Red Hat Ceph Storage 7

インストールガイド

Red Hat Enterprise Linux への Red Hat Ceph Storage のインストール

Last Updated: 2024-03-28

Red Hat Ceph Storage 7 インストールガイド

Red Hat Enterprise Linux への Red Hat Ceph Storage のインストール

法律上の通知

概要

本書では、AMD64 および Intel 64 のアーキテクチャーで実行している Red Hat Enterprise Linux に Red Hat Ceph Storage をインストールする方法を説明します。 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 RED HAT CEPH STORAGE

Red Hat Ceph Storage は、スケーラブルでオープンなソフトウェア定義のストレージプラットフォームであり、エンタープライズ向けバージョンの Ceph ストレージシステムと Ceph 管理プラットフォーム、デプロイメントユーティリティー、およびサポートサービスを組み合わせたものです。

Red Hat Ceph Storage は、クラウドインフラストラクチャーおよび Web スケールオブジェクトストレージ用に設計されています。Red Hat Ceph Storage クラスターは、以下のタイプのノードで設定されます。

Ceph Monitor

各 Ceph Monitor ノードは **ceph-mon** デーモンを実行し、ストレージクラスターマップのマスターコピーを維持します。ストレージクラスターマップには、ストレージクラスタートポロジーが含まれます。Ceph ストレージクラスターに接続するクライアントは、Ceph Monitor からストレージクラスターマッピングの現在のコピーを取得します。これにより、クライアントはストレージクラスターからデータの読み取りおよび書き込みが可能になります。

重要

ストレージクラスターは、1 つの Ceph Monitor でのみ実行できます。ただし、実稼働環境のストレージクラスターで高可用性を確保するために、RedHat は少なくとも 3 つの Ceph Monitor ノードを使用したデプロイメントのみをサポートします。Red Hat は、750 Ceph OSD を超えるストレージクラスター用に合計 5 つの Ceph Monitor をデプロイすることを推奨します。

Ceph Manager

Ceph Manager デーモン **ceph-mgr** は、Ceph Monitor ノードで実行されている Ceph Monitor デーモンと共存し、追加のサービスを提供します。Ceph Manager は、Ceph Manager モジュールを使用してその他の監視システムおよび管理システム用のインターフェイスを提供します。Ceph Manager デーモンの実行は、通常のストレージクラスター操作の要件です。

Ceph OSD

各 Ceph Object Storage Device(OSD) ノードは **ceph-osd** デーモンを実行し、ノードに接続されている論理ディスクと対話します。ストレージクラスターは、データをこれらの Ceph OSD ノードに保存します。

Ceph は、OSD ノードを非常に少ない状態で実行できます (デフォルトは 3 つですが、実稼働ストレージクラスターは適度な規模から初めてより良いパフォーマンスが向上します)。たとえば、ストレージクラスター内の 50 の Ceph OSD など。理想的には、Ceph Storage クラスターには複数の OSD ノードがあり、CRUSH マップを適宜設定して障害のドメインを分離できることが望ましいと言えます。

Ceph MDS

各 Ceph Metadata Server (MDS) ノードは **ceph-mds** デーモンを実行し、Ceph File System (CephFS) に保管されたファイルに関するメタデータを管理します。Ceph MDS デーモンは、共有ストレージクラスターへのアクセスも調整します。

Ceph Object Gateway

Ceph Object Gateway ノードは **ceph-radosgw** デーモンを実行し、**librados** 上に構築されたオブジェクトストレージインターフェイスで、アプリケーションに Ceph ストレージクラスターへの RESTful アクセスポイントを提供します。Ceph Object Gateway は以下の 2 つのインターフェイスをサポートします。

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

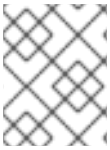
- Swift
  OpenStack Swift API の大規模なサブセットと互換性のあるインターフェイスでオブジェクトストレージ機能を提供します。

関連情報

- Ceph アーキテクチャーの詳細は、**Red Hat Ceph ストレージ管理ガイド**を参照してください。
- ハードウェアの最小推奨事項は、**Red Hat Ceph Storage ハードウェア選択ガイド**を参照してください。

第2章 RED HAT CEPH STORAGE に関する考慮事項および推奨事項

ストレージ管理者は、Red Hat Ceph Storage クラスターを実行する前に考慮すべき内容を基本的に理解しておくようにしてください。ハードウェアおよびネットワークの要件、Red Hat Ceph Storage クラスターと適切に機能するワークロードのタイプや Red Hat の推奨事項を確認してください。Red Hat Ceph Storage は、特定のビジネスニーズまたは要件に基づいて異なるワークロードに使用できます。Red Hat Ceph Storage をインストールする前に、Ceph ストレージクラスターを効率的に実行するには、ビジネス要件を達成するのに必要な計画を立てます。



注記

特定のユースケースで Red Hat Ceph Storage クラスターの使用計画にサポートが必要ですか ?サポートが必要な場合は、Red Hat 担当者にお問い合わせください。

2.1. RED HAT CEPH STORAGE の基本的な考慮事項

Red Hat Ceph Storage を使用するための最初の考慮事項は、データのストレージストラテジーの開発についてです。ストレージストラテジーとは、特定のユースケースに対応するためのデータを保管する手法を指します。OpenStack などのクラウドプラットフォームのボリュームおよびイメージを保存する必要がある場合は、ジャーナル用に Solid State Drives(SSD) を使用する高速な Serial Attached SCSI(SAS) ドライブにデータを保存することができます。一方、S3 または Swift 準拠のゲートウェイのオブジェクトデータを保存する必要がある場合は、従来の Serial Advanced Technology Attachment(SATA) ドライブなど、より経済的な方法を使用できます。Red Hat Ceph Storage は、同じストレージクラスターの両方のシナリオに対応しますが、クラウドプラットフォーム用に高速ストレージストラテジーと、オブジェクトストア用に従来のストレージを提供する手段が必要です。

Ceph のデプロイメントを正常に実行するための最も重要な手順の 1 つとして、クラスターのユースケースとワークロードに適した価格性能比のプロファイルを特定します。ユースケースに適したハードウェアを選択することが重要です。たとえば、コールドストレージアプリケーション用に IOPS が最適化されたハードウェアを選択すると、ハードウェアのコストが必要以上に増加します。また、IOPS が重視されるワークロードにおいて、より魅力的な価格帯に対して容量が最適化されたハードウェアを選択すると、パフォーマンスの低下に不満を持つユーザーが出てくる可能性が高くなります。

Red Hat Ceph Storage は、複数のストレージストラテジーをサポートできます。健全なストレージ戦略を策定するには、ユースケース、費用対効果、パフォーマンスのトレードオフ、データの耐久性などを考慮する必要があります。

ユースケース

Ceph は大容量のストレージを提供し、多くのユースケースをサポートします。

- Ceph Block Device クライアントは、クラウドプラットフォーム向けの代表的なストレージバックエンドで、ボリュームやイメージに対して制限なくストレージを提供し、コピーオンライトクローニングなど、高パフォーマンス機能を備えています。
- Ceph Object Gateway クライアントは、音声、ビットマップ、ビデオなどのオブジェクト向けの RESTful S3 準拠のオブジェクトおよび Swift 準拠のオブジェクトストレージを提供するクラウドプラットフォームの主要なストレージバックエンドです。
- 従来のファイルストレージである Ceph ファイルシステム。

コスト vs. パフォーマンス

速度、サイズ、耐久性など高いほうが優れています。ただし、優れた品質にはそれぞれコストがかかる

ので、費用対効果の面でトレードオフがあります。パフォーマンスの観点からでは、以下のユースケースを考慮してください。SSD は、比較的小規模なデータおよびジャーナリングのために非常に高速ストレージを提供できます。データベースやオブジェクトインデックスの保存には、非常に高速な SSD のプールが有効ですが、他のデータの保存にはコストがかかりすぎてしまいます。SSD ジャーナリングのある SAS ドライブは、ボリュームやイメージを安価かつ高速なパフォーマンスで提供できます。SSD ジャーナリングのない SATA ドライブは、全体的なパフォーマンスは低くなりますが、ストレージの価格を安価に抑えることができます。OSD の CRUSH 階層を作成する場合は、ユースケースと許容コスト/パフォーマンスのトレードオフを考慮する必要があります。

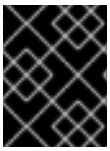
データの持続性

大規模なクラスターでは、ハードウェア障害は想定されており、例外ではありません。ただし依然として、データの損失および中断は受け入れられません。そのため、データの持続性は非常に重要になります。Ceph は、オブジェクトの複数のレプリカコピー、またはイレイジャーコーディングおよび複数のコーディングのチャンクでデータの持続性に対応します。複数のコピーまたはコーディングチャンクにより、さらに費用対効果の面でのトレードオフが分かります。コピーやコーディングのチャンクが少ない場合にはコストがかかりませんが、パフォーマンスが低下した状態で、書き込み要求に対応できなくなる可能性があります。通常、追加のコピーまたはコーディングチャンクが 2 つあるオブジェクトを使用すると、ストレージクラスターが復旧する間に、パフォーマンスが低下した状態でクラスターの書き込みを行うことができます。

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。

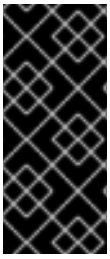
イレイジャーコーディングでは、データをデータチャンクとコーディングチャンクに分けて保存します。データチャンクが失われた場合には、イレイジャーコーディングにより、残りのデータチャンクとコーディングチャンクで失われたデータチャンクを回復できます。イレイジャーコーディングはレプリケーションに比べて大幅に経済的です。たとえば、データチャンク 8 つとコーディングチャンク 3 つのイレイジャーコーディングを使用すると、データのコピーが 3 つある状態と同じ冗長性が得られます。ただし、このようなエンコーディングスキームでは、初期のデータ保存量が約 1.5 倍になるのに対し、レプリケーションでは 3 倍になります。

CRUSH アルゴリズムは、Ceph が、ストレージクラスター内の異なる場所に追加のコピーまたはコーディングチャンクを保存して、このプロセスをサポートします。これにより、1 つのストレージデバイスまたはホストに障害が発生しても、データ損失を回避するために必要なコピーやコーディングチャンクがすべて失われないようにします。費用対効果の面でのトレードオフやデータの耐性を考慮してストレージ戦略を計画し、ストレージプールとして Ceph クライアントに提示します。



重要

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。



重要

Ceph のオブジェクトコピーやコーディングチャンクを使用すると、RAID ソリューションが古く感じられます。Ceph はすでにデータの持続性に対応しており、質の低い RAID ではパフォーマンスに悪影響があり、RAID を使用してデータを復元すると、ディープコピーや消失訂正を使用するよりもはるかにスピードが遅くなるので、RAID は使用しないでください。

関連情報

- 詳細は、**Red Hat Ceph Storage インストールガイド** の **Red Hat Ceph Storage の最小ハードウェア要件** セクションを参照してください。

2.2. RED HAT CEPH STORAGE ワークロードに関する考慮事項

Ceph Storage クラスターの主な利点の 1 つとして、パフォーマンスドメインを使用して、同じストレージクラスター内のさまざまなタイプのワークロードをサポートする機能があります。各パフォーマンスドメインには、異なるハードウェア設定を関連付けることができます。ストレージ管理者は、ストレージプールを適切なパフォーマンスドメインに配置し、特定のパフォーマンスとコストプロファイルに合わせたストレージをアプリケーションに提供できます。これらのパフォーマンスドメインに適切なサイズ設定と最適化されたサーバーを選択することは、Red Hat Ceph Storage クラスターを設計するのに不可欠な要素です。

データの読み取りおよび書き込みを行う Ceph クライアントインターフェイスに対して、Ceph Storage クラスターはクライアントがデータを格納する単純なプールとして表示されます。ただし、ストレージクラスターは、クライアントインターフェイスから完全に透過的な方法で多くの複雑な操作を実行します。Ceph クライアントおよび Ceph オブジェクトストレージデーモン (Ceph OSD または単に OSD) はいずれも、オブジェクトのストレージおよび取得にスケーラブルなハッシュ (CRUSH) アルゴリズムで制御されたレプリケーションを使用します。Ceph OSD は、ストレージクラスター内のコンテナーで実行できます。

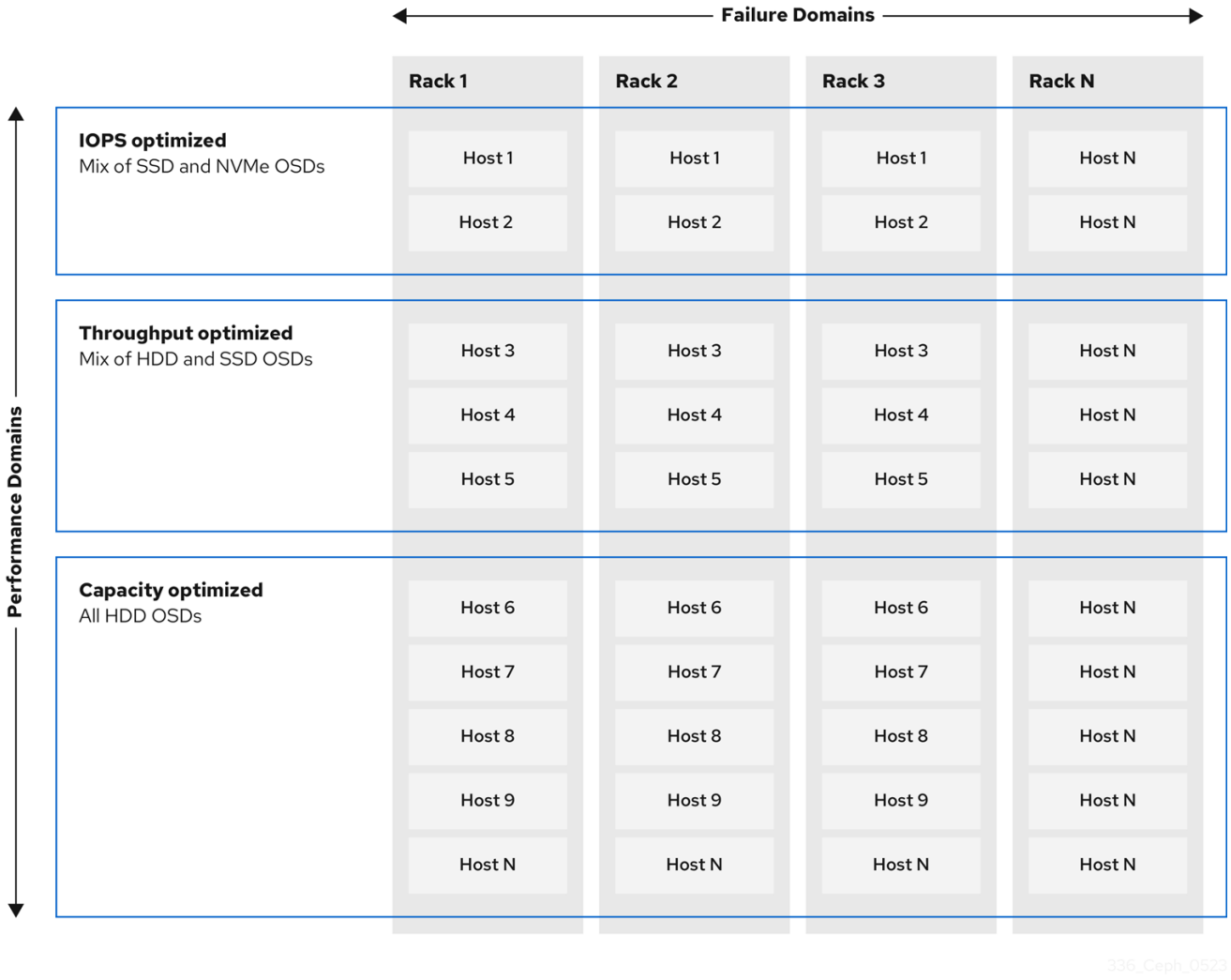
CRUSH マップはクラスターリソースのトポロジーを表し、マップは、クラスター内のクライアントホストと Ceph Monitor ホストの両方に存在します。Ceph クライアントおよび Ceph OSD はどちらも CRUSH マップと CRUSH アルゴリズムを使用します。Ceph クライアントは OSD と直接通信することで、オブジェクト検索の集中化とパフォーマンスのボトルネックとなる可能性を排除します。CRUSH マップとピアとの通信を認識することで、OSD は動的障害復旧のレプリケーション、バックフィル、およびリカバリーを処理できます。

Ceph は CRUSH マップを使用して障害ドメインを実装します。Ceph は CRUSH マップを使用してパフォーマンスドメインの実装も行います。パフォーマンスドメインは、基礎となるハードウェアのパフォーマンスプロファイルを反映させます。CRUSH マップは Ceph のデータの格納方法を記述し、これは単純な階層 (例: 非周期グラフ) およびルールセットとして実装されます。CRUSH マップは複数の階層をサポートし、ハードウェアパフォーマンスプロファイルのタイプを別のタイプから分離できます。Ceph では、デバイスの classes でパフォーマンスドメインを実装しています。

たとえば、これらのパフォーマンスドメインを同じ Red Hat Ceph Storage クラスター内に共存させることができます。

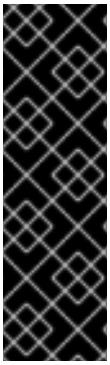
- ハードディスクドライブ (HDD) は、一般的にコストと容量を重視したワークロードに適しています。
- スループットを区別するワークロードは通常、ソリッドステートドライブ (SSD) の Ceph 書き込みジャーナルで HDD を使用します。
- MySQL や MariaDB のような IOPS を多用するワークロードでは、SSD を使用することが多いです。

図2.1 パフォーマンスおよび障害ドメイン



ワークロード

Red Hat Ceph Storage は、3 つの主要なワークロードに対して最適化されています。



重要

ストレージクラスターの価格とパフォーマンスに大きな影響を与えるので、どのハードウェアを購入するかを検討する**前**に、Red Hat Ceph Storage クラスターで実行するワークロードを慎重に検討してください。たとえば、ワークロードの容量が最適化されいるにも拘らず、スループットが最適化されたワークロードに、対象のハードウェアがより適している場合に、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、ワークロードのスループットが最適化されていて、容量が最適化されたワークロードに、対象のハードウェアが適している場合は、ストレージクラスターのパフォーマンスが低下します。

- **IOPS を最適化:** IOPS (Input, Output per Second) が最適化されたデプロイメントは、MYSQL や MariaDB インスタンスを OpenStack 上の仮想マシンとして稼働させるなど、クラウドコンピューティングの操作に適しています。IOPS が最適化された導入では、15k RPM の SAS ドライブや、頻繁な書き込み操作を処理するための個別の SSD ジャーナルなど、より高性能なストレージが必要となります。一部の IOPS のシナリオでは、すべてのフラッシュストレージを使用して IOPS と総スループットが向上します。
  IOPS が最適化されたストレージクラスターには、以下のプロパティーがあります。

- IOPS あたり最小コスト
- 1 GB あたりの最大 IOPS。
- 99 パーセンタイルのレイテンシーの一貫性。

IOPS に最適化されたストレージクラスターの用途は以下のとおりです。

- 典型的なブロックストレージ。
- ハードドライブ (HDD) の 3x レプリケーションまたはソリッドステートドライブ (SSD) の 2x レプリケーション。
- OpenStack クラウド上の MySQL

- **最適化されたスループット**: スループットが最適化されたデプロイメントは、グラフィック、音声、ビデオコンテンツなどの大量のデータを提供するのに適しています。スループットが最適化されたデプロイメントには、高帯域幅のネットワークハードウェア、コントローラー、高速シーケンシャル読み取り/書き込み機能のあるハードディスクドライブが必要です。高速なデータアクセスが必要な場合は、スループットを最適化したストレージ戦略を使用します。また、高速な書き込み性能が必要な場合は、ジャーナルに SSD (Solid State Disks) を使用すると、書き込み性能が大幅に向上します。
  スループットが最適化されたストレージクラスターには、以下のような特性があります。
  - MBps あたりの最小コスト (スループット)。
  - TB あたり最も高い MBps。
  - BTU あたりの最大 MBps
  - Watt あたりの MBps の最大数。
  - 97 パーセンタイルのレイテンシーの一貫性。

  スループットを最適化したストレージクラスターの用途は以下のとおりです。
  - ブロックまたはオブジェクトストレージ。
  - 3x レプリケーション。
  - ビデオ、音声、およびイメージのアクティブなパフォーマンスストレージ。
  - 4K 映像などのストリーミングメディア
- **最適化された容量**: 容量が最適化されたデプロイメントは、大量のデータを可能な限り安価に保存するのに適しています。容量が最適化されたデプロイメントは通常、パフォーマンスがより魅力的な価格と引き換えになります。たとえば、容量を最適化したデプロイメントでは、ジャーナリングに SSD を使用するのではなく、より低速で安価な SATA ドライブを使用し、ジャーナルを同じ場所に配置することがよくあります。
  コストと容量が最適化されたストレージクラスターには、次のような特性があります。
  - TB あたり最小コスト
  - TB あたり最小の BTU 数。
  - TB あたりに必要な最小 Watt。

  コストと容量が最適化されたストレージクラスターの用途は以下のとおりです。

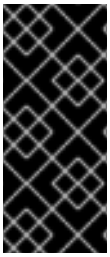
- 典型的なオブジェクトストレージ。
- 使用可能な容量を最大化するイレイジャーコーディング
- オブジェクトアーカイブ。
- ビデオ、音声、およびイメージオブジェクトのリポジトリー。

2.3. RED HAT CEPH STORAGE のネットワークに関する考察

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、ストレージクラスターが IOPS 不足になることです。また、ストレージクラスターがストレージ容量を使い果たす、はるか前に、帯域幅の制約が原因でスループットが不足することがあります。つまり、価格対性能の要求を満たすには、ネットワークのハードウェア設定が選択されたワークロードをサポートする必要があります。

ストレージ管理者は、ストレージクラスターをできるだけ早く復旧することを望みます。ストレージクラスターネットワークの帯域幅要件を慎重に検討し、ネットワークリンクのオーバーサブスクリプションに注意してください。また、クライアント間のトラフィックからクラスター内のトラフィックを分離します。また、SSD (Solid State Disk) やフラッシュ、NVMe などの高性能なストレージデバイスの使用を検討する場合には、ネットワークパフォーマンスの重要性が増していることも考慮してください。

Ceph はパブリックネットワークとストレージクラスターネットワークをサポートしています。パブリックネットワークは、クライアントのトラフィックと Ceph Monitor との通信を処理します。ストレージクラスターネットワークは、Ceph OSD のハートビート、レプリケーション、バックフィル、リカバリーのトラフィックを処理します。ストレージハードウェアには、**最低でも** 10GB/s のイーサネットリンクを 1 つ使用し、接続性とスループット向けにさらに 10GB/s イーサネットリンクを追加できます。



重要

Red Hat では、レプリケートされたプールをもとに **osd_pool_default_size** を使用してパブリックネットワークの倍数となるように、ストレージクラスターネットワークに帯域幅を割り当てることを推奨しています。また、Red Hat はパブリックネットワークとストレージクラスターネットワークを別々のネットワークカードで実行することを推奨しています。

重要

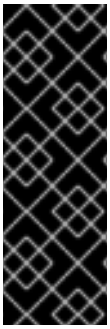
Red Hat では、実稼働環境での Red Hat Ceph Storage のデプロイメントに 10GB/s のイーサネットを使用することを推奨しています。1GB/s のイーサネットネットワークは、実稼働環境のストレージクラスターには適していません。

ドライブに障害が発生した場合に、1 Gb/秒ネットワークで 1 TB のデータを複製するには 3 時間、1 Gb/秒ネットワークで 10 TB を複製するには 30 時間かかります。10 TB を使用するのが一般的なドライブ設定です。一方、10 Gb/秒のイーサネットネットワークでは、レプリケーションの時間は 1 TB で 20 分、10 TB で 1 時間です。Ceph OSD に障害が発生した場合には、ストレージクラスターは、含まれるデータをプール内の他の Ceph OSD にレプリケートして復元することに注意してください。

ラックなどの大規模なドメインに障害が発生した場合は、ストレージクラスターが帯域幅を大幅に消費します。複数のラックで設定されるストレージクラスター (大規模なストレージ実装では一般的) を構築する際には、最適なパフォーマンスを得るために、ファットツリー設計でスイッチ間のネットワーク帯域幅をできるだけ多く利用することを検討してください。一般的な 10 Gb/s イーサネットスイッチには、48 個の 10 Gb/s ポートと 4 個の 40 Gb/s ポートがあります。スループットを最大にするには、Spine で 40 GB ポートを使用します。または、QSFP+ および SFP+ ケーブルを使用する未使用の 10

GB/s ポートを別のラックおよびスパインルーターに接続するために、さらに 40 GB/s のポートに集計することを検討します。また、LACP モード 4 でネットワークインターフェイスを結合することも検討してください。また、特にバックエンドやクラスターのネットワークでは、ジャンボフレーム、最大伝送単位 (MTU) 9000 を使用してください。

Red Hat Ceph Storage クラスター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確認します。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。フロント側のネットワークには、最低でも 10 GB/s のイーサネット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 40GB/s のイーサネットを使用することを検討してください。



重要

ネットワークの最適化には、CPU/帯域幅の比率を高めるためにジャンボフレームを使用し、非ブロックのネットワークスイッチのバックプレーンを使用することを Red Hat は推奨します。Red Hat Ceph Storage では、パブリックネットワークとクラスターネットワークの両方で、通信パスにあるすべてのネットワークデバイスに同じ MTU 値がエンドツーエンドで必要となります。Red Hat Ceph Storage クラスターを実稼働環境で使用する前に、環境内のすべてのホストとネットワーク機器で MTU 値が同じであることを確認します。

関連情報

- 詳細は、**Red Hat Ceph Storage 設定ガイド** の **プライベートネットワークの設定** セクションを参照してください。
- 詳細は、**Red Hat Ceph Storage 設定ガイド** の **パブリックネットワークの設定** セクションを参照してください。
- 詳細は、**Red Hat Ceph Storage 設定ガイド** の **クラスターへの複数のパブリックネットワークの設定** セクションを参照してください。

2.4. OSD ホストで RAID コントローラーを使用する際の考慮事項

必要に応じて、OSD ホストで RAID コントローラーを使用することを検討してください。考慮すべき事項を以下に示します。

- OSD ホストに 1 - 2 Gb のキャッシュがインストールされている RAID コントローラーがある場合は、ライトバックキャッシュを有効にすると、I/O 書き込みスループットが向上する可能性があります。ただし、キャッシュは不揮発性である必要があります。
- 最新の RAID コントローラーにはスーパーキャパシエーターがあり、電力損失イベント中に不揮発性 NAND メモリーに揮発性メモリーを流すのに十分な電力が提供されます。電源の復旧後に、特定のコントローラーとそのファームウェアがどのように動作するかを理解することが重要です。
- RAID コントローラーによっては、手動の介入が必要になります。ハードドライブは、ディスクキャッシュをデフォルトで有効または無効にすべきかどうかに関わらず、オペレーティングシステムにアドバタイズします。ただし、特定の RAID コントローラーとファームウェアは、このような情報を提供しません。ファイルシステムが破損しないように、ディスクレベルのキャッシュが無効になっていることを確認します。
- ライトバックキャッシュを有効にして、各 Ceph OSD データドライブにライトバックを設定して、単一の RAID 0 ボリュームを作成します。

- Serial Attached SCSI (SAS) または SATA 接続の Solid-state Drive (SSD) ディスクも RAID コントローラーに存在する場合は、コントローラーとファームウェアが **pass-through** モードをサポートしているかどうかを確認します。**pass-through** モードを有効にすると、キャッシュロジックが回避され、通常は高速メディアの待ち時間が大幅に低くなります。

2.5. CEPH 実行時の LINUX カーネルのチューニングに関する考察

実稼働環境用の Red Hat Ceph Storage 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。ストレージクラスター内の全ホストに調整が設定されていることを確認します。また、Red Hat サポートでケースを開き、追加でアドバイスを求めることもできます。

ファイル記述子の増加

Ceph Object Gateway は、ファイル記述子が不足すると停止することがあります。Ceph Object Gateway ホストの **/etc/security/limits.conf** ファイルを変更して、Ceph Object Gateway のファイル記述子を増やすことができます。

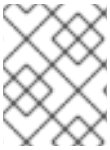
```
ceph       soft    nofile     unlimited
```

大規模ストレージクラスターの ulimit 値の調整

たとえば、1024 個以上の Ceph OSD を使用する大規模なストレージクラスターで Ceph 管理コマンドを実行する場合は、次の内容で管理コマンドを実行する各ホストに **/etc/security/limits.d/50-ceph.conf** ファイルを作成します。

```
USER_NAME       soft    nproc     unlimited
```

USER_NAME は、Ceph の管理コマンドを実行する root 以外のユーザーのアカウント名に置き換えます。



注記

Red Hat Enterprise Linux では、root ユーザーの **ulimit** 値はすでにデフォルトで **unlimited** に設定されています。

2.6. コロケーションの仕組みとその利点

コンテナー化された Ceph デーモンを同じホストの同じ場所に配置できます。Ceph のサービスの一部を共存する利点を以下に示します。

- 小規模での総所有コスト (TCO) の大幅な改善
- 最小設定の場合は、6 ホストから 3 ホストまで削減します。
- より簡単なアップグレード
- リソース分離の改善

コロケーションの仕組み
Cephadm オーケストレーターを利用すると、次のリストにある 1 つのデーモンを 1 つ以上の OSD デーモン (ceph-osd) と併置できます。

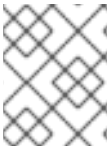
- Ceph Monitor (**ceph-mon**) および Ceph Manager (**ceph-mgr**) デーモン

- Ceph Object Gateway (**nfs-ganesha**) 用の NFS Ganesha (nfs-ganesha)
- RBD ミラー (**rbd-mirror**)
- 可観測性スタック (Grafana)

さらに、Ceph Object Gateway (**radosgw**) (RGW) および Ceph File System (**ceph-mds**) の場合は、OSD デーモンと上記のリストのデーモン (RBD ミラーを除く) のいずれかと同じ場所に配置できます。

注記

特定のノード上で 2 つの同じ種類のデーモンを共存させることはサポートされていません。



注記

ceph-mon と **ceph-mgr** は密接に連携するため、コロケーションのために、2 つの別のデーモンとしてカウントしません。

注記

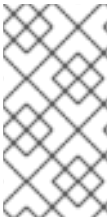
Red Hat は、Ceph Object Gateway を Ceph OSD コンテナーと併置してパフォーマンスを向上することを推奨します。

上記で共有したコロケーションルールを使用すると、これらのルールに準拠する次の最小クラスターサイズが得られます。

例 1

1. メディア: フルフラッシュシステム (SSD)
2. 使用例: ブロック (RBD) とファイル (CephFS)、またはオブジェクト (Ceph Object Gateway)
3. ノード数: 3
4. レプリケーションスキーム: 2

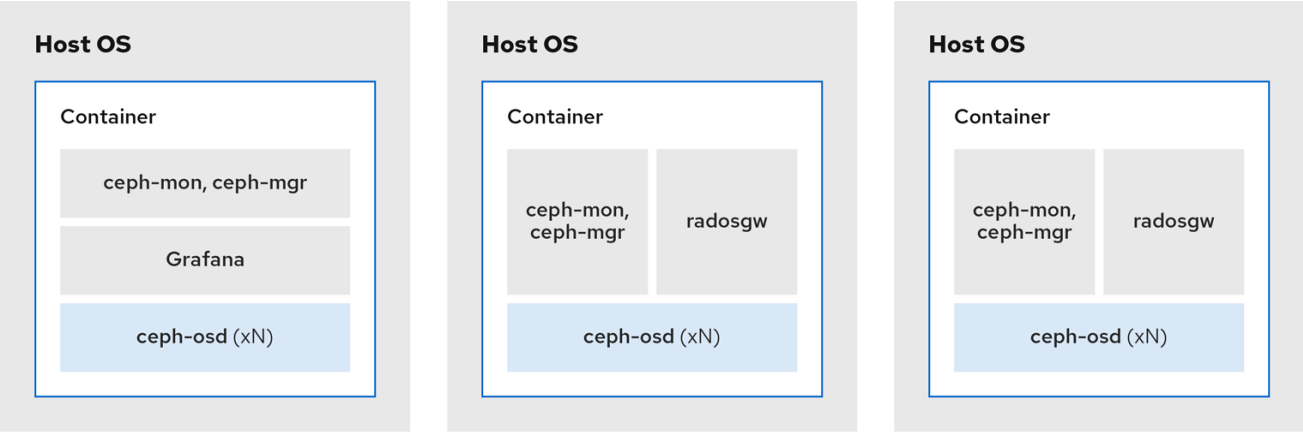
ホスト	デーモン	デーモン	デーモン
host1	OSD	Monitor/Manager	Grafana
host2	OSD	Monitor/Manager	RGW または CephFS
host3	OSD	Monitor/Manager	RGW または CephFS



注記

レプリカが 3 つあるストレージクラスターの最小サイズは 4 ノードです。同様に、レプリカが 2 つあるストレージクラスターのサイズは 3 ノードクラスターです。クラスターが長期間にわたって劣化状態にならないように、要件として、クラスター内に追加のノードを備えたレプリケーション要素の一定数のノードを用意します。

図2.2 同じ場所に配置されたデーモンの例 1



例 2

1. メディア: フルフラッシュシステム (SSD) またはスピニングデバイス (HDD)
2. 使用例: ブロック (RBD)、ファイル (CephFS)、およびオブジェクト (Ceph Object Gateway)
3. ノード数: 4
4. レプリケーションスキーム: 3

ホスト	デーモン	デーモン	デーモン
host1	OSD	Grafana	CephFS
host2	OSD	Monitor/Manager	RGW
host3	OSD	Monitor/Manager	RGW
host4	OSD	Monitor/Manager	CephFS

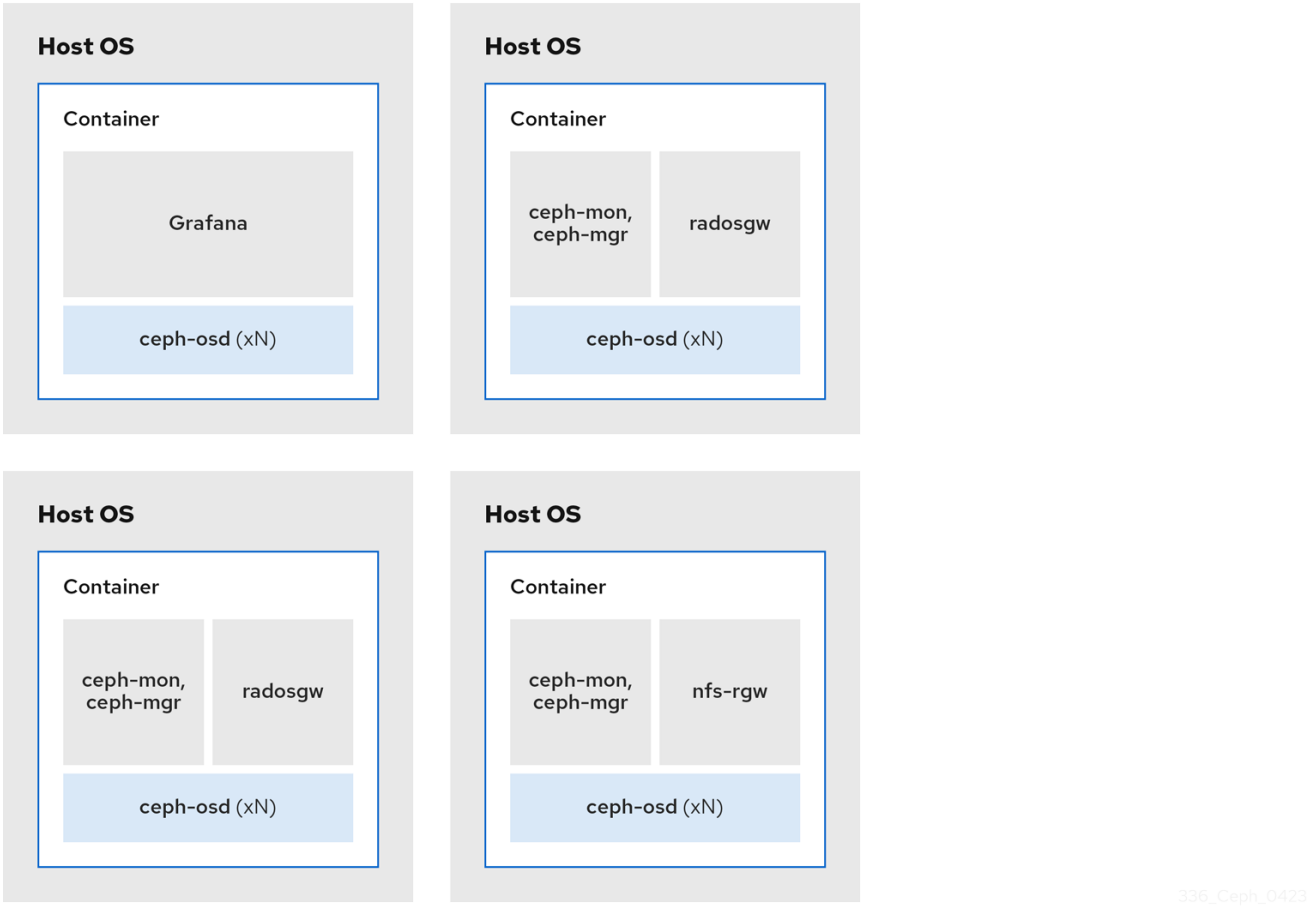
図2.3 同じ場所に配置されたデーモンの例 2

例 3

1. メディア: フルフラッシュシステム (SSD) またはスピニングデバイス (HDD)
2. 使用例: ブロック (RBD)、オブジェクト (Ceph Object Gateway)、および Ceph Object Gateway の NFS
3. ノード数: 4
4. レプリケーションスキーム: 3

ホスト	デーモン	デーモン	デーモン
host1	OSD	Grafana	
host2	OSD	Monitor/Manager	RGW
host3	OSD	Monitor/Manager	RGW
host4	OSD	Monitor/Manager	NFS (RGW)

図2.4 同じ場所に配置されたデーモンの例 3



以下の図は、同じ場所に置かれたデーモンと、同じ場所に置かれていないデーモンを使用するストレージクラスターの相違点を示しています。

図2.5 同じ場所に配置されたデーモン

図2.6 同じ場所に配置されていないデーモン

2.7. RED HAT CEPH STORAGE のオペレーティングシステム要件

Red Hat Enterprise Linux のエンタイトルメントは、Red Hat Ceph Storage のサブスクリプションに含まれます。Red Hat Ceph Storage 7 のリリースは、Red Hat Enterprise Linux 9.2 でサポートされています。

Red Hat Ceph Storage 7 はコンテナーベースのデプロイメントでのみサポートされます。

すべてのノードで同じアーキテクチャーとデプロイメントタイプを使用します。たとえば、AMD64 アーキテクチャーと Intel 64 アーキテクチャーの両方を備えたノードを混在させたり、コンテナーベースのデプロイメントを備えたノードを混在させたりしないでください。

重要

Red Hat は、異種のアーキテクチャーやデプロイメントタイプを持つクラスターをサポートしません。

SELinux

デフォルトでは、SELinux は **Enforcing** モードに設定され、**ceph-selinux** パッケージがインストールされます。SELinux の追加情報は、**データセキュリティーおよび強化ガイド** および **Red Hat Enterprise Linux 9 Using SELinux ガイド** を参照してください。

関連情報

- Red Hat Enterprise Linux

2.8. RED HAT CEPH STORAGE の最小ハードウェア要件

Red Hat Ceph Storage は、プロプライエタリーではない、商用ハードウェアでも動作します。小規模な実稼働クラスターや開発クラスターは、適度なハードウェアで性能を最適化せずに動作させることができます。

注記

ディスク領域の要件は、**/var/lib/ceph/** ディレクトリー下の Ceph デーモンのデフォルトパスに基づいています。

表2.1 コンテナー

Process	条件	最小推奨
ceph-osd-container	プロセッサー	OSD コンテナーごとに 1x AMD64 または Intel 64 CPU CORE。
	RAM	OSD コンテナーごとに少なくとも 5 GB の RAM。
	ノード数	少なくとも 3 つのノードが必要です。
	OS ディスク	ホストごとに 1x OS ディスク。
	OSD ストレージ	OSD コンテナーごとに 1x ストレージドライブ。OS ディスクと共有できません。
	block.db	任意ですが、Red Hat は、デーモンごとに SSD、NVMe または Optane パーティション、または lvm を 1 つ推奨します。サイズ設定は、オブジェクト、ファイルおよび混合ワークロード用に BlueStore に **block.data** の 4%、ブロックデバイス、Openstack cinder 、および Openstack cinder ワークロード用に BlueStore に **block.data** の 1% です。

Process	条件	最小推奨
	block.wal	任意ですが、デーモンごとに 1x SSD、NVMe または Optane パーティション、または論理ボリューム。サイズが小さい (10 GB など) を使用し、**block.db** デバイスよりも高速の場合にのみ使用します。
ネットワーク	2x 10GB イーサネット NIC	**ceph-mon-container**
プロセッサー	mon-container ごとに 1x AMD64 または Intel 64 CPU CORE	
RAM	**mon-container** あたり 3 GB	
ディスク容量	**mon-container** ごとに 10 GB (50 GB 推奨)	
監視ディスク	任意で、**Monitor rocksdb** データ用の 1x SSD ディスク	
ネットワーク	2x 1 GB のイーサネット NIC (10 GB の推奨)	
Prometheus	**/var/lib/ceph/** ディレクトリーの下に 20 GB ~ 50 GB が **/var/** ディレクトリーの下のコンテンツを保護するために別のファイルシステムとして作成されます。	**ceph-mgr-container**
プロセッサー	**mgr-container** ごとに 1x AMD64 または Intel 64 CPU CORE	
RAM	**mgr-container** あたり 3 GB	
ネットワーク	2x 1 GB のイーサネット NIC (10 GB の推奨)	**ceph-radosgw-container**

Process	条件	最小推奨
プロセッサー	radosgw-container ごとに 1x AMD64 または Intel 64 CPU CORE	
RAM	デーモンごとに 1 GB	
ディスク容量	デーモンごとに 5 GB	
ネットワーク	1x 1 GB のイーサネット NIC	**ceph-mds-container**
プロセッサー	mds-container ごとに 1x AMD64 または Intel 64 CPU CORE	
RAM	**mds-container** あたり 3 GB この数は、設定可能な MDS キャッシュサイズに大きく依存します。通常、RAM 要件は、**mds_cache_memory_limit** 設定に設定された量の 2 倍です。また、これはデーモンのためのメモリーであり、全体的なシステムメモリーではないことにも注意してください。	
ディスク容量	**mds-container** ごとに 2 GB、さらにデバッグロギングに必要な追加の領域を考慮すると、20GB から始めてみることが推奨されます。	

第3章 RED HAT CEPH STORAGE のインストール

ストレージ管理者は、**cephadm** ユーティリティーを使用して、新しい Red Hat Ceph Storage クラスターをデプロイできます。

cephadm ユーティリティーは、Ceph クラスターのライフサイクル全体を管理します。インストールおよび管理タスクは、2 種類の操作で設定されます。

- Day One 操作では、単一ノードで実行される、最小限のコンテナー化された Ceph ストレージクラスターのインストールとブートストラップが行われます。Day One には、Monitor および Manager デーモンのデプロイや Ceph OSD の追加も含まれます。
- Day Two 操作では、Ceph オーケストレーションインターフェイスである **cephadm orch** または Red Hat Ceph Storage Dashboard を使用して、他の Ceph サービスをストレージクラスターに追加することで、ストレージクラスターを拡張します。

前提条件

- アクティブなインターネット接続のある稼働中の仮想マシン (VM) またはベアメタルサーバー 1 つ以上。
- **ansible-core** が AppStream にバンドルされている Red Hat Enterprise Linux 9.2。
- 適切なエンタイトルメントを持つ有効な Red Hat サブスクリプション。
- 全ノードへの root レベルのアクセス。
- Red Hat Registry にアクセスするためのアクティブな Red Hat Network (RHN) またはサービスアカウント。
- iptables サービスの更新によってクラスターに問題が発生しないように、iptables で問題となる設定を削除します。例は、**Red Hat Ceph Storage 設定ガイド** の **デフォルトの Ceph ポート用にファイアウォールルールが設定されていることの確認** セクションを参照してください。

3.1. CEPHADM ユーティリティー

cephadm ユーティリティーは、Ceph ストレージクラスターをデプロイし、管理します。コマンドラインインターフェイス (CLI) と Red Hat Ceph Storage Dashboard Web インターフェイスの両方と密接に統合されているため、いずれの環境からでもストレージクラスターを管理できます。**cephadm** は SSH を使用してマネージャーデーモンからホストに接続し、Ceph デーモンコンテナーを追加、削除、または更新します。Ansible または Rook などの外部の設定やオーケストレーションツールに依存しません。

注記

cephadm ユーティリティーは、ホストでプリフライト Playbook を実行した後に使用できます。

cephadm ユーティリティーは、2 つの主要コンポーネントで設定されます。

- **cephadm** シェル。
- **cephadm** オーケストレーター。

cephadm シェル

cephadm シェルは、コンテナー内の **bash** シェルを起動します。このシェルを使用して、インストールやブートストラップなどの “Day One” クラスターセットアップタスクを実行したり、**ceph** コマンドを使用したりします。

cephadm シェルの起動方法の詳細は、**cephadm シェルの起動** を参照してください。

cephadm オーケストレーター

cephadm オーケストレーターを使用して、ストレージクラスターの拡張や Ceph デーモンおよびサービスのプロビジョニングなど、“Day Two” の Ceph 機能を実行します。コマンドラインインターフェイス (CLI) または Web ベースの Red Hat Ceph Storage Dashboard のいずれかを使用して、**cephadm** オーケストレーターを使用できます。オーケストラクターコマンドは **ceph orch** の形式を取ります。

cephadm スクリプトは、Ceph Manager によって使用される Ceph オーケストレーションモジュールと対話します。

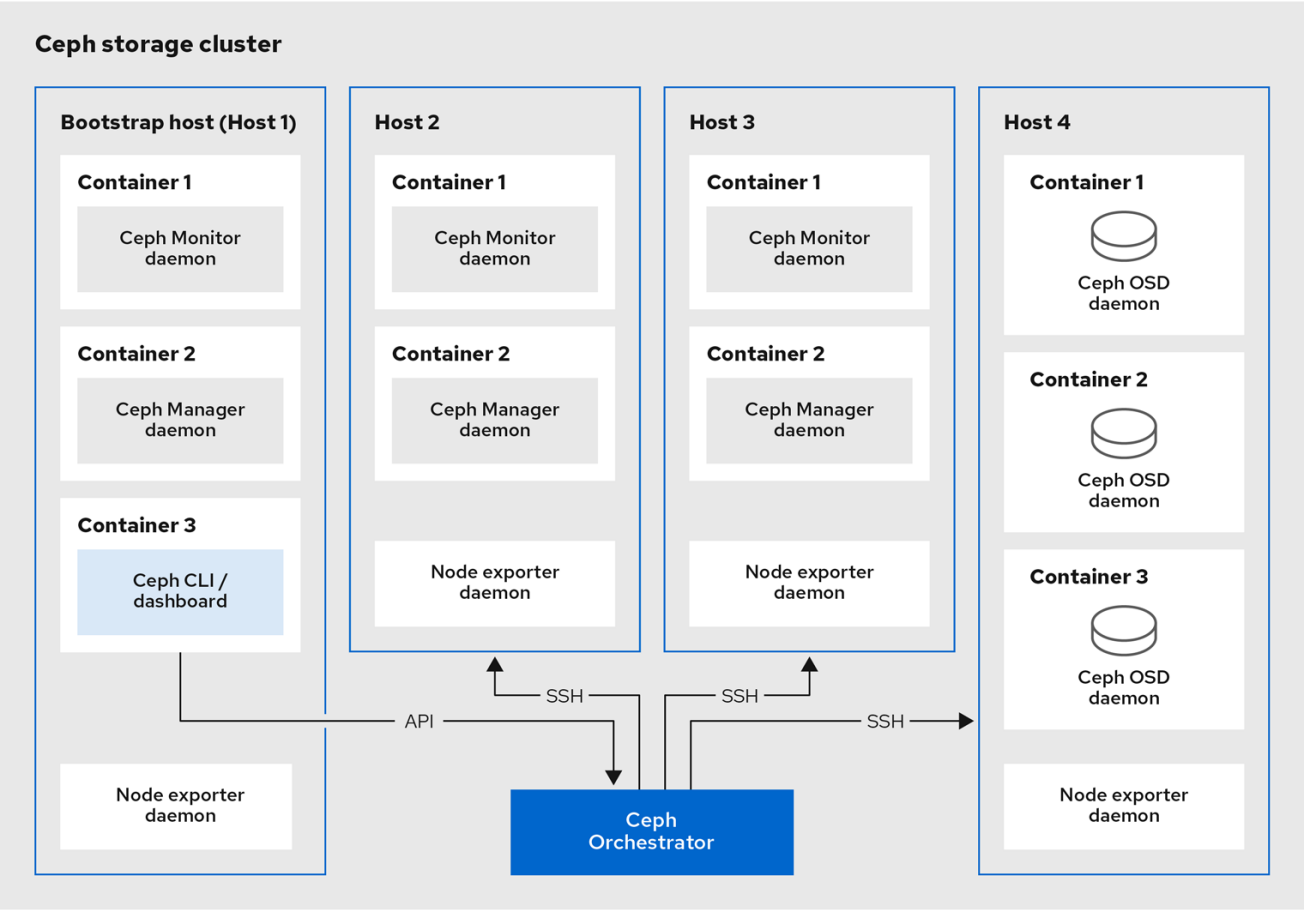
3.2. CEPHADM の仕組み

cephadm コマンドは、Red Hat Ceph Storage クラスターの完全なライフサイクルを管理します。**cephadm** コマンドは、以下の操作を行うことができます。

- 新しい Red Hat Ceph Storage クラスターをブートストラップします。
- Red Hat Ceph Storage コマンドラインインターフェイス (CLI) と連携するコンテナー化されたシェルを起動します。
- コンテナー化されたデーモンのデバッグを支援します。

cephadm コマンドは **ssh** を使用してストレージクラスターのノードと通信します。これにより、外部ツールを使用せずに Red Hat Ceph Storage コンテナーを追加、削除、または更新できます。ブートストラッププロセス時に **ssh** キーのペアを生成するか、独自の **ssh** キーを使用します。

cephadm ブートストラッププロセスでは、1 つの Ceph Monitor と 1 つの Ceph Manager で設定される、単一のノードに小規模なストレージクラスターと、必要な依存関係が作成されます。次に、オーケストレーター CLI または Red Hat Ceph Storage Dashboard を使用し、ノードが含まれるようにストレージクラスターを拡張し、すべての Red Hat Ceph Storage デーモンおよびサービスをプロビジョニングします。CLI または Red Hat Ceph Storage Dashboard の Web インターフェイスから管理機能を実行できます。



3.3. CEPHADM-ANSIBLE PLAYBOOK

cephadm-ansible パッケージは、**cephadm** で対応していないワークフローを単純化する Ansible Playbook のコレクションです。インストール後に、Playbook は **/usr/share/cephadm-ansible/** にあります。

cephadm-ansible パッケージには、以下の Playbook が含まれます。

- **cephadm-preflight.yml**
- **cephadm-clients.yml**
- **cephadm-purge-cluster.yml**

cephadm-preflight Playbook

cephadm-preflight Playbook を使用して、ストレージクラスターをブートストラップする前に、そして新規ノードまたはクライアントをストレージクラスターに追加する前にホストの初期設定を行います。この Playbook は、Ceph リポジトリーを設定し、**podman**、**lvm2**、**chronyd**、**cephadm** などの前提条件をインストールします。

cephadm-clients Playbook

cephadm-clients Playbook を使用してクライアントホストを設定します。この Playbook は、Ceph クライアントのグループへの設定およびキーリングファイルの分散を処理します。

cephadm-purge-cluster Playbook

cephadm-purge-cluster Playbook を使用して Ceph クラスターを削除します。この Playbook は、cephadm で管理される Ceph クラスターをパージします。

関連情報

- **cephadm-preflight** Playbook の詳細は、プリフライト Playbook の実行 を参照してください。
- **cephadm-clients** Playbook の詳細は、cephadm-clients Playbook の実行 を参照してください。
- **cephadm-purge-cluster** Playbook の詳細は、Ceph ストレージクラスターのパージ を参照してください。

3.4. RED HAT CEPH STORAGE ノードの CDN への登録およびサブスクリプションの割り当て

Red Hat Ceph Storage 7.0 は、Red Hat Enterprise Linux 9.2 でサポートされています。

重要

Red Hat Enterprise Linux 8.x を使用する場合、管理ノードは Red Hat Ceph Storage でサポートされている Red Hat Enterprise Linux 9.x バージョンを実行している必要があります。

完全な互換性情報は、互換性ガイド を参照してください。

前提条件

- アクティブなインターネット接続のある稼働中の仮想マシン (VM) またはベアメタルサーバー 1 つ以上。
- **ansible-core** が AppStream にバンドルされている Red Hat Enterprise Linux 9.2。
- 適切なエンタイトルメントを持つ有効な Red Hat サブスクリプション。
- 全ノードへの root レベルのアクセス。

手順

1. ノードを登録します。プロンプトが表示されたら、適切な Red Hat カスタマーポータルの認証情報を入力します。

   構文

   ```
   subscription-manager register
   ```

2. CDN から最新のサブスクリプションデータをプルします。

   構文

   ```
   subscription-manager refresh
   ```

3. Red Hat Ceph Storage で利用可能なサブスクリプションのリストを表示します。

構文

```
subscription-manager list --available --matches 'Red Hat Ceph Storage'
```

4. 適切なサブスクリプションを特定し、プール ID を取得します。

5. ソフトウェアエンタイトルメントにアクセスするには、プール ID をアタッチします。直前の手順で特定したプール ID を使用します。

構文

```
subscription-manager attach --pool=POOL_ID
```

6. デフォルトのソフトウェアリポジトリーを無効にしてから、該当するバージョンの Red Hat Enterprise Linux でサーバーおよび extras リポジトリーを有効にします。

Red Hat Enterprise Linux 9

```
subscription-manager repos --disable=*
subscription-manager repos --enable=rhel-9-for-x86_64-baseos-rpms
subscription-manager repos --enable=rhel-9-for-x86_64-appstream-rpms
```

7. システムを更新して、Red Hat Enterprise Linux の最新パッケージを受け取ります。

構文

```
# dnf update
```

8. Red Hat Ceph Storage 7 コンテンツをサブスクライブします。How to Register Ceph with Red Hat Satellite 6 の手順に従います。

9. **ceph-tools** リポジトリーを有効にします。

Red Hat Enterprise Linux 9

```
subscription-manager repos --enable=rhceph-7-tools-for-rhel-9-x86_64-rpms
```

10. クラスターに追加するすべてのノードで上記の手順を繰り返します。

11. **cephadm-ansible** をインストールします。

構文

```
dnf install cephadm-ansible
```

3.5. ANSIBLE インベントリー場所の設定

cephadm-ansible ステージングおよび実稼働環境のインベントリー場所ファイルを設定できます。Ansible インベントリーホストファイルには、ストレージクラスターの一部であるすべてのホストが含まれます。インベントリーホストファイルで個別にノードをリスト表示すること

も、**[mons]**、**[osds]**、**[rgws]** などのグループを作成して、インベントリーを明確にし、Playbook の実行時にグループまたはノードをターゲットにするための **--limit** オプションの使用を排除することもできます。

注記

クライアントをデプロイする場合は、専用の **[clients]** グループにクライアントノードを定義する必要があります。

前提条件

- Ansible 管理ノード。
- Ansible 管理ノードへの root レベルのアクセス。
- **cephadm-ansible** パッケージがノードにインストールされている。

手順

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

```
[root@admin ~]# cd /usr/share/cephadm-ansible
```

2. 必要に応じて、ステージングおよび実稼働環境用のサブディレクトリーを作成します。

```
[root@admin cephadm-ansible]# mkdir -p inventory/staging inventory/production
```

3. 必要に応じて、**ansible.cfg** ファイルを編集し、以下の行を追加してデフォルトのインベントリーの場所を割り当てます。

```
[defaults]
inventory = ./inventory/staging
```

4. 必要に応じて、各環境の インベントリー **hosts** ファイルを作成します。

```
[root@admin cephadm-ansible]# touch inventory/staging/hosts
[root@admin cephadm-ansible]# touch inventory/production/hosts
```

5. 各 **hosts** ファイルを開いて編集し、ノードおよび **[admin]** グループを追加します。

```
NODE_NAME_1
NODE_NAME_2

[admin]
ADMIN_NODE_NAME_1
```

- NODE_NAME_1 および NODE_NAME_2 を、モニター、OSD、MDS、ゲートウェイノードなどの Ceph ノードに置き換えます。
- ADMIN_NODE_NAME_1 は、管理キーリングが保存されるノードの名前に置き換えます。

例

```
host02
host03
host04

[admin]
host01
```

注記

ansible.cfg ファイルのインベントリーの場所を staging に設定する場合は、以下のようにステージング環境で Playbook を実行する必要があります。

構文

```
ansible-playbook -i inventory/staging/hosts PLAYBOOK.yml
```

実稼働環境で Playbook を実行するには、以下を実行します。

構文

```
ansible-playbook -i inventory/production/hosts PLAYBOOK.yml
```

3.6. RED HAT ENTERPRISE LINUX 9 で ROOT ユーザーとして SSH ログインを有効にする

Red Hat Enterprise Linux 9 は、**/etc/ssh/sshd_config** ファイルで **PermitRootLogin** パラメーターが **yes** に設定されている場合でも、root ユーザーとしての SSH ログインをサポートしません。次のエラーが表示されます。

例

```
[root@host01 ~]# ssh root@myhostname
root@myhostname password:
Permission denied, please try again.
```

次のいずれかの方法を実行して、root ユーザーとしてのログインを有効にすることができます。

- Red Hat Enterprise Linux 9 のインストール中に root パスワードを設定するときに、パスワードによる root SSH ログインを許可するフラグを使用します。
- Red Hat Enterprise Linux 9 のインストール後に、**PermitRootLogin** パラメーターを手動で設定します。

このセクションでは、**PermitRootLogin** パラメーターの手動設定について説明します。

前提条件

- 全ノードへの root レベルのアクセス。

手順

1. **etc/ssh/sshd_config** ファイルを開き、**PermitRootLogin** を **yes** に設定します。

例

```
[root@admin ~]# echo 'PermitRootLogin yes' >> /etc/ssh/sshd_config.d/01-
permitrootlogin.conf
```

2. **SSH** サービスを再起動します。

   例

   ```
   [root@admin ~]# systemctl restart sshd.service
   ```

3. **root** ユーザーとしてノードにログインします。

   構文

   ```
   ssh root@HOST_NAME
   ```

   HOST_NAME は、Ceph ノードのホスト名に置き換えます。

   例

   ```
   [root@admin ~]# ssh root@host01
   ```

   プロンプトに従い **root** パスワードを入力します。

関連情報

- 詳細は、Not able to login as root user via ssh in RHEL 9 server ナレッジベースソリューションを参照してください。

3.7. SUDO アクセスのある ANSIBLE ユーザーの作成

ストレージクラスター内のすべてのノードでパスワードなしの **root** アクセスで Ansible ユーザーを作成して、**cephadm-ansible** Playbook を実行することができます。Ansible ユーザーは、パスワードを要求されずにソフトウェアをインストールし、設定ファイルを作成するための **root** 権限を持つユーザーとして、すべての Red Hat Ceph Storage ノードにログインできる必要があります。

前提条件

- 全ノードへの root レベルのアクセス。
- Red Hat Enterprise Linux 9 の場合、root ユーザーとしてログインするには、Red Hat Enterprise Linux 9 で root ユーザーとして SSH ログインを有効にする を参照してください。

手順

1. **root** ユーザーとしてノードにログインします。

   構文

   ```
   ssh root@HOST_NAME
   ```

HOST_NAME は、Ceph ノードのホスト名に置き換えます。

例

```
[root@admin ~]# ssh root@host01
```

プロンプトに従い **root** パスワードを入力します。

2. 新しい Ansible ユーザーを作成します。

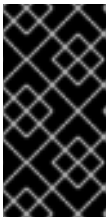
構文

```
adduser USER_NAME
```

USER_NAME は、Ansible ユーザーの新しいユーザー名に置き換えます。

例

```
[root@host01 ~]# adduser ceph-admin
```



重要

ceph をユーザー名として使用しないでください。**ceph** ユーザー名は、Ceph デーモン用に予約されます。クラスター全体で統一されたユーザー名を使用すると、使いやすさが向上しますが、侵入者は通常、そのユーザー名をブルートフォース攻撃に使用するため、明白なユーザー名の使用は避けてください。

3. このユーザーに新しいパスワードを設定します。

構文

```
passwd USER_NAME
```

USER_NAME は、Ansible ユーザーの新しいユーザー名に置き換えます。

例

```
[root@host01 ~]# passwd ceph-admin
```

プロンプトが表示されたら、新しいパスワードを 2 回入力します。

4. 新規に作成されたユーザーの **sudo** アクセスを設定します。

構文

```
cat << EOF >/etc/sudoers.d/USER_NAME
$USER_NAME ALL = (root) NOPASSWD:ALL
EOF
```

USER_NAME は、Ansible ユーザーの新しいユーザー名に置き換えます。

例

```
[root@host01 ~]# cat << EOF >/etc/sudoers.d/ceph-admin
ceph-admin ALL = (root) NOPASSWD:ALL
EOF
```

5. 正しいファイル権限を新しいファイルに割り当てます。

   構文

   ```
   chmod 0440 /etc/sudoers.d/USER_NAME
   ```

   USER_NAME は、Ansible ユーザーの新しいユーザー名に置き換えます。

   例

   ```
   [root@host01 ~]# chmod 0440 /etc/sudoers.d/ceph-admin
   ```

6. ストレージクラスター内のすべてのノードで上記の手順を繰り返します。

関連情報

- ユーザーアカウントの作成に関する詳細は、Red Hat Enterprise Linux 9 の **Configuring basic system settings** の Getting started with managing user accounts を参照してください。

3.8. SSH の設定

ストレージ管理者は、Cephadm を使用して、SSH キーを使用してリモートホストで安全に認証できます。SSH キーは、リモートホストに接続するためにモニターに保存されます。

前提条件

- Ansible 管理ノード。
- Ansible 管理ノードへの root レベルのアクセス。
- **cephadm-ansible** パッケージがノードにインストールされている。

手順

1. **cephadm-ansible** ディレクトリーに移動します。
2. 新しい SSH キーを生成します。

   例

   ```
   [ceph-admin@admin cephadm-ansible]$ ceph cephadm generate-key
   ```

3. SSH キーの公開部分を取得します。

   例

   ```
   [ceph-admin@admin cephadm-ansible]$ ceph cephadm get-pub-key
   ```

4. 現在保存されている SSH キーを削除します。

   例

   ```
   [ceph-admin@admin cephadm-ansible]$ceph cephadm clear-key
   ```

5. **mgr** デーモンを再起動して、設定を再読み込みします。

   例

   ```
   [ceph-admin@admin cephadm-ansible]$ ceph mgr fail
   ```

3.8.1. 別の SSH ユーザーの設定

ストレージ管理者は、パスワードの入力を求められずにコンテナーイメージのダウンロード、コンテナーの起動、コマンドの実行を行うための十分な権限を持つすべての Ceph クラスターノードにログインできる非 root SSH ユーザーを設定できます。

重要

非 root SSH ユーザーを設定する前に、クラスター SSH キーをユーザーの **authorized_keys** ファイルに追加する必要があり、非 root ユーザーは **パスワードなしの** sudo アクセスを持っている必要があります。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- Ansible 管理ノード。
- Ansible 管理ノードへの root レベルのアクセス。
- **cephadm-ansible** パッケージがノードにインストールされている。
- クラスター SSH キーをユーザーの **authorized_keys** に追加します。
- 非 root ユーザーに対して **passwordless** sudo アクセスを有効にします。

手順

1. **cephadm-ansible** ディレクトリーに移動します。
2. すべての Cephadm 操作を実行するユーザーの名前を Cephadm に指定します。

   構文

   ```
   [ceph-admin@admin cephadm-ansible]$ ceph cephadm set-user <user>
   ```

   例

   ```
   [ceph-admin@admin cephadm-ansible]$ ceph cephadm set-user user
   ```

3. SSH 公開キーを取得します。

構文

```
ceph cephadm get-pub-key > ~/ceph.pub
```

例

```
[ceph-admin@admin cephadm-ansible]$ ceph cephadm get-pub-key > ~/ceph.pub
```

4. SSH キーをすべてのホストにコピーします。

構文

```
ssh-copy-id -f -i ~/ceph.pub USER@HOST
```

例

```
[ceph-admin@admin cephadm-ansible]$ ssh-copy-id ceph-admin@host01
```

3.9. ANSIBLE のパスワードなし SSH の有効化

Ansible 管理ノードで SSH キーペアを生成し、ストレージクラスター内の各ノードに公開キーを配布して、Ansible がパスワードの入力を求められることなくノードにアクセスできるようにします。

前提条件

- Ansible 管理ノードへのアクセス
- ストレージクラスター内のすべてのノードへの sudo アクセスのある Ansible ユーザー
- Red Hat Enterprise Linux 9 の場合、root ユーザーとしてログインするには、Red Hat Enterprise Linux 9 で root ユーザーとして SSH ログインを有効にする を参照してください。

手順

1. SSH キーペアを生成し、デフォルトのファイル名を受け入れ、パスフレーズを空のままにします。

```
[ceph-admin@admin ~]$ ssh-keygen
```

2. 公開鍵をストレージクラスター内のすべてのノードにコピーします。

```
ssh-copy-id USER_NAME@HOST_NAME
```

USER_NAME は、Ansible ユーザーの新しいユーザー名に置き換えます。**HOST_NAME** は、Ceph ノードのホスト名に置き換えます。

例

```
[ceph-admin@admin ~]$ ssh-copy-id ceph-admin@host01
```

3. ユーザーの SSH の **config** ファイルを作成します。

-

```
[ceph-admin@admin ~]$ touch ~/.ssh/config
```

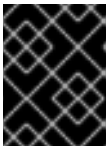
4. **config** ファイルを編集するために開きます。ストレージクラスター内の各ノードの **Hostname** および **User** オプションの値を設定します。

```
Host host01
  Hostname HOST_NAME
  User USER_NAME
Host host02
  Hostname HOST_NAME
  User USER_NAME
...
```

HOST_NAME は、Ceph ノードのホスト名に置き換えます。**USER_NAME** は、Ansible ユーザーの新しいユーザー名に置き換えます。

例

```
Host host01
  Hostname host01
  User ceph-admin
Host host02
  Hostname host02
  User ceph-admin
Host host03
  Hostname host03
  User ceph-admin
```



重要

~/.ssh/config ファイルを設定すると、**ansible-playbook** コマンドを実行するたびに **-u USER_NAME** オプションを指定する必要がありません。

5. **~/.ssh/config** ファイルに正しいファイルパーミッションを設定します。

```
[ceph-admin@admin ~]$ chmod 600 ~/.ssh/config
```

関連情報

- **ssh_config(5)** の man ページ。
- Using secure communications between two systems with OpenSSHを参照してください。

3.10. プリフライト PLAYBOOK の実行

この Ansible Playbook は Ceph リポジトリーを設定し、ブートストラップ用にストレージクラスターを準備します。また、**podman**、**lvm2**、**chronyd**、**cephadm** などのいくつかの前提条件もインストールします。**cephadm-ansible** および **cephadm-** preflight.yml のデフォルトの場所は **/usr/share/cephadm-ansible** です。

プリフライト Playbook は **cephadm-ansible** インベントリーファイルを使用して、ストレージクラスターのすべての管理者およびノードを識別します。

インベントリーファイルのデフォルトの場所は **/usr/share/cephadm-ansible/hosts** です。以下の例は、一般的なインベントリーファイルの構造を示しています。

例

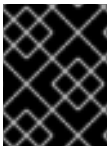
```
host02
host03
host04

[admin]
host01
```

インベントリーファイルの **[admin]** グループには、管理者キーリングが保存されるノードの名前が含まれます。新規ストレージクラスターでは、**[admin]** グループのノードがブートストラップノードになります。クラスターのブートストラップ後に管理ホストを追加する場合、詳細は **インストールガイド** の 管理ノードのセットアップ を参照してください。

注記

初期ホストをブートストラップする前に、プリフライト Playbook を実行します。



重要

非接続インストールを実行している場合は、非接続インストールのためのプリフライト Playbook の実行 を参照してください。

前提条件

- Ansible 管理ノードへの root レベルのアクセス。
- ストレージクラスター内のすべてのノードへの sudo アクセスおよびパスワードなしの **ssh** アクセスのある Ansible ユーザー。

注記

以下の例では、host01 がブートストラップノードです。

手順

1. **/usr/share/cephadm-ansible** ディレクトリーに移動します。
2. **hosts** ファイルを開いて編集し、ノードを追加します。

   例

   ```
   host02
   host03
   host04

   [admin]
   host01
   ```

3. プリフライト Playbook を実行します。

構文

```
ansible-playbook -i INVENTORY_FILE cephadm-preflight.yml --extra-vars
"ceph_origin=rhcs"
```

例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml --
extra-vars "ceph_origin=rhcs"
```

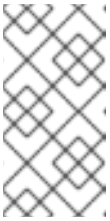
インストールが完了すると、**cephadm** は **/usr/sbin/** ディレクトリーに配置されます。

- **--limit** オプションを使用して、ストレージクラスターの選択したホストでプリフライト Playbook を実行します。

  構文

  ```
  ansible-playbook -i INVENTORY_FILE cephadm-preflight.yml --extra-vars
  "ceph_origin=rhcs" --limit GROUP_NAME|NODE_NAME
  ```

  GROUP_NAME は、インベントリーファイルからのグループ名に置き換えます。NODE_NAME は、インベントリーファイルからの特定のノード名に置き換えます。

  注記

  必要に応じて、**[mons]**、**[osds]**、**[mgrs]** などのグループ名で、インベントリーファイルのノードをグループ化できます。ただし、管理ノードを **[admin]** グループに追加し、クライアントを **[clients]** グループに追加する必要があります。

  例

  ```
  [ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml
  --extra-vars "ceph_origin=rhcs" --limit clients
  [ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml
  --extra-vars "ceph_origin=rhcs" --limit host01
  ```

- プリフライト Playbook を実行すると、**cephadm-ansible** は自動的にクライアントノードに **chronyd** および **ceph-common** をインストールします。
  プリフライト Playbook は **chronyd** をインストールしますが、単一の NTP ソース用に設定します。複数のソースを設定する場合、または非接続環境の場合は、次のドキュメントを参照してください。

  - How to configure chrony?
  - Best practices for NTP.
  - 基本的な chrony NTP のトラブルシューティング

3.11. 新しいストレージクラスターのブートストラップ

cephadm ユーティリティーは、ブートストラッププロセス中に以下のタスクを実行します。

- ローカルノード上に新しい RedHat Ceph Storage クラスターの Ceph Monitor デーモンと Ceph Manager デーモンをコンテナーとしてインストールし、開始します。
- **/etc/ceph** ディレクトリーを作成します。
- Red Hat Ceph Storage クラスターの **/etc/ceph/ceph.pub** に公開鍵のコピーを書き込み、SSH キーを root ユーザーの **/root/.ssh/authorized_keys** ファイルに追加します。
- **_admin** ラベルをブートストラップノードに適用します。
- 新しいクラスターと通信するために必要な最小限の設定ファイルを **/etc/ceph/ceph.conf** に書き込みます。
- **client.admin** 管理秘密鍵のコピーを **/etc/ceph/ceph.client.admin.keyring** に書き込みます。
- prometheus、grafana、および **node-exporter** や **alert-manager** などの他のツールを使用して、基本的なモニタリングスタックをデプロイします。

重要

非接続インストールを実行している場合は、**非接続インストールの実行** を参照してください。

注記

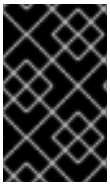
新しいストレージクラスターで実行する既存の prometheus サービスがある場合、または Rook で Ceph を実行している場合は、**cephadm bootstrap** コマンドで **--skip-monitoring-stack** オプションを使用します。このオプションは、後で手動で設定できるように、基本的なモニタリングスタックを迂回します。

重要

監視スタックをデプロイする場合は、**Red Hat Ceph Storage オペレーションガイド** の **Ceph Orchestrator を使用したモニタリングスタックのデプロイ** を参照してください。

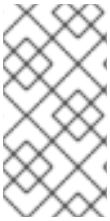
重要

ブートストラップにより、Dashboard への初期ログインのデフォルトのユーザー名およびパスワードが提供されます。ブートストラップでは、ログイン後にパスワードを変更する必要があります。



重要

ブートストラッププロセスを開始する前に、使用するコンテナーイメージに **cephadm** と同じバージョンの Red Hat Ceph Storage があることを確認します。2 つのバージョンが一致しないと **Creating initial admin user** 段階でブートストラップに失敗します。



注記

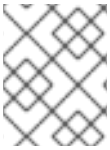
ブートストラッププロセスを開始する前に、**registry.redhat.io** コンテナーレジストリーのユーザー名とパスワードを作成する必要があります。Red Hat コンテナーレジストリー認証の詳細については、ナレッジベースの記事 **Red Hat Container Registry Authentication** を参照してください。

前提条件

- 最初の Ceph Monitor コンテナーの IP アドレス。これはストレージクラスターの最初のノードの IP アドレスでもあります。
- **registry.redhat.io** へのログインアクセス。
- 少なくとも 10 GB の空き容量がある **/var/lib/containers/**。
- 全ノードへの root レベルのアクセス。

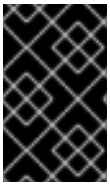
注記

ストレージクラスターに複数のネットワークおよびインターフェイスが含まれる場合、ストレージクラスターを使用するすべてのノードからアクセス可能なネットワークを選択するようにしてください。



注記

ローカルノードが完全修飾ドメイン名 (FQDN) を使用する場合は、コマンドラインで **--allow-fqdn-hostname** オプションを **cephadm bootstrap** に追加します。



重要

クラスターの最初の Monitor ノードにするノードで **cephadm bootstrap** を実行します。**IP_ADDRESS** オプションは、**cephadm bootstrap** の実行に使用するノードの IP アドレスでなければなりません。

注記

IPV6 アドレスを使用してストレージクラスターをデプロイする場合は、**--mon-ip IP_ADDRESS** オプションで IPV6 アドレス形式を使用します。例: **cephadm bootstrap --mon-ip 2620:52:0:880:225:90ff:fefc:2536 --registry-json /etc/mylogin.json**

手順

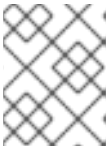
1. ストレージクラスターをブートストラップします。

   構文

   ```
   cephadm bootstrap --cluster-network NETWORK_CIDR --mon-ip IP_ADDRESS --registry-url registry.redhat.io --registry-username USER_NAME --registry-password PASSWORD --yes-i-know
   ```

   例

   ```
   [root@host01 ~]# cephadm bootstrap --cluster-network 10.10.128.0/24 --mon-ip 10.10.128.68 --registry-url registry.redhat.io --registry-username myuser1 --registry-password mypassword1 --yes-i-know
   ```



注記

内部クラストラフィックをパブリックネットワークでルーティングする必要がある場合、**--cluster-network NETWORK_CIDR** オプションを省略できます。

スクリプトが完了するまで数分かかります。スクリプトが完了すると、Red Hat Ceph Storage Dashboard URL へのクレデンシャル、Ceph コマンドラインインターフェイス (CLI) にアクセスするコマンド、およびテレメトリーを有効にする要求が提供されます。

```
Ceph Dashboard is now available at:

             URL: https://host01:8443/
            User: admin
        Password: i8nhu7zham

Enabling client.admin keyring and conf on hosts with "admin" label
You can access the Ceph CLI with:

        sudo /usr/sbin/cephadm shell --fsid 266ee7a8-2a05-11eb-b846-5254002d4916 -c
/etc/ceph/ceph.conf -k /etc/ceph/ceph.client.admin.keyring

Please consider enabling telemetry to help improve Ceph:

        ceph telemetry on

For more information see:

        https://docs.ceph.com/docs/master/mgr/telemetry/

Bootstrap complete.
```

関連情報

- 推奨される bootstrap コマンドのオプションの詳細は、**推奨される cephadm bootstrap コマンドのオプション** を参照してください。
- bootstrap コマンドで使用できるオプションの詳細は、**ブートストラップコマンドのオプション** を参照してください。
- JSON ファイルを使用してブートストラッププロセスのログインクレデンシャルが含まれるようにする方法は、**JSON ファイルを使用したログイン情報の保護** を参照してください。

3.11.1. 推奨される cephadm bootstrap コマンドのオプション

cephadm bootstrap コマンドには、ファイルの場所の指定、**ssh** の設定、パスワードの設定、およびその他の初期設定タスクの実行を可能にする複数のオプションがあります。

Red Hat は、**cephadm bootstrap** にコマンドオプションの基本セットを使用することを推奨します。初期クラスターの稼働後に追加オプションを設定できます。

以下の例は、推奨されるオプションを指定する方法を示しています。

構文

```
cephadm bootstrap --ssh-user USER_NAME --mon-ip IP_ADDRESS --allow-fqdn-hostname --
registry-json REGISTRY_JSON
```

例

```
[root@host01 ~]# cephadm bootstrap --ssh-user ceph --mon-ip 10.10.128.68 --allow-fqdn-hostname
--registry-json /etc/mylogin.json
```

関連情報

- **--registry-json** オプションの詳細は、**JSON ファイルを使用したログイン情報の保護** を参照してください。
- 利用可能なすべての **cephadm bootstrap** オプションの詳細は、**ブートストラップコマンドのオプション** を参照してください。
- root 以外のユーザーとしてストレージクラスターをブートストラップする方法は、**root 以外のユーザーでのストレージクラスターのブートストラップ** を参照してください。

3.11.2. JSON ファイルを使用したログイン情報の保護

ストレージ管理者は、ログインおよびパスワード情報を JSON ファイルに追加し、ブートストラップするために JSON ファイルを参照することがあります。これにより、ログインクレデンシャルが公開されないように保護されます。

注記

cephadm --registry-login コマンドで JSON ファイルを使用することもできます。

前提条件

- 最初の Ceph Monitor コンテナーの IP アドレス。これはストレージクラスターの最初のノードの IP アドレスでもあります。
- **registry.redhat.io** へのログインアクセス。
- 少なくとも 10 GB の空き容量がある **/var/lib/containers/**。
- 全ノードへの root レベルのアクセス。

手順

1. JSON ファイルを作成します。この例では、ファイルの名前は **mylogin.json** です。

   構文

   ```
   {
    "url":"REGISTRY_URL",
    "username":"USER_NAME",
    "password":"PASSWORD"
   }
   ```

   例

```
{
 "url":"registry.redhat.io",
 "username":"myuser1",
 "password":"mypassword1"
}
```

2. ストレージクラスターをブートストラップします。

構文

```
cephadm bootstrap --mon-ip IP_ADDRESS --registry-json /etc/mylogin.json
```

例

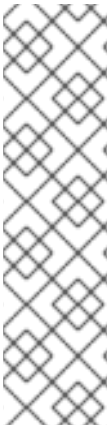
```
[root@host01 ~]# cephadm bootstrap --mon-ip 10.10.128.68 --registry-json /etc/mylogin.json
```

3.11.3. サービス設定ファイルを使用したストレージクラスターのブートストラップ

ストレージクラスターをブートストラップし、サービス設定ファイルを使用して追加のホストとデーモンを設定するには、**cephadm bootstrap** コマンドで **--apply-spec** オプションを使用します。設定ファイルは、デプロイするサービスのサービスタイプ、配置、および指定されたノードが含まれる **.yaml** ファイルです。

注記

マルチサイトなどのアプリケーションにデフォルト以外のレルムまたはゾーンを使用する場合は、それらを設定ファイルに追加して **--apply-spec** オプションを使用する代わりに、ストレージクラスターをブートストラップした後に Ceph Object Gateway デーモンを設定します。これにより、Ceph Object Gateway デーモンをデプロイする前に必要なレルムまたはゾーンを作成する機会が得られます。詳細は、**Red Hat Ceph Storage オペレーションガイド** を参照してください。



注記

NFS-Ganesha ゲートウェイまたは Metadata Server (MDS) サービスをデプロイする場合は、ストレージクラスターのブートストラップ後にそれらを設定します。

- Ceph NFS-Ganesha ゲートウェイをデプロイするには、最初に RADOS プールを作成する必要があります。
- MDS サービスをデプロイするには、最初に CephFS ボリュームを作成する必要があります。

詳細は、**Red Hat Ceph Storage オペレーションガイド** を参照してください。

前提条件

- 1 つ以上の稼働中の仮想マシン (VM) またはサーバー。
- **ansible-core** が AppStream にバンドルされている Red Hat Enterprise Linux 9.2。
- 全ノードへの root レベルのアクセス。

- **registry.redhat.io** へのログインアクセス。
- パスワードなしの **ssh** がストレージクラスター内のすべてのホストに設定されます。
- **cephadm** は、ストレージクラスターの最初の Monitor ノードにするノードにインストールされます。

手順

1. ブートストラップホストにログインします。
2. ストレージクラスターのサービス設定の **.yaml** ファイルを作成します。このサンプルファイルは、**cephadm bootstrap** に対して、最初のホストおよび 2 つの追加ホストを設定するよう指示し、利用可能なすべてのディスクに OSD を作成するように指定します。

   例

   ```
   service_type: host
   addr: host01
   hostname: host01
   ---
   service_type: host
   addr: host02
   hostname: host02
   ---
   service_type: host
   addr: host03
   hostname: host03
   ---
   service_type: host
   addr: host04
   hostname: host04
   ---
   service_type: mon
   placement:
     host_pattern: "host[0-2]"
   ---
   service_type: osd
   service_id: my_osds
   placement:
     host_pattern: "host[1-3]"
   data_devices:
     all: true
   ```

3. **--apply-spec** オプションを使用してストレージクラスターをブートストラップします。

   構文

   ```
   cephadm bootstrap --apply-spec CONFIGURATION_FILE_NAME --mon-ip
   MONITOR_IP_ADDRESS --registry-url registry.redhat.io --registry-username USER_NAME
   --registry-password PASSWORD
   ```

   例

```
[root@host01 ~]# cephadm bootstrap --apply-spec initial-config.yaml --mon-ip 10.10.128.68 --registry-url registry.redhat.io --registry-username myuser1 --registry-password mypassword1
```

スクリプトが完了するまで数分かかります。スクリプトが完了すると、Red Hat Ceph Storage Dashboard URL へのクレデンシャル、Ceph コマンドラインインターフェイス (CLI) にアクセスするコマンド、およびテレメトリーを有効にする要求が提供されます。

4. ストレージクラスターが稼働状態になったら、追加のデーモンとサービスの設定の詳細について Red Hat Ceph Storage オペレーションガイド を参照してください。

関連情報

- bootstrap コマンドで使用できるオプションの詳細は、ブートストラップコマンドのオプション を参照してください。

3.11.4. root 以外のユーザーでのストレージクラスターのブートストラップ

パスワードなしの sudo 権限がある場合は、root 以外のユーザーとしてストレージクラスターをブートストラップできます。

ブートストラップノードで root 以外のユーザーとして Red Hat Ceph Storage クラスターをブートストラップするには、**cephadm bootstrap** コマンドで **--ssh-user** オプションを使用します。**--ssh-user** は、クラスターノードへの SSH 接続のユーザーを指定します。

root 以外のユーザーには、パスワードなしの **sudo** アクセスが必要です。

前提条件

- 最初の Ceph Monitor コンテナーの IP アドレス。これはストレージクラスターの最初の Monitor ノードの IP アドレスでもあります。
- **registry.redhat.io** へのログインアクセス。
- 少なくとも 10 GB の空き容量がある **/var/lib/containers/**。
- オプション: SSH 公開鍵と秘密鍵。
- ブートストラップノードへのパスワードなしの **sudo** アクセス。
- root 以外のユーザーは、クラスターの一部となるすべてのノードに対してパスワードなしの **sudo** アクセスがある。

手順

1. ブートストラップノードで **sudo** に変更します。

   構文

   ```
   su - SSH_USER_NAME
   ```

   例

```
[root@host01 ~]# su - ceph
Last login: Tue Sep 14 12:00:29 EST 2021 on pts/0
```

2. ブートストラップノードへの SSH 接続を確認します。

例

```
[ceph@host01 ~]$ ssh host01
Last login: Tue Sep 14 12:03:29 EST 2021 on pts/0
```

3. オプション: **cephadm bootstrap** コマンドを呼び出します。

注記

秘密キーと公開キーの使用はオプションです。SSH キーがまだ作成されていない場合は、この手順で作成できます。

--ssh-private-key オプションおよび **--ssh-public-key** オプションを追加します。

構文

```
sudo cephadm bootstrap --ssh-user USER_NAME --mon-ip IP_ADDRESS --ssh-private-key
PRIVATE_KEY --ssh-public-key PUBLIC_KEY --registry-url registry.redhat.io --registry-
username USER_NAME --registry-password PASSWORD
```

例

```
sudo cephadm bootstrap --ssh-user ceph --mon-ip 10.10.128.68 --ssh-private-key
/home/ceph/.ssh/id_rsa --ssh-public-key /home/ceph/.ssh/id_rsa.pub --registry-url
registry.redhat.io --registry-username myuser1 --registry-password mypassword1
```

関連情報

- 利用可能なすべての **cephadm bootstrap** オプションの詳細は、ブートストラップコマンドのオプション を参照してください。
- Ansible を使用してルートレスクラスターのブートストラップを自動化する方法の詳細については、ナレッジベースの記事 Red Hat Ceph Storage 6 rootless deployment Using ansible ad-hoc commands を参照してください。
- sudo 権限の詳細は、sudo アクセスの管理 を参照してください。

3.11.5. ブートストラップコマンドのオプション

cephadm bootstrap コマンドは、ローカルホストで Ceph Storage クラスターをブートストラップします。これは、MON デーモンと MGR デーモンをブートストラップノードにデプロイし、モニタリングスタックをローカルホストに自動的にデプロイし、**ceph orch host add HOSTNAME** を呼び出します。

以下の表は、**cephadm bootstrap** に利用可能なオプションをまとめています。

cephadm bootstrap オプション	説明
--config **CONFIG_FILE**, -c **CONFIG_FILE**	**CONFIG_FILE** は、Bootrap コマンドで使用する **ceph.conf** ファイルです。
--cluster-network **NETWORK_CIDR**	内部クラストートラフィック用に **NETWORK_CIDR** で定義されるサブネットを使用します。これは CIDR 表記で指定されます。例:**10.10.128.0/24**
--mon-id **MON_ID**	**MON_ID** という名前のホストでブートストラップします。デフォルト値はローカルホストです。
--mon-addrv **MON_ADDRV**	mon IP (例: [v2:localipaddr:3300,v1:localipaddr:6789])
--mon-ip **IP_ADDRESS**	**cephadm bootstrap** の実行に使用するノードの IP アドレス。
--mgr-id **MGR_ID**	MGR ノードをインストールするホスト ID。デフォルト: 無作為に生成されます。
--fsid **FSID**	クラスター FSID
--output-dir **OUTPUT_DIR**	このディレクトリーを使用して、設定、キーリング、および公開鍵ファイルを作成します。
--output-keyring **OUTPUT_KEYRING**	この場所を使用して、新規クラスターの admin キーおよび mon キーでキーリングファイルを作成します。
--output-config **OUTPUT_CONFIG**	この場所を使用して、新しいクラスターに接続するために設定ファイルを書き込みます。
--output-pub-ssh-key **OUTPUT_PUB_SSH_KEY**	この場所を使用して、クラスターの公開 SSH 鍵を書き込みます。
--skip-ssh	ローカルホストで ssh キーの設定を省略します。
--initial-dashboard-user **INITIAL_DASHBOARD_USER**	Dashboard の初期ユーザー。
--initial-dashboard-password **INITIAL_DASHBOARD_PASSWORD**	初期ダッシュボードユーザーの初期パスワード。
--ssl-dashboard-port **SSL_DASHBOARD_PORT**	SSL を使用してダッシュボードとの接続に使用されるポート番号。
--dashboard-key **DASHBOARD_KEY**	Dashboard キー。

cephadm bootstrap オプション	説明
--dashboard-crt **DASHBOARD_CRT**	Dashboard の証明書。
--ssh-config **SSH_CONFIG**	SSH 設定。
--ssh-private-key **SSH_PRIVATE_KEY**	SSH 秘密鍵。
--ssh-public-key **SSH_PUBLIC_KEY**	SSH 公開鍵。
--ssh-user **SSH_USER**	クラスターホストへの SSH 接続のユーザーを設定します。root 以外のユーザーには、パスワードを使用しない sudo が必要です。
--skip-mon-network	ブートストラップ mon ip に基づいて mon public_network を設定します。
--skip-dashboard	Ceph Dashboard を有効にしません。
--dashboard-password-noupdate	Dashboard のパスワードの強制変更を無効にします。
--no-minimize-config	設定ファイルを同化して最小化しません。
--skip-ping-check	mon IP が ping 可能であることは検証しません。
--skip-pull	ブートストラップ前に最新のイメージをプルしません。
--skip-firewalld	firewalld を設定しません。
--allow-overwrite	既存の –output-* config/keyring/ssh ファイルの上書きを許可します。
--allow-fqdn-hostname	完全修飾ホスト名を許可します。
--skip-prepare-host	ホストを準備しません。
--orphan-initial-daemons	最初の mon、mgr、および crash サービス仕様を作成しません。
--skip-monitoring-stack	モニタリングスタック (prometheus、grafana、alertmanager、node-exporter) を自動的にプロビジョニングしません。
--apply-spec **APPLY_SPEC**	ブートストラップ後にクラスター仕様ファイルを適用します (ssh キーをコピーし、ホストを追加し、サービスを適用します)。

cephadm bootstrap オプション	説明
--registry-url **REGISTRY_URL**	ログインするカスタムレジストリーの URL を指定します。例: **registry.redhat.io**
--registry-username **REGISTRY_USERNAME**	カスタムレジストリーへのログインアカウントのユーザー名。
--registry-password **REGISTRY_PASSWORD**	カスタムレジストリーへのログインアカウントのパスワード
--registry-json **REGISTRY_JSON**	レジストリーのログイン情報が含まれる JSON ファイル。

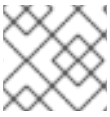
関連情報

- **--skip-monitoring-stack** オプションの詳細は、**ホストの追加** を参照してください。
- **registry-json** オプションを使用してレジストリーにログインする方法については、**registry-login** コマンドのヘルプを参照してください。
- **cephadm** オプションの詳細は、**cephadm** のヘルプを参照してください。

3.11.6. 非接続インストールのプライベートレジストリーの設定

非接続インストール手順を使用して **cephadm** をインストールし、ストレージクラスターをプライベートネットワークでブートストラップできます。非接続インストールでは、インストールにプライベートレジストリーが使用されます。デプロイメント時に Red Hat Ceph Storage ノードがインターネットにアクセスできない場合は、この手順を使用します。

認証および自己署名証明書を使用してセキュアなプライベートレジストリーを設定するには、以下の手順に従います。インターネットとローカルクラスターの両方にアクセスできるノードで、以下の手順を実行します。



注記

実稼働環境に非セキュアなレジストリーを使用することは推奨されていません。

前提条件

- アクティブなインターネット接続のある稼働中の仮想マシン (VM) またはサーバー 1 つ以上。
- **ansible-core** が AppStream にバンドルされている Red Hat Enterprise Linux 9.2。
- **registry.redhat.io** へのログインアクセス。
- 全ノードへの root レベルのアクセス。

手順

1. パブリックネットワークとクラスターノードの両方にアクセスできるノードにログインします。

2. ノードを登録します。プロンプトが表示されたら、適切な Red Hat カスタマーポータルの認証情報を入力します。

   例

   ```
   [root@admin ~]# subscription-manager register
   ```

3. 最新のサブスクリプションデータをプルします。

   例

   ```
   [root@admin ~]# subscription-manager refresh
   ```

4. Red Hat Ceph Storage で利用可能なサブスクリプションのリストを表示します。

   例

   ```
   [root@admin ~]# subscription-manager list --available --all --matches="*Ceph*"
   ```

   Red Hat Ceph Storage の利用可能なサブスクリプションのリストから Pool ID をコピーします。

5. サブスクリプションを割り当て、ソフトウェアエンタイトルメントにアクセスします。

   構文

   ```
   subscription-manager attach --pool=POOL_ID
   ```

   POOL_ID を前のステップで特定したプール ID に置き換えます。

6. デフォルトのソフトウェアリポジトリーを無効にし、サーバーおよび追加のリポジトリーを有効にします。

   Red Hat Enterprise Linux 9

   ```
   [root@admin ~]# subscription-manager repos --disable=*
   [root@admin ~]# subscription-manager repos --enable=rhel-9-for-x86_64-baseos-rpms
   [root@admin ~]# subscription-manager repos --enable=rhel-9-for-x86_64-appstream-rpms
   ```

7. **podman** パッケージおよび **httpd-tools** パッケージをインストールします。

   例

   ```
   [root@admin ~]# dnf install -y podman httpd-tools
   ```

8. プライベートレジストリーのフォルダーを作成します。

   例

   ```
   [root@admin ~]# mkdir -p /opt/registry/{auth,certs,data}
   ```

レジストリーは **/opt/registry** に保存され、ディレクトリーはレジストリーを実行しているコンテナーにマウントされます。

- **auth** ディレクトリーは、レジストリーが認証に使用する **htpasswd** ファイルを保存します。
- **certs** ディレクトリーは、レジストリーが認証に使用する証明書を保存します。
- **data** ディレクトリーはレジストリーイメージを保存します。

9. プライベートレジストリーにアクセスするための認証情報を作成します。

構文

```
htpasswd -bBc /opt/registry/auth/htpasswd PRIVATE_REGISTRY_USERNAME PRIVATE_REGISTRY_PASSWORD
```

- **b** オプションは、コマンドラインからパスワードを提供します。
- **B** オプションは、**Bcrypt** 暗号化を使用してパスワードを保存します。
- **c** オプションは **htpasswd** ファイルを作成します。
- PRIVATE_REGISTRY_USERNAME を、プライベートレジストリー用に作成するユーザー名に置き換えます。
- PRIVATE_REGISTRY_PASSWORD をプライベートレジストリーのユーザー名用に作成するパスワードに置き換えます。

例

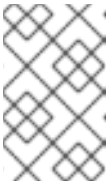
```
[root@admin ~]# htpasswd -bBc /opt/registry/auth/htpasswd myregistryusername myregistrypassword1
```

10. 自己署名証明書を作成します。

構文

```
openssl req -newkey rsa:4096 -nodes -sha256 -keyout /opt/registry/certs/domain.key -x509 -days 365 -out /opt/registry/certs/domain.crt -addext "subjectAltName = DNS:LOCAL_NODE_FQDN"
```

- LOCAL_NODE_FQDN を、プライベートレジストリーノードの完全修飾ホスト名に置き換えます。



注記

証明書のそれぞれのオプションを求めるプロンプトが出されます。**CN=** 値はノードのホスト名であり、DNS または **/etc/hosts** ファイルで解決できる必要があります。

例

```
[root@admin ~]# openssl req -newkey rsa:4096 -nodes -sha256 -keyout
/opt/registry/certs/domain.key -x509 -days 365 -out /opt/registry/certs/domain.crt -addext
"subjectAltName = DNS:admin.lab.redhat.com"
```

注記

自己署名証明書を作成する場合は、適切な Subject Alternative Name (SAN) で証明書を作成してください。適切な SAN を含まない証明書の TLS 検証を必要とする Podman コマンドは、エラー**x509: certificate relies on legacy Common Name field, use SANs or temporarily enable Common Name matching with GODEBUG=x509ignoreCN=0**を返します。

11. **domain.cert** へのシンボリックリンクを作成し、**skopeo** がファイル拡張子 **.cert** の証明書を特定できるようにします。

    例

    ```
    [root@admin ~]# ln -s /opt/registry/certs/domain.crt /opt/registry/certs/domain.cert
    ```

12. プライベートレジストリーノードの信頼済みリストに証明書を追加します。

    構文

    ```
    cp /opt/registry/certs/domain.crt /etc/pki/ca-trust/source/anchors/
    update-ca-trust
    trust list | grep -i "LOCAL_NODE_FQDN"
    ```

    LOCAL_NODE_FQDN を、プライベートレジストリーノードの FQDN に置き換えます。

    例

    ```
    [root@admin ~]# cp /opt/registry/certs/domain.crt /etc/pki/ca-trust/source/anchors/
    [root@admin ~]# update-ca-trust
    [root@admin ~]# trust list | grep -i "admin.lab.redhat.com"

        label: admin.lab.redhat.com
    ```

13. インストールのためにプライベートレジストリーにアクセスする任意のノードに証明書をコピーし、信頼されるリストを更新します。

    例

    ```
    [root@admin ~]# scp /opt/registry/certs/domain.crt root@host01:/etc/pki/ca-
    trust/source/anchors/
    [root@admin ~]# ssh root@host01
    [root@host01 ~]# update-ca-trust
    [root@host01 ~]# trust list | grep -i "admin.lab.redhat.com"

        label: admin.lab.redhat.com
    ```

14. ローカルのセキュアなプライベートレジストリーを起動します。

    構文

```
podman run --restart=always --name NAME_OF_CONTAINER \
-p 5000:5000 -v /opt/registry/data:/var/lib/registry:z \
-v /opt/registry/auth:/auth:z \
-v /opt/registry/certs:/certs:z \
-e "REGISTRY_AUTH=htpasswd" \
-e "REGISTRY_AUTH_HTPASSWD_REALM=Registry Realm" \
-e REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd \
-e "REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt" \
-e "REGISTRY_HTTP_TLS_KEY=/certs/domain.key" \
-e REGISTRY_COMPATIBILITY_SCHEMA1_ENABLED=true \
-d registry:2
```

NAME_OF_CONTAINER をコンテナーに割り当てる名前に置き換えます。

例

```
[root@admin ~]# podman run --restart=always --name myprivateregistry \
-p 5000:5000 -v /opt/registry/data:/var/lib/registry:z \
-v /opt/registry/auth:/auth:z \
-v /opt/registry/certs:/certs:z \
-e "REGISTRY_AUTH=htpasswd" \
-e "REGISTRY_AUTH_HTPASSWD_REALM=Registry Realm" \
-e REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd \
-e "REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt" \
-e "REGISTRY_HTTP_TLS_KEY=/certs/domain.key" \
-e REGISTRY_COMPATIBILITY_SCHEMA1_ENABLED=true \
-d registry:2
```

これにより、ポート 5000 でプライベートレジストリーを起動し、レジストリーを実行するコンテナーにレジストリーディレクトリーのボリュームをマウントします。

15. ローカルレジストリーノードで、**registry.redhat.io** がコンテナーレジストリーの検索パスにあることを確認します。

    a. **/etc/containers/registries.conf** ファイルを編集するために開き、**registry.redhat.io** を **unqualified-search-registries** リストに追加します (存在しない場合)。

    例

    ```
    unqualified-search-registries = ["registry.redhat.io", "registry.access.redhat.com",
    "registry.fedoraproject.org", "registry.centos.org", "docker.io"]
    ```

16. Red Hat カスタマーポータルの認証情報を使用して **registry.redhat.io** にログインします。

    構文

    ```
    podman login registry.redhat.io
    ```

17. 以下の Red Hat Ceph Storage 7 イメージ、Prometheus イメージ、および Dashboard イメージを Red Hat カスタマーポータルからプライベートレジストリーにコピーします。

    表3.1 スタックを監視するためのカスタムイメージの詳細

モニタリングスタックコンポーネント	イメージの詳細
Prometheus	registry.redhat.io/openshift4/ose-prometheus:v4.12
Grafana	registry.redhat.io/rhceph/grafana-rhel9:latest
Node-exporter	registry.redhat.io/openshift4/ose-prometheus-node-exporter:v4.12
AlertManager	registry.redhat.io/openshift4/ose-prometheus-alertmanager:v4.12
HAProxy	registry.redhat.io/rhceph/rhceph-haproxy-rhel9:latest
keepalived	registry.redhat.io/rhceph/keepalived-rhel9:latest
SNMP ゲートウェイ	registry.redhat.io/rhceph/snmp-notifier-rhel9:latest

構文

```
podman run -v /CERTIFICATE_DIRECTORY_PATH:/certs:Z -v
/CERTIFICATE_DIRECTORY_PATH/domain.cert:/certs/domain.cert:Z  --rm
registry.redhat.io/rhel9/skopeo:8.5-8 skopeo copy  --remove-signatures --src-creds
RED_HAT_CUSTOMER_PORTAL_LOGIN:RED_HAT_CUSTOMER_PORTAL_PASSWOR
D --dest-cert-dir=./certs/ --dest-creds
PRIVATE_REGISTRY_USERNAME:PRIVATE_REGISTRY_PASSWORD
docker://registry.redhat.io/SRC_IMAGE:SRC_TAG
docker://LOCAL_NODE_FQDN:5000/DST_IMAGE:DST_TAG
```

- CERTIFICATE_DIRECTORY_PATH は、自己署名証明書へのディレクトリーパスに置き換えます。
- RED_HAT_CUSTOMER_PORTAL_LOGIN および RED_HAT_CUSTOMER_PORTAL_PASSWORD は、ご自分の Red Hat カスタマーポータルの認証情報に置き換えてください。
- PRIVATE_REGISTRY_USERNAME および PRIVATE_REGISTRY_PASSWORD をプライベートレジストリーの認証情報に置き換えます。
- SRC_IMAGE および SRC_TAG を、registry.redhat.io からコピーするイメージの名前およびタグに置き換えます。
- DST_IMAGE および DST_TAG を、プライベートレジストリーにコピーするイメージの名前およびタグに置き換えます。
- LOCAL_NODE_FQDN を、プライベートレジストリーの FQDN に置き換えます。

  例

  ```
  [root@admin ~]#  podman run -v /opt/registry/certs:/certs:Z -v
  ```

```
/opt/registry/certs/domain.cert:/certs/domain.cert:Z --rm registry.redhat.io/rhel9/skopeo
skopeo copy --remove-signatures --src-creds myusername:mypassword1 --dest-cert-
dir=./certs/ --dest-creds myregistryusername:myregistrypassword1
docker://registry.redhat.io/rhceph/rhceph-7-rhel9:latest
docker://admin.lab.redhat.com:5000/rhceph/rhceph-7-rhel9:latest

[root@admin ~]# podman run -v /opt/registry/certs:/certs:Z -v
/opt/registry/certs/domain.cert:/certs/domain.cert:Z --rm registry.redhat.io/rhel9/skopeo
skopeo copy --remove-signatures --src-creds myusername:mypassword1 --dest-cert-
dir=./certs/ --dest-creds myregistryusername:myregistrypassword1
docker://registry.redhat.io/openshift4/ose-prometheus-node-exporter:v4.12
docker://admin.lab.redhat.com:5000/openshift4/ose-prometheus-node-exporter:v4.12

[root@admin ~]# podman run -v /opt/registry/certs:/certs:Z -v
/opt/registry/certs/domain.cert:/certs/domain.cert:Z --rm registry.redhat.io/rhel9/skopeo
skopeo copy --remove-signatures --src-creds myusername:mypassword1 --dest-cert-
dir=./certs/ --dest-creds myregistryusername:myregistrypassword1
docker://registry.redhat.io/rhceph/grafana-rhel9:latest
docker://admin.lab.redhat.com:5000/rhceph/grafana-rhel9:latest

[root@admin ~]# podman run -v /opt/registry/certs:/certs:Z -v
/opt/registry/certs/domain.cert:/certs/domain.cert:Z --rm registry.redhat.io/rhel9/skopeo
skopeo copy --remove-signatures --src-creds myusername:mypassword1 --dest-cert-
dir=./certs/ --dest-creds myregistryusername:myregistrypassword1
docker://registry.redhat.io/openshift4/ose-prometheus:v4.12
docker://admin.lab.redhat.com:5000/openshift4/ose-prometheus:v4.12

[root@admin ~]# podman run -v /opt/registry/certs:/certs:Z -v
/opt/registry/certs/domain.cert:/certs/domain.cert:Z --rm registry.redhat.io/rhel9/skopeo
skopeo copy --remove-signatures --src-creds myusername:mypassword1 --dest-cert-
dir=./certs/ --dest-creds myregistryusername:myregistrypassword1
docker://registry.redhat.io/openshift4/ose-prometheus-alertmanager:v4.12
docker://admin.lab.redhat.com:5000/openshift4/ose-prometheus-alertmanager:v4.12
```

18. **curl** コマンドを使用して、イメージがローカルレジストリーにあることを確認します。

構文

```
curl -u PRIVATE_REGISTRY_USERNAME:PRIVATE_REGISTRY_PASSWORD
https://LOCAL_NODE_FQDN:5000/v2/_catalog
```

例

```
[root@admin ~]# curl -u myregistryusername:myregistrypassword1
https://admin.lab.redhat.com:5000/v2/_catalog

{"repositories":["openshift4/ose-prometheus","openshift4/ose-prometheus-
alertmanager","openshift4/ose-prometheus-node-exporter","rhceph/rhceph-7-dashboard-
rhel9","rhceph/rhceph-7-rhel9"]}
```

関連情報

- さまざまなイメージの Ceph パッケージバージョンの詳細は、Red Hat Ceph Storage リリースと対応する Ceph パッケージバージョンとは何ですか？のナレッジベースソリューションを参照してください。

3.11.7. 非接続インストールのためのプリフライト Playbook の実行

cephadm-preflight.yml Ansible Playbook を使用して、Ceph リポジトリーを設定し、ブートストラップ用にストレージクラスターを準備します。また、**podman**、**lvm2**、**chronyd**、**cephadm** などのいくつかの前提条件もインストールします。

プリフライト Playbook は **cephadm-ansible** インベントリーの **hosts** ファイルを使用して、ストレージクラスターのすべてのノードを識別します。**cephadm-ansible**、**cephadm-preflight.yml**、およびインベントリー **hosts** ファイルのデフォルトの場所は **/usr/share/cephadm-ansible/** です。

以下の例は、一般的なインベントリーファイルの構造を示しています。

例

```
host02
host03
host04

[admin]
host01
```

インベントリーファイルの **[admin]** グループには、管理者キーリングが保存されるノードの名前が含まれます。

注記

初期ホストをブートストラップする前に、プリフライト Playbook を実行します。

前提条件

- **cephadm-ansible** パッケージが Ansible 管理ノードにインストールされている。
- ストレージクラスター内のすべてのノードへの root レベルのアクセス。
- パスワードなしの **ssh** がストレージクラスター内のすべてのホストに設定されます。
- 以下のリポジトリーが有効なローカルの YUM リポジトリーサーバーにアクセスするように設定されているノード。
  - rhel-9-for-x86_64-baseos-rpms
  - rhel-9-for-x86_64-appstream-rpms
  - rhceph-7-tools-for-rhel-9-x86_64-rpms

注記

ローカル YUM リポジトリーの設定の詳細は、ナレッジベースの記事 Creating a Local Repository and Sharing with Disconnected/Offline/Air-gapped Systems を参照してください。

手順

1. Ansible 管理ノードの **/usr/share/cephadm-ansible** ディレクトリーに移動します。
2. **hosts** ファイルを開いて編集し、ノードを追加します。
3. ローカルの YUM リポジトリーを使用するために **ceph_origin** パラメーターを **custom** に設定してプリフライト Playbook を実行します。

   構文

   ```
   ansible-playbook -i INVENTORY_FILE cephadm-preflight.yml --extra-vars
   "ceph_origin=custom" -e "custom_repo_url=CUSTOM_REPO_URL"
   ```

   例

   ```
   [ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml --
   extra-vars "ceph_origin=custom" -e
   "custom_repo_url=http://mycustomrepo.lab.redhat.com/x86_64/os/"
   ```

   インストールが完了すると、**cephadm** は **/usr/sbin/** ディレクトリーに配置されます。

   注記

   registries.conf ファイルの内容に Ansible Playbook を取り込みます。

   構文

   ```
   ansible-playbook -vvv -i INVENTORY_HOST_FILE_ cephadm-set-container-
   insecure-registries.yml -e insecure_registry=REGISTRY_URL
   ```

   例

   ```
   [root@admin ~]# ansible-playbook -vvv -i hosts cephadm-set-container-
   insecure-registries.yml -e insecure_registry=host01:5050
   ```

4. あるいは、**--limit** オプションを使用して、ストレージクラスターの選択したホストでプリフライト Playbook を実行することができます。

   構文

   ```
   ansible-playbook -i INVENTORY_FILE cephadm-preflight.yml --extra-vars
   "ceph_origin=custom" -e "custom_repo_url=CUSTOM_REPO_URL" --limit
   GROUP_NAME|NODE_NAME
   ```

   GROUP_NAME は、インベントリーファイルからのグループ名に置き換えます。NODE_NAME は、インベントリーファイルからの特定のノード名に置き換えます。

   例

   ```
   [ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml --
   extra-vars "ceph_origin=custom" -e
   ```

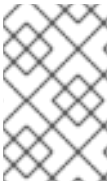
```
"custom_repo_url=http://mycustomrepo.lab.redhat.com/x86_64/os/" --limit clients
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml --
extra-vars "ceph_origin=custom" -e
"custom_repo_url=http://mycustomrepo.lab.redhat.com/x86_64/os/" --limit host02
```

注記

プリフライト Playbook を実行すると、**cephadm-ansible** は自動的にクライアントノードに **chronyd** および **ceph-common** をインストールします。

3.11.8. 非接続インストールの実行

インストールを実行する前に、Red Hat レジストリーにアクセスできるプロキシーホストから Red Hat Ceph Storage コンテナーイメージを取得するか、イメージをローカルレジストリーにコピーして Red Hat Ceph Storage コンテナーイメージを取得する必要があります。



注記

ローカルレジストリーがローカルレジストリーと共に自己署名証明書を使用する場合は、信頼されるルート証明書をブートストラップホストに追加してください。詳細は、**非接続インストールのプライベートレジストリーの設定** を参照してください。

重要

ブートストラッププロセスを開始する前に、使用するコンテナーイメージに **cephadm** と同じバージョンの Red Hat Ceph Storage があることを確認します。2 つのバージョンが一致しないと **Creating initial admin user** 段階でブートストラップに失敗します。

前提条件

- 1 つ以上の稼働中の仮想マシン (VM) またはサーバー。
- 全ノードへの root レベルのアクセス。
- パスワードなしの **ssh** がストレージクラスター内のすべてのホストに設定されます。
- ストレージクラスターのブートストラップホストでプリフライト Playbook が実行されている。詳細は、**非接続インストールのためのプリフライト Playbook の実行** を参照してください。
- プライベートレジストリーが設定され、ブートストラップノードがこれにアクセスできる。詳細は、**非接続インストールのプライベートレジストリーの設定** を参照してください。
- Red Hat Ceph Storage コンテナーイメージがカスタムレジストリーに存在する。

手順

1. ブートストラップホストにログインします。
2. ストレージクラスターをブートストラップします。

   構文

   ```
   cephadm --image
   PRIVATE_REGISTRY_NODE_FQDN:5000/CUSTOM_IMAGE_NAME:IMAGE_TAG
   ```

```
bootstrap --mon-ip IP_ADDRESS --registry-url PRIVATE_REGISTRY_NODE_FQDN:5000 --
registry-username PRIVATE_REGISTRY_USERNAME --registry-password
PRIVATE_REGISTRY_PASSWORD
```

- PRIVATE_REGISTRY_NODE_FQDN をプライベートレジストリーの完全修飾ドメイン名に置き換えます。
- CUSTOM_IMAGE_NAME および IMAGE_TAG を、プライベートレジストリーにある Red Hat Ceph Storage コンテナーイメージの名前およびタグに置き換えます。
- IP_ADDRESS は、**cephadm bootstrap** の実行に使用するノードの IP アドレスに置き換えます。
- PRIVATE_REGISTRY_USERNAME を、プライベートレジストリー用に作成するユーザー名に置き換えます。
- PRIVATE_REGISTRY_PASSWORD をプライベートレジストリーのユーザー名用に作成するパスワードに置き換えます。

例

```
[root@host01 ~]# cephadm --image admin.lab.redhat.com:5000/rhceph-7-rhel9:latest
bootstrap --mon-ip 10.10.128.68 --registry-url admin.lab.redhat.com:5000 --registry-
username myregistryusername --registry-password myregistrypassword1
```

スクリプトが完了するまで数分かかります。スクリプトが完了すると、Red Hat Ceph Storage Dashboard URL へのクレデンシャル、Ceph コマンドラインインターフェイス (CLI) にアクセスするコマンド、およびテレメトリーを有効にする要求が提供されます。

```
Ceph Dashboard is now available at:

             URL: https://host01:8443/
            User: admin
        Password: i8nhu7zham

Enabling client.admin keyring and conf on hosts with "admin" label
You can access the Ceph CLI with:

        sudo /usr/sbin/cephadm shell --fsid 266ee7a8-2a05-11eb-b846-5254002d4916 -c
/etc/ceph/ceph.conf -k /etc/ceph/ceph.client.admin.keyring

Please consider enabling telemetry to help improve Ceph:

        ceph telemetry on

For more information see:

        https://docs.ceph.com/docs/master/mgr/telemetry/

Bootstrap complete.
```

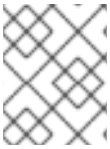
ブートストラッププロセスが完了したら、**非接続インストールのカスタムコンテナーイメージの設定変更** を参照してコンテナーイメージを設定します。

関連情報

- ストレージクラスターが稼働状態になったら、追加のデーモンとサービスの設定の詳細について Red Hat Ceph Storage オペレーションガイド を参照してください。

3.11.9. 非接続インストールのカスタムコンテナーイメージの設定変更

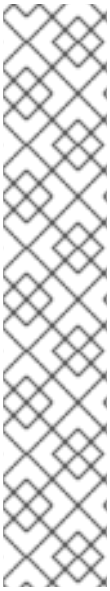
非接続ノードの最初のブートストラップを実行した後に、モニタリングスタックデーモンのカスタムコンテナーイメージを指定する必要があります。ノードはデフォルトのコンテナーレジストリーにアクセスできないため、スタックデーモンのモニタリングのためのデフォルトのコンテナーイメージを上書きできます。



注記

設定を変更する前に、初期ホストでのブートストラッププロセスが完了していることを確認してください。

デフォルトでは、モニタリングスタックコンポーネントは、プライマリー Ceph イメージに基づいてデプロイされます。ストレージクラスターの切断された環境では、最新の利用可能な監視スタックコンポーネントイメージを使用できます。



注記

カスタムレジストリーを使用する場合は、Ceph デーモンを追加する前に、新しく追加したノードのカスタムレジストリーにログインしてください。

構文

```
# ceph cephadm registry-login --registry-url CUSTOM_REGISTRY_NAME  --registry_username REGISTRY_USERNAME --registry_password REGISTRY_PASSWORD
```

例

```
# ceph cephadm registry-login --registry-url myregistry --registry_username myregistryusername --registry_password myregistrypassword1
```

前提条件

- 1 つ以上の稼働中の仮想マシン (VM) またはサーバー。
- **ansible-core** が AppStream にバンドルされている Red Hat Enterprise Linux 9.2。
- 全ノードへの root レベルのアクセス。
- パスワードなしの **ssh** がストレージクラスター内のすべてのホストに設定されます。

手順

1. **ceph config** コマンドを使用して、カスタムコンテナーイメージを設定します。

   構文

   ```
   ceph config set mgr mgr/cephadm/OPTION_NAME CUSTOM_REGISTRY_NAME/CONTAINER_NAME
   ```

OPTION_NAME には、以下のオプションを使用します。

```
container_image_prometheus
container_image_grafana
container_image_alertmanager
container_image_node_exporter
```

例

```
[root@host01 ~]# ceph config set mgr mgr/cephadm/container_image_prometheus
myregistry/mycontainer
[root@host01 ~]# ceph config set mgr mgr/cephadm/container_image_grafana
myregistry/mycontainer
[root@host01 ~]# ceph config set mgr mgr/cephadm/container_image_alertmanager
myregistry/mycontainer
[root@host01 ~]# ceph config set mgr mgr/cephadm/container_image_node_exporter
myregistry/mycontainer
```

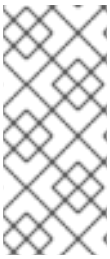
2. **node_exporter** を再デプロイします。

構文

```
ceph orch redeploy node-exporter
```

注記

いずれかのサービスがデプロイされない場合は、**ceph orch redeploy** コマンドを使用してサービスを再デプロイできます。



注記

カスタムイメージを設定することにより、設定イメージ名とタグのデフォルト値がオーバーライドされますが、上書きされません。デフォルトの値は、更新が利用可能になると変更されます。カスタムイメージを設定すると、自動更新のカスタムイメージを設定したコンポーネントを設定できなくなります。更新をインストールするには、設定イメージ名とタグを手動で更新する必要があります。

- デフォルト設定の使用に戻す場合は、カスタムコンテナーイメージをリセットできます。**ceph config rm** を使用して、設定オプションをリセットします。

構文

```
ceph config rm mgr mgr/cephadm/OPTION_NAME
```

例

```
ceph config rm mgr mgr/cephadm/container_image_prometheus
```

関連情報

- 非接続インストールの実行の詳細は、**非接続インストールの実行** を参照してください。

3.12. SSH 鍵の配布

鍵を手動で作成して配布する代わりに、**cephadm-distribute-ssh-key.yml** Playbook を使用して SSH 鍵を配布できます。Playbook は、インベントリー内のすべてのホストに SSH 公開鍵を配布します。

また、Ansible 管理ノードで SSH 鍵ペアを生成し、ストレージクラスター内の各ノードに公開鍵を配布して、Ansible がパスワードの入力を求められることなくノードにアクセスできるようにすることもできます。

前提条件

- Ansible は管理ノードにインストールされている。
- Ansible 管理ノードへのアクセス
- ストレージクラスター内のすべてのノードへの sudo アクセスのある Ansible ユーザー
- ブートストラップが完了している。**Red Hat Ceph Storage インストールガイド** の **新しいストレージクラスターのブートストラップ** セクションを参照してください。

手順

1. Ansible 管理ノードの **/usr/share/cephadm-ansible** ディレクトリーに移動します。

   例

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

2. Ansible 管理ノードから、SSH 鍵を配布します。オプションの **cephadm_pubkey_path** パラメーターは、Ansible コントローラーホスト上の SSH 公開鍵ファイルの完全パス名です。

   注記

   cephadm_pubkey_path が指定されていない場合、Playbook は **cephadm get-pub-key** コマンドから鍵を取得します。これは、少なくとも最小限のクラスターのブートストラップを設定したことを意味します。

   構文

   ```
   ansible-playbook -i INVENTORY_HOST_FILE cephadm-distribute-ssh-key.yml -e
   cephadm_ssh_user=USER_NAME -e cephadm_pubkey_path= home/cephadm/ceph.key -e
   admin_node=ADMIN_NODE_NAME_1
   ```

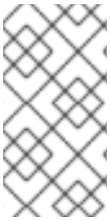
   例

   ```
   [ansible@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-distribute-ssh-
   key.yml -e cephadm_ssh_user=ceph-admin -e
   cephadm_pubkey_path=/home/cephadm/ceph.key -e admin_node=host01

   [ansible@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-distribute-ssh-
   key.yml -e cephadm_ssh_user=ceph-admin -e admin_node=host01
   ```

3.13. CEPHADM シェルの起動

cephadm shell コマンドは、すべての Ceph パッケージがインストールされたコンテナーで **bash** シェルを起動します。このシェルを使用して、インストールやブートストラップなどの “Day One” クラスターセットアップタスクを実行したり、**ceph** コマンドを実行したりします。



注記

ノードの **/etc/ceph/** に設定およびキーリングファイルが含まれる場合、コンテナー環境はこれらのファイルの値を **cephadm** シェルのデフォルトとして使用します。MON ノードで **cephadm** シェルを実行すると、**cephadm** シェルはデフォルト設定を使用するのではなく、MON コンテナーからデフォルト設定を継承します。

前提条件

- インストールされ、ブートストラップされたストレージクラスター。
- ストレージクラスター内のすべてのノードへの root レベルのアクセス。

手順

次のいずれかの方法で **cephadm** シェルを開きます。

- システムプロンプトで **cephadm shell** を入力します。この例では、シェル内から **ceph -s** コマンドを実行します。

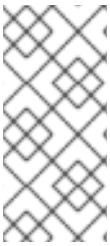
  例

  ```
  [root@host01 ~]# cephadm shell
  [ceph: root@host01 /]# ceph -s
  ```

- システムプロンプトで、**cephadm shell** と、実行するコマンドを入力します。

  例

  ```
  [root@host01 ~]# cephadm shell ceph -s
  ```



注記

cephadm シェルを終了するには、**exit** コマンドを使用します。

```
[ceph: root@host01 /]# exit
[root@host01 ~]#
```

3.14. クラスターインストールの確認

クラスターのインストールが完了したら、Red Hat Ceph Storage 7 のインストールが正常に実行されていることを確認できます。

ストレージクラスターのインストールを root ユーザーとして検証する方法は 2 つあります。

- **podman ps** コマンドを実行します。

- **cephadm shell ceph -s** を実行します。

前提条件

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

手順

- **podman ps** コマンドを実行します。

  例

  ```
  [root@host01 ~]# podman ps
  ```

  注記

  Red Hat Ceph Storage 7 では、**systemd** ユニットの形式が変更になりました。**NAMES** 列に、ユニットファイルに **FSID** が含まれるようになりました。

- **cephadm shell ceph -s** コマンドを実行します。

  例

  ```
  [root@host01 ~]# cephadm shell ceph -s

    cluster:
      id:     f64f341c-655d-11eb-8778-fa163e914bcc
      health: HEALTH_OK

    services:
      mon: 3 daemons, quorum host01,host02,host03 (age 94m)
      mgr: host01.lbnhug(active, since 59m), standbys: host02.rofgay, host03.ohipra
      mds: 1/1 daemons up, 1 standby
      osd: 18 osds: 18 up (since 10m), 18 in (since 10m)
      rgw: 4 daemons active (2 hosts, 1 zones)

    data:
      volumes: 1/1 healthy
      pools:   8 pools, 225 pgs
      objects: 230 objects, 9.9 KiB
      usage:   271 MiB used, 269 GiB / 270 GiB avail
      pgs:     225 active+clean

    io:
      client:   85 B/s rd, 0 op/s rd, 0 op/s wr
  ```

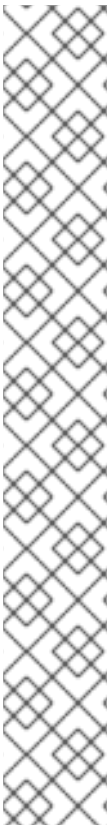
  

  注記

  ストレージクラスターのヘルスのホストとしてのステータスは、**HEALTH_WARN** で、デーモンは追加されません。

3.15. ホストの追加

Red Hat Ceph Storage インストールをブートストラップすると、同じコンテナー内の 1 つの Monitor デーモンと 1 つの Manager デーモンで設定される作業ストレージクラスターが作成されます。ストレージ管理者は、追加のホストをストレージクラスターに追加し、それらを設定することができます。



注記

- プリフライト Playbook を実行すると、Ansible インベントリーファイルに記載されているすべてのホストに **podman**、**lvm2**、**chronyd**、および **cephadm** がインストールされます。
- カスタムレジストリーを使用する場合は、Ceph デーモンを追加する前に、新しく追加したノードのカスタムレジストリーにログインしてください。

.Syntax
```
[source,subs="verbatim,quotes"]
----
# ceph cephadm registry-login --registry-url _CUSTOM_REGISTRY_NAME_
--registry_username _REGISTRY_USERNAME_ --registry_password
_REGISTRY_PASSWORD_
----
```

.Example
```
----
# ceph cephadm registry-login --registry-url myregistry --registry_username
myregistryusername --registry_password myregistrypassword1
----
```

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ルートレベル、またはストレージクラスター内のすべてのノードへの sudo アクセス権を持つユーザー。
- ノードを CDN に登録して、サブスクリプションを割り当てます。
- ストレージクラスター内のすべてのノードへの sudo アクセスおよびパスワードなしの **ssh** アクセスのある Ansible ユーザー。

手順

+

注記

次の手順では、示されているように **root**、またはユーザーがブートストラップされているユーザー名のいずれかを使用します。

1. 管理キーリングが含まれるノードから、新規ホストの root ユーザーの **authorized_keys** ファイルにストレージクラスターの公開 SSH 鍵をインストールします。

構文

```
ssh-copy-id -f -i /etc/ceph/ceph.pub user@NEWHOST
```

例

```
[root@host01 ~]# ssh-copy-id -f -i /etc/ceph/ceph.pub root@host02
[root@host01 ~]# ssh-copy-id -f -i /etc/ceph/ceph.pub root@host03
```

2. Ansible 管理ノードの **/usr/share/cephadm-ansible** ディレクトリーに移動します。

例

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

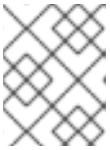
3. Ansible 管理ノードから、新しいホストを Ansible インベントリーファイルに追加します。ファイルのデフォルトの場所は **/usr/share/cephadm-ansible/hosts/** です。以下の例は、一般的なインベントリーファイルの構造を示しています。

例

```
[ceph-admin@admin ~]$ cat hosts

host02
host03
host04

[admin]
host01
```



注記

新しいホストを Ansible インベントリーファイルに追加し、ホストでプリフライト Playbook を実行している場合は、ステップ 4 に進みます。

4. **--limit** オプションを指定して、プリフライト Playbook を実行します。

構文

```
ansible-playbook -i INVENTORY_FILE cephadm-preflight.yml --extra-vars
"ceph_origin=rhcs" --limit NEWHOST
```

例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml --
extra-vars "ceph_origin=rhcs" --limit host02
```

プリフライト Playbook は、新しいホストに **podman**、**lvm2**、**chronyd**、および **cephadm** をインストールします。インストールが完了すると、**cephadm** は **/usr/sbin/** ディレクトリーに配置されます。

5. ブートストラップノードから、**cephadm** オーケストレーターを使用して、新しいホストをストレージクラスターに追加します。

構文

```
ceph orch host add NEWHOST
```

例

```
[ceph: root@host01 /]# ceph orch host add host02
Added host 'host02' with addr '10.10.128.69'
[ceph: root@host01 /]# ceph orch host add host03
Added host 'host03' with addr '10.10.128.70'
```

6. オプション: プリフライト Playbook を実行する前後に、IP アドレスでノードを追加することもできます。ストレージクラスター環境に DNS が設定されていない場合は、ホスト名とともに、IP アドレスでホストを追加できます。

構文

```
ceph orch host add HOSTNAME IP_ADDRESS
```

例

```
[ceph: root@host01 /]# ceph orch host add host02 10.10.128.69
Added host 'host02' with addr '10.10.128.69'
```

検証

- ストレージクラスターのステータスを表示し、新しいホストが追加されたことを確認します。ホストの **STATUS** は、**ceph orch host ls** コマンドの出力では空白になります。

  例

  ```
  [ceph: root@host01 /]# ceph orch host ls
  ```

関連情報

- **Red Hat Ceph Storage インストールガイド** の **Red Hat Ceph Storage ノードの CDN への登録およびサブスクリプションの割り当て** セクションを参照してください。
- **Red Hat Ceph Storage インストールガイド** の **sudo アクセスのある Ansible ユーザーの作成** セクションを参照してください。

3.15.1. addr オプションを使用したホストの特定

addr オプションは、ホストに接続するための追加の方法を提供します。ホストの IP アドレスを **addr** オプションに追加します。**ssh** がホスト名でホストに接続できない場合は、**addr** に保存されている値を使用して、IP アドレスでホストに到達します。

前提条件

- インストールされ、ブートストラップされたストレージクラスター。
- ストレージクラスター内のすべてのノードへの root レベルのアクセス。

手順

cephadm シェル内からこの手順を実行します。

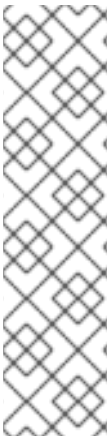
1. IP アドレスを追加します。

   構文

   ```
   ceph orch host add HOSTNAME IP_ADDR
   ```

   例

   ```
   [ceph: root@host01 /]# ceph orch host add host01 10.10.128.68
   ```



注記

ホスト名でホストを追加すると、ホストが IPv4 アドレスではなく IPv6 アドレスで追加される場合は、**ceph orch host** を使用してそのホストの IP アドレスを指定します。

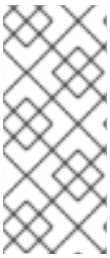
```
ceph orch host set-addr HOSTNAME IP_ADDR
```

追加したホストの IP アドレスを IPv6 形式から IPv4 形式に変換するには、次のコマンドを使用します。

```
ceph orch host set-addr HOSTNAME IPV4_ADDRESS
```

3.15.2. 複数のホストの追加

YAML ファイルを使用して、複数のホストをストレージクラスターに同時に追加します。



注記

必ずホストコンテナー内に **hosts.yaml** ファイルを作成するか、ローカルホストにファイルを作成してから、**cephadm** シェルを使用してファイルをコンテナー内にマウントします。**cephadm** シェルは、マウントしたファイルを **/mnt** に自動的に配置します。ローカルホストにファイルを直接作成し、マウントする代わりに **hosts.yaml** ファイルを適用すると、**File does not exist** というエラーが表示される可能性があります。

前提条件

- インストールされ、ブートストラップされたストレージクラスター。
- ストレージクラスター内のすべてのノードへの root レベルのアクセス。

手順

1. **ssh** 公開鍵を、追加する各ホストにコピーします。
2. テキストエディターを使用して **hosts.yaml** ファイルを作成します。
3. 以下の例のように、**host.yaml** ファイルにホストの説明を追加します。各ホストにデプロイするデーモンの配置を識別するラベルを含めます。各ホストの説明は、3 つのダッシュ (---) で区切ります。

   例

```
service_type: host
addr:
hostname: host02
labels:
- mon
- osd
- mgr
---
service_type: host
addr:
hostname: host03
labels:
- mon
- osd
- mgr
---
service_type: host
addr:
hostname: host04
labels:
- mon
- osd
```

4. ホストコンテナー内に **hosts.yaml** ファイルを作成した場合は、**ceph orch apply** コマンドを実行します。

   例

   ```
   [root@host01 ~]# ceph orch apply -i hosts.yaml
   Added host 'host02' with addr '10.10.128.69'
   Added host 'host03' with addr '10.10.128.70'
   Added host 'host04' with addr '10.10.128.71'
   ```

5. ローカルホストで直接 **hosts.yaml** ファイルを作成した場合は、**cephadm** シェルを使用してファイルをマウントします。

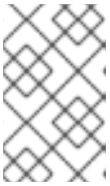
   例

   ```
   [root@host01 ~]# cephadm shell --mount hosts.yaml -- ceph orch apply -i /mnt/hosts.yaml
   ```

6. ホストおよびそれらのラベルのリストを表示します。

   例

   ```
   [root@host01 ~]# ceph orch host ls
   HOST      ADDR      LABELS          STATUS
   host02   host02    mon osd mgr
   host03   host03    mon osd mgr
   host04   host04    mon osd
   ```



注記

ホストがオンラインで正常に動作している場合、そのステータスは空白になります。オフラインホストには OFFLINE のステータスが表示され、メンテナンスモードのホストには MAINTENANCE のステータスが表示されます。

3.15.3. 非接続デプロイメントでのホストの追加

プライベートネットワークでストレージクラスターを実行し、ホストドメイン名サーバー (DNS) がプライベート IP 経由で到達できない場合は、ストレージクラスターに追加する各ホストのホスト名と IP アドレスの両方を含める必要があります。

前提条件

- 実行中のストレージクラスター。
- ストレージクラスター内のすべてのホストへの root レベルのアクセス。

手順

1. **cephadm** シェルを実行します。

   構文

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

2. ホストを追加します。

   構文

   ```
   ceph orch host add HOST_NAME HOST_ADDRESS
   ```

   例

   ```
   [ceph: root@host01 /]# ceph orch host add host03 10.10.128.70
   ```

3.15.4. ホストの削除

Ceph Orchestrator で、Ceph クラスターのホストを削除できます。すべてのデーモンは、**_no_schedule** ラベルを追加する **drain** オプションで削除され、操作が完了するまでデーモンまたはクラスターをデプロイメントできないようにします。

重要

ブートストラップホストを削除する場合は、ホストを削除する前に、必ず管理キーリングと設定ファイルをストレージクラスター内の別のホストにコピーしてください。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- すべてのノードへの root レベルのアクセス。

- ホストがストレージクラスターに追加されている。
- すべてのサービスがデプロイされている。
- Cephadm が、サービスを削除する必要があるノードにデプロイされている。

手順

1. Cephadm シェルにログインします。

   例

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

2. ホストの詳細を取得します。

   例

   ```
   [ceph: root@host01 /]# ceph orch host ls
   ```

3. ホストからすべてのデーモンをドレインします。

   構文

   ```
   ceph orch host drain HOSTNAME
   ```

   例

   ```
   [ceph: root@host01 /]# ceph orch host drain host02
   ```

   _no_schedule ラベルは、デプロイメントをブロックするホストに自動的に適用されます。

4. OSD の削除のステータスを確認します。

   例

   ```
   [ceph: root@host01 /]# ceph orch osd rm status
   ```

   OSD に配置グループ (PG) が残っていない場合、OSD は廃止され、ストレージクラスターから削除されます。

5. ストレージクラスターからすべてのデーモンが削除されているかどうかを確認します。

   構文

   ```
   ceph orch ps HOSTNAME
   ```

   例

   ```
   [ceph: root@host01 /]# ceph orch ps host02
   ```

6. ホストを削除。

構文

```
ceph orch host rm HOSTNAME
```

例

```
[ceph: root@host01 /]# ceph orch host rm host02
```

関連情報

- 詳細は **Red Hat Ceph Storage オペレーションガイド** の **Ceph Orchestrator を使用したホストの追加** セクションを参照してください。
- 詳細は **Red Hat Ceph Storage オペレーションガイド** の **Ceph Orchestrator を使用したホストの一覧表示** セクションを参照してください。

3.16. ホストのラベル付け

Ceph オーケストレーターは、ホストへのラベルの割り当てをサポートします。ラベルは自由形式であり、特別な意味はありません。つまり、**mon**、**monitor**、**mycluster_monitor**、またはその他のテキスト文字列を使用できます。各ホストに複数のラベルを指定できます。

たとえば、Ceph Monitor デーモンを配置するすべてのホストに **mon** ラベルを、Ceph Manager デーモンを配置するすべてのホストに **mgr** を、Ceph Object Gateway デーモンに **rgw** をといった具合に適用します。

ストレージクラスター内のすべてのホストにラベルを付けると、各ホストで実行されているデーモンをすばやく識別できるため、システム管理タスクが簡素化されます。さらに、Ceph オーケストレーターまたは YAML ファイルを使用して、特定のホストラベルを持つホストにデーモンをデプロイまたは削除できます。

3.16.1. ホストへのラベルの追加

Ceph オーケストレーターを使用して、ラベルをホストに追加します。ラベルを使用して、デーモンの配置を指定できます。

ラベルの例の一部は、ホストにデプロイされるサービスに基づいて、**mgr**、**mon**、および **osd** になります。各ホストに複数のラベルを指定できます。

cephadm に特別な意味を持ち、_ で始まる以下のホストラベルを追加することもできます。

- **_no_schedule**: このラベルは、**cephadm** がホスト上でデーモンをスケジュールまたはデプロイすることを阻止します。すでに Ceph デーモンが含まれている既存のホストに追加されると、これにより、**cephadm** は、自動的に削除されない OSD を除いて、それらのデーモンを別の場所に移動します。ホストに **_no_schedule** ラベルが追加されると、デーモンはそのホストにデプロイされません。ホストが削除される前にデーモンがドレインされると、そのホストに **_no_schedule** ラベルが設定されます。
- **_no_autotune_memory**: このラベルは、ホスト上のメモリーを自動調整しません。そのホスト上の 1 つ以上のデーモンに対して、**osd_memory_target_autotune** オプションまたは他の同様のオプションが有効になっている場合でも、デーモンメモリーが調整されることを阻止します。
- **_admin**: デフォルトでは、**_admin** ラベルはストレージクラスター内のブートストラップされた

ホストに適用され、**client.admin** キーは、**ceph orch client-keyring {ls|set|rm}** 関数でそのホストに配布されるように設定されます。このラベルを追加のホストに追加すると、通常、**cephadm** は設定ファイルとキーリングファイルを **/etc/ceph** ディレクトリー にデプロイします。

前提条件

- インストールされ、ブートストラップされたストレージクラスター。
- ストレージクラスター内のすべてのノードへの root レベルのアクセス。
- ホストがストレージクラスターに追加されている。

手順

1. Cephadm シェルにログインします。

   例

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

2. ホストにラベルを追加します。

   構文

   ```
   ceph orch host label add HOSTNAME LABEL
   ```

   例

   ```
   [ceph: root@host01 /]# ceph orch host label add host02 mon
   ```

検証

- ホストをリスト表示します。

  例

  ```
  [ceph: root@host01 /]# ceph orch host ls
  ```

3.16.2. ホストからのラベルの削除

Ceph オーケストレーターを使用して、ホストからラベルを削除します。

前提条件

- インストールされ、ブートストラップされたストレージクラスター。
- ストレージクラスター内のすべてのノードへの root レベルのアクセス。

手順

1. **cephadm** シェルを起動します。

```
[root@host01 ~]# cephadm shell
[ceph: root@host01 /]#
```

2. ラベルを削除します。

構文

```
ceph orch host label rm HOSTNAME LABEL
```

例

```
[ceph: root@host01 /]# ceph orch host label rm host02 mon
```

検証

- ホストをリスト表示します。

例

```
[ceph: root@host01 /]# ceph orch host ls
```

3.16.3. ホストラベルを使用した特定ホストへのデーモンのデプロイ

ホストラベルを使用して、特定のホストにデーモンをデプロイできます。ホストラベルを使用して特定のホストにデーモンをデプロイするには、次の 2 つの方法があります。

- コマンドラインから **--placement** オプションを使用する。
- YAML ファイルを使用する。

前提条件

- インストールされ、ブートストラップされたストレージクラスター。
- ストレージクラスター内のすべてのノードへの root レベルのアクセス。

手順

1. Cephadm シェルにログインします。

例

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

2. 現在のホストとラベルをリスト表示します。

例

```
[ceph: root@host01 /]# ceph orch host ls
HOST      ADDR     LABELS              STATUS
host01             _admin mon osd mgr
host02             mon osd mgr mylabel
```

- **方法 1: --placement** オプションを使用して、コマンドラインからデーモンをデプロイします。

  構文

  ```
  ceph orch apply DAEMON --placement="label:LABEL"
  ```

  例

  ```
  [ceph: root@host01 /]# ceph orch apply prometheus --placement="label:mylabel"
  ```

- **方法 2**: デーモンを YAML ファイルの特定のホストラベルに割り当てるには、YAML ファイルでサービスタイプおよびラベルを指定します。

  a. **placement.yml** ファイルを作成します。

  例

  ```
  [ceph: root@host01 /]# vi placement.yml
  ```

  b. **placement.yml** ファイルでサービスの種類とラベルを指定します。

  例

  ```
  service_type: prometheus
  placement:
    label: "mylabel"
  ```

  c. デーモン配置ファイルを適用します。

  構文

  ```
  ceph orch apply -i FILENAME
  ```

  例

  ```
  [ceph: root@host01 /]# ceph orch apply -i placement.yml
  Scheduled prometheus update…
  ```

検証

- デーモンのステータスをリスト表示します。

  構文

  ```
  ceph orch ps --daemon_type=DAEMON_NAME
  ```

  例

  ```
  [ceph: root@host01 /]# ceph orch ps --daemon_type=prometheus
  NAME              HOST   PORTS   STATUS        REFRESHED  AGE  MEM USE  MEM LIM
  VERSION  IMAGE ID      CONTAINER ID
  ```

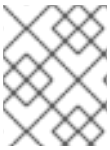
```
prometheus.host02  host02  *:9095  running (2h)     8m ago   2h    85.3M        -  2.22.2
ac25aac5d567  ad8c7593d7c0
```

3.17. MONITOR サービスの追加

一般的な Red Hat Ceph Storage ストレージクラスターには、3 つまたは 5 つのモニターデーモンが異なるホストにデプロイされます。ストレージクラスターに 5 つ以上のホストがある場合、Red Hat は 5 つの Monitor ノードをデプロイすることを推奨します。

注記

ファイアウォールの場合は、**Red Hat Ceph Storage Configuration ガイド** の **Ceph Monitor ノードのファイアウォール設定** セクションを参照してください。



注記

ブートストラップノードは、ストレージクラスターの初期モニターです。デプロイするホストのリストにブートストラップノードを含めるようにしてください。

注記

Monitor サービスを複数の特定のホストに適用する場合は、必ず同じ **ceph orch apply** コマンド内でホスト名をすべて指定してください。**ceph orch apply mon --placement host1** を指定してから、**ceph orch apply mon --placement host2** と指定すると、2 つ目のコマンドにより、host1 の Monitor サービスが削除され、Monitor サービスが host2 に適用されます。

Monitor ノードまたはクラスター全体が単一のサブネットにある場合、**cephadm** は新規ホストをクラスターに追加する際に最大 5 つの Monitor デーモンを自動的を追加します。**cephadm** は、新しいホストで Monitor デーモンを自動的に設定します。新しいホストは、ストレージクラスターの最初の (ブートストラップ) ホストと同じサブネットにあります。また、**cephadm** はストレージクラスターのサイズの変更に対応するようモニターをデプロイし、スケーリングすることもできます。

前提条件

- ストレージクラスター内のすべてのホストへの root レベルのアクセス。
- 実行中のストレージクラスター。

手順

1. 5 つの Monitor デーモンをストレージクラスター内の 5 つのランダムなホストに適用します。

```
ceph orch apply mon 5
```

2. 自動モニターのデプロイメントを無効にします。

```
ceph orch apply mon --unmanaged
```

3.17.1. 特定のホストへのモニターノードの追加

ホストラベルを使用して、Monitor ノードが含まれるホストを特定します。

前提条件

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

手順

1. **mon** ラベルをホストに割り当てます。

   構文

   ```
   ceph orch host label add HOSTNAME mon
   ```

   例

   ```
   [ceph: root@host01 /]# ceph orch host label add host01 mon
   ```

2. 現在のホストおよびラベルを表示します。

   構文

   ```
   ceph orch host ls
   ```

   例

   ```
   [ceph: root@host01 /]# ceph orch host label add host02 mon
   [ceph: root@host01 /]# ceph orch host label add host03 mon
   [ceph: root@host01 /]# ceph orch host ls
   HOST   ADDR   LABELS  STATUS
   host01        mon
   host02        mon
   host03        mon
   host04
   host05
   host06
   ```

3. ホストラベルに基づいてモニターをデプロイします。

   構文

   ```
   ceph orch apply mon label:mon
   ```

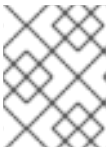
4. 特定のホストセットにモニターをデプロイします。

   構文

   ```
   ceph orch apply mon HOSTNAME1,HOSTNAME2,HOSTNAME3
   ```

   例

   ```
   [root@host01 ~]# ceph orch apply mon host01,host02,host03
   ```



注記

デプロイするホストのリストにブートストラップノードを含めるようにしてください。

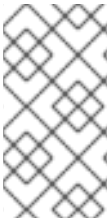
3.18. 管理ノードの設定

ストレージノードを使用してストレージクラスターを管理します。

管理ノードには、クラスター設定ファイルと管理キーリングの両方が含まれます。これらのファイルはどちらも **/etc/ceph** ディレクトリーに保存され、ストレージクラスターの名前を接頭辞として使用します。

たとえば、デフォルトの ceph クラスター名は **ceph** です。デフォルトの名前を使用するクラスターでは、管理キーリングの名前は **/etc/ceph/ceph.client.admin.keyring** になります。対応するクラスター設定ファイルの名前は **/etc/ceph/ceph.conf** です。

ストレージクラスター内の追加のホストを管理ノードとして設定するには、管理者ノードとして指定するホストに **_admin** ラベルを適用します。



注記

デフォルトでは、**_admin** ラベルをノードに適用した後に、**cephadm** は **ceph.conf** および **client.admin** キーリングファイルをそのノードにコピーします。**--skip-admin-label** オプションが **cephadm bootstrap** コマンドで指定されていない限り、**_admin** ラベルはブートストラップノードに自動的に適用されます。

前提条件

- **cephadm** がインストールされた実行中のストレージクラスター。
- ストレージクラスターが Monitor ノードおよび Manager ノードを実行している。
- クラスター内のすべてのノードへの root レベルのアクセス。

手順

1. **ceph orch host ls** を使用して、ストレージクラスター内のホストを表示します。

   例

   ```
   [root@host01 ~]# ceph orch host ls
   HOST   ADDR   LABELS  STATUS
   host01        mon,mgr,_admin
   host02        mon
   host03        mon,mgr
   host04
   host05
   host06
   ```

2. ストレージクラスターの admin ホストを指定するには、**_admin** ラベルを使用します。最良の結果を得るには、このホストで Monitor デーモンと Manager デーモンの両方が実行されている必要があります。

   構文

```
ceph orch host label add HOSTNAME _admin
```

例

```
[root@host01 ~]#  ceph orch host label add host03 _admin
```

3. admin ホストに **_admin** ラベルがあることを確認します。

例

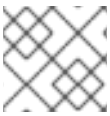
```
[root@host01 ~]#  ceph orch host ls
HOST   ADDR   LABELS  STATUS
host01        mon,mgr,_admin
host02        mon
host03        mon,mgr,_admin
host04
host05
host06
```

4. 管理ノードにログインして、ストレージクラスターを管理します。

3.18.1. ホストラベルを使用した Ceph モニターノードのデプロイメント

一般的な Red Hat Ceph Storage ストレージクラスターには、3 つまたは 5 つの Ceph Monitor デーモンが異なるホストにデプロイされます。ストレージクラスターに 5 つ以上のホストがある場合、Red Hat は 5 つの Ceph Monitor ノードをデプロイすることを推奨します。

Ceph Monitor ノードまたはクラスター全体が単一のサブネットにある場合、**cephadm** は新しいノードをクラスターに追加する際に最大 5 つの Ceph Monitor デーモンを自動的を追加します。**cephadm** は、新しいノードで Ceph Monitor デーモンを自動的に設定します。新しいノードは、ストレージクラスターの最初の (ブートストラップ) ノードと同じサブネットにあります。また、**cephadm** はストレージクラスターのサイズの変更に対応するようモニターをデプロイし、スケーリングすることもできます。



注記

ホストラベルを使用して、Ceph Monitor ノードが含まれるホストを特定します。

前提条件

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

手順

1. mon ラベルをホストに割り当てます。

構文

```
ceph orch host label add HOSTNAME mon
```

例

```
[ceph: root@host01 /]# ceph orch host label add host02 mon
[ceph: root@host01 /]# ceph orch host label add host03 mon
```

2. 現在のホストおよびラベルを表示します。

構文

```
ceph orch host ls
```

例

```
[ceph: root@host01 /]# ceph orch host ls
HOST   ADDR   LABELS  STATUS
host01        mon,mgr,_admin
host02        mon
host03        mon
host04
host05
host06
```

- ホストラベルに基づいて Ceph Monitor デーモンをデプロイします。

  構文

  ```
  ceph orch apply mon label:mon
  ```

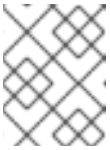
- Ceph Monitor デーモンを特定のホストセットにデプロイします。

  構文

  ```
  ceph orch apply mon HOSTNAME1,HOSTNAME2,HOSTNAME3
  ```

  例

  ```
  [ceph: root@host01 /]# ceph orch apply mon host01,host02,host03
  ```



注記

デプロイするホストのリストにブートストラップノードを含めるようにしてください。

3.18.2. IP アドレスまたはネットワーク名を使用した Ceph Monitor ノードの追加

一般的な Red Hat Ceph Storage ストレージクラスターには、3 つまたは 5 つのモニターデーモンが異なるホストにデプロイされます。ストレージクラスターに 5 つ以上のホストがある場合、Red Hat は 5 つの Monitor ノードをデプロイすることを推奨します。

Monitor ノードまたはクラスター全体が単一のサブネットにある場合、**cephadm** は新しいノードをクラスターに追加する際に最大 5 つの Monitor デーモンを自動的を追加します。Monitor デーモンを新しいノード上で設定する必要はありません。新しいノードは、ストレージクラスターの最初のノードと同

じサブネットにあります。ストレージクラスターの最初のノードはブートストラップノードです。また、**cephadm** はストレージクラスターのサイズの変更に対応するようモニターをデプロイし、スケーリングすることもできます。

前提条件

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

手順

1. 追加の Ceph Monitor ノードをそれぞれデプロイするには、以下を実行します。

   構文

   ```
   ceph orch apply mon NODE:IP_ADDRESS_OR_NETWORK_NAME
   [NODE:IP_ADDRESS_OR_NETWORK_NAME...]
   ```

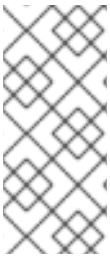
   例

   ```
   [ceph: root@host01 /]# ceph orch apply mon host02:10.10.128.69 host03:mynetwork
   ```

3.19. MANAGER サービスの追加

cephadm は、ブートストラッププロセス中にブートストラップノードに Manager デーモンを自動的にインストールします。Ceph オーケストレーターを使用して、追加の Manager デーモンをデプロイします。

Ceph オーケストレーターはデフォルトで 2 つの Manager デーモンをデプロイします。異なる数の Manager デーモンをデプロイするには、別の数を指定します。Manager デーモンがデプロイされるホストを指定しないと、Ceph オーケストレーターはホストをランダムに選択し、Manager デーモンをそれらにデプロイします。



注記

Manager デーモンを複数の特定のホストに適用する場合は、必ず同じ **ceph orch apply** コマンド内でホスト名をすべて指定してください。**ceph orch apply mgr --placement host1** を指定し、**ceph orch apply mgr --placement host2** を指定すると、2 つ目のコマンドにより、host1 の Manager デーモンが削除され、Manager デーモンが host2 に適用されます。

Red Hat は **--placement** オプションを使用して特定のホストにデプロイすることを推奨します。

前提条件

- 実行中のストレージクラスター。

手順

- 特定の数の Manager デーモンを無作為に選択したホストに適用することを指定するには、以下を実行します。

構文

```
ceph orch apply mgr NUMBER_OF_DAEMONS
```

例

```
[ceph: root@host01 /]# ceph orch apply mgr 3
```

- Manager デーモンをストレージクラスターの特定ホストに追加するには、以下を実行します。

構文

```
ceph orch apply mgr --placement "HOSTNAME1 HOSTNAME2 HOSTNAME3"
```

例

```
[ceph: root@host01 /]# ceph orch apply mgr --placement "host02 host03 host04"
```

3.20. OSD の追加

Cephadm は、利用できないデバイスに OSD をプロビジョニングしません。ストレージデバイスは、以下の条件すべてを満たす場合に利用可能であると見なされます。

- デバイスにはパーティションがない。
- デバイスをマウントしてはいけない。
- デバイスにはファイルシステムを含めることはできない。
- デバイスには Ceph BlueStore OSD を含めることはできない。
- デバイスは 5 GB を超える必要がある。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。

手順

1. OSD をデプロイするために利用可能なデバイスをリスト表示します。

   構文

   ```
   ceph orch device ls [--hostname=HOSTNAME1 HOSTNAME2] [--wide] [--refresh]
   ```

   例

   ```
   [ceph: root@host01 /]# ceph orch device ls --wide --refresh
   ```

2. OSD を特定のホストまたは利用可能なすべてのデバイスにデプロイできます。

   - 特定のホストの特定のデバイスから OSD を作成するには、以下を実行します。

構文

```
ceph orch daemon add osd HOSTNAME:DEVICE_PATH
```

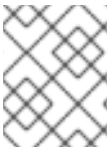
例

```
[ceph: root@host01 /]# ceph orch daemon add osd host02:/dev/sdb
```

- 使用可能な未使用のデバイスに OSD をデプロイするには、**--all-available-devices** オプションを使用します。

例

```
[ceph: root@host01 /]# ceph orch apply osd --all-available-devices
```



注記

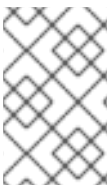
このコマンドは、併置された WAL および DB デーモンを作成します。コロケートされていないデーモンを作成する場合は、このコマンドを使用しないでください。

関連情報

- OSD のドライブ仕様の詳細は、**Red Hat Ceph Storage オペレーションガイド**の **OSD をデプロイするための高度なサービス指定およびフィルター** セクションを参照してください。
- デバイス上のデータを消去するための zapping デバイスの詳細は、**Red Hat Ceph Storage オペレーションガイド** の **Ceph OSD デプロイメントのデバイスの消去** セクションを参照してください。

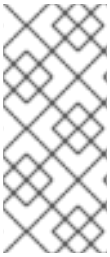
3.21. CEPHADM-CLIENTS PLAYBOOK の実行

cephadm-clients.yml Playbook は、設定および管理キーリングファイルの Ceph クライアントのグループに分散を処理します。



注記

Playbook の実行時に設定ファイルを指定しない場合には、Playbook は最小限の設定ファイルを生成し、配布します。デフォルトでは、生成されたファイルは **/etc/ceph/ceph.conf** にあります。



注記

cephadm-ansible Playbook を使用していない場合は、Ceph クラスターをアップグレードした後、クライアントノードの **ceph-common** パッケージとクライアントライブラリーをアップグレードする必要があります。詳細は、**Red Hat Ceph Storage アップグレードガイド** の **Red Hat Ceph Storage クラスターのアップグレード** セクションを参照してください。

前提条件

- Ansible 管理ノードへの root レベルのアクセス。

- ストレージクラスター内のすべてのノードへの sudo アクセスおよびパスワードなしの **ssh** アクセスのある Ansible ユーザー。
- **cephadm-ansible** パッケージがインストールされている。
- ストレージクラスターの初期ホストでプリフライト Playbook が実行されている。詳細は、**プリフライト Playbook の実行** を参照してください。
- **client_group** 変数は Ansible インベントリーファイルに指定する必要があります。
- **[admin]** グループは、管理キーリングが **/etc/ceph/ceph.client.admin.keyring** にあるノードを持つインベントリーファイルに定義されます。

手順

1. /usr/share/cephadm-ansible ディレクトリーに移動します。
2. クライアントグループの初期ホストで、**cephadm-clients.yml** Playbook を実行します。**PATH_TO_KEYRING** の admin ホストの admin キーリングへのフルパス名を使用します。オプション: 使用する既存の設定ファイルを指定する場合は、**CONFIG-FILE** の設定ファイルへの完全パスを指定します。**ANSIBLE_GROUP_NAME** のクライアントのグループには、Ansible グループ名を使用します。管理キーリングと設定ファイルが FSID 用に保存されるクラスターの **FSID** を使用します。FSID のデフォルトのパスは **/var/lib/ceph/** です。

   構文

   ```
   ansible-playbook -i hosts cephadm-clients.yml -extra-vars '{"fsid":"FSID",
   "client_group":"ANSIBLE_GROUP_NAME", "keyring":"PATH_TO_KEYRING",
   "conf":"CONFIG_FILE"}'
   ```

   例

   ```
   [ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-clients.yml --
   extra-vars '{"fsid":"be3ca2b2-27db-11ec-892b-
   005056833d58","client_group":"fs_clients","keyring":"/etc/ceph/fs.keyring", "conf":
   "/etc/ceph/ceph.conf"}'
   ```

インストールが完了すると、グループに指定したクライアントには管理キーリングが設定されます。設定ファイルを指定しない場合には、**cephadm-ansible** は各クライアントに最小限のデフォルト設定ファイルを作成します。

関連情報

- 管理キーの詳細は、**Red Hat Ceph Storage 管理ガイド** の **Ceph ユーザー管理** セクションを参照してください。

3.22. CEPHADM を使用したオペレーティングシステムのチューニングプロファイルの管理

ストレージ管理者は、**cephadm** を使用して、**sysctl** 設定を Red Hat Ceph Storage クラスター内の特定のホストに適用するオペレーティングシステムのチューニングプロファイルを作成および管理できます。オペレーティングシステムのチューニングを行うと、Red Hat Ceph Storage クラスターのパフォーマンスを向上できる機会が増えます。

関連情報

- カーネルパラメーターの設定について、詳しくは **sysctl (8)** の man ページを参照してください。
- チューニングされたプロファイルの詳細は、**TuneD プロファイルのカスタマイズ** を参照してください。

3.22.1. チューニングプロファイルの作成

カーネルパラメーターを含む YAML 仕様ファイルを作成するか、オーケストレーター CLI を使用してカーネルパラメーター設定を定義することで、チューニングプロファイルを作成できます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- 管理ホストへの root レベルのアクセス。
- **tuned** パッケージのインストール。

方法 1:

- YAML 仕様を作成して適用することで、チューニングプロファイルを作成します。
  a. Ceph 管理ホストから、YAML 仕様ファイルを作成します。

  構文

  ```
  touch TUNED_PROFILE_NAME.yaml
  ```

  例

  ```
  [root@host01 ~]# touch mon_hosts_profile.yaml
  ```

  b. YAML ファイルを編集して、チューニングパラメーターを含めます。

  構文

  ```
  profile_name: PROFILE_NAME
  placement:
    hosts:
      - HOST1
      - HOST2
  settings:
    SYSCTL_PARAMETER: SYSCTL_PARAMETER_VALUE
  ```

  例

  ```
  profile_name: mon_hosts_profile
  placement:
    hosts:
      - host01
      - host02
  ```

```
settings:
  fs.file-max: 1000000
  vm.swappiness: 13
```

c. チューニングプロファイルを適用します。

構文

```
ceph orch tuned-profile apply -i TUNED_PROFILE_NAME.yaml
```

例

```
[root@host01 ~]# ceph orch tuned-profile apply -i mon_hosts_profile.yaml

Saved tuned profile mon_hosts_profile
```

この例では、プロファイルを **host01** と **host02** の **/etc/sysctl.d/** に書き込み、各ホストで **sysctl --system** を実行して、再起動せずに sysctl 変数をリロードします。

注記

Cephadm は、プロファイルファイル名を **/etc/sysctl.d/** の下に **TUNED_PROFILE_NAME** -cephadm-tuned-profile.conf として書き込みます。ここでの **TUNED_PROFILE_NAME** は、YAML 仕様で指定した **profile_name** です。**sysctl** コマンドは、設定を行うファイルの名前の辞書的順序に従い、設定を適用します。複数のファイルに同じ設定が含まれている場合、辞書的順序で最新の名前を持つファイルのエントリーが優先されます。存在する可能性のある他の設定ファイルの前後に設定を適用するには、必要に応じて仕様ファイルに **profile_name** を設定します。

注記

Cephadm は、**sysctl** 設定をホストレベルでのみ適用し、特定のデーモンやコンテナーには適用しません。

方法 2:

- オーケストレーター CLI を使用してチューニングプロファイルを作成します。

a. Ceph 管理ホストから、チューニングプロファイルの名前、配置、設定を指定します。

構文

```
ceph orch tuned-profile apply PROFILE_NAME --placement='HOST1,HOST2' --
settings='SETTING_NAME1=VALUE1,SETTING_NAME2=VALUE2'
```

例

```
[root@host01 ~]# ceph orch tuned-profile apply osd_hosts_profile --
placement='host04,host05' --settings='fs.file-max=200000,vm.swappiness=19'

Saved tuned profile osd_hosts_profile
```

検証

- **cephadm** が管理しているチューニングプロファイルを一覧表示します。

  例

```
[root@host01 /]# ceph orch tuned-profile ls

profile_name: osd_hosts_profile
placement: host04;host05
settings:
  fs.file-max: 200000
  vm.swappiness: 19
```

3.22.2. チューニングプロファイルの表示

cephadm が管理するすべてのチューニングプロファイルを表示するには、**tuned-profile ls** コマンドを実行します。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- 管理ホストへの root レベルのアクセス。
- **tuned** パッケージのインストール。

手順

- Ceph 管理ホストから、チューニングプロファイルを一覧表示します。

  構文

```
ceph orch tuned-profile ls
```

  例

```
[root@host01 /]# ceph orch tuned-profile ls

profile_name: osd_hosts_profile
placement: host04;host05
settings:
  fs.file-max: 200000
  vm.swappiness: 19
---
profile_name: mon_hosts_profile
placement: host01;host02
settings:
  fs.file-max: 1000000
  vm.swappiness: 13
```

注記

プロファイルを変更して再適用する必要がある場合、**--format yaml** パラメーターを **tuned-profile ls** コマンドに渡すと、コピーして再適用できる形式でプロファイルが表示されます。

例

```
[root@host01 /]# ceph orch tuned-profile ls --format yaml

placement:
  hosts:
  - host01
  - host02
profile_name: mon_hosts_profile
settings:
  vm.swappiness: '13'
  fs.file-max: 1000000
```

3.22.3. チューニングプロファイルの変更

チューニングプロファイルを作成したら、既存のチューニングプロファイルを変更して、必要に応じて **sysctl** 設定を調整できます。

既存のチューニングプロファイルは、次の 2 つの方法で変更できます。

- 同じプロファイル名で YAML 仕様を再適用します。
- 設定を調整するには、**tuned-profile add-setting** および **rm-setting** パラメーターを使用します。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- 管理ホストへの root レベルのアクセス。
- **tuned** パッケージのインストール。

方法 1:

- **tuned-profile add-setting** および **rm-setting** パラメーターを使用して設定を変更します。
  a. Ceph 管理ホストから、既存のプロファイルの設定を追加または変更します。

  構文

  ```
  ceph orch tuned-profile add-setting PROFILE_NAME SETTING_NAME VALUE
  ```

  例

  ```
  [root@host01 ~]# ceph orch tuned-profile add-setting mon_hosts_profile
  vm.vfs_cache_pressure 110
  ```

```
Added setting vm.vfs_cache_pressure with value 110 to tuned profile mon_hosts_profile
```

b. 既存のプロファイルから設定を削除するには:

構文

```
ceph orch tuned-profile rm-setting PROFILE_NAME SETTING_NAME
```

例

```
[root@host01 ~]# ceph orch tuned-profile rm-setting mon_hosts_profile
vm.vfs_cache_pressure

Removed setting vm.vfs_cache_pressure from tuned profile mon_hosts_profile
```

方法 2:

- 同じプロファイル名で YAML 仕様を再適用して設定を変更します。

a. Ceph 管理ホストから、YAML 仕様ファイルを作成するか、既存の仕様ファイルを変更します。

構文

```
vi TUNED_PROFILE_NAME.yaml
```

例

```
[root@host01 ~]# vi mon_hosts_profile.yaml
```

b. YAML ファイルを編集して、変更する調整済みパラメーターを含めます。

構文

```
profile_name: PROFILE_NAME
placement:
  hosts:
    - HOST1
    - HOST2
settings:
  SYSCTL_PARAMETER: SYSCTL_PARAMETER_VALUE
```

例

```
profile_name: mon_hosts_profile
placement:
  hosts:
    - host01
    - host02
settings:
  fs.file-max: 2000000
  vm.swappiness: 15
```

c. チューニングプロファイルを適用します。

構文

```
ceph orch tuned-profile apply -i TUNED_PROFILE_NAME.yaml
```

例

```
[root@host01 ~]# ceph orch tuned-profile apply -i mon_hosts_profile.yaml

Saved tuned profile mon_hosts_profile
```

注記

配置を変更するには、同じ名前のプロファイルを再適用する必要があります。Cephadm は名前でプロファイルを追跡するため、既存のプロファイルと同じ名前のプロファイルを適用すると、古いプロファイルが上書きされます。

3.22.4. チューニングプロファイルの削除

ストレージ管理者は、**tuned-profile rm** コマンドを使用して、**cephadm** で管理する必要がなくなったチューニングプロファイルを削除できます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- 管理ホストへの root レベルのアクセス。
- **tuned** パッケージのインストール。

手順

1. Ceph 管理ホストから、**cephadm** が管理しているチューニングプロファイルを表示します。

   例

   ```
   [root@host01 ~]# ceph orch tuned-profile ls
   ```

2. チューニングプロファイルを削除します。

   構文

   ```
   ceph orch tuned-profile rm TUNED_PROFILE_NAME
   ```

   例

   ```
   [root@host01 ~]# ceph orch tuned-profile rm mon_hosts_profile

   Removed tuned profile mon_hosts_profile
   ```

cephadm がチューニングプロファイルを削除すると、対応するホストの **/etc/sysctl.d** ディレクトリーに書き込まれていたプロファイルファイルが削除されます。

3.23. CEPH ストレージクラスターのパージ

Ceph ストレージクラスターをパージすると、サーバー上の以前のデプロイメントから残っているデータまたは接続がすべて消去されます。Ansible はサポートされていないため、**cephadm rm-cluster** コマンドを使用します。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。

手順

1. **cephadm** を無効にしてすべてのオーケストレーション操作を停止し、新しいデーモンのデプロイを回避します。

   例

   ```
   [ceph: root#host01 /]# ceph mgr module disable cephadm
   ```

2. クラスターの FSID を取得します。

   例

   ```
   [ceph: root#host01 /]# ceph fsid
   ```

3. cephadm シェルを終了します。

   例

   ```
   [ceph: root@host01 /]# exit
   ```

4. クラスター内のすべてのホストから Ceph デーモンをパージします。

   構文

   ```
   cephadm rm-cluster --force --zap-osds  --fsid FSID
   ```

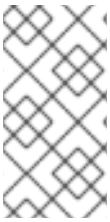
   例

   ```
   [root@host01 ~]# cephadm rm-cluster --force --zap-osds  --fsid a6ca415a-cde7-11eb-a41a-002590fc2544
   ```

3.24. クライアントノードのデプロイ

ストレージ管理者は、**cephadm-preflight.yml** および **cephadm-clients.yml** Playbook を実行してクライアントノードをデプロイできます。**cephadm-preflight.yml** Playbook は、Ceph リポジトリーを設定し、ブートストラップ用にストレージクラスターを準備します。また、**podman**、**lvm2**、**chronyd**、**cephadm** などのいくつかの前提条件もインストールします。

cephadm-clients.yml Playbook は、設定およびキーリングファイルの Ceph クライアントのグループに分散を処理します。



注記

cephadm-ansible Playbook を使用していない場合は、Ceph クラスターをアップグレードした後、クライアントノードの **ceph-common** パッケージとクライアントライブラリーをアップグレードする必要があります。詳細は、Red Hat Ceph Storage クラスターのアップグレード を参照してください。

前提条件

- Ansible 管理ノードへの root レベルのアクセス。
- ストレージクラスター内のすべてのノードへの sudo アクセスおよびパスワードなしの **ssh** アクセスのある Ansible ユーザー。
- **cephadm-ansible** パッケージのインストール。
- **[admin]** グループは、管理キーリングが **/etc/ceph/ceph.client.admin.keyring** にあるノードを持つインベントリーファイルに定義されます。

手順

1. Ansible ユーザーとして、Ansible 管理ノードの **/usr/share/cephadm-ansible** ディレクトリーに移動します。

   例

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

2. **hosts** インベントリーファイルを開いて編集し、**[clients]** グループとクライアントをインベントリーに追加します。

   例

   ```
   host02
   host03
   host04

   [admin]
   host01

   [clients]
   client01
   client02
   client03
   ```

3. **cephadm-preflight.yml** playbook を実行して、前提条件をクライアントにインストールします。

   構文

```
ansible-playbook -i INVENTORY_FILE cephadm-preflight.yml --limit
CLIENT_GROUP_NAME|CLIENT_NODE_NAME
```

例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml --
limit clients
```

4. **cephadm-clients.yml** playbook を実行して、キーリングと Ceph 設定ファイルを一連のクライアントに配布します。

   a. カスタム宛先キーリング名でキーリングをコピーするには、以下を実行します。

   構文

   ```
   ansible-playbook -i INVENTORY_FILE cephadm-clients.yml --extra-vars
   '{"fsid":"FSID","keyring":"KEYRING_PATH","client_group":"CLIENT_GROUP_NAME","c
   onf":"CEPH_CONFIGURATION_PATH","keyring_dest":"KEYRING_DESTINATION_PA
   TH"}'
   ```

   - **INVENTORY_FILE** を Ansible インベントリーのファイル名に置き換えます。
   - **FSID** をクラスターの FSID に置き換えます。
   - **KEYRING_PATH** を、クライアントにコピーする管理ホスト上のキーリングへのフルパス名に置き換えます。
   - オプション: **CLIENT_GROUP_NAME** を、セットアップするクライアントの Ansible グループ名に置き換えます。
   - オプション: **CEPH_CONFIGURATION_PATH** を、管理ノード上の Ceph 設定ファイルへのフルパスに置き換えます。
   - オプション: **KEYRING_DESTINATION_PATH** を、キーリングがコピーされる宛先の絶対パス名に置き換えます。

   注記

   Playbook の実行時に conf オプションで設定ファイルを指定しない場合、Playbook は最小限の設定ファイルを生成して配布します。デフォルトでは、生成されたファイルは **/etc/ceph/ceph.conf** にあります。

   例

   ```
   [ceph-admin@host01 cephadm-ansible]$ ansible-playbook -i hosts
   cephadm-clients.yml --extra-vars '{"fsid":"266ee7a8-2a05-11eb-b846-
   5254002d4916","keyring":"/etc/ceph/ceph.client.admin.keyring","client_g
   roup":"clients","conf":"/etc/ceph/ceph.conf","keyring_dest":"/etc/ceph/cus
   tom.name.ceph.keyring"}'
   ```

   b. デフォルトの宛先キーリング名 **ceph.keyring** でキーリングをコピーし、デフォルトのクライアントグループを使用するには、以下を実行します。

構文

```
ansible-playbook -i INVENTORY_FILE cephadm-clients.yml --extra-vars
'{"fsid":"FSID","keyring":"KEYRING_PATH","conf":"CONF_PATH"}'
```

検証

クライアントノードにログインし、キーリングと設定ファイルが存在することを確認します。

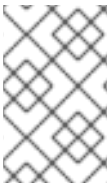
例

```
[user@client01 ~]# ls -l /etc/ceph/

-rw-------. 1 ceph ceph 151 Jul 11 12:23 custom.name.ceph.keyring
-rw-------. 1 ceph ceph 151 Jul 11 12:23 ceph.keyring
-rw-------. 1 ceph ceph 269 Jul 11 12:23 ceph.conf
```

第4章 CEPHADM-ANSIBLE モジュールを使用した RED HAT CEPH STORAGE クラスターの管理

ストレージ管理者として、Ansible Playbook で **cephadm-ansible** モジュールを使用して、Red Hat Ceph Storage クラスターを管理することができます。**cephadm-ansible** パッケージは、クラスターを管理するための独自の Ansible Playbook を作成できるように、**cephadm** 呼び出しをラップするいくつかのモジュールを提供します。



注記

現時点では、**cephadm-ansible** モジュールは最も重要なタスクのみをサポートしています。**cephadm-ansible** モジュールでカバーされていない操作は、Playbook で **command** または **shell** Ansible モジュールを使用して完了する必要があります。

4.1. CEPHADM-ANSIBLE モジュール

cephadm-ansible モジュールは、**cephadm** および **ceph orch** コマンドのラッパーを提供することで、Ansible Playbook の作成を簡素化するモジュールのコレクションです。モジュールを使用して独自の Ansible Playbook を作成し、1 つ以上のモジュールを使用してクラスターを管理できます。

cephadm-ansible パッケージには、次のモジュールが含まれています。

- **cephadm_bootstrap**
- **ceph_orch_host**
- **ceph_config**
- **ceph_orch_apply**
- **ceph_orch_daemon**
- **cephadm_registry_login**

4.2. CEPHADM-ANSIBLE モジュールのオプション

次の表に、**cephadm-ansible** モジュールで使用可能なオプションを示します。Ansible Playbook でモジュールを使用する場合は、必須としてリストされているオプションを設定する必要があります。デフォルト値 **true** でリストされているオプションは、モジュールの使用時にオプションが自動的に設定され、Playbook で指定する必要がないことを示します。たとえば、**cephadm_bootstrap** モジュールの場合、**dashboard: false** を設定しない限り、Ceph Dashboard がインストールされます。

表4.1 cephadm_bootstrap モジュールで利用可能なオプション

cephadm_bootstrap	説明	必須	デフォルト
mon_ip	Ceph Monitor の IP アドレス。	true	
image	Ceph コンテナーイメージ。	false	

cephadm_bootstrap	説明	必須	デフォルト
docker	**podman** の代わりに **docker** を使用します。	false	
fsid	Ceph FSID を定義します。	false	
pull	Ceph コンテナーイメージをプルします。	false	true
dashboard	Ceph Dashboard をデプロイします。	false	true
dashboard_user	特定の Ceph Dashboard ユーザーを指定します。	false	
dashboard_password	Ceph Dashboard のパスワード。	false	
monitoring	モニタリングスタックをデプロイします。	false	true
firewalld	firewalld を使用してファイアウォールルールを管理します。	false	true
allow_overwrite	既存の --output-config、--output-keyring、または --output-pub-ssh-key ファイルの上書きを許可します。	false	false
registry_url	カスタムレジストリーの URL。	false	
registry_username	カスタムレジストリーのユーザー名。	false	
registry_password	カスタムレジストリーのパスワード。	false	
registry_json	カスタムレジストリーログイン情報を含む JSON ファイル。	false	

cephadm_bootstrap	説明	必須	デフォルト
ssh_user	ホストへの **cephadm** ssh に使用する SSH ユーザー。	false	
ssh_config	**cephadm** SSH クライアントの SSH 設定ファイルのパス。	false	
allow_fqdn_hostname	完全修飾ドメイン名 (FQDN) であるホスト名を許可します。	false	false
cluster_network	クラスターのレプリケーション、リカバリー、およびハートビートに使用するサブネット。	false	

表4.2 ceph_orch_host モジュールで使用可能なオプション

ceph_orch_host	説明	必須	デフォルト
fsid	対話する Ceph クラスターの FSID。	false	
image	使用する Ceph コンテナーイメージ。	false	
name	追加、削除、または更新するホストの名前。	true	
address	ホストの IP アドレス。	**state** が **present** である場合は true。	
set_admin_label	指定したホストに **_admin** ラベルを設定します。	false	false
labels	ホストに適用するラベルのリスト。	false	[]

ceph_orch_host	説明	必須	デフォルト
state	**present** に設定すると、name で指定された **name** が存在することが保証されます。**absent** に設定すると、**name** で指定されたホストが削除されます。**drain** に設定すると、**name** で指定されたホストからすべてのデーモンを削除するようにスケジュールされます。	false	あり

表4.3 ceph_config モジュールで利用可能なオプション

ceph_config	説明	必須	デフォルト
fsid	対話する Ceph クラスターの FSID。	false	
image	使用する Ceph コンテナーイメージ。	false	
action	**option** で指定されたパラメーターを **set** または **get** するかどうか。	false	set
who	設定を設定するデーモン。	true	
option	**set** または **get** するパラメーターの名前。	true	
値	設定するパラメーターの値。	アクションが **set** である場合は true	

表4.4 ceph_orch_apply モジュールで使用可能なオプション

ceph_orch_apply	説明	必須
fsid	対話する Ceph クラスターの FSID。	false
image	使用する Ceph コンテナーイメージ。	false

ceph_orch_apply	説明	必須
spec	適用するサービス仕様。	true

表4.5 ceph_orch_daemon モジュールで使用可能なオプション

ceph_orch_daemon	説明	必須
fsid	対話する Ceph クラスターの FSID。	false
image	使用する Ceph コンテナーイメージ。	false
state	**name** で指定されたサービスの望ましい状態。	true **started** の場合、サービスが確実に開始されます。 **stopped** の場合、サービスが確実に停止されます。 **restarted** の場合、サービスが再起動されます。
daemon_id	サービスの ID。	true
daemon_type	サービスのタイプ。	true

表4.6 cephadm_registry_login モジュールで利用可能なオプション

cephadm_registry_login	説明	必須	デフォルト
state	レジストリーのログインまたはログアウト。	false	login
docker	**podman** の代わりに **docker** を使用します。	false	
registry_url	カスタムレジストリーの URL。	false	
registry_username	カスタムレジストリーのユーザー名。	**state** が **login** の場合は **true**。	
registry_password	カスタムレジストリーのパスワード。	**state** が **login** の場合は **true**。	

cephadm_registry_login	説明	必須	デフォルト
registry_json	JSON ファイルへのパス。このファイルは、このタスクを実行する前にリモートホストに存在している必要があります。このオプションは現在サポートされていません。		

4.3. CEPHADM_BOOTSTRAP および CEPHADM_REGISTRY_LOGIN モジュールを使用したストレージクラスターのブートストラップ

ストレージ管理者は、Ansible Playbook で **cephadm_bootstrap** および **cephadm_registry_login** モジュールを使用して、Ansible を使用してストレージクラスターをブートストラップできます。

前提条件

- 最初の Ceph Monitor コンテナーの IP アドレス。これはストレージクラスターの最初のノードの IP アドレスでもあります。
- **registry.redhat.io** へのログインアクセス。
- 少なくとも 10 GB の空き容量がある **/var/lib/containers/**。
- **ansible-core** が AppStream にバンドルされている Red Hat Enterprise Linux 9.2。
- Ansible 管理ノードへの **cephadm-ansible** パッケージのインストール。
- パスワードなしの SSH がストレージクラスター内のすべてのホストに設定されます。
- ホストは CDN に登録されます。

手順

1. Ansible 管理ノードにログインします。
2. Ansible 管理ノードの **/usr/share/cephadm-ansible** ディレクトリーに移動します。

   例

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

3. **hosts** ファイルを作成し、ホスト、ラベルを追加し、ストレージクラスター内の最初のホストの IP アドレスを監視します。

   構文

   ```
   sudo vi INVENTORY_FILE

   HOST1 labels="['LABEL1', 'LABEL2']"
   ```

```
HOST2 labels="['LABEL1', 'LABEL2']"
HOST3 labels="['LABEL1']"

[admin]
ADMIN_HOST monitor_address=MONITOR_IP_ADDRESS labels="['ADMIN_LABEL',
'LABEL1', 'LABEL2']"
```

例

```
[ceph-admin@admin cephadm-ansible]$ sudo vi hosts

host02 labels="['mon', 'mgr']"
host03 labels="['mon', 'mgr']"
host04 labels="['osd']"
host05 labels="['osd']"
host06 labels="['osd']"

[admin]
host01 monitor_address=10.10.128.68 labels="['_admin', 'mon', 'mgr']"
```

4. プリフライト Playbook を実行します。

構文

```
ansible-playbook -i INVENTORY_FILE cephadm-preflight.yml --extra-vars
"ceph_origin=rhcs"
```

例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml --
extra-vars "ceph_origin=rhcs"
```

5. クラスターをブートストラップする Playbook を作成します。

構文

```
sudo vi PLAYBOOK_FILENAME.yml

---
- name: NAME_OF_PLAY
  hosts: BOOTSTRAP_HOST
  become: USE_ELEVATED_PRIVILEGES
  gather_facts: GATHER_FACTS_ABOUT_REMOTE_HOSTS
  tasks:
    -name: NAME_OF_TASK
     cephadm_registry_login:
       state: STATE
       registry_url: REGISTRY_URL
       registry_username: REGISTRY_USER_NAME
       registry_password: REGISTRY_PASSWORD

    - name: NAME_OF_TASK
      cephadm_bootstrap:
        mon_ip: "{{ monitor_address }}"
```

```
        dashboard_user: DASHBOARD_USER
        dashboard_password: DASHBOARD_PASSWORD
        allow_fqdn_hostname: ALLOW_FQDN_HOSTNAME
        cluster_network: NETWORK_CIDR
```

例

```
[ceph-admin@admin cephadm-ansible]$ sudo vi bootstrap.yml

---
- name: bootstrap the cluster
  hosts: host01
  become: true
  gather_facts: false
  tasks:
    - name: login to registry
      cephadm_registry_login:
        state: login
        registry_url: registry.redhat.io
        registry_username: user1
        registry_password: mypassword1

    - name: bootstrap initial cluster
      cephadm_bootstrap:
        mon_ip: "{{ monitor_address }}"
        dashboard_user: mydashboarduser
        dashboard_password: mydashboardpassword
        allow_fqdn_hostname: true
        cluster_network: 10.10.128.0/28
```

6. Playbook を実行します。

構文

```
ansible-playbook -i INVENTORY_FILE PLAYBOOK_FILENAME.yml -vvv
```

例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts bootstrap.yml -vvv
```

検証

- Playbook を実行した後、Ansible の出力を確認します。

4.4. CEPH_ORCH_HOST モジュールを使用したホストの追加または削除

ストレージ管理者は、Ansible Playbook の **ceph_orch_host** モジュールを使用して、ストレージクラスター内のホストを追加および削除できます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。

- ノードを CDN に登録して、サブスクリプションを割り当てます。
- ストレージクラスター内のすべてのノードへの sudo アクセスおよびパスワードなしの SSH アクセスのある Ansible ユーザー。
- Ansible 管理ノードへの **cephadm-ansible** パッケージのインストール。
- 新しいホストには、ストレージクラスターの公開 SSH キーがあります。ストレージクラスターの公開 SSH キーを新しいホストにコピーする方法の詳細については、**ホストの追加** を参照してください。

手順

1. 次の手順を使用して、新しいホストをクラスターに追加します。
   a. Ansible 管理ノードにログインします。
   b. Ansible 管理ノードの **/usr/share/cephadm-ansible** ディレクトリーに移動します。

      例

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

   c. 新しいホストとラベルを Ansible インベントリーファイルに追加します。

      構文

      ```
      sudo vi INVENTORY_FILE

      NEW_HOST1 labels="['LABEL1', 'LABEL2']"
      NEW_HOST2 labels="['LABEL1', 'LABEL2']"
      NEW_HOST3 labels="['LABEL1']"

      [admin]
      ADMIN_HOST monitor_address=MONITOR_IP_ADDRESS labels="['ADMIN_LABEL',
      'LABEL1', 'LABEL2']"
      ```

      例

      ```
      [ceph-admin@admin cephadm-ansible]$ sudo vi hosts

      host02 labels="['mon', 'mgr']"
      host03 labels="['mon', 'mgr']"
      host04 labels="['osd']"
      host05 labels="['osd']"
      host06 labels="['osd']"

      [admin]
      host01 monitor_address= 10.10.128.68 labels="['_admin', 'mon', 'mgr']"
      ```

   d. **--limit** オプションを指定して、プリフライト Playbook を実行します。

      構文

```
ansible-playbook -i INVENTORY_FILE cephadm-preflight.yml --extra-vars
"ceph_origin=rhcs" --limit NEWHOST
```

例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml
--extra-vars "ceph_origin=rhcs" --limit host02
```

プリフライト Playbook は、新しいホストに **podman**、**lvm2**、**chronyd**、および **cephadm** をインストールします。インストールが完了すると、**cephadm** は **/usr/sbin/** ディレクトリーに配置されます。

e. 新しいホストをクラスターに追加する Playbook を作成します。

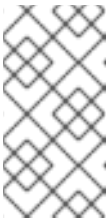
構文

```
sudo vi PLAYBOOK_FILENAME.yml

---
- name: PLAY_NAME
  hosts: HOSTS_OR_HOST_GROUPS
  become: USE_ELEVATED_PRIVILEGES
  gather_facts: GATHER_FACTS_ABOUT_REMOTE_HOSTS
  tasks:
    - name: NAME_OF_TASK
      ceph_orch_host:
        name: "{{ ansible_facts['hostname'] }}"
        address: "{{ ansible_facts['default_ipv4']['address'] }}"
        labels: "{{ labels }}"
      delegate_to: HOST_TO_DELEGATE_TASK_TO

    - name: NAME_OF_TASK
      when: inventory_hostname in groups['admin']
      ansible.builtin.shell:
        cmd: CEPH_COMMAND_TO_RUN
      register: REGISTER_NAME

    - name: NAME_OF_TASK
      when: inventory_hostname in groups['admin']
      debug:
        msg: "{{ REGISTER_NAME.stdout }}"
```



注記

デフォルトでは、Ansible は Playbook の **hosts** 行に一致するホストですべてのタスクを実行します。**ceph orch** コマンドは、管理キーリングと Ceph 設定ファイルを含むホストで実行する必要があります。**delegate_to** キーワードを使用して、クラスター内の管理ホストを指定します。

例

```
[ceph-admin@admin cephadm-ansible]$ sudo vi add-hosts.yml
```

```
---
- name: add additional hosts to the cluster
  hosts: all
  become: true
  gather_facts: true
  tasks:
    - name: add hosts to the cluster
      ceph_orch_host:
        name: "{{ ansible_facts['hostname'] }}"
        address: "{{ ansible_facts['default_ipv4']['address'] }}"
        labels: "{{ labels }}"
      delegate_to: host01

    - name: list hosts in the cluster
      when: inventory_hostname in groups['admin']
      ansible.builtin.shell:
        cmd: ceph orch host ls
      register: host_list

    - name: print current list of hosts
      when: inventory_hostname in groups['admin']
      debug:
        msg: "{{ host_list.stdout }}"
```

この例では、Playbook は新しいホストをクラスターに追加し、ホストの現在のリストを表示します。

f. Playbook を実行して、追加のホストをクラスターに追加します。

構文

```
ansible-playbook -i INVENTORY_FILE PLAYBOOK_FILENAME.yml
```

例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts add-hosts.yml
```

2. 次の手順を使用して、クラスターからホストを削除します。

a. Ansible 管理ノードにログインします。

b. Ansible 管理ノードの **/usr/share/cephadm-ansible** ディレクトリーに移動します。

例

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

c. クラスターからホストを削除する Playbook を作成します。

構文

```
sudo vi PLAYBOOK_FILENAME.yml

---
```

```
- name: NAME_OF_PLAY
  hosts: ADMIN_HOST
  become: USE_ELEVATED_PRIVILEGES
  gather_facts: GATHER_FACTS_ABOUT_REMOTE_HOSTS
  tasks:
    - name: NAME_OF_TASK
      ceph_orch_host:
        name: HOST_TO_REMOVE
        state: STATE

    - name: NAME_OF_TASK
      ceph_orch_host:
        name: HOST_TO_REMOVE
        state: STATE
      retries: NUMBER_OF_RETRIES
      delay: DELAY
      until: CONTINUE_UNTIL
      register: REGISTER_NAME

    - name: NAME_OF_TASK
      ansible.builtin.shell:
        cmd: ceph orch host ls
      register: REGISTER_NAME

    - name: NAME_OF_TASK
        debug:
          msg: "{{ REGISTER_NAME.stdout }}"
```

例

```
[ceph-admin@admin cephadm-ansible]$ sudo vi remove-hosts.yml

---
- name: remove host
  hosts: host01
  become: true
  gather_facts: true
  tasks:
    - name: drain host07
      ceph_orch_host:
        name: host07
        state: drain

    - name: remove host from the cluster
      ceph_orch_host:
        name: host07
        state: absent
      retries: 20
      delay: 1
      until: result is succeeded
      register: result

     - name: list hosts in the cluster
       ansible.builtin.shell:
         cmd: ceph orch host ls
       register: host_list
```

```
- name: print current list of hosts
  debug:
    msg: "{{ host_list.stdout }}"
```

この例では、playbook タスクは **host07** 上のすべてのデーモンをドレインし、クラスターからホストを削除し、ホストの現在のリストを表示します。

d. Playbook を実行して、クラスターからホストを削除します。

構文

```
ansible-playbook -i INVENTORY_FILE PLAYBOOK_FILENAME.yml
```

例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts remove-hosts.yml
```

検証

- クラスター内のホストの現在のリストを表示する Ansible タスクの出力を確認します。

例

```
TASK [print current hosts]
*****************************************************************************************************
Friday 24 June 2022  14:52:40 -0400 (0:00:03.365)       0:02:31.702 ***********
ok: [host01] =>
  msg: |-
    HOST    ADDR          LABELS         STATUS
    host01  10.10.128.68  _admin mon mgr
    host02  10.10.128.69  mon mgr
    host03  10.10.128.70  mon mgr
    host04  10.10.128.71  osd
    host05  10.10.128.72  osd
    host06  10.10.128.73  osd
```

4.5. CEPH_CONFIG モジュールを使用した設定オプションの設定

ストレージ管理者は、**ceph_config** モジュールを使用して Red Hat Ceph Storage 設定オプションを設定または取得できます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ストレージクラスター内のすべてのノードへの sudo アクセスおよびパスワードなしの SSH アクセスのある Ansible ユーザー。
- Ansible 管理ノードへの **cephadm-ansible** パッケージのインストール。

- Ansible インベントリーファイルには、クラスターと管理ホストが含まれている。ホストをストレージクラスターに追加する方法の詳細については、**ceph_orch_host モジュールを使用したホストの追加または削除** を参照してください。

手順

1. Ansible 管理ノードにログインします。
2. Ansible 管理ノードの **/usr/share/cephadm-ansible** ディレクトリーに移動します。

   例

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

3. 設定を変更して Playbook を作成します。

   構文

   ```
   sudo vi PLAYBOOK_FILENAME.yml

   ---
   - name: PLAY_NAME
     hosts: ADMIN_HOST
     become: USE_ELEVATED_PRIVILEGES
     gather_facts: GATHER_FACTS_ABOUT_REMOTE_HOSTS
     tasks:
       - name: NAME_OF_TASK
         ceph_config:
           action: GET_OR_SET
           who: DAEMON_TO_SET_CONFIGURATION_TO
           option: CEPH_CONFIGURATION_OPTION
           value: VALUE_OF_PARAMETER_TO_SET

       - name: NAME_OF_TASK
         ceph_config:
           action: GET_OR_SET
           who: DAEMON_TO_SET_CONFIGURATION_TO
           option: CEPH_CONFIGURATION_OPTION
         register: REGISTER_NAME

       - name: NAME_OF_TASK
         debug:
           msg: "MESSAGE_TO_DISPLAY {{ REGISTER_NAME.stdout }}"
   ```

   例

   ```
   [ceph-admin@admin cephadm-ansible]$ sudo vi change_configuration.yml

   ---
   - name: set pool delete
     hosts: host01
     become: true
     gather_facts: false
     tasks:
       - name: set the allow pool delete option
   ```

```
    ceph_config:
      action: set
      who: mon
      option: mon_allow_pool_delete
      value: true

  - name: get the allow pool delete setting
    ceph_config:
      action: get
      who: mon
      option: mon_allow_pool_delete
    register: verify_mon_allow_pool_delete

  - name: print current mon_allow_pool_delete setting
    debug:
      msg: "the value of 'mon_allow_pool_delete' is {{ verify_mon_allow_pool_delete.stdout }}"
```

この例では、Playbook は最初に **mon_allow_pool_delete** オプションを **false** に設定します。その後、Playbook は現在の **mon_allow_pool_delete** 設定を取得し、その値を Ansible 出力に表示します。

4. Playbook を実行します。

構文

```
ansible-playbook -i INVENTORY_FILE _PLAYBOOK_FILENAME.yml
```

例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts change_configuration.yml
```

検証

- Playbook タスクからの出力を確認します。

例

```
TASK [print current mon_allow_pool_delete setting]
*************************************************************
Wednesday 29 June 2022  13:51:41 -0400 (0:00:05.523)       0:00:17.953 ********
ok: [host01] =>
  msg: the value of 'mon_allow_pool_delete' is true
```

関連情報

- 設定オプションの詳細は、Red Hat Ceph Storage 設定ガイド を参照してください。

4.6. CEPH_ORCH_APPLY モジュールを使用したサービス仕様の適用

ストレージ管理者は、Ansible Playbook の **ceph_orch_apply** モジュールを使用して、ストレージクラスターにサービス仕様を適用できます。サービス仕様は、Ceph サービスのデプロイに使用されるサービス属性および設定を指定するデータ構造です。サービス仕様を使用し

て、**mon**、**crash**、**mds**、**mgr**、**osd**、**rdb**、または **rbd-mirror** などの Ceph サービスタイプをデプロイできます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ストレージクラスター内のすべてのノードへの sudo アクセスおよびパスワードなしの SSH アクセスのある Ansible ユーザー。
- Ansible 管理ノードへの **cephadm-ansible** パッケージのインストール。
- Ansible インベントリーファイルには、クラスターと管理ホストが含まれている。ホストをストレージクラスターに追加する方法の詳細については、**ceph_orch_host モジュールを使用したホストの追加または削除** を参照してください。

手順

1. Ansible 管理ノードにログインします。
2. Ansible 管理ノードの **/usr/share/cephadm-ansible** ディレクトリーに移動します。

   例

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

3. サービス仕様を使用して Playbook を作成します。

   構文

   ```
   sudo vi PLAYBOOK_FILENAME.yml

   ---
   - name: PLAY_NAME
     hosts: HOSTS_OR_HOST_GROUPS
     become: USE_ELEVATED_PRIVILEGES
     gather_facts: GATHER_FACTS_ABOUT_REMOTE_HOSTS
     tasks:
       - name: NAME_OF_TASK
         ceph_orch_apply:
           spec: |
             service_type: SERVICE_TYPE
             service_id: UNIQUE_NAME_OF_SERVICE
             placement:
               host_pattern: 'HOST_PATTERN_TO_SELECT_HOSTS'
               label: LABEL
             spec:
               SPECIFICATION_OPTIONS:
   ```

   例

   ```
   [ceph-admin@admin cephadm-ansible]$ sudo vi deploy_osd_service.yml

   ---
   ```

```
- name: deploy osd service
  hosts: host01
  become: true
  gather_facts: true
  tasks:
    - name: apply osd spec
      ceph_orch_apply:
        spec: |
          service_type: osd
          service_id: osd
          placement:
            host_pattern: '*'
            label: osd
          spec:
            data_devices:
              all: true
```

この例では、Playbook はラベル **osd** を持つすべてのホストに Ceph OSD サービスをデプロイします。

4. Playbook を実行します。

構文

```
ansible-playbook -i INVENTORY_FILE _PLAYBOOK_FILENAME.yml
```

例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts deploy_osd_service.yml
```

検証

- Playbook タスクからの出力を確認します。

関連情報

- サービス仕様オプションの詳細は、Red Hat Ceph Storage Operations Guide を参照してください。

4.7. CEPH_ORCH_DAEMON モジュールを使用した CEPH デーモンの状態の管理

ストレージ管理者は、Ansible Playbook の **ceph_orch_daemon** モジュールを使用して、ホストで Ceph デーモンを開始、停止、および再起動できます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスターがある。
- ストレージクラスター内のすべてのノードへの sudo アクセスおよびパスワードなしの SSH アクセスのある Ansible ユーザー。
- Ansible 管理ノードへの **cephadm-ansible** パッケージのインストール。

- Ansible インベントリーファイルには、クラスターと管理ホストが含まれている。ホストをストレージクラスターに追加する方法の詳細については、**ceph_orch_host モジュールを使用したホストの追加または削除** を参照してください。

手順

1. Ansible 管理ノードにログインします。
2. Ansible 管理ノードの **/usr/share/cephadm-ansible** ディレクトリーに移動します。

   例

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

3. デーモンの状態が変化する Playbook を作成します。

   構文

   ```
   sudo vi PLAYBOOK_FILENAME.yml

   ---
   - name: PLAY_NAME
     hosts: ADMIN_HOST
     become: USE_ELEVATED_PRIVILEGES
     gather_facts: GATHER_FACTS_ABOUT_REMOTE_HOSTS
     tasks:
       - name: NAME_OF_TASK
         ceph_orch_daemon:
           state: STATE_OF_SERVICE
           daemon_id: DAEMON_ID
           daemon_type: TYPE_OF_SERVICE
   ```

   例

   ```
   [ceph-admin@admin cephadm-ansible]$ sudo vi restart_services.yml

   ---
   - name: start and stop services
     hosts: host01
     become: true
     gather_facts: false
     tasks:
       - name: start osd.0
         ceph_orch_daemon:
           state: started
           daemon_id: 0
           daemon_type: osd

       - name: stop mon.host02
         ceph_orch_daemon:
           state: stopped
           daemon_id: host02
           daemon_type: mon
   ```

この例では、Playbook は ID **0** で OSD を開始し、ID が **host02** の Ceph Monitor を停止します。

4. Playbook を実行します。

構文

```
ansible-playbook -i INVENTORY_FILE _PLAYBOOK_FILENAME.yml
```

例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts restart_services.yml
```

検証

- Playbook タスクからの出力を確認します。

第5章 次のステップDAY 2

ストレージ管理者は、Red Hat Ceph Storage 7 のインストールと設定が完了したら、ストレージクラスターに対して Day Two 操作を実行する準備が整います。これらの操作には、メタデータサーバー (MDS) およびオブジェクトゲートウェイ (RGW) の追加や、NFS などのサービスの設定が含まれます。

cephadm オーケストレーターを使用して Day Two 操作を実行する方法の詳細は、Red Hat Ceph Storage 7 Operations Guide を参照してください。

Day Two 操作で Ceph Object Gateway をデプロイ、設定および管理するには、Red Hat Ceph Storage 7 Object Gateway Guide を参照してください。

付録A CEPH ANSIBLE と CEPHADM の比較

Cephadm は、ストレージクラスターのコンテナー化されたデプロイメントに使用します。

以下の表は、Day One と Day Two 操作で Ceph クラスターのコンテナー化されたデプロイメントを管理する場合の Cephadm と Ceph-Ansible Playbook を比較しています。

表A.1 Day One 操作

説明	Ceph-Ansible	Cephadm
Red Hat Ceph Storage クラスターのインストール	**site-container.yml** Playbook を実行します。	**cephadm bootstrap** コマンドを実行して、管理ノードでクラスターをブートストラップします。
ホストの追加	Ceph Ansible インベントリーを使用します。	**ceph orch add host HOST_NAME** を実行して、ホストをクラスターに追加します。
モニターの追加	**add-mon.yml** Playbook を実行します。	**ceph orch apply mon** コマンドを実行します。
マネージャーの追加	**site-container.yml** Playbook を実行します。	**ceph orch apply mgr** コマンドを実行します。
OSD の追加	**add-osd.yml** Playbook を実行します。	**ceph orch apply osd** コマンドを実行して、利用可能なすべてのデバイスまたは特定のホストに OSD を追加します。
特定デバイスでの OSD の追加	**osd.yml** ファイルの **devices** を選択し、**add-osd.yml** Playbook を実行します。	**osd.yml** ファイルの **data_devices** 下にある **paths** フィルターを選択し、**ceph orch apply -i FILE_NAME.yml** コマンドを実行します。
MDS の追加	**site-container.yml** Playbook を実行します。	**ceph orch apply FILESYSTEM_NAME** コマンドを実行して MDS を追加します。
Ceph Object Gateway の追加	**site-container.yml** Playbook を実行します。	**ceph orch apply rgw** コマンドを実行して、Ceph Object Gateway を追加します。

表A.2 Day Two 操作

説明	Ceph-Ansible	Cephadm

説明	Ceph-Ansible	Cephadm
ホストの削除	Ansible インベントリーを使用します。	**ceph orch host rm HOST_NAME** を実行してホストを削除します。
モニターの削除	**shrink-mon.yml** Playbook を実行します。	**ceph orch apply mon** を実行して、他のモニターを再デプロイします。
マネージャーの削除	**shrink-mon.yml** Playbook を実行します。	**ceph orch apply mgr** を実行して、他のマネージャーを再デプロイします。
OSD の削除	**shrink-osd.yml** Playbook を実行します。	**ceph orch osd rm OSD_ID** を実行して OSD を削除します。
MDS の削除	**shrink-mds.yml** Playbook を実行します。	**ceph orch rm SERVICE_NAME** を実行して特定のサービスを削除します。
NFS プロトコル上の Ceph ファイルシステムのエクスポート	Red Hat Ceph Storage 4 ではサポートされません。	**ceph nfs export create** コマンドを実行します。
Ceph Object Gateway のデプロイメント	**site-container.yml** Playbook を実行します。	**ceph orch apply rgw SERVICE_NAME** を実行して Ceph Object Gateway サービスをデプロイします。
Ceph Object Gateway の削除	**shrink-rgw.yml** Playbook を実行します。	**ceph orch rm SERVICE_NAME** を実行して特定のサービスを削除します。
ブロックデバイスのミラーリング	**site-container.yml** Playbook を実行します。	**ceph orch apply rbd-mirror** コマンドを実行します。
Red Hat Ceph Storage のマイナーバージョンアップグレード	**infrastructure-playbooks/rolling_update.yml** Playbook を実行します。	**ceph orch upgrade start** コマンドを実行します。
モニタリングスタックのデプロイメント	インストール時に **all.yml** ファイルを編集します。	サービスを指定した後に **ceph orch apply -i FILE.yml** を実行します。

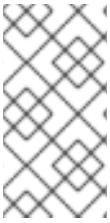
関連情報

- Ceph Orchestrator の使用に関する詳細は、Red Hat Ceph Storage オペレーションガイド を参照してください。

付録B CEPHADM コマンド

cephadm は、Cephadm Orchestrator のローカルホストを管理するコマンドラインツールです。現在のホストの状態を調査および変更するコマンドを提供します。

通常、コマンドの一部はデバッグに使用されます。



注記

cephadm はすべてのホストでは必要ありませんが、特定のデーモンを調査するときに便利です。**cephadm-ansible-preflight** Playbook はすべてのホストに **cephadm** をインストールし、**cephadm-ansible purge** Playbook では、適切に機能させるには、全ホストに **cephadm** をインストールする必要があります。

adopt

説明

アップグレードしたストレージクラスターデーモンを変換して、**cephadm** を実行します。

構文

```
cephadm adopt [-h] --name DAEMON_NAME --style STYLE [--cluster CLUSTER] --legacy-dir
[LEGACY_DIR] --config-json CONFIG_JSON] [--skip-firewalld] [--skip-pull]
```

例

```
[root@host01 ~]# cephadm adopt --style=legacy --name prometheus.host02
```

ceph-volume

説明

このコマンドは、特定のホスト上の全デバイスをリスト表示に使用されます。プラグ可能なツールを使用して、**lvm** や物理ディスクなど、さまざまなデバイス技術を持つ Deploys OSD コンテナー内で **ceph-volume** を実行するか、推測可能かつ強固な方法で、OSD の準備、有効化、開始する方法を進めていきます。

構文

```
cephadm ceph-volume inventory/simple/raw/lvm [-h] [--fsid FSID] [--config-json
CONFIG_JSON] [--config CONFIG, -c CONFIG] [--keyring KEYRING, -k KEYRING]
```

例

```
[root@nhost01 ~]# cephadm ceph-volume inventory --fsid f64f341c-655d-11eb-8778-
fa163e914bcc
```

check-host

説明

Ceph クラスターに適したホスト設定を確認します。

構文

```
cephadm check-host [--expect-hostname HOSTNAME]
```

例

```
[root@host01 ~]# cephadm check-host --expect-hostname host02
```

deploy

説明

デーモンをローカルホストにデプロイします。

構文

```
cephadm shell deploy DAEMON_TYPE [-h] [--name DAEMON_NAME] [--fsid FSID] [--config CONFIG, -c CONFIG] [--config-json CONFIG_JSON] [--keyring KEYRING] [--key KEY] [--osd-fsid OSD_FSID] [--skip-firewalld] [--tcp-ports TCP_PORTS] [--reconfig] [--allow-ptrace] [--memory-request MEMORY_REQUEST] [--memory-limit MEMORY_LIMIT] [--meta-json META_JSON]
```

例

```
[root@host01 ~]# cephadm shell deploy mon --fsid f64f341c-655d-11eb-8778-fa163e914bcc
```

enter

説明

実行中のデーモンコンテナー内でインタラクティブシェルを実行します。

構文

```
cephadm enter [-h] [--fsid FSID] --name NAME [command [command …]]
```

例

```
[root@host01 ~]# cephadm enter --name 52c611f2b1d9
```

help

説明

cephadm によってサポートされるすべてのコマンドを表示します。

構文

```
cephadm help
```

例

```
[root@host01 ~]# cephadm help
```

install

説明

パッケージをインストールします。

構文

```
cephadm install PACKAGES
```

例

```
[root@host01 ~]# cephadm install ceph-common ceph-osd
```

inspect-image

説明

ローカルの Ceph コンテナーイメージを検査します。

構文

```
cephadm --image IMAGE_ID inspect-image
```

例

```
[root@host01 ~]# cephadm --image
13ea90216d0be03003d12d7869f72ad9de5cec9e54a27fd308e01e467c0d4a0a inspect-image
```

list-networks

説明

IP ネットワークをリスト表示します。

構文

```
cephadm list-networks
```

例

```
[root@host01 ~]# cephadm list-networks
```

ls

説明

ホストの **cephadm** が認識するデーモンインスタンスをリスト表示します。コマンド実行時間を短縮するには **--no-detail** を使用できます。これにより、デーモンごとの名前、fsid、スタイル、および systemd ユニットの詳細が提供されます。**--legacy-dir** オプションを使用して、デーモンを検索するレガシーベースディレクトリーを指定できます。

構文

```
cephadm ls [--no-detail] [--legacy-dir LEGACY_DIR]
```

例

```
[root@host01 ~]# cephadm ls --no-detail
```

logs

説明

デーモンコンテナーの **journald** ログを出力します。これは **journalctl** コマンドに似ています。

構文

```
cephadm logs [--fsid FSID] --name DAEMON_NAME
cephadm logs [--fsid FSID] --name DAEMON_NAME -- -n NUMBER # Last N lines
cephadm logs [--fsid FSID] --name DAEMON_NAME -- -f # Follow the logs
```

例

```
[root@host01 ~]# cephadm logs --fsid 57bddb48-ee04-11eb-9962-001a4a000672 --name
osd.8
[root@host01 ~]# cephadm logs --fsid 57bddb48-ee04-11eb-9962-001a4a000672 --name
osd.8 -- -n 20
[root@host01 ~]# cephadm logs --fsid 57bddb48-ee04-11eb-9962-001a4a000672 --name
osd.8 -- -f
```

prepare-host

説明

cephadm のホストを準備します。

構文

```
cephadm prepare-host [--expect-hostname HOSTNAME]
```

例

```
[root@host01 ~]# cephadm prepare-host
[root@host01 ~]# cephadm prepare-host --expect-hostname host01
```

pull

説明

Ceph イメージをプルします。

構文

```
cephadm [-h] [--image IMAGE_ID] pull
```

例

```
[root@host01 ~]# cephadm --image
13ea90216d0be03003d12d7869f72ad9de5cec9e54a27fd308e01e467c0d4a0a pull
```

registry-login

説明

認証されたレジストリーの cephadm ログイン情報を提供します。Cephadm はそのレジストリーに呼び出したホストのログ記録を試行します。

構文

```
cephadm registry-login --registry-url [REGISTRY_URL] --registry-username [USERNAME] --registry-password [PASSWORD] [--fsid FSID] [--registry-json JSON_FILE]
```

例

```
[root@host01 ~]# cephadm registry-login --registry-url registry.redhat.io --registry-username myuser1 --registry-password mypassword1
```

また、以下のようにフォーマットされたログイン情報が含まれる JSON レジストリーファイルを使用することもできます。

構文

```
cat REGISTRY_FILE

{
 "url":"REGISTRY_URL",
 "username":"REGISTRY_USERNAME",
 "password":"REGISTRY_PASSWORD"
}
```

例

```
[root@host01 ~]# cat registry_file

{
 "url":"registry.redhat.io",
 "username":"myuser",
 "password":"mypass"
}

[root@host01 ~]# cephadm registry-login -i registry_file
```

rm-daemon

説明

特定のデーモンインスタンスを削除します。ホストで **cephadm rm-daemon** コマンドを直接実行すると、コマンドはデーモンを削除しますが、**cephadm mgr** モジュールは、デーモンがないことを通知して再デプロイします。このコマンドは問題が含まれており、実験的な目的およびデバッグにのみ使用する必要があります。

構文

```
cephadm rm-daemon [--fsid FSID] [--name DAEMON_NAME] [--force ] [--force-delete-data]
```

例

```
[root@host01 ~]# cephadm rm-daemon --fsid f64f341c-655d-11eb-8778-fa163e914bcc --name osd.8
```

rm-cluster

説明

実行先の特定のホストのストレージクラスターからすべてのデーモンを削除します。**rm-daemon** と同様に、この方法でいくつかのデーモンが削除され、Ceph Orchestrator は一時停止されず、これらのデーモンでマネージド外ではないサービスに所属するものがある場合は、**cephadm** オーケストレーターにより、そこに再デプロイされます。

構文

```
cephadm rm-cluster [--fsid FSID] [--force]
```

例

```
[root@host01 ~]# cephadm rm-cluster --fsid f64f341c-655d-11eb-8778-fa163e914bcc
```

重要

クラスターの削除実行の一環としてノードをより適切にクリーンアップするために、**cephadm rm-cluster** コマンドの実行時に **/var/log/ceph** ディレクトリーの下のクラスターログが削除されます。**--keep-logs** が **rm-cluster** コマンドに指定されていない限り、クラスターログは削除されます。

注記

ホストが Cephadm によって管理され、Cephadm Managerモジュールが有効になって実行されている既存のクラスタの一部であるホストで **cephadm rm-cluster** コマンドを実行すると、Cephadm がすぐに新しいデーモンのデプロイを開始し、さらにログが表示される可能性があります。これを回避するには、クラスターをパージする前に cephadm mgr モジュールを無効にします。

```
# ceph mgr module disable cephadm
```

rm-repo

説明

パッケージリポジトリーの設定を削除します。これは主に Red Hat Ceph Storage の非接続インストールに使用されます。

構文

```
cephadm rm-repo [-h]
```

例

```
[root@host01 ~]# cephadm rm-repo
```

run

説明

フォアグラウンドのコンテナーで Ceph デーモンを実行します。

構文

```
cephadm run [--fsid FSID] --name DAEMON_NAME
```

例

```
[root@host01 ~]# cephadm run --fsid f64f341c-655d-11eb-8778-fa163e914bcc --name osd.8
```

shell

説明

推論または指定した Ceph クラスターを使用して、Ceph コマンドにアクセスできるインタラクティブシェルを実行します。**cephadm shell** コマンドを使用してシェルに移動し、シェル内ですべてのオーケストレーターコマンドを実行できます。

構文

```
cephadm shell  [--fsid FSID] [--name DAEMON_NAME, -n DAEMON_NAME] [--config CONFIG, -c CONFIG] [--mount MOUNT, -m MOUNT] [--keyring KEYRING, -k KEYRING] [--env ENV, -e ENV]
```

例

```
[root@host01 ~]# cephadm shell -- ceph orch ls
[root@host01 ~]# cephadm shell
```

unit

説明

この操作でデーモンを起動、停止、再起動、有効化、および無効にします。これは、デーモンの **systemd** ユニットで動作します。

構文

```
cephadm unit [--fsid FSID] --name DAEMON_NAME start/stop/restart/enable/disable
```

例

```
[root@host01 ~]# cephadm unit --fsid f64f341c-655d-11eb-8778-fa163e914bcc --name osd.8 start
```

version

説明

ストレージクラスターのバージョンを提供します。

構文

```
cephadm version
```

例

```
[root@host01 ~]# cephadm version
```