain>**
rgw_ldap_searchdn	ベースドメインとも呼ばれる LDAP 検索ドメイン名。	文字列	**cn=users,cn=accounts,dc=example,dc=com**
rgw_ldap_binddn	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	**uid=admin,cn=users,dc=example,dc=com**
rgw_ldap_secret	**rgw_ldap_binddn** の認証情報を含むファイル	文字列	**/etc/openldap/secret**
rgw_ldap_dnattr	Ceph Object Gateway のユーザー名を含む LDAP 属性 (binddns 形式)。	文字列	**uid**

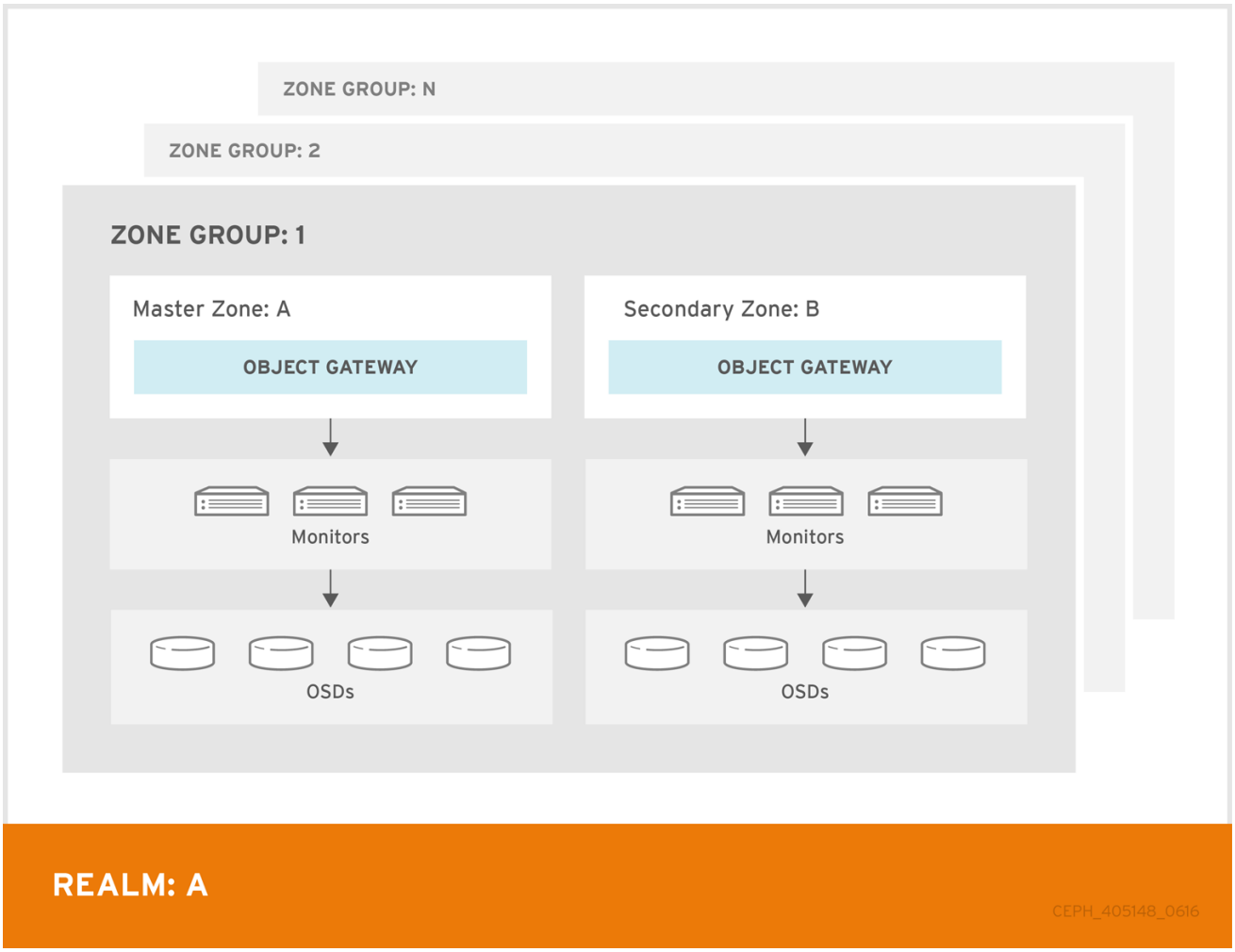
第5章 マルチサイト

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。各ゾーンはアクティブで、書き込み操作を受け取る可能性があります。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。

- **マルチゾーングループ:** 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のオブジェクト ID を確保します。

- **複数のレルム:** Ceph Object Gateway は、レルムの概念をサポートします。レルムは、単一のゾーングループまたは複数のゾーングループと、レルムのグローバルに一意の名前空間です。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。



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

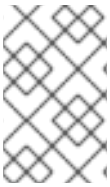
レルムのマスターゾーングループ内のマスターゾーンは、(**radosgw-admin** CLI によって作成された) ユーザー、クォータ、バケットを含むレルムのメタデータのマスターコピーを保存するロールを果たします。このメタデータは、セカンダリーゾーンおよびセカンダリーゾーングループに自動的に同期されます。**radosgw-admin** CLI で実行されるメタデータ操作は、セカンダリーゾーングループおよびゾーンに確実に同期されるように、マスターゾーングループのマスターゾーン内のホストで **実行する必要** があります。現在、セカンダリーゾーンとゾーングループでメタデータ操作を実行する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

- **us-east.rgw.log**
- **us-east.rgw.buckets.index**
- **us-east.rgw.buckets.data**
- **us-east.rgw.buckets.non-ec**
- **us-east.rgw.meta:users.keys**
- **us-east.rgw.meta:users.email**
- **us-east.rgw.meta:users.swift**
- **us-east.rgw.meta:users.uid**

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 Installation Guide 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。

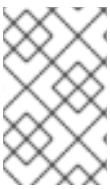
```
[rgws]
<rgw-host-name-1>
<rgw-host-name-2>
<rgw-host-name[3..10]>
```

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ロールも果たします。

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 はレルムに一意の 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 \
```

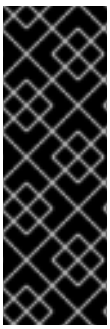
```
--master --default \
--endpoints={http://fqdn:port}[,{http://fqdn:port}]
```

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。

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

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ーングループから削除してください。



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** ゾーングループ、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。

default ゾーンおよびゾー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。

1. ゾーングループとゾーンを削除します。

   例

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

2. クラスター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。

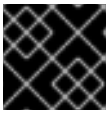
   例

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

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

   例

   ```
   [root@master-zone]# ceph osd pool delete default.rgw.control default.rgw.control --yes-i-really-really-mean-it
   [root@master-zone]# ceph osd pool delete default.rgw.data.root default.rgw.data.root --yes-i-really-really-mean-it
   [root@master-zone]# ceph osd pool delete default.rgw.log default.rgw.log --yes-i-really-really-mean-it
   [root@master-zone]# ceph osd pool delete default.rgw.users.uid default.rgw.users.uid --yes-i-really-really-mean-it
   ```



重要

プールを削除した後、Ceph Object Gateway プロセスを再起動します。

5.4.5. システムユーザーの作成

ceph-radosgw デーモンは、レル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.rgw0]
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。

```
# systemctl start ceph-radosgw@rgw.`hostname -s`.rgw0
# systemctl enable ceph-radosgw@rgw.`hostname -s`.rgw0
```

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。

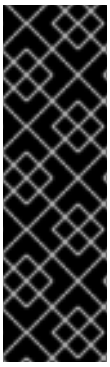
```
# systemctl restart ceph-radosgw@rgw.`hostname -s`.rgw0
```

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。

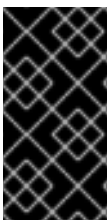
セカンダリーゾーンを提供するために識別されたホストでコマンドラインインターフェイスを開いて、マルチサイト設定用のセカンダリーゾーンを作成します。ゾーングループ ID、新しいゾーン名、およびゾーンのエンドポイントを指定します。**--master** フラグまたは **--default** フラグを使用 **しないでください**。デフォルトでは、すべてのゾーンはアクティブ/アクティブ設定で実行されます。つまり、ゲートウェイクライアントは任意のゾーンにデータを書き込むことができ、ゾーンはゾーングループ内の他のすべてのゾーンにデータを複製します。セカンダリーゾーンが書き込み操作を受け入れない場合は、**--read-only** 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にアクティブ-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。

構文

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

例

```
[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。

```
# ceph osd pool delete default.rgw.control default.rgw.control --yes-i-really-really-mean-it
# ceph osd pool delete default.rgw.data.root default.rgw.data.root --yes-i-really-really-mean-it
# ceph osd pool delete default.rgw.log default.rgw.log --yes-i-really-really-mean-it
# ceph osd pool delete default.rgw.users.uid default.rgw.users.uid --yes-i-really-really-mean-it
```

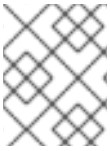
重要

プールの削除後に、RGW プロセスを再起動します。

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.rgw0]
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。

```
# systemctl start ceph-radosgw@rgw.`hostname -s`.rgw0
# systemctl enable ceph-radosgw@rgw.`hostname -s`.rgw0
```

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。

```
# systemctl restart ceph-radosgw@rgw.`hostname -s`.rgw0
```

5.6. アーカイブ同期モジュールの設定 (テクノロジープレビュー)

アーカイブ同期モジュールは、Ceph Object Gateway の S3 オブジェクトのバージョン管理機能を利用して、アーカイブゾーンを設定します。アーカイブゾーンには、アーカイブゾーンに関連付けられたゲートウェイを介してのみ削除できる S3 オブジェクトのバージョンの履歴があります。すべてのデータ更新およびメタデータを取得し、それらを S3 オブジェクトのバージョンとして統合します。

重要

アーカイブ同期モジュールは、テクノロジープレビュー機能のみです。テクノロジープレビュー機能は、Red Hat の実稼働環境でのサービスレベルアグリーメント (SLA) ではサポートされていないため、Red Hat では実稼働環境での使用を推奨していません。テクノロジープレビューの機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。詳細は、Red Hat テクノロジープレビュー機能のサポート範囲 を参照してください。

前提条件

- 稼働中の Red Hat Ceph Storage Cluster
- **root** または **sudo** アクセス
- Ceph Object Gateway のインストール

手順

1. **archive** 階層を使用して新しいゾーンを作成する際に、アーカイブ同期モジュールを設定します。

構文

```
radosgw-admin zone create --rgw-zonegroup={ZONE_GROUP_NAME} --rgw-zone={ZONE_NAME}
--endpoints={http://fqdn:port}[,{http://fqdn:port] --tier-type=archive
```

例

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

関連情報

- 詳細は、**Red Hat Ceph Storage Object Gateway ガイド**の **マルチサイトレルムの確立** セクションを参照してください。

5.7. 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 active-active 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 --read-only=false
```

2. 期間を更新して、変更を反映します。

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

3. Ceph Object Gateway を再起動します。

```
# systemctl restart ceph-radosgw@rgw.`hostname -s`.rgw0
```

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ルムをプルします。

```
# 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. 復旧されたゾーンで Ceph Object Gateway を再起動します。

```
# systemctl restart ceph-radosgw@rgw.`hostname -s`.rgw0
```

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

```
# systemctl restart ceph-radosgw@rgw.`hostname -s`.rgw0
```

5.8. 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. Ceph Object Gateway を再起動します。

```
# systemctl restart ceph-radosgw@rgw.`hostname -s`.rgw0
```

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。

5.9. マルチサイトのコマンドラインの使用方法

5.9.1. 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 は新規クラスターのレルムを作成することを推奨します。

5.9.1.1. 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。

5.9.1.2. 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。

5.9.1.3. 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
```

5.9.1.4. 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、レルムプロパティーを使用して 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。

5.9.1.5. 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
```

5.9.1.6. 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
```

5.9.1.7. 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
```

5.9.1.8. レルムのプル

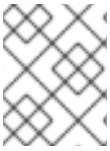
マスターゾーングループとマスターゾーンを含むノードからセカンダリーゾーングループまたはゾーンを含むノードにレルムをプルするには、レルム設定を受け取るノードで **realm pull** を実行します。

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

5.9.1.9. 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。

5.9.2. 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

期間を更新するステップはクラスター全体に変更を伝播するため、**radosgw-admin zonegroup** 操作はレルム内の任意のノードで実行できます。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**。

5.9.2.1. 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。

5.9.2.2. 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
```

5.9.2.3. 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
```

5.9.2.4. 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
```

5.9.2.5. 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
```

5.9.2.6. 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
```

-

5.9.2.7. ゾーングループの一覧表示

Ceph クラスターには、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"
    ]
}
```

5.9.2.8. 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": 11,
            "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": 11,
            "read_only": "false"
        }
    ],
    "placement_targets": [
        {
            "name": "default-placement",
            "tags": []
        }
    ],
    "default_placement": "default-placement",
    "realm_id": "ae031368-8715-4e27-9a99-0c9468852cfe"
}
```

5.9.2.9. ゾーングループの設定

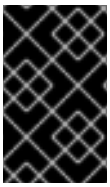
ゾーングループの定義は、少なくとも必要な設定を指定して JSON オブジェクトの作成で設定されます。

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
```

5.9.2.10. 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、クラスターの **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": 11,
                        "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": 11,
                "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
```

5.9.3. 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**。

5.9.3.1. ゾーンの作成

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
```

5.9.3.2. 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 --rgw-zonegroup=<name>\
                          --rgw-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

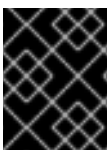
この手順では、ゾーン内のホストで MUST を実行する **必要があります**。

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。

```
# ceph osd pool delete <del-zone>.rgw.control <del-zone>.rgw.control --yes-i-really-really-mean-it
# ceph osd pool delete <del-zone>.rgw.data.root <del-zone>.rgw.data.root --yes-i-really-really-mean-it
# ceph osd pool delete <del-zone>.rgw.log <del-zone>.rgw.log --yes-i-really-really-mean-it
# ceph osd pool delete <del-zone>.rgw.users.uid <del-zone>.rgw.users.uid --yes-i-really-really-mean-it
```

重要

プールの削除後に、RGW プロセスを再起動します。

5.9.3.3. 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
```

5.9.3.4. 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。

```
# radosgw-admin zone list
```

5.9.3.5. 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。

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

default ゾーンは以下のようになります。

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

5.9.3.6. 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。プールの設定に関する詳細は、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]# 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。

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

5.9.3.7. 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
```

5.10. 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。

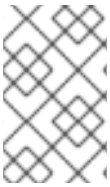
- **default.rgw.control**

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。

Name	説明	型	デフォルト
rgw_zone	ゲートウェイインスタンスのゾーンの名前。	文字列	なし
rgw_zonegroup	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
rgw_zonegroup_root_pool	ゾーングループの root プール。	文字列	**.rgw.root**
rgw_zone_root_pool	ゾーンの root プール。	文字列	**.rgw.root**
rgw_default_zone_group_info_oid	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	**default.zonegroup**

5.11. マルチサイトでのバケットの手動再シャーディング

マルチサイトクラスターでバケットを手動で再シャーディングするには、次の手順を使用します。



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 Ceph 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。

   構文

   ```
   # 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。

   構文

   ```
   # 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。

   構文

   ```
   # 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。

   構文

   ```
   # 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

- 詳細は、**Red Hat Ceph Storage Object Gateway 設定および管理ガイド**の **マルチサイト設定でのバケットインデックスシャード化の設定** を参照してください。

5.12. 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": 11,
            "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": 11,
            "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。

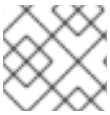
```
# radosgw-admin period update --commit
```

関連情報

- レルム
- ゾーングループ
- ゾーン
- インストールガイド

5.13. 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 Ceph Object Gateway 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の 2 つの稼働中の Red Hat Ceph Storage データセンター。
- すべてのノードへの root または sudo アクセス。
- 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します。
- Ceph Object Gateway ノード上で、**Red Hat Ceph Storage 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します。
- 各 Ceph Object Gateway ノードについて、**Red Hat Ceph Storage 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 を実施します。

手順

1. 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。

   構文

```
radosgw-admin realm create --rgw-realm=REALM_NAME --default
```

例

```
[root@rgw1 ~]# radosgw-admin realm create --rgw-realm=ldc1 --default
```

2. 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。

構文

```
radosgw-admin zonegroup create --rgw-zonegroup=ZONE_GROUP_NAME --
endpoints=http://RGW_NODE_NAME:80 --rgw-realm=REALM_NAME --master --default
```

例

```
[root@rgw1 ~]# radosgw-admin zonegroup create --rgw-zonegroup=ldc1zg --
endpoints=http://rgw1:80 --rgw-realm=ldc1 --master --default
```

3. 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。

構文

```
radosgw-admin zone create --rgw-zonegroup=ZONE_GROUP_NAME --rgw-
zone=ZONE_NAME --master --default --endpoints=HTTP_FQDN[,HTTP_FQDN]
```

例

```
[root@rgw1 ~]# radosgw-admin zone create --rgw-zonegroup=ldc1zg --rgw-zone=ldc1z --
master --default --endpoints=http://rgw.example.com
```

4. 期間をコミットします。

例

```
[root@rgw1 ~]# radosgw-admin period update --commit
```

5. **ceph.conf** を、**rgw_realm** 名、**rgw_zonegroup** 名、および **rgw_zone** 名で更新します。

構文

```
rgw_realm = REALM_NAME
rgw_zonegroup = ZONE_GROUP_NAME
rgw_zone = ZONE_NAME
```

例

```
rgw_realm = ldc1
rgw_zonegroup = ldc1zg
rgw_zone = ldc1z
```

6. RGW デーモンを再起動します。

構文

```
systemctl restart ceph-radosgw@rgw.$(hostname -s).rgw0.service
```

7. 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。

   構文

   ```
   radosgw-admin realm create --rgw-realm=REALM_NAME --default
   ```

   例

   ```
   [root@rgw2 ~]# radosgw-admin realm create --rgw-realm=ldc2 --default
   ```

8. 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。

   構文

   ```
   radosgw-admin zonegroup create --rgw-zonegroup=ZONE_GROUP_NAME --
   endpoints=http://RGW_NODE_NAME:80 --rgw-realm=REALM_NAME --master --default
   ```

   例

   ```
   [root@rgw2 ~]# radosgw-admin zonegroup create --rgw-zonegroup=ldc2zg --
   endpoints=http://rgw2:80 --rgw-realm=ldc2 --master --default
   ```

9. 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。

   構文

   ```
   radosgw-admin zone create --rgw-zonegroup=ZONE_GROUP_NAME --rgw-
   zone=ZONE_NAME --master --default --endpoints=HTTP_FQDN[, HTTP_FQDN]
   ```

   例

   ```
   [root@rgw2 ~]# radosgw-admin zone create --rgw-zonegroup=ldc2zg --rgw-zone=ldc2z --
   master --default --endpoints=http://rgw.example.com
   ```

10. 期間をコミットします。

    例

    ```
    [root@rgw2 ~]# radosgw-admin period update --commit
    ```

11. **ceph.conf** を、**rgw_realm** 名、**rgw_zonegroup** 名、および **rgw_zone** 名で更新します。

    構文

    ```
    rgw_realm = REALM_NAME
    rgw_zonegroup = ZONE_GROUP_NAME
    rgw_zone = ZONE_NAME
    ```

    例

```
rgw_realm = ldc2
rgw_zonegroup = ldc2zg
rgw_zone = ldc2z
```

12. RGW デーモンを再起動します。

構文

```
systemctl restart ceph-radosgw@rgw.$(hostname -s).rgw0.service
```

13. 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。

構文

```
radosgw-admin realm create --rgw-realm=REPLICATED_REALM_1 --default
```

例

```
[user@rgw1 ~] radosgw-admin realm create --rgw-realm=rdc1 --default
```

--default フラグを使用して、レプリケートされたレルムをプライマリーサイトにデフォルト設定します。

14. 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。

構文

```
radosgw-admin zonegroup create --rgw-zonegroup=RGW_ZONE_GROUP --
endpoints=http://_RGW_NODE_NAME:80 --rgw-realm=_RGW_REALM_NAME --master --
default
```

例

```
[root@rgw1 ~]# radosgw-admin zonegroup create --rgw-zonegroup=rdc1zg --
endpoints=http://rgw1:80 --rgw-realm=rdc1 --master --default
```

15. 最初のデータセンターにマスターゾーンを作成します。

構文

```
radosgw-admin zone create --rgw-zonegroup=RGW_ZONE_GROUP --rgw-
zone=_MASTER_RGW_NODE_NAME --master --default --
endpoints=HTTP_FQDN[,HTTP_FQDN]
```

例

```
[root@rgw1 ~]# radosgw-admin zone create --rgw-zonegroup=rdc1zg --rgw-zone=rdc1z --
master --default --endpoints=http://rgw.example.com
```

16. レプリケーション/同期ユーザーを作成し、システムユーザーをマルチサイトのマスターゾーンに追加します。

構文

```
radosgw-admin user create --uid="r_REPLICATION_SYNCHRONIZATION_USER_" --
display-name="Replication-Synchronization User" --system
radosgw-admin zone modify --rgw-zone=RGW_ZONE --access-key=ACCESS_KEY --
secret=SECRET_KEY
```

例

```
[root@rgw1 ~]# radosgw-admin zone modify --rgw-zone=rdc1zg --access-
key=3QV0D6ZMMCJZMSCXJ2QJ --
secret=VpvQWcsfI9OPzUCpR4kynDLAbqa1OIKqRB6WEnH8
```

17. 期間をコミットします。

構文

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

18. 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。

構文

```
rgw_realm = REALM_NAME
rgw_zonegroup = ZONE_GROUP_NAME
rgw_zone = ZONE_NAME
```

例

```
rgw_realm = rdc1
rgw_zonegroup = rdc1zg
rgw_zone = rdc1z
```

19. RGW デーモンを再起動します。

構文

```
systemctl restart ceph-radosgw@rgw.$(hostname -s).rgw0.service
```

20. 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。

構文

```
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
```

21. 最初のデータセンターから期間をプルします。

構文

```
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
```

22. 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。

構文

```
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

```
[root@rgw2 ~]# radosgw-admin zone create --rgw-zone=rdc2z --rgw-zonegroup=rdc1zg --endpoints=https://tower-osd4.cephtips.com --access-key=3QV0D6ZMMCJZMSCXJ2QJ --secret-key=VpvQWcsfI9OPzUCpR4kynDLAbqa1OIKqRB6WEnH8
```

23. 期間をコミットします。

構文

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

24. 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。

構文

```
rgw_realm = REALM_NAME
rgw_zonegroup = ZONE_GROUP_NAME
rgw_zone = ZONE_NAME
```

例

```
rgw realm = rdc1
rgw zonegroup = rdc1zg
rgw zone = rdc2z
```

25. Ceph Object Gateway デーモンを再起動します。

構文

```
systemctl restart ceph-radosgw@rgw.$(hostname -s).rgw0.service
```

26. 2 番目のデータセンターにログインし、master レルムで同期のステータスを確認します。

構文

```
radosgw-admin sync status
```

例

```
[root@rgw2 ~]# 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)
```

27. 最初のデータセンターにログインし、レプリケーション同期レルムで同期ステータスを確認します。

構文

```
radosgw-admin sync status --rgw-realm RGW_REALM_NAME
```

例

```
[root@rgw1 ~]# 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
```

28. 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。

構文

```
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

```
[root@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、ユーザーはデフォルトのレルムに作成されます。ユーザーがローカルレルム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。