Red Hat OpenStack Platform 15

ハイパーコンバージドインフラストラクチャーガイド

Red Hat OpenStack Platform オーバークラウドにおけるハイパーコンバージドインフラストラクチャーの設定についての理解

Last Updated: 2022-09-15

Red Hat OpenStack Platform 15 ハイパーコンバージドインフラストラクチャーガイド

Red Hat OpenStack Platform オーバークラウドにおけるハイパーコンバージドインフラストラクチャーの設定についての理解

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

概要

本ガイドでは、Red Hat OpenStack Platform のハイパーコンバージェンスの実装について説明します。この実装では、Compute サービスと Ceph Storage サービスが同じホストに配置されます。

目次

第1章 RED HAT OPENSTACK PLATFORM ハイパーコンバージドインフラストラクチャー

Red Hat OpenStack Platform (RHOSP) ハイパーコンバージドインフラストラクチャー (HCI) は、ハイパーコンバージドノードで構成されます。リソースの使用率を最適化するために、サービスがこのハイパーコンバージドノード上で共存します。RHOSP HCI では、Compute サービスとストレージサービスがハイパーコンバージドノード上で共存します。ハイパーコンバージドノードのみのオーバークラウド、またはハイパーコンバージドノードを通常のコンピュートノードおよび Ceph Storage ノードと混在させたオーバークラウドをデプロイすることが可能です。

注記

Red Hat Ceph Storage をストレージプロバイダーとして使用する必要があります。

ヒント

- Ceph のメモリー設定を自動的に調整するには、ceph-ansible 3.2 以降を使用します。
- BlueStore のメモリー処理機能を利用するには、BlueStore を HCI デプロイメントのバックエンドとして使用します。

本ガイドでは、オーバークラウド上に HCI をデプロイする方法、およびオーバークラウドの他の機能 (例: ネットワーク機能の仮想化) と統合する方法について説明します。ハイパーコンバージドノード上における Compute サービスと Ceph Storage サービスの両方のパフォーマンスを最適な状態にする方法についても記載しています。

1.1. 前提条件

- アンダークラウドをデプロイしている。アンダークラウドのデプロイ方法についての説明は、『director のインストールと使用方法』 を参照してください。
- お使いの環境で、Compute および Ceph Storage の要件を満たすノードをプロビジョニング可能である。詳しい情報は、『Director Installation and Usage』の「Basic overcloud deployment」を参照して ください。
- 環境内の全ノードを登録している。詳しくは、「 ノードの登録 」を参照してください。
- 環境内の全ノードがタグ付けされている。詳しくは、『Deploying an overcloud with containerized Red Hat Ceph』の「 Manually tagging nodes」を参照し てください。
- Compute サービスおよび Ceph OSD サービスに使用するノード上のディスクをクリーンアップしている。詳しくは、「 Ceph Storage ノードのディスクのクリーニング 」を参照してください。
- オーバークラウドノードを Red Hat コンテンツ配信ネットワークまたは Red Hat Satellite サーバーに登録するための準備を行っている。詳しい情報は、『オーバークラウドの高度なカスタマイズ』の「Ansible ベースのオーバークラウド登録 」を参照してください。

1.2. 参考資料

Red Hat OpenStack Platform (RHOSP) に関する詳しい情報は、以下のガイドを参照してください。

- director のインストールと使用 方法： 本ガイドでは、RHOSP 環境（アンダークラウドおよびオーバークラウドの両方）のエンドツーエンドのデプロイメントに関するガイダンスを提供します。
- オーバークラウドの高度なカスタマイズ: このガイドでは、director を使用して RHOSP の高度な機能を設定する方法について記載しています（例： カスタムロールの使用方法）。
- 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： このガイドでは、Red Hat Ceph Storage をストレージプロバイダーとして使用するオーバークラウドのデプロイ方法について記載 しています。
- ネットワーク ガイド: このガイドでは、RHOSP のネットワーク設定タスクについて詳細に説明しています。
- Hyper-converged Red Hat OpenStack Platform 10 and Red Hat Ceph Storage 2 : このガイドは、極めて特殊なハードウェアに HCI が設定された環境をデプロイする方法について説明したリファレンスアーキテクチャーを提供します。

第2章 RED HAT OPENSTACK PLATFORM HCI の設定とデプロイ

以下の手順で、Red Hat OpenStack Platform (RHOSP) HCI を設定およびデプロイする手順の概要を説明します。各ステップの詳細を、以降のセクションで説明します。

手順

1. ハイパーコンバージドノード向けの事前定義されたカスタムオーバークラウドロール **ComputeHCI** を準備する。
2. リソース分離を設定する。
3. ストレージ管理ネットワークのポートを NIC にマッピングする。
4. オーバークラウドをデプロイする。
5. (オプション) ハイパーコンバージドノードをスケーリングする。

第3章 ハイパーコンバージドノード向けのオーバークラウドロールの準備

ハイパーコンバージドノードを使用するには、そのノードにロールを定義する必要があります。Red Hat OpenStack Platform (RHOSP) は、ハイパーコンバージドノード向けの事前定義されたロール **ComputeHCI** を提供します。このロールにより、Compute サービスと Ceph オブジェクトストレージデーモン (OSD) サービスを共存させ、同じハイパーコンバージドノード上にまとめてデプロイすることができます。**ComputeHCI** ロールを使用するには、デプロイメントで使用するその他の全ロールに加えて、このロールが含まれるカスタムの **roles_data.yaml** ファイルを生成する必要があります。

以下の手順で、この事前定義されたロールの使用方法および設定方法について説明します。

手順

1. オーバークラウドに使用するその他のロールに加えて **ComputeHCI** ロールが含まれるカスタムの **roles_data.yaml** ファイルを作成します。

```
$ openstack overcloud roles generate -o /home/stack/roles_data.yaml Controller ComputeHCI Compute CephStorage
```

   カスタムロールに関する詳しい情報は、『オーバークラウドの高度なカスタマイズ』の「コンポーザブルサービスとカスタムロール」および「roles_data ファイルの検証」を参照してください。

2. **ports.yaml** という名前の新規 heat テンプレートを **~/templates** に作成します。

3. **ports.yaml** ファイルに以下の設定を追加して、**ComputeHCI** ロールのポート割り当てを設定します。

```
resource_registry:
  OS::TripleO::ComputeHCI::Ports::ExternalPort: /usr/share/openstack-tripleo-heat-templates/network/ports/<ext_port_file>.yaml
  OS::TripleO::ComputeHCI::Ports::InternalApiPort: /usr/share/openstack-tripleo-heat-templates/network/ports/internal_api.yaml
  OS::TripleO::ComputeHCI::Ports::StoragePort: /usr/share/openstack-tripleo-heat-templates/network/ports/storage.yaml
  OS::TripleO::ComputeHCI::Ports::TenantPort: /usr/share/openstack-tripleo-heat-templates/network/ports/tenant.yaml
  OS::TripleO::ComputeHCI::Ports::StorageMgmtPort: /usr/share/openstack-tripleo-heat-templates/network/ports/<storage_mgmt_file>.yaml
```

   - 外部ポートファイルの名前に置き換え **<ext_port_file>** てください。DVR を使用している場合は「external」に設定します。それ以外の場合は「noop」に設定します。DVR の詳細は、「分散仮想ルーティング(DVR)の設定」を参照してください。
   - & **lt;storage_mgmt_file>** をストレージ管理ファイルの名前に置き換えます。以下のいずれかの値に設定します。

値	説明
storage_mgmt	IP プールから選択しない場合や、環境で IPv6 アドレスを使用しない場合に使用します。

値	説明
storage_mgmt_from_pool	ComputeHCI ロールが IP プールから選択するようにする場合に使用します。
storage_mgmt_v6	環境で IPv6 アドレスを使用する場合に使用します。
storage_mgmt_from_pool_v6	ComputeHCI ロールが IPv6 アドレスプールから選択するようにする場合に使用します。

詳細は、「 基本的なネットワーク分離 」を参照してください。

4. ComputeHCI ロール用のフレーバーを作成します。

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

5. フレーバーの属性を設定します。

```
$ openstack flavor set --property "cpu_arch"="x86_64" \
--property "capabilities:boot_option"="local" \
--property "resources:CUSTOM_BAREMETAL"="1" \
--property "resources:DISK_GB"="0" \
--property "resources:MEMORY_MB"="0" \
--property "resources:VCPU"="0" computeHCI
```

6. フレーバーを新規プロファイルにマッピングします。

```
$ openstack flavor set --property "capabilities:profile"="computeHCI" computeHCI
```

7. ノード一覧を取得して UUID を把握します。

```
$ openstack baremetal node list
```

8. ノードを新規プロファイルにタグ付けします。

```
$ openstack baremetal node set --property
capabilities='profile:computeHCI,boot_option:local' <UUID>
```

詳しくは、「 ノードの手動でのタグ付け」および「 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。

9. **computeHCI** フレーバーを ComputeHCI 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:
  OvercloudComputeHCIFlavor: computeHCI
  ComputeHCICount: 3
```

第4章 ハイパーコンバージドノード上におけるリソース分離の設定

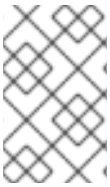
ハイパーコンバージドノード上で Ceph OSD サービスと Compute サービスを共存させると、お互いが同じホスト上に存在することを認識しないため、Ceph サービスと Compute サービス間でリソースの競合が発生するリスクがあります。リソースの競合が発生すると、サービスのパフォーマンスが低下し、ハイパーコンバージェンスの利点が打ち消される可能性があります。

以降のセクションで、競合を防ぐために Ceph サービスおよび Compute サービスの両方にリソースの分離をどのように設定すかについて詳しく説明します。

4.1. COMPUTE 用 CPU/メモリーリソースの確保

director の提供するデフォルトのプラン環境ファイルにより、デプロイメント時のハイパーコンバージドノードのリソース制約が設定されます。このプラン環境ファイルは、OpenStack Workflow に以下のプロセスを実施するように指示します。

1. ノードのハードウェア の検査 時に収集したハードウェアイントロスペクションデータを取得します。
2. そのデータに基づき、ハイパーコンバージドノード上の Compute の最適な CPU およびメモリー割り当て負荷を算出する。
3. これらの制約を設定し Compute に CPU/メモリーリソースを確保するのに必要なパラメーターを自動生成する。これらのパラメーターは、**plan-environment-derived-params.yaml** ファイルの **hci_profile_config** セクションで定義されます。



注記

Compute の **reserved_host_memory** および **cpu_allocation_ratio** の設定値を算出するのに、各ワークロードプロファイルの **average_guest_memory_size_in_mb** および **average_guest_cpu_utilization_percentage** パラメーターが使用されます。

Compute 環境ファイルに以下のパラメーターを追加して、自動生成される Compute 設定を上書きすることができます。

自動生成される nova.conf パラメーター	Compute 環境ファイルのオーバーライド	説明
reserved_host_memory	parameter_defaults: ComputeHCIParameters: NovaReservedHostMemory: 181000	ハイパーコンバージドノード上で、Ceph OSD サービスおよびゲストインスタンスごとのオーバーヘッドに確保する RAM 容量を設定します。
cpu_allocation_ratio	parameter_defaults: ComputeHCIExtraConfig: nova::cpu_allocation_ratio: 8.2	インスタンスをデプロイするコンピュートノードを選択する際に Compute スケジューラーが使用すべき比率を設定します。

これらのオーバーライドは、ComputeHCI ロールを使用するすべてのノード (つまり、すべてのハイパーコンバージドノード) に適用されます。**NovaReservedHostMemory** および **nova::cpu_allocation_ratio** の最適な値を手動で決定する方法に関する詳しい情報は、「 Compute CPU and Memory Calculator 」を参照してください。

ヒント

以下のスクリプトを使用して、ハイパーコンバージドノードの **NovaReservedHostMemory** および **cpu_allocation_ratio** の基準値を算出することができます。

nova_mem_cpu_calc.py

4.2. CEPH 用 CPU/メモリーリソースの確保

以下の手順で、Ceph 用に CPU/メモリーリソースを確保する方法について説明します。

手順

1. **/home/stack/templates/storage-container-config.yaml** のパラメーター **is_hci** を「true」に設定します。

```
parameter_defaults:
  CephAnsibleExtraConfig:
    is_hci: true
```

この設定により、**ceph-ansible** は Ceph 用のメモリーリソースを確保することができ、HCI デプロイメントの **osd_memory_target** パラメーター設定を自動的に調整することで、Ceph OSD によるメモリー増大を低減することができます。

警告

Red Hat では、**ceph_osd_docker_memory_limit** パラメーターを直接上書きすることは推奨しません。

注記

FileStore または BlueStore どちらのバックエンドが使用されていても、ceph-ansible 3.2 では、**ceph_osd_docker_memory_limit** は Ansible の検出したホストの最大メモリーに自動的に設定されます。

2. (オプション) デフォルトでは、**ceph-ansible** は Ceph OSD ごとに 1 つの仮想 CPU を確保します。Ceph OSD ごとに複数の CPU が必要な場合には、以下の設定を **/home/stack/templates/storage-container-config.yaml** に追加し、**ceph_osd_docker_cpu_limit** を必要な CPU 上限に設定します。

```
parameter_defaults:
  CephAnsibleExtraConfig:
    ceph_osd_docker_cpu_limit: 2
```

ハードウェアおよびワークロードに基づいて CPU リソースを調整する方法の詳細は、『Red Hat Ceph Storage Hardware Selection Guide』を参照してください。

4.3. CEPH のバックフィルおよびリカバリー操作の削減

Ceph OSD が削除されると、Ceph はバックフィルおよびリカバリー操作を使用してクラスターをリバランスします。Ceph は配置グループポリシーに従って、データのコピーを複数保管するためにこの操作を実行します。これらの操作は、システムリソースを使用します。Ceph クラスターに負荷がかかっている場合、リソースがバックフィルおよびリカバリーに回されるので、パフォーマンスが低下します。

OSD 削除時のこのパフォーマンスへの影響を軽減するには、バックフィルおよびリカバリー操作の優先度を低くすることができます。この手法のマイナス面は、データのレプリカがより少ない状態が長くなるので、データがリスクにさらされる可能性が若干高くなることです。

以下の表で説明するパラメーターが、バックフィルおよびリカバリー操作の優先度を設定するのに使用されます。

パラメーター	説明	デフォルト値
osd_recovery_op_priority	OSD クライアントの操作に関して、リカバリー操作の優先度を設定します。	3
osd_recovery_max_active	同時に要求できる 1 OSD あたりのアクティブなリカバリー要求の数を設定します。要求が増えるにつれてリカバリーは加速されますが、それらの要求によりクラスターにかかる負荷が増大します。レイテンシーを低くするには、このパラメーターを 1 に設定します。	3
osd_max_backfills	単一の OSD との間で許容されるバックフィルの最大数を設定します。	1

このデフォルト設定を変更するには、**~/templates** に以下の内容の環境ファイル **ceph-backfill-recovery.yaml** を追加します。

```
parameter_defaults:
  CephConfigOverrides:
    osd_recovery_op_priority: ${priority_value}
    osd_recovery_max_active: ${no_active_recovery_requests}
    osd_max_backfills: ${max_no_backfills}
```

第5章 ストレージ管理ネットワークポートの NIC へのマッピング

以下の手順で、ストレージ管理ネットワークのポートをハイパーコンバージドノードの物理 NIC にマッピングする方法について説明します。

手順

1. お使いの環境の **compute.yaml** heat テンプレートファイルを **/usr/share/openstack-tripleo-heat-templates/network/config** ディレクトリーからコピーします。以下のオプションを使用することができます。

   - single-nic-vlans
   - single-nic-linux-bridge-vlans
   - multiple-nics
   - bond-with-vlans
     NIC の設定に関する詳細は、各テンプレートのディレクトリーの **README.md** を参照してください。

2. **~/templates** に **nic-configs** という名前の新規ディレクトリーを作成します。

3. **compute.yaml** テンプレートのコピーを **~/templates/nic-configs/** に貼り付け、その名前を **compute-hci.yaml** に変更します。

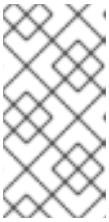
4. **~/templates/nic-configs/compute-hci.yaml** の **parameters:** セクションで、以下の定義を確認します。

   ```
   StorageMgmtNetworkVlanID:
     default: 40
     description: Vlan ID for the storage mgmt network traffic.
     type: number
   ```

   StorageMgmtNetworkVlanID の定義がまだ **compute-hci.yaml** ファイルになければ、その定義を追加します。

5. 各 HCI ノードで **StorageMgmtNetworkVlanID** を特定の NIC にマッピングします。たとえば、VLAN を単一の NIC にトランク接続する場合、**~/templates/nic-configs/compute-hci.yaml** の **network_config:** セクションに以下のエントリーを追加します。

   ```
           type: vlan
           device: em2
           mtu: 9000
           use_dhcp: false
           vlan_id: {get_param: StorageMgmtNetworkVlanID}
           addresses:
             -
               ip_netmask: {get_param: StorageMgmtIpSubnet}
   ```



注記

NIC を **StorageMgmtNetworkVlanID** にマッピングする際に、MTU を 9000 (ジャンボフレーム) に設定して、Red Hat Ceph Storage のパフォーマンスを向上させます。詳しくは、『Networking Guide』の「 Configuring MTU Settings in Director 」および「 Configuring jumbo frames 」を参照してください。

6. **~/templates/network.yaml** という名前でネットワークの環境ファイルを作成します。
7. 以下の設定を **network.yaml** ファイルに追加します。

```
resource_registry:
OS::TripleO::ComputeHCI::Net::SoftwareConfig: /home/stack/templates/nic-configs/compute-hci.yaml
```

このファイルは、後でオーバークラウドのデプロイメント中に、カスタマイズされた Compute NIC テンプレート (**~/templates/nic-configs/compute-hci.yaml**) を呼び出すのに使用されます。

~/templates/network.yaml を使用して、ネットワーク設定のパラメーターを定義したり、カスタマイズされたネットワーク用の heat テンプレートを追加したりすることができます。詳しくは、『Advanced Overcloud Customization』の「 Custom network environment file 」を参照してください。

第6章 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。

6.1. 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
```

6.2. 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
```

第7章 オーバークラウドのデプロイ

前提条件

- その他すべての Ceph の設定に、別のベース環境ファイルを 1 つ (または複数) 使用している (例: **/home/stack/templates/storage-config.yaml**)。詳しくは、「 ストレージサービスのカスタマイズ 」および「 環境ファイルのサンプル： Ceph クラスターの作成」を参照してください。
- ベース環境ファイルで、各ロールに割り当てるノード数を定義している。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。
- アンダークラウドのインストール時に、**undercloud.conf** ファイルで **generate_service_certificate=false** と設定している。設定しない場合は、「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。

重要

RHOSP HCI 環境をデプロイする際には、インスタンス HA を有効にしないでください。Ceph を使用したハイパーコンバージド RHOSP デプロイメントでインスタンス HA を使用する場合は、Red Hat の担当者にお問い合わせください。

手順

以下のコマンドを実行して HCI オーバークラウドをデプロイします。

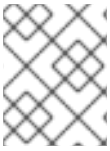
```
$ openstack overcloud deploy --templates \
  -p /usr/share/openstack-tripleo-heat-templates/plan-samples/plan-environment-derived-
params.yaml \
  -r /home/stack/templates/roles_data.yaml \
  -e /home/stack/templates/ports.yaml \
  -e /home/stack/templates/environment-rhel-registration.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml \
  -e /home/stack/templates/storage-config.yaml \
  -e /home/stack/templates/storage-container-config.yaml \
  -e /home/stack/templates/network.yaml \
  [-e /home/stack/templates/ceph-backfill-recovery.yaml \ ]
  [-e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-sriov.yaml \]
  [-e /home/stack/templates/network-environment.yaml \ ]
  [-e <additional environment files for your planned overcloud deployment> \ ]
  --ntp-server pool.ntp.org
```

詳細は以下のようになります。

引数	説明
--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。

引数	説明
-p /usr/share/openstack-tripleo-heat-templates/plan-samples/plan-environment-derived-params.yaml	派生パラメーターのワークフローをデプロイメント中に実行して、ハイパーコンバージドのデプロイメントに確保するメモリーおよびCPU の容量を計算することを指定します。
-r /home/stack/templates/roles_data.yaml	「ハイパーコンバージドノード向けのオーバークラウドロールの準備」の手順で作成した、ComputeHCI ロールが含まれるカスタムのロール定義ファイルを指定します。
-e /home/stack/templates/ports.yaml	「ハイパーコンバージドノード向けのオーバークラウドロールの準備」の手順で作成した、ComputeHCI ロールのポートを設定する環境ファイルを追加します。
-e /home/stack/templates/environment-rhel-registration.yaml	「rhsm コンポーザブルサービスを使用したオーバークラウドの登録」で説明するように、オーバークラウドノードを登録する環境ファイルを 追加します。
-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml	すべてのデフォルト設定でコンテナー化された Red Hat Ceph クラスターをデプロイするベース環境ファイルを追加します。詳しくは、『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』を 参照してください。
-e /home/stack/templates/storage-config.yaml	その他すべての Ceph 設定を定義するカスタム環境ファイルを追加します。環境ファイルの例の詳細は、「環境ファイルのサンプル：Ceph クラスターの作成」を参照してください。このサンプル環境ファイルは、使用するフレーバーおよびロールごとに割り当てるノード数も指定します。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。
-e /home/stack/templates/storage-container-config.yaml	各 Ceph OSD ストレージコンテナー用に CPU およびメモリーを確保します (「Ceph 用 CPU/メモリーリソースの確保」を参照)。
-e /home/stack/templates/network.yaml	「ストレージ管理ネットワークポートの NIC へのマッピング」の手順で作成した環境ファイルを追加します。
-e /home/stack/templates/ceph-backfill-recovery.yaml	(オプション)「Ceph のバックフィルおよびリカバリー操作の削減」で作成した環境ファイルを追加します。

引数	説明
-e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-sriov.yaml	(オプション) Single-Root Input/Output Virtualization (SR-IOV) 用の環境ファイルを追加します。
-e /home/stack/templates/network-environment.yaml	(オプション) SR-IOV ネットワーク設定を適用する環境ファイルを追加します。
-e <environment file>	(オプション) 予定しているオーバークラウドデプロイメント用のその他の環境ファイルがあれば追加します。
--ntp-server pool.ntp.org	NTP サーバーを設定します。



注記

現在、HCI でサポートされるネットワーク機能仮想化 (NFV) の実装は SR-IOV だけです。

デプロイメントオプションの全一覧を表示するには、以下のコマンドを実行します。

```
$ openstack help overcloud deploy
```

デプロイメントオプションの詳細は、「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。

ヒント

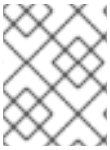
アンサー ファイルを使用して、デプロイメントに追加する環境ファイルを指定することも可能です。詳しくは、『director のインストールと使用方法』の「オーバークラウド作成時の環境ファイルの 追加」を参照してください。

第8章 ハイパーコンバージドノードのスケーリング

HCI ノードをスケールアップまたはスケールダウンする場合、コンピュートノードまたは Ceph Storage ノードのスケーリングと同じ原則および手法が適用されます。

8.1. スケールアップ

HCI 環境のハイパーコンバージドノードをスケールアップするには、ハイパーコンバージドノードではないノードをスケールアップするのと同じ手順に従います。詳しくは、『Director Installation and Usage』の「 Adding nodes to the overcloud 」を参照してください。



注記

新規ノードをタグ付けする場合には、必ず適切なフレーバーを使用するようにしてください。

OSD を Ceph Storage クラスターに追加して HCI ノードをスケールアップする方法は、『**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**』の「 OSD の Ceph Storage ノードへ の追加」を参照してください。

8.2. スケールダウン

手順

1. HCI ノード上の Ceph OSD サービスを無効にして、リバランスします。HCI ノードまたは Ceph Storage ノードを削除する際に、director は Red Hat Ceph Storage クラスターを自的にリバランスしないので、このステップが必要となります。

1. HCI ノードからインスタンスを移行します。詳しくは、「 コンピュートノード間の仮想マシンインスタンスの移行 」を参照してください。

2. ノード上の Compute サービスを無効にして、そのノードが新規インスタンスの作成に使用されるのを防ぎます。

3. オーバークラウドからノードを削除します。

ステップ 3 および 4 については、「 コンピュートノードの削除 」を参照してください。

付録A 付録

A.1. COMPUTE の CPU およびメモリーの計算

以降のサブセクションで、OpenStack Workflow がどのように CPU およびメモリーの最適設定を計算するかについて説明します。

A.1.1. NovaReservedHostMemory

NovaReservedHostMemory パラメーターは、ホストノードに確保するメモリー容量 (MB 単位) を設定します。ハイパーコンバージドノードの適切な値を決定するには、各 OSD が 3 GB のメモリーを消費すると仮定します。メモリーが 256 GB で OSD が 10 のノードの場合には、Ceph に 30 GB のメモリーを割り当てて、226 GB を Compute に残します。このメモリー容量のノードでは、たとえば、それぞれ 2 GB のメモリーを使用するインスタンスを 113 台ホストすることができます。

ただし、**ハイパーバイザー** 用に、インスタンス 1 台あたりの追加のオーバーヘッドを考慮する必要があります。このオーバーヘッドが 0.5 GB と仮定すると、同じノードでは、90 インスタンスしかホストできません。これは、226 GB を 2.5 GB で割ったものです。ホストノードに確保するメモリー容量 (Compute サービスが使用してはならないメモリー) は以下のように算出します。

(In * Ov) + (Os * RA)

ここで、

- **In**: インスタンス数
- **Ov**: インスタンス 1 台あたりに必要なオーバヘッド用のメモリー容量
- **Os**: ノード上の OSD 数
- **RA**: 各 OSD に割り当てる必要のある RAM 容量

90 台のインスタンスの場合には、(90 * 0.5) + (10 * 3) = 75 GB という計算になります。Compute サービスには、この値を MB 単位で指定します (75000)。

以下の Python コードにより、この計算を行うことができます。

```
left_over_mem = mem - (GB_per_OSD * osds)
number_of_guests = int(left_over_mem /
    (average_guest_size + GB_overhead_per_guest))
nova_reserved_mem_MB = MB_per_GB * (
    (GB_per_OSD * osds) +
    (number_of_guests * GB_overhead_per_guest))
```

A.1.2. cpu_allocation_ratio

Compute スケジューラーは、インスタンスをデプロイするコンピュートノードを選択する際に **cpu_allocation_ratio** を使用します。デフォルトでは、この値は **16.0** (16:1) です。1 台のノードに 56 コアある場合には、Compute スケジューラーは 1 台のノードで 896 の仮想 CPU を使用するのに十分なインスタンスをスケジュールすることになります。この値を超えると、そのノードはそれ以上インスタンスをホストできないと見なされます。

ハイパーコンバージドノードに適切な **cpu_allocation_ratio** を決定するには、各 Ceph OSD が最小で 1 コアを使用すると仮定します (ワークロードが I/O 集中型で、SSD を使用しないノード上にある場合を

除く)。56 コア、10 OSD のノードでは、この計算で 46 コアが Compute に確保されます。各インスタンスが割り当てられた CPU の 100 パーセント使用すると、この比率は単にインスタンスの仮想 CPU 数をコア数で除算した値となります (46 / 56 = 0.8)。ただし、インスタンスは通常割り当てられた CPU を 100 パーセント使用することはないため、ゲストに必要な仮想 CPU 数を決定する際には、予想される使用率を考慮に入れて、cpu_allocation_ratio を高くすることができます。

したがって、インスタンスが仮想 CPU の 10 パーセント (または 0.1) のみを使用すると予想できる場合には、インスタンス用の仮想 CPU は 46 / 0.1 = 460 の式で示すことができます。この値をコア数 (56) で除算すると、比率は約 8 に増えます。

以下の Python コードにより、この計算を行うことができます。

```
cores_per_OSD = 1.0
average_guest_util = 0.1 # 10%
nonceph_cores = cores - (cores_per_OSD * osds)
guest_vCPUs = nonceph_cores / average_guest_util
cpu_allocation_ratio = guest_vCPUs / cores
```