Red Hat OpenStack Platform 16.1

コンテナー化された Red Hat Ceph を持つオーバークラウドのデプロイ

director でコンテナー化された Red Hat Ceph クラスターをデプロイして使用する設定

Last Updated: 2023-11-07

Red Hat OpenStack Platform 16.1 コンテナー化された Red Hat Ceph を持つオーバークラウドのデプロイ

director でコンテナー化された Red Hat Ceph クラスターをデプロイして使用する設定

OpenStack Team
rhos-docs@redhat.com

法律上の通知

概要

本ガイドでは、Red Hat OpenStack Platform director を使用して、コンテナー化された Red Hat Ceph Storage クラスターを持つオーバークラウドを作成する方法について説明します。これには、director を使用して Ceph クラスターをカスタマイズする手順が含まれます。

目次

多様性を受け入れるオープンソースの強化

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、Red Hat CTO である Chris Wright のメッセージ を参照してください。

RED HAT ドキュメントへのフィードバック (英語のみ)

Red Hat ドキュメントに対するご意見をお聞かせください。ドキュメントの改善点があればお知らせください。

ドキュメントへのダイレクトフィードバック (DDF) 機能の使用 (英語版のみ)

特定の文章、段落、またはコードブロックに対して直接コメントを送付するには、DDF の **Add Feedback** 機能を使用してください。なお、この機能は英語版のドキュメントでのみご利用いただけます。

1. **Multi-page HTML** 形式でドキュメントを表示します。
2. ドキュメントの右上隅に **Feedback** ボタンが表示されていることを確認してください。
3. コメントするテキスト部分をハイライト表示します。
4. **Add Feedback** をクリックします。
5. **Add Feedback** フィールドにコメントを入力します。
6. オプション: ドキュメントチームが問題の詳細を確認する際に使用できるメールアドレスを記入してください。
7. **Submit** をクリックします。

第1章 はじめに

Red Hat OpenStack Platform director は、オーバークラウドと呼ばれるクラウド環境を作成します。director を使用して、Red Hat Ceph Storage (director で作成した Ceph Storage クラスターおよび既存の Ceph Storage クラスターの両方) との統合を含む、オーバークラウドの追加機能を設定することができます。

本ガイドでは、既存の Ceph Storage クラスターをオーバークラウドと統合する手順について説明します。この場合、director はストレージ用途に Ceph Storage クラスターを使用するようにオーバークラウドを設定します。クラスターそのものは、オーバークラウド設定とは独立して管理およびスケーリングします。

本ガイドでは、オーバークラウドと共にコンテナー化された Red Hat Ceph Storage クラスターをデプロイする手順について説明します。director は、**ceph-ansible** パッケージで提供される Ansible Playbook を使用して、コンテナー化された Ceph クラスターをデプロイします。director は、クラスターの設定およびスケーリング操作も管理します。

Red Hat OpenStack Platform (RHOSP) のコンテナー化されたサービスに関する詳細は、**director のインストールと使用方法**の CLI ツールを使用した基本的なオーバークラウドの設定 を参照してください。

1.1. CEPH STORAGE の概要

Red Hat Ceph Storage は、優れたパフォーマンス、信頼性、スケーラビリティーを提供するように設計された、分散型のデータオブジェクトストアです。非構造化データに対応しており、クライアントが新しいタイプのオブジェクトインターフェイスと従来のインターフェイスを同時に使用できる分散型のオブジェクトストアは、今後のストレージのあるべき姿です。すべての Ceph デプロイメントの中核となる Ceph Storage クラスターは、複数の種類のデーモンで設定されますが、主要なデーモンは以下の 2 つになります。

Ceph OSD (Object Storage Daemon)

Ceph OSD は、Ceph クライアントの代わりにデータを格納します。また、Ceph OSD は Ceph ノードの CPU とメモリーを使用して、データの複製、リバランス、復旧、監視、レポート作成を実行します。

Ceph Monitor

Ceph monitor は、ストレージクラスターの現在の状態を含む Ceph Storage クラスターのマッピングのマスターコピーを管理します。

Red Hat Ceph Storage に関する詳細は、Red Hat Ceph Storage Architecture Guide を参照してください。

1.2. 要件

本ガイドは、director のインストールと使用方法 ガイドの補足情報を提供します。

オーバークラウドと共にコンテナー化された Ceph Storage クラスターをデプロイする前に、お使いの環境に以下の設定が含まれている必要があります。

- Red Hat OpenStack Platform (RHOSP) director をインストールしたアンダークラウドホスト。詳細は、アンダークラウドへの director のインストール を参照してください。
- Red Hat Ceph Storage に推奨される追加のハードウェア。推奨されるハードウェアの詳細は、Red Hat Ceph Storage Hardware Guide を参照してください。

重要

Ceph Monitor サービスは、オーバークラウドのコントローラーノードにインストールされるので、パフォーマンスの問題を避けるために十分なリソースを提供する必要があります。お使いの環境のコントローラーノードが、最低でも 16 GB のメモリーおよび Ceph monitor データ用にソリッドステートドライブ (SSD) ストレージを使用するようにしてください。中/大容量の Ceph ストレージでは、Ceph monitor データ用に少なくとも 500 GB のストレージを確保してください。クラスターが不安定になった際の levelDB サイズの増大を防ぐためには、この容量が必要です。Ceph Storage クラスターの一般的なサイズの例を以下に示します。

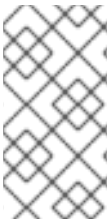
- 小規模: 250 テラバイト
- 中規模: 1 ペタバイト
- 大規模: 2 ペタバイト以上

Red Hat OpenStack Platform director を使用して Ceph Storage ノードを作成する場合は、以下の要件に注意してください。

1.2.1. Ceph Storage ノードの要件

Ceph Storage ノードは、Red Hat OpenStack Platform 環境でオブジェクトストレージを提供するロールを果たします。

Ceph Storage ノードのプロセッサー、メモリー、ネットワークインターフェイスカード (NIC)、およびディスクレイアウトを選択する方法の情報は、**Red Hat Ceph Storage ハードウェアガイド**で Red Hat Ceph Storage におけるハードウェア選択に関する推奨事項 を確認してください。各 Ceph Storage ノードにも、Intelligent Platform Management Interface (IPMI) 機能などのサポート対象の電源管理インターフェイスがサーバーのマザーボードに搭載されている必要があります。



注記

Red Hat OpenStack Platform (RHOSP) director は **ceph-ansible** を使用しますが、Ceph Storage ノードのルートディスクへの OSD インストールには対応しません。したがって、サポートされる Ceph Storage ノードには少なくとも 2 つのディスクが必要になります。

Ceph Storage ノードと RHEL の互換性

- RHOSP 16.1 は RHEL 8.2 でサポートされています。ただし、Ceph Storage ロールにマップされているホストは、最新のメジャー RHEL リリースに更新されます。RHOSP 16.1 以降にアップグレードする前に、Red Hat ナレッジベースの記事 Red Hat Ceph Storage: Supported configurations を確認してください。

配置グループ (PG)

- デプロイメントの規模によらず、動的で効率的なオブジェクトの追跡を容易に実施するために、Ceph Storage では配置グループ (PG) が使用されています。OSD の障害やクラスターのリバランスの際には、Ceph は配置グループおよびその内容を移動または複製することができるので、Ceph Storage クラスターは効率的にリバランスおよび復旧を行うことができます。
- director が作成するデフォルトの配置グループ数が常に最適とは限らないので、実際の要件に応じて正しい配置グループ数を計算することが重要です。配置グループの計算ツールを使用して、正しい配置グループ数を計算することができます。PG の計算ツールを使用するには、

Ceph クラスターに関するその他の属性 (OSD の数など) と共に、サービスごとに予測されるストレージ使用量をパーセンテージで入力します。計算ツールは、プールごとに最適な PG 数を返します。詳細は、Ceph Placement Groups (PGs) per Pool Calculator を参照してください。

- 自動スケーリングは、配置グループを管理するもう 1 つの方法です。自動スケーリング機能では、具体的な配置グループ数ではなく、サービスごとに予想される Ceph Storage 要件をパーセンテージで設定します。Ceph は、クラスターの使用状況に応じて配置グループを自動的にスケーリングします。詳細は、**Red Hat Ceph Storage ストレージストラテジーガイド**の 配置グループの自動スケーリング を参照してください。

プロセッサー

- Intel 64 または AMD64 CPU 拡張機能をサポートする 64 ビット x86 プロセッサー。

ネットワークインターフェイスカード

- 最小 1 枚の 1 Gbps ネットワークインターフェイスカード (NIC)。ただし、Red Hat では実稼働環境の場合には最低でも NIC を 2 枚使用することを推奨します。ボンディングインターフェイス向けやタグ付けされた VLAN トラフィックを委譲する場合は、追加の NIC を使用します。特に大量のトラフィックを処理する Red Hat OpenStack Platform (RHOSP) 環境を構築する場合には、ストレージノードに 10 Gbps インターフェイスを使用します。

電源管理

- 各コントローラーノードには、Intelligent Platform Management Interface (IPMI) 機能などのサポート対象の電源管理インターフェイスがサーバーのマザーボードに搭載されている必要があります。

1.3. 関連情報

/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml 環境ファイルの設定により、director は ceph-ansible プロジェクトから提供される Playbook を使用します。これらの Playbook はアンダークラウドの **/usr/share/ceph-ansible/** にインストールされます。特に以下のファイルには、Playbook により適用されるすべてのデフォルト設定が含まれています。

- **/usr/share/ceph-ansible/group_vars/all.yml.sample**

警告

ceph-ansible は Playbook を使用してコンテナー化された Ceph Storage をデプロイしますが、デプロイメントをカスタマイズするためにこれらのファイルを編集しないでください。その代わりに、heat 環境ファイルを使用して、これらの Playbook により設定されるデフォルトを上書きしてください。**ceph-ansible** Playbook を直接編集すると、デプロイメントは失敗します。

コンテナー化された Ceph Storage 向けに director により適用されるデフォルト設定について詳しく知るには、**/usr/share/openstack-tripleo-heat-templates/deployment/ceph-ansible** の heat テンプレートを参照してください。

注記

これらのテンプレートを理解するには、環境ファイルおよび heat テンプレートが director でどのように機能するかを熟知している必要があります。Heat テンプレートの概要 および 環境ファイル を参照してください。

RHOSP のコンテナー化されたサービスに関する詳細は、**director のインストールと使用方法** の CLI ツールを使用した基本的なオーバークラウドの設定 を参照してください。

第2章 オーバークラウドデプロイメント用の CEPH STORAGE ノードの準備

このシナリオのすべてのノードは、電源管理に IPMI を使用したベアメタルシステムです。director は Red Hat Enterprise Linux 8 イメージを各ノードにコピーするため、これらのノードにはオペレーティングシステムは必要ありません。また、これらのノード上の Ceph Storage サービスはコンテナー化されています。イントロスペクションおよびプロビジョニングのプロセスの間、director はプロビジョニングネットワークを介して各ノードと通信します。すべてのノードは、ネイティブの VLAN を通じてこのネットワークに接続されます。

2.1. CEPH STORAGE ノードのディスクのクリーニング

Ceph Storage OSD およびジャーナルのパーティションには GPT ディスクラベルが必要です。これは、Ceph OSD サービスをインストールする前に Ceph Storage 上の追加のディスクを GPT に変換する必要があることを意味します。director が GPT ラベルをディスクに設定できるようにするには、ディスクからすべてのメタデータを削除する必要があります。

以下の設定をお使いの **/home/stack/undercloud.conf** ファイルに追加すると、director がデフォルトでディスクのメタデータをすべて削除するように設定できます。

```
clean_nodes=true
```

このオプションでは、Bare Metal Provisioning サービスが追加のステップを実行してノードを起動し、ノードが **available** に設定されるたびにディスクのクリーニングを実行します。このプロセスでは、初回のイントロスペクションが終了して、各デプロイメントが開始する前に電源サイクルがもう 1 つ追加されます。Bare Metal Provisioning サービスは **wipefs --force --all** コマンドを使用してクリーニングを実行します。

このオプションを設定したら、**openstack undercloud install** コマンドを実行して、この設定変更を有効にします。

警告

wipefs --force --all コマンドにより、ディスク上の全データおよびメタデータが削除されますが、Secure Erase は実行されません。Secure Erase には非常に長い時間がかかります。

2.2. ノードの登録

ノードインベントリーファイル (**instackenv.json**) を JSON 形式で director にインポートし、director がノードと通信できるようにします。このインベントリーファイルには、director がノードを登録するのに使用できるハードウェアおよび電源管理の情報が含まれています。

```
{
    "nodes":[
        {
            "mac":[
                "b1:b1:b1:b1:b1:b1"
            ],
```

```
    "cpu":"4",
    "memory":"6144",
    "disk":"40",
    "arch":"x86_64",
    "pm_type":"ipmi",
    "pm_user":"admin",
    "pm_password":"p@55w0rd!",
    "pm_addr":"192.0.2.205"
  },
  {
    "mac":[
      "b2:b2:b2:b2:b2:b2"
    ],
    "cpu":"4",
    "memory":"6144",
    "disk":"40",
    "arch":"x86_64",
    "pm_type":"ipmi",
    "pm_user":"admin",
    "pm_password":"p@55w0rd!",
    "pm_addr":"192.0.2.206"
  },
  {
    "mac":[
      "b3:b3:b3:b3:b3:b3"
    ],
    "cpu":"4",
    "memory":"6144",
    "disk":"40",
    "arch":"x86_64",
    "pm_type":"ipmi",
    "pm_user":"admin",
    "pm_password":"p@55w0rd!",
    "pm_addr":"192.0.2.207"
  },
  {
    "mac":[
      "c1:c1:c1:c1:c1:c1"
    ],
    "cpu":"4",
    "memory":"6144",
    "disk":"40",
    "arch":"x86_64",
    "pm_type":"ipmi",
    "pm_user":"admin",
    "pm_password":"p@55w0rd!",
    "pm_addr":"192.0.2.208"
  },
  {
    "mac":[
      "c2:c2:c2:c2:c2:c2"
    ],
    "cpu":"4",
    "memory":"6144",
    "disk":"40",
    "arch":"x86_64",
```

```
        "pm_type":"ipmi",
        "pm_user":"admin",
        "pm_password":"p@55w0rd!",
        "pm_addr":"192.0.2.209"
    },
    {
        "mac":[
            "c3:c3:c3:c3:c3:c3"
        ],
        "cpu":"4",
        "memory":"6144",
        "disk":"40",
        "arch":"x86_64",
        "pm_type":"ipmi",
        "pm_user":"admin",
        "pm_password":"p@55w0rd!",
        "pm_addr":"192.0.2.210"
    },
    {
        "mac":[
            "d1:d1:d1:d1:d1:d1"
        ],
        "cpu":"4",
        "memory":"6144",
        "disk":"40",
        "arch":"x86_64",
        "pm_type":"ipmi",
        "pm_user":"admin",
        "pm_password":"p@55w0rd!",
        "pm_addr":"192.0.2.211"
    },
    {
        "mac":[
            "d2:d2:d2:d2:d2:d2"
        ],
        "cpu":"4",
        "memory":"6144",
        "disk":"40",
        "arch":"x86_64",
        "pm_type":"ipmi",
        "pm_user":"admin",
        "pm_password":"p@55w0rd!",
        "pm_addr":"192.0.2.212"
    },
    {
        "mac":[
            "d3:d3:d3:d3:d3:d3"
        ],
        "cpu":"4",
        "memory":"6144",
        "disk":"40",
        "arch":"x86_64",
        "pm_type":"ipmi",
        "pm_user":"admin",
        "pm_password":"p@55w0rd!",
        "pm_addr":"192.0.2.213"
```

```
    }
  ]
}
```

手順

1. インベントリーファイルを作成したら、そのファイルを stack ユーザーのホームディレクトリーに保存します (**/home/stack/instackenv.json**)。
2. stack ユーザーを初期化し、続いて **instackenv.json** インベントリーファイルを director にインポートします。

```
$ source ~/stackrc
$ openstack overcloud node import ~/instackenv.json
```

openstack overcloud node import コマンドは、インベントリーファイルをインポートし、各ノードを director に登録します。

3. カーネルと ramdisk イメージを各ノードに割り当てます。

```
$ openstack overcloud node configure <node>
```

結果

director でのノードの登録、設定が完了しました。

2.3. CEPH STORAGE のデプロイメント前の検証

オーバークラウドのデプロイメントが失敗しないようにするには、必要なパッケージがサーバーに存在することを確認します。

2.3.1. ceph-ansible パッケージバージョンの確認

アンダークラウドには Ansible ベースの検証が含まれ、これを実行してオーバークラウドをデプロイする前に潜在的な問題を特定することができます。これらの検証は、典型的な問題が発生する前にそれらを特定し、オーバークラウドのデプロイメントの失敗を回避するのに役立ちます。

手順

ceph-ansible パッケージの修正バージョンがインストールされていることを確認してください。

```
$ ansible-playbook -i /usr/bin/tripleo-ansible-inventory /usr/share/ansible/validation-playbooks/ceph-ansible-installed.yaml
```

2.3.2. 事前にプロビジョニングされたノード用のパッケージの確認

Ceph は、特定のパッケージセットを持つオーバークラウドノードにのみサービスを提供することができます。事前にプロビジョニングされたノードを使用する場合には、これらのパッケージが存在することを確認することができます。

事前にプロビジョニングされたノードの詳細は、事前にプロビジョニングされたノードを使用した基本的なオーバークラウドの設定 を参照してください。

手順

サーバーに必要なパッケージが含まれていることを確認します。

```
ansible-playbook -i /usr/bin/tripleo-ansible-inventory /usr/share/ansible/validation-playbooks/ceph-dependencies-installed.yaml
```

2.4. 手動によるプロファイルへのノードのタグ付け

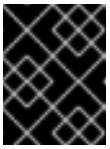
各ノードの登録後、ハードウェアを検査して、ノードを特定のプロファイルにタグ付けする必要があります。プロファイルタグを使用してノードをフレーバーに照合してから、フレーバーをデプロイメントロールに割り当てます。

手順

1. ハードウェアのイントロスペクションをトリガーして、各ノードのハードウェア属性を取得します。

```
$ openstack overcloud node introspect --all-manageable --provide
```

   - **--all-manageable** オプションを使用して、管理状態にあるノードのみをイントロスペクションします。ここでは、すべてのノードが管理状態にあります。
   - **--provide** オプションは、イントロスペクション後に全ノードを **active** の状態にリセットします。



重要

このプロセスが正常に完了したことを確認します。ベアメタルノードの場合には、通常 15 分ほどかかります。

2. ノードリストを取得して UUID を把握します。

```
$ openstack baremetal node list
```

3. 各ノードの **properties/capabilities** パラメーターに profile オプションを追加して、ノードを特定のプロファイルに手動でタグ付けします。**profile** オプションを追加すると、適切なプロファイルにノードをタグ付けします。

注記

手動でのタグ付けの代わりに、Automated Health Check (AHC) ツールを使用し、ベンチマークデータに基づいて、多数のノードに自動でタグ付けします。

たとえば、標準的なデプロイメントには、**control**、**compute**、および **ceph-storage** の 3 つのプロファイルが含まれます。以下のコマンドを実行して、3 つのノードを各プロファイルにタグ付けします。

```
$ openstack baremetal node set --property capabilities='profile:control,boot_option:local' 1a4e30da-b6dc-499d-ba87-0bd8a3819bc0
$ openstack baremetal node set --property capabilities='profile:control,boot_option:local' 6faba1a9-e2d8-4b7c-95a2-c7fbdc12129a
$ openstack baremetal node set --property capabilities='profile:control,boot_option:local' 6faba1a9-e2d8-4b7c-95a2-c7fbdc12129a
```

```
$ openstack baremetal node set --property capabilities='profile:compute,boot_option:local'
484587b2-b3b3-40d5-925b-a26a2fa3036f
$ openstack baremetal node set --property capabilities='profile:compute,boot_option:local'
d010460b-38f2-4800-9cc4-d69f0d067efe
$ openstack baremetal node set --property capabilities='profile:compute,boot_option:local'
d930e613-3e14-44b9-8240-4f3559801ea6
$ openstack baremetal node set --property capabilities='profile:ceph-
storage,boot_option:local' 484587b2-b3b3-40d5-925b-a26a2fa3036f
$ openstack baremetal node set --property capabilities='profile:ceph-
storage,boot_option:local' d010460b-38f2-4800-9cc4-d69f0d067efe
$ openstack baremetal node set --property capabilities='profile:ceph-
storage,boot_option:local' d930e613-3e14-44b9-8240-4f3559801ea6
```

ヒント

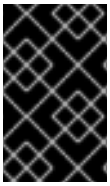
Ceph MON サービスおよび Ceph MDS サービス用のノードのタグ付けに使用できる新しいカスタムプロファイルを設定することも可能です。詳しくは、3章*専用ノード上での Ceph サービスのデプロイ*を参照してください。

2.5. マルチディスククラスターのルートディスクの定義

ノードで複数のディスクが使用されている場合には、director はプロビジョニング時にルートディスクを特定する必要があります。たとえば、ほとんどの Ceph Storage ノードでは、複数のディスクが使用されます。デフォルトのプロビジョニングプロセスでは、director はルートディスクにオーバークラウドイメージを書き込みます。

以下の属性を定義すると、director がルートディスクを特定するのに役立ちます。

- **model** (文字列): デバイスの ID
- **vendor** (文字列): デバイスのベンダー
- **serial** (文字列): ディスクのシリアル番号
- **hctl** (文字列): SCSI のホスト、チャンネル、ターゲット、Lun
- **size** (整数): デバイスのサイズ (GB 単位)
- **wwn** (文字列): 一意のストレージ ID
- **wwn_with_extension** (文字列): ベンダー拡張子を追加した一意のストレージ ID
- **wwn_vendor_extension** (文字列): 一意のベンダーストレージ ID
- **rotational** (ブール値): 回転式デバイス (HDD) には true、そうでない場合 (SSD) には false
- **name** (文字列): デバイス名 (例: /dev/sdb1)



重要

name プロパティーは、永続デバイス名が付いたデバイスにのみ使用します。他のデバイスのルートディスクを設定する際に、**name** を使用しないでください。この値は、ノードのブート時に変更される可能性があります。

シリアル番号を使用してルートデバイスを指定することができます。

手順

1. 各ノードのハードウェアイントロスペクションからのディスク情報を確認します。以下のコマンドを実行して、ノードのディスク情報を表示します。

```
(undercloud)$ openstack baremetal introspection data save 1a4e30da-b6dc-499d-ba87-0bd8a3819bc0 | jq ".inventory.disks"
```

たとえば、1 つのノードのデータで 3 つのディスクが表示される場合があります。

```
[
 {
  "size": 299439751168,
  "rotational": true,
  "vendor": "DELL",
  "name": "/dev/sda",
  "wwn_vendor_extension": "0x1ea4dcc412a9632b",
  "wwn_with_extension": "0x61866da04f3807001ea4dcc412a9632b",
  "model": "PERC H330 Mini",
  "wwn": "0x61866da04f380700",
  "serial": "61866da04f3807001ea4dcc412a9632b"
 }
 {
  "size": 299439751168,
  "rotational": true,
  "vendor": "DELL",
  "name": "/dev/sdb",
  "wwn_vendor_extension": "0x1ea4e13c12e36ad6",
  "wwn_with_extension": "0x61866da04f380d001ea4e13c12e36ad6",
  "model": "PERC H330 Mini",
  "wwn": "0x61866da04f380d00",
  "serial": "61866da04f380d001ea4e13c12e36ad6"
 }
 {
  "size": 299439751168,
  "rotational": true,
  "vendor": "DELL",
  "name": "/dev/sdc",
  "wwn_vendor_extension": "0x1ea4e31e121cfb45",
  "wwn_with_extension": "0x61866da04f37fc001ea4e31e121cfb45",
  "model": "PERC H330 Mini",
  "wwn": "0x61866da04f37fc00",
  "serial": "61866da04f37fc001ea4e31e121cfb45"
 }
]
```

2. **openstack baremetal node set --property root_device=** を入力して、ノードのルートディスクを設定します。ルートディスクを定義するのに最も適切なハードウェア属性値を指定します。

```
(undercloud)$ openstack baremetal node set --property root_device='{"serial":"<serial_number>"}' <node-uuid>
```

たとえば、ルートデバイスをシリアル番号が **61866da04f380d001ea4e13c12e36ad6** の disk 2 に設定するには、以下のコマンドを実行します。

```
(undercloud)$ openstack baremetal node set --property root_device='{"serial":
"61866da04f380d001ea4e13c12e36ad6"}' 1a4e30da-b6dc-499d-ba87-0bd8a3819bc0
```

注記

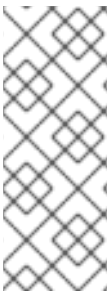
各ノードの BIOS を設定して、選択したルートディスクからの起動を含めるようにします。最初にネットワークからのブートを試み、次にルートディスクからのブートを試みるように、ブート順序を設定します。

director は、ルートディスクとして使用する特定のディスクを把握します。**openstack overcloud deploy** コマンドを実行すると、director はオーバークラウドをプロビジョニングし、ルートディスクにオーバークラウドのイメージを書き込みます。

2.6. OVERCLOUD-MINIMAL イメージの使用による RED HAT サブスクリプションエンタイトルメントの使用回避

デフォルトでは、プロビジョニングプロセス中 director はルートディスクに QCOW2 **overcloud-full** イメージを書き込みます。**overcloud-full** イメージには、有効な Red Hat サブスクリプションが使用されます。ただし、**overcloud-minimal** イメージを使用して、たとえばベア OS をプロビジョニングすることもできます。この場合、他の OpenStack サービスは使用されないので、サブスクリプションエンタイトルメントは消費されません。

この典型的なユースケースは、Ceph デーモンのみを持つノードをプロビジョニングする場合です。この場合や類似のユースケースでは、**overcloud-minimal** イメージのオプションを使用して、有償の Red Hat サブスクリプションが限度に達するのを避けることができます。**overcloud-minimal** イメージの取得方法についての情報は、オーバークラウドノードのイメージの取得 を参照してください。



注記

Red Hat OpenStack Platform (RHOSP) のサブスクリプションには Open vSwitch (OVS) が含まれますが、**overcloud-minimal** イメージを使用する場合には、OVS などのコアサービスは利用できません。Ceph Storage ノードをデプロイするのに OVS は必要ありません。**ovs_bond** を使用してボンディングを定義する代わりに、**linux_bond** を使用します。**linux_bond** の詳細は、オーバークラウドの高度なカスタマイズの Linux ボンディングのオプション を参照してください。

手順

1. **overcloud-minimal** イメージを使用するように director を設定するには、以下のイメージ定義を含む環境ファイルを作成します。

```
parameter_defaults:
  <roleName>Image: overcloud-minimal
```

2. **<roleName>** をロール名に置き換え、ロール名に **Image** を追加します。Ceph Storage ノードの **overcloud-minimal** イメージの例を以下に示します。

```
parameter_defaults:
  CephStorageImage: overcloud-minimal
```

3. **roles_data.yaml** ロール定義ファイルで、**rhsm_enforce** パラメーターを **False** に設定します。

```
rhsm_enforce: False
```

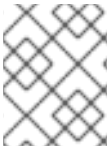
4. この環境ファイルを **openstack overcloud deploy** コマンドに渡します。

注記

overcloud-minimal イメージでは、標準の Linux ブリッジしかサポートされません。OVS は Red Hat OpenStack Platform のサブスクリプションエンタイトルメントが必要な OpenStack サービスなので、このイメージでは OVS はサポートされません。

第3章 専用ノード上での CEPH サービスのデプロイ

デフォルトでは、director は Ceph MON サービスおよび Ceph MDS サービスをコントローラーノードにデプロイします。これは、小規模なデプロイメントに適しています。しかし、大規模なデプロイメントの場合には、Ceph クラスターのパフォーマンスを向上させるために、Ceph MON サービスおよび Ceph MDS サービスを専用のノードにデプロイすることを推奨します。専用ノードで分離するサービス用のカスタムロールを作成します。



注記

カスタムロールについての詳細は、オーバークラウドの高度なカスタマイズ ガイドの 新規ロールの作成 を参照してください。

director は、全オーバークラウドロールのデフォルトのリファレンスとして以下のファイルを使用します。

- **/usr/share/openstack-tripleo-heat-templates/roles_data.yaml**

3.1. カスタムロールファイルの作成

カスタムロールファイルを作成するには、以下の手順を実施します。

手順

1. カスタムロールを追加できるように、**/home/stack/templates/** の **roles_data.yaml** ファイルのコピーを作成します。

```
$ cp /usr/share/openstack-tripleo-heat-templates/roles_data.yaml
/home/stack/templates/roles_data_custom.yaml
```

2. **openstack overcloud deploy** コマンドに新しいカスタムロールファイルを追加します。

3.2. CEPH MON サービス向けのカスタムロールとフレーバーの作成

Ceph MON ロール向けのカスタムロール **CephMon** およびフレーバー **ceph-mon** を作成するには、以下の手順を実施します。デフォルトのロールのデータファイルは、すでにコピー済みのはずです (詳細は*3章 専用ノード上での Ceph サービスのデプロイ*を参照)。

手順

1. **/home/stack/templates/roles_data_custom.yaml** ファイルを開きます。
2. Ceph MON サービスのサービスエントリー **OS::TripleO::Services::CephMon** を Controller ロールから削除します。
3. **OS::TripleO::Services::CephClient** サービスを Controller ロールに追加します。

```
[...]
- name: Controller # the 'primary' role goes first
  CountDefault: 1
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephMds
```

```
    - OS::TripleO::Services::CephClient
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::CephRbdMirror
    - OS::TripleO::Services::CephRgw
    - OS::TripleO::Services::CinderApi
[...]
```

4. **roles_data_custom.yaml** ファイルの末尾に、Ceph MON サービスおよびその他すべての必要なノードサービスを含むカスタムの **CephMon** ロールを追加します。

```
- name: CephMon
  ServicesDefault:
    # Common Services
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CertmongerUser
    - OS::TripleO::Services::Collectd
    - OS::TripleO::Services::Docker
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::ContainersLogrotateCrond
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::Tuned
    # Role-Specific Services
    - OS::TripleO::Services::CephMon
```

5. **openstack flavor create** コマンドを入力し、**CephMon** ロール用に **ceph-mon** という名前の新規フレーバーを定義します。

```
$ openstack flavor create --id auto --ram 6144 --disk 40 --vcpus 4 ceph-mon
```

注記

このコマンドの詳細については、**openstack flavor create --help** と入力します。

6. このフレーバーを新規プロファイルにマッピングします。このプロファイルも **ceph-mon** という名前です。

```
$ openstack flavor set --property "cpu_arch"="x86_64" --property
"capabilities:boot_option"="local" --property "capabilities:profile"="ceph-mon" ceph-mon
```

注記

このコマンドの詳細については、**openstack flavor set --help** と入力します。

7. ノードを新しい **ceph-mon** プロファイルにタグ付けします。

```
$ openstack baremetal node set --property capabilities='profile:ceph-mon,boot_option:local'
UUID
```

8. **ceph-mon** フレーバーを CephMon ロールに関連付けるには、以下の設定を **node-info.yaml** ファイルに追加します。

```
parameter_defaults:
  OvercloudCephMonFlavor: CephMon
  CephMonCount: 3
```

ノードのタグ付けに関する詳細は、「手動によるプロファイルへのノードのタグ付け」を参照してください。カスタムロールプロファイルの詳細は、プロファイルへのノードのタグ付け を参照してください。

3.3. CEPH MDS サービス向けのカスタムロールとフレーバーの作成

Ceph MDS ロール用のカスタムロール **CephMDS** およびフレーバー **ceph-mds** を作成するには、以下の手順を実施します。デフォルトのロールのデータファイルは、すでにコピー済みのはずです (詳細は*3章 専用ノード上での Ceph サービスのデプロイ*を参照)。

手順

1. **/home/stack/templates/roles_data_custom.yaml** ファイルを開きます。
2. Ceph MDS サービスのサービスエントリー **OS::TripleO::Services::CephMds** を Controller ロールから削除します。

```
[...]
- name: Controller # the 'primary' role goes first
  CountDefault: 1
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    # - OS::TripleO::Services::CephMds (1)
    - OS::TripleO::Services::CephMon
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::CephRbdMirror
    - OS::TripleO::Services::CephRgw
    - OS::TripleO::Services::CinderApi
[...]
```

(1) この行をコメントアウトします。次のステップで、このサービスを新しいカスタムロールに追加します。

3. **roles_data_custom.yaml** ファイルの末尾に、Ceph MDS サービスおよびその他すべての必要なノードサービスを含むカスタムの **CephMDS** ロールを追加します。

```
- name: CephMDS
  ServicesDefault:
    # Common Services
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CertmongerUser
    - OS::TripleO::Services::Collectd
```

```
    - OS::TripleO::Services::Docker
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::ContainersLogrotateCrond
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::Tuned
    # Role-Specific Services
    - OS::TripleO::Services::CephMds
    - OS::TripleO::Services::CephClient 1
```

1 Ceph MDS サービスには、Ceph MON サービスまたは Ceph Client サービスのいずれかで設定できる管理キーリングが必要です。Ceph MON サービスがない専用のノードに Ceph MDS をデプロイする場合には、新しい **CephMDS** ロールに Ceph クライアントサービスも追加する必要があります。

4. **openstack flavor create** コマンドを入力し、このロール用に **ceph-mds** という名前の新規フレーバーを定義します。

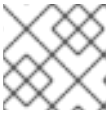
```
$ openstack flavor create --id auto --ram 6144 --disk 40 --vcpus 4 ceph-mds
```

注記

このコマンドの詳細については、**openstack flavor create --help** と入力します。

5. 新規の **ceph-mds** フレーバーを新規プロファイルにマッピングします。このプロファイルも、**ceph-mds** という名前になります。

```
$ openstack flavor set --property "cpu_arch"="x86_64" --property
"capabilities:boot_option"="local" --property "capabilities:profile"="ceph-mds" ceph-mds
```



注記

このコマンドの詳細については、**openstack flavor set --help** と入力します。

6. ノードを新しい **ceph-mds** プロファイルにタグ付けします。

```
$ openstack baremetal node set --property capabilities='profile:ceph-mds,boot_option:local'
UUID
```

ノードのタグ付けに関する詳細は、「手動によるプロファイルへのノードのタグ付け」を参照してください。カスタムロールプロファイルの詳細は、プロファイルへのノードのタグ付け を参照してください。

第4章 ストレージサービスのカスタマイズ

director の提供する heat テンプレートコレクションには、基本的な Ceph Storage 設定を有効にするために必要なテンプレートおよび環境ファイルがすでに含まれています。

director は、**/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml** 環境ファイルを使用して Ceph クラスターを作成し、デプロイ中にこれをオーバークラウドと統合します。このクラスターは、コンテナー化された Ceph Storage ノードを特色とします。OpenStack のコンテナー化されたサービスに関する詳細は、**director のインストールと使用方法** の CLI ツールを使用した基本的なオーバークラウドの設定 を参照してください。

Red Hat OpenStack director により、基本的なデフォルト設定もデプロイされた Ceph クラスターに適用されます。また、カスタム環境ファイルに追加された設定はすべて定義する必要があります。

手順

1. **/home/stack/templates/** に **storage-config.yaml** ファイルを作成します。この例では、**~/templates/storage-config.yaml** ファイルには、お使いの環境用のオーバークラウド関連のほとんどのカスタム設定が含まれています。カスタム環境ファイルに追加するパラメーターは、**/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml** ファイルの対応するデフォルト設定を上書きします。

2. **~/templates/storage-config.yaml** に **parameter_defaults** セクションを追加します。このセクションには、お使いのオーバークラウド用のカスタム設定が含まれます。たとえば、ネットワークサービス (**neutron**) のネットワーク種別として **vxlan** を設定するには、以下のスニペットをカスタム環境ファイルに追加します。

```
parameter_defaults:
  NeutronNetworkType: vxlan
```

3. 必要に応じて、実際の要件に応じて **parameter_defaults** 配下に以下のオプションを設定します。

オプション	説明	デフォルト値
CinderEnableIscsiBackend	iSCSI バックエンドを有効にします。	false
CinderEnableRbdBackend	Ceph Storage バックエンドを有効にします。	true
CinderBackupBackend	Ceph または swift をボリュームのバックアップのバックエンドとして設定します。詳細は、「バックアップサービスで Ceph を使用する設定」を参照してください。	ceph
NovaEnableRbdBackend	Nova の一時ストレージ用に Ceph Storage を有効にします。	true

オプション	説明	デフォルト値
GlanceBackend	Image サービスが使用するバックエンドを定義します。**rbd** (Ceph)、**swift**、または **file** を設定可能です。	rbd
GnocchiBackend	Telemetry サービスが使用するバックエンドを定義します。**rbd** (Ceph)、**swift**、または **file** を設定可能です。	rbd

注記

デフォルト設定を使用する場合には、**~/templates/storage-config.yaml** からオプションを省くことができます。

カスタム環境ファイルの内容は、以下のセクションで適用する設定により異なります。完全な例は付録 A *環境ファイルのサンプル: Ceph Storage クラスターの作成*を参照してください。

以下のサブセクションでは、director が適用する一般的なデフォルトのストレージサービスの設定を上書きする方法について説明します。

4.1. CEPH METADATA SERVER の有効化

Ceph Metadata Server (MDS) は **ceph-mds** デーモンを実行し、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。CephFS は、NFS を介して使用できます。NFS を介した CephFS の使用に関する詳細は、File System Guide および CephFS via NFS Back End Guide for the Shared File System Service を参照してください。

注記

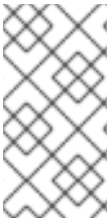
Red Hat は、Shared File System サービスの NFS バックエンドを介した CephFS のみを使用した Ceph MDS のデプロイをサポートします。

手順

Ceph Metadata Server を有効にするには、オーバークラウド作成時に以下の環境ファイルを呼び出します。

- **/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-mds.yaml**

詳細は、「オーバークラウドデプロイメントの開始」 を参照してください。Ceph Metadata Server に関する詳細は、Configuring Metadata Server Daemons を参照してください。



注記

デフォルトでは、Ceph Metadata Server はコントローラーノードにデプロイされます。Ceph Metadata Server を専用のノードにデプロイすることができます。詳細は、「Ceph MDS サービス向けのカスタムロールとフレーバーの作成」を参照してください。

4.2. CEPH OBJECT GATEWAY の有効化

Ceph Object Gateway (RGW) は、Ceph Storage クラスター内のオブジェクトストレージケイパビリティーへのインターフェイスを使用してアプリケーションを提供します。RGW をデプロイする際には、デフォルトの Object Storage サービス (**swift**) を Ceph に置き換えることができます。詳細は、Object Gateway Configuration and Administration Guide を参照してください。

手順

デプロイメントで RGW を有効にするには、オーバークラウド作成時に以下の環境ファイルを呼び出します。

- **/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-rgw.yaml**

詳細は、「オーバークラウドデプロイメントの開始」を参照してください。

デフォルトでは、Ceph Storage は OSD ごとに 250 の配置グループを許可します。RGW を有効にすると、Ceph Storage は RGW が必要とする追加プールを 6 つ作成します。新しいプールは以下の通りです。

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

注記

デプロイメントでは、**default** はプールが属するゾーンの名前に置き換えられます。

したがって、RGW を有効にする際には、新しいプールに対応するように、**CephPoolDefaultPgNum** パラメーターを使用してデフォルトの **pg_num** を設定します。Ceph プールの配置グループ数を計算する方法の詳細は、「異なる Ceph プールへのカスタムの属性の割り当て」を参照してください。

デフォルトの Object Storage サービスは、Ceph Object Gateway に直接置き換えられます。したがって、通常 **swift** を使用するその他すべてのサービスは、swift の代わりに Ceph Object Gateway をシームレスに使用することができます (追加設定は必要ありません)。詳細は、Block Storage Backup Guide を参照してください。

4.3. 外部の CEPH OBJECT GATEWAY を使用するための CEPH OBJECT STORE の設定

Red Hat OpenStack Platform (RHOSP) director は、外部の Ceph Object Gateway (RGW) を Object Store サービスとして設定することをサポートしています。外部 RGW サービスで認証するには、Identity サービス (keystone) のユーザーとそのロールを確認するように RGW を設定する必要があります。

外部 Ceph Object Gateway の設定方法に関する詳細は、**Ceph Object Gateway ガイドでの Keystone の使用**の Keystone 認証を使用するように Ceph Object Gateway を設定 を参照してください。

手順

1. カスタム環境ファイル (**swift-external-params.yaml** 等) に以下の **parameter_defaults** を追加し、実際のデプロイメントに合わせて値を調整します。

```
parameter_defaults:
  ExternalSwiftPublicUrl: 'http://<Public RGW endpoint or
loadbalancer>:8080/swift/v1/AUTH_%(project_id)s'
  ExternalSwiftInternalUrl: 'http://<Internal RGW endpoint>:8080/swift/v1/AUTH_%
(project_id)s'
  ExternalSwiftAdminUrl: 'http://<Admin RGW endpoint>:8080/swift/v1/AUTH_%(project_id)s'
  ExternalSwiftUserTenant: 'service'
  SwiftPassword: 'choose_a_random_password'
```

注記

サンプルコードスニペットには、お使いの環境で使用する値とは異なるパラメーター値が含まれる場合があります。

- リモート RGW インスタンスがリッスンするデフォルトのポートは **8080** です。外部 RGW の設定方法によっては、ポートが異なる場合があります。
- オーバークラウドで作成した **swift** ユーザーは、**SwiftPassword** パラメーターで定義したパスワードを使用します。**rgw_keystone_admin_password** を使用し、Identity サービスに対する認証に同じパスワードを使用するように外部 RGW インスタンスを設定する必要があります。

2. Ceph 設定ファイルに以下のコードを追加して、Identity サービスを使用するように RGW を設定します。変数の値を実際の環境に応じて置き換えます。

```
  rgw_keystone_api_version = 3
  rgw_keystone_url = http://<public Keystone endpoint>:5000/
  rgw_keystone_accepted_roles = member, Member, admin
  rgw_keystone_accepted_admin_roles = ResellerAdmin, swiftoperator
  rgw_keystone_admin_domain = default
  rgw_keystone_admin_project = service
  rgw_keystone_admin_user = swift
  rgw_keystone_admin_password =
<password_as_defined_in_the_environment_parameters>
  rgw_keystone_implicit_tenants = true
  rgw_keystone_revocation_interval = 0
  rgw_s3_auth_use_keystone = true
  rgw_swift_versioning_enabled = true
  rgw_swift_account_in_url = true
```

注記

デフォルトでは、director は Identity サービスに以下のロールとユーザーを作成します。

- rgw_keystone_accepted_admin_roles: ResellerAdmin, swiftoperator
- rgw_keystone_admin_domain: default
- rgw_keystone_admin_project: service
- rgw_keystone_admin_user: swift

3. デプロイメントに該当するその他の環境ファイルと共に、追加の環境ファイルを指定して、オーバークラウドをデプロイします。

```
openstack overcloud deploy --templates \
-e <your_environment_files>
-e /usr/share/openstack-tripleo-heat-templates/environments/swift-external.yaml
-e swift-external-params.yaml
```

検証

1. アンダークラウドに **stack** ユーザーとしてログインします。
2. source コマンドで **overcloudrc** ファイルを読み込みます。

```
$ source ~/stackrc
```

3. エンドポイントが Identity サービス (keystone) に存在することを確認します。

```
$ openstack endpoint list --service object-store

+---------+-----------+-------+-------+---------+-----------+---------------+
| ID | Region    | Service Name | Service Type | Enabled | Interface | URL |
+---------+-----------+-------+-------+---------+-----------+---------------+
| 233b7ea32aaf40c1ad782c696128aa0e | regionOne | swift | object-store | True    | admin     |
http://192.168.24.3:8080/v1/AUTH_%(project_id)s |
| 4ccde35ac76444d7bb82c5816a97abd8 | regionOne | swift | object-store | True    | public    |
https://192.168.24.2:13808/v1/AUTH_%(project_id)s |
| b4ff283f445348639864f560aa2b2b41 | regionOne | swift | object-store | True    | internal  |
http://192.168.24.3:8080/v1/AUTH_%(project_id)s |
+---------+-----------+-------+-------+---------+-----------+---------------+
```

4. テストコンテナーを作成します。

```
$ openstack container create <testcontainer>
+---------------+--------------+----------------------------------+
| account | container | x-trans-id |
+---------------+--------------+----------------------------------+
| AUTH_2852da3cf2fc490081114c434d1fc157 | testcontainer | tx6f5253e710a2449b8ef7e-
005f2d29e8 |
+---------------+--------------+----------------------------------+
```

5. 設定ファイルを作成し、データをコンテナーにアップロードできることを確認します。

```
$ openstack object create testcontainer undercloud.conf
+----------------+--------------+----------------------------------+
| object         | container    | etag                             |
+----------------+--------------+----------------------------------+
| undercloud.conf | testcontainer | 09fcffe126cac1dbac7b89b8fd7a3e4b |
+----------------+--------------+----------------------------------+
```

6. テストコンテナーを削除します。

```
$ openstack container delete -r <testcontainer>
```

4.4. バックアップサービスで CEPH を使用する設定

Block Storage Backup サービス (**cinder-backup**) は、デフォルトでは無効になっています。Block Storage Backup サービスを有効にするには、以下の手順を実行します。

手順

オーバークラウドの作成時に、以下の環境ファイルを呼び出します。

- **/usr/share/openstack-tripleo-heat-templates/environments/cinder-backup.yaml**

4.5. CEPH ノード向けの複数のボンディングされたインターフェイスの設定

ボンディングされたインターフェイスを使用して、複数の NIC を組み合わせ、ネットワーク接続に冗長性を追加します。Ceph ノードに十分な NIC がある場合には、各ノードに複数のボンディングされたインターフェイスを作成して冗長化機能を拡張することができます。

これにより、ノードが必要とする各ネットワーク接続にボンディングされたインターフェイスの使用が可能となります。これにより、冗長性と各ネットワーク専用の接続の両方が提供されます。

ボンディングされたインターフェイスを最も簡単に実装するには、2 つのボンディングを使用して、Ceph ノードが使用する各ストレージネットワークに 1 つずつボンディングを設定します。Ceph ノードが使用するストレージネットワークは以下のとおりです。

フロントエンドストレージネットワーク (StorageNet)

Ceph クライアントはこのネットワークを使用して、対応する Ceph クラスターと対話します。

バックエンドストレージネットワーク (StorageMgmtNet)

Ceph クラスターはこのネットワークを使用して、クラスターの配置グループポリシーに従ってデータのバランスを取ります。詳細は、**Red Hat Ceph Storage Architecture Guide** の Placement Groups (PGs) を参照してください。

複数のボンディングされたインターフェイスを設定するには、新しいネットワークインターフェイステンプレートを作成する必要があります。これは、director が複数のボンディングされた NIC をデプロイするために使用できるサンプルテンプレートを提供しないからです。ただし、director は単一のボンディングされたインターフェイスをデプロイするテンプレートは提供します。このテンプレートは **/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml** です。このテンプレートでは、追加の NIC 用に追加のボンディングされたインターフェイスを定義することができます。

注記

カスタムインターフェイステンプレートの作成に関する詳細は、**オーバークラウドの高度なカスタマイズ**ガイドの カスタムネットワークインターフェイステンプレート を参照してください。

以下のスニペットには、**/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml** ファイルで定義されている単一のボンディングされたインターフェイス用のデフォルトの定義が記載されています。

```
type: ovs_bridge // 1
name: br-bond
members:
  -
    type: ovs_bond // 2
    name: bond1 // 3
    ovs_options: {get_param: BondInterfaceOvsOptions} 4
    members: // 5
      -
        type: interface
        name: nic2
        primary: true
      -
        type: interface
        name: nic3
  -
    type: vlan // 6
    device: bond1 // 7
    vlan_id: {get_param: StorageNetworkVlanID}
    addresses:
      -
        ip_netmask: {get_param: StorageIpSubnet}
  -
    type: vlan
    device: bond1
    vlan_id: {get_param: StorageMgmtNetworkVlanID}
    addresses:
      -
        ip_netmask: {get_param: StorageMgmtIpSubnet}
```

1. **br-bond** という名前の単一のブリッジは、このテンプレートで定義されているボンディングをメンバーとします。この行は、ブリッジの種別 (OVS) を定義しています。

2. **br-bond** ブリッジの最初のメンバーは、ボンディングされたインターフェイス自体で、**bond1** という名前が付いています。この行は、**bond1** のボンディングの種別を定義しており、これも OVS に指定されています。

3. デフォルトのボンディングの名前は **bond1** です。

4. **ovs_options** のエントリーは、director が特定のボンディングモジュールディレクティブのセットを使用するように指示します。これらのディレクティブは、**BondInterfaceOvsOptions** に渡されます。これもこのファイルで設定できます。ボンディングモジュールディレクティブの設定に関する詳細は、「ボンディングモジュールのディレクティブの設定」を参照してください。

5 ボンディングの **members** セクションは、**bond1** でボンディングされるネットワークインターフェイスを定義します。この例では、ボンディングされるインターフェイスは **nic2** (プライマリー

6 **br-bond** ブリッジには、他にも 2 つのメンバーが含まれています。これは、フロントエンドストレージネットワーク (**StorageNetwork**) とバックエンドストレージネットワーク (**StorageMgmtNetwork**) の両方の VLAN です。

7 **device** パラメーターは、VLAN が使用しなければならないデバイスを定義します。この例では、両方の VLAN がボンディングされたインターフェイス **bond1** を使用します。

NIC を少なくとも 2 つ追加すると、ブリッジとボンディングされたインターフェイスをもう 1 つ定義できます。次に、VLAN の 1 つを新しいボンディングされたインターフェイスに移動できます。これにより、両方のストレージネットワーク接続のスループットと信頼性が向上します。

/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml ファイルをこの目的でカスタマイズする場合には、Red Hat では、デフォルトの OVS (**type: ovs_bond**) の代わりに Linux ボンディング (**type: linux_bond**) を使用することを推奨しています。このボンディングの種別は、エンタープライズレベルの実稼働デプロイメントにより適しています。

以下の編集済みスニペットは、**bond2** という名前の新しい Linux ボンディングをメンバーとする追加の OVS ブリッジ (**br-bond2**) を定義します。**bond2** インターフェイスは、2 つの追加の NIC (**nic4** と **nic5**) を使用し、バックエンドストレージネットワークトラフィック専用に使用されます。

```
  type: ovs_bridge
  name: br-bond
  members:
    -
      type: linux_bond
      name: bond1
      bonding_options: {get_param: BondInterfaceOvsOptions} // 1
      members:
        -
          type: interface
          name: nic2
          primary: true
        -
          type: interface
          name: nic3
    -
      type: vlan
      device: bond1
      vlan_id: {get_param: StorageNetworkVlanID}
      addresses:
        -
          ip_netmask: {get_param: StorageIpSubnet}
-
  type: ovs_bridge
  name: br-bond2
  members:
    -
      type: linux_bond
      name: bond2
      bonding_options: {get_param: BondInterfaceOvsOptions}
      members:
        -
```

```
        type: interface
        name: nic4
        primary: true
      -
        type: interface
        name: nic5
  -
    type: vlan
    device: bond1
    vlan_id: {get_param: StorageMgmtNetworkVlanID}
    addresses:
      -
        ip_netmask: {get_param: StorageMgmtIpSubnet}
```

1 **bond1** および **bond2** は両方とも (OVS ではなく) Linux ボンディングであるため、**ovs_options** の代わりに **bonding_options** を使用してボンディングディレクティブを設定します。詳細は、「ボンディングモジュールのディレクティブの設定」 を参照してください。

このカスタマイズされたテンプレートの完全な内容は、付録B *カスタムインターフェイステンプレートの例: 複数のボンディングされたインターフェイス*を参照してください。

4.5.1. ボンディングモジュールのディレクティブの設定

ボンディングされたインターフェイスを追加および設定したら、**BondInterfaceOvsOptions** パラメーターを使用して各ボンディングされたインターフェイスが使用するディレクティブを設定します。この情報は、**/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml** ファイルの **parameters:** セクションにあります。以下のスニペットには、このパラメーターのデフォルトの定義 (空欄) を示しています。

```
BondInterfaceOvsOptions:
    default: ''
    description: The ovs_options string for the bond interface. Set
                 things like lacp=active and/or bond_mode=balance-slb
                 using this option.
    type: string
```

default: の行に必要なオプションを定義します。たとえば、802.3ad (モード 4) と LACP レート 1 (fast) を使用するには、**'mode=4 lacp_rate=1'** を以下のように設定します。

```
BondInterfaceOvsOptions:
    default: 'mode=4 lacp_rate=1'
    description: The bonding_options string for the bond interface. Set
                 things like lacp=active and/or bond_mode=balance-slb
                 using this option.
    type: string
```

サポートされるその他のボンディングオプションに関する詳細は、**オーバークラウドの高度なカスタマイズ**ガイドの Open vSwitch ボンディングのオプション を参照してください。カスタマイズした **/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml** テンプレートの完全な内容は、付録B *カスタムインターフェイステンプレートの例: 複数のボンディングされたインターフェイス*を参照してください。

第5章 CEPH STORAGE クラスターのカスタマイズ

director は、デフォルト設定を使用してコンテナー化された Red Hat Ceph Storage をデプロイします。Ceph Storage をカスタマイズするには、デフォルト設定を上書きします。

前提条件

コンテナー化された Ceph Storage をデプロイするには、オーバークラウドのデプロイメント時に **/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml** ファイルを追加します。この環境ファイルは、以下のリソースを定義します。

- **CephAnsibleDisksConfig**: このリソースは Ceph Storage ノードのディスクレイアウトをマッピングします。詳細は、「Ceph Storage ノードのディスクレイアウトのマッピング」 を参照してください。
- **CephConfigOverrides**: このリソースは、その他のすべてのカスタム設定を Ceph Storage クラスターに適用します。

これらのリソースを使用して、director がコンテナー化された Ceph Storage に設定するすべてのデフォルトを上書きします。

手順

1. Red Hat Ceph Storage 4 Tools リポジトリーを有効にします。

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

2. **ceph-ansible** パッケージをアンダークラウドにインストールします。

```
$ sudo dnf install ceph-ansible
```

3. Ceph Storage クラスターをカスタマイズするには、新しい環境ファイル (**/home/stack/templates/ceph-config.yaml** など) でカスタムのパラメーターを定義します。お使いの環境ファイルの **parameter_defaults** セクションで以下の構文を使用して、Ceph Storage クラスターの設定を適用することができます。

```
parameter_defaults:
  CephConfigOverrides:
    section:
      KEY:VALUE
```

注記

CephConfigOverrides パラメーターは、**ceph.conf** ファイルの **[global]** セクションのほか、**[osd]**、**[mon]**、および **[client]** などの他のセクションにも適用できます。セクションを指定すると、**key:value** データは指定されたセクションに送信されます。セクションを指定しないと、データはデフォルトで **[global]** セクションに送信されます。Ceph Storage の設定、カスタマイズ、およびサポートされるパラメーターに関する詳細は、Red Hat Ceph Storage 設定ガイド を参照してください。

4. **KEY** および **VALUE** を適用する Ceph クラスター設定に置き換えてください。たとえば、**global** セクションでは **max_open_files** が **KEY** で、**131072** が対応する **VALUE** になります。

```
parameter_defaults:
  CephConfigOverrides:
    global:
      max_open_files: 131072
    osd:
      osd_scrub_during_recovery: false
```

この設定は、Ceph クラスターの設定ファイルで定義された以下のような設定となります。

```
[global]
max_open_files = 131072
[osd]
osd_scrub_during_recovery = false
```

5.1. CEPH-ANSIBLE グループ変数の設定

ceph-ansible ツールは、Ceph Storage クラスターをインストールおよび管理するために使用される Playbook です。

ceph-ansible ツールには **group_vars** ディレクトリーがあり、設定オプションとこれらのオプションのデフォルト設定を定義します。**group_vars** ディレクトリーを使用して Ceph Storage パラメーターを設定します。

group_vars ディレクトリーの詳細は、**Red Hat Ceph Storage Installation Guide**の Installing a Red Hat Ceph Storage cluster を参照してください。

director の変数デフォルトを変更するには、**CephAnsibleExtraConfig** パラメーターを使用して heat 環境ファイルの新しい値を渡します。たとえば、**ceph-ansible** のグループ変数 **journal_size** を 40960 に設定するには、**journal_size** を以下のように定義した環境ファイルを作成します。

```
parameter_defaults:
  CephAnsibleExtraConfig:
    journal_size: 40960
```

重要

ceph-ansible のグループ変数を変更するには、オーバーライドパラメーターを使用します。アンダークラウドの **/usr/share/ceph-ansible** ディレクトリーのグループ変数を直接編集しないでください。

5.2. CEPH STORAGE を使用する RED HAT OPENSTACK PLATFORM 向けの CEPH コンテナー

OpenStack Platform が Ceph を使用するように設定するには、Ceph コンテナーが必要です。これは、外部の Ceph クラスターの場合でも同じです。Red Hat Enterprise Linux 8 と互換性を持たせるには、Red Hat OpenStack Platform (RHOSP) 16 には Red Hat Ceph Storage 4 が必要です。Ceph Storage 4 コンテナーは、registry.redhat.io (認証が必要なレジストリー) でホストされます。

heat 環境パラメーター **ContainerImageRegistryCredentials** を使用して、**registry.redhat.io** で認証することができます。詳細は、コンテナーイメージ準備のパラメーター を参照してください。

5.3. CEPH STORAGE ノードのディスクレイアウトのマッピング

コンテナー化された Ceph Storage をデプロイする場合には、ディスクレイアウトをマッピングし、Ceph OSD サービス用に専用のブロックデバイスを指定する必要があります。カスタム Ceph パラメーターを定義するために作成した環境ファイル (**/home/stack/templates/ceph-config.yaml**) で、このマッピングを行うことができます。

parameter_defaults で **CephAnsibleDisksConfig** リソースを使用して、ディスクレイアウトをマッピングします。このリソースでは、以下の変数が使用されます。

変数	必須/オプション	デフォルト値 (未設定の場合)	説明
osd_scenario	必須	lvm 注記: デフォルト値は **lvm** です。	**lvm** の値により、**ceph-ansible** は **ceph-volume** を使用して OSD、**block.db** および BlueStore WAL デバイスを設定することができます。
devices	必須	なし。変数を設定する必要があります。	ノード上の OSD に使用するブロックデバイスのリスト。
dedicated_devices	必須 (ただし **osd_scenario** が **non-collocated** の場合のみ)	devices	**devices** パラメーターの各エントリーを専用のジャーナル用ブロックデバイスにマッピングするブロックデバイスのリスト。この変数は、**osd_scenario=non-collocated** の場合にのみ使用できます。
dmcrypt	オプション	false	OSD に保存されるデータが暗号化されているか (**true**)、暗号化されていないか (**false**) を設定します。

変数	必須/オプション	デフォルト値 (未設定の場合)	説明
osd_objectstore	オプション	bluestore 注記: デフォルト値は **bluestore** です。	Ceph の使用するストレージバックエンドを設定します。 注記: 値のデフォルトは **bluestore** ですが、コレートまたは非コレートのシナリオで **osd_scenario** を **filestore** に設定できます。**dedicated_devices** がジャーナリングディスクを識別する非コレートのシナリオで、**filestore** に値を設定することができます。デバイスに定義されたディスクを分割し、OSD データとジャーナリングデータの両方を同じデバイスに保存するコレートのシナリオで、値を **filestore** に設定できます。

5.3.1. BlueStore の使用

手順

1. Ceph OSD として使用するブロックデバイスを指定するには、以下のスニペットのバリエーションを使用します。

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sdb
      - /dev/sdc
      - /dev/sdd
      - /dev/nvme0n1
    osd_scenario: lvm
    osd_objectstore: bluestore
```

2. **/dev/nvme0n1** は高パフォーマンスのデバイスクラスにあるため、例に示す **parameter_defaults** では、**/dev/sdb**、**/dev/sdc**、および **/dev/sdd** 上で動作する 3 つの OSD が作成されます。この 3 つの OSD は、**block.db** および BlueStore WAL デバイスに **/dev/nvme0n1** を使用します。**ceph-volume** ツールは、**batch** サブコマンドを使用してこれを実施します。Ceph Storage ノードごとに同じ設定が複製され、ハードウェアが統一されていることを前提とします。**block.db** および BlueStore WAL データが OSD と同じディスクに存在する場合は、以下に示すようにパラメーターのデフォルトを変更します。

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sdb
      - /dev/sdc
      - /dev/sdd
    osd_scenario: lvm
    osd_objectstore: bluestore
```

5.3.2. 永続デバイス名が付いたデバイスの参照

手順

1. 一部のノードでは、**/dev/sdb** および **/dev/sdc** 等のディスクパスが、リブート中に同じブロックデバイスをポイントしない場合があります。このようなケースが Ceph Storage ノードで見られる場合は、各ディスクに **/dev/disk/by-path/** シンボリックリンクを指定して、ブロックデバイスのマッピングがデプロイメント全体で一貫性を保つようにします。

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:10:0
      - /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:11:0

    dedicated_devices:
      - /dev/nvme0n1
      - /dev/nvme0n1
```

2. (オプション) オーバークラウドのデプロイメント前に OSD デバイスのリストを設定しなければならないので、ディスクデバイスの PCI パスを把握/設定できない場合があります。このような場合には、イントロスペクション中にブロックデバイスの **/dev/disk/by-path/symlink** データを収集します。
以下の例の最初のコマンドを実行して、サーバー **b08-h03-r620-hci** のアンダークラウド Object Storage サービス (swift) からイントロスペクションデータをダウンロードし、そのデータを **b08-h03-r620-hci.json** と呼ばれるファイルに保存します。2 番目のコマンドを実行して、"by-path" を grep します。このコマンドの出力には、ディスクの特定に使用できる一意の **/dev/disk/by-path** 値が含まれます。

```
(undercloud) [stack@b08-h02-r620 ironic]$ openstack baremetal introspection data save
b08-h03-r620-hci | jq . > b08-h03-r620-hci.json
(undercloud) [stack@b08-h02-r620 ironic]$ grep by-path b08-h03-r620-hci.json
        "by_path": "/dev/disk/by-path/pci-0000:02:00.0-scsi-0:2:0:0",
        "by_path": "/dev/disk/by-path/pci-0000:02:00.0-scsi-0:2:1:0",
        "by_path": "/dev/disk/by-path/pci-0000:02:00.0-scsi-0:2:3:0",
        "by_path": "/dev/disk/by-path/pci-0000:02:00.0-scsi-0:2:4:0",
        "by_path": "/dev/disk/by-path/pci-0000:02:00.0-scsi-0:2:5:0",
        "by_path": "/dev/disk/by-path/pci-0000:02:00.0-scsi-0:2:6:0",
        "by_path": "/dev/disk/by-path/pci-0000:02:00.0-scsi-0:2:7:0",
        "by_path": "/dev/disk/by-path/pci-0000:02:00.0-scsi-0:2:0:0",
```

ストレージデバイスの命名規則の詳細は、**ストレージデバイスの管理**ガイドの 永続的な命名属性の概要 を参照してください。

5.3.3. 高度なシナリオでの OSD の設定

環境ファイルにおいて、**CephAnsibleDisksConfig** リソースの **devices** 変数で、OSD に使用するブロックデバイスをリスト表示します。

他のデバイス設定パラメーターを指定せずに **devices** 変数を使用する場合、**ceph-volume lvm batch** は、高パフォーマンスデバイスを低速なデバイスの **block.db** として均等に共有し、OSD 設定を自動的に最適化します。

以下の手順を使用して、**ceph-volume lvm batch** モードで実行しないように **devices** を設定できます。

5.3.3.1. block.db を使用したパフォーマンスの向上

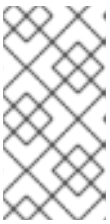
block.db を使用すると、スループットを高め、応答時間を改善することで、Ceph Storage クラスターのパフォーマンスを向上させることができます。**block.db** は、データセグメントと BlueStore ログ先行書き込み (WAL) で設定されるデータベースです。

手順

1. 環境ファイルに以下の内容を追加してください。

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sda
      - /dev/sdb
      - /dev/nvme0n1
      - /dev/sdc
      - /dev/sdd
      - /dev/nvme0n2
    osd_scenario: lvm
    osd_objectstore: bluestore
```

これにより、**sda**、**sdb**、**sdc**、および **sdd** の 4 つの OSD が設定されます。各ペアには、独自のデータベース **nvem0n1** および **nvme0n2** があります。



注記

devices リスト内のデバイスの順序は重要です。ドライブに続いて block.db および BlueStore WAL (DB-WAL) デバイスをリスト表示します。この例では、**nvme0n1** は **sda** および **sdb** の DB-WAL で、**nvme0n2** は **sdc** および **sdd** の DB-WAL です。詳細は、BlueStore の使用 を参照してください。

2. オーバークラウドのデプロイ時に、**-e** オプションを使用して、デプロイメントコマンドに新しいコンテンツが含まれる環境ファイルを追加します。

5.3.3.2. 専用のログ先行書き込み (WAL) デバイスの使用

専用のログ先行書き込み (WAL) デバイスを指定できます。**devices**、**dedicated_devices**、および **bluestore_wal_devices** を一緒に使用すると、OSD のすべてのコンポーネントを別々のデバイスに分割できるため、パフォーマンスが向上します。

以下の手順例では、別の追加ディクショナリー **bluestore_wal_devices** が、NVMe デバイス **nvme0n1** および **nvme0n2** のログ先行書き込みを分離します。

手順

1. 環境ファイルに以下の内容を追加してください。

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sda
      - /dev/sdb
    dedicated_devices:
      - /dev/sdx
      - /dev/sdy
    bluestore_wal_devices:
      - /dev/nvme0n1
      - /dev/nvme0n2
```

2. オーバークラウドのデプロイ時に、**-e** オプションを使用して、デプロイメントコマンドに新しいコンテンツが含まれる環境ファイルを追加します。

5.3.3.3. 制御性の向上のための事前作成 LVM の使用

以前の高度なシナリオでは、**ceph-volume** は異なる種別のデバイスリストを使用して OSD の論理ボリュームを作成します。**ceph-volume** を実行する前に論理ボリュームを作成し、それらの論理ボリュームの **lvm_volumes** リストを **ceph-volume** に渡すこともできます。これには、論理ボリュームを事前に作成する必要がありますが、制御がより正確になります。director はハードウェアをプロビジョニングするロールりも果たすため、初回ブートスクリプトを使用してこれらの LVM を事前に作成する必要があります。

手順

1. **OS::TripleO::NodeUserData** リソース種別として heat テンプレートを登録する、以下の内容の環境ファイル **/home/stack/templates/firstboot.yaml** を作成します。

```
resource_registry:
  OS::TripleO::NodeUserData: /home/stack/templates/ceph-lvm.yaml
```

2. 環境ファイル **/home/stack/templates/ceph-lvm.yaml** を作成します。3 つの物理ボリュームを含む以下の例のようなリストを追加します。デバイスリストが長い場合は、要件に応じて例をデプロイメントします。

```
heat_template_version: 2014-10-16

description: >
  Extra hostname configuration

resources:
  userdata:
    type: OS::Heat::MultipartMime
    properties:
      parts:
      - config: {get_resource: ceph_lvm_config}

  ceph_lvm_config:
    type: OS::Heat::SoftwareConfig
    properties:
```

```
      config: |
        #!/bin/bash -x
        pvcreate /dev/sda
        vgcreate ceph_vg_hdd /dev/sda
        pvcreate /dev/sdb
        vgcreate ceph_vg_ssd /dev/sdb
        pvcreate /dev/nvme0n1
        vgcreate ceph_vg_nvme /dev/nvme0n1
        lvcreate -n ceph_lv_wal1 -L 50G ceph_vg_nvme
        lvcreate -n ceph_lv_db1 -L 500G ceph_vg_ssd
        lvcreate -n ceph_lv_data1 -L 5T ceph_vg_hdd
        lvs

  outputs:
    OS::stack_id:
      value: {get_resource: userdata}
```

3. 以下の方法で、デバイスリストの代わりに **lvm_volumes** パラメーターを使用します。これは、ボリュームグループと論理ボリュームがすでに作成されていることを前提とします。このシナリオの一般的なユースケースでは、WAL および DB LV が SSD 上にあり、データ LV が HDD 上にあります。

```
parameter_defaults:
  CephAnsibleDisksConfig:
    osd_objectstore: bluestore
    osd_scenario: lvm
    lvm_volumes:
      - data: ceph_lv_data1
        data_vg: ceph_vg_hdd
        db: ceph_lv_db1
        db_vg: ceph_vg_ssd
        wal: ceph_lv_wal1
        wal_vg: ceph_vg_nvme
```

4. オーバークラウドのデプロイ時に、**-e** オプションを使用して、デプロイメントコマンドに新しいコンテンツが含まれる環境ファイルを追加します。

注記

WAL デバイスが DB デバイスよりも優れたパフォーマンスのハードウェアに存在する場合に限り、別の WAL デバイスを指定する必要があります。通常、別の DB デバイスを作成するだけで十分で、同じパーティションが WAL 機能に使用されます。

5.4. 異なる CEPH プールへのカスタムの属性の割り当て

デフォルトでは、director で作成した Ceph Storage プールには、同じ配置グループの数 (**pg_num** および **pgp_num**) とサイズが設定されます。5章*Ceph Storage クラスターのカスタマイズ*に記載のいずれかの方法を使用して、これらの設定をグローバルに上書きすることが可能です。これを行うと、すべてのプールに同じ値が適用されます。

CephPools パラメーターを使用して各 Ceph Storage プールに異なる属性を適用するか、新しいカスタムプールを作成します。

手順

1. **POOL** を、設定するプールの名前に置き換えます。

```
parameter_defaults:
  CephPools:
    - name: POOL
```

2. 以下のいずれかを実行して配置グループを設定します。

   - デフォルト設定を手動で上書きするには、**pg_num** を配置グループの数に設定します。

     ```
     parameter_defaults:
       CephPools:
         - name: POOL
           pg_num: 128
           application: rbd
     ```

   - あるいは、配置グループを自動的にスケーリングするには、**pg_autoscale_mode** を **True** に設定し、**target_size_ratio** を予想される Ceph Storage 要件の割合に設定します。

     ```
     parameter_defaults:
       CephPools:
         - name: POOL
           pg_autoscale_mode: True
           target_size_ratio: PERCENTAGE
           application: rbd
     ```

     PERCENTAGE を小数に置き換えます。たとえば、0.5 は 50 パーセントです。割合の合計は 1.0 または 100 パーセントと同じでなければなりません。

     以下の値は例としてのみ例示します。

     ```
     paramter_defaults:
       CephPools:
         - {"name": backups, "target_size_ratio": 0.1, "pg_autoscale_mode": True, "application":
     rbd}
         - {"name": volumes, "target_size_ratio": 0.5, "pg_autoscale_mode": True, "application":
     rbd}
         - {"name": vms,     "target_size_ratio": 0.2, "pg_autoscale_mode": True, "application":
     rbd}
         - {"name": images,  "target_size_ratio": 0.2, "pg_autoscale_mode": True, "application":
     rbd}
     ```

     詳細は、**Red Hat Ceph Storage インストールガイド**の 配置グループ autoscaler 参照してください。

3. アプリケーション種別を指定します。
   Compute、Block Storage、および Image Storage のアプリケーション種別は、rbd です。ただし、プールを使用する対象に応じて、異なるアプリケーション種別を指定できます。

   たとえば、gnocchi メトリックプールのアプリケーション種別は **openstack_gnocchi** です。
   詳細は、**Storage Strategies Guide**の Enable Application を参照してください。

注記

CephPools パラメーターを使用しない場合には、director により適切なアプリケーションの種別が自動的に設定されます。ただし、デフォルトのプールのリストだけが対象です。

4. オプション: カスタムプールを作成するために **custompool** というプールを追加し、お使いの環境の要件に固有のパラメーターを設定します。

```
parameter_defaults:
  CephPools:
    - name: custompool
      pg_num: 128
      application: rbd
```

ヒント

一般的な Ceph ユースケースの標準的なプール設定は、Ceph Placement Groups (PGs) per Pool Calculator を参照してください。この計算ツールは通常、Ceph プールを手動で設定するためのコマンドの生成に使用されます。このデプロイメントでは、仕様に基づいて director がプールを設定します。

警告

Red Hat Ceph Storage 3 (Luminous) では、OSD に設定可能な最大 PG 数 (デフォルトは 200) にハード制限が追加されました。このパラメーターは 200 を超える値に上書きしないでください。Ceph PG の数が最大値を超えたことで問題が発生した場合には、**mon_max_pg_per_osd** ではなく、**pg_num** をプールごとに調整して問題に対処します。

5.5. 異なる CEPH STORAGE ノードのパラメーターのオーバーライド

CephStorage や **ComputeHCI** など、Ceph OSD をホスティングするロールを持つすべてのノードは、「Ceph Storage ノードのディスクレイアウトのマッピング」 で作成されたグローバルな **devices** および **dedicated_devices** リストを使用します。これらのリストは、これらすべてのサーバーが同じハードウェアを持っていることを前提としています。これが異なるハードウェアを備えたサーバーがある場合は、ノード固有のディスク設定を使用して、さまざまな **devices** および **dedicated_devices** リストの詳細で Director を更新する必要があります。

注記

Ceph OSD をホスティングするロールには、**roles_data.yaml** ファイルに **OS::TripleO::Services::CephOSD** サービスが含まれます。

他のノードと同じハードウェアを持たない Ceph Storage ノードは、パフォーマンスの問題を引き起こす可能性があります。Red Hat OpenStack Platform(RHOSP) 環境で、標準的なノードとノード固有のオーバーライドを使用して設定したノードとの間の差異が大きくなるほど、パフォーマンスが低下する可能性が高くなります。

5.5.1. ノード固有のディスク設定

Director は、同じハードウェアを持たないサービス用に設定する必要があります。これは、ノード固有のディスク設定と呼ばれます。

次のいずれかの方法を使用して、ノード固有のディスク設定を作成できます。

- 自動: JSON heat 環境ファイルを生成して、ノード固有のディスク設定を自動的に作成できます。
- 手動: ノードのディスクレイアウトを変更して、ノード固有のディスク設定を作成できます。

5.5.1.1. Ceph デバイス用の JSON heat 環境ファイルの生成

/usr/share/openstack-tripleo-heat-templates/tools/make_ceph_disk_list.py スクリプトを使用すると、Bare Metal Provisioning サービス (ironic) のイントロスペクションデータから有効な JSON heat 環境ファイルを自動的に作成することができます。この JSON ファイルを使用して、ノード固有のディスク設定を director に渡します。

手順

1. デプロイする Ceph ノードの Bare Metal Provisioning サービスからイントロスペクションデータをエクスポートします。

```
openstack baremetal introspection data save oc0-ceph-0 > ceph0.json
openstack baremetal introspection data save oc0-ceph-1 > ceph1.json
...
```

2. ユーティリティーをアンダークラウド上の **stack** ユーザーのホームディレクトリーにコピーし、それを使用して **node_data_lookup.json** ファイルを生成します。

```
./make_ceph_disk_list.py -i ceph*.json -o node_data_lookup.json -k by_path
```

3. **NodeDataLookup** はデプロイメント中に 1 回しか定義できないため、Ceph OSD をホストするすべてのノードの **openstack baremetal introspection data save** コマンドからイントロスペクションデータファイルをユーティリティーに渡します。**-i** オプションには、入力として ***.json** ような表現またはファイルのリストを指定することができます。
-k オプションを使用して、OSD ディスクの識別に使用する Bare Metal Provisioning ディスクデータ構造のキーを定義します。**name** は使用しないでください。**/dev/sdd** のようなデバイスのファイルを生成し、リブート時に同じデバイスをポイントするとは限らないためです。代わりに **by_path** を使います。**-k** を指定しない場合は、この設定がデフォルトとなります。

   Bare Metal Provisioning サービスは、システムで利用可能なディスクのいずれかをルートディスクとして確保します。ユーティリティーは、生成されたデバイスのリストからルートディスクを常に除外します。

4. (オプション): **./make_ceph_disk_list.py –help** を使用すると、他の利用可能なオプションを確認できます。

5. オーバークラウドをデプロイする際に、ご自分の環境に該当するその他の環境ファイルと共に **node_data_lookup.json** ファイルを追加します。

```
$ openstack overcloud deploy \
--templates \
…
```

```
-e <existing_overcloud_environment_files> \
-e node_data_lookup.json \
…
```

5.5.1.2. Ceph Storage ノードのディスクレイアウトの変更

重要

非同種の Ceph Storage ノードは、パフォーマンスの問題を引き起こす可能性があります。Red Hat OpenStack Platform(RHOSP) 環境で、標準的なノードとノード固有のオーバーライドを使用して設定したノードとの間の差異が大きくなるほど、パフォーマンスが低下する可能性が高くなります。

ノード固有のディスク設定を director に渡すには、**node-spec-overrides.yaml** 等の heat 環境ファイルを **openstack overcloud deploy** コマンドに渡す必要があります。そして、この環境ファイルの内容で、それぞれのサーバーをマシン固有の UUID およびグローバル変数を上書きするローカル変数のリストで識別する必要があります。

マシン固有の UUID は、個々のサーバー向けに、または Bare Metal Provisioning サービス (ironic) データベースから抽出することができます。

注記

以下の手順では、埋め込まれた有効な JSO が含まれる有効な YAML 環境ファイルを作成します。**make_ceph_disk_list.py** で完全な JSON ファイルを生成し、それを YAML のようにデプロイメントコマンドに渡すこともできます。詳細については、Ceph デバイス用の JSON heat 環境ファイルの生成 を参照してください。

手順

1. 個々のサーバーの UUID を把握するには、そのサーバーにログインして以下のコマンドを入力します。

```
$ dmidecode -s system-uuid
```

2. Bare Metal Provisioning サービスデータベースから UUID を抽出するには、アンダークラウドで以下のコマンドを入力します。

```
$ openstack baremetal introspection data save NODE-ID | jq .extra.system.product.uuid
```

警告

アンダークラウドのインストールまたはアップグレードおよびイントロスペクションの前に、**undercloud.conf** で **inspection_extras = true** がない場合には、マシン固有の UUID は Bare Metal Provisioning サービスデータベースには含まれません。

重要

マシン固有の UUID は、Bare Metal Provisioning サービスの UUID ではありません。

有効な **node-spec-overrides.yaml** ファイルは、以下のようになる可能性があります。

```
parameter_defaults:
  NodeDataLookup: {"32E87B4C-C4A7-418E-865B-191684A6883B": {"devices":
["/dev/sdc"]}}
```

3. 最初の 2 行以降は、すべて有効な JSON でなければなりません。**jq** コマンドを使用して、JSON が有効であることを確認します。

   a. 最初の 2 行 (**parameter_defaults:** および **NodeDataLookup:**) を一時的にファイルから削除します。

   b. **cat node-spec-overrides.yaml | jq .** を実行します。

4. **node-spec-overrides.yaml** ファイルが大きくなるにつれ、**jq** コマンドを使用して組み込まれた JSON が有効であることを確認することもできます。たとえば、**devices** および **dedicated_devices** のリストは同じ長さでなければならないので、以下のコマンドを使用して、デプロイメントを開始する前に両者が同じ長さであることを確認します。以下の例では、**node-spec-c05-h17-h21-h25-6048r.yaml** にはラック c05 に 3 つのサーバーがあります。ラック c05 のスロット h17、h21、および h25 にはディスクがありません。

```
(undercloud) [stack@b08-h02-r620 tht]$ cat node-spec-c05-h17-h21-h25-6048r.yaml | jq '.[] |
.devices | length'
33
30
33
(undercloud) [stack@b08-h02-r620 tht]$ cat node-spec-c05-h17-h21-h25-6048r.yaml | jq '.[] |
.dedicated_devices | length'
33
30
33
(undercloud) [stack@b08-h02-r620 tht]$
```

5. JSON を検証した後に、有効な環境 YAML ファイルとなるように 2 つの行 (**parameter_defaults:** および **NodeDataLookup:**) を戻し、**-e** を使用してこの環境ファイルをデプロイメントに追加します。以下の例では、更新した Heat 環境ファイルは Ceph デプロイメントの **NodeDataLookup** を使用しています。すべてのサーバーのデバイスリストは 35 のディスクで設定されていましたが、1 つのサーバーだけ 1 つのディスクがありませんでした。この環境ファイルは、その 1 つのノードについてのみデフォルトのデバイスリストを上書きし、グローバルリストの代わりに使用する必要のある 34 のディスクのリストを提供します。

6. JSON を検証した後に、有効な環境 YAML ファイルとなるように 2 つの行 (**parameter_defaults:** および **NodeDataLookup:**) を戻し、**-e** を使用してこの環境ファイルをデプロイメントコマンドに追加します。
   以下の例では、更新した heat 環境ファイルは Ceph デプロイメントの **NodeDataLookup** を使用しています。すべてのサーバーのデバイスリストは 35 のディスクで設定されていましたが、1 つのサーバーだけ 1 つのディスクがありませんでした。この環境ファイルは、その 1 つのノードについてのみデフォルトのデバイスリストを上書きし、グローバルリストの代わりに使用する必要のある 34 のディスクのリストをノードに提供します。

```
parameter_defaults:
  # c05-h01-6048r is missing scsi-0:2:35:0 (00000000-0000-0000-0000-0CC47A6EFD0C)
  NodeDataLookup: {
    "00000000-0000-0000-0000-0CC47A6EFD0C": {
      "devices": [
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:1:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:32:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:2:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:3:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:4:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:5:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:6:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:33:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:7:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:8:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:34:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:9:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:10:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:11:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:12:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:13:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:14:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:15:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:16:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:17:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:18:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:19:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:20:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:21:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:22:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:23:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:24:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:25:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:26:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:27:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:28:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:29:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:30:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:31:0"
        ],
      "dedicated_devices": [
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
```

```
"/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1"
  ]
 }
}
```

5.5.2. BlueStore block.db サイズの変更

BlueStore **block.db** はデータセグメントと BlueStore ログ先行書き込み (WAL) のデータベースです。データベースのサイズを変更するには、2 つの方法があります。これらの方法のいずれかを選択して、サイズを変更します。

5.5.2.1. ceph-volume 使用時の BlueStore block.db サイズの変更

ceph-volume を使用するときに **block.db** のサイズをオーバーライドするには、次の手順を使用します。**ceph-volume** は **osd_scenario: lvm** のときに使用されます。**ceph-volume** は **block.db** のサイズを自動的に設定します。しかし、高度なシナリオでは **block.db** のサイズをオーバーライドすることができます。

次の例では、使用される **block_db_size** が **ceph-volume** コールに渡されるように、Ceph 設定ファイルのオーバーライドではなく、**ceph-ansible** ホスト変数を使用しています。

手順

1. 以下のような内容の JSON 環境ファイルを作成しますが、必要に応じて値を置き換えてください。

```
{
  "parameter_defaults": {
    "NodeDataLookup": {
      "32e87b4c-c4a7-41be-865b-191684a6883b": {
        "block_db_size": 3221225472
      },
      "ea6a84d6-cf89-4fe2-b7bd-869b3fe4dd6b": {
        "block_db_size": 3221225472
      }
```

```
    }
  }
}
```

2. オーバークラウドをデプロイする際に、ご自分の環境に該当するその他の環境ファイルと共に JSON ファイルを追加します。

```
$ openstack overcloud deploy \
--templates \
…
-e <existing_overcloud_environment_files> \
-e <json_environment_file>  \
…
```

5.5.2.2. ceph-disk 使用時の BlueStore block.db サイズの変更

ceph-disk を使用するときに **block.db** のサイズをオーバーライドするには、次の手順を使用します。**osd_scenario: non-collocated** または **osd_scenario: collocated** の場合は、**ceph-disk** が使用されます。

次の例では、特定のノードの Ceph 設定のオーバーライドを使用して、**blustore_block_db_size** を設定しています。この Ceph 設定オプションは、**ceph-volume** の使用時には無視されますが、**ceph-disk** ではこの設定オプションが使用されます。

手順

1. 以下のような内容の JSON 環境ファイルを作成しますが、必要に応じて値を置き換えてください。

```
{
  "parameter_defaults": {
    "NodeDataLookup": {
      "32e87b4c-c4a7-41be-865b-191684a6883b": {
        "ceph_conf_overrides": {
          "osd": {
            "bluestore_block_db_size": 3221225472
          }
        }
      },
      "ea6a84d6-cf89-4fe2-b7bd-869b3fe4dd6b": {
        "ceph_conf_overrides": {
          "osd": {
            "bluestore_block_db_size": 3221225472
          }
        }
      }
    }
  }
}
```

2. オーバークラウドをデプロイする際に、ご自分の環境に該当するその他の環境ファイルと共に JSON ファイルを追加します。

```
$ openstack overcloud deploy \
```

```
--templates \
…
-e <existing_overcloud_environment_files> \
-e <json_environment_file>  \
…
```

5.6. 大規模 CEPH クラスターでの再開待機時間の延長

デプロイメント時に、OSD やモニターなどの Ceph サービスは再起動され、サービスが再度実行されるまでデプロイメントは続行されません。Ansible は 15 秒間待って (待機) サービスの開始を確認する行為を 5 回繰り返します (リトライ)。サービスが再開されない場合には、オペレーターが操作できるようにデプロイメントは停止します。

Ceph クラスターのサイズによっては、リトライまたは待機の値を増加しなければならない場合があります。これらのパラメーターの正確な名前およびそのデフォルト値は以下のとおりです。

```
health_mon_check_retries: 5
health_mon_check_delay: 15
health_osd_check_retries: 5
health_osd_check_delay: 15
```

手順

1. **CephAnsibleExtraConfig** パラメーターを更新して、待機およびリトライのデフォルト値を変更します。

```
parameter_defaults:
  CephAnsibleExtraConfig:
    health_osd_check_delay: 40
    health_osd_check_retries: 30
    health_mon_check_delay: 20
    health_mon_check_retries: 10
```

この例でのクラスターは、Ceph OSD の場合は 30 回確認 (確認ごとに 40 秒間待機) し、Ceph MON の場合は 20 回確認 (確認ごとに 10 秒間待機) します。

2. 変更を反映するには、**openstack overcloud deploy** を使用して **-e** を指定し、更新された **yaml** ファイルを渡します。

5.7. ANSIBLE 環境変数のオーバーライド

Red Hat OpenStack Platform Workflow サービス (mistral) は Ansible を使用して Ceph Storage を設定しますが、Ansible 環境変数を使用して Ansible 環境をカスタマイズすることができます。

手順

ANSIBLE_* 環境変数をオーバーライドするには、**CephAnsibleEnvironmentVariables** heat テンプレートパラメーターを使用します。

以下に示す設定例では、フォークおよび SSH のリトライ回数を増やします。

```
parameter_defaults:
  CephAnsibleEnvironmentVariables:
    ANSIBLE_SSH_RETRIES: '6'
```

```
DEFAULT_FORKS: '35'
```

Ansible 環境変数の詳細は、Ansible Configuration Settings を参照してください。

Ceph Storage クラスターのカスタマイズ方法に関する詳細は、Ceph Storage Cluster のカスタマイズ を参照してください。

5.8. CEPH 伝送時暗号化の有効化

Red Hat Ceph Storage 4 以降、messenger バージョン 2 プロトコルの導入により、ネットワーク上のすべての Ceph トラフィックの暗号化を有効にすることができます。messenger v2 の **secure mode** 設定は、Ceph デーモンと Ceph クライアント間の通信を暗号化するため、エンドツーエンドの暗号化を行います。

この機能は、本リリースでは **テクノロジープレビュー** として提供しているため、Red Hat では全面的にはサポートしていません。これは、テスト用途にのみご利用いただく機能です。実稼働環境にはデプロイしないでください。テクノロジープレビュー機能についての詳しい情報は、対象範囲の詳細 を参照してください。

注記

この機能は現在テクノロジープレビューとして提供されていますが、Red Hat OpenStack Platform (RHOSP) バージョン 16.1 以降で使用することが意図されています。外部の Red Hat Ceph Storage バージョン 4 を使用する RHOSP バージョン 13 デプロイメントではサポートされません。詳しい情報は、**Red Hat Ceph Storage アーキテクチャーガイド**の Ceph on-wire 暗号化 を参照してください。

RHOSP で Ceph 伝送時暗号化を有効にするには、お使いの環境オーバーライドファイルで以下のパラメーターを設定します。

```
parameter_defaults:
  CephConfigOverrides:
    global:
      ms_cluster_mode: secure
      ms_service_mode: secure
      ms_client_mode: secure
```

Ceph 伝送時暗号化に関する詳しい情報は、**アーキテクチャーガイド**の Ceph on-wire 暗号化 を参照してください。

第6章 DIRECTOR を使用した CEPH STORAGE クラスターのさまざまなワークロード用のパフォーマンス層の定義

Red Hat OpenStack Platform (RHOSP) director を使用して、異なる Red Hat Ceph Storage パフォーマンス層をデプロイすることができます。Ceph CRUSH ルールと **CephPools** director パラメーターを組み合わせ、デバイスクラス機能を使用して異なる層を構築して、異なるパフォーマンス要件のあるワークロードに対応することができます。たとえば、通常のワークロード用に HDD クラスと、高パフォーマンスのロードのために SSD 上でのみデータを分散する SSD クラスを定義できます。このシナリオでは、新規の Block Storage ボリュームを作成する場合には、HDD または SSD のいずれかのパフォーマンス層を選択することができます。

警告

既存の環境でパフォーマンス層を定義すると、Ceph クラスター内で大量のデータが移動する場合があります。スタックの更新時に director がトリガーする **ceph-ansible** には、プールがクラスターにすでに定義されているかどうかや、データが含まれるかどうかを確認するロジックはありません。つまり、プールに関連付けられたデフォルトの CRUSH ルールを変更すると、データの移動が行われるため、既存の環境でパフォーマンス層を定義することは危険となる可能性があります。ノードの追加または削除に関する支援または推奨事項が必要な場合は、Red Hat サポートにお問い合わせください。

注記

Ceph はディスク種別を自動検出し、Linux カーネルが公開するハードウェアプロパティーに基づいて、これを対応するデバイスクラス (HDD、SSD、または NVMe のいずれか) に割り当てます。ただし、必要に応じてカテゴリーをカスタマイズすることもできます。

前提条件

- 新規デプロイメントでは、Red Hat Ceph Storage (RHCS) バージョン 4.1 以降
- 既存のデプロイメントの場合は、Red Hat Ceph Storage (RHCS) バージョン 4.2 以降

異なる Red Hat Ceph Storage パフォーマンス層をデプロイするには、CRUSH マップの詳細が含まれる新規の環境ファイルを作成し、これをデプロイメントコマンドに追加します。

以下の手順では、各 Ceph Storage ノードには OSD が 3 つ含まれます。**sdb** および **sdc** は動作中のディスクで、**sdc** は SSD です。Ceph は正しいディスク種別を自動的に検出します。次に、HDD および SSD の 2 つの CRUSH ルールを設定し、2 つのデバイスクラスにそれぞれマッピングします。HDD ルールはデフォルトで、別のルールでプールを設定しない限り、すべてのプールに適用されます。

最後に、**fastpool** と呼ばれる追加のプールを作成し、SSD ルールにマッピングします。このプールは、最終的に Block Storage (cinder) バックエンドを通じて公開されます。この Block Storage バックエンドを使用するすべてのワークロードは、パフォーマンスを高速にする場合にのみ SSD によってサポートされます。これは、データまたはボリュームから起動 (boot from volume) のいずれかに活用できます。

6.1. パフォーマンス層の設定

警告

既存の環境でパフォーマンス層を定義すると、Ceph クラスター内で大量のデータが移動する場合があります。スタックの更新時に director がトリガーする **ceph-ansible** には、プールがクラスターにすでに定義されているかどうかや、データが含まれるかどうかを確認するロジックはありません。つまり、プールに関連付けられたデフォルトの CRUSH ルールを変更すると、データの移動が行わ

れるため、既存の環境でパフォーマンス層を定義することは危険となる可能性があります。ノードの追加または削除に関する支援または推奨事項が必要な場合は、Red Hat サポートにお問い合わせください。

director はこの機能に対応するための特定のパラメーターを公開しませんが、以下の手順に従って **ceph-ansible** の想定される変数を生成することができます。

手順

1. アンダークラウドノードに **stack** ユーザーとしてログインします。
2. Ceph config パラメーターおよびデバイスクラス変数を含む環境ファイル (**/home/stack/templates/ceph-config.yaml** 等) を作成します。あるいは、既存の環境ファイルに以下の設定を追加することができます。
3. 環境ファイルで **CephAnsibleDisksConfig** パラメーターを使用して、Ceph OSD として使用するブロックデバイスをリスト表示します。

```
CephAnsibleDisksConfig:
  devices:
    - /dev/sdb
    - /dev/sdc
    - /dev/sdd
  osd_scenario: lvm
  osd_objectstore: bluestore
```

4. オプション: Ceph はディスクの種別を自動的に検出して、対応するデバイスクラスに割り当てます。ただし、**crush_device_class** プロパティーを使用して、特定のデバイスを特定のクラスに属するように強制したり、独自のカスタムクラスを作成したりすることもできます。以下の例は、指定したクラスを持つ OSD のリストと同じです。

```
CephAnsibleDisksConfig:
  lvm_volumes:
    - data: '/dev/sdb'
      crush_device_class: 'hdd'
    - data: '/dev/sdc'
      crush_device_class: 'hdd'
    - data: '/dev/sdd'
      crush_device_class: 'ssd'
  osd_scenario: lvm
  osd_objectstore: bluestore
```

5. **CephAnsibleExtraVars** パラメーターを追加します。**crush_rules** パラメーターには、定義した各クラスまたは Ceph が自動検出する各クラスにルールが含まれている必要があります。新しいプールを作成する際にルールが指定されていない場合は、Ceph が使用するルールがデフォルトとして選択されます。

```
CephAnsibleExtraConfig:
  crush_rule_config: true
  create_crush_tree: true
  crush_rules:
    - name: HDD
      root: default
      type: host
      class: hdd
```

```
    default: true
  - name: SSD
    root: default
    type: host
    class: ssd
    default: false
```

6. **CephPools** パラメーターを追加します。

   - **rule_name** パラメーターを使用して、デフォルトのルールを使用しない各プールの層を指定します。以下の例では、**fastpool** プールは、fast tier として設定された SSD デバイスクラスを使用して Block Storage ボリュームを管理します。

   - **<appropriate_PG_num>** を配置グループ (PG) の適切な数に置き換えます。あるいは、配置グループの自動スケーラーを使用して、Ceph プールの PG の数を計算します。
     詳細は、異なる Ceph プールへのカスタム属性の割り当て を参照してください。

   - **CinderRbdExtraPools** パラメーターを使用して、**fastpool** を Block Storage バックエンドとして設定します。

     ```
     CephPools:
       - name: fastpool
         pg_num: <appropraiate_PG_num>
         rule_name: SSD
         application: rbd
     CinderRbdExtraPools: fastpool
     ```

7. 以下の例を使用して、環境ファイルに正しい値が含まれることを確認します。

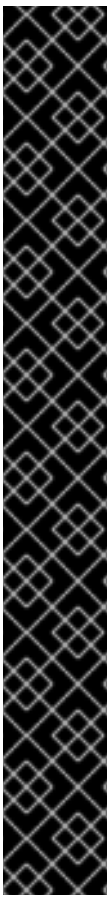
```
parameter_defaults:
    CephAnsibleDisksConfig:
        devices:
            - '/dev/sdb'
            - '/dev/sdc'
            - '/dev/sdd'
        osd_scenario: lvm
        osd_objectstore: bluestore
    CephAnsibleExtraConfig:
        crush_rule_config: true
        create_crush_tree: true
        crush_rules:
            - name: HDD
              root: default
              type: host
              class: hdd
              default: true
            - name: SSD
              root: default
              type: host
              class: ssd
              default: false
    CinderRbdExtraPools: fastpool
    CephPools:
        - name: fastpool
```

```
    pg_num: <appropriate_PG_num>
    rule_name: SSD
    application: rbd
```

8. **openstack overcloud deploy** コマンドで新規の環境ファイルを指定します。

```
$ openstack overcloud deploy \
--templates \
…
-e <other_overcloud_environment_files> \
-e /home/stack/templates/ceph-config.yaml  \
…
```

<other_overcloud_environment_files> をデプロイメントに追加する環境ファイルのリストに置き換えます。



重要

既存の Ceph クラスターに環境ファイルを適用すると、既存の Ceph プールは新しいルールで更新されません。このため、デプロイメントの完了後に以下のコマンドを入力して、指定したプールにルールを設定する必要があります。

```
$ ceph osd pool set <pool> crush_rule <rule>
```

- **<pool>** を新しいルールを適用するプールの名前に置き換えます。
- **<rule>** を **crush_rules** パラメーターで指定したルール名の 1 つに置き換えます。
- **<appropriate_PG_num>** を配置グループの適切な数または **target_size_ratio** に置き換え、**pg_autoscale_mode** を **true** に設定します。

このコマンドを使用してルールを変更するたびに、既存のエントリーを更新するか、既存のテンプレートの **CephPools** パラメーターに新しいエントリーを追加します。

```
CephPools:
  - name: <pool>
    pg_num: <appropriate_PG_num>
    rule_name: <rule>
    application: rbd
```

6.2. BLOCK STORAGE (CINDER) 種別の新しい CEPH プールへのマッピング

警告

既存の環境でパフォーマンス層を定義すると、Ceph クラスター内で大量のデータが移動する場合があります。スタックの更新時に director がトリガーする **ceph-ansible** には、プールがクラスターにすでに定義されているかどうかや、データが含まれるかどうかを確認するロジックはありません。つまり、プールに関連付けられたデフォルトの CRUSH ルールを変更すると、データの移動が行われるため、既存の環境でパフォーマンス層を定義することは危険となる可能性があります。ノードの追加または削除に関する支援または推奨事項が必要な場合は、Red Hat サポートにお問い合わせください。

設定手順を完了したら、Block Storage (cinder) を使用して作成した **fastpool** 層にマッピングされた種別を作成して、RHOSP テナントにパフォーマンス層の機能を使用できるようにします。

手順

1. アンダークラウドノードに **stack** ユーザーとしてログインします。
2. source コマンドで **overcloudrc** ファイルを読み込みます。

```
$ source overcloudrc
```

3. Block Storage ボリュームの既存種別を確認します。

```
$ cinder type-list
```

4. 新規の Block Storage ボリューム fast_tier を作成します。

```
$ cinder type-create fast_tier
```

5. Block Storage 種別が作成されていることを確認します。

```
$ cinder type-list
```

6. **fast_tier** Block Storage 種別が利用可能な場合は、作成した新しい層の Block Storage ボリュームバックエンドとして **fastpool** を設定します。

```
$ cinder type-key fast_tier set volume_backend_name=tripleo_ceph_fastpool
```

7. 新しい層を使用して、新しいボリュームを作成します。

```
$ cinder create 1 --volume-type fast_tier --name fastdisk
```

6.3. CRUSH ルールが作成され、プールが正しい CRUSH ルールに設定されていることの確認

警告

既存の環境でパフォーマンス層を定義すると、Ceph クラスター内で大量のデータが移動する場合があります。スタックの更新時に director がトリガーする **ceph-ansible** には、プールがクラスターにすでに定義されているかどうかや、データが含まれるかどうかを確認するロジックはありません。つまり、プールに関連付けられたデフォルトの CRUSH ルールを変更すると、データの移動が行われるため、既存の環境でパフォーマンス層を定義することは危険となる可能性があります。ノードの追加または削除に関する支援または推奨事項が必要な場合は、Red Hat サポートにお問い合わせください。

手順

1. オーバークラウドコントローラーノードに **heat-admin** ユーザーとしてログインします。
2. OSD 層が正常に設定されていることを確認するには、以下のコマンドを入力します。**<controller_hostname>** を、ホストコントローラーノードの名前に置き換えます。

```
$ sudo podman exec -it ceph-mon-<controller_hostname> ceph osd tree
```

3. 作成されるツリービューで、各 OSD に設定したデバイスクラスが **CLASS** コラムに正しく表示されることを確認します。

4. また、以下のコマンドで、OSD がデバイスクラスに正しく割り当てられていることを確認します。**<controller_hostname>** を、ホストコントローラーノードの名前に置き換えます。

```
$ sudo podman exec -it ceph-mon-<controller_hostname> ceph osd crush tree --show-shadow
```

5. 作成された階層を以下のコマンドによる結果と比較し、ルールごとに同じ値が適用されることを確認します。

   - **<controller_hostname>** を、ホストコントローラーノードの名前に置き換えます。
   - **<rule_name>** を、チェックするルールの名前に置き換えます。

```
$ sudo podman exec <controller_hostname> ceph osd crush rule dump <rule_name>
```

6. 作成したルール名と ID が、デプロイメント中に使用した **crush_rules** パラメーターに準じて正しいことを確認します。**<controller_hostname>** を、ホストコントローラーノードの名前に置き換えます。

```
$ sudo podman exec -it ceph-mon-<controller_hostname>  ceph osd crush rule dump | grep -E "rule_(id|name)"
```

7. Ceph プールが、ステップ 3 で取得した正しい CRUSH ルール ID に関連付けられていることを確認します。**<controller_hostname>** を、ホストコントローラーノードの名前に置き換えます。

```
$ sudo podman exec -it ceph-mon-<controller_hostname> ceph osd dump | grep pool
```

8. 各プールについて、ルール ID が想定するルール名と一致することを確認してください。

第7章 オーバークラウドの作成

カスタム環境ファイルの作成が完了したら、それぞれのロールで使用するフレーバーおよびノードを指定し、続いてデプロイメントを実施することができます。以下のサブセクションで、両方の手順について詳細に説明します。

7.1. ロールへのノードとフレーバーの割り当て

オーバークラウドのデプロイメントプランニングでは、各ロールに割り当てるノード数とフレーバーを指定する必要があります。すべての Heat テンプレートのパラメーターと同様に、これらのロールの仕様は環境ファイル (ここでは **~/templates/storage-config.yaml**) の **parameter_defaults** セクションで宣言する必要があります。

この設定には、以下のパラメーターを使用します。

表7.1 オーバークラウドノードのロールとフレーバー

Heat テンプレートのパラメーター	説明
ControllerCount	スケールアウトするコントローラーノード数
OvercloudControlFlavor	コントローラーノードに使用するフレーバー (**control**)
ComputeCount	スケールアウトするコンピュートノード数
OvercloudComputeFlavor	コンピュートノード (**compute**) に使用するフレーバー
CephStorageCount	スケールアウトする Ceph Storage (OSD) ノード数
OvercloudCephStorageFlavor	Ceph Storage (OSD) ノード (**ceph-storage**) に使用するフレーバー
CephMonCount	スケールアウトする専用の Ceph MON ノード数
OvercloudCephMonFlavor	専用の Ceph MON ノード (**ceph-mon**) に使用するフレーバー
CephMdsCount	スケールアウトする専用の Ceph MDS ノード数
OvercloudCephMdsFlavor	専用の Ceph MDS ノード (**ceph-mds**) に使用するフレーバー

重要

(**ceph-mon** および **ceph-mds** フレーバーと共に)
CephMonCount、**CephMdsCount**、**OvercloudCephMonFlavor**、および **OvercloudCephMdsFlavor** のパラメーターは、*3章専用ノード上での Ceph サービスのデプロイ*に記載のように、カスタムの **CephMON** および **CephMds** ロールを作成した場合のみ有効となります。

たとえば、オーバークラウドが各ロール (Controller、Compute、Ceph-Storage、CephMon) に 3 つずつノードをデプロイするように設定するには、**parameter_defaults** に以下の設定を追加します。

```
parameter_defaults:
  ControllerCount: 3
  OvercloudControlFlavor: control
  ComputeCount: 3
  OvercloudComputeFlavor: compute
  CephStorageCount: 3
  OvercloudCephStorageFlavor: ceph-storage
  CephMonCount: 3
  OvercloudCephMonFlavor: ceph-mon
  CephMdsCount: 3
  OvercloudCephMdsFlavor: ceph-mds
```

注記

Heat テンプレートのパラメーターのより詳細なリストは、director のインストールと使用方法 ガイドの Creating the Overcloud with the CLI Tools を参照してください。

7.2. オーバークラウドデプロイメントの開始

注記

アンダークラウドのインストール時に、**undercloud.conf** ファイルに **generate_service_certificate=false** を設定します。設定しない場合は、**オーバークラウドの高度なカスタマイズ**ガイドの オーバークラウドのパブリックエンドポイントでの SSL/TLS の有効化 で説明したように、オーバークラウドのデプロイ時にトラストアンカーを挿入する必要があります。

注記

オーバークラウドのデプロイメント中に Ceph Dashboard を追加する場合は、*8章Red Hat Ceph Storage Dashboard のオーバークラウドデプロイメントへの追加*を参照してください。

オーバークラウドの作成には、**openstack overcloud deploy** コマンドに追加の引数を指定する必要があります。以下に例を示します。

```
$ openstack overcloud deploy --templates -r /home/stack/templates/roles_data_custom.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-rgw.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-mds.yaml
  -e /usr/share/openstack-tripleo-heat-templates/environments/cinder-backup.yaml \
  -e /home/stack/templates/storage-config.yaml \
  -e /home/stack/templates/ceph-config.yaml \
  --ntp-server pool.ntp.org
```

-

上記のコマンドは、以下のオプションを使用します。

- **--templates**: デフォルトの Heat テンプレートコレクション (**/usr/share/openstack-tripleo-heat-templates/**) からオーバークラウドを作成します。
- **-r /home/stack/templates/roles_data_custom.yaml**:3章*専用ノード上での Ceph サービスのデプロイ*でカスタマイズしたロールの定義ファイルを指定し、カスタムロールを Ceph MON サービスまたは Ceph MDS サービスに追加します。これらのロールにより、いずれかのサービスを専用のノードにインストールすることができます。
- **-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml**: Ceph クラスターを作成するように director を設定します。この環境ファイルは、特に **コンテナー化された** Ceph Storage ノードを持つ Ceph クラスターをデプロイします。
- **-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-rgw.yaml**:「Ceph Object Gateway の有効化」 で説明するように、Ceph Object Gateway を有効にします。
- **-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-mds.yaml**:「Ceph Metadata Server の有効化」 で説明するように、Ceph Metadata Server を有効にします。
- **-e /usr/share/openstack-tripleo-heat-templates/environments/cinder-backup.yaml**:「バックアップサービスで Ceph を使用する設定」で説明するように、Block Storage Backup サービス (**cinder-backup**) を有効にします。
- **-e /home/stack/templates/storage-config.yaml**: Ceph Storage のカスタム設定が含まれる環境ファイルを追加します。
- **-e /home/stack/templates/ceph-config.yaml**:5章*Ceph Storage クラスターのカスタマイズ*で説明するように、Ceph クラスターのカスタム設定が含まれる環境ファイルを追加します。
- **--ntp-server pool.ntp.org**: NTP サーバーを設定します。

ヒント

アンサーファイル を使用して、すべてのテンプレートおよび環境ファイルを呼び出すこともできます。たとえば、以下のコマンドを使用して、同一のオーバークラウドをデプロイすることができます。

```
$ openstack overcloud deploy -r /home/stack/templates/roles_data_custom.yaml \
  --answers-file /home/stack/templates/answers.yaml --ntp-server pool.ntp.org
```

この場合、アンサーファイル **/home/stack/templates/answers.yaml** の内容は以下のようになります。

```
templates: /usr/share/openstack-tripleo-heat-templates/
environments:
  - /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml
  - /usr/share/openstack-tripleo-heat-templates/environments/ceph-rgw.yaml
  - /usr/share/openstack-tripleo-heat-templates/environments/ceph-mds.yaml
  - /usr/share/openstack-tripleo-heat-templates/environments/cinder-backup.yaml
  - /home/stack/templates/storage-config.yaml
  - /home/stack/templates/ceph-config.yaml
```

詳細は、オーバークラウドデプロイメントへの環境ファイルの追加 を参照してください。

オプションの完全なリストを表示するには、以下のコマンドを入力します。

```
$ openstack help overcloud deploy
```

詳細は、**director のインストールと使用方法**ガイドの CLI ツールを使用した基本的なオーバークラウドの設定 を参照してください。

オーバークラウドの作成プロセスが開始され、director によりノードがプロビジョニングされます。このプロセスは完了するまで多少時間がかかります。オーバークラウドの作成のステータスを確認するには、**stack** ユーザーとして別のターミナルを開き、以下のコマンドを入力します。

```
$ source ~/stackrc
$ openstack stack list --nested
```

7.2.1. ceph-ansible を実行するノードの限定

ceph-ansible を実行するノードを限定することで、デプロイメントの更新時間を短縮することができます。Red Hat OpenStack Platform (RHOSP) で Ceph の設定に **config-download** が使用されている場合、デプロイメント全体に対して **config-download** および **ceph-ansible** を実行する代わりに、**--limit** オプションを使用してノードのリストを指定することができます。この機能は、たとえばオーバークラウドをスケールアップする場合や障害の発生したディスクを置き換える場合に役立ちます。このようなシナリオでは、環境に追加する新規ノードでのみデプロイメントを実行することができます。

障害の発生したディスクの置き換え時に --limit を使用するシナリオの例

以下の手順例では、Ceph ストレージノード **oc0-cephstorage-0** でディスク障害が発生したため、ベンダーでフォーマット済みの新規ディスクを受け入れます。新しいディスクを OSD として使用することができるように、Ansible を **oc0-cephstorage-0** ノード上で実行する必要があります。しかし、その他すべての Ceph Storage ノードで実行する必要はありません。例として記述した環境ファイルおよびノードの名前を、実際の環境に適した名前に置き換えてください。

手順

1. アンダークラウドノードに **stack** ユーザーとしてログインし、source コマンドで **stackrc** 認証情報ファイルを読み込みます。

   ```
   # source stackrc
   ```

2. 新規ディスクが不足している OSD を起動するのに使用されるように、以下の手順の 1 つを実施します。

   - **--limit** オプションを使用して **ceph-ansible** を実行するノードを指定し、スタックの更新を実行する。

     ```
     $ openstack overcloud deploy --templates \
       -r /home/stack/roles_data.yaml \
       -n /usr/share/openstack-tripleo-heat-templates/network_data_dashboard.yaml \
       -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
     ansible.yaml \
       -e ~/my-ceph-settings.yaml \
       -e <other-environment_files> \
       --limit oc0-controller-0:oc0-controller-2:oc0-controller-1:oc0-cephstorage-0:undercloud
     ```

この例では、Ceph mon の OSD 定義を変更するのに Ansible が必要なため、コントローラーが含まれています。

- **config-download** が **ansible-playbook-command.sh** スクリプトを生成した場合は、**--limit** オプションを指定してスクリプトを実行し、指定されたノードを **ceph-ansible** に渡すこともできます。

```
./ansible-playbook-command.sh --limit oc0-controller-0:oc0-controller-2:oc0-controller-1:oc0-cephstorage-0:undercloud
```

Warning

必ずアンダークラウドを制限リストに含める必要があります。含めないと、**--limit** を使用する際に **ceph-ansible** を実行することができません。この注意が必要なのは、アンダークラウドでのみ実行される **external_deploy_steps_tasks** Playbook により **ceph-ansible** が実行されるためです。

第8章 RED HAT CEPH STORAGE DASHBOARD のオーバークラウドデプロイメントへの追加

Red Hat Ceph Storage Dashboard はデフォルトで無効になっていますが、Red Hat OpenStack Platform director を使用してオーバークラウドで有効にすることができます。Ceph Dashboard は組み込みの Web ベースの Ceph 管理および監視アプリケーションであり、クラスター内のさまざまな側面およびオブジェクトを管理します。Red Hat Ceph Storage Dashboard は、以下のコンポーネントで設定されています。

- Ceph Dashboard Manager モジュール: ユーザーインターフェイスを提供し、プラットフォームフロントエンド Grafana が組み込まれています。
- Prometheus: モニタリングプラグイン
- Alertmanager: アラートを Dashboard に送信します。
- Node Exporters: クラスターデータを Dashboard にエクスポートします。

注記

この機能は、Ceph Storage 4.1 以降でサポートされます。システムにインストールされている Ceph Storage のバージョンを確認する方法についての詳細は、What are the Red Hat Ceph Storage releases and corresponding Ceph package versions? を参照してください。

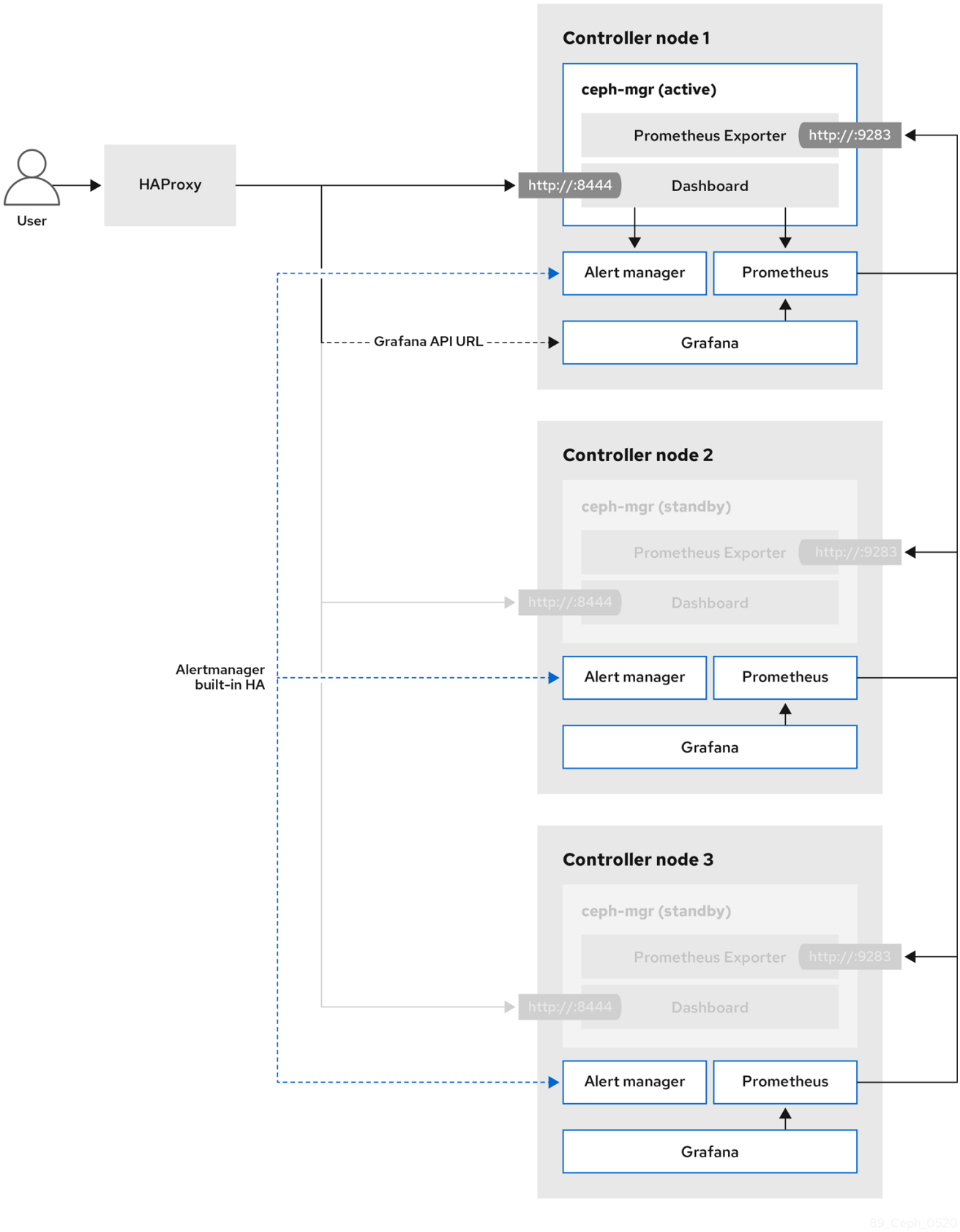
注記

Red Hat Ceph Storage Dashboard は、常に他の Ceph Manager コンポーネントと同じノード上に配置されます。

注記

オーバークラウドの初回のデプロイメント時に Ceph Dashboard を追加する場合は、本章の手順を実施してから、「オーバークラウドデプロイメントの開始」で初期オーバークラウドをデプロイします。

以下の図は、Red Hat OpenStack Platform 上の Ceph Dashboard のアーキテクチャーを示しています。



Dashboard およびその機能と制限についての詳細は、**Red Hat Ceph Storage Dashboard Guide**の Dashboard features を参照してください。

Ceph Dashboard における TLS everywhere

Dashboard フロントエンドは、TLS everywhere フレームワークと完全に統合されています。必要な環境ファイルがあり、そのファイルが overcloud deploy コマンドに含まれている場合は、TLS

everywhere を有効にすることができます。これにより、Grafana と Ceph Dashboard の両方の証明書要求がトリガーされ、生成された証明書とキーファイルがオーバークラウドのデプロイメント時に **ceph-ansible** に渡されます。Dashboard およびその他の openstack サービス向けに TLS を有効にする方法についての説明および詳細は、**オーバークラウドの高度なカスタマイズ**ガイドの以下の章を参照してください。

- オーバークラウドのパブリックエンドポイントでの SSL/TLS の有効化
- Identity Management を使用した内部およびパブリックエンドポイントでの SSL/TLS の有効化

注記

Ceph Dashboard に到達するポートは、TLS-everywhere コンテキストでも同じになります。

8.1. CEPH DASHBOARD に必要なコンテナーの追加

Ceph Dashboard テンプレートをオーバークラウドに追加する前に、**containers-prepare-parameter.yaml** ファイルを使用して必要なコンテナーを追加する必要があります。コンテナーイメージを準備するために **containers-prepare-parameter.yaml** ファイルを生成するには、以下の手順を実行します。

手順

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. デフォルトのコンテナーイメージ準備ファイルを生成します。

```
$ sudo openstack tripleo container image prepare default \
  --local-push-destination \
  --output-env-file containers-prepare-parameter.yaml
```

3. **containers-prepare-parameter.yaml** ファイルを編集し、要件に合わせて変更を加えます。以下の **containers-prepare-parameter.yaml** ファイルのサンプルには、Grafana、Prometheus、Alertmanager、および Node Exporter などの Dashboard サービスに関連するイメージの場所およびタグが含まれます。特定のシナリオに応じて値を編集します。

```
parameter_defaults:
    ContainerImagePrepare:
    - push_destination: true
        set:
            ceph_alertmanager_image: ose-prometheus-alertmanager
            ceph_alertmanager_namespace: registry.redhat.io/openshift4
            ceph_alertmanager_tag: v4.1
            ceph_grafana_image: rhceph-4-dashboard-rhel8
            ceph_grafana_namespace: registry.redhat.io/rhceph
            ceph_grafana_tag: 4
            ceph_image: rhceph-4-rhel8
            ceph_namespace: registry.redhat.io/rhceph
            ceph_node_exporter_image: ose-prometheus-node-exporter
            ceph_node_exporter_namespace: registry.redhat.io/openshift4
            ceph_node_exporter_tag: v4.1
            ceph_prometheus_image: ose-prometheus
            ceph_prometheus_namespace: registry.redhat.io/openshift4
            ceph_prometheus_tag: v4.1
            ceph_tag: latest
```

containers-prepare-parameter.yaml ファイルを使用したレジストリーおよびイメージ設定についての詳細は、**Transitioning to Containerized Services**ガイドの Container image preparation parameters を参照してください。

8.2. CEPH DASHBOARD のデプロイ

注記

コンポーザブルネットワークを使用して Ceph Dashboard をデプロイする場合は、「コンポーザブルネットワークを使用した Ceph Dashboard のデプロイ」を参照してください。

注記

Ceph Dashboard の管理ユーザーロールは、デフォルトで読み取り専用モードに設定されています。Ceph Dashboard のデフォルトの管理モードを変更するには、「デフォルト権限の変更」を参照してください。

手順

1. アンダークラウドノードに **stack** ユーザーとしてログインします。
2. **openstack overcloud deploy** コマンドに、デプロイメントに含まれるすべての環境ファイルと共に以下の環境ファイルを追加します。

```
$ openstack overcloud deploy \
 --templates \
 -e <overcloud_environment_files> \
 -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml \
 -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-dashboard.yaml
```

<overcloud_environment_files> をデプロイメントに追加する環境ファイルのリストに置き換えます。

結果

これにより、デプロイメントは grafana、prometheus、alertmanager、および node-exporter コンテナーが含まれる外部スタックを設定します。Ceph Dashboard Manager モジュールは、このスタックのバックエンドで、grafana レイアウトを組み込むことで、Ceph クラスター固有のメトリックをエンドユーザーに提供します。

8.3. コンポーザブルネットワークを使用した CEPH DASHBOARD のデプロイ

デフォルトのプロビジョニングネットワークではなく、コンポーザブルネットワークに Ceph Dashboard をデプロイすることができます。これにより、プロビジョニングネットワークに Ceph Dashboard サービスを公開する必要がなくなります。Dashboard をコンポーザブルネットワークにデプロイする際に、別個の承認プロファイルを実装することもできます。

最初にオーバークラウドをデプロイする場合にのみ Dashboard を新規ネットワークに適用することができるので、デプロイ前に使用するネットワークを選択する必要があります。Dashboard を既存の外部ネットワークに適用することや、プロビジョニングネットワーク以外の既存のネットワークのいずれかを再利用することはできません。デプロイ前にコンポーザブルネットワークを選択するには、以下の手順を使用します。

手順

1. アンダークラウドに stack ユーザーとしてログインします。
2. Controller 固有のロールを生成して、Dashboard コンポーザブルネットワークを追加します。

```
$ openstack overcloud roles generate -o /home/stack/roles_data_dashboard.yaml
ControllerStorageDashboard Compute BlockStorage ObjectStorage CephStorage
```

結果

- 新しい **ControllerStorageDashboard** ロールは、コマンドの出力として定義される **roles_data.yaml** 内に生成されます。overcloud deploy コマンドを使用する場合には、このファイルをテンプレートリストに含める必要があります。
  注: ControllerStorageDashboard ロールには、**CephNFS** も **network_data_dashboard.yaml** も含まれていません。
- director は、コンポーザブルネットワークを定義するネットワーク環境ファイルを提供します。このファイルのデフォルトの場所は、**/usr/share/openstack-tripleo-heat-templates/network_data_dashboard.yaml** です。overcloud deploy コマンドを使用する場合には、このファイルをオーバークラウドのテンプレートリストに含める必要があります。

3. **openstack overcloud deploy** コマンドに、デプロイメントに含まれるすべての環境ファイルと共に以下の環境ファイルを追加します。

```
$ openstack overcloud deploy \
  --templates \
  -r /home/stack/roles_data.yaml \
  -n /usr/share/openstack-tripleo-heat-templates/network_data_dashboard.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
  -e <overcloud_environment_files> \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
dashboard.yaml
```

<overcloud_environment_files> をデプロイメントに追加する環境ファイルのリストに置き換えます。

結果

これにより、デプロイメントは grafana、prometheus、alertmanager、および node-exporter コンテナーが含まれる外部スタックを設定します。Ceph Dashboard Manager モジュールは、このスタックのバックエンドで、grafana レイアウトを組み込むことで、Ceph クラスター固有のメトリックをエンドユーザーに提供します。

8.4. デフォルト権限の変更

Ceph クラスターの安全な監視用に、Ceph Dashboard の管理ユーザーロールはデフォルトで読み取り専用モードに設定されます。管理ユーザーが Dashboard を使用して Ceph クラスターの要素を変更できるよう管理者権限を昇格させるには、**CephDashboardAdminRO** パラメーターを使用してデフォルトの管理者権限を変更します。

警告

完全な権限を持つユーザーは、director が設定するクラスターの要素を変更する可能性があります。これは、スタックの更新の実行時に、director が設定したオプションと競合する可能性があります。この問題を回避するには、Ceph Dashboard で director の設定オプション (Ceph OSP プール属性など) を変更しないでください。

手順

1. アンダークラウドに **stack** ユーザーとしてログインします。
2. 以下の **ceph_dashboard_admin.yaml** 環境ファイルを作成します。

```
parameter_defaults:
  CephDashboardAdminRO: false
```

3. overcloud deploy コマンドを実行して、既存のスタックを更新し、既存のデプロイメントに含まれるその他すべての環境ファイルと共に作成した環境ファイルを追加します。

```
$ openstack overcloud deploy \
--templates \
-e <existing_overcloud_environment_files> \
-e ceph_dashboard_admin.yml
```

<existing_overcloud_environment_files> を既存のデプロイメントに含まれる環境ファイルのリストに置き換えます。

8.5. CEPH DASHBOARD へのアクセス

Ceph Dashboard が正常に実行されていることを確認するには、以下の検証手順を実施してアクセスし、Ceph クラスターから表示されるデータが正しいことを確認します。

手順

1. アンダークラウドノードに **stack** ユーザーとしてログインします。
2. Dashboard の管理ログイン認証情報を取得します。

```
[stack@undercloud ~]$ grep dashboard_admin_password /var/lib/mistral/overcloud/ceph-ansible/group_vars/all.yml
```

3. Ceph Dashboard にアクセスするための仮想 IP アドレスを取得します。

```
[stack@undercloud-0 ~]$ grep dashboard_frontend_vip /var/lib/mistral/overcloud/ceph-ansible/group_vars/all.yml
```

4. Web ブラウザーを使用してフロントエンドの仮想 IP をポイントし、Dashboard にアクセスします。director はプロビジョニングネットワーク上で Dashboard を設定して公開するので、取得した仮想 IP を使用して、TCP ポート 8444 の Dashboard に直接アクセスできます。以下の条件を満たしていることを確認します。
   - Web クライアントホストは、プロビジョニングネットワークに接続されたレイヤー 2 です。

- プロビジョニングネットワークは適切にルーティングまたはプロキシーされ、Web クライアントホストからアクセスできます。これらの条件が満たされていない場合は、SSH トンネルを開いて、オーバークラウド上の Dashboard の仮想 IP に到達することができます。

  ```
  client_host$ ssh -L 8444:<dashboard_vip>:8444 stack@<your undercloud>
  ```

  <dashboard_vip> を取得したコントロールプレーンの仮想 IP の IP アドレスに置き換えます。

5. Dashboard にアクセスするには、Web ブラウザーで http://localhost:8444 にアクセスし、以下の詳細でログインします。

   - **ceph-ansible** が作成するデフォルトのユーザー: **admin**
   - **/var/lib/mistral/overcloud/ceph-ansible/group_vars/all.yml** のパスワード

   結果

   - Ceph Dashboard にアクセスできます。
   - Dashboard に表示される数字およびグラフには、CLI コマンドの **ceph -s** が返すのと同じクラスターステータスが反映されます。

Red Hat Ceph Storage Dashboard に関する詳細は、Red Hat Ceph Storage Dashboard Guideを参照してください。

第9章 デプロイメント後

以下のサブセクションでは、Ceph クラスターを管理するためのデプロイメント後の操作についていくつか説明します。

9.1. オーバークラウドへのアクセス

director は、アンダークラウドからオーバークラウドと対話するための設定を行い、認証をサポートするスクリプトを作成します。director は、このファイル (**overcloudrc**) を **stack** ユーザーのホームディレクトリーに保存します。このファイルを使用するには、以下のコマンドを実行します。

```
$ source ~/overcloudrc
```

これにより、アンダークラウド CLI からオーバークラウドと対話するために必要な環境変数が読み込まれます。アンダークラウドとの対話に戻るには、以下のコマンドを実行します。

```
$ source ~/stackrc
```

9.2. CEPH STORAGE ノードの監視

オーバークラウドを作成したら、Ceph Storage クラスターのステータスをチェックして、正常に機能していることを確認します。

手順

1. コントローラーノードに **heat-admin** ユーザーとしてログインします。

   ```
   $ nova list
   $ ssh heat-admin@192.168.0.25
   ```

2. クラスターの正常性を確認します。

   ```
   $ sudo podman exec ceph-mon-<HOSTNAME> ceph health
   ```

   クラスターに問題がない場合は、上記のコマンドにより、**HEALTH_OK** というレポートが返されます。これは、クラスターを安全に使用できることを意味します。

3. Ceph monitor サービスを実行するオーバークラウドノードにログインし、クラスター内の全 OSD のステータスを確認します。

   ```
   $ sudo podman exec ceph-mon-<HOSTNAME> ceph osd tree
   ```

4. Ceph Monitor クォーラムのステータスを確認します。

   ```
   $ sudo podman exec ceph-mon-<HOSTNAME> ceph quorum_status
   ```

   これにより、クォーラムに参加するモニターとどれがリーダーであるかが表示されます。

5. すべての Ceph OSD が動作中であることを確認します。

   ```
   $ sudo podman exec ceph-mon-<HOSTNAME> ceph osd stat
   ```

Ceph Storage Cluster の監視に関する詳細は、**Red Hat Ceph Storage Administration Guide**の Monitoring を参照してください。

第10章 環境のリブート

環境をリブートする必要がある状況が発生する場合があります。たとえば、物理サーバーを変更する必要がある場合や、停電から復旧する必要がある場合などです。このような状況では、Ceph Storage ノードが正常に起動されることが重要です。

以下の順序でノードを起動してください。

- **すべての Ceph Monitor ノードを最初に起動します**: これにより、高可用性クラスター内の Ceph Monitor サービスを確実にアクティブにします。デフォルトでは、Ceph Monitor サービスは、コントローラーノードにインストールされます。Ceph Monitor がカスタムロールで Controller とは別の場合には、このカスタムの Ceph Monitor ロールを必ずアクティブにしてください。
- **すべての Ceph Storage ノードを起動します**: これにより、Ceph OSD クラスターはコントローラーノード上のアクティブな Ceph Monitor クラスターに接続できるようになります。

10.1. CEPH STORAGE (OSD) クラスターのリブート

Ceph Storage (OSD) ノードのクラスターを再起動するには、以下の手順を実施します。

前提条件

- **ceph-mon** サービスを実行している Ceph Monitor または Controller ノードで、Red Hat Ceph Storage クラスターのステータスが正常であり、pg ステータスが **active+clean** であることを確認する。

```
$ sudo podman exec -it ceph-mon-controller-0 ceph -s
```

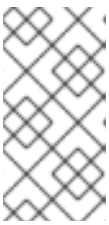
Ceph クラスターが正常な場合、**HEALTH_OK** のステータスが返されます。

Ceph クラスターのステータスが異常な場合、**HEALTH_WARN** または **HEALTH_ERR** のステータスを返す。トラブルシューティングのガイダンスについては、Red Hat Ceph Storage 4 トラブルシューティングガイドを 参照してください。

手順

1. **ceph-mon** サービスを実行している Ceph Monitor または Controller ノードにログインし、Ceph Storage クラスターのリバランスを一時的に無効にします。

```
$ sudo podman exec -it ceph-mon-controller-0 ceph osd set noout
$ sudo podman exec -it ceph-mon-controller-0 ceph osd set norebalance
```



注記

マルチスタックまたは分散コンピュートノード (DCN) アーキテクチャーを使用している場合は、 **noout**フラグと**norebalance**フラグの設定時にクラスター名を指定する必要があります。例: **sudo podman exec -it ceph-mon-controller-0 ceph osd set noout --cluster <cluster_name>**

2. 再起動する最初の Ceph Storage ノードを選択し、そのノードにログインします。
3. ノードを再起動します。

```
$ sudo reboot
```

4. ノードがブートするまで待ちます。

5. ノードにログインして、クラスターのステータスを確認します。

```
$ sudo podman exec -it ceph-mon-controller-0 ceph status
```

pgmap により、すべての **pgs** が正常な状態 (**active+clean**) として報告されることを確認します。

6. ノードからログアウトして、次のノードをリブートし、ステータスを確認します。全 Ceph Storage ノードがリブートされるまで、このプロセスを繰り返します。

7. 完了したら、**ceph-mon** サービスを実行している Ceph Monitor または Controller ノードにログインし、クラスターの再調整を再度有効にします。

```
$ sudo podman exec -it ceph-mon-controller-0 ceph osd unset noout
$ sudo podman exec -it ceph-mon-controller-0 ceph osd unset norebalance
```

注記

マルチスタックまたは分散コンピュートノード (DCN) アーキテクチャーを使用している場合は、 **noout**フラグと**norebalance**フラグの設定解除時にクラスター名を指定する必要があります。例: **sudo podman exec -it ceph-mon-controller-0 ceph osd set noout --cluster <cluster_name>**

8. 最終のステータスチェックを実行して、クラスターが **HEALTH_OK** を報告していることを確認します。

```
$ sudo podman exec -it ceph-mon-controller-0 ceph status
```

すべてのオーバークラウドノードが同時に起動する状況が発生した場合には、Ceph OSD サービスが Ceph Storage ノード上で正常に起動されない場合があります。そのような場合には、Ceph Storage OSD をリブートして、Ceph Monitor サービスに接続できるようにします。

以下のコマンドを使用して、Ceph Storage ノードクラスターの **HEALTH_OK** のステータスを確認します。

```
$ sudo ceph status
```

第11章 CEPH STORAGE クラスターのスケーリング

11.1. CEPH STORAGE クラスターのスケールアップ

必要な Ceph Storage ノードの数でデプロイメントを再度実行して、オーバークラウド内の Ceph Storage ノードの数をスケールアップすることができます。

この操作を実行する前に、更新するデプロイメント用に十分なノードがあることを確認してください。これらのノードは、director に登録済みで、適宜にタグ付けされている必要があります。

新規 Ceph Storage ノードの登録

新しい Ceph Storage ノードを director に登録するには、以下の手順を実施します。

手順

1. アンダークラウドに **stack** ユーザーとしてログインし、director の設定を初期化します。

```
$ source ~/stackrc
```

2. 新規ノードの定義テンプレート (例: **instackenv-scale.json**) で、新しいノードのハードウェアと電源管理の詳細を定義します。

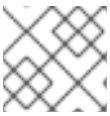
3. このファイルを director にインポートします。

```
$ openstack overcloud node import ~/instackenv-scale.json
```

ノードの定義テンプレートをインポートすると、そのテンプレートで定義されている各ノードが director に登録されます。

4. カーネルと ramdisk イメージを全ノードに割り当てます。

```
$ openstack overcloud node configure
```



注記

新規ノードの登録に関する詳しい情報は、「ノードの登録」を参照してください。

手動による新規ノードのタグ付け

各ノードの登録後、ハードウェアを検査して、ノードを特定のプロファイルにタグ付けする必要があります。プロファイルタグを使用してノードをフレーバーに照合してから、フレーバーをデプロイメントロールに割り当てます。

手順

1. ハードウェアのイントロスペクションをトリガーして、各ノードのハードウェア属性を取得します。

```
$ openstack overcloud node introspect --all-manageable --provide
```

- **--all-manageable** オプションを使用して、管理状態にあるノードのみをイントロスペクションします。ここでは、すべてのノードが管理状態にあります。

- **--provide** オプションは、イントロスペクション後に全ノードを **active** の状態にリセットします。

重要

このプロセスが正常に完了したことを確認します。ベアメタルノードの場合には、通常 15 分ほどかかります。

2. ノードリストを取得して UUID を把握します。

```
$ openstack baremetal node list
```

3. 各ノードの **properties/capabilities** パラメーターに profile オプションを追加して、ノードを特定のプロファイルに手動でタグ付けします。**profile** オプションを追加すると、適切なプロファイルにノードをタグ付けします。

注記

手動でのタグ付けの代わりに、Automated Health Check (AHC) ツールを使用し、ベンチマークデータに基づいて、多数のノードに自動でタグ付けします。たとえば、以下のコマンドは、3 つの追加のノードを **ceph-storage** プロファイルでタグ付けします。

```
$ openstack baremetal node set --property capabilities='profile:baremetal,boot_option:local'
551d81f5-4df2-4e0f-93da-6c5de0b868f7
$ openstack baremetal node set --property capabilities='profile:baremetal,boot_option:local'
5e735154-bd6b-42dd-9cc2-b6195c4196d7
$ openstack baremetal node set --property capabilities='profile:baremetal,boot_option:local'
1a2b090c-299d-4c20-a25d-57dd21a7085b
```

ヒント

タグ付け/登録したノードが複数のディスクを使用している場合には、director が各ノードで特定のルートディスクを使用するように設定できます。詳細は、「マルチディスククラスターのルートディスクの定義」 を参照してください。

Ceph Storage ノードを追加したオーバークラウドの再デプロイ

新規ノードを登録してタグ付けした後に、オーバークラウドを再デプロイして Ceph Storage ノードの数をスケールアップすることができます。

手順

1. オーバークラウドを再デプロイする前に、環境ファイル (ここでは ~/**templates/storage-config.yaml**) の **parameter_defaults** にある **CephStorageCount** パラメーターを設定します。「ロールへのノードとフレーバーの割り当て」では、オーバークラウドは 3 つの Ceph Storage ノードでデプロイするように設定されています。以下の例では、オーバークラウドを 6 つのノードにスケーリングしています。

```
parameter_defaults:
  ControllerCount: 3
  OvercloudControlFlavor: control
  ComputeCount: 3
```

```
OvercloudComputeFlavor: compute
CephStorageCount: 6
OvercloudCephStorageFlavor: ceph-storage
CephMonCount: 3
OvercloudCephMonFlavor: ceph-mon
```

2. オーバークラウドを再デプロイします。オーバークラウドには、3 つの Ceph Storage ノードではなく 6 つの Ceph Storage ノードが含まれるようになりました。

11.2. CEPH STORAGE ノードのスケールダウンと置き換え

場合によっては、Ceph クラスターのスケールダウン、または Ceph Storage ノードのを置き換え (Ceph Storage ノードに問題がある場合など) が必要となる可能性があります。いずれの場合も、データの損失を避けるために、オーバークラウドから削除する Ceph Storage ノードを無効にしてリバランスする必要があります。

注記

以下の手順では、**Red Hat Ceph Storage Administration Guide**からのステップを使用して、手動で Ceph Storage ノードを削除します。Ceph Storage ノードの手動削除に関する詳細は、コンテナー内で実行される Ceph デーモンの開始、停止、および再起動 および コマンドラインインターフェイスを使用した Ceph OSD の削除 を参照してください。

手順

1. コントローラーノードに **heat-admin** ユーザーとしてログインします。director の **stack** ユーザーには、**heat-admin** ユーザーにアクセスするための SSH キーがあります。

2. OSD ツリーをリスト表示して、お使いのノードの OSD を検索します。たとえば、削除するノードには、以下の OSD が含まれる場合があります。

```
-2 0.09998     host overcloud-cephstorage-0
0 0.04999        osd.0                 up  1.00000          1.00000
1 0.04999        osd.1                 up  1.00000          1.00000
```

3. Ceph Storage ノードの OSD を無効化します。今回は、OSD ID は 0 と 1 です。

```
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph osd out 0
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph osd out 1
```

4. Ceph Storage クラスターがリバランスを開始します。このプロセスが完了するまで待機してください。以下のコマンドを使用して、ステータスを確認できます。

```
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph -w
```

5. Ceph クラスターのリバランスが完了したら、削除する Ceph Storage ノード (ここでは **overcloud-cephstorage-0**) に **heat-admin** ユーザーとしてログインし、ノードを停止し無効にします。

```
[heat-admin@overcloud-cephstorage-0 ~]$ sudo systemctl stop ceph-osd@0
[heat-admin@overcloud-cephstorage-0 ~]$ sudo systemctl stop ceph-osd@1
[heat-admin@overcloud-cephstorage-0 ~]$ sudo systemctl disable ceph-osd@0
[heat-admin@overcloud-cephstorage-0 ~]$ sudo systemctl disable ceph-osd@1
```

6. OSD を停止します。

```
[heat-admin@overcloud-cephstorage-0 ~]$ sudo systemctl stop ceph-osd@0
[heat-admin@overcloud-cephstorage-0 ~]$ sudo systemctl stop ceph-osd@1
```

7. コントローラーノードにログインしたら、CRUSH マップから OSD を削除して、データを受信しないようにします。

```
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph osd crush remove osd.0
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph osd crush remove osd.1
```

8. OSD 認証キーを削除します。

```
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph auth del osd.0
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph auth del osd.1
```

9. クラスターから OSD を削除します。

```
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph osd rm 0
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph osd rm 1
```

10. CRUSH マップからストレージノードを削除します。

```
[heat-admin@overcloud-controller-0 ~]$ sudo docker exec ceph-mon-<HOSTNAME> ceph osd crush rm <NODE>
[heat-admin@overcloud-controller-0 ~]$ sudo ceph osd crush remove <NODE>
```

CRUSH ツリーを検索して、CRUSH マップに定義されている <NODE> の名前を確認できます。

```
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph osd crush tree | grep overcloud-osd-compute-3 -A 4
            "name": "overcloud-osd-compute-3",
            "type": "host",
            "type_id": 1,
            "items": []
        },
[heat-admin@overcloud-controller-0 ~]$
```

CRUSH ツリーで、アイテムリストが空であることを確認します。リストが空でない場合は、ステップ 7 を再度実施してください。

11. ノードからログアウトして、**stack** ユーザーとしてアンダークラウドに戻ります。

```
[heat-admin@overcloud-controller-0 ~]$ exit
[stack@director ~]$
```

12. director が再度プロビジョニングしないように、Ceph Storage ノードを無効にします。

```
[stack@director ~]$ openstack baremetal node list
[stack@director ~]$ openstack baremetal node maintenance set UUID
```

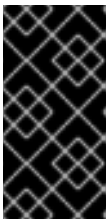
13. Ceph Storage ノードを削除するには、ローカルのテンプレートファイルを使用して、director の **overcloud** スタックへの更新が必要です。最初にオーバークラウドスタックの UUID を特定します。

```
$ openstack stack list
```

14. 削除する Ceph Storage ノードの UUID を特定します。

```
$ openstack server list
```

15. スタックからノードを削除し、それに応じてプランを更新します。



重要

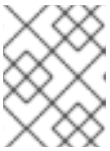
オーバークラウドの作成時に追加の環境ファイルを渡した場合には、予定外の変更がオーバークラウドに加えられないように、ここで **-e** オプションを使用して環境ファイルを再度渡します。詳しい情報は、**director のインストールと使用方法**ガイドの オーバークラウド環境の変更 を参照してください。

```
$ openstack overcloud node delete /
--stack <stack-name> /
--templates /
-e <other-environment-files> /
<node_UUID>
```

16. stack が更新を完了するまで待機します。**heat stack-list --show-nested** コマンドを使用して、stack の更新を監視します。

17. 新規ノードを director のノードプールに追加して、Ceph Storage ノードとしてデプロイします。環境ファイル (ここでは **~/templates/storage-config.yaml**) の **parameter_defaults** の **CephStorageCount** パラメーターを使用して、オーバークラウド内の Ceph Storage ノードの合計数を定義します。

```
parameter_defaults:
  ControllerCount: 3
  OvercloudControlFlavor: control
  ComputeCount: 3
  OvercloudComputeFlavor: compute
  CephStorageCount: 3
  OvercloudCephStorageFlavor: ceph-storage
  CephMonCount: 3
  OvercloudCephMonFlavor: ceph-mon
```



注記

ロールごとのノード数を定義する方法の詳細は、「ロールへのノードとフレーバーの割り当て」を参照してください。

18. 環境ファイルを更新したら、オーバークラウドを再デプロイします。

```
$ openstack overcloud deploy --templates -e <ENVIRONMENT_FILE>
```

director は、新しいノードをプロビジョニングし、新しいノードの詳細を用いて stack 全体を更新します。

19. **heat-admin** ユーザーとしてコントローラーノードにログインし、Ceph Storage ノードのステータスを確認します。

```
[heat-admin@overcloud-controller-0 ~]$ sudo ceph status
```

20. **osdmap** セクションの値がクラスターで必要なノード数と一致していることを確認します。削除した Ceph Storage ノードは新規ノードに置き換えられます。

11.3. OSD の CEPH STORAGE ノードへの追加

この手順では、OSD をノードに追加する方法を説明します。Ceph OSD に関する詳細は、**Red Hat Ceph Storage Operations Guide**の Ceph OSDs を参照してください。

手順

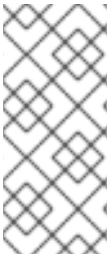
1. 以下の heat テンプレートは、OSD デバイスを 3 つ持つ Ceph Storage をデプロイします。

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sdb
      - /dev/sdc
      - /dev/sdd
    osd_scenario: lvm
    osd_objectstore: bluestore
```

2. OSD を追加するには、「Ceph Storage ノードのディスクレイアウトのマッピング」の説明に従って、ノードのディスクレイアウトを更新します。以下の例では、**/dev/sde** をテンプレートに追加します。

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sdb
      - /dev/sdc
      - /dev/sdd
      - /dev/sde
    osd_scenario: lvm
    osd_objectstore: bluestore
```

3. **openstack overcloud deploy** を実行してオーバークラウドを更新します。



注記

この例では、OSD を持つすべてのホストに、**/dev/sde** という新しいデバイスがあります。すべてのノードに新規デバイスを持たせる必要がない場合は、heat テンプレートを更新します。異なる **devices** リストを使用してホストを定義する方法については、「異なる Ceph Storage ノードのパラメーターのオーバーライド」 と 「Ceph Storage ノードのディスクレイアウトの変更」 を参照してください。

11.4. CEPH STORAGE ノードからの OSD の削除

この手順では、ノードから OSD を削除する方法を説明します。環境について以下を前提とします。

- サーバー (**ceph-storage0**) には、**/dev/sde** で実行している OSD (**ceph-osd@4**) がある。
- Ceph monitor サービス (**ceph-mon**) が **controller0** で実行されている。
- ストレージクラスターの割合がほぼ完全とならないように、利用可能な OSD が十分にある。

Ceph OSD に関する詳細は、**Red Hat Ceph Storage Operations Guide** の Ceph OSDs を参照してください。

手順

1. **ceph-storage0** に SSH 接続し、 **root** でログインします。
2. OSD サービスを無効にし、停止します。

```
[root@ceph-storage0 ~]# systemctl disable ceph-osd@4
[root@ceph-stoarge0 ~]# systemctl stop ceph-osd@4
```

3. **ceph-storage0** からの接続を切断します。
4. **controller0** に SSH 接続し、 **root** でログインします。
5. Ceph monitor コンテナーの名前を特定します。

```
[root@controller0 ~]# podman ps | grep ceph-mon
ceph-mon-controller0
[root@controller0 ~]#
```

6. Ceph monitor コンテナーを有効にして、望ましくない OSD を **out** とマークします。

```
[root@controller0 ~]# podman exec ceph-mon-controller0 ceph osd out 4
```

注記

このコマンドにより、Ceph はストレージクラスターをリバランスし、データをクラスター内の他の OSD にコピーします。クラスターは、リバランスが完了するまで、一時的に **active+clean** 状態から離れます。

7. 以下のコマンドを実行し、ストレージクラスターの状態が **active+clean** になるまで待機します。

```
[root@controller0 ~]# podman exec ceph-mon-controller0 ceph -w
```

8. CRUSH マップから OSD を削除して、データを受信しないようにします。

```
[root@controller0 ~]# podman exec ceph-mon-controller0 ceph osd crush remove osd.4
```

9. OSD 認証キーを削除します。

```
[root@controller0 ~]# podman exec ceph-mon-controller0 ceph auth del osd.4
```

10. OSD を削除します。

```
[root@controller0 ~]# podman exec ceph-mon-controller0 ceph osd rm 4
```

11. **controller0** から接続を解除します。

12. **stack** ユーザーとしてアンダークラウドに SSH 接続し、**CephAnsibleDisksConfig** パラメーターを定義した heat 環境ファイルを見つけます。

13. heat テンプレートに OSD が 4 つ含まれています。

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sdb
      - /dev/sdc
      - /dev/sdd
      - /dev/sde
    osd_scenario: lvm
    osd_objectstore: bluestore
```

14. テンプレートを変更して **/dev/sde** を削除します。

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sdb
      - /dev/sdc
      - /dev/sdd
    osd_scenario: lvm
    osd_objectstore: bluestore
```

15. **openstack overcloud deploy** を実行してオーバークラウドを更新します。

注記

この例では、OSD を持つすべてのホストから **/dev/sde** デバイスを削除します。すべてのノードから同じデバイスを削除しない場合には、heat テンプレートを更新します。異なる **devices** 一覧を持つホストを定義する方法は、「異なる Ceph Storage ノードのパラメーターのオーバーライド」 を参照してください。

第12章 障害のあるディスクの置き換え

Ceph クラスターでいずれかのディスクに障害が発生した場合には、以下の手順を実行してこれを置き換えます。

1. デバイス名が変更されているかどうかを判断します。「デバイス名の変更の有無についての確認」を参照してください。
2. OSD がダウンし、破棄されていることを確認します。「OSD がダウンし、破棄されていることの確認」を参照してください。
3. システムから古いディスクを削除し、交換ディスクをインストールする場合は、「システムから古いディスクを削除し、交換ディスクをインストールします。」を参照してください。
4. ディスクの置き換えが正常に行われたことを確認します。「ディスクの置き換えが正常に行われたことの確認」を参照してください。

12.1. デバイス名の変更の有無についての確認

ディスクを置き換える前に、オペレーティングシステムで交換用 OSD の交換ディスクの名前が置き換える必要のあるデバイスの名前とは異なるかどうかを判別します。交換ディスクの名前が異なる場合は、デバイスリストの Ansible パラメーターを更新し、**ceph-ansible** のその後の実行 (director が **ceph-ansible** を実行する場合を含む) がこの変更が原因で失敗しないようにする必要があります。director を使用する際に変更する必要のあるデバイスリストの例については、「Ceph Storage ノードのディスクレイアウトのマッピング」を参照してください。

警告

デバイス名が変更され、以下の手順に従って **ceph-ansible** または director 外でシステムを更新する場合、設定管理ツールは、システム定義ファイルを更新し、設定がエラーなしに再度アサートされるまで管理しているシステムと同期しなくなる可能性があります。

ストレージデバイスの永続的な命名

sd ドライバーが管理するストレージデバイスは、再起動後も常に同じ名前を持つとは限りません。たとえば、通常は **/dev/sdc** で識別されるディスクには **/dev/sdb** という名前が付けられる可能性があります。また、交換ディスク **/dev/sdc** は、これを **/dev/sdc** の置き換えとして使用する必要がある場合でも **/dev/sdd** としてオペレーティングシステムに表示される可能性があります。この問題を回避するには、永続的な名前を使用して、**/dev/disk/by-*** のパターンに一致させます。詳細は、Red Hat Enterprise Linux (RHEL) 7**ストレージ管理ガイド**の 永続的な命名 を参照してください。

Ceph のデプロイに使用する命名方法によっては、OSD の置き換え後に **devices** リストを更新する必要がある場合があります。デバイスリストを変更する必要があるかどうかを判断するには、以下の命名方法のリストを使用します。

メジャー番号およびマイナー番号の範囲を使用する方法

sd を使用している場合で、新規ディスクのインストール後もこれを引き続き使用する場合、名前が変更されているかどうかを確認します。名前が変更されていない場合、たとえば、同じ名前が **/dev/sdd** として正しく表示される場合、ディスク置き換えの手順の完了後に名前を変更する必要は

ありません。

重要

この命名方法は、時間の経過と共に名前の一貫性を保てなくなるリスクがあるために推奨されません。詳細は、RHEL 7**ストレージ管理ガイド**の 永続的な命名 を参照してください。

by-path 方式

この方法を使用して、同じスロットに交換ディスクを追加する場合は、パスの一貫性が保たれるため、変更は必要ありません。

重要

メジャー番号とマイナー番号の範囲を使用する方法よりもこの命名方法を使用することが推奨されますが、ターゲット番号が変更されないように注意してください。たとえば、ホストアダプターが別の PCI スロットに移動する場合、永続バインディングを使用し、名前を更新します。さらに、SCSI ホスト番号は、HBA がプローブに失敗した場合、ドライバーが別の順序でロードされる場合、または新規の HBA がシステムにインストールされる場合に変更される可能性があります。**by-path** 命名方法は、RHEL7 と RHEL8 間で異なります。詳細は、

- 記事 [What is the difference between "by-path" links created in RHEL8 and RHEL7?] https://access.redhat.com/solutions/5171991
- RHEL 8**ファイルシステムの管理**の 永続的な命名属性の概要

by-uuid 方式

この方法を使用する場合は、**blkid** ユーティリティーを使用して、古いディスクと同じ UUID を持つように新しいディスクを設定できます。詳細は、RHEL 7**ストレージ管理ガイド**の 永続的な命名 を参照してください。

by-id 方式

この方法を使用する場合は、この識別子がデバイスのプロパティーであり、デバイスは置き換えられているため、デバイスリストを変更する必要があります。

新しいディスクをシステムに追加する際に、**RHEL7 ストレージ管理ガイド**(永続的な命名 を参照) に従って、デバイス名が変更されないように永続的な命名属性を変更できる場合、デバイスリストを更新して **ceph-ansible** を再実行したり、director をトリガーして **ceph-ansible** を再実行したりする必要はなく、ディスク交換の手順に進むことができます。ただし、**ceph-ansible** を再実行して、変更によって不整合が生じていないことを確認できます。

12.2. OSD がダウンし、破棄されていることの確認

Ceph Monitor をホストするサーバーで、実行中のモニターコンテナーで **ceph** コマンドを使用し、置き換える必要のある OSD が停止していることを確認してから、これを破棄します。

手順

1. 実行中の Ceph モニターコンテナーの名前を特定し、これを **MON** という環境変数に保存します。

```
MON=$(podman ps | grep ceph-mon | awk {'print $1'})
```

2. **ceph** コマンドにエイリアスを設定し、これが実行中の Ceph モニターコンテナー内で実行されるようにします。

```
alias ceph="podman exec $MON ceph"
```

3. 新規エイリアスを使用して、置き換える OSD が停止していることを確認します。

```
[root@overcloud-controller-0 ~]# ceph osd tree | grep 27
27   hdd 0.04790         osd.27                 down  1.00000 1.00000
```

4. OSD を破棄します。以下のコマンド例は **OSD 27** を破棄します。

```
[root@overcloud-controller-0 ~]# ceph osd destroy 27 --yes-i-really-mean-it
destroyed osd.27
```

12.3. システムから古いディスクを削除し、交換ディスクをインストールします。

置き換える必要のある OSD のあるコンテナーホストで、システムから古いディスクを削除し、交換ディスクをインストールします。

前提条件

- デバイス ID が変更されたことを確認します。詳細は、「デバイス名の変更の有無についての確認」 を参照してください。

ceph-volume コマンドは Ceph コンテナーにありますが、オーバークラウドノードにはインストールされません。**ceph-volume** コマンドが Ceph コンテナー内で **ceph-volume** バイナリーを実行できるようにエイリアスを作成します。次に、**ceph-volume** コマンドを使用して新規ディスクをクリーンアップし、これを OSD として追加します。

手順

1. 障害のある OSD が実行されていないことを確認します。

```
systemctl stop ceph-osd@27
```

2. ceph コンテナーイメージのイメージ ID を特定して、これを **IMG** という環境変数に保存します。

```
IMG=$(podman images | grep ceph | awk {'print $3'})
```

3. **ceph-volume** コマンドにエイリアスを設定し、これが **ceph-volume** エントリーポイントおよび関連するディレクトリーを使用して **$IMG** Ceph コンテナー内で実行されるようにします。

```
alias ceph-volume="podman run --rm --privileged --net=host --ipc=host -v
/run/lock/lvm:/run/lock/lvm:z -v /var/run/udev/:/var/run/udev/:z -v /dev:/dev -v
/etc/ceph:/etc/ceph:z -v /var/lib/ceph/:/var/lib/ceph/:z -v /var/log/ceph/:/var/log/ceph/:z --
entrypoint=ceph-volume $IMG --cluster ceph"
```

4. aliased コマンドが正常に実行されていることを確認します。

```
ceph-volume lvm list
```

5. 新規 OSD デバイスが LVM に含まれていないことを確認します。**pvdisplay** コマンドを使用してデバイスを検査し、**VG Name** フィールドが空であることを確認します。**<NEW_DEVICE>** を新規 OSD デバイスの **/dev/*** パスに置き換えます。

```
[root@overcloud-computehci-2 ~]# pvdisplay <NEW_DEVICE>
  --- Physical volume ---
  PV Name               /dev/sdj
  VG Name               ceph-0fb0de13-fc8e-44c8-99ea-911e343191d2
  PV Size               50.00 GiB / not usable 1.00 GiB
  Allocatable           yes (but full)
  PE Size               1.00 GiB
  Total PE              49
  Free PE               0
  Allocated PE          49
  PV UUID               kOO0If-ge2F-UH44-6S1z-9tAv-7ypT-7by4cp
[root@overcloud-computehci-2 ~]#
```

この **VG Name** フィールドが空でない場合は、デバイスは削除する必要のあるボリュームグループに属します。

6. デバイスがボリュームグループに属する場合は、**lvdisplay** コマンドを使用して、ボリュームグループに論理ボリュームがあるかどうかを確認します。**<VOLUME_GROUP>** を、**pvdisplay** コマンドから取得した **VG Name** フィールドの値に置き換えます。

```
[root@overcloud-computehci-2 ~]# lvdisplay | grep <VOLUME_GROUP>
  LV Path                /dev/ceph-0fb0de13-fc8e-44c8-99ea-911e343191d2/osd-data-a0810722-
7673-43c7-8511-2fd9db1dbbc6
  VG Name                ceph-0fb0de13-fc8e-44c8-99ea-911e343191d2
[root@overcloud-computehci-2 ~]#
```

この **LV Path** フィールドが空でない場合は、デバイスには、削除する必要のある論理ボリュームが含まれます。

7. 新規デバイスが論理ボリュームまたはボリュームグループの一部である場合、論理ボリューム、ボリュームグループ、および LVM システム内の物理ボリュームとしてのデバイスの関連付けを削除します。

   - **<LV_PATH>** を **LV Path** フィールドの値に置き換えます。
   - **<VOLUME_GROUP>** を **VG Name** フィールドの値に置き換えます。
   - **<NEW_DEVICE>** を新規 OSD デバイスの **/dev/*** パスに置き換えます。

```
[root@overcloud-computehci-2 ~]# lvremove --force <LV_PATH>
  Logical volume "osd-data-a0810722-7673-43c7-8511-2fd9db1dbbc6" successfully
removed
```

```
[root@overcloud-computehci-2 ~]# vgremove --force <VOLUME_GROUP>
  Volume group "ceph-0fb0de13-fc8e-44c8-99ea-911e343191d2" successfully removed
```

```
[root@overcloud-computehci-2 ~]# pvremove <NEW_DEVICE>
  Labels on physical volume "/dev/sdj" successfully wiped.
```

8. 新しい OSD デバイスがクリーンであることを確認します。以下の例では、デバイスは **/dev/sdj** になります。

```
[root@overcloud-computehci-2 ~]# ceph-volume lvm zap /dev/sdj
--> Zapping: /dev/sdj
--> --destroy was not specified, but zapping a whole device will remove the partition table
Running command: /usr/sbin/wipefs --all /dev/sdj
Running command: /bin/dd if=/dev/zero of=/dev/sdj bs=1M count=10
 stderr: 10+0 records in
10+0 records out
10485760 bytes (10 MB, 10 MiB) copied, 0.010618 s, 988 MB/s
--> Zapping successful for: <Raw Device: /dev/sdj>
[root@overcloud-computehci-2 ~]#
```

9. 新しいデバイスを使用して既存の OSD ID で新しい OSD を作成しますが、**ceph-volume** が OSD の起動を試行しないように **--no-systemd** を渡します。これはコンテナー内からは実行できません。

```
ceph-volume lvm create --osd-id 27 --data /dev/sdj --no-systemd
```

10. コンテナー外で OSD を起動します。

```
systemctl start ceph-osd@27
```

12.4. ディスクの置き換えが正常に行われたことの確認

アンダークラウドでディスクの置き換えが正常に行われたことを確認するには、以下の手順を実行します。

手順

1. デバイス名が変更されたかどうかを確認し、Ceph のデプロイに使用した命名方法に従ってデバイスリストを更新します。詳細は、「デバイス名の変更の有無についての確認」 を参照してください。

2. 変更に不整合が生じないようにするには、overcloud deploy コマンドを再実行して、スタックの更新を実行します。

3. 異なるデバイスリストを持つホストがある場合は、例外を定義する必要がある場合があります。たとえば、以下に示す heat 環境ファイルのサンプルを使用して、3 つの OSD デバイスを持つノードをデプロイできます。

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sdb
      - /dev/sdc
      - /dev/sdd
    osd_scenario: lvm
    osd_objectstore: bluestore
```

CephAnsibleDisksConfig パラメーターは OSD をホストするすべてのノードに適用されるため、**devices** パラメーターを新しいデバイスリストで更新することはできません。代わりに、別のデバイスリストを持つ新規ホストの例外を定義する必要があります。例外の定義に関する

詳細は、「異なる Ceph Storage ノードのパラメーターのオーバーライド」 および 「Ceph Storage ノードのディスクレイアウトの変更」 を参照してください。

付録A 環境ファイルのサンプル: CEPH STORAGE クラスターの作成

以下のカスタム環境ファイルは、*2章オーバークラウドデプロイメント用の Ceph Storage ノードの準備*で説明したオプションの多くを使用しています。このサンプルには、コメントアウトされているオプションは含まれません。環境ファイルの概要については、オーバークラウドの高度なカスタマイズ ガイドの 環境ファイル を参照してください。

/home/stack/templates/storage-config.yaml

```
parameter_defaults: (1)
  CinderBackupBackend: ceph (2)
  CephAnsibleDisksConfig: (3)
    osd_scenario: lvm
    osd_objectstore: bluestore
    dmcrypt: true
    devices:
      - /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:10:0
      - /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:11:0
      - /dev/nvme0n1
  ControllerCount: 3 (4)
  OvercloudControlFlavor: control
  ComputeCount: 3
  OvercloudComputeFlavor: compute
  CephStorageCount: 3
  OvercloudCephStorageFlavor: ceph-storage
  CephMonCount: 3
  OvercloudCephMonFlavor: ceph-mon
  CephMdsCount: 3
  OvercloudCephMdsFlavor: ceph-mds
  NeutronNetworkType: vxlan (5)
```

(1) **parameter_defaults** セクションは、全テンプレート内のパラメーターのデフォルト値を変更します。ここに記載のエントリーの大半は*4章ストレージサービスのカスタマイズ*で説明しています。

(2) Ceph Object Gateway をデプロイする場合には、Ceph Object Storage (**ceph-rgw**) をバックアップのターゲットとして使用することができます。このターゲットを設定するには、**CinderBackupBackend** を **swift** に設定します。詳しくは、「Ceph Object Gateway の有効化」を参照してください。

(3) **CephAnsibleDisksConfig** セクションは、BlueStore を使用するデプロイメントのカスタムディスクレイアウトを定義します。

(4) 各ロールでは ***Count** パラメーターでノード数を割り当て、**Overcloud*Flavor** パラメーターでフレーバーを割り当てます。たとえば、**ControllerCount: 3** は 3 つのノードを Controller ロールに割り当て、**OvercloudControlFlavor: control** は各ロールが **control** フレーバーを使用するように設定します。詳しくは、「ロールへのノードとフレーバーの割り当て」を参照してください。

注記

(**ceph-mon** および **ceph-mds** フレーバーと共に)
CephMonCount、**CephMdsCount**、**OvercloudCephMonFlavor**、および **OvercloudCephMdsFlavor** のパラメーターは、3章*専用ノード上での Ceph サービスのデプロイ*に記載のように、カスタムの **CephMON** および **CephMds** ロールを作成した場合のみ有効となります。

5 **NeutronNetworkType:** は、**neutron** サービスが使用すべきネットワークの種別 (ここでは **vxlan**) を設定します。

付録B カスタムインターフェイステンプレートの例: 複数のボンディングされたインターフェイス

以下のテンプレートは、**/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml** をカスタマイズしたバージョンです。バックエンドとフロントエンドのストレージネットワークトラフィックを分離するための複数のボンディングインターフェイスと、両方の接続用の冗長性の機能が含まれています (「Ceph ノード向けの複数のボンディングされたインターフェイスの設定」で説明)。

また、カスタムのボンディングオプションも使用します (「ボンディングモジュールのディレクティブの設定」で説明する **'mode=4 lacp_rate=1'**)。

/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml (カスタム)

```
heat_template_version: 2015-04-30

description: >
  Software Config to drive os-net-config with 2 bonded nics on a bridge
  with VLANs attached for the ceph storage role.

parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
  InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal API network
    type: string
  StorageIpSubnet:
    default: ''
    description: IP address/subnet on the storage network
    type: string
  StorageMgmtIpSubnet:
    default: ''
    description: IP address/subnet on the storage mgmt network
    type: string
  TenantIpSubnet:
    default: ''
    description: IP address/subnet on the tenant network
    type: string
  ManagementIpSubnet: # Only populated when including environments/network-management.yaml
    default: ''
    description: IP address/subnet on the management network
    type: string
  BondInterfaceOvsOptions:
    default: 'mode=4 lacp_rate=1'
    description: The bonding_options string for the bond interface. Set
                 things like lacp=active and/or bond_mode=balance-slb
                 using this option.
```

```
    type: string
    constraints:
      - allowed_pattern: "^((?!balance.tcp).)*$"
        description: |
          The balance-tcp bond mode is known to cause packet loss and
          should not be used in BondInterfaceOvsOptions.
  ExternalNetworkVlanID:
    default: 10
    description: Vlan ID for the external network traffic.
    type: number
  InternalApiNetworkVlanID:
    default: 20
    description: Vlan ID for the internal_api network traffic.
    type: number
  StorageNetworkVlanID:
    default: 30
    description: Vlan ID for the storage network traffic.
    type: number
  StorageMgmtNetworkVlanID:
    default: 40
    description: Vlan ID for the storage mgmt network traffic.
    type: number
  TenantNetworkVlanID:
    default: 50
    description: Vlan ID for the tenant network traffic.
    type: number
  ManagementNetworkVlanID:
    default: 60
    description: Vlan ID for the management network traffic.
    type: number
  ControlPlaneSubnetCidr: # Override this via parameter_defaults
    default: '24'
    description: The subnet CIDR of the control plane network.
    type: string
  ControlPlaneDefaultRoute: # Override this via parameter_defaults
    description: The default route of the control plane network.
    type: string
  ExternalInterfaceDefaultRoute: # Not used by default in this template
    default: '10.0.0.1'
    description: The default route of the external network.
    type: string
  ManagementInterfaceDefaultRoute: # Commented out by default in this template
    default: unset
    description: The default route of the management network.
    type: string
  DnsServers: # Override this via parameter_defaults
    default: []
    description: A list of DNS servers (2 max for some implementations) that will be added to
resolv.conf.
    type: comma_delimited_list
  EC2MetadataIp: # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
    type: string

resources:
  OsNetConfigImpl:
```

```
type: OS::Heat::StructuredConfig
properties:
  group: os-apply-config
  config:
    os_net_config:
      network_config:
        -
          type: interface
          name: nic1
          use_dhcp: false
          dns_servers: {get_param: DnsServers}
          addresses:
            -
              ip_netmask:
                list_join:
                  - '/'
                  - - {get_param: ControlPlaneIp}
                    - {get_param: ControlPlaneSubnetCidr}
          routes:
            -
              ip_netmask: 169.254.169.254/32
              next_hop: {get_param: EC2MetadataIp}
            -
              default: true
              next_hop: {get_param: ControlPlaneDefaultRoute}
        -
          type: ovs_bridge
          name: br-bond
          members:
            -
              type: linux_bond
              name: bond1
              bonding_options: {get_param: BondInterfaceOvsOptions}
              members:
                -
                  type: interface
                  name: nic2
                  primary: true
                -
                  type: interface
                  name: nic3
            -
              type: vlan
              device: bond1
              vlan_id: {get_param: StorageNetworkVlanID}
              addresses:
                -
                  ip_netmask: {get_param: StorageIpSubnet}
        -
          type: ovs_bridge
          name: br-bond2
          members:
            -
              type: linux_bond
              name: bond2
              bonding_options: {get_param: BondInterfaceOvsOptions}
```

```
            members:
              -
                type: interface
                name: nic4
                primary: true
              -
                type: interface
                name: nic5
          -
            type: vlan
            device: bond1
            vlan_id: {get_param: StorageMgmtNetworkVlanID}
            addresses:
              -
                ip_netmask: {get_param: StorageMgmtIpSubnet}
outputs:
  OS::stack_id:
    description: The OsNetConfigImpl resource.
    value: {get_resource: OsNetConfigImpl}
```