Red Hat Ceph Storage 4

実稼働環境向け Ceph Object Gateway ガイド

実稼働環境向けの Ceph Storage クラスターおよび Ceph Object Gateway クラスターのプランニング、設計、およびデプロイ

Last Updated: 2023-08-23

Red Hat Ceph Storage 4 実稼働環境向け Ceph Object Gateway ガイド

実稼働環境向けの Ceph Storage クラスターおよび Ceph Object Gateway クラスターのプランニング、設計、およびデプロイ

法律上の通知

概要

本ガイドでは、クラスターのプランニング、ハードウェアに関する考慮事項、ストレージ戦略の開発、ゲートウェイおよびロードバランサーの設定、ならびに Ceph Object Gateway の使用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 はじめに

『実稼働環境向け Ceph Object Gateway』 ガイドへようこそ。本ガイドでは、実稼働環境で使用するための Ceph Storage クラスターおよび Ceph Object Gateway クラスターの構築に関するトピックについて説明します。

1.1. 本ガイドの対象

本ガイドは、実稼働環境向けに Ceph Object Gateway 環境のデプロイを検討しているユーザーを対象にしています。実稼働環境向けの Ceph Storage クラスターおよび Ceph Object Gateway クラスターのプランニング、設計、およびデプロイを行う一連のトピックを、一般的な Ceph ドキュメントへのリンク(該当する場合)と共に提供します。

1.2. 想定条件

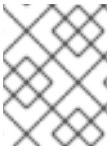
本ガイドでは、読者が Ceph Storage クラスターと Ceph Object Gateway を基本的に理解していることを前提としています。Ceph の経験のない読者は、本ガイドに進む前に、小規模な Ceph テスト環境を設定するか、Ceph Sandbox 環境を使用して Ceph の概念を理解することを検討する必要があります。

本ガイドでは、単一の Ceph Storage クラスターと、同じゾーン内の複数の Ceph Object Gateway インスタンスで構成される単一サイトのクラスターを想定しています。本ガイドでは、セカンダリーゾーングループおよびゾーンに必要な命名変更を加えたゾーングループおよびゾーンごとに、本ガイドの手順を繰り返すことで、単一サイトのクラスターがマルチゾーンおよびマルチサイトクラスターに拡張することを前提としています。

1.3. 対象範囲

本ガイドでは、実稼働環境用の Ceph Storage クラスターおよび Ceph Object Gateway を設定する際の、以下のトピックについて説明します。

- クラスターの計画
- ハードウェアの検討
- クラスターの設定
- Ceph のデプロイ
- ストレージストラテジーの開発
- ゲートウェイの設定
- その他のユースケース



注記

本書は、ハードウェア、インストール、管理、および Ceph Object Gateway ガイドの補完を目的としています。本ガイドは、他のガイドを置き換えません。

第2章 クラスターの計画

Ceph Object Gateway で使用するクラスターのプランニングには、いくつかの重要な考慮事項があります。

- ユースケースの特定
- データ永続性方法の選択
- マルチサイトデプロイメントの検討

これらの要因は、ハードウェアを検討 する際に大きな影響を及ぼします。ハードウェアを選択する前に、これらの要素を慎重に検討してください。

2.1. ユースケースの特定

Ceph Storage は、多くの異なるタイプのストレージユースケースに対応できます。Ceph Object Storage の場合、典型的なユースケースは以下のとおりです。

- **スループットの最適化:** スループットが最適なクラスターは、迅速なデータへのアクセスを確保するために検索します。ホストバスアダプター(HBA)、高速な連続的な読み取り/書き込み特性を持つストレージメディア、高いネットワーク帯域幅により、グラフィック、オーディオ/ビデオのストリーミング機能などのアプリケーション機能が提供されます。スループットが最適化されたクラスターは、書き込みパフォーマンスを考慮するかどうかも検討します。ジャーナリングに SSD を使用するスループットが最適化されたクラスターは、書き込みパフォーマンスを大幅に向上します。これは、CCTV ストリームを保存するアプリケーションにとって重要です。スループットが最適化されたクラスターでは、HBA(Host Bus Adapter)コントローラーのスループット特性、および4K ビデオのストリーミングなどの集中型アプリケーションのネットワークスループットを考慮する必要があります。HBA ベースのハードウェアコントローラーでは、オンボードコントローラーと比べて大幅なパフォーマンスの向上が見られます。

- **容量の最適化:** 容量を最適化したクラスターは、テラバイトあたりのコストを最小限に抑えられるようにします。容量が最適化されたクラスターでは、あまり高価ではないストレージメディアが使用され、多くの場合、頻繁にはアクセスされないレガシーの経理記録や古い電子メールをアーカイブするなど、アプリケーションで個別の SSD ジャーナルの追加コストが発生しないようにします。

- **IOPS の最適化:** IOPS 最適化クラスターは、読み取り/書き込みの集約型のワークロードに対して高パフォーマンスを提供することを重視しています。IOPS が最適化されたワークロードは Ceph Object Gateway では一般的ではありませんが、SSD、Flash メモリー、または NVMe CRUSH 階層を使用してサポートされるます。

クラスターの価格とパフォーマンスに大きく影響するため、ハードウェアを考慮する **前** にストレージのユースケースを慎重に検討してください。たとえば、ユースケースが容量の最適化で、ハードウェアがスループット最適化のユースケースにより適したものの場合、ハードウェアが必要以上に高価になってしまいます。逆に、ユースケースがスループットの最適化で、ハードウェアが容量最適化のユースケースにより適したものの場合、クラスターのパフォーマンスが低下します。

また、Ceph Object Gateway はストレージポリシーをサポートするため、前述の **すべて** のシナリオに対して CRUSH 階層を作成して、API でサポートされているストレージポリシーで呼び出すことができます。詳細は、「データの配置ストラテジーの作成」を参照してください。

2.2. データ永続性手法の選択

クラスター設計では、データの永続性手段も考慮する必要があります。Ceph Storage はレプリケーションまたはイレイジャーコーディングのいずれかを使用して、データの永続性を確保します。

レプリケーションでは、ハードウェア障害に備えて、障害ドメインをまたいで 1 つ以上のデータの冗長コピーを保存します。しかし、データの冗長コピーは、規模が大きくなるとコスト高になります。たとえば、1 ペタバイトのデータを 3 つのレプリケーションで保存するには、少なくとも容量が 3 ペタバイトあるストレージクラスターが必要になります。

『Red Hat Ceph Storage 4 ストレージストラテジーガイド』の「イレイジャーコーディング」セクションでは、イレイジャーコーディングがデータをデータチャンクおよびコーディングチャンクとして保存する方法を説明しています。データチャンクが失われた場合には、イレイジャーコーディングにより、残りのデータチャンクとコーディングチャンクで失われたデータチャンクを回復できます。イレイジャーコーディングはレプリケーションに比べて大幅に経済的です。たとえば、データチャンク 8 つとコーディングチャンク3 つのイレイジャーコーディングを使用すると、データのコピーが 3 つある状態と同じ冗長性が得られます。ただし、このようなエンコーディングスキームでは、初期のデータ保存量が約 1.5 倍になるのに対し、レプリケーションでは 3 倍になります。

注記

データストレージプール **のみ** がイレイジャーコーディングを使用できます。サービスデータやバケットインデックスを格納するプールはレプリケーションを使用します。

2.3. マルチサイトデプロイメントの検討

クラスターの設計に関するもう 1 つの重要な点として、クラスターが 1 つのデータセンターサイトにあるか、複数のデータセンターサイトにまたがるかどうかを判別することです。マルチサイトクラスターは、地理的に分散したフェイルオーバーと、長期的な停電、地震、ハリケーン、洪水、その他の災害からの復旧の恩恵を受けます。さらに、アクティブ/アクティブ構成のマルチサイトクラスターは、クライアントアプリケーションをコンテンツ配信ネットワークの形式で最も近い利用可能なクラスターに転送できます。データを可能な限りクライアントの近く配置することは、4k ビデオのストリーミングなど、スループット集約型のワークロードではますます重要になります。

マルチサイトクラスターの詳細は、『Red Hat Ceph Storage 4 Ceph Object Gateway 設定および管理ガイド』の「マルチサイト」の章を参照してください。

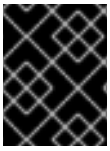
注記

Red Hat は、Ceph Storage プールを作成する「前」に、レルム、ゾーングループ、およびゾーン名を特定することが推奨されます。一部のプール名には、慣例によりゾーン名を先頭に追加する必要があります。

第3章 ハードウェアの検討

ハードウェアを検討することは、実稼働環境用の Ceph Storage クラスターおよび Ceph Object Gateway クラスターを構築する上で重要です。以下は、留意事項の概要です。

- ストレージのサイズ設定の検討
- ストレージの密度の検討
- 無停電電源装置の検討
- ネットワークハードウェアの検討
- ユースケース用のハードウェアの選択
- インデックスのメディアの選択
- ノードのモニター用にメディアの選択



重要

クラスターのコンピューティングハードウェアとネットワークハードウェアを特定して購入する **前に**、これらの要素を検討してください。

3.1. ストレージのサイズ設定の検討

クラスター設計における最も重要な要因の 1 つは、ストレージ要件 (サイズ調整) を決定することです。Ceph Storage は、ペタバイト以上に拡張できるように設計されています。Ceph Storage クラスターの一般的なサイズの例を以下に示します。

- **小規模:** 250 テラバイト
- **中規模:** 1 ペタバイト
- **大規模:** 2 ペタバイト以上。

サイジングには、現在のニーズと近い将来のニーズを含める必要があります。ゲートウェイクライアントがクラスターに新しいデータを追加する速度を考慮してください。これは、ユースケースごとに異なる可能性があります。たとえば、CCTVビデオ、4kビデオ、または医用画像を記録すると、金融市場データなどのストレージをあまり消費しない情報よりもはるかに迅速に大量のデータを追加できます。さらに、レプリケーションやイレイジャーコーディングなどの データ永続性 の方法が、必要なストレージメディアに大きく影響することに注意してください。

サイズ設定の詳細は、『Red Hat Ceph Storage ハードウェア選択ガイド』 およびそれに関連する OSD ハードウェアの選択に関するリンクを参照してください。

3.2. ストレージの密度の検討

クラスター設計のもう 1 つの重要な要素には、ストレージの密度があります。通常、クラスターは、レプリケート、バックフィル、復旧時に適切なパフォーマンスを確保するために、少なくとも 10 個のノードにデータを保存する必要があります。クラスター内に少なくとも 10 個のノードがあるノードに障害が発生した場合、データの 10% のみが残りのノードに移動する必要があります。ノードの数が大幅に少ない場合は、より高い割合のデータを存続するノードに移動する必要があります。さらに、クラ

スターがデータを書き込むことができるように、ノードの障害に対応するために **full_ratio** および **near_full_ratio** を設定する必要があります。このため、ストレージ密度を考慮することが重要です。ストレージの密度が高いことは、必ずしも適切とは限りません。

より高いストレージ密度よりも多くのノードを優先するもう 1 つの要因は、イレイジャーコーディングです。イレイジャーコーディングを使用してオブジェクトを作成し、**ノード** を最小 CRUSH 障害ドメインとして使用する場合、クラスターにはデータおよびコーディングチャンクと同じ数のノードが必要になります。たとえば、**k=8, m=3** を使用するクラスターでは、各データまたはコーディングチャンクが別のノードに保存されるように、最低でも 11 個のノードが必要です。

ホットスワップも重要な考慮事項になります。最新のサーバーの多くは、ドライブのホットスワップに対応します。ただし、一部のハードウェア構成では、ドライブを交換するために複数のドライブを取り外す必要があります。Red Hat は、障害が発生したディスクをスワップアウトするときに必要以上の OSD をダウンさせる可能性があるため、このような構成は回避することを推奨します。

3.3. ネットワークハードウェアの検討

Ceph Storage の主な利点は、容量、IOPS、およびスループットを個別にスケーリングできることです。クラウドストレージソリューションの重要な点は、ネットワークのレイテンシーなどの要因により、クラスターの IOPS が不足するか、あるいはクラスターのストレージ容量が不足するはるか前に、帯域幅の制約によりスループットが不足することにあります。つまり、価格対性能のターゲットを満たすには、ネットワークのハードウェア構成がユースケースをサポートする必要があります。SSD (Solid State Disk) やフラッシュ、NVMe などの高性能なストレージ手段の使用を検討する場合、ネットワークのパフォーマンスの重要性が増しています。

Ceph Storage のもう 1 つの重要な留意点として、クライアントおよびデータのモニター用にフロントサイドまたはパブリックネットワークをサポートし、ハートビート、データのレプリケーション、および復元用にバックサイドまたはクラスターネットワークをサポートすることが挙げられます。つまり、バックエンドネットワークまたはクラスターネットワークには、**常に** フロントエンドまたはパブリックネットワークよりも多くのネットワークリソースが必要になります。データプールがデータの永続性にレプリケーションまたはイレイジャーコーディングを使用するかどうかに応じて、バックサイドまたはクラスターネットワークのネットワーク要件を適切に定量的に設定する必要があります。

最後に、Ceph をインストールしてテストする前に、ネットワークのスループットを検証します。Ceph のパフォーマンスに関する問題のほとんどは、ネットワークの問題から始まります。Cat-6 ケーブルのねじれや曲がりといった単純なネットワークの問題は、帯域幅の低下につながります。フロントサイドのネットワークには、最低でも10 Gbe を使用してください。大規模なクラスターの場合には、バックエンドやクラスターのネットワークに 40 Gbe を使用することを検討してください。あるいは、ネットワークをボンディングするには、LACP モード 4 を使用します。さらに、特にバックエンドまたはクラスターネットワークでは、ジャンボフレーム(MTU 9000)を使用します。

3.4. 無停電電源装置の検討

Ceph の書き込みは極めて単純(イチかゼロか)なため、Ceph OSD ノードでは無停電電源(UPS)に投資する必要はありません。ただし、Red Hat は、Ceph Monitor ノードには UPS を利用することを推奨します。監視には、同期書き込みレイテンシーの影響を受ける **leveldb** を使用します。電源が切れるとデータが破損し、クラスターの状態を復元するのにテクニカルサポートが必要になる場合があります。

ストレージコントローラーがライトバックキャッシュを使用する場合、Ceph OSD は UPS の使用による利点を得られます。このシナリオでは、コントローラーがライトバックキャッシュを時間内にフラッシュしない場合に、UPS は電力停止時にファイルシステムが破損するのを防ぐのに役立ちます。

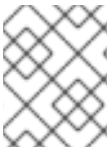
3.5. ユースケース用のハードウェアの選択

Ceph Storage の主な利点は、多くのユースケースをサポートするように設定できることです。通常、Red Hat では、特定のユースケースで OSD ホストを同じように設定することを推奨します。Ceph Storage クラスターの主な 3 つのユースケースは以下のとおりです。

- 最適化した IOPS
- 最適化されたスループット
- 最適化された容量

これらのユースケースでは通常、他の要因と共にドライブ、HBA コントローラー、ネットワーク要件が異なるため、単一ノード構成のこれらのユースケースをすべて促進するために一連の同一ホストを設定することはできますが、必ずしも推奨される方法ではありません。

同じホストを使用して複数の CRUSH 階層を容易に使用するには、CRUSH マップの実際のホスト名ではなく、論理名を使用します。さらに、Ansible などのデプロイメントツールは、すべての OSD をデフォルトの **[osds]** グループにデプロイするのではなく、各ユースケースのためにグループを検討する必要があります。



注記

通常、高 IOPS、高スループット、または大容量などの単一のユースケースに対応するホストを設定して管理する方が容易です。

3.6. インデックスのメディアの選択

ユースケースに関係なく、Ceph Object Gateway で使用する OSD ハードウェアを選択する場合は、インデックスプールを格納するための高性能ドライブが少なくとも 1 つある OSD ノードが必要です。これは、バケットに多数のオブジェクトが含まれる場合は、特に重要になります。

ホストには、SSD ドライブまたは NVMe ドライブが少なくとも 1 つ必要です。Red Hat のラボテストでは、NVMe ドライブは、同じドライブ上の OSD ジャーナルおよびインデックスプールの両方をサポートするのに十分なパフォーマンスを示していますが、NVMe ドライブの異なるパーティションにはそれぞれ対応します。

注記

Red Hat は、インデックスプールの HDD デバイスに対応していません。サポート対象構成の情報については、「Red Hat Ceph Storage: Supported configurations」の記事を参照してください。

インデックスエントリーは約 200 バイトのデータで、**leveldb** にオブジェクトマップ (omap) として保管されます。これはごくわずかな量のデータですが、Ceph Object Gatewayを使用すると、1 つのバケットに数千万から数億のオブジェクトが含まれる可能性があります。インデックスプールを高性能ストレージメディアの CRUSH 階層にマッピングすることにより、バケットに非常に多くのオブジェクトが含まれている場合に、レイテンシーが短くなり、パフォーマンスが劇的に向上します。

重要

実稼働クラスターでは、OSD ジャーナルとインデックスプールを保存するのに、標準の OSD ノードに SSD または NVMe ドライブが 1 つ以上含まれます。同じ物理ドライブを使用する場合は、別のパーティションまたは論理ボリュームを使用します。

3.7. ノードのモニター用にメディアの選択

Ceph モニターは、同期書き込みレイテンシーの影響を受ける **leveldb** を使用します。Red Hat は、SSD を使用してモニターデータを保存することを強く推奨します。選択した SSD に連続した書き込みおよびスループットの特徴が十分にあることを確認してください。

第4章 クラスターの設定

実稼働クラスターの初期設定は、概念実証用のシステムの設定と同じです。唯一のマテリアルの違いは、最初のデプロイメントでは実稼働レベルのハードウェアを使用することです。まず、『Red Hat Ceph Storage 4 インストールガイド』の「Red Hat Ceph Storage のインストール要件」 の章に従い、各ノードで適切な手順を実施します。以下のセクションでは、実稼働クラスターに関する追加の説明を提供します。

4.1. ホストの命名

ホストに名前を付ける際には、そのユースケースとパフォーマンスプロファイルについて検討してください。たとえば、ホストがクライアントデータを保存する場合は、ハードウェア構成およびパフォーマンスプロファイルに従って名前を付けることを検討してください。以下に例を示します。

- **data-ssd-1、data-ssd-2**
- **hot-storage-1、hot-storage-2**
- **sata-1、sata-2**
- **sas-ssd-1、sas-ssd-2**

命名規則により、クラスターの管理およびハードウェアの問題が発生した際のトラブルシューティングが容易になります。

ホストに複数のユースケース用のハードウェアが含まれている場合（たとえば、ホストに、データ用の SSD、SAS ドライブとジャーナル用の SSD、SATA ドライブとコールドストレージ用の共存するジャーナルが含まれる）、ホストには汎用的な名前を選択します。以下に例を示します。

- **osd-node-1 osd-node-2**

汎用的なホスト名は、必要に応じて CRUSH 階層で論理ホスト名を使用する場合に拡張できます。以下に例を示します。

- **osd-node-1-ssd osd-node-1-sata osd-node-1-sas-ssd osd-node-1-bucket-index**
- **osd-node-2-ssd osd-node-2-sata osd-node-2-sas-ssd osd-node-2-bucket-index**

詳細は、「CRUSH マップの論理ホスト名」を参照してください。

4.2. カーネルのチューニング

実稼働環境用のクラスターでは、一般的にオペレーティングシステムのチューニング (特に制限とメモリー割り当て) が有効です。クラスター内の全ノードに調整が設定されていることを確認します。その他のガイダンスは、Red Hat サポートにお問い合わせください。

4.2.1. OSD 用の空きメモリーの確保

OSD メモリー割り当て要求時にメモリー関連のエラーが不十分な状態を防ぐためには、**ceph-ansible** ノードの **group_vars/all.yml** の **os_tuning_params** オプションを設定します。このオプションは、予約する物理メモリーのサイズを指定します。システムメモリーの量に応じて設定を行うことが推奨されます。以下に例を示します。

- RAM が 64 GB の場合は、1GB を確保します。

```
vm.min_free_kbytes = 1048576
```

- RAM が 128 GB の場合は、2 GB を確保します。

```
vm.min_free_kbytes = 2097152
```

- RAM が 256 GB の場合は、3 GB を確保します。

```
vm.min_free_kbytes = 3145728
```

4.2.2. ファイル記述子の増加

ファイル記述子が不足すると、Ceph Object Gateway がハングアップする可能性があります。Ceph Object Gateway ノードで **/etc/security/limits.conf** を変更して、Ceph Object Gateway のファイル記述子を増やします。以下に例を示します。

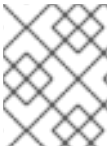
```
ceph       soft    nofile     unlimited
```

4.2.3. 大規模なクラスターでの **ulimit** の調整

大規模なクラスター(例：1024 OSD 以上)で Ceph の管理コマンドを実行する管理者の場合は、管理コマンドを実行する各ノードに、次の内容の **/etc/security/limits.d/50-ceph.conf** ファイルを作成します。

```
<username>       soft    nproc     unlimited
```

<username> を、Ceph 管理者コマンドを実行する root 以外のアカウントの名前に置き換えます。



注記

Red Hat Enterprise Linux では、root ユーザーの ulimit は既にデフォルトで "unlimited" に設定されています。

4.3. ANSIBLE グループの設定

この手順は、Ansible を使用した Ceph のデプロイでの使用のみを目的としています。**ceph-ansible** パッケージはすでにデフォルトの **osds** グループで設定されています。クラスターにユースケースおよびストレージポリシーが 1 つしかない場合は、『Red Hat Ceph Storage インストールガイド』の「Red Hat Ceph Storage クラスターのインストール」セクションに記載の手順に進んでください。クラスターが複数のユースケースおよびストレージポリシーをサポートする場合、それぞれにグループを作成します。各ユースケースは、**/usr/share/ceph-ansible/group_vars/osd.sample** をグループ名のファイルにコピーする必要があります。たとえば、ストレージクラスターに IOPS 最適化ユースケース、スループット最適化ユースケース、および容量最適化ユースケースがある場合は、各ユースケースのグループを表す個別のファイルを作成します。

```
cd /usr/share/ceph-ansible/group_vars/
cp osds.sample osds-iops
cp osds.sample osds-throughput
cp osds.sample osds-capacity
```

次に、ユースケースに従って各ファイルを設定します。

グループ変数ファイルが設定されたら、**site.yml** ファイルを編集して、各新しいグループが含まれるようにします。以下に例を示します。

```
- hosts: osds-iops
  gather_facts: false
  become: True
  roles:
  - ceph-osd

- hosts: osds-throughput
  gather_facts: false
  become: True
  roles:
  - ceph-osd

- hosts: osds-capacity
  gather_facts: false
  become: True
  roles:
  - ceph-osd
```

最後に、**/etc/ansible/hosts** ファイルに、グループに関連付けられた OSD ノードを対応するグループ名の下に配置します。以下に例を示します。

```
[osds-iops]
<ceph-host-name> devices="[ '<device_1>', '<device_2>' ]"

[osds-throughput]
<ceph-host-name> devices="[ '<device_1>', '<device_2>' ]"

[osds-capacity]
<ceph-host-name> devices="[ '<device_1>', '<device_2>' ]"
```

4.4. CEPH の設定

通常、管理者は **/usr/share/ceph-ansible/group_vars** ディレクトリーにある Ceph Ansible 設定ファイルを使用して、初回のデプロイ前に Red Hat Ceph Storage クラスターを設定する必要があります。

『Red Hat Ceph Storage インストールガイド』の「Red Hat Ceph Storage クラスターのインストール」セクションを参照してください。

- モニターの場合、**sample.mons.yml** ファイルから **mons.yml** ファイルを作成します。
- OSD の場合は、**sample.osds.yml** ファイルから **osds.yml** ファイルを作成します。
- クラスターは、**sample.all.yml** ファイルから **all.yml** ファイルを作成します。

『インストールガイド』の説明に従って設定を変更します。

また、「Ceph Object Gateway のインストール」 を参照して、**sample.rgws.yml** から **rgws.yml** ファイルを作成します。

注記

前述のファイルの設定は、**ceph_conf_overrides** の設定よりも優先される場合があります。

mons.yml ファイル、**osds.yml** ファイル、または **rgws.yml** ファイルに、対応する値を使用しない設定を設定するには、**all.yml** ファイルの **ceph_conf_overrides** セクションに構成設定を追加します。以下に例を示します。

```
ceph_conf_overrides:
  global:
    osd_pool_default_pg_num: <number>
```

設定ファイルセクションの詳細は、「設定ファイル構造」を参照してください。

重要

Ansible 設定ファイルで Ceph 構成設定を指定することと、Ceph 設定ファイルでどのようにレンダリングするかには、構文上の違いがあります。

RHCS バージョン 3.1 以前では、Ceph 設定ファイルは **ini** スタイルの概念を使用します。Ceph 設定ファイルの **[global]** 等のセクションは **global:** として指定し、独自の行にインデントする必要があります。特定のデーモンインスタンスの設定セクションを指定することもできます。たとえば、**all.yml** ファイルの **ceph_conf_overrides** セクションに **osd.1:** を追加すると、Ceph 設定ファイルの **[osd.1]** としてレンダリングされ、このセクションの下にある設定は **osd.1** のみに適用されます。

Ceph 構成設定は、スペースではなくハイフン (**-**) またはアンダースコア (_) が含まれ、等号 (**=**) ではなく、コロン (**:**) で終了するはずです。

Ceph クラスターをデプロイする前に、以下の設定を検討してください。Ceph の構成設定を行う場合、Red Hat は **ceph-ansible** 設定ファイルに値を設定することを推奨します。これにより、Ceph 設定ファイルが自動生成されます。

4.4.1. ジャーナルサイズの設定

Ceph クラスターのジャーナルサイズを設定します。Ansible などの設定ツールは、デフォルト値を持つ可能性があります。通常、ジャーナルサイズは同期間隔のプロダクト、低速のディスク、およびネットワークスループットを把握し、プロダクトを 2 倍します。

詳細は、Red Hat Ceph Storage 4 設定ガイドの「ジャーナル設定」セクションを参照してください。

4.4.2. バックフィルおよび復旧設定の調整

I/O は、バックフィルと復旧操作の両方による悪影響を受け、パフォーマンスの低下およびエンドユーザーの不満に繋がります。クラスターの拡張または復旧中の I/O 要求に対応しやすくするには、以下のオプションおよび値を Ceph 設定ファイルに設定します。

```
[osd]
osd_max_backfills = 1
osd_recovery_max_active = 1
osd_recovery_op_priority = 1
```

4.4.3. クラスターマップサイズの調整

Red Hat Ceph Storage バージョン 2 以前では、クラスターに数千の OSD がある場合に、クラスターマップをダウンロードし、そのファイルサイズを確認します。デフォルトでは、**ceph-osd** デーモンは以前の osdmaps を 500 個キャッシュします。重複排除を使用しても、マップはデーモンごとに大量の

メモリーを消費する可能性があります。Ceph 設定ファイルでキャッシュサイズを調整すると、メモリー消費が大幅に削減される可能性があります。以下に例を示します。

```
[global]
osd_map_message_max=10

[osd]
osd_map_cache_size=20
osd_map_max_advance=10
osd_map_share_max_epochs=10
osd_pg_epoch_persisted_max_stale=10
```

Red Hat Ceph Storage バージョン 3 以降では、**ceph-manager** デーモンが PG クエリーを処理するため、クラスターマップはパフォーマンスに影響しません。

4.4.4. スクラビングの調整

デフォルトでは、Ceph はライトスクラブを日単位で実行し、ディープスクラブを週単位で実行します。ライトスクラブは、オブジェクトサイズとチェックサムをチェックして、PG が同じオブジェクトデータを保存していることを確認します。時間の経過とともに、オブジェクトのサイズやチェックサムに関係なく、ディスクセクターが悪化する可能性があります。ディープスクラブは、そのレプリカと共にオブジェクトのコンテンツをチェックして、実際のコンテンツが同じであることを確認します。そのため、**fsck** の方法で、ディープスクラブがデータの整合性を保ちますが、この手順ではクラスターに I/O のペナルティーを課すことになります。わずかなスクラビングは I/O に影響を及ぼす可能性があります。

デフォルト設定では、Ceph OSD が、ピーク動作時間や高負荷の期間などの不適切な時間にスクラブを開始できます。スクラブ操作がエンドユーザーの操作と競合する場合、エンドユーザーは遅延とパフォーマンスの低下を経験する可能性があります。

エンドユーザーのパフォーマンスの低下を防ぐために、Ceph は、スクラブを負荷の低い期間またはオフピーク時間に制限できるいくつかのスクラブ設定を提供します。詳細は、**『Red Hat Ceph Storage Configuration Guide』**の**「Scrubbing the OSD」**セクションを参照してください。

クラスターで日中に高負荷が発生し、深夜に低負荷が発生する場合は、スクラブを夜間に制限することを検討してください。以下に例を示します。

```
[osd]
osd_scrub_begin_hour = 23   #23:01H, or 10:01PM.
osd_scrub_end_hour = 6      #06:01H or 6:01AM.
```

時間制約がスクラブスケジュールを判断する効果的な方法ではない場合は、**osd_scrub_load_threshold** の使用を検討してください。デフォルト値は **0.5** ですが、負荷が低い場合に変更できます。以下に例を示します。

```
[osd]
osd_scrub_load_threshold = 0.25
```

4.4.5. objecter_inflight_ops を増やします。

RHCS 3.0 以前のリリースでは、スケーラビリティーを強化するために、**objecter_inflight_ops** をバージョン 3.1 以降のリリースのデフォルトサイズに増やすことを検討してください。

```
objecter_inflight_ops = 24576
```

-

4.4.6. rgw_thread_pool_size を増やします。

RHCS 3.0 以前のリリースでは、スケーラビリティーを強化するために、**rgw_thread_pool_size** のバージョンをバージョン 3.1 以降のリリースのデフォルトサイズに増やすことを検討してください。以下に例を示します。

```
rgw_thread_pool_size = 512
```

4.4.7. ガベージコレクション設定の調整

Ceph Object Gateway は、新規および上書きされたオブジェクトのストレージをすぐに割り当てます。また、マルチパートアップロードの一部も一部のストレージを使用します。

Ceph Object Gateway は、バケットインデックスからオブジェクトを削除した後に、削除されたオブジェクトに使用されるストレージ領域をパージします。同様に、Ceph Object Gateway は、マルチパートアップロードの完了後にマルチパートアップロードに関連付けられたデータを削除します。または、アップロードが非アクティブであるか、設定可能な期間完了に失敗した場合には、これを削除します。Red Hat Ceph Storage 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(GC)と呼ばれます。

ガベージコレクション待ちのオブジェクトを表示するには、以下のコマンドを実行します。

```
radosgw-admin gc list
```

ガベージコレクションは、ストレージ管理者が Ceph Object Gateway をどのように設定するかに応じて、負荷の低いタイミングで継続的に実行するバックグラウンドアクティビティーです。デフォルトでは、Ceph Object Gateway はガベージコレクション操作を継続的に実行します。ガベッジコレクションの操作は Ceph Object Gateway の通常の機能であるため、特にオブジェクト削除操作では、ガベージコレクションの対象となるオブジェクトはほとんどの場合存在します。

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レートをアウトプレートできます。これは、多くのオブジェクトが短期間保存され、その後に削除される、削除の多いワークロードが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、ストレージ管理者は、以下の設定パラメーターで、他の操作に関連したガベージコレクション操作の優先度を増やすことができます。

- **rgw_gc_obj_min_wait** 設定オプションは、削除されたオブジェクトのデータをパージする前に待機する最小時間 (秒単位) です。デフォルト値は 2 時間（7200 秒）です。クライアントはオブジェクトを読み取る可能性があるため、オブジェクトはすぐにパージされません。削除で負荷が大きい場合、この設定は過剰なストレージを消費するか、パージする多数のオブジェクトが存在する可能性があります。Red Hat は、この値を 30 分以下の 1800 秒に設定しないことを推奨します。

- **rgw_gc_processor_period** 設定オプションは、ガベージコレクションサイクルのランタイムです。つまり、ガベージコレクションスレッドが連続して実行されるまでの時間です。ガベージコレクションがこの期間よりも長く実行される場合、Ceph Object Gateway はガベージコレクションサイクルを再度実行する前に待機しません。

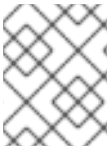
- **rgw_gc_max_concurrent_io** 設定オプションは、削除されたデータをパージする際にゲートウェイガベッジコレクションスレッドが使用する同時 IO 操作の最大数を指定します。負荷が大きい場合には、この設定を多数の同時 IO 操作に増やすことを検討してください。

- **rgw_gc_max_trim_chunk** 設定オプションは、単一の操作でガベージコレクターログから削除する鍵の最大数を指定します。大きな操作を削除する場合、それぞれのガベージコレクション

操作中により多くのオブジェクトがパージされるようにキーの最大数を増やすことを検討してください。

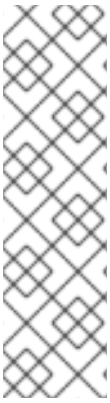
Red Hat Ceph Storage 4.1 以降、ガベッジコレクションログからインデックスオブジェクトの OMAP をオフロードすると、ストレージクラスターのガベージコレクションアクティビティーのパフォーマンスへの影響が低下します。Ceph Object Gateway に新たな設定パラメーターが追加され、以下のようにガベージコレクションキューを調整します。

- The **rgw_gc_max_deferred_entries_size** 設定オプションは、ガベージコレクションのキューに遅延エントリーの最大サイズを設定します。
- **rgw_gc_max_queue_size** 設定オプションは、ガベッジコレクションに使用する最大キューサイズを設定します。この値は、**osd_max_object_size** から **rgw_gc_max_deferred_entries_size** を引いた値から 1KB を引いた値よりも大きくすることはできません。
- **rgw_gc_max_deferred** 設定オプションは、ガベージコレクションキューに保存された遅延エントリーの最大数を設定します。



注記

これらのガベージコレクション設定パラメーターは、Red Hat Ceph Storage 4 以降を対象としています。



注記

テストでは、50% の削除操作と 50% の書き込み操作など、均等にバランスの取れた削除/書き込みワークロードを使用すると、ストレージクラスターは 11 時間で完全にいっぱいになります。これは、Ceph Object Gateway のガベージコレクションの削除操作によるペースの保持に失敗するためです。この場合、クラスターのステータスは **HEALTH_ERR** 状態に切り替わります。並列ガベッジコレクションの調整可能パラメーターの設定項目の設定により、テストするストレージクラスターセットが大幅に遅延し、多くのワークロードに役立ちます。典型的な実際のストレージクラスターワークロードは、主にガベージコレクションによりストレージクラスターがいっぱいになる可能性は想定されていません。

管理者は、デプロイ後のランタイム時に Ceph の設定を変更することもできます。詳細は、「実行時に特定構成の設定」を参照してください。

第5章 CEPH のデプロイ

前提条件と初期チューニングが完了したら、Ceph クラスターをデプロイすることを検討してください。実稼働クラスターをデプロイする場合、Red Hat では、初期モニタークラスターと、**active + clean** の状態に達するのに十分な OSD ノードを設定することを推奨します。詳細は、『Red Hat Ceph Storage 4 インストールガイド』の「Red Hat Ceph Storage クラスターのインストール」 セクションを参照してください。

次に、管理ノードに Ceph CLI クライアントをインストールします。詳細は、『Red Hat Ceph Storage 4 インストールガイド』のの「Ceph コマンドラインインターフェースのインストール」セクションを参照してください。

初期クラスターが実行されたら、以下のセクションの設定を Ceph 設定ファイルに追加することを検討してください。

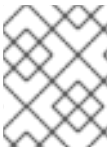
第6章 クラスターの拡張

初期クラスターが実行し、**active+clean** の状態になったら、追加の OSD ノードおよび Ceph Object Gateway ノードをクラスターに追加します。各ノードへの「カーネルのチューニング」に記載の手順を適用します。ノードの追加に関する詳細は、『Red Hat Ceph Storage 4 管理ガイド』の「OSD ノードの追加および削除」セクションを参照してください。

クラスターに追加された各 OSD ノードについて、クライアントデータを格納するノードの各ドライブごとに OSD をクラスターに追加します。詳細は、『Red Hat Ceph Storage 4 管理ガイド』の「OSD の追加」セクションを参照してください。Ansible を使用して OSD ノードを追加する場合は、「Ansible グループの設定」を参照し、クラスターが複数のユースケースをサポートする場合は OSD ノードを適切なグループに追加します。

各 Ceph Object Gateway ノードに、ゲートウェイインスタンスをインストールします。詳細は、『Red Hat Ceph Storage 4 インストールガイド』の「Ceph Object Gateway のインストール」 セクションを参照してください。

クラスターが **active+clean** 状態に戻ると、オーバーライド を削除して「ストレージストラテジーの開発」に進みます。



注記

「ノードの追加」の手順 3 および**「コマンドラインインターフェースを使用した OSD の追加」**の手順 10 は、「CRUSH 階層の開発」から始まるトピックで再考します。

第7章 ストレージストラテジーの開発

実稼働環境での使用に Ceph Storage クラスターおよび Ceph Object Gateway をセットアップする際の困難な要素の 1 つは、効果的なストレージ計画を定義することです。ストレージ計画には、以下の要素が含まれます。

- CRUSH 階層の開発
- CRUSH ルートの作成
- CRUSH マップでの論理ホスト名の使用
- CRUSH ルールの作成
- ルートプールの作成
- レルムの作成
- サービスプールの作成
- データ配置ストラテジーの作成

ストレージストラテジーおよびコマンドラインの使用方法に関する一般的なガイダンスは、Red Hat Ceph Storage 4 の『ストレージストラテジー』ガイドを参照してください。

7.1. CRUSH 階層の開発

Ceph クラスターおよび Object Gateway をデプロイする場合、通常オブジェクトゲートウェイにはデフォルトのゾーングループおよびゾーンがあります。Ceph ストレージクラスターにはデフォルトのプールがあり、次に、デフォルトの CRUSH 階層およびデフォルトの CRUSH ルールで CRUSH マップを使用します。

重要

デフォルトの **rbd** プールはデフォルトの CRUSH ルールを使用できます。Ceph クライアントがデフォルトのルールまたは階層を使用してクライアントデータを保存している場合は、それらを削除 **しないでください**。

CRUSH 階層に関する一般的な詳細は、『ストレージストラテジー』ガイドの「CRUSH 管理」セクションを参照してください。

実稼働ゲートウェイは通常、ゲートウェイの用途と地理的位置に応じて名前が付けられたカスタムの レルム、ゾーングループ、およびゾーン を使用します。また、Ceph クラスターには、複数の CRUSH 階層を持つ CRUSH マップがあります。

- **サービスプール**: 少なくとも 1 つの CRUSH 階層はサービスプール用であり、場合によってはデータ用になります。サービスプールには、**.rgw.root** と、ゾーンに関連付けられたサービスプールが含まれます。サービスプールは、通常単一の CRUSH 階層下にあり、データの持続性のためにレプリケーションを使用します。データプールは CRUSH 階層を使用することもできますが、通常プールはデータの耐久性のためにイレイジャーコーディングで設定されます。
- **インデックス**: 少なくとも 1 つの CRUSH 階層はインデックスプール用にある **必要があり**、CRUSH 階層は SSD ドライブや NVMe ドライブなどの高パフォーマンスのメディアにマップされます。バケットインデックスはパフォーマンスのボトルネックとなる可能性があります。この CRUSH 階層で SSD または NVMe ドライブを使用することを強く推奨します。縮小するに

は、OSD ジャーナルに使用される SSD または NVMe ドライブのインデックス用にパーティションを作成します。さらに、インデックスはバケットシャーディングで設定する必要があります。詳細は、「インデックスプールの作成」およびサポートリンクを参照してください。

- **配置プール:** 各配置ターゲットの配置プールには、バケットインデックス、データバケット、およびバケットの追加が含まれます。これらのプールは、個別の CRUSH 階層下に分類される場合があります。Ceph Object Gateway は複数のストレージポリシーをサポートすることができるため、ストレージポリシーのバケットプールは異なる CRUSH 階層に関連付け、IOPS 最適化、スループット最適化、容量最適化などの異なるユースケースを反映できます。バケットインデックスプールには、SSD ドライブ、NVMe ドライブなどの高性能記憶媒体にバケットインデックスプールをマップするために、独自の CRUSH 階層を使用 **すべきです**。

7.1.1. CRUSH ルートの作成

管理ノードのコマンドラインから、各 CRUSH 階層に対して CRUSH マップに CRUSH ルートを作成します。また、潜在的にデータストレージプールを担うことができるサービスプールに、少なくとも 1 つの CRUSH 階層がある **必要があります**。そのような SSD、NVMe ドライブなどの高性能ストレージメディアにマッピングされたバケットインデックスプールに、少なくとも 1 つの CRUSH 階層がある **はずです**。

CRUSH 階層の詳細は、『**Red Hat Ceph Storage Storage Strategies Guide 4**』の「CRUSH Hierarchies」セクションを参照してください。

CRUSHマップを手動で編集するには、**Red Hat Ceph Storage Storage Strategies Guide 4**の Editing a CRUSH Map セクションを参照してください。

以下の例では、**data0**、**data1**、および **data2** という名前のホストは、同じ物理ホストを参照する CRUSH の階層が複数存在するため、**data0-sas-ssd**、**data0-index** などの拡張論理名を使用します。

一般的な CRUSH ルートは、SAS ドライブを持つノードとジャーナル用の SSD を表す可能性があります。以下に例を示します。

```
##
# SAS-SSD ROOT DECLARATION
##

root sas-ssd {
  id -1   # do not change unnecessarily
  # weight 0.000
  alg straw
  hash 0  # rjenkins1
  item data2-sas-ssd weight 4.000
  item data1-sas-ssd weight 4.000
  item data0-sas-ssd weight 4.000
}
```

バケットインデックスの CRUSH ルートは、SSD や NVMe ドライブなどの高パフォーマンスメディアを表す **はずです**。OSD ジャーナルを格納する SSD または NVMe メディアにパーティションを作成することを検討してください。以下に例を示します。

```
##
# INDEX ROOT DECLARATION
##

root index {
```

```
  id -2    # do not change unnecessarily
  # weight 0.000
  alg straw
  hash 0  # rjenkins1
  item data2-index weight 1.000
  item data1-index weight 1.000
  item data0-index weight 1.000
}
```

7.1.2. CRUSH マップでの論理ホスト名の使用

RHCS 3 以降のリリースでは、CRUSH はストレージデバイスの「class」の概念をサポートします。これは、RHCS 2 以前のバージョンではサポートされないためです。NVMe、SSD、HDD などの複数のストレージデバイスを含むホストまたはノードを持つ RHCS 3 クラスターでは、デバイスクラスを持つ単一の CRUSH 階層を使用して、ストレージデバイスの異なるクラスを区別します。これにより、論理ホスト名を使用する必要がなくなります。RHCS 2 以前のリリースでは、複数の CRUSH 階層を使用し、デバイスの各クラスに 1 つ、論理ホスト名を使用して CRUSH 階層内のホストまたはノードを区別します。

CRUSH マップでは、ホスト名は一意で、1 回のみ使用する必要があります。ホストが複数の CRUSH 階層とユースケースに対応する場合、CRUSH マップは実際のホスト名の代わりに論理ホスト名を使用して、ホスト名が一度だけ使用されるようにします。たとえば、ノードには、SSD などの複数のドライブ、SSD ジャーナルを備えた SAS ドライブ、および同一ジャーナルを持つ SATA ドライブが複数ある場合があります。RHCS 2 以前のリリースでは、同じホストに複数の CRUSH 階層を作成するには、階層は実際のホスト名の代わりに論理ホスト名を使用する必要があります。そのため、バケット名は CRUSH 階層内で一意となるようにします。たとえば、ホスト名が **data2** の場合、CRUSH 階層は、**data2-sas-ssd**、**data2-index** などの論理名を使用する可能性があります。以下に例を示します。

```
host data2-sas-ssd {
  id -11    # do not change unnecessarily
  # weight 0.000
  alg straw
  hash 0  # rjenkins1
  item osd.0 weight 1.000
  item osd.1 weight 1.000
  item osd.2 weight 1.000
  item osd.3 weight 1.000
}
```

前述の例では、ホスト **data2** は論理名 **data2-sas-ssd** を使用して、SSD にジャーナルのある SAS ドライブを 1 つの階層にマッピングします。上記の例の **osd.0** から **osd.3** の OSD ID は、高スループットのハードウェア構成で SSD ジャーナルを使用する SAS ドライブを表しています。これらの OSD ID は、以下の例の OSD ID とは異なります。

以下の例では、ホスト **data2** は論理名 **data2-index** を使用してバケットインデックスの SSD ドライブを 2 つ目の階層にマッピングします。以下の例の OSD ID の **osd.4** は、バケットインデックスプール専用に使用される SSD ドライブまたはその他の高速ストレージメディアを示しています。

```
host data2-index {
  id -21    # do not change unnecessarily
  # weight 0.000
  alg straw
  hash 0  # rjenkins1
  item osd.4 weight 1.000
}
```

重要

論理ホスト名を使用する場合は、以下の設定のいずれかが Ceph 設定ファイルに存在することを確認します。これにより、OSD 起動スクリプトが起動時に実際のホスト名を使用しなくなり、CRUSH マップのデータの検索に失敗します。

CRUSH マップが論理ホスト名を使用する場合、この例では、OSD の起動スクリプトが、初期化時に実際のホスト名に従ってホストを特定しないようにします。Ceph 設定ファイルの **[global]** セクションに、以下の設定を追加します。

```
osd_crush_update_on_start = false
```

論理ホスト名を定義する別の方法は、Ceph 設定ファイルの **[osd.<ID>]** セクションで各 OSD の CRUSH マップの場所を定義することです。これにより、OSD の起動スクリプトが定義する場所が上書きされます。前述の例からは、エントリーは以下のようになります。

```
[osd.0]
osd crush location = "host=data2-sas-ssd"

[osd.1]
osd crush location = "host=data2-sas-ssd"

[osd.2]
osd crush location = "host=data2-sas-ssd"

[osd.3]
osd crush location = "host=data2-sas-ssd"

[osd.4]
osd crush location = "host=data2-index"
```

重要

CRUSH マップが実際のホスト名ではなく論理ホスト名を使用する場合、Ceph Storage Cluster は OSD が実際のホスト名にマップされ、実際のホスト名は CRUSH マップにないことを想定し、Ceph Storage Cluster は OSD が実際のホスト名にないことを想定し、Ceph Storage Cluster クライアントは OSD とそのデータを見つけません。

7.1.3. CRUSH ルールの作成

デフォルトの CRUSH 階層と同様に、CRUSH マップにはデフォルトの CRUSH ルールも含まれます。

注記

デフォルトの **rbd** プールはこのルールを使用できます。他のプールを使用して顧客データを保存する場合は、デフォルトのルールを削除しないでください。

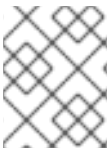
CRUSH ルールの一般的な詳細は、Red Hat Ceph Storage 4 の『ストレージストラテジー』ガイドの「CRUSH ルール」セクションを参照してください。CRUSH マップを手動で編集するには、Red Hat Ceph Storage 4 の『ストレージストラテジー』ガイドの「CRUSH マップ」セクションを参照してください。

CRUSH 階層ごとに、CRUSH ルールを作成します。以下の例は、**.rgw.root** を含むサービスプールを保

存する CRUSH 階層のルールを示しています。この例では、ルート **sas-ssd** がメインの CRUSH 階層として機能します。デフォルトのルールと区別するために、**rgw-service** という名前を使用します。**step take sas-ssd** 行は、「CRUSH ルートの作成」で作成された **sas-ssd** ルートを使用するようにプールに指示します。このルートの子バケットには、SAS ドライブを備えた OSD と、高スループットハードウェア構成のジャーナルに対して SSD または NVMe ドライブなどの高性能ストレージメディアが含まれます。**step chooseleaf** の **type rack** 部分が障害ドメインになります。以下の例では、ラックです。

```
##
# SERVICE RULE DECLARATION
##

rule rgw-service {
 type replicated
 min_size 1
 max_size 10
 step take sas-ssd
 step chooseleaf firstn 0 type rack
 step emit
}
```



注記

前述の例では、データが 3 回複製される 3 回複製される場合は、同様の数の OSD ノードを含むクラスターに 3 つ以上のラックが存在する必要があります。

ヒント

type replicated 設定は、データ永続性、レプリカ数、またはイレイジャーコーディングとは **関係ありません。複製** のみがサポートされます。

以下の例は、データプールを保存する CRUSH 階層のルールを示しています。この例では、root **sas-ssd** は、サービスルールと同じ CRUSH 階層としてメインの CRUSH 階層として機能します。 **rgw-throughput** を使用して、デフォルトのルールと **rgw-service** と区別します。**step take sas-ssd** 行は、「CRUSH ルートの作成」で作成された **sas-ssd** ルートを使用するようにプールに指示します。このルートの子バケットには、SAS ドライブを備えた OSD と、高スループットハードウェア構成の SSD または NVMe ドライブなどの高性能ストレージメディアが含まれます。**step chooseleaf** の **type host** 部分障害ドメインになります。以下の例では、ホストです。ルールは同じ CRUSH 階層を使用し、異なる障害ドメインを使用することに注意してください。

```
##
# THROUGHPUT RULE DECLARATION
##

rule rgw-throughput {
 type replicated
 min_size 1
 max_size 10
 step take sas-ssd
 step chooseleaf firstn 0 type host
 step emit
}
```

注記

前述の例では、プールが大量のデータでイレイジャーコーディングを使用し、デフォルトよりもエンコーディングのチャンクを使用する場合、イレイジャーコーディングのチャンクを容易にするために、同様の数の OSD ノードを含むクラスター内のラックが少なくとも数ある必要があります。小規模なクラスターの場合、これは実用的ではない可能性があるため、前述の例では **host** を CRUSH 障害ドメインとして使用します。

以下の例は、インデックスプールを保存する CRUSH 階層のルールを示しています。この例では、root の **index** は主な CRUSH 階層として機能します。**rgw-index** を使用して、**rgw-service** と **rgw-throughput** と区別します。**step take index** 行は、「CRUSH ルートの作成」で作成された **index** root を使用するようにプールに指示します。その子バケットには、SSD ドライブ、NVMe ドライブなどの高性能ストレージメディア、またはOSD ジャーナルも格納する SSD ドライブまたは NVMe ドライブ上のパーティションが含まれます。**step chooseleaf** の **type rack** 部分が障害ドメインになります。以下の例では、ラックです。

```
##
# INDEX RULE DECLARATION
##

rule rgw-index {
 type replicated
 min_size 1
 max_size 10
 step take index
 step chooseleaf firstn 0 type rack
 step emit
}
```

7.2. ルートプールの作成

Ceph Object Gateway 設定は、レルム、ゾーングループ、ゾーンなど、**.rgw.root** という名前のプールに保存されます。通常、その名前はゾーン名の前に追加されません。

.rgw.root

Ceph Storage クラスターが実行中の場合には、新しいルールを使用して **.rgw.root** プールを作成します。PG 数の詳細は、『ストレージストラテジーガイド』の「Ceph Placement Groups (PGs) per Pool Calculator」および「配置グループ」の章を参照してください。プールの作成に関する詳細は、『ストレージストラテジーガイド』のCreate a Pool「プールの作成」セクションを参照してください。

この場合、プールは **replicated** を使用し、データ永続性に **erasure** は使用 **しません**。以下に例を示します。

```
# ceph osd pool create .rgw.root 32 32 replicated sas-ssd
```

注記

.rgw.root を含むサービスプールの場合は、「Ceph Placement Groups (PGs) per Pool Calculator」から提案される PG 数は、1 OSD あたりのターゲットの PG よりもはるかに少なくなります。また、OSD の数が、calculator のステップ 3 で設定されていることを確認します。

このプールが作成されると、Ceph Object Gateway は設定データをプールに保存できます。

7.3. レルムの作成

Ceph Object Gateway をサポートする Ceph Storage プールは、ゾーングループ内のゾーンに適用されます。デフォルトでは、Ceph Object Gateway はデフォルトのゾーングループおよびゾーンを定義します。

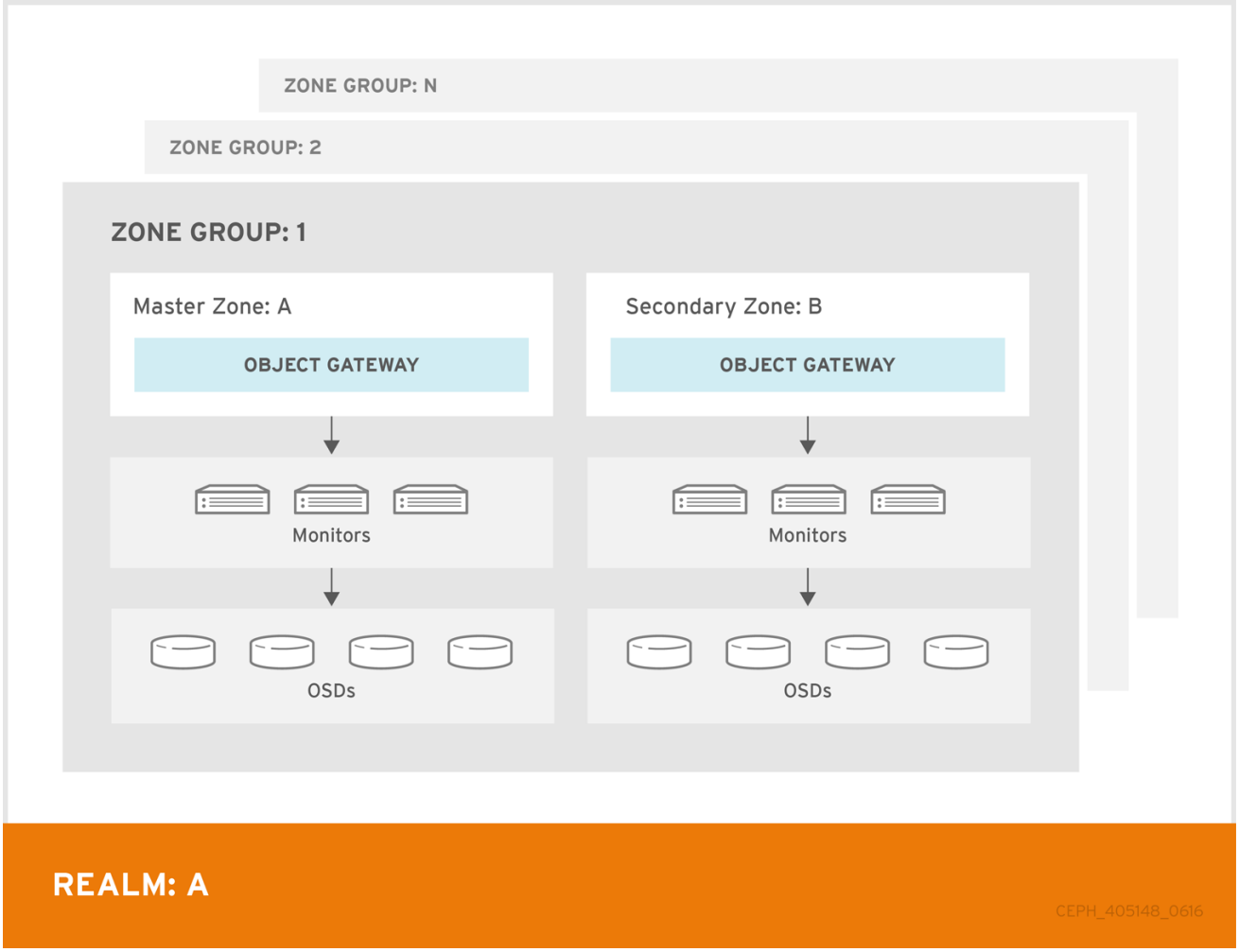
マスターゾーングループおよびゾーンについては、Red Hatは新しいレルム、ゾーングループ、およびゾーンを作成することを推奨します。次に、デフォルトゾーンとそのプールがすでに生成されている場合は削除します。「マルチサイト」操作用にクラスターを設定するため、「マスターゾーンの設定」をベストプラクティスとして使用します。

1. 「レルムを作成」します。詳細は、「レルム」を参照してください。
2. マスターゾーングループを作成 します。ゾーングループの詳細は、「ゾーングループ」を参照してください。
3. 「マスターゾーンを作成」します。ゾーンの詳細は、「ゾーン」を参照してください。
4. デフォルトのゾーングループおよびゾーンを削除 します。あなたは、デフォルトのプールが作成され、そこにクライアントデータが格納されていない場合は、デフォルトのプールを削除**することができます。.rgw.root** プールは削除 **しないでください。**
5. システムユーザーを作成します。
6. 期間を更新します。
7. Ceph 設定ファイルを更新します。

注記

ゲートウェイがプールを手動で作成する可能性があるため、この手順ではゲートウェイの起動手順を省略します。特定の CRUSH ルールおよびデータ永続性手段を指定するには、プールを手動で作成します。

新しいレルム、ゾーングループ、およびゾーンを設定すると、クラスターは、ゾーングループに複数のゾーンがあるマルチサイトクラスターに拡張するための準備が整っています。つまり、クラスターはフェイルオーバーおよび障害復旧用に拡張および設定できます。詳細は、「マルチサイトでのクラスターの拡張」を参照してください。



Red Hat Ceph Storage 2 では、複数サイトの設定はデフォルトでアクティブ/アクティブになっています。複数サイトのクラスターをデプロイする場合、ゾーンとその基礎となる Ceph ストレージクラスターは異なる地理的リージョンにある場合があります。各ゾーンには、同じ名前空間内の各オブジェクトのディープコピーがあるため、ユーザーは物理的に最も近いゾーンからコピーにアクセスでき、レイテンシーが短縮されます。ただし、セカンダリーゾーンがフェイルオーバーおよび障害復旧のみを目的としている場合、クラスターはアクティブ/パッシブモードで設定される場合があります。

注記

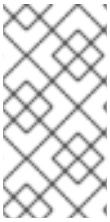
複数のゾーンを持つゾーングループの使用がサポートされます。複数のゾーングループの使用はテクノロジープレビューとしてのみ提供されており、実稼働環境ではサポートされません。

7.4. サービスプールの作成

Ceph Object Gateway は、さまざまなサービス機能に多くのプールを使用し、バケットインデックス、データ、およびその他の情報を保存するために個別の配置プールセットを使用します。

プールの配置グループをピアリングするのはコンピューティングコストがかかるため、Red Hat は通常、Ceph Object Gateway のサービスプールがデータストレージプールよりもはるかに少ない配置グループを使用することを推奨しています。

サービスプールは、サービス制御、ガベージコレクション、ロギング、ユーザー情報、および使用状況などに関連するオブジェクトを保存します。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。



注記

Red Hat Ceph Storage 4.1 以降、ガベッジコレクションは、OMAP ではなく通常の RADOS オブジェクトでログプールを使用します。今後は、より多くの機能はメタデータをログプールに保管します。したがって、ログプールに NVMe/SSD OSD を使用することを強く推奨します。

- **.<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。

「ゾーンの取得」の手順を実行し、プール名を表示します。

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

radosgw-admin はゾーンを作成すると、プール名は、ゾーン名を先頭に追加する **必要があります**。たとえば、**us-west** の名前ゾーンは、次のようになり何かというプールの名前を持っている **必要があります**。

```
{ "domain_root": ".rgw.root",
  "control_pool": ".us-west.rgw.control",
  "gc_pool": ".us-west.rgw.gc",
  "log_pool": ".us-west.log",
  "intent_log_pool": ".us-west.intent-log",
  "usage_log_pool": ".us-west.usage",
  "user_keys_pool": ".us-west.users.keys",
  "user_email_pool": ".us-west.users.email",
  "user_swift_pool": ".us-west.users.swift",
  "user_uid_pool": ".us-west.users.uid",
  "system_key": { "access_key": "", "secret_key": ""},
  "placement_pools": [
   {  "key": "default-placement",
      "val": { "index_pool": ".us-west.rgw.buckets.index",
               "data_pool": ".us-west.rgw.buckets",
               "data_extra_pool": ".us-west.rgw.buckets.non-ec"
               "index_type": 0
```

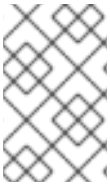
```
        }
    }
  ]
}
```

control_pool から始まり、**user_uid_pool** で終わる場合は、ゾーン名の前にプール名が追加されていれば、そのゾーン名を使用してプールを作成します。前述の例に従うと、プールの作成は以下のようになります。

```
# ceph osd pool create .us-west.rgw.control 32 32 replicated rgw-service
...
# ceph osd pool create .us-west.users.uid 32 32 replicated rgw-service
```

以前の例から、**rgw-service** ルールは、SSD ジャーナルおよび **rack** を CRUSH 障害ドメインとして持つ SAS ドライブの CRUSH 階層を表します。前述の例は、「CRUSH Root の作成」および「CRUSH ルールの作成」を参照してください。

PG 数の詳細は、『ストレージストラテジーガイド』の「Ceph Placement Groups (PGs) per Pool Calculator」および「配置グループ」を参照してください。プールの作成に関する詳細は、『ストレージストラテジーガイド』の「プールの作成」セクションを参照してください。



注記

サービスプールの場合、計算結果から推奨される PG 数は、OSD ごとのターゲット PG よりもはるかに少なくなります。計算のステップ 3 で、正しい OSD 数を指定するようにしてください。

通常、**.rgw.root** プールとサービスプールは同じ CRUSH 階層を使用し、CRUSH ルールの障害ドメインとして少なくとも **node** を使用する必要があります。**.rgw.root** プールと同様に、サービスプールは、データの耐久性のために、**erasure ではなく**、**replicated** を使用する必要があります。

7.5. データ配置ストラテジーの作成

Ceph Object Gateway には、**default-placement** と呼ばれるデフォルトのストレージポリシーがあります。クラスターにストレージポリシーが 1 つしかない場合は、**default-placement** ポリシーで十分です。このデフォルトの配置ポリシーはゾーングループの設定から参照され、ゾーン設定で定義されます。

詳細は、Red Hat Enterprise Linux 用の Red Hat Ceph Storage 4 Ceph Object Gateway ガイドの「ストレージポリシー」セクションを参照してください。

IOPS 最適化、スループットの最適化、容量最適化など、複数のユースケースをサポートするクラスターの場合、ゾーングループ設定の配置ターゲットのセットとゾーン設定の配置プールのセットは各ストレージポリシーを表します。

以下のセクションでは、ストレージポリシーを作成し、それをデフォルトのポリシーに設定する方法を説明します。また、この例では、デフォルトのポリシーがスループット最適化ハードウェアプロファイルを使用することを前提としています。トピックには以下が含まれます。

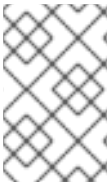
- インデックスプールの作成
- データプールの作成
- データ追加プールの作成

- ゾーングループへの配置ターゲットの設定
- ゾーンへの配置プールの設定
- データ配置の概要

7.5.1. インデックスプールの作成

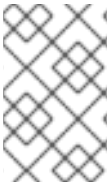
デフォルトでは、Ceph Object Gateway はバケットのオブジェクトをインデックスにマッピングするため、ゲートウェイクライアントはバケット内のオブジェクト一覧を他の項目と共に要求できます。一般的なユースケースでは、ユーザーにバケットおよびバケットごとに制限された数のオブジェクトがあるクォータが使用されますが、バケットは列挙可能なオブジェクトを保存できます。バケットが大量のオブジェクトを保存する場合、SSD ドライブや NVMe ドライブなどの高パフォーマンスストレージメディアを使用してデータを保存することで、インデックスのパフォーマンスが大幅に向上します。さらに、バケットのシャード化により、パフォーマンスが大幅に向上します。

PG 数の詳細は、『ストレージストラテジーガイド』の「Ceph Placement Groups (PGs) per Pool Calculator」および「配置グループ」を参照してください。プールの作成に関する詳細は、『ストレージストラテジーガイド』の「プールの作成」セクションを参照してください。



注記

PG per Pool Calculator では、インデックスプールに対してプールあたりの PG 数を少なくすることが推奨されています。ただし、PG 数はサービスプールの PG 数の約 2 倍です。



注記

Red Hat は、インデックスプールの HDD デバイスに対応していません。サポート対象構成の情報については、「**Red Hat Ceph Storage: Supported configurations**」の記事を参照してください。

インデックスプールを作成するには、**ceph osd pool create** にプール名、PG および PGP の数、**replicated** データ永続性メソッド、およびルール名を指定して作成します。

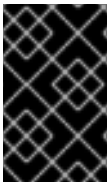
重要

バケットが 10 万を超えるオブジェクトを保存する場合は、バケットのオブジェクト数が増える際にインデックスのパフォーマンスが低下しないようにバケットのシャード化を設定します。『Ceph Object Gateway ガイドの設定および管理ガイド』の「バケットシャーディングの構成」セクションを参照してください。元の構成が適切でなくなった場合のバケットの再シャーディングの詳細については、『Ceph Object Gateway ガイド構成および管理ガイド』の「バケットインデックスの再シャーディング」セクションも参照してください。

7.5.2. データプールの作成

データプールは、Ceph Object Gateway が特定のストレージポリシーのオブジェクトデータを保管する場所です。データプールは、減らしたサービスプールの PG 数ではなく、PG 数を完全に補完する必要があります。データプールは、それが実質的に複製するよりも効率的であり、データの耐久性を維持しながら、大幅に容量要件を減らすことができますよう、イレイジャーコーディングを使用して検討 **すべきです**。

イレイジャーコーディングを使用するには、イレイジャーコードプロファイルを作成します。詳細は、『ストレージストラテジーガイド』の「イレイジャーコーディングプロファイル」セクションを参照してください。



重要

プールの作成後にプロファイルを変更できないため、正しいプロファイルを選択することが重要です。プロファイルを変更するには、別のプロファイルで新しいプールを作成し、オブジェクトを古いプールから新しいプールに移行する必要があります。

デフォルト設定は 2 つのデータチャンクと 1 つのエンコーディングチャンクです。つまり、1 つの OSD のみが失われる可能性があります。回復性が高い場合は、大量のデータおよびエンコーディングチャンクを考慮してください。たとえば、大規模なシステムでは 8 個のデータチャンクと 3 つのエンコーディングチャンクを使用しているため、3 つの OSD が失敗してもデータが失われることはありません。

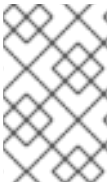
重要

各データおよびエンコードチャンク SHOULD は、最低でも別のノードまたはホストに保存されます。小規模なクラスターの場合、これにより、大量のデータおよびエンコードチャンクを使用する場合に、**rack** の非現実障害ドメインを最低限の CRUSH 障害ドメインとして使用します。そのため、データプールは、最小 CRUSH 障害ドメインとして、**host** を持つ別の CRUSH 階層を使用するのが一般的です。Red Hat は、最小障害ドメインとして **host** を推奨します。イレイジャーコードチャンクが同じホスト内の OSD に保存されていると、ジャーナルやネットワークカードなどのホストの障害により、データが失われる可能性があります。

データの追加プールを作成するには、**ceph osd pool create** にプール名、PG および PGP の数、**erasure** データ永続性メソッド、イレイジャーコードプロファイル、およびルール名を指定して実行します。

7.5.3. データ追加プールの作成

data_extra_pool は、イレイジャーコーディングを使用できないデータ向けです。たとえば、マルチパートアップロードにより、複数部分でのモジションなどの大規模なオブジェクトのアップロードが可能です。これらのパーツは、最初にイレイジャーコーディングなしで保存する必要があります。イレイジャーコーディングは、部分的なアップロードではなく、オブジェクト全体に適用されます。



注記

PG per Pool Calculator では、**data_extra_pool** 対してプールあたりの PG 数を少なくすることが推奨されています。ただし、PG 数はサービスプールとバケットインデックスプールと同じ PG の約 2 倍です。

データの追加プールを作成するには、**ceph osd pool create** にプール名、PG および PGP の数、**replicated** データ永続性メソッド、およびルール名を指定して作成します。以下に例を示します。

```
# ceph osd pool create .us-west.rgw.buckets.non-ec 64 64 replicated rgw-service
```

7.5.4. ゾーングループへの配置ターゲットの設定

プールを作成したら、ゾーングループに配置ターゲットを作成します。ゾーングループを取得するには、次のコマンドを実行して、ゾーングループの設定を **zonegroup.json** というファイルに出力します。

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

ファイルの内容は以下のようになります。

```
{
  "id": "90b28698-e7c3-462c-a42d-4aa780d24eda",
  "name": "us",
  "api_name": "us",
  "is_master": "true",
  "endpoints": [
    "http:\/\/rgw1:80"
  ],
  "hostnames": [],
  "hostnames_s3website": [],
  "master_zone": "9248cab2-afe7-43d8-a661-a40bf316665e",
  "zones": [
      {
        "id": "9248cab2-afe7-43d8-a661-a40bf316665e",
        "name": "us-east",
        "endpoints": [
            "http:\/\/rgw1"
        ],
        "log_meta": "true",
        "log_data": "true",
        "bucket_index_max_shards": 0,
        "read_only": "false"
      },
      {
        "id": "d1024e59-7d28-49d1-8222-af101965a939",
        "name": "us-west",
        "endpoints": [
            "http:\/\/rgw2:80"
        ],
        "log_meta": "false",
        "log_data": "true",
        "bucket_index_max_shards": 0,
        "read_only": "false"
      }
  ],
  "placement_targets": [
      {
        "name": "default-placement",
        "tags": []
      }
  ],
  "default_placement": "default-placement",
  "realm_id": "ae031368-8715-4e27-9a99-0c9468852cfe"
}
```

placement_targets セクションには、各ストレージポリシーが一覧表示されます。デフォルトでは、**default-placement** という配置ターゲットが含まれます。デフォルトの配置ターゲットは、**placement_targets** セクションの直後に識別されます。

throughput-optimized と呼ばれる配置ターゲットをデフォルトターゲットとして **throughput-optimized** とすると、ゾーングループ構成の **placement_targets** セクションと **default_placement** 設定を次のように変更する必要があります。

```
{
  ...
  "placement_targets": [
     {
       "name": "throughput-optimized",
       "tags": []
     }
  ],
  "default_placement": "throughput-optimized",
  ...
}
```

最後に、変更した **zonegroup.json** ファイルの設定でゾーングループ構成を設定し、期間を更新します。以下に例を示します。

```
# radosgw-admin zonegroup set [--rgw-zonegroup=<zonegroup>] --infile zonegroup.json
# radosgw-admin period update --commit
```

7.5.5. ゾーンへの配置プールの設定

ゾーングループに新しい **throughput-optimized** 配置ターゲットが割り当てられたら、ゾーン設定で **throughput-optimized** の配置プールをマッピングします。この手順では、関連するプールへの **default-placement** のマッピングが、配置グループの **throughput-optimized** セットに置き換えられます。

「ゾーンの取得」の手順を実行し、プール名を表示します。

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

us-west という名前のゾーンを前提とすると、ファイルの内容は以下のようになります。

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

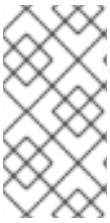
```
                "data_extra_pool": ".us-west.rgw.buckets.non-ec"
                "index_type": 0
              }
    }
  ]
}
```

ゾーン設定の **placement_pools** セクションは、配置プールのセットを定義します。配置プールの各セットは、ストレージポリシーを定義します。ファイルを変更して **default-placement** エントリーを削除し、**throughput-optimized** エントリーを、前のステップで作成したプールに置き換えます。以下に例を示します。

```
{
...
"placement_pools": [
    {  "key": "throughput-optimized",
       "val": { "index_pool": ".us-west.rgw.buckets.index",
                "data_pool": ".us-west.rgw.buckets.throughput"}
                "data_extra_pool": ".us-west.rgw.buckets.non-ec",
                "index_type": 0
    }
  ]
}
```

最後に、変更した **zone.json** ファイルからゾーン設定を設定し、期間を更新します。以下に例を示します。

```
# radosgw-admin zone set --rgw-zone={zone-name} --infile zone.json
# radosgw-admin period update --commit
```



注記

index_pool は、SSD や他の高パフォーマンスストレージを含むインデックスプールおよび CRUSH 階層を参照し、**data_pool** は PG に完全補完されたプール、高スループットのホストバスアダプターの CRUSH 階層、ジャーナル用の SAS ドライブおよび SSD を参照します。

7.5.6. データ配置の概要

クライアント要求を処理する際に、Ceph Object Gateway はデフォルトのストレージポリシーとして新たな **throughput-optimized** ターゲットを使用します。以下の手順を使用して、マルチサイト構成で異なるゾーンおよびゾーングループに同じターゲットを確立し、必要に応じてプールのゾーン名を置き換えます。

以下の手順を使用して、追加のストレージポリシーを確立します。各ターゲットおよび配置プールのセットの命名は任意です。これには、**fast**、**streaming**、**cold-storage**、またはその他の適切な名前を使用できます。ただし、各セットには、ゾーングループの **placement_targets** の下に対応するエントリーが必要であり、ターゲットの 1 つは **default_placement** 設定で参照される **必要があります**。また、ゾーンには、ポリシーごとに構成された対応するプールのセットが必要です。

クライアント要求が **X-Storage-Policy** と別のターゲットを指定しない限り、クライアントリクエストは常にデフォルトのターゲットを使用します。オブジェクトゲートウェイクライアントの使用例は、「コンテナーの作成」を参照してください。

第8章 ゲートウェイの設定

実稼働環境用に Ceph Object Gateway を準備する最後のステップでは、Civetweb、ファイアウォールポート、DNS およびロードバランサーを設定する必要があります。トピックには以下が含まれます。

- Civetweb の設定
- ファイアウォールポートの設定
- DNS ワイルドカードの設定
- ロードバランサーの設定

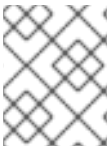
8.1. CIVETWEB の設定

Ceph Object Gateway の「インストール」時に選択した内容に応じて、Ceph 構成ファイルには 「レルムの作成」に関連する手順からの追加の変更を加えた Ceph ObjectGateway の各インスタンスのエントリーがすでに含まれています。

デフォルト設定の最も一般的な設定変更は、Ansible が設定したデフォルトのポート **8080** を別のポート (**80** など) に変更することです。「CivetWeb ポートの変更」を参照してください。

Civetweb に特有の追加設定があります。詳細は、「Civetweb 設定オプション」を参照してください。

上書きされる可能性のある追加設定があります。詳細は、「Object Gateway 設定リファレンス」を参照してください。



注記

その他のユースケースのセクションは、サードパーティーのコンポーネントで Ceph Object Gateway を使用するための詳細な設定例を提供します。

8.2. ファイアウォールポートの設定

Civetweb のデフォルトポートを変更する場合は、クライアントアクセス用に対応するポートが開いていることを確認してください。詳細は、Red Hat Enterprise Linux の『Red Hat Ceph Storage 4 インストールガイド』の「Red Hat Ceph Storage 用ファイアウォールの設定」 セクションを参照してください。

8.3. DNS ワイルドカードの設定

S3 スタイルのサブドメインには、バケット名が CNAME 拡張として組み込まれています。ワイルドカードを DNS に追加して、S3 スタイルのサブドメインを容易にします。詳細は、Red Hat Ceph Storage 4 の『Ceph Object Gateway 設定および管理ガイド』の「DNS へのワイルドカードの追加」セクションを参照してください。

8.4. ロードバランサーの設定

ゾーンには通常、実稼働の負荷を処理し、高可用性を維持するために Ceph Object Gateway のインスタンスが複数含まれます。実稼働クラスターは通常、ロードバランサーを使用してゲートウェイインスタンス間でリクエストを割り当てます。

また、以前の Civetweb バージョンは HTTPS をサポートしません。ロードバランサーは、SSL リクエストを受け入れ、SSL 接続を終了し、HTTP 経由でリクエストをゲートウェイインスタンスに渡すように設定できます。

Ceph Storage は高可用性を維持することを目的としています。このため、Red Hat は HAProxy または keepalived を使用することを推奨しています。詳細は、Ceph Object Gateway の『設定および管理ガイド』の「HAProxy/keepalived の設定」セクションを参照してください。

8.5. BEAST フロントエンドの使用

Ceph Object Gateway は、CivetWeb および Beast 埋め込み HTTP サーバーをフロントエンドとして提供します。Beast フロントエンドは、HTTP 解析に **Boost.Beast** ライブラリーを使用し、非同期ネットワーク I/O に **Boost.Asio** ライブラリーを使用します。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

8.6. BEAST 設定オプション

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

オプション	詳細	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

オプション	詳細	デフォルト
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

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。

第9章 その他のユースケース

クラスターが稼働したら、追加のユースケースを考慮する必要があります。

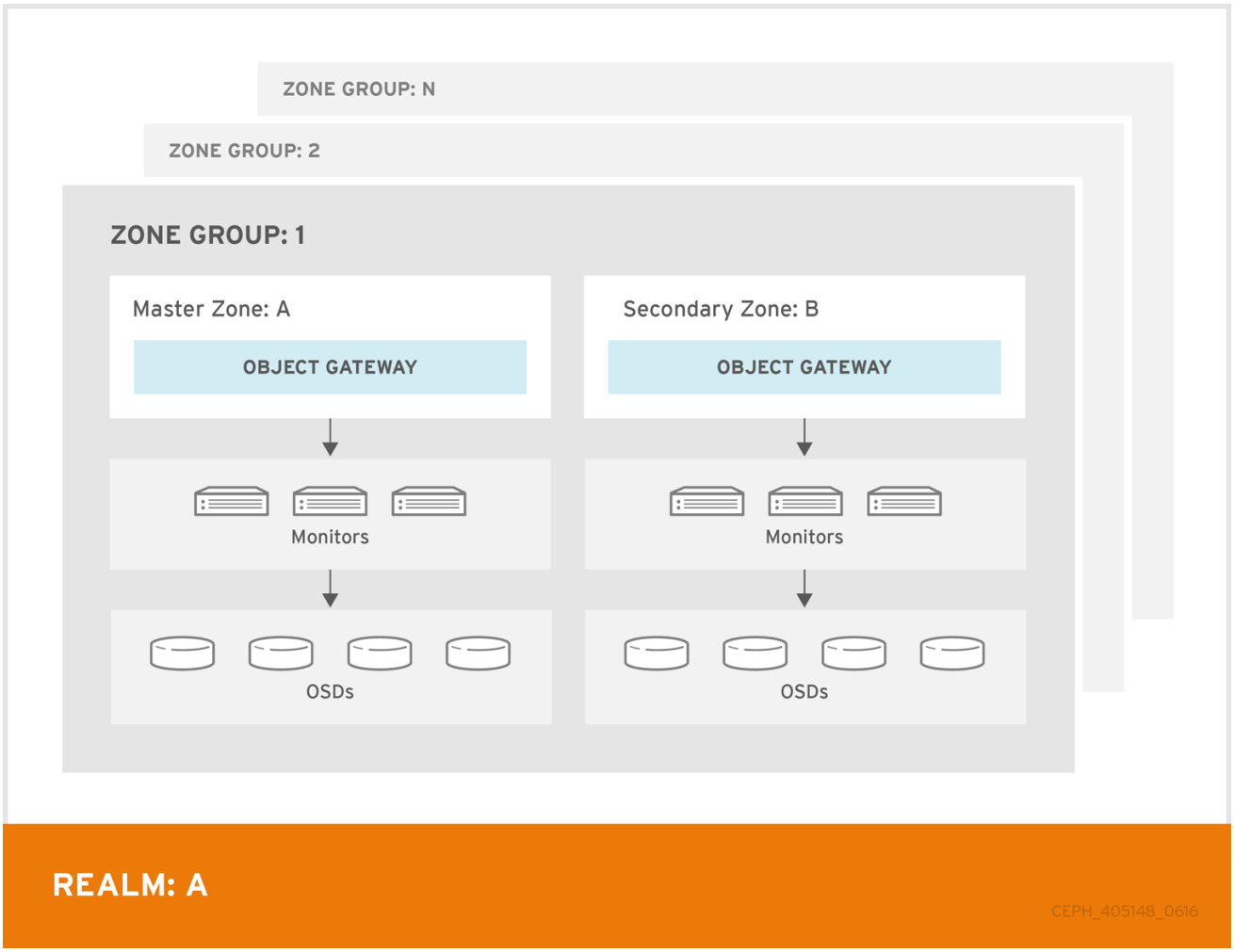
- マルチサイトを使用したクラスターの拡張
- NFS-Ganesha を使用したデータの移行
- 静的 Web ホスト用のクラスターの設定
- LDAP/AD を使用したクラスターの設定
- Keystone を使用するようにクラスターを設定

9.1. マルチサイトを使用したクラスターの拡張

ストレージ計画を開発する場合、レルムを作成する手順により、クラスターが独自のレルム、マスターゾーングループ、およびマスターゾーンでマルチサイトを使用するように設定されます。

通常の実稼働クラスターには、障害の発生時にバックアップとして機能するための別の物理的な場所に、独自の Ceph Storage クラスターを持つセカンダリーゾーンがあります。セカンダリーゾーンを設定するには、本ガイドの手順を繰り返します。通常、セカンダリーゾーンはマスターゾーンと同じハードウェア設定とサイジングである必要があります。詳細は、「セカンダリーゾーンの設定」を参照してください。

セカンダリーゾーンを追加すると、「フェイルオーバーおよび災害リカバリー」機能がクラスターに追加されます。



9.2. NFS GANESHA を使用したデータの移行

Ceph Object Gateway および Ceph Storage クラスターがファイルシステムベースのストレージソリューションを置き換える場合は、Ceph の NFS-Ganesha ソリューションを使用してファイルシステムから Ceph Object Gateway にデータを移行することを検討してください。

Ceph Object Gateway の『設定および管理ガイド』の「名前空間の NFS-Ganesha へのエクスポート」の章を参照してください。

9.3. 静的 WEB ホスト用のクラスターの設定

従来の Web ホストでは、各 Web サイトに Web サーバーを設定しなければならない場合があります。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。

詳細は、「静的 Web ホスト用のゲートウェイの設定」を参照してください。

9.4. LDAP/AD のクラスターの設定

Ceph Object Gateway をユーザーおよびアプリケーション用にデプロイしている組織は、Ceph Object Gateway ユーザーを作成する代わりに、Light-weight Directory Access Protocol (LDAP)または Microsoft Active Directory (AD)を使用して Ceph Object Gateway との認証を行うことを選択できます。

LDAP/AD を使用すると、Ceph Object Gateway は組織の LDAP/AD シングルサインオンシステムと統合できます。

詳細は、Red Hat Ceph Storage 4 の「LDAP/AD を使用する Ceph Object Gateway ガイド」 を参照してください。

9.5. OPENSTACK KEYSTONE を使用するクラスターの設定

OpenStack Swift の代わりに Ceph Object Gateway をデプロイする場合、Ceph Object Gateway ユーザーを作成する代わりに、OpenStack Keystone を使用してユーザーを認証するようにゲートウェイを設定することができます。

詳細は、Red Hat Ceph Storage 4 の『Keystone を使用した Ceph Object Gateway ユーザーの使用』 を参照してください。

第10章 NVME と LVM の最適な使用

1. 要約
   以下の手順では、高速の NVMe ベースの SSD を使用する場合に、Object Gateway の使用に Ceph をデプロイする方法を示しています（これは SATA SSD にも適用されます）。ジャーナルとバケットインデックスは、共に高速ストレージデバイスに置かれるため、1 つのデバイスにすべてのジャーナルがある場合よりもパフォーマンスを向上させることができます。この構成では、**osd_scenario** を **lvm** に設定する必要があります。

   2 つの設定例の手順を以下に示します。

   - 1 つのバケットインデックスを使用する 1 つの NVMe デバイスおよび 4 つ以上の HDD: 1 つの NVMe デバイス
   - 2 つの NVMe デバイスと 2 つのバケットインデックスを使用する少なくとも 4 つのHDD: 2 つの NVMe デバイス

2. 詳細
   最も基本的な Ceph 設定では、**collocated** の **osd_scenario** 設定を使用します。これにより、1 つのストレージデバイスに OSD データとそのジャーナルが共に保存されます（これは「コロケーション」です）。一般的なサーバー設定には、HDD と SSD の両方が含まれます。HDD は通常 SSD よりも大きいため、ほとんどのストレージ領域を活用するために、共存する構成では HDD が選択され、OSD データとジャーナルの両方をそこに配置します。ただし、ジャーナルはより高速 SSD にあることが理想的です。別のオプションとして、**non-collocated** の **osd_scenario** 設定を使用します。これにより、ジャーナル専用のデバイスを設定できるため、OSD データを HDD に、ジャーナルを SSD に配置することができます。

   OSD データおよびジャーナルの他に、Object Gateway を使用する場合はバケットインデックスをデバイス上に保存する必要があります。この場合、HDD が OSD データを保持し、1 つの SSD がジャーナルを保持し、別の SSD がバケットインデックスを保持するようにCeph が設定されます。これにより、すべてのジャーナルを保持する SSD が飽和し、一方でバケットインデックスを保持する SSD の使用率が低くなり、非常に不均衡な状態が生じる可能性があります。

   解決策は、**osd_scenario** を **lvm** に設定し、論理ボリュームマネージャー (LVM) を使用して、複数の目的で 1 つの SSD デバイスを分割することです。これにより、1 つのデバイスにジャーナルとバケットインデックスを隣りあわせて存在させることができます。最も重要な点として、ジャーナルは複数の SSD 上に存在でき、ジャーナルの集中した IO データの転送を複数のデバイスに分散させることができます。

   Ceph のインストールに使用される **ceph-ansible** RPM が提供する通常の Ansible Playbook (site.yml、osds.yml など) は、複数の目的で 1 つのデバイスを使用することをサポートしません。

   今後、通常の Ansible Playbook は複数の目的で 1 つのデバイスの使用をサポートします。その間、SSD 使用率を最適化するのに必要な論理ボリューム (LV) の作成を容易にするために、Playbook の **lv-create.yml** および **lv-vars.yaml** が提供されます。**lv-create.yml** を実行すると、**site.yml** が正常に実行でき、新たに作成された論理ボリュームが使用されます。

重要

以下の手順は、新しい BlueStore ストレージバックエンドではなく、FileStore ストレージバックエンドにのみ適用されます。

10.1. 1 つの NVME デバイスの使用

1 つの NVMe デバイスで Object Gateway を使用するために Ceph をデプロイするには、以下の手順に従います。

10.1.1. すべての既存 Ceph クラスターの削除

Ceph がすでに設定されている場合は、最初からやり直すために削除してください。この目的のために、Ansible Playbook (**purge-cluster.yml**) が提供されます。

```
$ ansible-playbook infrastructure-playbooks/purge-cluster.yml -i hosts
```

purge-cluster.yml の使用方法は、『インストールガイド』の「**Ansible を使用した Ceph クラスターのパージ**」を参照してください。

重要

以下の手順に従って Ceph を再デプロイするためにサーバーを準備するには、クラスターのパージだけでは十分でない可能性があります。Ceph が使用するストレージデバイス上のファイルシステム、GPT、RAID、またはその他の署名があると問題が発生する可能性があります。**wipefs** を使用して署名を削除する手順は、「Ansible Playbook lv-create.yml の実行」に記載されています。

10.1.2. 通常インストール用のクラスター設定

NVMe や LVM の考慮事項以外に、通常通りクラスターを設定しますが、Ansible Playbook を実行する前に停止します。その後、クラスターのインストール設定は、Object Gateway をサポートするために、特に NVMe/LVM の使用を最適化するために調整されます。その時のみ Ansible Playbook を実行する必要があります。

通常のインストール用にストレージクラスターを設定するには、『Red Hat Ceph Storage インストールガイド』を参照してください。特に、Ansible ログディレクトリーを作成する手順 9 での「Red Hat Ceph Storage Cluster のインストール」の手順を完了します。**ansible-playbook site.yml -i hosts** の実行時に、手順 10 の前に停止します。

重要

ここから、「Ceph for NVMe のインストールと成功の確認」までの手順がすべて完了するまで、**ansible-playbook site.yml -i hosts** を実行しないでください。

10.1.3. NVMe デバイスおよび HDD デバイスの特定

lsblk を使用して、サーバーに接続されている NVMe デバイスおよび HDD デバイスを特定します。**lsblk** からの出力例は次のとおりです。

```
[root@c04-h05-6048r ~]# lsblk
NAME    MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
sda       8:0    0 465.8G  0 disk
├─sda1    8:1    0     4G  0 part
│ └─md1   9:1    0     4G  0 raid1 [SWAP]
├─sda2    8:2    0   512M  0 part
│ └─md0   9:0    0   512M  0 raid1 /boot
└─sda3    8:3    0 461.3G  0 part
  └─md2   9:2    0 461.1G  0 raid1 /
sdb       8:16   0 465.8G  0 disk
├─sdb1    8:17   0     4G  0 part
```

```
│ └─md1   9:1    0     4G  0 raid1 [SWAP]
├─sdb2    8:18   0   512M  0 part
│ └─md0   9:0    0   512M  0 raid1 /boot
└─sdb3    8:19   0 461.3G  0 part
  └─md2   9:2    0 461.1G  0 raid1 /
sdc       8:32   0   1.8T  0 disk
sdd       8:48   0   1.8T  0 disk
sde       8:64   0   1.8T  0 disk
sdf       8:80   0   1.8T  0 disk
sdg       8:96   0   1.8T  0 disk
sdh       8:112  0   1.8T  0 disk
sdi       8:128  0   1.8T  0 disk
sdj       8:144  0   1.8T  0 disk
sdk       8:160  0   1.8T  0 disk
sdl       8:176  0   1.8T  0 disk
sdm       8:192  0   1.8T  0 disk
sdn       8:208  0   1.8T  0 disk
sdo       8:224  0   1.8T  0 disk
sdp       8:240  0   1.8T  0 disk
sdq       65:0   0   1.8T  0 disk
sdr       65:16  0   1.8T  0 disk
sds       65:32  0   1.8T  0 disk
sdt       65:48  0   1.8T  0 disk
sdu       65:64  0   1.8T  0 disk
sdv       65:80  0   1.8T  0 disk
sdw       65:96  0   1.8T  0 disk
sdx       65:112 0   1.8T  0 disk
sdy       65:128 0   1.8T  0 disk
sdz       65:144 0   1.8T  0 disk
sdaa      65:160 0   1.8T  0 disk
sdab      65:176 0   1.8T  0 disk
sdac      65:192 0   1.8T  0 disk
sdad      65:208 0   1.8T  0 disk
sdae      65:224 0   1.8T  0 disk
sdaf      65:240 0   1.8T  0 disk
sdag      66:0   0   1.8T  0 disk
sdah      66:16  0   1.8T  0 disk
sdai      66:32  0   1.8T  0 disk
sdaj      66:48  0   1.8T  0 disk
sdak      66:64  0   1.8T  0 disk
sdal      66:80  0   1.8T  0 disk
nvme0n1 259:0    0 745.2G  0 disk
nvme1n1 259:1    0 745.2G  0 disk
```

この例では、以下の未処理のブロックデバイスが使用されます。

NVMe デバイス

1. **/dev/nvme0n1**

HDD デバイス

1. **/dev/sdc**
2. **/dev/sdd**

3. **/dev/sde**

4. **/dev/sdf**

lv_vars.yaml ファイルは、選択したデバイスで論理ボリュームの作成を設定します。NVMe、NVMe ベースのバケットインデックス、および HDD ベースの OSD にジャーナルを作成します。実際の論理ボリュームの作成は、**lv_vars.yaml** を読み込む **lv-create.yml** により開始されます。

このファイルでは、1 度に 1 つの NVMe デバイスのみが参照される必要があります。2 つの NVMe デバイスで Ceph を最適に使用する方法は、「2 つの NVMe デバイスの使用」 を参照してください。

10.1.4. lv_vars.yaml へのデバイスの追加

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

```
# cd /usr/share/ceph-ansible
```

2. 以下の行が含まれるようにファイルを編集します。

```
nvme_device: /dev/nvme0n1
hdd_devices:
  - /dev/sdc
  - /dev/sdd
  - /dev/sde
  - /dev/sdf
```

10.1.5. lv-create.yml Ansible Playbook の実行

Playbook **lv-create.yml** の目的は、単一の NVMe にオブジェクトゲートウェイバケットインデックスおよびジャーナルに論理ボリュームを作成することです。これは、**osd_scenario=lvm** を使用して行います。Ansible Playbook の **lv-create.yml** により、複雑な LVM の作成および設定を自動化することで Ceph を簡単に設定することができます。

1. ストレージデバイスが未処理であることを確認します。
lv-create.yml を実行して、NVMe デバイスおよび HDD デバイスに論理ボリュームを作成する前に、ファイルシステム、GPT、RAID、その他の署名がないことを確認してください。

   生でない場合は、**lv-create.yml** を実行すると、以下のエラーで失敗する可能性があります。

```
device /dev/sdc excluded by a filter
```

2. ストレージデバイスの署名を消去します（オプション）。
デバイスに署名がある場合は、**wipefs** を使用して消去できます。

   wipefs を使用してデバイスを消去する例を以下に示します。

```
[root@c04-h01-6048r ~]# wipefs -a /dev/sdc
/dev/sdc: 8 bytes were erased at offset 0x00000200 (gpt): 45 46 49 20 50 41 52 54
/dev/sdc: 8 bytes were erased at offset 0x1d19ffffe00 (gpt): 45 46 49 20 50 41 52 54
/dev/sdc: 2 bytes were erased at offset 0x000001fe (PMBR): 55 aa
/dev/sdc: calling ioclt to re-read partition table: Success
[root@c04-h01-6048r ~]# wipefs -a /dev/sdd
/dev/sdd: 8 bytes were erased at offset 0x00000200 (gpt): 45 46 49 20 50 41 52 54
/dev/sdd: 8 bytes were erased at offset 0x1d19ffffe00 (gpt): 45 46 49 20 50 41 52 54
```

```
/dev/sdd: 2 bytes were erased at offset 0x000001fe (PMBR): 55 aa
/dev/sdd: calling ioclt to re-read partition table: Success
[root@c04-h01-6048r ~]# wipefs -a /dev/sde
/dev/sde: 8 bytes were erased at offset 0x00000200 (gpt): 45 46 49 20 50 41 52 54
/dev/sde: 8 bytes were erased at offset 0x1d19ffffe00 (gpt): 45 46 49 20 50 41 52 54
/dev/sde: 2 bytes were erased at offset 0x000001fe (PMBR): 55 aa
/dev/sde: calling ioclt to re-read partition table: Success
[root@c04-h01-6048r ~]# wipefs -a /dev/sdf
/dev/sdf: 8 bytes were erased at offset 0x00000200 (gpt): 45 46 49 20 50 41 52 54
/dev/sdf: 8 bytes were erased at offset 0x1d19ffffe00 (gpt): 45 46 49 20 50 41 52 54
/dev/sdf: 2 bytes were erased at offset 0x000001fe (PMBR): 55 aa
/dev/sdf: calling ioclt to re-read partition table: Success
```

3. Ansible Playbook の **lv-teardown.yml** を実行します。
lv-create.yml を実行する前に、常に **lv-teardown.yml** を実行します。

   Ansible Playbook の **lv-teardown.yml** を実行します。

```
$ ansible-playbook infrastructure-playbooks/lv-teardown.yml -i hosts
```

警告

Ansible スクリプト **lv-teardown.yml** を実行する場合は、注意が必要です。データが破棄されます。重要なデータのバックアップを作成してください。

4. Ansible Playbook の **lv-create.yml** を実行します。

```
$ ansible-playbook infrastructure-playbooks/lv-create.yml -i hosts
```

5. エラーなしで **lv-create.yml** が完了すると、次のセクションを続行して、適切に機能していることを確認します。

10.1.6. LVM 設定の確認

1. **lv-created.log** を確認します。
Ansible Playbook の **lv-create.yml** が正常に完了すると、設定情報が **lv-created.log** に書き込まれます。この情報は後で **group_vars/osds.yml** にコピーされます。**lv-created.log** を開き、以下の例のような情報を探します。

```
- data: ceph-bucket-index-1
  data_vg: ceph-nvme-vg-nvme0n1
  journal: ceph-journal-bucket-index-1-nvme0n1
  journal_vg: ceph-nvme-vg-nvme0n1
- data: ceph-hdd-lv-sdc
  data_vg: ceph-hdd-vg-sdc
  journal: ceph-journal-sdc
  journal_vg: ceph-nvme-vg-nvme0n1
- data: ceph-hdd-lv-sdd
```

```
    data_vg: ceph-hdd-vg-sdd
    journal: ceph-journal-sdd
    journal_vg: ceph-nvme-vg-nvme0n1
  - data: ceph-hdd-lv-sde
    data_vg: ceph-hdd-vg-sde
    journal: ceph-journal-sde
    journal_vg: ceph-nvme-vg-nvme0n1
  - data: ceph-hdd-lv-sdf
    data_vg: ceph-hdd-vg-sdf
    journal: ceph-journal-sdf
    journal_vg: ceph-nvme-vg-nvme0n1
```

2. LVM 設定を確認します。

1つの NVMe デバイスと 4 つの HDD の例に基づいて、以下の論理ボリューム(LV)を作成する必要があります。

HDD ごとに 1 つのジャーナル LV を NVMe に配置（/dev/nvme0n1 に 4 つの LV）

HDD ごとに 1 つのデータ LV を各 HDD に配置（HDD ごとに 1 つの LV）

バケットインデックスごとに 1 つのジャーナル LV を NVMe に配置（/dev/nvme0n1 に 1 つの LV）

バケットインデックスごとに 1 つのデータ LV を NVMe に配置（/dev/nvme0n1 に 1 つの LV）

LV は、**lsblk** および **lvscan** の出力で確認できます。上記の例では、Ceph に 10 の LV があるはずです。簡単な正常性チェックとして、Ceph LV をカウントして、少なくとも 10 個あることを確認できますが、理想的には、適切な数の LV が正しいストレージデバイス (NVMe / HDD) に作成されたことを確認してください。

lsblk からの出力例を以下に示します。

```
[root@c04-h01-6048r ~]# lsblk
NAME                                  MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
sda                                     8:0    0 465.8G  0 disk
├─sda1                                  8:1    0     4G  0 part
│ └─md1                                 9:1    0     4G  0 raid1 [SWAP]
├─sda2                                  8:2    0   512M  0 part
│ └─md0                                 9:0    0   512M  0 raid1 /boot
└─sda3                                  8:3    0 461.3G  0 part
  └─md2                                 9:2    0 461.1G  0 raid1 /
sdb                                     8:16   0 465.8G  0 disk
├─sdb1                                  8:17   0     4G  0 part
│ └─md1                                 9:1    0     4G  0 raid1 [SWAP]
├─sdb2                                  8:18   0   512M  0 part
│ └─md0                                 9:0    0   512M  0 raid1 /boot
└─sdb3                                  8:19   0 461.3G  0 part
  └─md2                                 9:2    0 461.1G  0 raid1 /
sdc                                     8:32   0   1.8T  0 disk
└─ceph--hdd--vg--sdc-ceph--hdd--lv--sdc                   253:6    0   1.8T  0 lvm
sdd                                     8:48   0   1.8T  0 disk
└─ceph--hdd--vg--sdd-ceph--hdd--lv--sdd                   253:7    0   1.8T  0 lvm
sde                                     8:64   0   1.8T  0 disk
└─ceph--hdd--vg--sde-ceph--hdd--lv--sde                   253:8    0   1.8T  0 lvm
sdf                                     8:80   0   1.8T  0 disk
```

```
└─ceph--hdd--vg--sdf-ceph--hdd--lv--sdf                    253:9    0   1.8T  0 lvm
sdg                                                         8:96   0   1.8T  0 disk
sdh                                                         8:112  0   1.8T  0 disk
sdi                                                         8:128  0   1.8T  0 disk
sdj                                                         8:144  0   1.8T  0 disk
sdk                                                         8:160  0   1.8T  0 disk
sdl                                                         8:176  0   1.8T  0 disk
sdm                                                         8:192  0   1.8T  0 disk
sdn                                                         8:208  0   1.8T  0 disk
sdo                                                         8:224  0   1.8T  0 disk
sdp                                                         8:240  0   1.8T  0 disk
sdq                                                        65:0    0   1.8T  0 disk
sdr                                                        65:16   0   1.8T  0 disk
sds                                                        65:32   0   1.8T  0 disk
sdt                                                        65:48   0   1.8T  0 disk
sdu                                                        65:64   0   1.8T  0 disk
sdv                                                        65:80   0   1.8T  0 disk
sdw                                                        65:96   0   1.8T  0 disk
sdx                                                        65:112  0   1.8T  0 disk
sdy                                                        65:128  0   1.8T  0 disk
sdz                                                        65:144  0   1.8T  0 disk
sdaa                                                       65:160  0   1.8T  0 disk
sdab                                                       65:176  0   1.8T  0 disk
sdac                                                       65:192  0   1.8T  0 disk
sdad                                                       65:208  0   1.8T  0 disk
sdae                                                       65:224  0   1.8T  0 disk
sdaf                                                       65:240  0   1.8T  0 disk
sdag                                                       66:0    0   1.8T  0 disk
sdah                                                       66:16   0   1.8T  0 disk
sdai                                                       66:32   0   1.8T  0 disk
sdaj                                                       66:48   0   1.8T  0 disk
sdak                                                       66:64   0   1.8T  0 disk
sdal                                                       66:80   0   1.8T  0 disk
nvme0n1                                                   259:0    0 745.2G  0 disk
├─ceph--nvme--vg--nvme0n1-ceph--journal--bucket--index--1--nvme0n1 253:0    0   5.4G  0
lvm
├─ceph--nvme--vg--nvme0n1-ceph--journal--sdc                253:1    0   5.4G  0 lvm
├─ceph--nvme--vg--nvme0n1-ceph--journal--sdd                253:2    0   5.4G  0 lvm
├─ceph--nvme--vg--nvme0n1-ceph--journal--sde                253:3    0   5.4G  0 lvm
├─ceph--nvme--vg--nvme0n1-ceph--journal--sdf                253:4    0   5.4G  0 lvm
└─ceph--nvme--vg--nvme0n1-ceph--bucket--index--1            253:5    0 718.4G  0 lvm
nvme1n1                                                   259:1    0 745.2G  0 disk
```

以下は、**lvscan** の出力例です。

```
[root@c04-h01-6048r ~]# lvscan
  ACTIVE            '/dev/ceph-hdd-vg-sdf/ceph-hdd-lv-sdf' [<1.82 TiB] inherit
  ACTIVE            '/dev/ceph-hdd-vg-sde/ceph-hdd-lv-sde' [<1.82 TiB] inherit
  ACTIVE            '/dev/ceph-hdd-vg-sdd/ceph-hdd-lv-sdd' [<1.82 TiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme0n1/ceph-journal-bucket-index-1-nvme0n1' [5.37
GiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme0n1/ceph-journal-sdc' [5.37 GiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme0n1/ceph-journal-sdd' [5.37 GiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme0n1/ceph-journal-sde' [5.37 GiB] inherit
```

```
  ACTIVE            '/dev/ceph-nvme-vg-nvme0n1/ceph-journal-sdf' [5.37 GiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme0n1/ceph-bucket-index-1' [<718.36 GiB] inherit
  ACTIVE            '/dev/ceph-hdd-vg-sdc/ceph-hdd-lv-sdc' [<1.82 TiB] inherit
```

10.1.7. osds.yml および all.yml Ansible Playbooks の編集

1. 前述の構成情報を **lv-created.log** から **lvm_volumes:** 行の下の**group_vars/osds.yml** にコピーします。

2. **osd_scenario:** を **lvm** に設定します。

```
osd_scenario: lvm
```

3. **all.yml** および **osds.yml** の **osd_objectstore: filestore** を設定します。
osds.yml ファイルは、以下のようになります。

```
# Variables here are applicable to all host groups NOT roles

osd_objectstore: filestore
osd_scenario: lvm
lvm_volumes:
  - data: ceph-bucket-index-1
    data_vg: ceph-nvme-vg-nvme0n1
    journal: ceph-journal-bucket-index-1-nvme0n1
    journal_vg: ceph-nvme-vg-nvme0n1
  - data: ceph-hdd-lv-sdc
    data_vg: ceph-hdd-vg-sdc
    journal: ceph-journal-sdc
    journal_vg: ceph-nvme-vg-nvme0n1
  - data: ceph-hdd-lv-sdd
    data_vg: ceph-hdd-vg-sdd
    journal: ceph-journal-sdd
    journal_vg: ceph-nvme-vg-nvme0n1
  - data: ceph-hdd-lv-sde
    data_vg: ceph-hdd-vg-sde
    journal: ceph-journal-sde
    journal_vg: ceph-nvme-vg-nvme0n1
  - data: ceph-hdd-lv-sdf
    data_vg: ceph-hdd-vg-sdf
    journal: ceph-journal-sdf
    journal_vg: ceph-nvme-vg-nvme0n1
```

10.1.8. NVMe 用の Ceph のインストールおよび正常な実行の確認

システムが NVMe と LVM を最適に使用するように Ceph を設定したら、これをインストールします。

1. Ansible Playbook の **site.yml** を実行して Ceph をインストールします。

```
$ ansible-playbook -v site.yml -i hosts
```

2. インストールの完了後に、Ceph が適切に実行されていることを確認します。

```
# ceph -s
```

```
# ceph osd tree
```

Ceph が適切に実行されていることを示す **ceph -s** の出力例：

```
# ceph -s
 cluster:
   id:     15d31a8c-3152-4fa2-8c4e-809b750924cd
   health: HEALTH_WARN
           Reduced data availability: 32 pgs inactive

 services:
   mon: 3 daemons, quorum b08-h03-r620,b08-h05-r620,b08-h06-r620
   mgr: b08-h03-r620(active), standbys: b08-h05-r620, b08-h06-r620
   osd: 35 osds: 35 up, 35 in

 data:
   pools:   4 pools, 32 pgs
   objects: 0 objects, 0 bytes
   usage:   0 kB used, 0 kB / 0 kB avail
   pgs:     100.000% pgs unknown
            32 unknown
```

Ceph が適切に実行されていることを示す **ceph osd tree** の出力例：

```
[root@c04-h01-6048r ~]# ceph osd tree
ID  CLASS WEIGHT   TYPE NAME             STATUS REWEIGHT PRI-AFF
 -1       55.81212 root default
-15        7.97316     host c04-h01-6048r
 13   hdd  1.81799         osd.13            up  1.00000 1.00000
 20   hdd  1.81799         osd.20            up  1.00000 1.00000
 26   hdd  1.81799         osd.26            up  1.00000 1.00000
 32   hdd  1.81799         osd.32            up  1.00000 1.00000
  6   ssd  0.70119         osd.6             up  1.00000 1.00000
 -3        7.97316     host c04-h05-6048r
 12   hdd  1.81799         osd.12            up  1.00000 1.00000
 17   hdd  1.81799         osd.17            up  1.00000 1.00000
 23   hdd  1.81799         osd.23            up  1.00000 1.00000
 29   hdd  1.81799         osd.29            up  1.00000 1.00000
  2   ssd  0.70119         osd.2             up  1.00000 1.00000
-13        7.97316     host c04-h09-6048r
 11   hdd  1.81799         osd.11            up  1.00000 1.00000
 16   hdd  1.81799         osd.16            up  1.00000 1.00000
 22   hdd  1.81799         osd.22            up  1.00000 1.00000
 27   hdd  1.81799         osd.27            up  1.00000 1.00000
  4   ssd  0.70119         osd.4             up  1.00000 1.00000
 -5        7.97316     host c04-h13-6048r
 10   hdd  1.81799         osd.10            up  1.00000 1.00000
 15   hdd  1.81799         osd.15            up  1.00000 1.00000
 21   hdd  1.81799         osd.21            up  1.00000 1.00000
 28   hdd  1.81799         osd.28            up  1.00000 1.00000
  1   ssd  0.70119         osd.1             up  1.00000 1.00000
 -9        7.97316     host c04-h21-6048r
  8   hdd  1.81799         osd.8             up  1.00000 1.00000
 18   hdd  1.81799         osd.18            up  1.00000 1.00000
 25   hdd  1.81799         osd.25            up  1.00000 1.00000
```

```
 30   hdd  1.81799        osd.30            up  1.00000 1.00000
  5   ssd  0.70119        osd.5             up  1.00000 1.00000
-11         7.97316     host c04-h25-6048r
  9   hdd  1.81799        osd.9             up  1.00000 1.00000
 14   hdd  1.81799        osd.14            up  1.00000 1.00000
 33   hdd  1.81799        osd.33            up  1.00000 1.00000
 34   hdd  1.81799        osd.34            up  1.00000 1.00000
  0   ssd  0.70119        osd.0             up  1.00000 1.00000
 -7         7.97316     host c04-h29-6048r
  7   hdd  1.81799        osd.7             up  1.00000 1.00000
 19   hdd  1.81799        osd.19            up  1.00000 1.00000
 24   hdd  1.81799        osd.24            up  1.00000 1.00000
 31   hdd  1.81799        osd.31            up  1.00000 1.00000
  3   ssd  0.70119        osd.3             up  1.00000 1.00000
```

Ceph Object Gateway 用に、1 つの NVMe デバイスおよび LVM を最適に使用するように Ceph が設定されました。

10.2. 2 つの NVME デバイスの使用

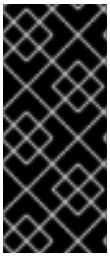
2 つの NVMe デバイスで Object Gateway を使用するために Ceph をデプロイするには、以下の手順に従います。

10.2.1. すべての既存 Ceph クラスターの削除

Ceph がすでに設定されている場合は、最初からやり直すために削除してください。この場合は、**purge-cluster.yml** という名前の Ansible Playbook ファイルが提供されます。

```
$ ansible-playbook purge-cluster.yml -i hosts
```

purge-cluster.yml の使用方法は、『インストールガイド』の**「Ansible を使用した Ceph クラスターのパージ」**を参照してください。



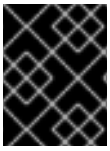
重要

以下の手順に従って Ceph を再デプロイするためにサーバーを準備するには、クラスターのパージだけでは十分でない可能性があります。Ceph が使用するストレージデバイス上のファイルシステム、GPT、RAID、またはその他の署名があると問題が発生する可能性があります。**wipefs** を使用して署名を削除する手順は、「Ansible Playbook lv-create.yml の実行」に記載されています。

10.2.2. 通常インストール用のクラスター設定

NVMe や LVM の考慮事項以外に、通常通りクラスターを設定しますが、Ansible Playbook を実行する前に停止します。その後、クラスターのインストール設定は、Object Gateway をサポートするために、特に NVMe/LVM の使用を最適化するために調整されます。その時のみ Ansible Playbook を実行する必要があります。

通常のインストール用にクラスターを設定するには、『インストールガイド』を参照してください。特に、Ansible ログディレクトリーを作成する手順 9 での「Red Hat Ceph Storage Cluster のインストール」の手順を完了します。**ansible-playbook site.yml -i hosts** を実行する前に、手順 10 を停止します。



重要

ここから、「Ceph for NVMe のインストールと成功の確認」までの手順がすべて完了するまで、**ansible-playbook site.yml -i hosts** を実行しないでください。

10.2.3. NVMe デバイスおよび HDD デバイスの特定

lsblk を使用して、サーバーに接続されている NVMe デバイスおよび HDD デバイスを特定します。**lsblk** からの出力例は次のとおりです。

```
[root@c04-h09-6048r ~]# lsblk
NAME                          MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda                             8:0    0 465.8G  0 disk
├─sda1                          8:1    0   512M  0 part /boot
└─sda2                          8:2    0 465.3G  0 part
  ├─vg_c04--h09--6048r-lv_root 253:0    0 464.8G  0 lvm  /
  └─vg_c04--h09--6048r-lv_swap 253:1    0   512M  0 lvm  [SWAP]
sdb                             8:16   0 465.8G  0 disk
sdc                             8:32   0   1.8T  0 disk
sdd                             8:48   0   1.8T  0 disk
sde                             8:64   0   1.8T  0 disk
sdf                             8:80   0   1.8T  0 disk
sdg                             8:96   0   1.8T  0 disk
sdh                             8:112  0   1.8T  0 disk
sdi                             8:128  0   1.8T  0 disk
sdj                             8:144  0   1.8T  0 disk
sdk                             8:160  0   1.8T  0 disk
sdl                             8:176  0   1.8T  0 disk
sdm                             8:192  0   1.8T  0 disk
sdn                             8:208  0   1.8T  0 disk
sdo                             8:224  0   1.8T  0 disk
sdp                             8:240  0   1.8T  0 disk
sdq                            65:0    0   1.8T  0 disk
sdr                            65:16   0   1.8T  0 disk
sds                            65:32   0   1.8T  0 disk
sdt                            65:48   0   1.8T  0 disk
sdu                            65:64   0   1.8T  0 disk
sdv                            65:80   0   1.8T  0 disk
sdw                            65:96   0   1.8T  0 disk
sdx                            65:112  0   1.8T  0 disk
sdy                            65:128  0   1.8T  0 disk
sdz                            65:144  0   1.8T  0 disk
sdaa                           65:160  0   1.8T  0 disk
sdab                           65:176  0   1.8T  0 disk
sdac                           65:192  0   1.8T  0 disk
sdad                           65:208  0   1.8T  0 disk
sdae                           65:224  0   1.8T  0 disk
sdaf                           65:240  0   1.8T  0 disk
sdag                           66:0    0   1.8T  0 disk
sdah                           66:16   0   1.8T  0 disk
sdai                           66:32   0   1.8T  0 disk
sdaj                           66:48   0   1.8T  0 disk
sdak                           66:64   0   1.8T  0 disk
sdal                           66:80   0   1.8T  0 disk
nvme0n1                       259:1    0 745.2G  0 disk
nvme1n1                       259:0    0 745.2G  0 disk
```

-

この例では、以下の未処理のブロックデバイスが使用されます。

NVMe デバイス

1. **/dev/nvme0n1**
2. **/dev/nvme1n1**

HDD デバイス

1. **/dev/sdc**
2. **/dev/sdd**
3. **/dev/sde**
4. **/dev/sdf**

lv_vars.yaml ファイルは、選択したデバイスで論理ボリュームの作成を設定します。NVMe、NVMe ベースのバケットインデックス、および HDD ベースの OSD にジャーナルを作成します。実際の論理ボリュームの作成は、**lv_vars.yaml** を読み込む **lv-create.yml** により開始されます。

このファイルでは、1 度に 1 つの NVMe デバイスのみが参照される必要があります。また、特定の NVMe デバイスに関連付けられる HDD デバイスのみを参照する必要もあります。複数の NVMe デバイスが含まれる OSD の場合は、各 NVMe の **lv_vars.yaml** を編集し、NVMe ごとに **lv-create.yml** を繰り返し実行します。これについては、以下で説明します。

この例では、**lv-create.yml** が最初に **/dev/nvme0n1** で実行され、その後 **/dev/nvme1n1** に再び実行されます。

10.2.4. lv_vars.yaml へのデバイスの追加

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

```
# cd /usr/share/ceph-ansible
```

2. **root** で Ansible Playbook **lv_vars.yaml** を現在のディレクトリーにコピーします。

```
# cp infrastructure-playbooks/vars/lv_vars.yaml .
```

3. 最初の実行では、次の行が含まれるようにファイルを編集します。

```
nvme_device: /dev/nvme0n1
hdd_devices:
  - /dev/sdc
  - /dev/sdd
```

ジャーナルサイズ、バケットインデックスの数、それらのサイズおよび名前、およびバケットインデックスのジャーナル名はすべて **lv_vars.yaml** で調整できます。詳細は、ファイル内のコメントを参照してください。

10.2.5. lv-create.yml Ansible Playbook の実行

Playbook **lv-create.yml** の目的は、単一の NVMe にオブジェクトゲートウェイバケットインデックスおよびジャーナルに論理ボリュームを作成することです。これは、**osd_scenario=non-collocated** ではなく **osd_scenario=lvm** を使用して行います。Ansible Playbook の **lv-create.yml** により、複雑な LVM の作成および設定を自動化することで Ceph を簡単に設定することができます。

1. **root** で、Ansible Playbook **lv-create.yml** を現在のディレクトリーにコピーします。

```
# cp infrastructure-playbooks/lv-create.yml .
```

2. ストレージデバイスが未処理であることを確認します。
lv-create.yml を実行して、NVMe デバイスおよび HDD デバイスに論理ボリュームを作成する前に、ファイルシステム、GPT、RAID、その他の署名がないことを確認してください。

   生でない場合には、**lv-create.yml** を実行すると、以下のエラーで失敗する可能性があります。

```
device /dev/sdc excluded by a filter
```

3. ストレージデバイスの署名を消去します（オプション）。
デバイスに署名がある場合は、**wipefs** を使用して消去できます。

   wipefs を使用してデバイスを消去する例を以下に示します。

```
[root@c04-h01-6048r ~]# wipefs -a /dev/sdc
/dev/sdc: 8 bytes were erased at offset 0x00000200 (gpt): 45 46 49 20 50 41 52 54
/dev/sdc: 8 bytes were erased at offset 0x1d19ffffe00 (gpt): 45 46 49 20 50 41 52 54
/dev/sdc: 2 bytes were erased at offset 0x000001fe (PMBR): 55 aa
/dev/sdc: calling ioclt to re-read partition table: Success
[root@c04-h01-6048r ~]# wipefs -a /dev/sdd
/dev/sdd: 8 bytes were erased at offset 0x00000200 (gpt): 45 46 49 20 50 41 52 54
/dev/sdd: 8 bytes were erased at offset 0x1d19ffffe00 (gpt): 45 46 49 20 50 41 52 54
/dev/sdd: 2 bytes were erased at offset 0x000001fe (PMBR): 55 aa
/dev/sdd: calling ioclt to re-read partition table: Success
[root@c04-h01-6048r ~]# wipefs -a /dev/sde
/dev/sde: 8 bytes were erased at offset 0x00000200 (gpt): 45 46 49 20 50 41 52 54
/dev/sde: 8 bytes were erased at offset 0x1d19ffffe00 (gpt): 45 46 49 20 50 41 52 54
/dev/sde: 2 bytes were erased at offset 0x000001fe (PMBR): 55 aa
/dev/sde: calling ioclt to re-read partition table: Success
[root@c04-h01-6048r ~]# wipefs -a /dev/sdf
/dev/sdf: 8 bytes were erased at offset 0x00000200 (gpt): 45 46 49 20 50 41 52 54
/dev/sdf: 8 bytes were erased at offset 0x1d19ffffe00 (gpt): 45 46 49 20 50 41 52 54
/dev/sdf: 2 bytes were erased at offset 0x000001fe (PMBR): 55 aa
/dev/sdf: calling ioclt to re-read partition table: Success
```

4. Ansible Playbook の **lv-teardown.yml** を実行します。
lv-create.yml を実行する前に、常に **lv-teardown.yml** を実行します。

   root で Ansible Playbook **lv-teardown.yml** を現在のディレクトリーにコピーします。

```
# cp infrastructure-playbooks/lv-teardown.yml .
```

   Ansible Playbook の **lv-teardown.yml** を実行します。

```
$ ansible-playbook lv-teardown.yml -i hosts
```

警告

Ansible スクリプト **lv-teardown.yml** を実行する場合は、注意が必要です。データが破棄されます。重要なデータのバックアップを作成してください。

5. Ansible Playbook の **lv-create.yml** を実行します。

```
$ ansible-playbook lv-create.yml -i hosts
```

10.2.6. 最初の NVMe LVM 設定のコピー

1. **lv-created.log** の確認
   Ansible Playbook の **lv-create.yml** が正常に完了すると、設定情報が **lv-created.log** に書き込まれます。**lv-created.log** を開き、以下の例のような情報を探します。

```
- data: ceph-bucket-index-1
  data_vg: ceph-nvme-vg-nvme0n1
  journal: ceph-journal-bucket-index-1-nvme0n1
  journal_vg: ceph-nvme-vg-nvme0n1
- data: ceph-hdd-lv-sdc
  data_vg: ceph-hdd-vg-sdc
  journal: ceph-journal-sdc
  journal_vg: ceph-nvme-vg-nvme0n1
- data: ceph-hdd-lv-sdd
  data_vg: ceph-hdd-vg-sdd
  journal: ceph-journal-sdd
  journal_vg: ceph-nvme-vg-nvme0n1
```

2. この情報を **lvm_volumes:** の下にある **group_vars/osds.yml** にコピーします。

10.2.7. NVMe デバイス 2 で Playbook lv-create.yml を実行します。

以下の手順は、2 番目の NVMe デバイスを設定する省略手順です。詳細なコンテキストが必要な場合は、上記の関連手順を参照してください。

1. **lv-vars.yaml** を、2 番目の NVMe および関連する HDD を使用するように変更します。
   この例では、**lv-vars.yaml** に以下のデバイスが設定されています。

```
nvme_device: /dev/nvme1n1
hdd_devices:
  - /dev/sde
  - /dev/sdf
```

2. **lv-teardown.yml** を実行します。

```
$ ansible-playbook lv-teardown.yml -i hosts
```

3. **lv-create.yml** を再度実行します。

```
$ ansible-playbook lv-create.yml -i hosts
```

10.2.8. 2 番目の NVMe LVM 設定のコピー

1. **lv-created.log** の確認
   Ansible Playbook の **lv-create.yml** が正常に完了すると、設定情報が **lv-created.log** に書き込まれます。**lv-created.log** を開き、以下の例のような情報を探します。

```
- data: ceph-bucket-index-1
  data_vg: ceph-nvme-vg-nvme1n1
  journal: ceph-journal-bucket-index-1-nvme1n1
  journal_vg: ceph-nvme-vg-nvme1n1
- data: ceph-hdd-lv-sde
  data_vg: ceph-hdd-vg-sde
  journal: ceph-journal-sde
  journal_vg: ceph-nvme-vg-nvme1n1
- data: ceph-hdd-lv-sdf
  data_vg: ceph-hdd-vg-sdf
  journal: ceph-journal-sdf
  journal_vg: ceph-nvme-vg-nvme1n1
```

2. この情報を、**lvm_volumes:** で入力した情報の下にある **group_vars/osds.yml** にコピーします。

10.2.9. LVM 設定の確認

1. LVM 設定の確認
   2 つの NVMe デバイスと 4 つの HDD の例に基づいて、以下の論理ボリューム(LV)を作成する必要があります。

   HDD ごとに 1 つのジャーナル LV を両方の NVMe デバイスに配置（/dev/nvme0n1 に 2 つの LVs、/dev/nvme1n1 に 2 つの LV）。

   HDD ごとに 1 つのデータ LV を各 HDD に配置（HDD ごとに 1 つの LV）

   バケットインデックスごとに 1 つのジャーナル LV を NVMe に配置（/dev/nvme0n1 に 1 つの LV、/dev/nvme1n1 に 1 つの LV）。

   バケットインデックスごとに 1 つのデータ LV を両方の NVMe デバイスに配置（/dev/nvme0n1 に 1 つの LV、/dev/nvme1n1 に 1 つの LV）。

   LV は、**lsblk** および **lvscan** の出力で確認できます。上記の例では、Ceph に 12 の LV があるはずです。簡単な正常性チェックとして、Ceph LV をカウントして、少なくとも 12 個あることを確認できますが、理想的には、適切な数の LV が正しいストレージデバイス (NVMe / HDD) に作成されたことを確認してください。

   lsblk からの出力例を以下に示します。

```
[root@c04-h01-6048r ~]# lsblk
NAME                                                  MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
sda                                                     8:0    0 465.8G  0 disk
├─sda1                                                  8:1    0     4G  0 part
```

```
│ └─md1                                                                  9:1    0     4G  0 raid1 [SWAP]
├─sda2                                                                   8:2    0   512M  0 part
│ └─md0                                                                  9:0    0   512M  0 raid1 /boot
└─sda3                                                                   8:3    0 461.3G  0 part
  └─md2                                                                  9:2    0 461.1G  0 raid1 /
sdb                                                                      8:16   0 465.8G  0 disk
├─sdb1                                                                   8:17   0     4G  0 part
│ └─md1                                                                  9:1    0     4G  0 raid1 [SWAP]
├─sdb2                                                                   8:18   0   512M  0 part
│ └─md0                                                                  9:0    0   512M  0 raid1 /boot
└─sdb3                                                                   8:19   0 461.3G  0 part
  └─md2                                                                  9:2    0 461.1G  0 raid1 /
sdc                                                                      8:32   0   1.8T  0 disk
└─ceph--hdd--vg--sdc-ceph--hdd--lv--sdc                                253:4    0   1.8T  0 lvm
sdd                                                                      8:48   0   1.8T  0 disk
└─ceph--hdd--vg--sdd-ceph--hdd--lv--sdd                                253:5    0   1.8T  0 lvm
sde                                                                      8:64   0   1.8T  0 disk
└─ceph--hdd--vg--sde-ceph--hdd--lv--sde                                253:10   0   1.8T  0 lvm
sdf                                                                      8:80   0   1.8T  0 disk
└─ceph--hdd--vg--sdf-ceph--hdd--lv--sdf                                253:11   0   1.8T  0 lvm
sdg                                                                      8:96   0   1.8T  0 disk
sdh                                                                      8:112  0   1.8T  0 disk
sdi                                                                      8:128  0   1.8T  0 disk
sdj                                                                      8:144  0   1.8T  0 disk
sdk                                                                      8:160  0   1.8T  0 disk
sdl                                                                      8:176  0   1.8T  0 disk
sdm                                                                      8:192  0   1.8T  0 disk
sdn                                                                      8:208  0   1.8T  0 disk
sdo                                                                      8:224  0   1.8T  0 disk
sdp                                                                      8:240  0   1.8T  0 disk
sdq                                                                      65:0   0   1.8T  0 disk
sdr                                                                      65:16  0   1.8T  0 disk
sds                                                                      65:32  0   1.8T  0 disk
sdt                                                                      65:48  0   1.8T  0 disk
sdu                                                                      65:64  0   1.8T  0 disk
sdv                                                                      65:80  0   1.8T  0 disk
sdw                                                                      65:96  0   1.8T  0 disk
sdx                                                                      65:112 0   1.8T  0 disk
sdy                                                                      65:128 0   1.8T  0 disk
sdz                                                                      65:144 0   1.8T  0 disk
sdaa                                                                     65:160 0   1.8T  0 disk
sdab                                                                     65:176 0   1.8T  0 disk
sdac                                                                     65:192 0   1.8T  0 disk
sdad                                                                     65:208 0   1.8T  0 disk
sdae                                                                     65:224 0   1.8T  0 disk
sdaf                                                                     65:240 0   1.8T  0 disk
sdag                                                                     66:0   0   1.8T  0 disk
sdah                                                                     66:16  0   1.8T  0 disk
sdai                                                                     66:32  0   1.8T  0 disk
sdaj                                                                     66:48  0   1.8T  0 disk
sdak                                                                     66:64  0   1.8T  0 disk
sdal                                                                     66:80  0   1.8T  0 disk
nvme0n1                                                                259:0    0 745.2G  0 disk
├─ceph--nvme--vg--nvme0n1-ceph--journal--bucket--index--1--nvme0n1 253:0    0   5.4G  0
lvm
├─ceph--nvme--vg--nvme0n1-ceph--journal--sdc                           253:1    0   5.4G  0 lvm
```

```
├─ceph--nvme--vg--nvme0n1-ceph--journal--sdd                    253:2    0   5.4G  0 lvm
└─ceph--nvme--vg--nvme0n1-ceph--bucket--index--1                253:3    0 729.1G  0 lvm
nvme1n1                                                 259:1    0 745.2G  0 disk
├─ceph--nvme--vg--nvme1n1-ceph--journal--bucket--index--1--nvme1n1 253:6    0   5.4G  0
lvm
├─ceph--nvme--vg--nvme1n1-ceph--journal--sde                    253:7    0   5.4G  0 lvm
├─ceph--nvme--vg--nvme1n1-ceph--journal--sdf                    253:8    0   5.4G  0 lvm
└─ceph--nvme--vg--nvme1n1-ceph--bucket--index--1                253:9    0 729.1G  0 lvm
```

lvscan からの出力例を以下に示します。

```
[root@c04-h01-6048r ~]# lvscan
  ACTIVE            '/dev/ceph-hdd-vg-sde/ceph-hdd-lv-sde' [<1.82 TiB] inherit
  ACTIVE            '/dev/ceph-hdd-vg-sdc/ceph-hdd-lv-sdc' [<1.82 TiB] inherit
  ACTIVE            '/dev/ceph-hdd-vg-sdf/ceph-hdd-lv-sdf' [<1.82 TiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme1n1/ceph-journal-bucket-index-1-nvme1n1' [5.37
GiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme1n1/ceph-journal-sde' [5.37 GiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme1n1/ceph-journal-sdf' [5.37 GiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme1n1/ceph-bucket-index-1' [<729.10 GiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme0n1/ceph-journal-bucket-index-1-nvme0n1' [5.37
GiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme0n1/ceph-journal-sdc' [5.37 GiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme0n1/ceph-journal-sdd' [5.37 GiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme0n1/ceph-bucket-index-1' [<729.10 GiB] inherit
  ACTIVE            '/dev/ceph-hdd-vg-sdd/ceph-hdd-lv-sdd' [<1.82 TiB] inherit
```

10.2.10. osds.yml および all.yml Ansible Playbooks の編集

1. **osd_objectstore** を **bluestore** に設定します。
   lv-create.log からの 2 番目の情報セットを **osds.yml** に追加する他に、**osd_objectstore** を **osds.yml** ファイルと **all.yml** ファイルの両方で **bluestore** に設定する必要があります。

   この行は、**osds.yml** と **all.yml** の両方で次のようになります。

   ```
   osd_objectstore: bluestore
   ```

2. **osds.yml** の **osd_scenario** を **lvm** に設定します。

   osds.yml ファイルは以下の例のようになります。

   ```
   # Variables here are applicable to all host groups NOT roles

   osd_objectstore: bluestore
   osd_scenario: lvm
   lvm_volumes:
     - data: ceph-bucket-index-1
       data_vg: ceph-nvme-vg-nvme0n1
       journal: ceph-journal-bucket-index-1-nvme0n1
       journal_vg: ceph-nvme-vg-nvme0n1
     - data: ceph-hdd-lv-sdc
       data_vg: ceph-hdd-vg-sdc
       journal: ceph-journal-sdc
       journal_vg: ceph-nvme-vg-nvme0n1
   ```

```
  - data: ceph-hdd-lv-sdd
    data_vg: ceph-hdd-vg-sdd
    journal: ceph-journal-sdd
    journal_vg: ceph-nvme-vg-nvme0n1
  - data: ceph-bucket-index-1
    data_vg: ceph-nvme-vg-nvme1n1
    journal: ceph-journal-bucket-index-1-nvme1n1
    journal_vg: ceph-nvme-vg-nvme1n1
  - data: ceph-hdd-lv-sde
    data_vg: ceph-hdd-vg-sde
    journal: ceph-journal-sde
    journal_vg: ceph-nvme-vg-nvme1n1
  - data: ceph-hdd-lv-sdf
    data_vg: ceph-hdd-vg-sdf
    journal: ceph-journal-sdf
    journal_vg: ceph-nvme-vg-nvme1n1
```

10.2.11. NVMe 用の Ceph のインストールおよび正常な実行の確認

1. Ansible Playbook の **site.yml** を実行して Ceph をインストールします。

```
$ ansible-playbook -v site.yml -i hosts
```

2. インストールの完了後に、Ceph が適切に実行されていることを確認します。

```
# ceph -s
```

```
# ceph osd tree
```

Ceph が適切に実行されていることを示す **ceph -s** の出力例：

```
# ceph -s
  cluster:
    id:     9ba22f4c-b53f-4c49-8c72-220aaf567c2b
    health: HEALTH_WARN
            Reduced data availability: 32 pgs inactive

  services:
    mon: 3 daemons, quorum b08-h03-r620,b08-h05-r620,b08-h06-r620
    mgr: b08-h03-r620(active), standbys: b08-h05-r620, b08-h06-r620
    osd: 42 osds: 42 up, 42 in

  data:
    pools:   4 pools, 32 pgs
    objects: 0 objects, 0 bytes
    usage:   0 kB used, 0 kB / 0 kB avail
    pgs:     100.000% pgs unknown
             32 unknown
```

Ceph が適切に実行されていることを示す **ceph osd tree** の出力例：

```
[root@c04-h01-6048r ~]# ceph osd tree
ID  CLASS WEIGHT   TYPE NAME             STATUS REWEIGHT PRI-AFF
 -1       60.86740 root default
```

```
 -7       8.69534     host c04-h01-6048r
 10   hdd  1.81799         osd.10            up  1.00000 1.00000
 13   hdd  1.81799         osd.13            up  1.00000 1.00000
 21   hdd  1.81799         osd.21            up  1.00000 1.00000
 27   hdd  1.81799         osd.27            up  1.00000 1.00000
  6   ssd  0.71169        osd.6             up  1.00000 1.00000
 15   ssd  0.71169         osd.15            up  1.00000 1.00000
 -3       8.69534     host c04-h05-6048r
  7   hdd  1.81799         osd.7             up  1.00000 1.00000
 20   hdd  1.81799         osd.20            up  1.00000 1.00000
 29   hdd  1.81799         osd.29            up  1.00000 1.00000
 38   hdd  1.81799         osd.38            up  1.00000 1.00000
  4   ssd  0.71169        osd.4             up  1.00000 1.00000
 25   ssd  0.71169         osd.25            up  1.00000 1.00000
-22       8.69534     host c04-h09-6048r
 17   hdd  1.81799         osd.17            up  1.00000 1.00000
 31   hdd  1.81799         osd.31            up  1.00000 1.00000
 35   hdd  1.81799         osd.35            up  1.00000 1.00000
 39   hdd  1.81799         osd.39            up  1.00000 1.00000
  5   ssd  0.71169        osd.5             up  1.00000 1.00000
 34   ssd  0.71169         osd.34            up  1.00000 1.00000
 -9       8.69534     host c04-h13-6048r
  8   hdd  1.81799         osd.8             up  1.00000 1.00000
 11   hdd  1.81799         osd.11            up  1.00000 1.00000
 30   hdd  1.81799         osd.30            up  1.00000 1.00000
 32   hdd  1.81799         osd.32            up  1.00000 1.00000
  0   ssd  0.71169        osd.0             up  1.00000 1.00000
 26   ssd  0.71169         osd.26            up  1.00000 1.00000
-19       8.69534     host c04-h21-6048r
 18   hdd  1.81799         osd.18            up  1.00000 1.00000
 23   hdd  1.81799         osd.23            up  1.00000 1.00000
 36   hdd  1.81799         osd.36            up  1.00000 1.00000
 40   hdd  1.81799         osd.40            up  1.00000 1.00000
  3   ssd  0.71169        osd.3             up  1.00000 1.00000
 33   ssd  0.71169         osd.33            up  1.00000 1.00000
-16       8.69534     host c04-h25-6048r
 16   hdd  1.81799         osd.16            up  1.00000 1.00000
 22   hdd  1.81799         osd.22            up  1.00000 1.00000
 37   hdd  1.81799         osd.37            up  1.00000 1.00000
 41   hdd  1.81799         osd.41            up  1.00000 1.00000
  1   ssd  0.71169        osd.1             up  1.00000 1.00000
 28   ssd  0.71169         osd.28            up  1.00000 1.00000
 -5       8.69534     host c04-h29-6048r
  9   hdd  1.81799         osd.9             up  1.00000 1.00000
 12   hdd  1.81799         osd.12            up  1.00000 1.00000
 19   hdd  1.81799         osd.19            up  1.00000 1.00000
 24   hdd  1.81799         osd.24            up  1.00000 1.00000
  2   ssd  0.71169        osd.2             up  1.00000 1.00000
 14   ssd  0.71169         osd.14            up  1.00000 1.00000
```

Object Storage Gateway 用に、2 つの NVMe デバイスおよび LVM を最適に使用するように Ceph が設定されました。