



Red Hat OpenStack Platform 15

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

director でコンテナ化された Red Hat Ceph クラスタをデプロイして使用する設
定

Red Hat OpenStack Platform 15 コンテナ化された 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

Enter your first name here. Enter your surname here.

Enter your organisation's name here. Enter your organisational division here.

Enter your email address here.

法律上の通知

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Deploying_an_overcloud_with_containerized_Red_Hat_Ceph.ent file |.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

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

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本ガイドでは、Red Hat OpenStack Platform director を使用して、コンテナ化された Red Hat Ceph Storage クラスタ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 クラスタをカスタマイズする手順が含まれます。

目次

第1章 はじめに	4
1.1. CEPH STORAGE の概要	4
1.2. 要件	4
1.2.1. Ceph Storage ノードの要件	5
1.3. 関連資料	6
第2章 オーバークラウドノードの準備	7
2.1. CEPH STORAGE ノードのディスクのクリーニング	7
2.2. ノードの登録	7
2.3. CEPH STORAGE のデプロイメント前の検証	10
2.3.1. ceph-ansible パッケージバージョンの確認	10
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	10
2.4. 手動によるプロファイルへのノードのタグ付け	11
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	12
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	14
第3章 専用ノード上での CEPH サービスのデプロイ	15
3.1. カスタムロールファイルの作成	15
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	15
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	17
第4章 ストレージサービスのカスタマイズ	19
4.1. CEPH METADATA SERVER の有効化	20
4.2. CEPH OBJECT GATEWAY の有効化	21
4.3. バックアップサービスで CEPH を使用する設定	21
4.4. 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ースの設定	22
4.4.1. ボンディングモジュールのディレクティブの設定	25
第5章 CEPH STORAGE クラスターのカスタマイズ	26
5.1. CEPH STORAGE を使用する RED HAT OPENSTACK PLATFORM 向けの CEPH コンテナ	27
5.2. CEPH STORAGE ノードのディスクレイアウトのマッピング	27
5.2.1. BlueStore の使用	28
5.2.2. 永続デバイス名が付いたデバイスの参照	28
5.3. 異なる CEPH プールへのカスタムの属性の割り当て	29
5.4. 異なる構成の CEPH STORAGE ノードへのディスクレイアウトのマッピング	30
5.5. 大規模 CEPH クラスターでの再開待機時間の延長	32
5.6. ANSIBLE 環境変数のオーバーライド	33
第6章 OPENSTACK への 2 層 CEPH STORAGE のデプロイ	34
6.1. CRUSH マップを作成します。	34
6.2. OSD のマッピング	34
6.3. レプリケーションファクターの設定	35
6.4. CRUSH 階層の定義	35
6.5. CRUSH マップルールの定義	38
6.6. OSP プールの設定	39
6.7. 新規プールを使用するために BLOCK STORAGE を設定	40
6.8. カスタマイズされた CRUSH マップの確認	40
第7章 オーバークラウドの作成	41
7.1. ロールへのノードとフレーバーの割り当て	41
7.2. オーバークラウドデプロイメントの開始	42

第8章 デプロイメント後	45
8.1. オーバークラウドへのアクセス	45
8.2. CEPH STORAGE ノードの監視	45
第9章 環境のリポート	47
9.1. CEPH STORAGE (OSD) クラスターのリポート	47
第10章 CEPH STORAGE クラスターのスケーリング	49
10.1. CEPH STORAGE クラスターのスケールアップ	49
10.2. CEPH STORAGE ノードのスケールダウンと置き換え	51
10.3. OSD の CEPH STORAGE ノードへの追加	53
10.4. CEPH STORAGE ノードからの OSD の削除	54
10.5. ディスク障害の処理	56
付録A 環境ファイルのサンプル: CEPH STORAGE クラスターの作成	57
付録B カスタムインターフェー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ース	59

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 は、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、オーバークラウドと共にコンテナ化された 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。

OpenStack のコンテナ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ースと従来のインターフェー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。Ceph デプロイメントはすべて、2種類のデーモンで構成される **Ceph Storage クラスター** を中心とします。

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。



重要

本ガイドでは、Ceph Block Storage および Ceph Object Gateway(RGW)の統合情報を提供します。Ceph File(CephFS)ストレージに関する情報は含まれません。

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 director をインストールしたアンダークラウドホスト。「[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。

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役割を果たします。

配置グループ

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 では配置グループ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 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 ([Ceph Placement Groups \(PGs\) per Pool Calculator](#) を参照)。

プロセッサ

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。

メモリ

一般的には、OSD ホスト毎に 16 GB の RAM をベースとし、さらに OSD デモン毎に 2 GB の RAM を追加することを推奨します。

ディスクのレイアウト

サイズはストレージ要件によって異なります。Red Hat では、Ceph Storage ノードの構成には、以下のレイアウト例に示すように 3 つ以上のディスクを含めることを推奨します。

- **/dev/sda**: ルートディスク。director は、主なオーバークラウドイメージをディスクにコピーします。ディスクに最小 40 GB の空きディスク領域があることを確認します。
- **/dev/sdb**: ジャーナルディスク。このディスクは、Ceph OSD ジャーナル向けにパーティションに分割されます。たとえば、**/dev/sdb1**、**/dev/sdb2**、および **/dev/sdb3** のようになります。ジャーナルディスクは、通常システムパフォーマンスの向上に役立つソリッドステートドライブ (SSD) です。
- **/dev/sdc** 以降: OSD ディスク。ストレージ要件で必要な数のディスクを使用します。



注記

Red Hat OpenStack Platform 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。

ネットワークインターフェースカード

最小1枚の1Gbps ネットワークインターフェースカード (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)。タグ付けされた VLAN トラフィックを委譲する場合や、ボンディングインターフェース向けの場合には追加のネットワークインターフェースを使用します。Red Hat では、特に大量のトラフィックを提供する OpenStack Platform 環境を構築する場合には、ストレージノードに 10 Gbps インターフェースを使用することを推奨しています。

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ー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. 関連資料

`/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。

Playbook コレクションの詳細は、このプロジェクトのドキュメント (<http://docs.ceph.com/ceph-ansible/master/>) を参照してください。

また、コンテナ化された 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 テンプレートの理解](#)」および「[環境ファイル](#)」を参照してください。

最後に、OpenStack のコンテナ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>

```

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/openstack-tripleo-
validations/validations/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

オーバークラウドのデプロイメントで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、Ceph サービスをホストするオーバークラウドノードに必要なパッケージがサーバーにあ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/openstack-tripleo-
validations/validations/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。

新規ノードを検査してタグ付けするには、以下の手順を実施します。

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. ノード一覧を取得して 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。

```
$ ironic node-update 1a4e30da-b6dc-499d-ba87-0bd8a3819bc0 add
properties/capabilities='profile:control,boot_option:local'
$ ironic node-update 6faba1a9-e2d8-4b7c-95a2-c7fdbc12129a add
properties/capabilities='profile:control,boot_option:local'
$ ironic node-update 5e3b2f50-fcd9-4404-b0a2-59d79924b38e add
properties/capabilities='profile:control,boot_option:local'
$ ironic node-update 484587b2-b3b3-40d5-925b-a26a2fa3036f add
properties/capabilities='profile:compute,boot_option:local'
$ ironic node-update d010460b-38f2-4800-9cc4-d69f0d067efe add
```

```

properties/capabilities='profile:compute,boot_option:local'
$ ironic node-update d930e613-3e14-44b9-8240-4f3559801ea6 add
properties/capabilities='profile:compute,boot_option:local'
$ ironic node-update da0cc61b-4882-45e0-9f43-fab65cf4e52b add
properties/capabilities='profile:ceph-storage,boot_option:local'
$ ironic node-update b9f70722-e124-4650-a9b1-aade8121b5ed add
properties/capabilities='profile:ceph-storage,boot_option:local'
$ ironic node-update 68bf8f29-7731-4148-ba16-efb31ab8d34f add
properties/capabilities='profile:ceph-storage,boot_option:local'

```

ヒント

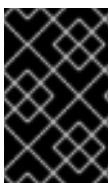
Ceph MON サービスおよび Ceph MDS サービス用のノードのタグ付けに使用できる新しいカスタムプロファイルを設定することも可能です。詳しくは、「[3章 専用ノード上での Ceph サービスのデプロイ](#)」を参照してください。

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。

以下のプロパティを定義すると、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には、以下の手順を実施します。

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。

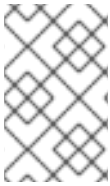
```
[
  {
    "size": 299439751168,
    "rotational": true,
    "vendor": "DELL",
    "name": "/dev/sda",
    "wwn_vendor_extension": "0x1ea4dcc412a9632b",
    "wwn_with_extension": "0x61866da04f3807001ea4dcc412a9632b",
    "model": "PERC H330 Mini",
    "wwn": "0x61866da04f380700",
    "serial": "61866da04f3807001ea4dcc412a9632b"
  }
  {
    "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=** コマンドを実行して、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起動を試み、次にルートディスクからの起動を試みるように、起動順序を設定します。

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ため、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。

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



注記

OVS は OpenStack サブスクリプションエンタイトルメントが必要な OpenStack サービスです。したがって、**overcloud-minimal** イメージでサポートされるのは標準の Linux ブリッジだけで、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](#)』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** コマンドを実行し、**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を確認するには、**openstack flavor create --help** を実行してください。

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** を実行してください。

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

```
$ ironic node-update UUID add properties/capabilities='profile:ceph-mon,boot_option:local'
```

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

- ❶ 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::ContainersLogrotateCron
- 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** コマンドを実行し、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を確認するには、**openstack flavor create --help** を実行してください。

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** を実行してください。

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

```
$ ironic node-update UUID add properties/capabilities='profile:ceph-mds,boot_option:local'
```

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。このクラスタは、コンテナ化された Ceph Storage ノードを特色とします。OpenStack のコンテナ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
<code>CinderEnableiscsiBackend</code>	iSCSI バックエンドを有効にします。	false
<code>CinderEnableRbdBackend</code>	Ceph Storage バックエンドを有効にします。	true
<code>CinderBackupBackend</code>	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
<code>NovaEnableRbdBackend</code>	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 バックエンドに CephFS を使用する構成についての詳細は、『[ファイルシステムガイド](#)』および『[Shared File System サービスの NFS バックエンドに CephFS を使用した場合のガイド](#)』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レイバリティへのインターフェー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** を必ず設定してください。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 をシームレスに使い始めることができます (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. 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`

詳細は、『[Block Storage Backup Guide](#)』を参照してください。

4.4. 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ースの設定

ボンディングされたインターフェースを使用して、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ー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ー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ー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。

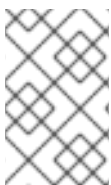
フロントエンドストレージネットワーク (StorageNet)

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ースを設定するには、新しいネットワークインターフェー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ー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ー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ーステンプレート [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ーステンプレート](#)」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ー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ース自体で、**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ースを定義します。この例では、ボンディングされるインターフェースは **nic2** (プライマリーインターフェースとして設定) と **nic3** を使用します。
- 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ース **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ースをもう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ースに移動できます。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** インターフェースは、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で使用されます。

```

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、

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ー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ース](#)」を参照してください。

4.4.1. 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ースを追加および設定したら、**BondInterfaceOvsOptions** パラメータを使用して各ボンディングされたインターフェー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](#)』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ー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ース](#)」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

コンテナ化された 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、以下のリソースも定義します。

CephAnsibleDisksConfig

このリソースにより、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

このリソースにより、その他すべてのカスタム設定を Ceph 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。

`/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml` 環境ファイルは、**ceph-ansible** パッケージが提供する Playbook を使用します。したがって、まずこのパッケージをアンダークラウドにインストールする必要があります。

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

Ceph 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（例：`/home/stack/templates/ceph-config.yaml`）でカスタムのパラメーターを定義します。ご自分の環境ファイルの `parameter_defaults` セクションで以下の構文を使用して、Ceph クラスターのグローバル設定を自由に適用することができます。

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

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、以下のスニペットについて考えてみましょう。

```
parameter_defaults:
  CephConfigOverrides:
    max_open_files: 131072
```

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
```

サポートされるパラメーターについての詳しい情報は、『[Red Hat Ceph Storage Configuration Guide](#)』を参照してください。



注記

CephConfigOverrides パラメーターは、**ceph.conf** ファイルの **[global]** セクションにしか適用されません。CephConfigOverrides パラメーターを使用して、**[osd]** セクション等の他のセクションに変更を加えることはできません。

ceph-ansible ツールには **group_vars** ディレクトリーがあり、これを使用してさまざまな Ceph パラメーターを設定することができます。詳細は、『インストール ガイド』の「[Red Hat Ceph Storage クラスターのインストール](#)」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。

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

OpenStack Platform が Ceph を使用するように設定するには、Ceph コンテナが必要です。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、OpenStack Platform 15 には Red Hat Ceph Storage 4 が必要です。Ceph Storage 4 コンテナは、`registry.redhat.io` (認証が必要なレジストリー) でホストされます。

「[Container image preparation parameters](#)」で説明されているように、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。

5.2. 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
<code>osd_scenario</code>	必須	<code>lvm</code>	lvm 設定により、ceph-ansible は ceph-volume を使用して OSD および BlueStore WAL デバイスを設定することができます。
<code>devices</code>	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一覧。

変数	必須/オプション	デフォルト値 (未設定の場合)	説明
dmccrypt	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。
osd_objectstore	オプション	bluestore	Ceph の使用するストレージバックエンドを設定します。

5.2.1. BlueStore の使用

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
```

/dev/nvme0n1 は高パフォーマンスのデバイスクラスであるため、例に示すパラメーターのデフォルトでは、**/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、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。BlueStore WAL データが OSD と同じディスクに存在する場合は、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
```

5.2.2. 永続デバイス名が付いたデバイスの参照

一部のノードでは、**/dev/sdb** および **/dev/sdc** などのディスクパスが、リポ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 **CephStorage** 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

オーバークラウドのデプロイメント前に OSD デバイスの一覧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に関する詳しい情報は、「[永続的な命名](#)」を参照してください。

コンテナ化された Ceph Storage の各ジャーナリングシナリオおよびディスクのマッピングに関する詳細は、[ceph-ansible のプロジェクトドキュメント](#) の「[OSD Scenario](#)」の項を参照してください。

5.3. 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 プールには、同じ配置グループ (`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で上書きし、全プールに同じ値を適用することが可能です。

それぞれの Ceph プールに異なる属性を適用することもできます。そのためには、以下のように `CephPools` パラメーターを使用します。

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

`POOL` を設定するプールの名前に置き換えて、さらに配置グループの数を表す `pg_num` 設定を指定します。この設定で、指定したプールのデフォルト `pg_num` が上書きされます。

`CephPools` パラメーターを使用する場合には、アプリケーションの種別も指定する必要があります。Compute、Block Storage、および Image Storage のアプリケーション種別は、例に示すように `rbd` に設定します。ただし、プールの使用目的に応じて、異なるアプリケーションの種別を指定しなければな

らない場合があります。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。

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一覧だけが対象です。

CephPools パラメーターを使用して、新たなカスタムプールを作成することもできます。たとえば、`custompool` というプールを追加するには、以下のように設定します。

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

これにより、デフォルトのプールに加えて新たなカスタムプールが作成されます。

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。

5.4. 異なる構成の CEPH STORAGE ノードへのディスクレイアウトのマッピング

デフォルトでは、(`roles_data.yaml` で `OS::TripleO::Services::CephOSD` サービスとして指定される) Ceph OSD をホストするロールのすべてのノード (例: `CephStorage` または `ComputeHCI` ノード) で、「[Ceph Storage ノードのディスクレイアウトのマッピング](#)」で設定したグローバル デバイス一覧を使用します。この場合、これらすべてのサーバーが同一のハードウェア構成であることを前提としています。これらのサーバーのサブセットが同一のハードウェア構成ではない場合には、director はこれらのサーバーごとに異なる デバイス 一覧があることに注意する必要があります。このような状況は、**ノード固有のディスク構成** と呼ばれます。

ノード固有のディスク構成を 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一覧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 Ironic データベースから抽出することができます。

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実行します。

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

Ironic 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実行します。

```
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` と設定しないと、マシン固有の UUID は Ironic データベースには含まれません。



重要

マシン固有の UUID は、Ironic UUID とは異なります。

有効な `node-spec-overrides.yaml` ファイルは、以下のようになります。

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

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容易に確認することができます。以下に例を示します。

1. 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。
2. `cat node-spec-overrides.yaml | jq .` を実行します。

`node-spec-overrides.yaml` ファイルが大きくなるにつれ、jq 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、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 JSON がデバイス一覧の正しい長さを定義することを確認します。

```
(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]$
```

上記の例では、`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。より複雑な例が、本セクションの最後に記載されています。

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一覧は 35 のディスクで構成されていましたが、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一覧を上書きし、グローバル一覧の代わりに使用する 34 のディスクの一覧を提供します。

```
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"
      ]
    }
  }
}
```

5.5. 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。

5.6. 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。

第6章 OPENSTACK への 2 層 CEPH STORAGE のデプロイ

OpenStack director を使用すると、特定の層に属する新たな Ceph ノードを Ceph クラスタに追加することで、さまざまな 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。

たとえば、SSD ドライブを持つ新たな Object Storage Daemon (OSD) ノードを既存の Ceph クラスタに追加して、これらのノードのデータを保存する専用の Block Storage (cinder) バックエンドを作成することができます。次に、新たな Block Storage ボリュームを作成するユーザーは、希望するパフォーマンス層 (HDD または新しい SSD) を選択することができます。

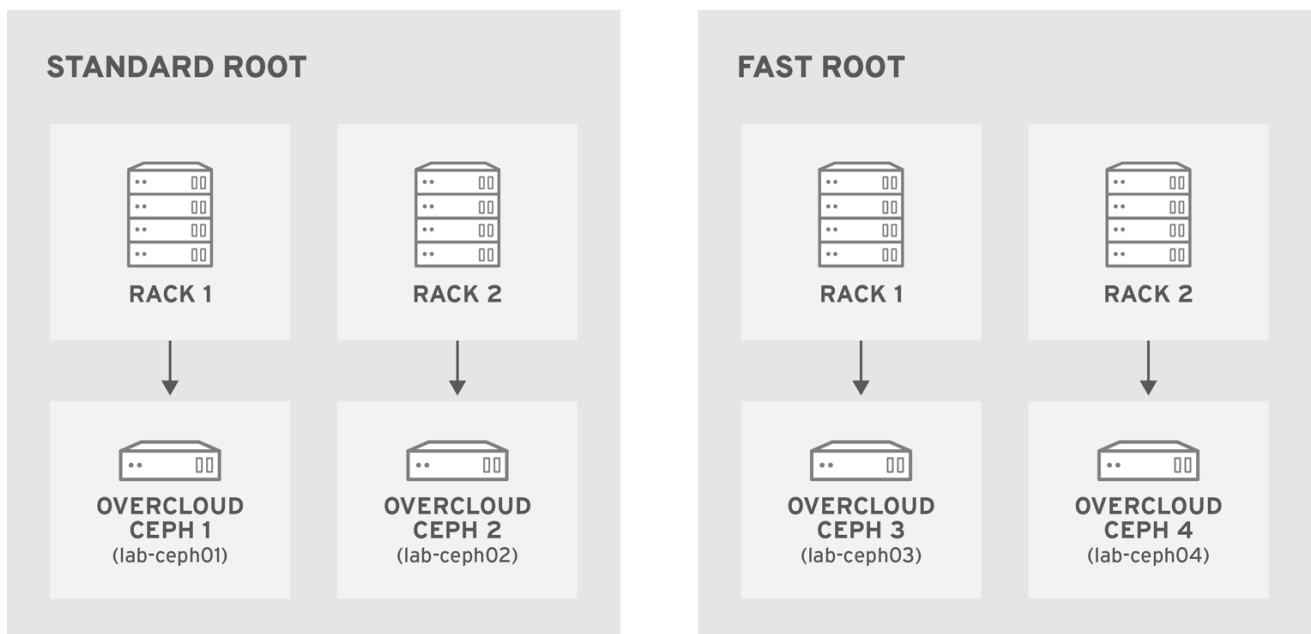
このタイプのデプロイメントでは、Red Hat OpenStack Platform director がカスタマイズされた CRUSH マップを ceph-ansible に渡す必要があります。CRUSH マップを使用すると、ディスクパフォーマンスに基づいて OSD ノードを分割することができますが、この機能を物理インフラストラクチャレイアウトのマッピングに使用することもできます。

以下のセクションでは、4つのノードのデプロイ方法を説明します。4つのうち、2つは SSD を使用し、他の2つは HDD を使用します。反復可能なパターンの通信用に簡単な例を示します。ただし、実稼働デプロイメントでは、より多くのノードおよび OSD を使用して、[「Red Hat Ceph Storage hardware selection guide」](#) に従ってサポートされるようにする必要があります。

6.1. CRUSH マップを作成します。

CRUSH マップを使用すると、OSD ノードを CRUSH ルートに配置できます。デフォルトでは、「デフォルト」のルートが作成され、これにすべての OSD ノードが含まれます。

指定のルート内では、物理トポロジー、ラック、ルームなどを定義してから、必要な階層 (またはバケット) に OSD ノードを配置します。デフォルトでは、物理トポロジーは定義されません。すべてのノードが同じラックにあるかのようなフラットな設計を前提とします。



OPENSTACK_481504_1118

カスタム CRUSH マップの作成に関する詳細は、『[Storage Strategies Guide](#)』の「[Crush Administration](#)」を参照してください。

6.2. OSD のマッピング

OSD をマップするには、以下の手順を実施します。

手順

1. OSD/ジャーナルのマッピングを宣言します。

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sda
      - /dev/sdb
    dedicated_devices:
      - /dev/sdc
      - /dev/sdc
    osd_scenario: non-collocated
    journal_size: 8192
```

6.3. レプリケーションファクターの設定

レプリケーションファクターを設定するには、以下の手順を実施します。



注記

通常、これは完全な SSD デプロイメントでのみサポートされています。「[Red Hat Ceph Storage でサポートされる構成](#)」を参照してください。

手順

1. デフォルトのレプリケーションファクターを 2 に設定します。この例では、4 つのノードを 2 つの異なるルートに分割します。

```
parameter_defaults:
  CephPoolDefaultSize: 2
```



注記

gnocchi をバックエンドとして使用するデプロイメントをアップグレードすると、デプロイメントがタイムアウトする可能性があります。このタイムアウトを阻止するには、以下の **CephPool** 定義を使用して gnocchi プールをカスタマイズします。

```
parameter_defaults
  CephPools: {"name": metrics, "pg_num": 128, "pgp_num": 128, "size": 1}
```

6.4. CRUSH 階層の定義

director は CRUSH 階層用のデータを提供しますが、実際には ceph-ansible が Ansible インベントリファイルを通じて CRUSH マッピングを取得し、そのデータを渡します。デフォルトのルートを維持しない場合は、各ノードのルートの場所を指定する必要があります。

たとえば、ノード lab-ceph01 (プロビジョニング IP 172.16.0.26) が **fast_root** 内の **rack1** に置かれる場合、Ansible インベントリは以下ようになります。

```
172.16.0.26:
```

```
osd_crush_location: {host: lab-ceph01, rack: rack1, root: fast_root}
```

director を使用して Ceph をデプロイする場合、Ansible インベントリは実際に作成するのではなく、生成されます。したがって、**NodeDataLookup** を使用してデータを追加する必要があります。

NodeDataLookup は、システムのマザーボードに保管されているシステム製品 UUID を指定することで機能します。Bare Metal サービス (ironic) も、イントロスペクションフェーズ後にこの情報を保管します。

2 層ストレージをサポートする CRUSH マップを作成するには、以下の手順を行います。

手順

1. 以下のコマンドを実行して、4 つのノードの UUID を取得します。

```
for ((x=1; x<=4; x++)); \
{ echo "Node overcloud-ceph0${x}"; \
openstack baremetal introspection data save overcloud-ceph0${x} | jq
.extra.system.product.uuid; }
Node overcloud-ceph01
"32C2BC31-F6BB-49AA-971A-377EFDDB111"
Node overcloud-ceph02
"76B4C69C-6915-4D30-AFFD-D16DB74F64ED"
Node overcloud-ceph03
"FE7CF7B20-5984-469F-872C-732E3FEF99BF"
Node overcloud-ceph04
"5FFFEFA5F-69E4-4A88-B9EA-62811C61C8B3"
```



注記

この例では、overcloud-ceph0[1-4] は Ironic ノードの名前です。これらは、(HostnameMap.yaml を介して) **lab-ceph0[1-4]** としてデプロイされます。

2. 以下のようにノードの配置を指定します。

ルート	ラック	ノード
standard_root	rack1_std	overcloud-ceph01 (lab-ceph01)
	rack2_std	overcloud-ceph02 (lab-ceph02)
fast_root	rack1_fast	overcloud-ceph03 (lab-ceph03)
	rack2_fast	overcloud-ceph04 (lab-ceph04)

**注記**

同じ名前のバケットを 2 つ持つことはできません。**lab-ceph01** と **lab-ceph03** が同じ物理ラックにある場合でも、**rack1** と呼ばれるバケットを 2 つ持つことはできません。そのため、**rack1_std** および **rack1_fast** と命名しました。

**注記**

この例では、複数のカスタムルートを示すために「standard_root」という特定のルートの作成方法を説明します。ただし、デフォルトのルートに HDD の OSD ノードを保持することもできたはずですが。

- 以下の **NodeDataLookup** 構文を使用します。

```
NodeDataLookup: {"SYSTEM_UUID": {"osd_crush_location": {"root": "$MY_ROOT", "rack": "$MY_RACK", "host": "$OVERCLOUD_NODE_HOSTNAME"}}
```

**注記**

システム UUID を指定してから、CRUSH 階層を上から順に指定する必要があります。また、**host** パラメーターは、Bare Metal サービス (ironic) ノード名ではなく、ノードのオーバークラウドホスト名を参照する必要があります。設定例と一致するには、以下を入力します。

```
parameter_defaults:
  NodeDataLookup: {"32C2BC31-F6BB-49AA-971A-377EFDDB111":
{"osd_crush_location": {"root": "standard_root", "rack": "rack1_std", "host": "lab-ceph01"},
  "76B4C69C-6915-4D30-AFFD-D16DB74F64ED": {"osd_crush_location": {"root":
"standard_root", "rack": "rack2_std", "host": "lab-ceph02"}},
  "FECF7B20-5984-469F-872C-732E3FEF99BF": {"osd_crush_location": {"root":
"fast_root", "rack": "rack1_fast", "host": "lab-ceph03"}},
  "5FFFEFA5F-69E4-4A88-B9EA-62811C61C8B3": {"osd_crush_location": {"root":
"fast_root", "rack": "rack2_fast", "host": "lab-ceph04"}}
```

- ceph-ansible レベルで CRUSH マップ管理を有効にします。

```
parameter_defaults:
  CephAnsibleExtraConfig:
    create_crush_tree: true
```

- スケジューラーヒントを使用して、Bare Metal サービスノードの UUID がホスト名に正しくマッピングされていることを確認します。

```
parameter_defaults:
  CephStorageCount: 4
  OvercloudCephStorageFlavor: ceph-storage
  CephStorageSchedulerHints:
    'capabilities:node': 'ceph-%index%'
```

- Bare Metal サービスノードを対応するヒントでタグ付けします。

```
openstack baremetal node set --property capabilities='profile:ceph-storage,node:ceph-0,boot_option:local' overcloud-ceph01
```

```
openstack baremetal node set --property capabilities=profile:ceph-storage,'node:ceph-1,boot_option:local' overcloud-ceph02
```

```
openstack baremetal node set --property capabilities='profile:ceph-storage,node:ceph-2,boot_option:local' overcloud-ceph03
```

```
openstack baremetal node set --property capabilities='profile:ceph-storage,node:ceph-3,boot_option:local' overcloud-ceph04
```



注記

予測可能な配置の詳細は、『オーバークラウドの高度なカスタマイズ』の「特定の [ノード ID の割り当て](#)」を参照してください。

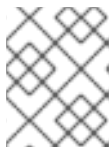
6.5. CRUSH マップルールの定義

ルールは、クラスター上でのデータの書き込み方法を定義します。CRUSH マップノードの配置が完了したら、CRUSH ルールを定義します。

手順

1. CRUSH ルールを定義するには、以下の構文を使用します。

```
parameter_defaults:
  CephAnsibleExtraConfig:
    crush_rules:
      - name: $RULE_NAME
        root: $ROOT_NAME
        type: $REPLICAT_DOMAIN
        default: true/false
```

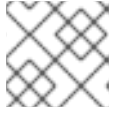


注記

Default パラメーターを **true** に設定すると、ルールを指定せずに新しいプールを作成する際にこのルールが使用されます。デフォルトのルールは1つだけです。

以下の例では、ルール **standard** は、ラックごとに1つの複製で **standard_root** でホストされる OSD ノードをポイントします。ルール **fast** は、ラックごとに1つの複製で **standard_root** でホストされる OSD ノードをポイントします。

```
parameter_defaults:
  CephAnsibleExtraConfig:
    crush_rule_config: true
    crush_rules:
      - name: standard
        root: standard_root
        type: rack
        default: true
      - name: fast
        root: fast_root
        type: rack
        default: false
```



注記

`crush_rule_config` を `true` に設定する必要があります。

6.6. OSP プールの設定

Ceph プールは、データを保管する方法を定義する CRUSH ルールで設定されます。この例では、**standard_root** (標準ルール) を使用するすべてのビルトイン OSP プールおよび **fast_root** (高速ルール) を使用する新しいプールを特長としています。

手順

1. プールプロパティを定義または変更するには、以下の構文を使用します。

```
- name: $POOL_NAME
  pg_num: $PG_COUNT
  rule_name: $RULE_NAME
  application: rbd
```

2. すべての OSP プールを一覧表示し、適切なルール (この場合は標準) を設定して **高速** ルールを使用する **tier2** という新しいプールを作成します。このプールは Block Storage (cinder) によって使用されます。

```
parameter_defaults:
  CephPools:
    - name: tier2
      pg_num: 64
      rule_name: fast
      application: rbd

    - name: volumes
      pg_num: 64
      rule_name: standard
      application: rbd

    - name: vms
      pg_num: 64
      rule_name: standard
      application: rbd

    - name: backups
      pg_num: 64
      rule_name: standard
      application: rbd

    - name: images
      pg_num: 64
      rule_name: standard
      application: rbd

    - name: metrics
      pg_num: 64
      rule_name: standard
      application: openstack_gnocchi
```

6.7. 新規プールを使用するために BLOCK STORAGE を設定

Ceph プールを **cinder.conf** ファイルに追加し、Block Storage (cinder) がこれを使用できるようにします。

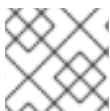
手順

1. 以下のように **cinder.conf** を更新します。

```
parameter_defaults:
  CinderRbdExtraPools:
    - tier2
```

6.8. カスタマイズされた CRUSH マップの確認

openstack overcloud deploy コマンドがオーバークラウドを作成または更新した後に、以下の手順に従って、カスタマイズされた CRUSH マップが正しく適用されたことを確認します。



注記

あるルートから別のルートにホストを移動する場合には注意してください。

手順

1. Ceph monitor ノードに接続し、以下のコマンドを実行します。

```
# ceph osd tree
ID WEIGHT  TYPE NAME                UP/DOWN REWEIGHT PRIMARY-AFFINITY
-7 0.39996  root standard_root
-6 0.19998  rack rack1_std
-5 0.19998  host lab-ceph02
 1 0.09999  osd.1      up 1.00000  1.00000
 4 0.09999  osd.4      up 1.00000  1.00000
-9 0.19998  rack rack2_std
-8 0.19998  host lab-ceph03
 0 0.09999  osd.0      up 1.00000  1.00000
 3 0.09999  osd.3      up 1.00000  1.00000
-4 0.19998  root fast_root
-3 0.19998  rack rack1_fast
-2 0.19998  host lab-ceph01
 2 0.09999  osd.2      up 1.00000  1.00000
 5 0.09999  osd.5      up 1.00000  1.00000
```

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のより詳細な一覧は、『[director のインストールと使用方法](#)』ガイドの「[CLI ツールを使用したオーバークラウドの作成](#)」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](#)」で説明するように、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。

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一覧を表示するには、以下を実行します。

```
$ openstack help overcloud deploy
```

詳しくは、『[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。

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、以下を実行します。

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


第8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。

8.1. オーバークラウドへのアクセス

director は、director ホストからオーバークラウドに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、director ホストの 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。director のホストとの対話に戻るには、以下のコマンドを実行します。

```
$ source ~/stackrc
```

8.2. 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 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。

```
$ 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9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。

9.1. 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リブートするには、以下の手順を実施します。

手順

1. Ceph MON またはコントローラーノードにログインして、Ceph Storage Cluster 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
```

2. リブートする最初の Ceph Storage ノードを選択し、そのノードにログインします。
3. ノードをリブートします。

```
$ 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 またはコントローラーノードにログインして、クラスターのリバランス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
```

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10章 CEPH STORAGE クラスターのスケールリング

10.1. 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、以下のステップを実行します。

1. director ホストに **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 OpenStack 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。

新規ノードを検査してタグ付けするには、以下の手順を実施します。

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. ノード一覧を取得して 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。

```
$ ironic node-update 551d81f5-4df2-4e0f-93da-6c5de0b868f7 add
properties/capabilities='profile:ceph-storage,boot_option:local'
$ ironic node-update 5e735154-bd6b-42dd-9cc2-b6195c4196d7 add
properties/capabilities='profile:ceph-storage,boot_option:local'
$ ironic node-update 1a2b090c-299d-4c20-a25d-57dd21a7085b add
properties/capabilities='profile:ceph-storage,boot_option:local'
```

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。その手順は、[「マルチディスククラスタ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

新規ノードの登録とタグ付け後に、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。この操作を行う際には、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。これを 6 ノードにスケールアップするには、以下の設定を使用します。

```
parameter_defaults:
  ControllerCount: 3
  OvercloudControlFlavor: control
  ComputeCount: 3
  OvercloudComputeFlavor: compute
  CephStorageCount: 6
  OvercloudCephStorageFlavor: ceph-storage
  CephMonCount: 3
  OvercloudCephMonFlavor: ceph-mon
```

この設定で再デプロイすると、オーバークラウドの Ceph Storage ノードは 3 つではなく 6 つとなるはずですが。

10.2. 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 クラスターをスケールダウンしたり、Ceph Storage ノードを置き換える（例：Ceph Storage ノードに問題がある場合など）必要が出てくる場合があります。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ースを使用した 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 ツリーを一覧表示して、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、ノードを停止します。

```
[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. 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 ~]$
```

11. 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
```

12. 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
```

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

```
$ openstack server list
```

14. 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。

```
$ openstack overcloud node delete --stack overcloud <NODE_UUID>
```



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**』の「**オーバークラウド環境の変更**」を参照してください。

director のオーケストレーションが完了したら、バックエンドをテストします。

1. 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。
2. 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** セクションで **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。

1. 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_FILES>
```

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。

2. **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
```

3. **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。

10.3. 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
```

- 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
```

- `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 を使用するすべてのホストに `/dev/sde` という新しいデバイスがあることを前提としています。全ノードで新しいデバイスを持つ必要がない場合は、以下に示すように Heat テンプレートを更新し、異なる `devices` 一覧を持つホストを定義する方法について、「[異なる構成の Ceph Storage ノードへ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。

10.4. 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

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

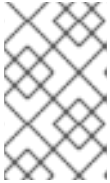
```
[root@ceph-storage0 ~]# systemctl disable ceph-osd@4
[root@ceph-storage0 ~]# systemctl stop ceph-osd@4
```

- `ceph-storage0` からの接続を切断します。
- `controller0` に SSH 接続し、`root` でログインします。
- Ceph monitor コンテナ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 を使用するすべてのホストから **/dev/sde** デバイスを削除していることを前提としています。すべてのノードから同じデバイスを削除しない場合は、以下に示すように heat テンプレートを更新し、異なる **devices** 一覧を持つホストを定義する方法について、[「異なる構成の Ceph Storage ノードへ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。

10.5. ディスク障害の処理

ディスクに障害が発生した場合は、『Red Hat Ceph Storage Operations Guide』の「[Handling a Disk Failure](#)」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](#)』の「環境ファイル」を参照してください。https://access.redhat.com/documentation/ja-jp/red_hat_openshift_platform/15/html-single/advanced_overcloud_customization#sect-Environment_Files

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

```
parameter_defaults: ❶
  CinderBackupBackend: ceph ❷
  CephAnsibleDisksConfig: ❸
    osd_scenario: lvm
    osd_objectstore: bluestore
    dmccrypt: 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 ❹
  OvercloudControlFlavor: control
  ComputeCount: 3
  OvercloudComputeFlavor: compute
  CephStorageCount: 3
  OvercloudCephStorageFlavor: ceph-storage
  CephMonCount: 3
  OvercloudCephMonFlavor: ceph-mon
  CephMdsCount: 3
  OvercloudCephMdsFlavor: ceph-mds
  NeutronNetworkType: vxlan ❺
```

- ❶ **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。
- ❷ 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。
- ❸ **CephAnsibleDisksConfig** セクションは、BlueStore および Ceph 3.2 以降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。FileStore および Ceph 3.1 以前を使用するデプロイメントの場合には、「[Ceph Storage ノードのディスクレイアウトのマッピング](#)」に記載の例を使用して **CephAnsibleDisksConfig** を変更してください。

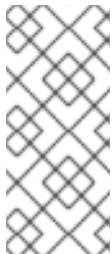


警告

上記の例では **osd_scenario: lvm** が使用され、新しいデプロイメントのデフォルトが **ceph-volume** により **bluestore** に設定されています。この設定は、ceph-ansible 3.2 以降および Ceph Luminous 以降でのみ利用可能です。ceph-ansible 3.2 で **filestore** をサポートするパラメーターは、後方互換性の目的です。したがって、既存の FileStore のデプロイメントでは、両方のバックエンドを維持するステップなしに **osd_objectstore** または **osd_scenario** パラメーターを単に変更しないでください。

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** サービスが使用する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ー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ース

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ースと、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ースの設定](#)」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}
```