Red Hat OpenStack Platform 17.1

分散コンピュートノード (DCN) アーキテクチャーのデプロイ

Red Hat OpenStack Platform のエッジおよびストレージ設定

Last Updated: 2024-06-12

Red Hat OpenStack Platform 17.1 分散コンピュートノード (DCN) アーキテクチャーのデプロイ

Red Hat OpenStack Platform のエッジおよびストレージ設定

OpenStack Team
rhos-docs@redhat.com

法律上の通知

概要

分散コンピュートノード (DCN) アーキテクチャーと共に Red Hat OpenStack Platform (RHOSP) をデプロイして、heat スタックの分離によりエッジサイトの運用性を向上させます。それぞれのサイトには、Image サービス (glance) のマルチストア用に、独自の Ceph ストレージバックエンドを設定することができます。

目次

多様性を受け入れるオープンソースの強化 4

RED HAT ドキュメントへのフィードバック (英語のみ) 5

第1章 DCN について 6
1.1. 分散コンピュートノードのアーキテクチャーに必要なソフトウェア 6
1.2. マルチスタック設計 7
1.3. DCN ストレージ 7
1.4. DCN エッジ 7

第2章 分散コンピュートノード (DCN) デプロイメントのプランニング 9
2.1. DCN アーキテクチャーのストレージに関する考慮事項 9
2.2. DCN アーキテクチャーのネットワークに関する考慮事項 9

第3章 アンダークラウドでのルーティング対応スパイン/リーフの設定 12
3.1. スパイン/リーフ用のプロビジョニングネットワークの設定 12
3.2. DHCP リレーの設定 14
3.3. リーフノードのロールの指定 17
3.4. ベアメタルノードのポートからコントロールプレーンのネットワークセグメントへのマッピング 18
3.5. スパイン/リーフ型のプロビジョニングネットワークへの新規リーフの追加 19

第4章 DCN デプロイメント用オーバークラウドテンプレートの準備 22
4.1. 個別の HEAT スタックを使用するための前提条件 22
4.2. 個別 HEAT スタックのデプロイメント例の制限 22
4.3. 個別 HEAT スタックのデプロイメントの設計 22
4.4. 個別の HEAT スタックの管理 23
4.5. コンテナーイメージの取得 23
4.6. エッジサイト用の高速データパスロールの作成 24

第5章 中央サイトのインストール 26
5.1. エッジストレージを持たない中央コントローラーのデプロイ 26
5.2. ストレージが設定された中央サイトのデプロイ 29
5.3. 外部 CEPH の統合 32

第6章 ストレージを使用しないエッジのデプロイ 37
6.1. ストレージを使用しない DCN エッジサイトのアーキテクチャー 37
6.2. ストレージを持たないエッジノードのデプロイ 38
6.3. エッジサイトでの特定イメージ種別の除外 40

第7章 エッジサイトでのストレージのデプロイ 42
7.1. ストレージを使用したエッジデプロイメントのロール 42
7.2. ストレージを備えた DCN エッジサイトのアーキテクチャー 43
7.3. ハイパーコンバージドストレージを備えた DCN エッジサイトのアーキテクチャー 44
7.4. ハイパーコンバージドストレージを使用したエッジサイトのデプロイメント 45
7.5. エッジでのインストール済み RED HAT CEPH STORAGE クラスターの使用 48
7.6. 中央サイトの更新 50
7.7. DCN への RED HAT CEPH STORAGE DASHBOARD のデプロイ 52

第8章 エッジサイトでのネットワークトラフィックの負荷分散 55
8.1. LOAD-BALANCING サービスアベイラビリティーゾーンのネットワークリソースの作成 55
8.2. LOAD-BALANCING サービスのアベイラビリティーゾーンの作成 57
8.3. アベイラビリティーゾーンでのロードバランサーの作成 61

第9章 DISTRIBUTEDCOMPUTEHCI ノードの置き換え 64

9.1. RED HAT CEPH STORAGE サービスの削除 64
9.2. IMAGE サービス (GLANCE) サービスの削除 66
9.3. BLOCK STORAGE (CINDER) サービスの削除 67
9.4. DISTRIBUTEDCOMPUTEHCI ノードの削除 67
9.5. 削除された DISTRIBUTEDCOMPUTEHCI ノードの置き換え 68
9.6. 置き換えられた DISTRIBUTEDCOMPUTEHCI ノードの機能の検証 69
9.7. DISTRIBUTEDCOMPUTEHCI の状態ダウンのトラブルシューティング 71

第10章 KEY MANAGER を含むデプロイ 73
10.1. KEY MANAGER が設定されたエッジサイトのデプロイ 73

第11章 GLANCE イメージの NOVA への事前キャッシュ 74
11.1. TRIPLEO_NOVA_IMAGE_CACHE.YML ANSIBLE PLAYBOOK の実行 74
11.2. パフォーマンスに関する考慮事項 75
11.3. DCN サイトへのイメージ配布の最適化 76
11.4. NOVA-CACHE クリーンアップの設定 76

第12章 DCN への TLS-E の適用 78
12.1. TLS-E を設定した分散コンピュートノードアーキテクチャーのデプロイ 78

第13章 外部アクセス用 CEPH キーの作成 80
13.1. 外部アクセス用 CEPH キーの作成 80
13.2. 外部 CEPH キーの使用 81

付録A デプロイメントの移行オプション 83
A.1. エッジストレージの検証 83
A.2. スパイン/リーフ型デプロイメントへの移行 86
A.3. マルチスタックデプロイメントへの移行 87
A.4. エッジサイト間のバックアップおよびリストア 87
A.5. DCN 環境でのオーバークラウドの導入と準備 88

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、用語の置き換えは、今後の複数のリリースにわたって段階的に実施されます。詳細は、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

RED HAT ドキュメントへのフィードバック (英語のみ)

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

Jira でドキュメントのフィードバックを提供する

ドキュメントに関するフィードバックを提供するには、Create Issue フォームを使用します。Red Hat OpenStack Platform Jira プロジェクトで Jira Issue が作成され、フィードバックの進行状況を追跡できます。

1. Jira にログインしていることを確認してください。Jira アカウントをお持ちでない場合は、アカウントを作成してフィードバックを送信してください。
2. Create Issue をクリックして、**Create Issue** ページを開きます。
3. **Summary** フィールドと **Description** フィールドに入力します。**Description** フィールドに、ドキュメントの URL、章またはセクション番号、および問題の詳しい説明を入力します。フォーム内の他のフィールドは変更しないでください。
4. **Create** をクリックします。

第1章 DCN について

分散コンピュートノード (DCN) アーキテクチャーは、共通の集中型コントロールプレーンを共有する一方で、リモートコンピュートノードおよびストレージノードをリモートでデプロイできるようにするエッジユースケース向けです。DCN アーキテクチャーでは、パフォーマンスを向上させるために、ワークロードを戦略的に運用上のニーズの近傍に配置することができます。

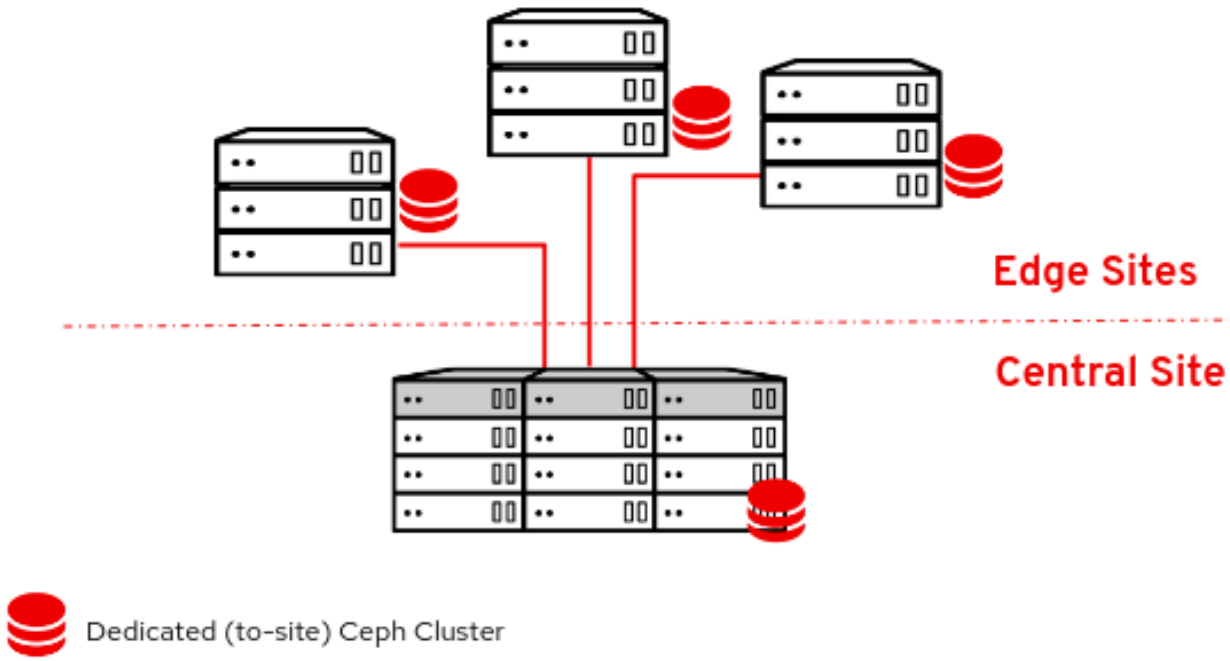
中央サイトは任意のロールで設定することができますが、最低でも 3 つのコントローラーが必要です。Compute ノードは、中央サイト以外にエッジサイトに設定することができます。

DCN のアーキテクチャーは、ハブとスポークによるルーティング対応ネットワークのデプロイメントです。DCN は、Red Hat OpenStack Platform director を使用した、ルーティング対応プロビジョニングおよびコントロールプレーンネットワーク向けのスパイン/リーフ型デプロイメントと類似しています。

- ハブは、コアルーターおよびデータセンターゲートウェイ (DC-GW) が含まれる中央サイトです。
- スポークはリモートのエッジサイトまたはリーフです。

エッジロケーションにはコントローラーがないため、Red Hat OpenStack Platform の従来のデプロイメントとはアーキテクチャー的に異なります。

- コントロールプレーンサービスは、中央サイトでリモートで実行されます。
- Pacemaker はインストールされません。
- Block Storage サービス (cinder) はアクティブ/アクティブモードで実行されます。
- etcd は分散ロックマネージャー (DLM) としてデプロイされます。



1.1. 分散コンピュートノードのアーキテクチャーに必要なソフトウェア

以下の表は、分散コンピュートノード (DCN) アーキテクチャーに Red Hat OpenStack Platform をデプロイするのに必要なソフトウェアおよび最小バージョンを示しています。

プラットフォーム	バージョン	任意
Red Hat Enterprise Linux	9.2	いいえ
Red Hat OpenStack Platform	17.1	いいえ
Red Hat Ceph Storage	5	はい

1.2. マルチスタック設計

DCN 設計を使用して Red Hat OpenStack Platform (RHOSP) をデプロイする場合、複数のスタックデプロイメントおよび管理に Red Hat director の機能を使用して、各サイトを個別のスタックとしてデプロイします。

デプロイメントが Red Hat OpenStack Platform 13 からのアップグレードでない限り、DCN アーキテクチャーを単一スタックとして管理する運用はサポートされません。既存のスタックを分割する手法はサポート対象外ですが、既存のデプロイメントにスタックを追加することができます。詳細は、「マルチスタックデプロイメントへの移行」 を参照してください。

中央サイトは RHOSP の従来のスタックデプロイメントですが、Compute ノードまたは Red Hat Ceph ストレージを中央スタックと共にデプロイする必要はありません。

DCN では、それぞれの場所を別のアベイラビリティーゾーン (AZ) としてデプロイします。

1.3. DCN ストレージ

ストレージなしで、またはハイパーコンバージドノード上の Ceph と共に、それぞれのエッジサイトをデプロイすることができます。デプロイするストレージは、デプロイするサイト専用のストレージです。

DCN アーキテクチャーでは、Glance のマルチストアが使用されます。ストレージなしでデプロイされたエッジサイトの場合、追加のツールを利用できるため、イメージを Compute サービス (nova) キャッシュにキャッシュして保存することができます。nova に glance イメージをキャッシュすることで、WAN リンクでイメージをダウンロードするプロセスを回避することで、インスタンスのブート時間が短縮されます。詳細は、11章*glance イメージの nova への事前キャッシュ* を参照してください。

1.4. DCN エッジ

分散コンピュートノードアーキテクチャーでは、コントロールノードを中央サイトにデプロイし、これらのコントローラーを使用して地理的に分散したエッジサイトを管理します。エッジサイトをデプロイするときは、コンピュートノードのみをデプロイするため、エッジサイトは Red Hat OpenStack Platform の従来のデプロイとはアーキテクチャーが異なります。エッジサイトでインスタンスを起動すると、必要なイメージがローカルの Image Service (glance) ストアに自動的にコピーされます。インスタンスの起動時に時間を節約するために、glance マルチストアを使用して central のイメージストアからエッジサイトにイメージをコピーできます。詳細は、複数のストアを持つ Image Service を参照してください。

エッジサイト:

- コントロールプレーンサービスは、中央サイトでリモートで実行されます。
- Pacemaker は DCN サイトでは実行されません。

- Block Storage サービス (cinder) はアクティブ/アクティブモードで実行されます。
- etcd は分散ロックマネージャー (DLM) としてデプロイされます。

第2章 分散コンピュートノード (DCN) デプロイメントのプランニング

DCN アーキテクチャーを計画する際に、必要なテクノロジーが利用可能で、サポートされていることを確認します。

2.1. DCN アーキテクチャーのストレージに関する考慮事項

現在、DCN アーキテクチャー向けには以下の機能はサポートされていません。

- エッジサイト間でのボリュームスナップショットのコピー。ボリュームからイメージを作成し、glance を使用してイメージをコピーすることで、これに対処することができます。イメージをコピーしたら、そこからボリュームを作成することができます。
- エッジサイトでの Ceph Rados Gateway (RGW)
- エッジサイトでの CephFS
- エッジサイトでのインスタンスの高可用性 (HA)
- サイト間での RBD ミラーリング
- エッジサイト間、または中央サイトからエッジサイトへの、ライブまたはコールドのインスタンスの移行。サイト境界内でインスタンスを移行することもできます。サイト間でイメージを移動するには、イメージのスナップショットを作成し、**glance image-import** を使用する必要があります。詳細は Confirming image snapshots can be created and copied between sites を確認を参照してください。

さらに、以下の点を考慮する必要があります。

- イメージをエッジサイトにコピーする前に、中央サイトにイメージをアップロードする必要があります。各イメージのコピーは、中央サイトの Image サービス (glance) に存在している必要があります。
- Image、Compute、Block Storage サービスに、RBD ストレージドライバーを使用する必要があります。
- それぞれのサイトで、一意のアベイラビリティーゾーンを割り当て、NovaComputeAvailabilityZone および CinderStorageAvailabilityZone パラメーターに同じ値を使用します。
- エッジサイトからセントラルサイトへ、またはその逆でオフラインのボリュームを移行することができます。エッジサイト間でボリュームを直接移行することはできません。

2.2. DCN アーキテクチャーのネットワークに関する考慮事項

現在、DCN アーキテクチャー向けには以下の機能はサポートされていません。

- DPDK ノード上の DHCP
- TC Flower ハードウェアオフロード向け conntrack

TC Flower ハードウェアオフロード向け conntrack は、テクノロジープレビューとして DCN で利用可能であるため、これらのソリューションを組み合わせて使用した場合、Red Hat では全面的にはサポートしていません。この機能は DCN ではテスト用途にのみ使用すべきで、実稼働環境にデプロイすべき

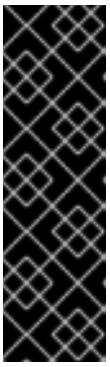
ではありません。テクノロジープレビュー機能についての詳しい情報は、対象範囲の詳細を参照してください。

以下の ML2/OVS テクノロジーが完全にサポートされています。

- DPDK ノードでの DHCP を使用しない OVS-DPDK
- SR-IOV
- TC Flower ハードウェアオフロード (conntrack を使用しない)
- Neutron アベイラビリティーゾーン (AZ) とエッジサイトのネットワークノードの組み合わせ (1 サイトにつき 1 AZ)
- ルーティング対応プロバイダーネットワーク

以下の ML2/OVN ネットワークテクノロジーが完全にサポートされます。

- DPDK ノードでの DHCP を使用しない OVS-DPDK
- SR-IOV (DHCP を使用しない)
- TC Flower ハードウェアオフロード (conntrack を使用しない)
- ルーティング対応プロバイダーネットワーク
- Neutron AZ がサポートされる OVN GW (ネットワークノード)



重要

OVNCMSOptions: 'enable-chassis-as-gw' を設定し、**OVNAvailabilityZone** パラメーターに 1 つ以上の AZ 値を指定して、すべてのルーターゲートウェイポートが必ず OpenStack コントローラーノード上に存在することを確認してください。これらのアクションを実行すると、ルーターはすべてのシャーシをルーターゲートウェイポートの潜在的なホストとしてスケジュールできなくなります。詳細は、**Red Hat OpenStack Platform ネットワークの設定** の ML2/OVN を使用してネットワークサービスアベイラビリティゾーンを設定する を参照してください。

さらに、以下の点を考慮する必要があります。

- ネットワークレイテンシー: 許容可能な性能を維持するための、ラウンドトリップタイム (RTT) で測定されるレイテンシーと予想される同時 API 操作の数のバランス。最大 TCP/IP スループットは、RTT と逆比例します。カーネル TCP パラメーターを調整することで、高帯域幅の高レイテンシー接続の問題を軽減できま。クロスサイト通信が 100 ミリ秒を超える場合は Red Hat サポートにお問い合わせください。
- ネットワークドロップアウト: エッジサイトで一時的に中央サイトへのネットワーク接続が失われると、その間は該当するエッジサイトで OpenStack コントロールプレーン API または CLI 操作を実行することができません。たとえば、エッジサイトの Compute ノードは、インスタンスのスナップショットの作成や認証トークンの発行、イメージの削除ができなくなります。この接続喪失の期間中、全般的な OpenStack コントロールプレーン API および CLI 操作は引き続き実施可能で、ネットワーク接続が機能しているその他のエッジサイトへの対応を続けることができます。イメージタイプ: Ceph ストレージと共に DCN アーキテクチャーをデプロイする場合は、raw 形式のイメージを使用する必要があります。

- イメージのサイズ:
  - オーバークラウドノードのイメージ: オーバークラウドノードのイメージは中央のアンダークラウドノードからダウンロードされます。プロビジョニング時に、これらのイメージの大きなファイルが必要なすべてのネットワークを通じて中央サイトからエッジサイトに転送される可能性があります。
  - インスタンスのイメージ: エッジサイトにブロックストレージがない場合には、初回使用時に Image サービスのイメージが WAN を通過します。その後のすべての使用のために、イメージは目的のエッジノードにローカルにコピーまたはキャッシュされます。glance イメージにはサイズの制限はありません。転送時間は、利用可能な帯域幅およびネットワークレイテンシーにより変動します。
    ブロックストレージがエッジサイトにある場合は、エッジサイトでのブート時間短縮のために、イメージが WAN を通じて非同期にコピーされます。

- プロバイダーネットワーク: これが DCN デプロイメントの推奨ネットワーク設定です。リモートサイトでプロバイダーネットワークを使用する場合は、利用可能なネットワークのアタッチ先に関して、Networking サービス (neutron) が何らかの制約を設けたりチェックを行ったりしない点に注意する必要があります。たとえば、エッジサイト A でしかプロバイダーネットワークを使用しない場合には、エッジサイト B では決してプロバイダーネットワークにアタッチしないようにする必要があります。これは、プロバイダーネットワークを Compute ノードにバインドする際に、プロバイダーネットワークに関するチェックが行われないためです。

- サイト固有のネットワーク: 特定のサイトに固有なネットワークを使用している場合には、DCN のネットワーク設定に制約が生じます。Compute ノードと共に集中 neutron コントローラーをデプロイする場合には、neutron では特定の Compute ノードをリモートノードとして識別するトリガーがありません。したがって、Compute ノードは他の Compute ノードのリストを取得し、自動的にそれぞれのノード間でトンネルを形成します。トンネルは、エッジサイト/エッジサイト間およびエッジサイト/中央サイト間で形成されます。VXLAN または Geneve を使用している場合には、すべてのサイトの全 Compute ノードが、他のすべての Compute ノードおよびコントローラーノードとトンネルを形成します (それらがローカルかリモートかにかかわらず)。すべてのノードで同じ neutron ネットワークを使用していれば、これは問題とはなりません。VLAN を使用している場合の neutron 設定では、すべての Compute ノードが同じブリッジマッピングを持ち、すべての VLAN が各サイトで利用可能でなければなりません。

- 追加のサイト: 中央サイトから追加のリモートサイトに拡張する必要がある場合には、Red Hat OpenStack Platform director で openstack CLI を使用して、新たなネットワークセグメントおよびサブネットを追加することができます。

- エッジサーバーが事前にプロビジョニングされていない場合は、ルーティングされたセグメントにイントロスペクションおよびプロビジョニング用の DHCP リレーを設定する必要があります。

- ルーティングは、クラウド上、または各エッジサイトをハブに接続するネットワークインフラストラクチャー内のいずれかに設定する必要があります。それぞれのサイトについて個別に、各 Red Hat OpenStack Platform クラスターネットワーク (外部、内部 API 等) の L3 サブネットを割り当てるネットワーク設計を実装する必要があります。

第3章 アンダークラウドでのルーティング対応スパイン/リーフの設定

本項では、コンポーザブルネットワークを使用するルーティング対応のスパイン/リーフを取り入れるための、アンダークラウド設定方法のユースケースについて説明します。

3.1. スパイン/リーフ用のプロビジョニングネットワークの設定

スパイン/リーフインフラストラクチャー用のプロビジョニングネットワークを設定するには、**undercloud.conf** ファイルを編集して、以下の手順で説明する該当パラメーターを設定します。

手順

1. アンダークラウドに **stack** ユーザーとしてログインします。
2. **undercloud.conf** ファイルがまだない場合には、サンプルのテンプレートファイルをコピーします。

```
[stack@director ~]$ cp /usr/share/python-tripleoclient/undercloud.conf.sample
~/undercloud.conf
```

3. **undercloud.conf** ファイルを編集します。
4. **[DEFAULT]** セクションに以下の値を設定します。
   a. **local_ip** を **leaf0** 上のアンダークラウド IP に設定します。

```
local_ip = 192.168.10.1/24
```

   b. **undercloud_public_host** をアンダークラウドの外部向け IP アドレスに設定します。

```
undercloud_public_host = 10.1.1.1
```

   c. **undercloud_admin_host** をアンダークラウドの管理用 IP アドレスに設定します。この IP アドレスは、通常 leaf0 上にあります。

```
undercloud_admin_host = 192.168.10.2
```

   d. **local_interface** を、ローカルネットワーク用にブリッジを設定するインターフェイスに設定します。

```
local_interface = eth1
```

   e. **enable_routed_networks** を **true** に設定します。

```
enable_routed_networks = true
```

   f. **subnets** パラメーターを使用してサブネットのリストを定義します。ルーティング対応のスパイン/リーフ内の各 L2 セグメントにサブネットを 1 つ定義します。

```
subnets = leaf0,leaf1,leaf2
```

g. **local_subnet** パラメーターを使用して、アンダークラウドにローカルな物理 L2 セグメントに関連付けられるサブネットを指定します。

```
local_subnet = leaf0
```

h. **undercloud_nameservers** の値を設定します。

```
undercloud_nameservers = 10.11.5.19,10.11.5.20
```

ヒント

アンダークラウドのネームサーバーに使用する DNS サーバーの現在の IP アドレスは、/etc/resolv.conf を参照して確認することができます。

5. **subnets** パラメーターで定義するサブネットごとに、新規セクションを作成します。

```
[leaf0]
cidr = 192.168.10.0/24
dhcp_start = 192.168.10.10
dhcp_end = 192.168.10.90
inspection_iprange = 192.168.10.100,192.168.10.190
gateway = 192.168.10.1
masquerade = False

[leaf1]
cidr = 192.168.11.0/24
dhcp_start = 192.168.11.10
dhcp_end = 192.168.11.90
inspection_iprange = 192.168.11.100,192.168.11.190
gateway = 192.168.11.1
masquerade = False

[leaf2]
cidr = 192.168.12.0/24
dhcp_start = 192.168.12.10
dhcp_end = 192.168.12.90
inspection_iprange = 192.168.12.100,192.168.12.190
gateway = 192.168.12.1
masquerade = False
```

6. **undercloud.conf** ファイルを保存します。

7. アンダークラウドをインストールするコマンドを実行します。

```
[stack@director ~]$ openstack undercloud install
```

この設定により、プロビジョニングネットワークまたはコントロールプレーン上に 3 つのサブネットが作成されます。オーバークラウドは、各ネットワークを使用して対応する各リーフ内にシステムをプロビジョニングします。

アンダークラウドへの DHCP 要求が適切にリレーされるようにするには、DHCP リレーを設定する必要がある場合があります。

3.2. DHCP リレーの設定

DHCP リレーサービスは、リクエストを転送したいリモートネットワークセグメントに接続されているスイッチ、ルーター、またはサーバーで実行します。

注記

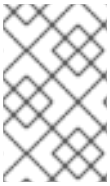
アンダークラウド上で DHCP リレーサービスを実行しないでください。

アンダークラウドは、プロビジョニングネットワーク上の 2 つの DHCP サーバーを使用します。

- イントロスペクション DHCP サーバー。
- プロビジョニング DHCP サーバー。

DHCP リレーは、アンダークラウド上の両方の DHCP サーバーに DHCP リクエストを転送するように設定する必要があります。

UDP ブロードキャストに対応するデバイスで UDP ブロードキャストを使用して、アンダークラウドのプロビジョニングネットワークが接続されている L2 ネットワークセグメントに DHCP 要求をリレーすることができます。または、DHCP 要求を特定の IP アドレスにリレーする UDP ユニキャストを使用することができます。



注記

特定のデバイス種別での DHCP リレーの設定は、本書の対象外となっています。本ガイドでは参考として、ISC DHCP ソフトウェアの実装を使用した DHCP リレー設定の例を説明します。詳細は、dhcrelay(8) の man ページを参照してください。

重要

DHCP オプション 79 は、一部のリレー、特に DHCPv6 アドレスを提供するリレー、および元の MAC アドレスを渡さないリレーに必要です。詳細は、RFC6939 を参照してください。

ブロードキャスト DHCP リレー

この方法では、UDP ブロードキャストトラフィックを使用して DHCP 要求を、DHCP サーバーが存在する L2 ネットワークセグメントにリレーします。ネットワークセグメント上のすべてのデバイスがブロードキャストトラフィックを受信します。UDP ブロードキャストを使用する場合は、アンダークラウド上の両方の DHCP サーバーがリレーされた DHCP 要求を受信します。実装に応じて、インターフェイスまたは IP ネットワークアドレスを指定して設定できます。

インターフェイス

DHCP 要求がリレーされる L2 ネットワークセグメントに接続されるインターフェイスを指定します。

IP ネットワークアドレス

DHCP 要求がリレーされる IP ネットワークのネットワークアドレスを指定します。

ユニキャスト DHCP リレー

この方法では、UDP ユニキャストトラフィックを使用して DHCP 要求を特定の DHCP サーバーにリレーします。UDP ユニキャストを使用する場合には、DHCP リレーを提供するデバイスが、アンダークラウド上のイントロスペクション用に使用されるインターフェイスに割り当てられた IP アドレス

と、**ctlplane** ネットワーク用の DHCP サービスをホストするために OpenStack Networking (neutron) サービスが作成するネットワーク名前空間の IP アドレスの両方に対して、DHCP 要求をリレーするように設定する必要があります。

イントロスペクションに使用されるインターフェイスは、**undercloud.conf** ファイルで **inspection_interface** として定義されるインターフェイスです。このパラメーターを設定していない場合には、アンダークラウドのデフォルトインターフェイスは **br-ctlplane** になります。

注記

br-ctlplane インターフェイスをイントロスペクションに使用するのは一般的です。**undercloud.conf** ファイルで **local_ip** として定義する IP アドレスは、**br-ctlplane** インターフェイス上にあります。

Neutron DHCP 名前空間に確保される IP アドレスは、**undercloud.conf** ファイルの **local_subnet** で設定する IP 範囲内で利用可能な最初のアドレスです。IP 範囲内の最初のアドレスは、設定の **dhcp_start** で定義するアドレスです。たとえば、以下の設定を使用する場合、**192.168.10.10** がその IP アドレスになります。

```
[DEFAULT]
local_subnet = leaf0
subnets = leaf0,leaf1,leaf2

[leaf0]
cidr = 192.168.10.0/24
dhcp_start = 192.168.10.10
dhcp_end = 192.168.10.90
inspection_iprange = 192.168.10.100,192.168.10.190
gateway = 192.168.10.1
masquerade = False
```

警告

DHCP 名前空間の IP アドレスは自動的に割り当てられます。多くの場合、これは IP 範囲の最初のアドレスになります。これを確認するには、アンダークラウドで以下のコマンドを実行します。

```
$ openstack port list --device-owner network:dhcp -c "Fixed IP Addresses"
+-----------------------------------------------------------------------+
| Fixed IP Addresses                                                    |
+-----------------------------------------------------------------------+
| ip_address='192.168.10.10', subnet_id='7526fbe3-f52a-4b39-a828-
ec59f4ed12b2' |
+-----------------------------------------------------------------------+
$ openstack subnet show 7526fbe3-f52a-4b39-a828-ec59f4ed12b2 -c name
+-------+--------+
| Field | Value  |
+-------+--------+
| name  | leaf0  |
+-------+--------+
```

dhcrelay の設定例

以下の例では、**dhcp** パッケージの **dhcrelay** コマンドは以下の設定を使用します。

- DHCP の受信要求をリレーするインターフェイスは **eth1**、**eth2**、**eth3** です。
- ネットワークセグメント上のアンダークラウドの DHCP サーバーが接続されているインターフェイスは **eth0** です。
- イントロスペクションに使用される DHCP サーバーがリッスンしている IP アドレスは **192.168.10.1** です。
- プロビジョニングに使用される DHCP サーバーがリッスンしている IP アドレスは **192.168.10.10** です。

これで、**dhcrelay** コマンドは以下のようになります。

- **dhcrelay** バージョン 4.2.x:

```
$ sudo dhcrelay -d --no-pid 192.168.10.10 192.168.10.1 \
  -i eth0 -i eth1 -i eth2 -i eth3
```

- **dhcrelay**バージョン 4.3.x 以降:

```
$ sudo dhcrelay -d --no-pid 192.168.10.10 192.168.10.1 \
  -iu eth0 -id eth1 -id eth2 -id eth3
```

Cisco IOS ルーティングスイッチの設定例

この例では、次のタスクを実行するために、以下に示す Cisco IOS 設定を使用しています。

- プロビジョニングネットワークに使用する VLAN を設定する。
- リーフの IP アドレスを追加する。
- IP アドレス **192.168.10.1** をリッスンするイントロスペクション用 DHCP サーバーに、UDP および BOOTP 要求を転送する。
- IP アドレス **192.168.10.10** をリッスンするプロビジョニング用 DHCP サーバーに、UDP および BOOTP 要求を転送する。

```
interface vlan 2
ip address 192.168.24.254 255.255.255.0
ip helper-address 192.168.10.1
ip helper-address 192.168.10.10
!
```

これでプロビジョニングネットワークの設定が完了したので、残りのオーバークラウドリーフネットワークを設定することができます。

3.3. リーフノードのロールの指定

各リーフネットワークのそれぞれのロールには、フレーバーとロールの割り当てが必要です。これにより、ノードを対応するリーフにタグ付けすることができます。各フレーバーを作成してロールに割り当てるには、以下の手順を実施します。

手順

1. **stackrc** ファイルを取得します。

```
[stack@director ~]$ source ~/stackrc
```

2. ノードリストを取得して UUID を把握します。

```
(undercloud)$ openstack baremetal node list
```

3. リーフネットワークとロールを識別するカスタムリソースクラスを使用して、ロールに指定する各ベアメタルノードを割り当てます。

```
openstack baremetal node set \
 --resource-class baremetal.<ROLE> <node>
```

   - <ROLE> をロールを識別する名前に置き換えます。
   - <node> をベアメタルノードの ID に置き換えます。
     たとえば、以下のコマンドを実行して、UUID 58c3d07e-24f2-48a7-bbb6-6843f0e8ee13 のノードを Leaf2 上の Compute ロールにタグ付けします。

```
(undercloud)$ openstack baremetal node set \
 --resource-class baremetal.COMPUTE-LEAF2 58c3d07e-24f2-48a7-bbb6-
6843f0e8ee13
```

4. 各ロールがまだ定義されていない場合は、**overcloud-baremetal-deploy.yaml** に追加します。

5. ロールのノードに割り当てるリソースクラスを定義します。

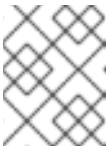
```
- name: <role>
  count: 1
  defaults:
    resource_class: baremetal.<ROLE>
```

- <role> をロールの名前に置き換えます。
- <ROLE> をロールを識別する名前に置き換えます。

6. Baremetal-deploy.yaml ファイルで、ロールのノードに割り当てるリソースクラスを定義します。展開するロール、プロファイル、数量、および関連付けられているネットワークを指定します。

```
- name: <role>
  count: 1
  hostname_format: <role>-%index%
  ansible_playbooks:
    - playbook: bm-deploy-playbook.yaml
  defaults:
    resource_class: baremetal.<ROLE>
    profile: control
    networks:
      - network: external
        subnet: external_subnet
      - network: internal_api
        subnet: internal_api_subnet01
      - network: storage
        subnet: storage_subnet01
      - network: storage_mgmt
        subnet: storage_mgmt_subnet01
      - network: tenant
        subnet: tenant_subnet01
    network_config:
      template: templates/multiple_nics/multiple_nics_dvr.j2
      default_route_network:
        - external
```

- <role> をロールの名前に置き換えます。
- <ROLE> をロールを識別する名前に置き換えます。



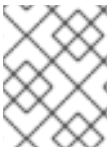
注記

/home/stack/<stack> に、デプロイするすべてのスタックに対して、**baremetal-deploy.yaml** 環境ファイルを作成する必要があります。

3.4. ベアメタルノードのポートからコントロールプレーンのネットワークセグメントへのマッピング

L3 ルーティング対応のネットワーク上でのデプロイメントを有効にするには、ベアメタルポートの **physical_network** フィールドを設定する必要があります。各ベアメタルポートは、OpenStack Bare Metal (ironic) サービス内のベアメタルノードに関連付けられます。物理ネットワーク名は、アンダーク

ラウドの設定の **subnets** オプションで指定する名前です。



注記

undercloud.conf ファイルの **local_subnet** で指定されるサブネットの物理ネットワーク名には、必ず **ctlplane** という名前が付けられます。

手順

1. **stackrc** ファイルを取得します。

```
$ source ~/stackrc
```

2. ベアメタルノードをチェックします。

```
$ openstack baremetal node list
```

3. ベアメタルノードは **enroll** または **manageable** の状態であることを確認してください。ベアメタルノードがこれらのいずれかの状態にない場合、ベアメタルポートで **physical_network** プロパティーを設定するコマンドは失敗します。全ノードを **manageable** の状態に設定するには、以下のコマンドを実行します。

```
$ for node in $(openstack baremetal node list -f value -c Name); do openstack baremetal
node manage $node --wait; done
```

4. ベアメタルポートとベアメタルノードの関連付けを確認します。

```
$ openstack baremetal port list --node <node-uuid>
```

5. ポートの **physical-network** パラメーターを設定します。以下の例では、**leaf0**、**leaf1**、および **leaf2** の 3 つのサブネットが設定で定義されています。local_subnet は **leaf0** です。**local_subnet** の物理ネットワークは常に **ctlplane** であるため、**leaf0** に接続されたベアメタルポートは必ず ctlplane を使用します。残りのポートは他のリーフ名を使用します。

```
$ openstack baremetal port set --physical-network ctlplane <port-uuid>
$ openstack baremetal port set --physical-network leaf1 <port-uuid>
$ openstack baremetal port set --physical-network leaf2 <port-uuid>
```

6. オーバークラウドをデプロイする前に、ノードをイントロスペクションします。**--all-manageable** オプションと **-provide** オプションを付けて、デプロイ可能なノードとして設定します。

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

3.5. スパイン/リーフ型のプロビジョニングネットワークへの新規リーフの追加

新しい物理サイトの追加など、ネットワーク容量を増やす場合には、新しいリーフと、対応するサブネットを Red Hat OpenStack Platform のスパイン/リーフ型のプロビジョニングネットワークに追加する必要がある場合があります。オーバークラウドでリーフをプロビジョニングする場合には、対応するアンダークラウドのリーフが使用されます。

前提条件

- RHOSP デプロイメントでスパイン/リーフ型ネットワークトポロジーが使用されている。

手順

1. アンダークラウドホストに stack ユーザーとしてログインします。
2. **stackrc** アンダークラウド認証情報ファイルを入手します。

```
$ source ~/stackrc
```

3. **/home/stack/undercloud.conf** ファイルで、以下の手順を実施します。
   a. **subnets** パラメーターを特定し、追加するリーフ用の新規サブネットを追加します。サブネットは、ルーティング対応のスパイン/リーフ内の L2 セグメントを表します。

      例

      以下の例では、新しいリーフ (**leaf3**) に新規サブネット (**leaf3**) が追加されます。

```
subnets = leaf0,leaf1,leaf2,leaf3
```

   b. 追加したサブネットのセクションを作成します。

      例

      以下の例では、新しいサブネット (**leaf3** に **[leaf3]** セクションが追加されます。

```
[leaf0]
cidr = 192.168.10.0/24
dhcp_start = 192.168.10.10
dhcp_end = 192.168.10.90
inspection_iprange = 192.168.10.100,192.168.10.190
gateway = 192.168.10.1
masquerade = False

[leaf1]
cidr = 192.168.11.0/24
dhcp_start = 192.168.11.10
dhcp_end = 192.168.11.90
inspection_iprange = 192.168.11.100,192.168.11.190
gateway = 192.168.11.1
masquerade = False

[leaf2]
cidr = 192.168.12.0/24
dhcp_start = 192.168.12.10
dhcp_end = 192.168.12.90
inspection_iprange = 192.168.12.100,192.168.12.190
gateway = 192.168.12.1
masquerade = False

[leaf3]
cidr = 192.168.13.0/24
dhcp_start = 192.168.13.10
```

```
dhcp_end = 192.168.13.90
inspection_iprange = 192.168.13.100,192.168.13.190
gateway = 192.168.13.1
masquerade = False
```

4. **undercloud.conf** ファイルを保存します。
5. アンダークラウドを再インストールします。

```
$ openstack undercloud install
```

関連情報

- スパイン/リーフ型デプロイメントへの新たなリーフの追加

第4章 DCN デプロイメント用オーバークラウドテンプレートの準備

4.1. 個別の HEAT スタックを使用するための前提条件

個別の heat スタックを使用してデプロイメントを作成するためには、お使いの環境が以下の前提条件を満たす必要があります。

- Red Hat OpenStack Platform director 17.1 のインストール済みインスタンス。
- Ceph Storage ユーザー: Red Hat Ceph Storage 5 へのアクセス
- 中央サイト: 中央コントローラーノードとしての機能を持つ 3 台のノード。3 台のコントローラーノードは、すべて同じ heat スタック内になければなりません。コントローラーノードまたはいずれかのコントロールプレーンサービスを異なる heat スタックに分割することはできません。
- エッジサイトに Ceph ストレージをデプロイする場合、中央サイトでは Ceph ストレージが要件となります。
- それぞれの追加 DCN サイト: 3 台の HCI Compute ノード
- すべてノードは事前にプロビジョニングされているか、中央のデプロイメントネットワークから PXE ブートできる必要があります。DHCP リレーを使用して、DCN 向けのこの接続を有効にすることができます。
- すべてのノードが ironic によってイントロスペクションされている。
- Red Hat では、<role>HostnameFormat パラメーターをデフォルト値 %stackname%-<role>-%index% のままにすることを推奨します。%stackname% の接頭辞を含めないと、オーバークラウドは別のスタックの分散コンピュートノードに同じホスト名を使用します。分散コンピュートノードが %stackname% の接頭辞を使用して、別のエッジサイトのノードと区別できるようにします。たとえば、**dcn0** と **dcn1** という名前の 2 つのエッジサイトをデプロイする場合、スタック名の接頭辞は、アンダークラウドで **openstack server list** コマンドを実行する際に dcn0-distributedcompute-0 と dcn1-distributedcompute-0 を区別するのに役立ちます。
- source コマンドで **centralrc** 認証ファイルを読み込み、エッジサイトおよび中央サイトでワークロードをスケジュールします。エッジサイト用に自動的に生成される認証ファイルは必要ありません。

4.2. 個別 HEAT スタックのデプロイメント例の制限

本セクションでは、Red Hat OpenStack Platform 上で個別の heat スタックを使用するデプロイメントの例について説明します。この環境の例には、以下の制限があります。

- スパイン/リーフ型ネットワーク: 本セクションの例は、分散コンピュートノード (DCN) デプロイメントで必要となるルーティング要件を示していません。
- Ironic DHCP リレー: 本セクションには、DHCP リレーと共に Ironic を設定する方法は含まれません。

4.3. 個別 HEAT スタックのデプロイメントの設計

個別の heat スタック内でデプロイメントを分割するには、まずコントロールプレーンと共に単一の

オーバークラウドをデプロイする必要があります。その後、分散コンピュートノード (DCN) サイト向けに個別のスタックを作成することができます。以下の例は、異なるノード種別の個別スタックを示しています。

- コントローラーノード: **central** (例) という名前の個別 heat スタックにより、コントローラーをデプロイします。DCN サイト向けの新規 heat スタックを作成する場合は、**central** スタックからのデータを使用してスタックを作成する必要があります。コントローラーノードは、あらゆるインスタンス管理タスクに利用できなければなりません。
- DCN サイト: **dcn0**、**dcn1** など一意の名前が付けられた個別の heat スタックを設定することができます。DHCP リレーを使用して、プロビジョニングネットワークをリモートサイトに拡張します。

注記

それぞれのスタック用に個別のアベイラビリティーゾーン (AZ) を作成する必要があります。

4.4. 個別の HEAT スタックの管理

本セクションの手順では、3 つの heat スタック (**central**、**dcn0**、および **dcn1**) 用の環境ファイルを取りまとめる方法について説明します。Red Hat では、各デプロイメントに関する情報を個別に維持するために、各 heat スタックのテンプレートを個別のディレクトリーに保管することを推奨します。

手順

1. **central** heat スタックを定義します。

```
$ mkdir central
$ touch central/overrides.yaml
```

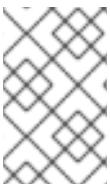
2. **central** heat スタックから、データを全 DCN サイト用の共通ディレクトリーに抽出します。

```
$ mkdir dcn-common
$ touch dcn-common/overrides.yaml
```

3. **dcn0** サイトを定義します。

```
$ mkdir dcn0
$ touch dcn0/overrides.yaml
```

さらに DCN サイトをデプロイするには、数字を増やして追加の **dcn** ディレクトリーを作成します。



注記

ファイル設定の例を示すために、touch コマンドを使用しています。デプロイメントに成功するためには、それぞれのファイルに適切なコンテンツが含まれている必要があります。

4.5. コンテナーイメージの取得

個別の heat スタックによるデプロイメントに必要なコンテナーイメージを取得するには、以下の手順およびサンプルファイルのコンテンツを使用します。エッジサイトの環境ファイルを指定して **openstack**

container image prepare コマンドを実行し、オプションまたはエッジサイト固有のサービスのコンテナーイメージが含まれるようにする必要があります。

詳細は、**director を使用した Red Hat OpenStack Platform のインストールと管理**ガイドの コンテナーイメージの準備 を参照してください。

手順

1. **containers.yaml** にレジストリーサービスアカウントの認証情報を追加します。

```
parameter_defaults:
  ContainerImagePrepare:
  - push_destination: true
    set:
      ceph_namespace: registry.redhat.io/rhceph
      ceph_image: rhceph-6-rhel9
      ceph_tag: latest
      name_prefix: openstack-
      namespace: registry.redhat.io/rhosp17-rhel9
      tag: latest
  ContainerImageRegistryCredentials:
    # https://access.redhat.com/RegistryAuthentication
    registry.redhat.io:
      registry-service-account-username: registry-service-account-password
```

2. **images-env.yaml** として環境ファイルを生成します。

```
sudo openstack tripleo container image prepare \
-e containers.yaml \
--output-env-file images-env.yaml
```

作成される **images-env.yaml** ファイルは、ファイルを生成したスタックのデプロイメント手順の一部として追加されます。

4.6. エッジサイト用の高速データパスロールの作成

エッジサイトで高速データパスサービスを使用するには、高速データパスとエッジサービスの両方を定義するカスタムロールを作成する必要があります。デプロイメントのロールファイルを作成する場合には、分散コンピュートノードアーキテクチャーおよび DPDK や SR-IOV 等の高速データパスサービスの両方に必要なサービスを定義する新たに作成されるロールを含めることができます。

以下の例では、DPDK と distributedCompute のカスタムロールを作成します。

前提条件

アンダークラウドの正常なインストール。詳細は、アンダークラウドのインストール を参照してください。

手順

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. デフォルトの **roles** ディレクトリーをコピーします。

```
cp -r /usr/share/openstack-tripleo-heat-templates/roles ~/.
```

3. **DistributedCompute.yaml** ファイルから **DistributedComputeDpdk.yaml** という新しいファイルを作成します。

```
cp roles/DistributedCompute.yaml roles/DistributedComputeDpdk.yaml
```

4. DPDK サービスを新しい **DistributedComputeDpdk.yaml** ファイルに追加します。**DistributedComputeDpdk.yaml** ファイルにないパラメーターを **ComputeOvsDpdk.yaml** ファイルで特定することで、追加が必要なパラメーターを特定できます。

```
diff -u roles/DistributedComputeDpdk.yaml roles/ComputeOvsDpdk.yaml
```

この出力では、**+** で始まるパラメーターは ComputeOvsDpdk.yaml ファイルに存在しますが、DistributedComputeDpdk.yaml ファイルにはありません。これらのパラメーターを新しい **DistributedComputeDpdk.yaml** ファイルに追加します。

5. **DistributedComputeDpdk.yaml** を使用して、**DistributedComputeDpdk** ロールファイルを作成します。

```
openstack overcloud roles generate --roles-path ~/roles/ -o ~/roles/roles-custom.yaml
DistributedComputeDpdk
```

これと同じ手法を使用して、要件を満たすように、エッジの SR-IOV または SR-IOV と DPDK の組み合わせ用に、高速データパスロールを作成することができます。

ブロックストレージを持たないエッジサイトをデプロイする場合は、以下を参照してください。

- 5章*中央サイトのインストール*
- 「ストレージを持たないエッジノードのデプロイ」

Red Hat Ceph Storage と共にエッジサイトをデプロイする場合は、以下を参照してください。

- 5章*中央サイトのインストール*
- 「ハイパーコンバージドストレージを使用したエッジサイトのデプロイメント」

第5章 中央サイトのインストール

分散コンピュートノード (DCN) アーキテクチャーを使用して Red Hat OpenStack プラットフォームをデプロイする場合は、事前にストレージ戦略を決定する必要があります。中央サイトに Red Hat Ceph Storage を設定せずに Red Hat OpenStack Platform をデプロイする場合は、どのエッジサイトにも Red Hat Ceph Storage をデプロイすることはできません。また、後で再デプロイして Red Hat Ceph Storage を中央サイトに追加するオプションはありません。

分散コンピュートノード (DCN) アーキテクチャーに中央サイトをデプロイする場合、クラスターをデプロイすることができます。

- Compute ノードあり/なし
- Red Hat Ceph Storage あり/なし

5.1. エッジストレージを持たない中央コントローラーのデプロイ

中央サイトで Image サービス (glance) のバックエンドとして Object Storage サービス (swift) を使用する場合は、エッジサイトにブロックストレージを持たない分散コンピュートノードクラスターをデプロイすることができます。各アーキテクチャーのロールおよびネットワークプロファイルが異なるため、ブロックストレージを設定せずにデプロイされたサイトは、後でブロックストレージを持つように更新することはできません。

重要: 以下の手順では Cinder のバックエンドとして lvm を使用していますが、実稼働環境用ではサポートされません。Cinder のバックエンドとして、認定されたブロックストレージソリューションをデプロイする必要があります。

一般的なオーバークラウドデプロイメントと同様に、中央コントローラークラスターをデプロイします。このクラスターには Compute ノードは必要ありません。したがって、Compute ノード数を **0** に設定し、デフォルトの **1** をオーバーライドすることができます。中央コントローラーには、ストレージおよび Oslo 設定に関して特定の要件があります。これらの要件を満たすには、以下の手順を使用します。

前提条件

- 環境に固有の **network_data.yaml** および **vip_data.yaml** ファイルを作成する必要がある。サンプルファイルは **/usr/share/openstack-tripleo-heat-templates/network-data-samples** にあります。
- 環境に固有の **overcloud-baremetal-deploy.yaml** ファイルを作成している。詳細は、オーバークラウド用のベアメタルノードのプロビジョニング を参照してください。

手順

以下の手順で、中央サイトの初回デプロイメント手順の概要を説明します。

注記

glance マルチストアを持たない DCN デプロイメントの例について、デプロイメントコマンドおよび環境ファイルを以下の手順で詳しく説明します。以下の手順には、ここでの目的とは関連しないが実際の設定には必要な要素 (ネットワーク設定など) は含まれていません。

1. アンダークラウドに stack ユーザーとしてログインします。

2. source コマンドで stackrc ファイルを読み込みます。

```
[stack@director ~]$ source /home/stack/stackrc
```

3. 環境ファイルを生成します。

```
sudo openstack tripleo container image prepare \
-e containers.yaml \
--output-env-file /home/stack/central/central-images-env.yaml
```

4. ホームディレクトリーに、デプロイする各スタックのディレクトリーを作成します。中央サイトの **network_data.yaml**、**vip_data.yaml**、および **overcloud-baremetal-deploy.yaml** テンプレートを **/home/stack/central/** に移動します。

```
mkdir /home/stack/central
mkdir /home/stack/dcn0
mkdir /home/stack/dcn1

mv network_data.yaml /home/stack/central
mv vip_data.yaml /home/stack/central
mv overcloud-baremetal-deploy.yaml /home/stack/central
```

5. オーバークラウドのネットワークをプロビジョニングします。このコマンドは、オーバークラウドネットワークの定義ファイルを入力として受け取ります。オーバークラウドをデプロイするには、コマンドで出力ファイルを使用する必要があります。

```
(undercloud)$ openstack overcloud network provision \
--output /home/stack/central/overcloud-networks-deployed.yaml \
/home/stack/central/network_data.yaml
```

6. オーバークラウドの仮想 IP をプロビジョニングします。このコマンドは、仮想 IP の定義ファイルを入力として受け取ります。オーバークラウドをデプロイするには、コマンドで出力ファイルを使用する必要があります。

```
(undercloud)$ openstack overcloud network vip provision \
--stack central \
--output /home/stack/central/overcloud-vip-deployed.yaml \
/home/stack/central/vip_data.yaml
```

7. ベアメタルインスタンスをプロビジョニングします。このコマンドは、ベアメタルノードの定義ファイルを入力として受け取ります。オーバークラウドをデプロイするには、コマンドで出力ファイルを使用する必要があります。

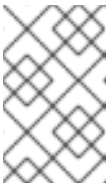
```
(undercloud)$ openstack overcloud node provision \
--stack central \
--network-config \
-o /home/stack/central/deployed_metal.yaml \
/home/stack/central/overcloud-baremetal-deploy.yaml
```

8. 以下のような設定で **central/overrides.yaml** という名前のファイルを作成します。

```
parameter_defaults:
  NtpServer:
```

```
    - 0.pool.ntp.org
    - 1.pool.ntp.org
  GlanceBackend: swift
```

- **ControllerCount: 3**: ノードを 3 台デプロイすることを指定します。これらのノードは、glance 用に swift を、cinder 用に lvm をそれぞれ使用し、エッジコンピュートノード用に control-plane サービスをホストします。
- **ComputeCount: 0**: オプションのパラメーターで、Compute ノードが中央コントローラーノードと共にデプロイされないようにします。
- **GlanceBackend: swift**: Image サービス (glance) のバックエンドとして Object Storage (swift) を使用することを指定します。
  この設定は、分散コンピュートノード (DCN) と以下のように連携します。
- DCN 上の Image サービスは、中央の Object Storage バックエンドから受けとるイメージのキャッシュコピーを作成します。Image サービスは、HTTP を使用して Object Storage からのイメージをローカルディスクキャッシュにコピーします。



注記

中央のコントローラーノードは、分散コンピュートノード (DCN) サイトに接続できる必要があります。中央コントローラーノードは、ルーティング対応のレイヤー 3 接続を使用することができます。

9. **site-name.yaml** 環境ファイルでサイトの命名規則を設定します。Nova アベイラビリティーゾーンと Cinder ストレージアベイラビリティーゾーンが一致している必要があります。

```
cat > /home/stack/central/site-name.yaml << EOF
parameter_defaults:
   NovaComputeAvailabilityZone: central
   ControllerExtraConfig:
      nova::availability_zone::default_schedule_zone: central
   NovaCrossAZAttach: false
EOF
```

10. 中央コントローラーノードをデプロイします。たとえば、以下の内容の **deploy.sh** ファイルを使用することができます。

```
openstack overcloud deploy \
--deployed-server \
--stack central \
--templates /usr/share/openstack-tripleo-heat-templates/ \
-n /home/stack/central/network_data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
-e /home/stack/central/overcloud-networks-deployed.yaml \
-e /home/stack/central/overcloud-vip-deployed.yaml \
-e /home/stack/central/deployed_metal.yaml
```

注記

openstack overcloud deploy コマンドに、ネットワーク設定用の heat テンプレートを追加する必要があります。エッジアーキテクチャーの設計には、スパイン/リーフ型ネットワークが必要です。詳細は、スパイン/リーフネットワークの設定 を参照してください。

5.2. ストレージが設定された中央サイトのデプロイ

マルチストアの Image サービスおよびバックエンドとしての Ceph Storage をデプロイするには、以下の手順を実施します。

前提条件

- 環境に固有の **network_data.yaml** および **vip_data.yaml** ファイルを作成する必要がある。サンプルファイルは **/usr/share/openstack-tripleo-heat-templates/network-data-samples** にあります。
- 環境に固有の **overcloud-baremetal-deploy.yaml** ファイルを作成している。詳細は、オーバークラウド用のベアメタルノードのプロビジョニング を参照してください。
- 中央サイトおよび各アベイラビリティーゾーンまたはストレージサービスが必要な各地区での Ceph クラスター用ハードウェア
- 中央サイトおよび各アベイラビリティーゾーンまたはストレージサービスが必要な各地区での 3 つの Image サービス (glance) サーバー用ハードウェア。エッジロケーションでは、Image サービスが DistributedComputeHCI ノードにデプロイされる。

手順

Image サービス (glance) を複数のストアで使用できるように、Red Hat OpenStack Platform の中央ロケーションをデプロイします。

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

```
[stack@director ~]$ source /home/stack/stackrc
```

3. 環境ファイル /home/stack/central/central-images-env.yaml を生成します

```
sudo openstack tripleo container image prepare \
-e containers.yaml \
--output-env-file /home/stack/central/central-images-env.yaml
```

4. 実際の環境に適したロールを使用して、中央サイト用のロールを生成します。

```
openstack overcloud roles generate Compute Controller CephStorage \
-o /home/stack/central/central_roles.yaml
```

5. ホームディレクトリーに、デプロイする各スタックのディレクトリーを作成します。中央サイトの **network_data.yaml**、**vip_data.yaml**、および **overcloud-baremetal-deploy.yaml** テンプレートを **/home/stack/central/** に移動します。

```
mkdir /home/stack/central
```

```
mkdir /home/stack/dcn0
mkdir /home/stack/dcn1

mv network_data.yaml /home/stack/central
mv vip_data.yaml /home/stack/central
mv overcloud-baremetal-deploy.yaml /home/stack/central
```

6. オーバークラウドのネットワークをプロビジョニングします。このコマンドは、オーバークラウドネットワークの定義ファイルを入力として受け取ります。オーバークラウドをデプロイするには、コマンドで出力ファイルを使用する必要があります。

```
openstack overcloud network provision \
--output /home/stack/central/overcloud-networks-deployed.yaml \
/home/stack/central/network_data.yaml
```

7. オーバークラウドの仮想 IP をプロビジョニングします。このコマンドは、仮想 IP の定義ファイルを入力として受け取ります。オーバークラウドをデプロイするには、コマンドで出力ファイルを使用する必要があります。

```
openstack overcloud network vip provision \
--stack central \
--output /home/stack/central/overcloud-vip-deployed.yaml \
/home/stack/central/vip_data.yaml
```

8. ベアメタルインスタンスをプロビジョニングします。このコマンドは、ベアメタルノードの定義ファイルを入力として受け取ります。オーバークラウドをデプロイするには、コマンドで出力ファイルを使用する必要があります。

```
openstack overcloud node provision \
--stack central \
--network-config \
-o /home/stack/central/deployed_metal.yaml \
/home/stack/central/overcloud-baremetal-deploy.yaml
```

9. ハイパーコンバージドストレージを使用して中央のロケーションをデプロイする場合は、次のパラメーターを使用して **initial-ceph.conf** 設定ファイルを作成する必要があります。詳細は、HCI 用の Red Hat Ceph Storage クラスターの設定 を参照してください。

```
[osd]
osd_memory_target_autotune = true
osd_numa_auto_affinity = true
[mgr]
mgr/cephadm/autotune_memory_target_ratio = 0.2
```

10. **deployed_metal.yaml** ファイルを **openstack overcloud ceph deploy** コマンドへの入力として使用します。**openstack overcloud ceph deploy command** は、デプロイされた Ceph クラスターを記述する yaml ファイルを出力します。

```
openstack overcloud ceph deploy \
--stack central \
/home/stack/central/deployed_metal.yaml \
--config /home/stack/central/initial-ceph.conf \ 1
--output /home/stack/central/deployed_ceph.yaml \
--container-image-prepare /home/stack/containers.yaml \
```

```
--network-data /home/stack/network-data.yaml \
--cluster central \
--roles-data /home/stack/central/central_roles.yaml
```

1 ハイパーコンバージドインフラストラクチャーをデプロイする場合にのみ、initial-ceph.com を含めます。

11. 続行する前に、Ceph デプロイメントが機能していることを確認してください。**ssh** を使用して、**ceph-mon** サービスを実行しているサーバーに接続します。HCI デプロイメントでは、これはコントローラーノードです。以下のコマンドを実行します。

```
cephadm shell --config /etc/ceph/central.conf \
--keyring /etc/ceph/central.client.admin.keyring
```

注記

--config および **--keyring** パラメーターを使用する必要があります。

12. **site-name.yaml** 環境ファイルでサイトの命名規則を設定します。Nova アベイラビリティーゾーンと Cinder ストレージアベイラビリティーゾーンが一致している必要があります。

```
parameter_defaults:
    NovaComputeAvailabilityZone: central
    ControllerExtraConfig:
        nova::availability_zone::default_schedule_zone: central
    NovaCrossAZAttach: false
    CinderStorageAvailabilityZone: central
    GlanceBackendID: central
```

13. 以下のような内容で glance.yaml テンプレートを設定します。

```
parameter_defaults:
    GlanceEnabledImportMethods: web-download,copy-image
    GlanceBackend: rbd
    GlanceStoreDescription: 'central rbd glance store'
    GlanceBackendID: central
    CephClusterName: central
```

14. 中央のロケーションにスタックをデプロイします。

```
openstack overcloud deploy \
--deployed-server \
--stack central \
--templates /usr/share/openstack-tripleo-heat-templates/ \
-r /home/stack/central/central_roles.yaml \
-n ~/network-data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/cephadm.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
-e /home/stack/central/overcloud-networks-deployed.yaml \
-e /home/stack/central/overcloud-vip-deployed.yaml \
```

```
-e /home/stack/central/deployed_metal.yaml \
-e /home/stack/central/deployed_ceph.yaml \
-e ~/central/glance.yaml
```

15. 中央のロケーションにオーバークラウドをデプロイすると、エッジサイトの追加のスタックデプロイメントの入力として必要なデータがエクスポートされ、**/home/stack/overcloud-deploy** ディレクトリーに配置されます。**central-export.yaml** ファイルが存在することを確認します。

```
stat /home/stack/overcloud-deploy/central/central-export.yaml
```

16. Ceph 固有のデータをエクスポートします。

```
openstack overcloud export ceph \
--stack central \
--output-file /home/stack/dcn-common/central_ceph_external.yaml
```

5.3. 外部 CEPH の統合

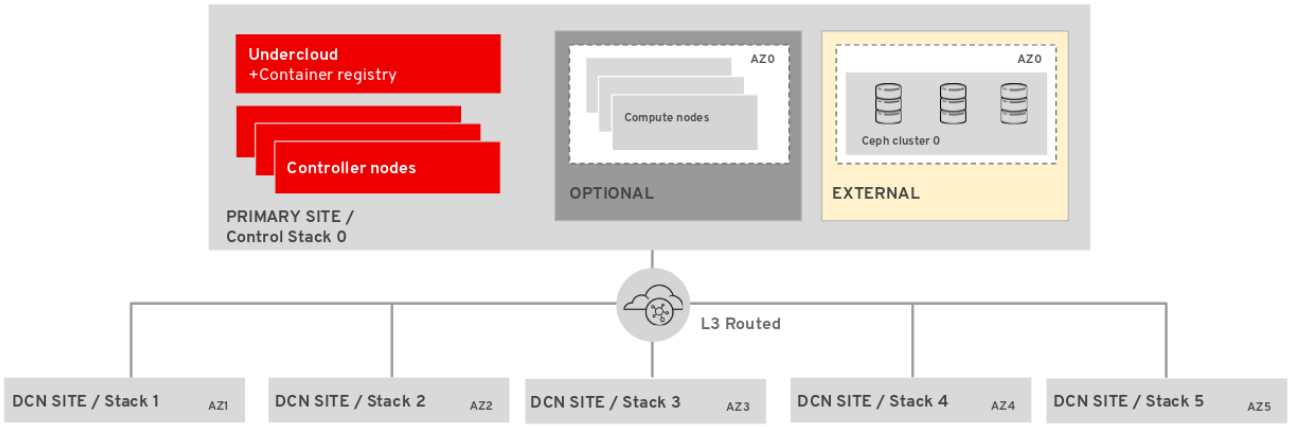
分散コンピュートノード (DCN) アーキテクチャーの中央サイトをデプロイし、事前にデプロイした Red Hat Ceph Storage ソリューションを統合することができます。director を使用せずに Red Hat Ceph Storage をデプロイすると、director は環境内の Red Hat Ceph Storage に関する情報を持ちません。**openstack overcloud export ceph コマンド** を実行することはできず、**central_ceph_external.yaml** を手動で作成する必要があります。

前提条件

- 環境に固有の **network_data.yaml** および **vip_data.yaml** ファイルを作成する必要がある。サンプルファイルは **/usr/share/openstack-tripleo-heat-templates/network-data-samples** にあります。
- 環境に固有の **overcloud-baremetal-deploy.yaml** ファイルを作成している。詳細は、オーバークラウド用のベアメタルノードのプロビジョニング を参照してください。
- 中央サイトおよび各アベイラビリティーゾーンまたはストレージサービスが必要な各地区での Ceph クラスター用ハードウェア

2 つまたはそれ以上のスタックで設定されるデプロイメントの例を以下に示します。

- 中央サイトに 1 つのスタック (**central**)
- エッジサイトに 1 つのスタック (**dcn0**)
- **dcn0** と同様にデプロイされた追加のスタック (**dcn1**、**dcn2**、等)



手順

既存の Red Hat Ceph Storage クラスターとの統合 に記載されているプロセスに従って、既存の Red Hat Ceph Storage ソリューションと統合されるように中央サイトをインストールできます。Red Hat Ceph Storage を DCN デプロイメントの中央サイトと統合するための特別な要件はありませんが、オーバークラウドをデプロイする前に DCN 固有の手順を完了する必要があります。

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

```
[stack@director ~]$ source ~/stackrc
```

3. 環境ファイル ~/central/central-images-env.yaml を生成します。

```
sudo openstack tripleo container image prepare \
-e containers.yaml \
--output-env-file ~/central/central-images-env.yaml
```

4. ホームディレクトリーに、デプロイする各スタックのディレクトリーを作成します。これを使用して、それぞれのサイト用に設計されたテンプレートを分離します。中央サイトの **network_data.yaml**、**vip_data.yaml**、および **overcloud-baremetal-deploy.yaml** テンプレートを **/home/stack/central/** に移動します。

```
mkdir /home/stack/central
mkdir /home/stack/dcn0
mkdir /home/stack/dcn1

mv network_data.yaml /home/stack/central
mv vip_data.yaml /home/stack/central
mv overcloud-baremetal-deploy.yaml /home/stack/central
```

5. オーバークラウドのネットワークをプロビジョニングします。このコマンドは、オーバークラウドネットワークの定義ファイルを入力として受け取ります。オーバークラウドをデプロイするには、コマンドで出力ファイルを使用する必要があります。

```
openstack overcloud network provision \
--output /home/stack/central/overcloud-networks-deployed.yaml \
/home/stack/central/network_data.yaml
```

6. オーバークラウドの仮想 IP をプロビジョニングします。このコマンドは、仮想 IP の定義ファイルを入力として受け取ります。オーバークラウドをデプロイするには、コマンドで出力ファイルを使用する必要があります。

```
openstack overcloud network vip provision \
--stack central \
--output /home/stack/central/overcloud-vip-deployed.yaml \
/home/stack/central/vip_data.yaml
```

7. ベアメタルインスタンスをプロビジョニングします。このコマンドは、ベアメタルノードの定義ファイルを入力として受け取ります。オーバークラウドをデプロイするには、コマンドで出力ファイルを使用する必要があります。

```
openstack overcloud node provision \
--stack central \
--network-config \
-o /home/stack/central/deployed_metal.yaml \
/home/stack/central/overcloud-baremetal-deploy.yaml
```

8. **site-name.yaml** 環境ファイルでサイトの命名規則を設定します。Compute (nova) アベイラビリティーゾーンと Block Storage (cinder) アベイラビリティーゾーンが一致している必要があります。

```
cat > /home/stack/central/site-name.yaml << EOF
parameter_defaults:
    NovaComputeAvailabilityZone: central
    ControllerExtraConfig:
        nova::availability_zone::default_schedule_zone: central
    NovaCrossAZAttach: false
    CinderStorageAvailabilityZone: central
    GlanceBackendID: central
EOF
```

9. 次のような内容で **external-ceph.yaml** テンプレートを設定します。

```
parameter_defaults:
    CinderEnableIscsiBackend: false
    CinderEnableRbdBackend: true
    CinderEnableNfsBackend: false
    NovaEnableRbdBackend: true
    GlanceBackend: rbd
    GlanceBackendID: central
    GlanceEnabledImportMethods: web-download,copy-image
    GlanceStoreDescription: 'central rbd glance store'
    CinderRbdPoolName: "openstack-cinder"
    NovaRbdPoolName: "openstack-nova"
    GlanceRbdPoolName: "openstack-images"
    CinderBackupRbdPoolName: "automation-backups"
    GnocchiRbdPoolName: "automation-metrics"
    CephClusterFSID: 38dd387e-837a-437c-891c-7fc69e17a3c
    CephClusterName: central
    CephExternalMonHost: 10.9.0.1,10.9.0.2,10.9.0.3
    CephClientKey: "AQAKtECeLemfiBBdQp7cjNYQRGW9y8GnhhFZg=="
    CephClientUserName: "openstack
```

10. 中央のロケーションをデプロイします。

```
openstack overcloud deploy \
--stack central \
--templates /usr/share/openstack-tripleo-heat-templates/ \
-n /home/stack/central/network-data.yaml \
...
-e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/external-ceph.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
-e /home/stack/central/overcloud-networks-deployed.yaml \
-e /home/stack/central/overcloud-vip-deployed.yaml \
-e /home/stack/central/deployed_metal.yaml \
-e /home/stack/central/external-ceph.yaml \
-e /home/stack/central/overcloud-networks-deployed.yaml \
-e /home/stack/central/central_roles.yaml
```

11. 中央のロケーションにオーバークラウドをデプロイすると、エッジサイトの追加のスタックデプロイメントの入力として必要なデータがエクスポートされ、**/home/stack/overcloud-deploy** ディレクトリーに配置されます。次の control-plane-export.yaml ファイルが存在することを確認します。

```
stat ~/overcloud-deploy/control-plane/control-plane-export.yaml
```

12. Red Hat Ceph Storage デプロイメントに関する詳細を **含む central_ceph_external.yaml** という環境ファイルを作成します。このファイルは、エッジサイトの追加のスタックデプロイメントに渡すことができます。

```
parameter_defaults:
  CephExternalMultiConfig:
    - cluster: "central"
      fsid: "3161a3b4-e5ff-42a0-9f53-860403b29a33"
      external_cluster_mon_ips: "172.16.11.84, 172.16.11.87, 172.16.11.92"
      keys:
        - name: "client.openstack"
          caps:
            mgr: "allow *"
            mon: "profile rbd"
            osd: "profile rbd pool=vms, profile rbd pool=volumes, profile rbd pool=images"
          key: "AQD29WteAAAAABAAphgOjFD7nyjdYe8Lz0mQ5Q=="
          mode: "0600"
      dashboard_enabled: false
      ceph_conf_overrides:
        client:
          keyring: /etc/ceph/central.client.openstack.keyring
```

- **fsid** パラメーターは、Ceph Storage クラスターのファイルシステム ID です。この値は、クラスター設定ファイルの **[global]** セクションで指定します。

```
[global]
fsid = 4b5c8c0a-ff60-454b-a1b4-9747aa737d19
...
```

- **key** パラメーターは、openstack アカウントの ceph クライアントキーです。

```
[root@ceph ~]# ceph auth list
...
[client.openstack]
    key = AQC+vYNXgDAgAhAAc8UoYt+OTz5uhV7ItLdwUw==
    caps mgr = "allow *"
    caps mon = "profile rbd"
    caps osd = "profile rbd pool=volumes, profile rbd pool=vms, profile rbd pool=images, profile rbd pool=backups, profile rbd pool=metrics"
...
```

central_ceph_external.yaml ファイルのサンプルにあるパラメーターの詳細は、カスタム環境ファイルの作成 を参照してください。

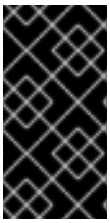
関連情報

- 外部 Ceph Storage クラスター統合の検証

第6章 ストレージを使用しないエッジのデプロイ

中央サイトで Image サービス (glance) のバックエンドとして Object Storage サービス (swift) を使用する場合は、エッジサイトにブロックストレージを持たない分散コンピュートノード (DCN) クラスターをデプロイすることができます。ブロックストレージを使用せずにサイトをデプロイした場合、後で更新してブロックストレージを使用することはできません。

ストレージなしでエッジサイトをデプロイメントする場合は、**compute** ロールを使用します。

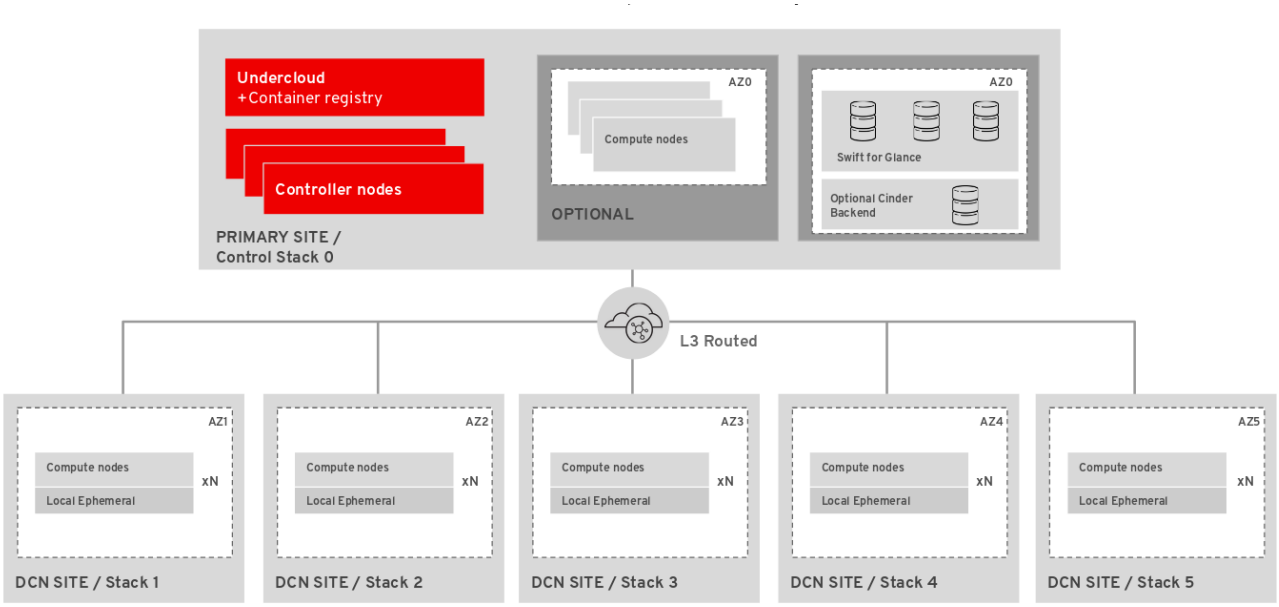


重要

以下の手順では Block Storage サービス (cinder) のバックエンドとして lvm を使用していますが、実稼働環境用ではサポートされません。Block Storage サービスのバックエンドとして、認定されたブロックストレージソリューションをデプロイする必要があります。

6.1. ストレージを使用しない DCN エッジサイトのアーキテクチャー

このアーキテクチャーをデプロイするには、**Compute** ロールを使用します。



エッジにブロックストレージがない場合

- コントロールプレーンの Object Storage (swift) サービスが、Image (glance) サービスのバックエンドとして使用されます。
- マルチバックエンドのイメージサービスは利用できません。
  - イメージはエッジサイトでローカルに Nova でキャッシュされます。詳細は11章*glance イメージの nova への事前キャッシュ*を参照してください。
- インスタンスは、Compute ノード上にローカルに保存されます。
- Block Storage (cinder) などのボリュームサービスは、エッジサイトでは利用できません。

重要

Red Hat Ceph ストレージを持たない中央サイトをデプロイする場合、後でストレージが設定されたエッジサイトをデプロイするオプションはありません。

エッジサイトでのブロックストレージを持たないデプロイに関する詳細は、「ストレージを持たないエッジノードのデプロイ」を参照してください。

6.2. ストレージを持たないエッジノードのデプロイ

Compute ノードをエッジサイトにデプロイする場合、中央のロケーションをコントロールプレーンとして使用します。新しい DCN スタックをデプロイメントに追加し、中央のロケーションから設定ファイルを再利用して新しい環境ファイルを作成できます。

前提条件

- 環境に固有の **network_data.yaml** ファイルを作成する必要がある。サンプルファイルは **/usr/share/openstack-tripleo-heat-templates/network-data-samples** にある。
- 環境に固有の **overcloud-baremetal-deploy.yaml** ファイルを作成している。詳細は、オーバークラウド用のベアメタルノードのプロビジョニング を参照してください。
- イメージをエッジサイトにコピーする前に、中央サイトにイメージをアップロードする必要があります。各イメージのコピーは、中央サイトの Image サービス (glance) に存在している必要があります。
- Image、Compute、Block Storage サービスに、RBD ストレージドライバーを使用する必要があります。

手順

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

```
[stack@director ~]$ source ~/stackrc
```

3. 環境ファイルを生成します ~/dcn0/dcn0-images-env.yaml[d]:

```
sudo[e] openstack tripleo container image prepare \
-e containers.yaml \
--output-env-file ~/dcn0/dcn0-images-env.yaml
```

4. エッジロケーションのロールファイルを生成します。実際の環境に適したロールを使用して、エッジロケーション用のロールを生成します。

```
(undercloud)$ openstack overcloud roles \
 generate Compute \
 -o /home/stack/dcn0/dcn0_roles.yaml
```

5. ネットワークオーバーレイに ML2/OVS を使用している場合は、Compute ロールを編集して、**NeutronDhcpAgent** サービスと **NeutronMetadataAgent** サービスを含める必要があります。

   a. Compute ロールのロールファイルを作成します。

   ```
   openstack overcloud roles \
   generate Compute \
   -o /home/stack/dcn0/dcn0_roles.yaml
   ```

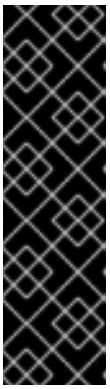
   b. /home/stack/dcn0/dcn0_roles.yaml ファイルを編集して、**NeutronDhcpAgent** および **NeutronMetadataAgent** サービスを含めます。

   ```
   ...
     - OS::TripleO::Services::MySQLClient
     - OS::TripleO::Services::NeutronBgpVpnBagpipe
   +  - OS::TripleO::Services::NeutronDhcpAgent
   +  - OS::TripleO::Services::NeutronMetadataAgent
     - OS::TripleO::Services::NeutronLinuxbridgeAgent
     - OS::TripleO::Services::NeutronVppAgent
     - OS::TripleO::Services::NovaAZConfig
     - OS::TripleO::Services::NovaCompute
   ...
   ```

   詳細は、ルーティング対応プロバイダーネットワークの準備 を参照してください。

6. オーバークラウドのネットワークをプロビジョニングします。このコマンドは、オーバークラウドネットワークの定義ファイルを入力として受け取ります。オーバークラウドをデプロイするには、コマンドで出力ファイルを使用する必要があります。

   ```
   (undercloud)$ openstack overcloud network provision \
   --output /home/stack/dcn0/overcloud-networks-deployed.yaml \
   /home/stack/dcn0/network_data.yaml
   ```



重要

network_data.yaml テンプレートに、中央サイトにネットワークをプロビジョニングしたときに含まれなかった追加のネットワークが含まれている場合は、中央サイトでネットワークプロビジョニングコマンドを再実行する必要があります。

```
(undercloud)$ openstack overcloud network provision \
--output /home/stack/central/overcloud-networks-deployed.yaml \
/home/stack/central/network_data.yaml
```

7. ベアメタルインスタンスをプロビジョニングします。このコマンドは、ベアメタルノードの定義ファイルを入力として受け取ります。オーバークラウドをデプロイするには、コマンドで出力ファイルを使用する必要があります。

   ```
   (undercloud)$ openstack overcloud node provision \
   --stack dcn0 \
   --network-config \
   -o /home/stack/dcn0/deployed_metal.yaml \
   ~/overcloud-baremetal-deploy.yaml
   ```

8. site-name.yaml 環境ファイルでサイトの命名規則を設定します。

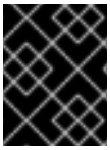
```
parameter_defaults:
  NovaComputeAvailabilityZone: dcn0
  ControllerExtraConfig:
    nova::availability_zone::default_schedule_zone: dcn0
  NovaCrossAZAttach: false
```

9. dcn0 エッジサイトのスタックをデプロイします。

```
openstack overcloud deploy \
--deployed-server \
--stack dcn0 \
--templates /usr/share/openstack-tripleo-heat-templates/ \
-r /home/stack/dcn0/dcn0_roles.yaml \
-n /home/stack/network_data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
-e /home/stack/overcloud-deploy/central/central-export.yaml \
-e /home/stack/dcn0/overcloud-networks-deployed.yaml \
-e /home/stack/dcn0/overcloud-vip-deployed.yaml \
-e /home/stack/dcn0/deployed_metal.yaml
```

6.3. エッジサイトでの特定イメージ種別の除外

デフォルトでは、Compute ノードは、サポートするすべてのイメージ形式を公開します。Compute ノードが Ceph ストレージを使用しない場合には、イメージ形式の公開から RAW イメージを除外することができます。RAW イメージ形式は、QCOW2 イメージよりも多くのネットワーク帯域幅およびローカルストレージを使用し、Ceph ストレージを持たないエッジサイトでの使用は非効率です。特定のイメージ形式を除外するには、**NovaImageTypeExcludeList** パラメーターを使用します。



重要

Ceph には RAW イメージが必要なので、このパラメーターは Ceph を使用するエッジサイトには使用しないでください。

注記

RAW イメージを公開しない Compute ノードは、RAW イメージから作成されたインスタンスをホストできません。これは、スナップショットの再デプロイおよび退避に影響を及ぼす可能性があります。

前提条件

- Red Hat OpenStack Platform director がインストールされている。
- 中央サイトがインストールされている。
- DCN のデプロイメントで Compute ノードが利用可能である。

手順

1. アンダークラウドホストに **stack** ユーザーとしてログインします。

2. source コマンドで **stackrc** 認証情報ファイルを読み込みます。

```
$ source ~/stackrc
```

3. カスタム環境ファイルの 1 つに **NovaImageTypeExcludeList** パラメーターを含めます。

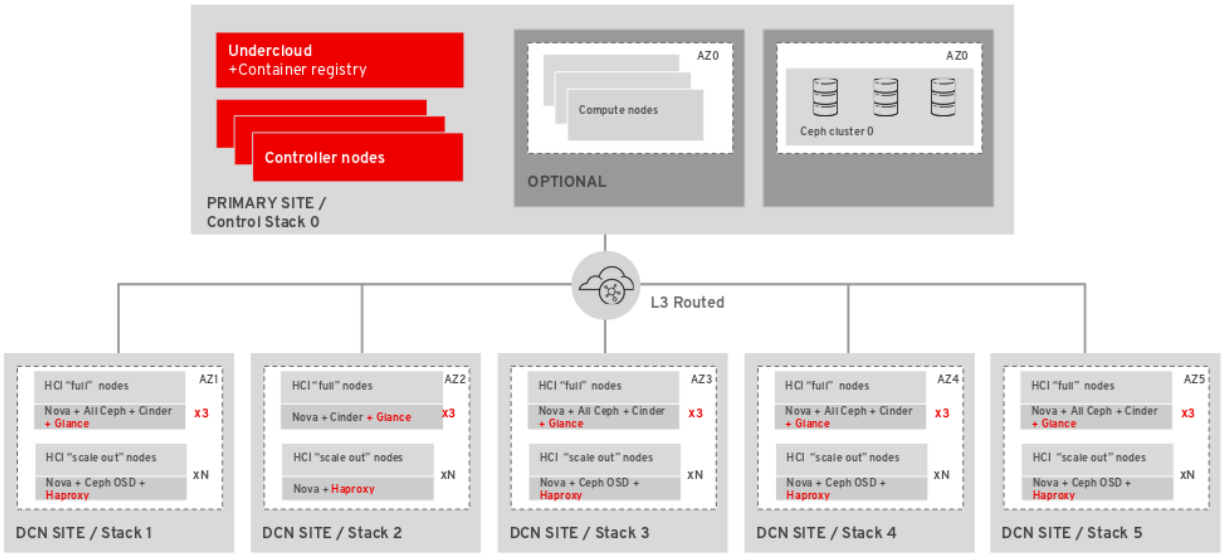
```
parameter_defaults:
  NovaImageTypeExcludeList:
      - raw
```

4. ご自分のデプロイメントに該当するその他の環境ファイルと共に、**NovaImageTypeExcludeList** パラメーターが含まれる環境ファイルをオーバークラウドデプロイメントコマンドに追加します。

```
openstack overcloud deploy --templates \
-n network_data.yaml \
-r roles_data.yaml \
-e <environment_files> \
-e <new_environment_file>
```

第7章 エッジサイトでのストレージのデプロイ

Red Hat OpenStack Platform director を活用して分散コンピュートノードのデプロイメントを拡張し、Red Hat OpenStack Platform と Ceph Storage を使用する利点と共に、エッジサイトに分散イメージの管理および永続ストレージを含めることができます。



7.1. ストレージを使用したエッジデプロイメントのロール

次のロールは、ストレージを備えたエッジデプロイメントで使用できます。選択した設定に基づいて、お使いの環境に適切なロールを選択します。

7.1.1. ハイパーコンバージドノードを使用しないストレージ

ストレージを備えたエッジをデプロイし、ハイパーコンバージドノードをデプロイしない場合は、次の 4 つのロールのいずれかを使用します。

DistributedCompute

DistributedCompute ロールは、ストレージデプロイメントの最初の 3 つのコンピュートノードに使用されます。Image サービスが中央のハブサイトではなくローカルのエッジサイトで使用されるように、**DistributedCompute** ロールには **GlanceApiEdge** サービスが含まれます。追加のノードには、**DistributedComputeScaleOut** ロールを使用します。

DistributedComputeScaleOut

DistributedComputeScaleOut ロールには **HAproxyEdge** サービスが含まれます。これにより、DistributedComputeScaleOut ロールに作成されたインスタンスが、Image サービスの要求をエッジサイトでそのサービスを提供するノードにプロキシー処理することができます。**DistributedCompute** のロールでノードを 3 台デプロイした後に、DistributedComputeScaleOut ロールを使用してコンピュートリソースをスケーリングすることができます。**DistrubutedComputeScaleOut** ロールでホストをデプロイする場合、最低限必要なホスト数はありません。

CephAll

CephAll ロールには、Ceph OSD、Ceph mon、および Ceph Mgr サービスが含まれます。CephAll ロールを使用して、最大 3 つのノードをデプロイすることができます。追加のストレージ容量については、CephStorage ロールを使用します。

CephStorage

CephStorage ロールには Ceph OSD サービスが含まれます。3 台の CephAll ノードが十分なストレージ容量を提供しない場合は、必要なだけ CephStorage ノードを追加します。

7.1.2. ハイパーコンバージドノードを備えたストレージ

ストレージを備えたエッジをデプロイしており、コンピュートとストレージを組み合わせたハイパーコンバージドノードを計画している場合は、次の 2 つのロールのいずれかを使用します。

DistributedComputeHCI

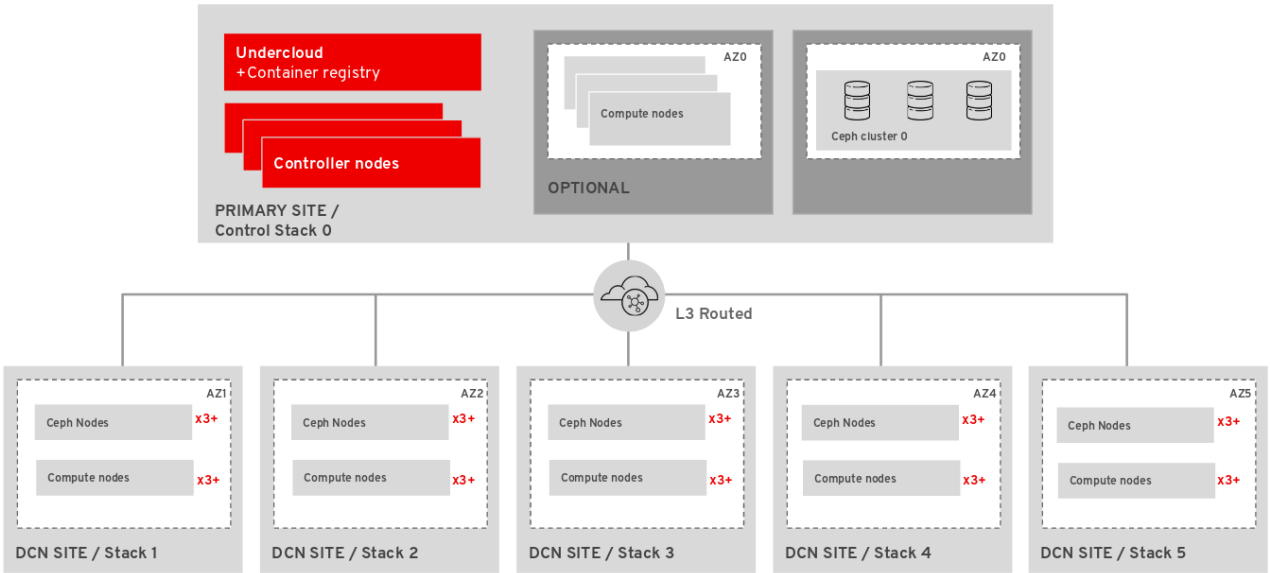
DistributedComputeHCI ロールは、Ceph Management および OSD サービスを追加することで、エッジサイトでのハイパーコンバージドのデプロイメントを可能にします。DistributedComputeHCI ロールを使用する場合は、必ず 3 台のノードを使用する必要があります。

DistributedComputeHCIScaleOut

DistributedComputeHCIScaleOut ロールには **Ceph OSD** サービスが含まれます。これにより、エッジサイトにノードがさらに追加される場合に、コンピュートリソースと共にストレージ容量をスケーリングすることができます。このロールには、イメージのダウンロード要求をエッジサイトの **GlanceAPIEdge** ノードにリダイレクトする **HAproxyEdge** サービスも含まれています。このロールにより、エッジサイトでのハイパーコンバージドのデプロイメントが可能になります。**DistributedComputeHCI** ロールを使用する場合は、必ず 3 台のノードを使用する必要があります。

7.2. ストレージを備えた DCN エッジサイトのアーキテクチャー

ストレージを持つ DCN をデプロイするには、Red Hat Ceph Storage を中央サイトにもデプロイする必要があります。**dcn-storage.yaml** および **cephadm.yaml** 環境ファイルを使用する必要があります。非ハイパーコンバージド Red Hat Ceph Storage ノードが含まれるエッジサイトの場合は、**DistributedCompute**、**DistributedComputeScaleOut**、**CephAll**、および **CephStorage** ロールを使用します。



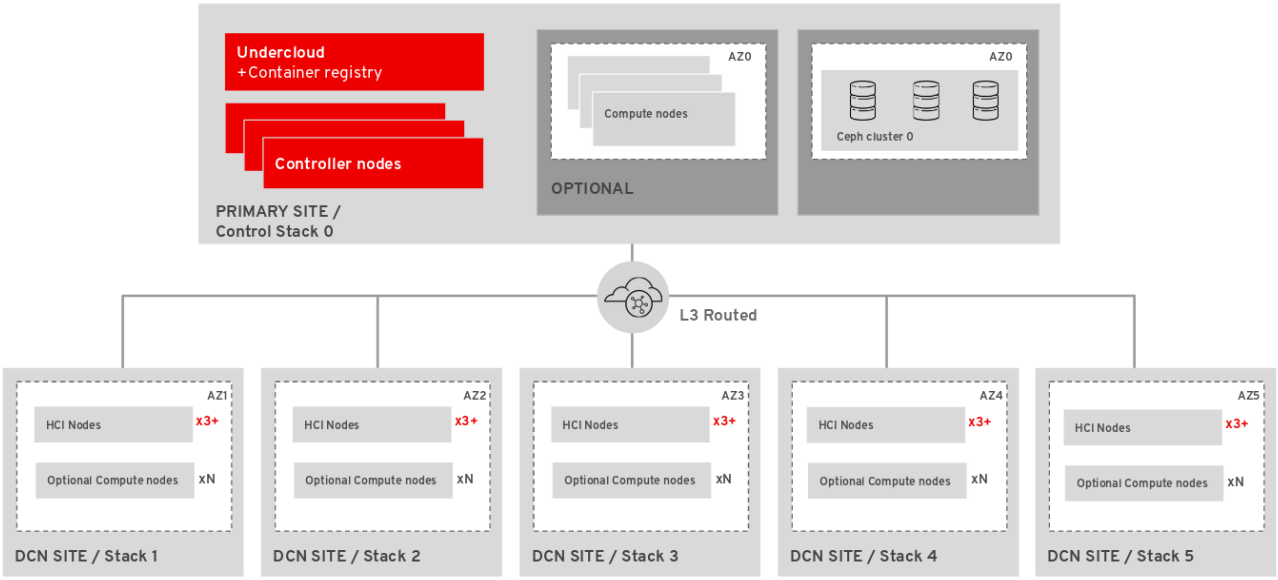
エッジにブロックストレージあり

- Red Hat Ceph Block Devices (RBD) が、Image (glance) サービスのバックエンドとして使用されます。
- マルチバックエンドの Image サービス (glance) が利用できるので、イメージを中央サイトと DCN サイト間でコピーすることができます。

- Block Storage (cinder) サービスはすべてのサイトで利用でき、Red Hat Ceph Block Devices (RBD) ドライバーを使用してアクセスされます。
- Block Storage (cinder) サービスは Compute ノードで実行され、Red Hat Ceph Storage は専用のストレージノードで個別に実行されます。
- Nova の一時ストレージは Ceph (RBD) がベースです。
  詳細は、「ストレージが設定された中央サイトのデプロイ」 を参照してください。

7.3. ハイパーコンバージドストレージを備えた DCN エッジサイトのアーキテクチャー

この設定をデプロイするには、Red Hat Ceph Storage を中央サイトにもデプロイする必要があります。**dcn-storage.yaml** および **cephadm.yaml** 環境ファイルを設定する必要があります。**DistributedComputeHCI** ロールおよび **DistributedComputeHCIScaleOut** ロールを使用します。**DistributedComputeScaleOut** ロールを使用して、Red Hat Ceph Storage サービスの提供に関与しない Compute ノードを追加することもできます。

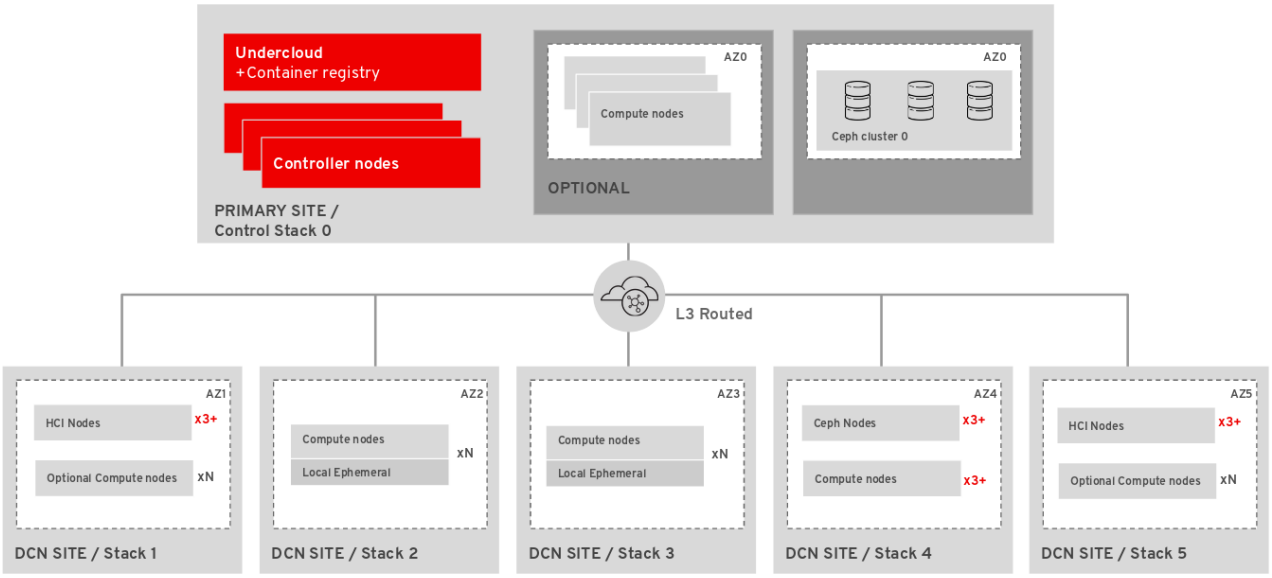


エッジにハイパーコンバージドストレージを搭載

- Red Hat Ceph Block Devices (RBD) が、Image (glance) サービスのバックエンドとして使用されます。
- マルチバックエンドの Image サービス (glance) が利用できるので、イメージを中央サイトと DCN サイト間でコピーすることができます。
- Block Storage (cinder) サービスはすべてのサイトで利用でき、Red Hat Ceph Block Devices (RBD) ドライバーを使用してアクセスされます。
- Block Storage サービスおよび Red Hat Ceph Storage はどちらも Compute ノード上で実行されます。
  詳細は、「ハイパーコンバージドストレージを使用したエッジサイトのデプロイメント」を参照してください。

分散コンピュートアーキテクチャーで Red Hat OpenStack Platform をデプロイする場合、それぞれのサイトに固有の設定を指定して、複数のストレージトポロジーをデプロイするオプションがあります。

ストレージが設定されたエッジサイトをデプロイするには、Red Hat Ceph ストレージと共に中央サイトをデプロイする必要があります。



7.4. ハイパーコンバージドストレージを使用したエッジサイトのデプロイメント

中央サイトをデプロイしたら、エッジサイトを構築し、各エッジロケーションがプライマリーとして自己のストレージバックエンドに接続し、さらに中央サイトのストレージバックエンドにも接続するようにします。スパイン/リーフ型ネットワーク設定に加えて、この設定には ceph が必要とする storage および storage_mgmt ネットワークを含める必要があります。詳しくは、Spine leaf networking を参照してください。イメージサービス (glance) イメージをサイト間で移動することができるように、中央サイトと各エッジサイトのストレージネットワーク間に接続が必要です。

中央サイトが各エッジサイトの mons および osds と通信できるようにしてください。ただし、ストレージ管理ネットワークは OSD のリバランスに使用されるため、サイト境界でストレージ管理ネットワークを終端する必要があります。

前提条件

- 環境に固有の **network_data.yaml** ファイルを作成する必要がある。サンプルファイルは **/usr/share/openstack-tripleo-heat-templates/network-data-samples** にある。
- 環境に固有の **overcloud-baremetal-deploy.yaml** ファイルを作成している。詳細は、オーバークラウド用のベアメタルノードのプロビジョニング を参照してください。
- 中央サイトおよび各アベイラビリティーゾーンまたはストレージサービスが必要な各地区での 3 つの Image サービス (glance) サーバー用ハードウェア。エッジロケーションでは、Image サービスが DistributedComputeHCI ノードにデプロイされる。

手順

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

```
[stack@director ~]$ source ~/stackrc
```

3. 環境ファイル ~/dcn0/dcn0-images-env.yaml を生成します。

```
sudo openstack tripleo container image prepare \
-e containers.yaml \
--output-env-file /home/stack/dcn0/dcn0-images-env.yaml
```

4. dcn0 エッジロケーションの適切なロールを生成します。

```
openstack overcloud roles generate DistributedComputeHCI
DistributedComputeHCIScaleOut \
-o ~/dcn0/dcn0_roles.yaml
```

5. オーバークラウドのネットワークをプロビジョニングします。このコマンドは、オーバークラウドネットワークの定義ファイルを入力として受け取ります。オーバークラウドをデプロイするには、コマンドで出力ファイルを使用する必要があります。

```
(undercloud)$ openstack overcloud network provision \
--output /home/stack/dcn0/overcloud-networks-deployed.yaml \
/home/stack/network_data.yaml
```

重要

network_data.yaml テンプレートに、中央サイトにネットワークをプロビジョニングしたときに含まれなかった追加のネットワークが含まれている場合は、中央サイトでネットワークプロビジョニングコマンドを再実行する必要があります。

```
(undercloud)$ openstack overcloud network provision \
--output /home/stack/central/overcloud-networks-deployed.yaml \
/home/stack/central/network_data.yaml
```

6. ベアメタルインスタンスをプロビジョニングします。このコマンドは、ベアメタルノードの定義ファイルを入力として受け取ります。オーバークラウドをデプロイするには、コマンドで出力ファイルを使用する必要があります。

```
(undercloud)$ openstack overcloud node provision \
--stack dcn0 \
--network-config \
-o /home/stack/dcn0/deployed_metal.yaml \
/home/stack/overcloud-baremetal-deploy.yaml
```

7. ハイパーコンバージドストレージを使用してエッジサイトをデプロイする場合は、次のパラメーターを使用して **initial-ceph.conf** 設定ファイルを作成する必要があります。詳細は、HCI 用の Red Hat Ceph Storage クラスターの設定 を参照してください。

```
[osd]
osd_memory_target_autotune = true
osd_numa_auto_affinity = true
[mgr]
mgr/cephadm/autotune_memory_target_ratio = 0.2
```

8. **deployed_metal.yaml** ファイルを **openstack overcloud ceph deploy** コマンドへの入力として使用します。**openstack overcloud ceph deploy command** は、デプロイされた Ceph クラスターを記述する yaml ファイルを出力します。

```
openstack overcloud ceph deploy \
/home/stack/dcn0/deployed_metal.yaml \
--stack dcn0 \
--config ~/dcn0/initial-ceph.conf \ 1
--output ~/dcn0/deployed_ceph.yaml \
--container-image-prepare ~/containers.yaml \
--network-data ~/network-data.yaml \
--cluster dcn0 \
--roles-data dcn_roles.yaml
```

1 ハイパーコンバージドインフラストラクチャーをデプロイする場合にのみ、initial-ceph.conf を含めます。

9. site-name.yaml 環境ファイルでサイトの命名規則を設定します。Nova アベイラビリティーゾーンと Cinder ストレージアベイラビリティーゾーンが一致している必要があります。

```
parameter_defaults:
    NovaComputeAvailabilityZone: dcn0
    ControllerExtraConfig:
        nova::availability_zone::default_schedule_zone: dcn0
    NovaCrossAZAttach: false
    CinderStorageAvailabilityZone: dcn0
    CinderVolumeCluster: dcn0
    GlanceBackendID: dcn0
```

10. 以下のような内容で glance.yaml テンプレートを設定します。

```
parameter_defaults:
    GlanceEnabledImportMethods: web-download,copy-image
    GlanceBackend: rbd
    GlanceStoreDescription: 'dcn0 rbd glance store'
    GlanceBackendID: dcn0
    GlanceMultistoreConfig:
      central:
        GlanceBackend: rbd
        GlanceStoreDescription: 'central rbd glance store'
        CephClusterName: central
```

11. dcn0 ロケーションのスタックをデプロイします:[d]

```
openstack overcloud deploy \
--deployed-server \
--stack dcn0 \
--templates /usr/share/openstack-tripleo-heat-templates/ \
-r ~/dcn0/dcn0_roles.yaml \
-n ~/dcn0/network-data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/dcn-storage.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/cephadm-rbd-only.yaml \
```

```
-e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
-e /home/stack/overcloud-deploy/central/central-export.yaml \
-e /home/stack/dcn0/deployed_ceph.yaml \
-e /home/stack/dcn-common/central_ceph_external.yaml \
-e /home/stack/dcn0/overcloud-vip-deployed.yaml \
-e /home/stack/dcn0/deployed_metal.yaml \
-e /home/stack/dcn0/overcloud-networks-deployed.yaml \
-e ~/control-plane/glance.yaml
```

7.5. エッジでのインストール済み RED HAT CEPH STORAGE クラスターの使用

既存の Ceph クラスターを使用するように Red Hat OpenStack Platform を設定することができます。これは外部 Ceph のデプロイメントと呼ばれます。

前提条件

- レイテンシー要件を超えないように、DCN サイトのローカルにインストール済みの Ceph クラスターを用意する必要がある。

手順

1. Ceph クラスターに以下のプールを作成します。中央サイトにデプロイしている場合は、**backups** および **metrics** プールを含めます。

```
[root@ceph ~]# ceph osd pool create volumes <_PGnum_>
[root@ceph ~]# ceph osd pool create images <_PGnum_>
[root@ceph ~]# ceph osd pool create vms <_PGnum_>
[root@ceph ~]# ceph osd pool create backups <_PGnum_>
[root@ceph ~]# ceph osd pool create metrics <_PGnum_>
```

<_PGnum_> は配置グループの数に置き換えます。適切な値を判断するには、Ceph Placement Groups (PGs) per Pool Calculator も使用できます。

2. Ceph に OpenStack クライアントユーザーを作成し、Red Hat OpenStack Platform 環境が適切なプールにアクセスできるようにします。

```
ceph auth add client.openstack mon 'allow r' osd 'allow class-read object_prefix rbd_children, allow rwx pool=volumes, allow rwx pool=vms, allow rwx pool=images'
```

返された、提供された Ceph クライアントキーを保存します。アンダークラウドを設定する際に、このキーを **CephClientKey** パラメーターの値として使用します。

注記

このコマンドを中央サイトで実行し、Cinder バックアップまたは Telemetry サービスを使用する予定の場合、コマンドに allow rwx pool=backups, allow pool=metrics を追加します。

3. Ceph Storage クラスターのファイルシステム ID を保存します。Ceph 設定ファイルの **[global]** セクションの **fsid** パラメーターの値は、ファイルシステム ID です。

```
[global]
fsid = 4b5c8c0a-ff60-454b-a1b4-9747aa737d19
...
```

この値は、アンダークラウドを設定する際に **CephClusterFSID** パラメーターの値として使用します。

4. アンダークラウド上で環境ファイルを作成し、ノードが非マネージド Ceph クラスターに接続するように設定します。ceph-external-<SITE>.yaml など、認識可能な命名規則を使用します。ここで、SITE はデプロイメントの場所に置き換えます (例: ceph-external-central.yaml、ceph-external-dcn1.yaml 等)。

```
parameter_defaults:
  # The cluster FSID
  CephClusterFSID: '4b5c8c0a-ff60-454b-a1b4-9747aa737d19'
  # The CephX user auth key
  CephClientKey: 'AQDLOh1VgEp6FRAAFzT7Zw+Y9V6JJExQAsRnRQ=='
  # The list of IPs or hostnames of the Ceph monitors
  CephExternalMonHost: '172.16.1.7, 172.16.1.8, 172.16.1.9'
  # The desired name of the generated key and conf files
  CephClusterName: dcn1
```

   a. CephClusterFSID および CephClientKey パラメーターには、以前に保存した値を使用します。

   b. CephExternalMonHost パラメーターの値として、Ceph モニターの ip アドレスのコンマ区切りリストを使用します。

   c. **CephClusterName** パラメーターには、エッジサイト間で一意の値を選択する必要があります。名前を再利用すると、設定ファイルが上書きされます。

5. Red Hat OpenStack Platform director を使用して Red Hat Ceph Storage を中央サイトにデプロイした場合、ceph の設定を環境ファイル **central_ceph_external.yaml** にエクスポートできます。この環境ファイルにより、DCN サイトが中央ハブの Ceph クラスターに接続されので、詳細は前の手順でデプロイした Ceph クラスターに固有のものです。

```
sudo -E openstack overcloud export ceph \
--stack central \
--output-file /home/stack/dcn-common/central_ceph_external.yaml
```

中央サイトの外に Red Hat Ceph Storage がデプロイされている場合、**openstack overcloud export ceph** コマンドを使用して **central_ceph_external.yaml** ファイルを生成することはできません。代わりに central_ceph_external.yaml ファイルを手動で作成する必要があります。

```
parameter_defaults:
  CephExternalMultiConfig:
    - cluster: "central"
      fsid: "3161a3b4-e5ff-42a0-9f53-860403b29a33"
      external_cluster_mon_ips: "172.16.11.84, 172.16.11.87, 172.16.11.92"
      keys:
        - name: "client.openstack"
          caps:
            mgr: "allow *"
            mon: "profile rbd"
            osd: "profile rbd pool=vms, profile rbd pool=volumes, profile rbd pool=images"
```

```
      key: "AQD29WteAAAAABAAphgOjFD7nyjdYe8Lz0mQ5Q=="
      mode: "0600"
  dashboard_enabled: false
  ceph_conf_overrides:
    client:
      keyring: /etc/ceph/central.client.openstack.keyring
```

6. 各サイトについて同様の詳細を記述した環境ファイルを作成し、中央サイトには非マネージド Red Hat Ceph Storage クラスターを使用します。**openstack overcloud export ceph** コマンドは、非マネージド Red Hat Ceph Storage クラスターがあるサイトでは機能しません。中央サイトを更新すると、このファイルによって、中央サイトはエッジサイトのストレージクラスターをセカンダリーロケーションとして使用できるようになります。

```
parameter_defaults:
  CephExternalMultiConfig:
cluster: dcn1
…
cluster: dcn2
…
```

7. オーバークラウドをデプロイする場合は、環境ファイル external-ceph.yaml、ceph-external-<SITE>.yaml、および central_ceph_external.yaml を使用します。

```
openstack overcloud deploy \
    --stack dcn1 \
    --templates /usr/share/openstack-tripleo-heat-templates/ \
    -r ~/dcn1/roles_data.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/external-ceph.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/dcn-storage.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
    -e /home/stack/dnc1/ceph-external-dcn1.yaml  \
    ....
    -e /home/stack/overcloud-deploy/central/central-export.yaml \
    -e /home/stack/dcn-common/central_ceph_external.yaml \
    -e /home/stack/dcn1/dcn_ceph_keys.yaml \
    -e /home/stack/dcn1/role-counts.yaml \
    -e /home/stack/dcn1/ceph.yaml \
    -e /home/stack/dcn1/site-name.yaml \
    -e /home/stack/dcn1/tuning.yaml \
    -e /home/stack/dcn1/glance.yaml
```

8. すべてのエッジサイトが配置された後、中央サイトを再デプロイします。

7.6. 中央サイトの更新

サンプルの手順を使用してすべてのエッジサイトを設定およびデプロイしたら、中央の Image サービスがイメージをエッジサイトにプッシュできるように、中央サイトの設定を更新します。

警告

この手順では、Image サービス (glance) を再起動し、長く実行されている Image サービスプロセスを中断します。たとえば、セントラルイメージサービスサーバーから DCN イメージサービスサーバーにイメージをコピーしている場合、そのイメージコピーは中断されるため、再起動する必要があります。詳細は、Clearing residual data after interrupted Image service processes を参照してください。

手順

1. 以下のような内容で **~/central/glance_update.yaml** ファイルを作成します。以下の例には、2 つのエッジサイト dcn0 および dcn1 の設定が含まれています。

```
parameter_defaults:
  GlanceEnabledImportMethods: web-download,copy-image
  GlanceBackend: rbd
  GlanceStoreDescription: 'central rbd glance store'
  CephClusterName: central
  GlanceBackendID: central
  GlanceMultistoreConfig:
    dcn0:
      GlanceBackend: rbd
      GlanceStoreDescription: 'dcn0 rbd glance store'
      CephClientUserName: 'openstack'
      CephClusterName: dcn0
      GlanceBackendID: dcn0
    dcn1:
      GlanceBackend: rbd
      GlanceStoreDescription: 'dcn1 rbd glance store'
      CephClientUserName: 'openstack'
      CephClusterName: dcn1
      GlanceBackendID: dcn1
```

2. **dcn_ceph.yaml** ファイルを作成します。以下の例では、このファイルは、エッジサイト **dcn0** および **dcn1** の Ceph クラスターのクライアントとして、中央サイトの glance サービスを設定します。

```
openstack overcloud export ceph \
--stack dcn0,dcn1 \
--output-file ~/central/dcn_ceph.yaml
```

3. 元のテンプレートを使用して中央サイトを再デプロイする際に、新たに作成した **dcn_ceph.yaml** および **glance_update.yaml** ファイルを追加します。

注記

中央スタックの作成時にリーフネットワークが最初に提供されていなかった場合は、他のエッジサイトからの **deployed_metal.yaml** を **overcloud deploy** コマンドに追加します。

```
openstack overcloud deploy \
--deployed-server \
--stack central \
--templates /usr/share/openstack-tripleo-heat-templates/ \
-r ~/central/central_roles.yaml \
-n ~/network-data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/dcn-storage.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/cephadm.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
-e /home/stack/central/overcloud-networks-deployed.yaml \
-e /home/stack/central/overcloud-vip-deployed.yaml \
-e /home/stack/central/deployed_metal.yaml \
-e /home/stack/central/deployed_ceph.yaml \
-e /home/stack/central/dcn_ceph.yaml \
-e /home/stack/central/glance_update.yaml
```

4. 中央サイトのコントローラーで、**cinder-volume** サービスを再起動します。**cinder-backup** サービスと一緒に中央サイトをデプロイしている場合は、**cinder-backup** サービスも再起動してください。

```
ssh tripleo-admin@controller-0 sudo pcs resource restart openstack-cinder-volume
ssh tripleo-admin@controller-0 sudo pcs resource restart openstack-cinder-backup
```

7.6.1. イメージサービスプロセス中断後の残存データの消去

中央サイトを再起動すると、長時間稼働している Image サービス (glance) のプロセスが中断されます。これらのプロセスを再起動する前に、まず、再起動した Controller ノード、および Ceph と Image サービスのデータベースに残っているデータをクリーンアップする必要があります。

手順

1. 再起動した Controller ノードの残留データを確認し、消去します。ステージングストア用の**glance-api.conf** ファイル内のファイルと、Image サービスデータベース内の対応するイメージ (例: **<image_ID>.raw**) を比較します。
   - これらの対応するイメージのステータスがインポート中の場合、イメージを再作成する必要があります。
   - イメージのステータスがアクティブの場合、ステージングからデータを削除し、コピーのインポートを再起動する必要があります。
2. Ceph ストアの残留データをチェックおよび消去します。ステージングエリアから消去したイメージは、そのイメージを含む Ceph ストアの **stores** プロパティーに一致するレコードがなければなりません。Ceph におけるイメージ名は、Image サービスデータベースのイメージ ID です。
3. イメージサービスデータベースのクリア中断されたインポートジョブからインポートステータスになっているイメージをすべて消去します。

```
$ glance image-delete <image_id>
```

7.7. DCN への RED HAT CEPH STORAGE DASHBOARD のデプロイ

手順

Red Hat Ceph Storage Dashboard を中央サイトにデプロイするには、Red Hat Ceph Storage Dashboard のオーバークラウドデプロイメントへの追加 を参照してください。中央サイトをデプロイする前に、これらの手順を完了する必要があります。

Red Hat Ceph Storage Dashboard をエッジロケーションにデプロイするには、中央サイトで完了した手順と同じ手順を実行します。ただし、以下の手順を実施する必要があります。

- 高可用性の仮想 IP を作成するには、負荷分散用に独自のソリューションをデプロイする必要があります。エッジサイトでは haproxy および pacemaker はデプロイされません。Red Hat Ceph Storage Dashboard をエッジロケーションにデプロイする場合、デプロイメントはストレージネットワーク上で公開されます。Dashboard は、負荷分散ソリューションなしに異なる IP アドレスを持つ 3 つの DistributedComputeHCI ノードにそれぞれインストールされます。

Ceph Dashboard を公開することのできる仮想 IP をホストする追加のネットワークを作成することができます。複数のスタックでネットワークリソースを再使用しないでください。ネットワークリソースの再利用に関する詳細は、複数スタックでのネットワークリソースの再利用 を参照してください。

この追加ネットワークリソースを作成するには、提供される **network_data_dashboard.yaml** heat テンプレートを使用します。作成されるネットワークの名前は **StorageDashboard** です。

手順

1. Red Hat OpenStack Platform director に **stack** としてログインします。
2. **DistributedComputeHCIDashboard** ロールおよびお使いの環境に適したその他のロールを生成します。

```
openstack overcloud roles generate DistributedComputeHCIDashboard -o ~/dnc0/roles.yaml
```

3. オーバークラウドデプロイコマンドに **roles.yaml** および **network_data_dashboard.yaml** を追加します。

```
$ openstack overcloud deploy --templates \
-r ~/<dcn>/<dcn_site_roles>.yaml \
-n /usr/share/openstack-tripleo-heat-templates/network_data_dashboard.yaml \
-e <overcloud_environment_files> \
...
-e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/cephadm-rbd-only.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/ceph-dashboard.yaml \
```

注記

デプロイメントでは、Dashboard が有効なストレージネットワーク上の 3 つの IP アドレスが提供されます。

検証

Dashboard が中央サイトで動作し、Ceph クラスターから表示されるデータが正しいことを確認するには、Ceph Dashboard へのアクセス を参照してください。

同様の手順で Dashboard がエッジロケーションで動作していることを確認できますが、エッジロケーションにロードバランサーが存在しないので例外があります。

1. 選択したスタックに固有の Dashboard 管理者のログイン認証情報を取得します。

```
grep grafana_admin /home/stack/config-download/<stack>/cephadm/cephadm-extra-vars-heat.yml
```

2. 選択したスタックに固有のインベントリー **/home/stack/config-download/<stack>/cephadm/inventory.yml** 内で、DistributedComputeHCI ロールホストリストを見つけて、3 つの **storage_ip** 値をすべて保存します。以下の例では、最初の 2 つの Dashboard IP は 172.16.11.84 と 172.16.11.87 です。

```
DistributedComputeHCI:
  hosts:
    dcn1-distributed-compute-hci-0:
      ansible_host: 192.168.24.16
...
storage_hostname: dcn1-distributed-compute-hci-0.storage.localdomain
storage_ip: 172.16.11.84
...
    dcn1-distributed-compute-hci-1:
ansible_host: 192.168.24.22
...
storage_hostname: dcn1-distributed-compute-hci-1.storage.localdomain
storage_ip: 172.16.11.87
```

3. これらの IP アドレスにアクセス可能な場合は、Ceph Dashboard がそのいずれかでアクティブであることを確認することができます。これらの IP アドレスはストレージネットワーク上にあり、ルーティングされません。これらの IP アドレスが利用できない場合、インベントリーから取得する 3 つの IP アドレスのロードバランサーを設定して、検証用に仮想 IP アドレスを取得する必要があります。

第8章 エッジサイトでのネットワークトラフィックの負荷分散

Red Hat OpenStack Platform (RHOSP) Load-balancing サービス (octavia) を使用して、エッジサイトにロードバランサーを作成し、トラフィックのスループットを向上させ、レイテンシーを短縮できます。

このセクションに含まれるトピックは次のとおりです。

- Load-balancing サービスアベイラビリティーゾーンのネットワークリソースの作成
- Load-balancing サービスアベイラビリティーゾーンの作成
- アベイラビリティーゾーンでのロードバランサーの作成

8.1. LOAD-BALANCING サービスアベイラビリティーゾーンのネットワークリソースの作成

Red Hat OpenStack Platform (RHOSP) Load-balancing サービス (octavia) アベイラビリティーゾーン (AZ) を作成するには、RHOSP 管理者として Ansible Playbook **octavia-dcn-deployment.yaml** を実行する必要があります。

octavia-dcn-deployment.yaml を実行すると、Load-balancing サービス AZ に必要なネットワーク、サブネット、ルーターなどのネットワークリソースを作成できます。Playbook には、AZ 名と各 AZ が使用する管理ネットワークを指定した設定入力ファイル **octavia-dcn-parameters.yaml** を指定します。

Playbook を実行して必要なネットワークリソースを作成した後、プロジェクト (テナント) ユーザーが分散コンピュートノード (DCN) ロケールに適した AZ にロードバランサーを作成できるようにする前に、実際の RHOSP Load-balancing サービス AZ を作成する必要があります。

この手順では、**az-central**、**az-dcn1**、および **az-dcn2** という名前の 3 つの Load-balancing サービス AZ に必要なネットワークリソースを作成する方法を示しています。これらの Load-balancing サービス AZ 名は、Compute サービス AZ の名前と一致し、このデプロイメントで使用される 3 つの DCN の名前でもあります。

前提条件

- 作成する Load-balancing サービス AZ ごとに 1 つの Compute サービス (nova) AZ がある。
- また、作成する Load-balancing サービス AZ ごとに Networking サービス (neutron) AZ が 1 つある。これらの Networking サービス AZ は、Compute サービス AZ の名前と一致する。
- Load-balancing サービスプロバイダードライバーが amphora である。OVN プロバイダードライバーは AZ をサポートしません。
- **admin** ロールが割り当てられた RHOSP ユーザーを使用する。

手順

1. Source コマンドで認証情報ファイルを読み込みます。

   例

   ```
   $ source ~/centralrc
   ```

2. **octavia-dcn-parameters.yaml** ファイルを作成し、以下に示す構文を使用して、Ansible Playbook で必要なネットワークリソースを作成する Load-balancing サービス AZ とその管理ネットワークを追加します。
octavia_controller_AZ_name の値は、すべての Load-balancer サービスが実行される AZ の名前です。

```
octavia_controller_availability_zone: <octavia_controller_AZ_name>
octavia_availability_zones:
  <octavia_controller_AZ_name>: # no cidr needed, it uses the already existing subnet
  <octavia_AZ_n>:
    lb_mgmt_subnet_cidr: <CIDR_address_n>
  <octavia_AZ_n2>:
    lb_mgmt_subnet_cidr: <CIDR_address_n2>
```

重要

指定する Load-balancing サービス AZ の名前は、既存の Compute サービス AZ の名前と一致する必要があります。Compute サービス AZ の名前を取得するには、**openstack availability zone list --compute** を実行します。

Ansible Playbook は、AZ ごとにネットワーク、サブネット、ルーターを作成し、それぞれ **lb-mgmt-<AZ_name>-net**、**lb-mgmt-<AZ_name>-subnet**、および **lb-mgmt-<AZ_name>-router** という規則に従って、**octavia-dcn-parameters.yaml** で指定した AZ 名を使用して名前を付けます。例外は **octavia_controller_AZ_name** のネットワークリソースです。Playbook は既存の負荷分散管理ネットワークとサブネット、それぞれ **lb-mgmt-net** と **lb-mgmt-subnet** を使用し、**lb-mgmt-router** という名前の関連ルーターを作成します。

この例では、**az-central**、**az-dcn1**、および **az-dcn2 の** 3 つの AZ が指定されています。**az-central** AZ は、既存の負荷分散管理ネットワーク **lb-mgmt-net** を使用します。他の 2 つの AZ は、それぞれ **172.47.0.0/16** と **172.48.0.0/16** を使用します。

例

```
octavia_controller_availability_zone: az-central
octavia_availability_zones:
  az-central: # no cidr needed; it uses the existing subnet
  az-dcn1:
    lb_mgmt_subnet_cidr: 172.47.0.0/16
  az-dcn2:
    lb_mgmt_subnet_cidr: 172.48.0.0/16
```

3. Ansible Playbook の **octavia-dcn-deployment.yaml** を実行し、**octavia-dcn-parameters.yaml** で作成した AZ 定義を含めます。

例

```
$ ansible-playbook -i overcloud-deploy/central/config-download/\
central/tripleo-ansible-inventory.yaml \
/usr/share/ansible/tripleo-playbooks/octavia-dcn-deployment.yaml \
-e @octavia-dcn-parameters.yaml -e stack=central -v
```

検証

1. 必要な **lb-mgmt-*** サブネットが存在することを確認します。

```
$ openstack subnet list -c Name -c Subnet
```

出力例

```
+-------------------------+------------------+
| Name                    | Subnet           |
+-------------------------+------------------+
| lb-mgmt-az-dcn2-subnet  | 172.48.0.0/16    |
| segment5                | 10.0.20.0/24     |
| segment3                | 10.101.30.0/24   |
| segment2                | 10.101.20.0/24   |
| lb-mgmt-az-dcn1-subnet  | 172.47.0.0/16    |
| heat_tempestconf_subnet | 192.168.199.0/24 |
| segment4                | 10.0.10.0/24     |
| lb-mgmt-subnet          | 172.24.0.0/16    |
| segment1                | 10.101.10.0/24   |
| lb-mgmt-backbone-subnet | 172.49.0.0/16    |
| segment6                | 10.0.30.0/24     |
+-------------------------+------------------+
```

2. 必要な仮想ルーターが存在することを確認します。

```
$ openstack router list -c Name -c Status
```

出力例

```
+------------------------+--------+
| Name                   | Status |
+------------------------+--------+
| lb-mgmt-az-dcn2-router | ACTIVE |
| lb-mgmt-az-dcn1-router | ACTIVE |
| lb-mgmt-router         | ACTIVE |
+------------------------+--------+
```

次のステップ

- Load-balancing サービスのアベイラビリティーゾーンの作成

8.2. LOAD-BALANCING サービスのアベイラビリティーゾーンの作成

Red Hat OpenStack Platform (RHOSP) の Load-balancing サービス (octavia) を使用すると、RHOSP 管理者はアベイラビリティーゾーン (AZ) を作成できます。これにより、プロジェクトユーザーは分散コンピュートノード (DCN) 環境にロードバランサーを作成して、トラフィックスループットを向上させ、レイテンシーを短縮できます。

Load-balancing サービス AZ を作成するには手順を 2 つ実行する必要があります。RHOSP 管理者は、まず AZ プロファイルを作成し、次にそのプロファイルを使用してユーザーに表示される Load-balancing サービス AZ を作成する必要があります。

AZ プロファイルには次のものが必要です。

- Compute サービス (nova) AZ の名前。
- 使用する管理ネットワーク。
  複数の管理ネットワークがあり、それぞれの AZ に固有のネットワークが 1 つある。中央の AZ は既存の負荷分散管理ネットワーク **lb-mgmt-net** を使用し、追加の AZ はそれぞれのネットワーク **lb-mgmt-<AZ_name>-net** を使用します (例: **lb-mgmt-az-dcn1-net)**、**lb-mgmt-az-dcn2-net** など)。

前提条件

- DCN 環境があり、その環境で **octavia-dcn-deployment.yaml** Ansible Playbook を実行して、必要なネットワークリソースが作成されている。
  詳細は、負荷分散サービスアベイラビリティゾーン用のネットワークリソースの作成 を参照してください。
- Load-balancing サービスプロバイダードライバーが amphora である。OVN プロバイダードライバーは AZ をサポートしません。
- **admin** ロールが割り当てられた RHOSP ユーザーを使用する。

手順

1. Source コマンドで認証情報ファイルを読み込みます。

   例

   ```
   $ source ~/centralrc
   ```

2. Load-balancing サービス AZ に名前を付けるために使用する Compute サービス AZ の名前を収集します。

   重要

   作成する Load-balancing サービス AZ の名前は、Compute サービス AZ の名前と一致する必要があります。

   ```
   $ openstack availability zone list --compute
   ```

   出力例

   ```
   +------------+-------------+
   | Zone Name  | Zone Status |
   +------------+-------------+
   | az-central | available   |
   | az-dcn1    | available   |
   | az-dcn2    | available   |
   | internal   | available   |
   +------------+-------------+
   ```

3. Load-balancing サービス AZ の作成に使用する管理ネットワークの ID を収集します。

   ```
   $ openstack network list -c Name -c ID
   ```

出力例

```
+--------------------------------------+-------------------------+
| ID                                   | Name                    |
+--------------------------------------+-------------------------+
| 10458d6b-e7c9-436f-92d9-711677c9d9fd | lb-mgmt-az-dcn2-net     |
| 662a94f5-51eb-4a4c-86c4-52dcbf471ef9 | lb-mgmt-net             |
| 6b97ef58-2a25-4ea5-931f-b7c07cd09474 | lb-mgmt-backbone-net    |
| 99f4215b-fad8-432d-8444-1f894154dc30 | heat_tempestconf_network |
| a2884aaf-846c-4936-9982-3083f6a71d9b | lb-mgmt-az-dcn1-net     |
| d7f7de6c-0e84-49e2-9042-697fa85d2532 | public                  |
| e887a9f9-15f7-4854-a797-033cedbfe5f3 | public2                 |
+--------------------------------------+-------------------------+
```

4. AZ プロファイルを作成します。この手順を繰り返して、作成する Load-balancing サービス AZ ごとに AZ プロファイルを作成します。

```
$ openstack loadbalancer availabilityzoneprofile create \
--name <AZ_profile_name> --provider amphora --availability-zone-data '{"compute_zone": "
<compute_AZ_name>","management_network": "<lb_mgmt_AZ_net_UUID>"}'
```

例: az-central のプロファイルを作成する

この例では、Compute AZ (**az-central**) で実行される Compute ノード上の管理ネットワーク (**lb-mgmt-net**) を使用する AZ プロファイル (**az_profile_central**) が作成されます。

```
$ openstack loadbalancer availabilityzoneprofile create \
--name az_profile_central --provider amphora --availability-zone-data \
'{"compute_zone": "az-central","management_network": \
"662a94f5-51eb-4a4c-86c4-52dcbf471ef9"}'
```

5. 手順 4 を繰り返して、作成する Load-balancing サービス AZ ごとに AZ プロファイルを作成します。

例: az-dcn1 のプロファイルを作成する

この例では、コンピュート AZ (**az-dcn1**) で実行される Compute ノード上の管理ネットワーク (**lb-mgmt-az-dcn1-net**) を使用する AZ プロファイル (**az-profile-dcn1**) が作成されます。

```
$ openstack loadbalancer availabilityzoneprofile create \
--name az-profile-dcn1 --provider amphora --availability-zone-data \
'{"compute-zone": "az-dcn1","management-network": \
"a2884aaf-846c-4936-9982-3083f6a71d9b"}'
```

例: az-dcn2 のプロファイルを作成する

この例では、Compute AZ (**az-dcn2**) で実行される Compute ノード上の管理ネットワーク (**lb-mgmt-az-dcn2-net**) を使用する AZ プロファイル (**az-profile-dcn2**) が作成されます。

```
$ openstack loadbalancer availabilityzoneprofile create \
--name az-profile-dcn2 --provider amphora --availability-zone-data \
'{"compute-zone": "az-dcn2","management-network": \
"10458d6b-e7c9-436f-92d9-711677c9d9fd"}'
```

6. AZ プロファイルを使用して、Load-balancing サービス AZ を作成します。各 AZ に適切なプロファイルを使用して、追加の AZ に対してこの手順を繰り返します。

例: AZ: az-central を作成する

この例では、AZ プロファイル (**az-profile-central**) を使用して、Load-balancing サービス AZ (**az-central**) が作成されます。

```
$ openstack loadbalancer availabilityzone create --name az-central \
--availabilityzoneprofile az-profile-central \
--description "AZ for Headquarters" --enable
```

例: AZ: az-dcn1 を作成する

この例では、AZ プロファイル (**az-profile-az-dcn1**) を使用して、Load-balancing サービス AZ (**az-dcn1**) が作成されます。

```
$ openstack loadbalancer availabilityzone create --name az-dcn1 \
--availabilityzoneprofile az-profile-az-dcn1 \
--description "AZ for South Region" --enable
```

例: AZ: az-dcn2 を作成する

この例では、AZ プロファイル (**az-profile-az-dcn2**) を使用して、Load-balancing サービス AZ (**az-dcn2**) が作成されます。

```
$ openstack loadbalancer availabilityzone create --name az-dcn2 \
--availabilityzoneprofile az-profile-az-dcn2 \
--description "AZ for North Region" --enable
```

検証

- AZ (**az-central**) が作成されたことを確認します。各 AZ に適切な名前を使用して、追加の AZ に対してこの手順を繰り返します。

例: az-central を検証する

```
$ openstack loadbalancer availabilityzone show az-central
```

出力例

```
+-------------------------------+--------------------------------------+
| Field                         | Value                                |
+-------------------------------+--------------------------------------+
| name                          | az-central                           |
| availability_zone_profile_id | 5ed25d22-52a5-48ad-85ec-255910791623 |
| enabled                       | True                                 |
| description                   | AZ for Headquarters                  |
+-------------------------------+--------------------------------------+
```

例: az-dcn1 を検証する

```
$ openstack loadbalancer availabilityzone show az-dcn1
```

出力例

```
+-----------------------------+-----------------------------------+
| Field                       | Value                             |
+-----------------------------+-----------------------------------+
| name                        | az-dcn1                           |
| availability_zone_profile_id | e0995a82-8e67-4cea-b32c-256cd61f9cf3 |
| enabled                     | True                              |
| description                 | AZ for South Region               |
+-----------------------------+-----------------------------------+
```

例: az-dcn2 を検証する

```
$ openstack loadbalancer availabilityzone show az-dcn2
```

出力例

```
+-----------------------------+-----------------------------------+
| Field                       | Value                             |
+-----------------------------+-----------------------------------+
| name                        | az-dcn2                           |
| availability_zone_profile_id | 306a4725-7dac-4046-8f16-f2e668ee5a8d |
| enabled                     | True                              |
| description                 | AZ for North Region               |
+-----------------------------+-----------------------------------+
```

次のステップ

- アベイラビリティーゾーンでのロードバランサーの作成

関連情報

- **コマンドラインインターフェイスリファレンス** の loadbalancer availabilityzoneprofile の作成
- **コマンドラインインターフェイスリファレンス** の ロードバランサーアベイラビリティーゾーンの作成

8.3. アベイラビリティーゾーンでのロードバランサーの作成

Red Hat OpenStack Platform (RHOSP) の Load-balancingサービス (octavia) を使用すると、分散コンピュートノード (DCN) 環境のアベイラビリティーゾーン (AZ) にロードバランサーを作成して、トラフィックのスループットを向上させ、レイテンシーを短縮できます。

前提条件

- RHOSP 管理者が Load-balancing サービス AZ を提供しておく。
- ロードバランサーに関連付けられた仮想 IP (VIP) ネットワークは、ロードバランサーがメンバーになっている AZ で利用できる必要があります。

手順

1. Source コマンドで認証情報ファイルを読み込みます。

例

```
$ source ~/centralrc
```

2. DCN 環境のロードバランサーを作成するには、**loadbalancer create** コマンドを **--availability-zone** オプションとともに使用し、適切な AZ を指定します。

例

たとえば、アベイラビリティーゾーン (**az-central**) のパブリックサブネット (**public_subnet**) 上に終了されない HTTPS ロードバランサー (**lb1**) を作成するには、次のコマンドを入力します。

```
$ openstack loadbalancer create --name lb1 --vip-subnet-id \
public_subnet --availability-zone az-central
```

3. リスナー、プール、ヘルスモニター、およびロードバランサーのメンバーを追加して、ロードバランサーの作成を続けます。
詳細は、load balancing as a service の設定ガイドを参照してください。

検証

- ロードバランサー (lb1) がアベイラビリティーゾーン (**az-central**) のメンバーであることを確認します。

例

```
$ openstack loadbalancer show lb1
```

出力例

```
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| admin_state_up      | True                                 |
| availability_zone   | az-central                           |
| created_at          | 2023-07-12T16:35:05                  |
| description         |                                      |
| flavor_id           | None                                 |
| id                  | 85c7e567-a0a7-4fcb-af89-a0bbc9abe3aa |
| listeners           |                                      |
| name                | lb1                                  |
| operating_status    | ONLINE                               |
| pools               |                                      |
| project_id          | d303d3bda9b34d73926dc46f4d0cb4bc     |
| provider            | amphora                              |
| provisioning_status | ACTIVE                               |
| updated_at          | 2023-07-12T16:36:45                  |
| vip_address         | 10.101.10.229                        |
| vip_network_id      | d7f7de6c-0e84-49e2-9042-697fa85d2532 |
| vip_port_id         | 7f916764-d171-4317-9c86-a1750a54b16e |
| vip_qos_policy_id   | None                                 |
```

```
| vip_subnet_id       | a421cbcf-c5db-4323-b7ab-1df20ee6acab |
| tags                |                                      |
+---------------------+--------------------------------------+
```

関連情報

- Load-balancing サービスのアベイラビリティーゾーンの作成
- **コマンドラインインターフェイスリファレンス** の loadbalancer
- **load balancing as a service の設定** ガイド

第9章 DISTRIBUTEDCOMPUTEHCI ノードの置き換え

ハードウェアのメンテナンス中に、エッジサイトの DistributedComputeHCI ノードのスケールダウン、スケールアップ、または置き換えが必要になる場合があります。DistributedComputeHCI ノードを置き換えるには、置き換えるノードからサービスを削除し、ノード数をスケールダウンしてから、これらのノードを再びスケールアップする手順に従います。

9.1. RED HAT CEPH STORAGE サービスの削除

クラスターから HCI (ハイパーコンバージド) ノードを削除する前に、Red Hat Ceph Storage サービスを削除する必要があります。Red Hat Ceph サービスを削除するには、削除するノードのクラスターサービスから **ceph-osd** サービスを無効にしてから、**mon** サービス、**mgr** サービス、および **osd** サービスを停止して無効にする必要があります。

手順

1. アンダークラウド上で、SSH を使用して、削除する DistributedComputeHCI ノードに接続します。

```
$ ssh tripleo-admin@<dcn-computehci-node>
```

2. cephadm シェルを起動します。削除するホストの設定ファイルとキーリングファイルを使用します。

```
$ sudo cephadm shell --config /etc/ceph/dcn2.conf \
--keyring /etc/ceph/dcn2.client.admin.keyring
```

3. 後の手順で参照できるように、削除する DistributedComputeHCI ノードに関連付けられている OSD (オブジェクトストレージデバイス) を記録します。

```
[ceph: root@dcn2-computehci2-1 ~]# ceph osd tree -c /etc/ceph/dcn2.conf
…
-3       0.24399     host dcn2-computehci2-1
 1   hdd 0.04880         osd.1                    up  1.00000 1.00000
 7   hdd 0.04880         osd.7                    up  1.00000 1.00000
11   hdd 0.04880         osd.11                   up  1.00000 1.00000
15   hdd 0.04880         osd.15                   up  1.00000 1.00000
18   hdd 0.04880         osd.18                   up  1.00000 1.00000
…
```

4. SSH を使用して同じクラスター内の別のノードに接続し、クラスターからモニターを削除します。

```
$ sudo cephadm shell --config /etc/ceph/dcn2.conf \
--keyring /etc/ceph/dcn2.client.admin.keyring

[ceph: root@dcn-computehci2-0]# ceph mon remove dcn2-computehci2-1 -c
/etc/ceph/dcn2.conf
removing mon.dcn2-computehci2-1 at [v2:172.23.3.153:3300/0,v1:172.23.3.153:6789/0],
there will be 2 monitors
```

5. SSH を使用して、クラスターから削除するノードに再度ログインします。

6. **mgr** サービスを停止し、無効にします。

```
[tripleo-admin@dcn2-computehci2-1 ~]$ sudo systemctl --type=service | grep ceph
ceph-crash@dcn2-computehci2-1.service    loaded active     running       Ceph crash dump
collector
ceph-mgr@dcn2-computehci2-1.service      loaded active     running       Ceph Manager

[tripleo-admin@dcn2-computehci2-1 ~]$ sudo systemctl stop ceph-mgr@dcn2-computehci2-
1

[tripleo-admin@dcn2-computehci2-1 ~]$ sudo systemctl --type=service | grep ceph
ceph-crash@dcn2-computehci2-1.service  loaded active running Ceph crash dump collector

[tripleo-admin@dcn2-computehci2-1 ~]$ sudo systemctl disable ceph-mgr@dcn2-
computehci2-1
Removed /etc/systemd/system/multi-user.target.wants/ceph-mgr@dcn2-computehci2-
1.service.
```

7. cephadm シェルを起動します。

```
$ sudo cephadm shell --config /etc/ceph/dcn2.conf \
--keyring /etc/ceph/dcn2.client.admin.keyring
```

8. ノードの **mgr** サービスがクラスターから削除されていることを確認します。

```
[ceph: root@dcn2-computehci2-1 ~]# ceph -s

cluster:
   id:     b9b53581-d590-41ac-8463-2f50aa985001
   health: HEALTH_WARN
           3 pools have too many placement groups
           mons are allowing insecure global_id reclaim

 services:
   mon: 2 daemons, quorum dcn2-computehci2-2,dcn2-computehci2-0 (age 2h)
   mgr: dcn2-computehci2-2(active, since 20h), standbys: dcn2-computehci2-0 (1)
   osd: 15 osds: 15 up (since 3h), 15 in (since 3h)

 data:
   pools:   3 pools, 384 pgs
   objects: 32 objects, 88 MiB
   usage:   16 GiB used, 734 GiB / 750 GiB avail
   pgs:     384 active+clean
```

(1) mgr サービスが正常に削除されると、mgr サービスが削除されたノードは表示されなくなります。

9. Red Hat Ceph Storage 仕様をエクスポートします。

```
[ceph: root@dcn2-computehci2-1 ~]# ceph orch ls --export > spec.yml
```

10. **spec.yaml** ファイルで仕様を編集します。

- spec.yml からホストの <dcn-computehci-node> のインスタンスをすべて削除します。

- 以下の <dcn-computehci-node> エントリーのインスタンスをすべて削除します。
  - service_type: osd
  - service_type: mon
  - service_type: host

11. Red Hat Ceph Storage 仕様を再適用します。

```
[ceph: root@dcn2-computehci2-1 /]# ceph orch apply -i spec.yml
```

12. **ceph osd tree** を使用して特定した OSD を削除します。

```
[ceph: root@dcn2-computehci2-1 /]# ceph orch osd rm --zap 1 7 11 15 18
Scheduled OSD(s) for removal
```

13. 削除する OSD のステータスを確認します。次のコマンドで出力が返されなくなるまで続行しないでください。

```
[ceph: root@dcn2-computehci2-1 /]# ceph orch osd rm status
OSD_ID  HOST                STATE     PG_COUNT  REPLACE  FORCE
DRAIN_STARTED_AT
1       dcn2-computehci2-1     draining  27       False    False  2021-04-23 21:35:51.215361
7       dcn2-computehci2-1     draining  8        False    False  2021-04-23 21:35:49.111500
11      dcn2-computehci2-1     draining  14       False    False  2021-04-23 21:35:50.243762
```

14. 削除するホストにデーモンが残っていないことを確認します。

```
[ceph: root@dcn2-computehci2-1 /]# ceph orch ps dcn2-computehci2-1
```

デーモンがまだ存在する場合は、次のコマンドで削除できます。

```
[ceph: root@dcn2-computehci2-1 /]# ceph orch host drain dcn2-computehci2-1
```

15. Red Hat Ceph Storage クラスターから <dcn-computehci-node> ホストを削除します。

```
[ceph: root@dcn2-computehci2-1 /]# ceph orch host rm dcn2-computehci2-1
Removed host 'dcn2-computehci2-1'
```

9.2. IMAGE サービス (GLANCE) サービスの削除

サービスからノードを削除した時に、Image サービスを削除します。

手順

- Image サービスを無効にするには、削除するノードで **systemctl** を使用して無効にします。

```
[root@dcn2-computehci2-1 ~]# systemctl stop tripleo_glance_api.service
[root@dcn2-computehci2-1 ~]# systemctl stop  tripleo_glance_api_tls_proxy.service

[root@dcn2-computehci2-1 ~]# systemctl disable tripleo_glance_api.service
```

```
Removed /etc/systemd/system/multi-user.target.wants/tripleo_glance_api.service.
[root@dcn2-computehci2-1 ~]# systemctl disable  tripleo_glance_api_tls_proxy.service
Removed /etc/systemd/system/multi-user.target.wants/tripleo_glance_api_tls_proxy.service.
```

9.3. BLOCK STORAGE (CINDER) サービスの削除

DistributedComputeHCI ノードをサービスから削除する際に、**cinder-volume** サービスと **etcd** サービスを DistributedComputeHCI ノード から削除する必要があります。

手順

1. 削除するノードの **cinder-volume** サービスを特定して無効にします。

```
(central) [stack@site-undercloud-0 ~]$ openstack volume service list --service cinder-volume
| cinder-volume | dcn2-computehci2-1@tripleo_ceph | az-dcn2    | enabled | up    | 2022-03-
23T17:41:43.000000 |
(central) [stack@site-undercloud-0 ~]$ openstack volume service set --disable dcn2-
computehci2-1@tripleo_ceph cinder-volume
```

2. スタック内の別の DistributedComputeHCI ノードにログインします。

```
$ ssh tripleo-admin@dcn2-computehci2-0
```

3. 削除するノードに関連付けられた **cinder-volume** サービスを削除します。

```
[root@dcn2-computehci2-0 ~]# podman exec -it cinder_volume cinder-manage service
remove cinder-volume dcn2-computehci2-1@tripleo_ceph
Service cinder-volume on host dcn2-computehci2-1@tripleo_ceph removed.
```

4. 削除するノードで **tripleo_cinder_volume** サービスを停止して無効にします。

```
[root@dcn2-computehci2-1 ~]# systemctl stop tripleo_cinder_volume.service
[root@dcn2-computehci2-1 ~]# systemctl disable tripleo_cinder_volume.service
Removed /etc/systemd/system/multi-user.target.wants/tripleo_cinder_volume.service
```

9.4. DISTRIBUTEDCOMPUTEHCI ノードの削除

provisioned パラメーターを **false** の値に設定し、ノードをスタックから削除します。**nova-compute** サービスを無効にして、関連するネットワークエージェントを削除します。

手順

1. **baremetal-deployment.yaml** ファイルをコピーします。

```
cp /home/stack/dcn2/overcloud-baremetal-deploy.yaml \
/home/stack/dcn2/baremetal-deployment-scaledown.yaml
```

2. **baremetal-deployement-scaledown.yaml** ファイルを編集します。削除するホストを特定し、**provisioned** パラメーターの値を **false** に設定します。

```
instances:
...
```

```
  - hostname: dcn2-computehci2-1
    provisioned: false
```

3. スタックからノードを削除します。

```
openstack overcloud node delete --stack dcn2 --baremetal-deployment
/home/stack/dcn2/baremetal_deployment_scaledown.yaml
```

4. オプション: ノードを再利用する場合は、ironic を使用してディスクをクリーンアップします。これは、ノードが Ceph OSD をホストする場合に必要です。

```
openstack baremetal node manage $UUID
openstack baremetal node clean $UUID --clean-steps '[{"interface":"deploy", "step":
"erase_devices_metadata"}]'
openstack baremetal provide $UUID
```

5. 中央サイトを再デプロイします。初期設定に使用するすべてのテンプレートを含めます。

```
openstack overcloud deploy \
--deployed-server \
--stack central \
--templates /usr/share/openstack-tripleo-heat-templates/ \
-r ~/control-plane/central_roles.yaml \
-n ~/network-data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/dcn-storage.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/cephadm.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
-e /home/stack/central/overcloud-networks-deployed.yaml \
-e /home/stack/central/overcloud-vip-deployed.yaml \
-e /home/stack/central/deployed_metal.yaml \
-e /home/stack/central/deployed_ceph.yaml \
-e /home/stack/central/dcn_ceph.yaml \
-e /home/stack/central/glance_update.yaml
```

9.5. 削除された DISTRIBUTEDCOMPUTEHCI ノードの置き換え

9.5.1. 削除された DistributedComputeHCI ノードの置き換え

DCN デプロイメントに新しい HCI ノードを追加するには、追加のノードでエッジスタックを再デプロイし、そのスタックの **ceph export** を実行してから、中央ロケーションのスタック更新を実行する必要があります。中央ロケーションのスタック更新により、エッジサイトに固有の設定が追加されます。

前提条件

ノードの置き換えまたは新しいノードの追加先となるスタックの nodes_data.yaml ファイルで、ノード数が正常である。

手順

1. デプロイスクリプトから呼び出されるテンプレートの 1 つで、**EtcdIntialClusterState** パラメーターを **existing** に設定する必要があります。

```
parameter_defaults:
  EtcdInitialClusterState: existing
```

2. スタックに固有のデプロイメントスクリプトを使用して再デプロイします。

```
(undercloud) [stack@site-undercloud-0 ~]$ ./overcloud_deploy_dcn2.sh
…
Overcloud Deployed without error
```

3. スタックから Red Hat Ceph Storage データをエクスポートします。

```
(undercloud) [stack@site-undercloud-0 ~]$ sudo -E openstack overcloud export ceph --stack
dcn1,dcn2 --config-download-dir /var/lib/mistral --output-file
~/central/dcn2_scale_up_ceph_external.yaml
```

4. 中央ロケーションのデプロイスクリプトで、dcn_ceph_external.yaml を新しく生成された dcn2_scale_up_ceph_external.yaml と置き換えます。

5. 中央でスタックの更新を実行します。

```
(undercloud) [stack@site-undercloud-0 ~]$ ./overcloud_deploy.sh
...
Overcloud Deployed without error
```

9.6. 置き換えられた DISTRIBUTEDCOMPUTEHCI ノードの機能の検証

1. **status** フィールドの値が **enabled** で、**State** フィールドの値が **up** であることを確認します。

```
(central) [stack@site-undercloud-0 ~]$ openstack compute service list -c Binary -c Host -c
Zone -c Status -c State
+----------------+----------------------------------------+------------+---------+-------+
| Binary         | Host                                   | Zone       | Status  | State |
+----------------+----------------------------------------+------------+---------+-------+
...
| nova-compute   | dcn1-compute1-0.redhat.local           | az-dcn1    | enabled | up    |
| nova-compute   | dcn1-compute1-1.redhat.local           | az-dcn1    | enabled | up    |
| nova-compute   | dcn2-computehciscaleout2-0.redhat.local | az-dcn2    | enabled | up    |
| nova-compute   | dcn2-computehci2-0.redhat.local        | az-dcn2    | enabled | up    |
| nova-compute   | dcn2-computescaleout2-0.redhat.local   | az-dcn2    | enabled | up    |
| nova-compute   | dcn2-computehci2-2.redhat.local        | az-dcn2    | enabled | up    |
...
```

2. すべてのネットワークエージェントが **up** 状態にあることを確認します。

```
(central) [stack@site-undercloud-0 ~]$ openstack network agent list -c "Agent Type" -c Host -
c Alive -c State
+--------------------+----------------------------------------+-------+-------+
| Agent Type         | Host                                   | Alive | State |
+--------------------+----------------------------------------+-------+-------+
| DHCP agent         | dcn3-compute3-1.redhat.local           | :-)   | UP    |
| Open vSwitch agent | central-computehci0-1.redhat.local     | :-)   | UP    |
| DHCP agent         | dcn3-compute3-0.redhat.local           | :-)   | UP    |
| DHCP agent         | central-controller0-2.redhat.local     | :-)   | UP    |
```

```
| Open vSwitch agent | dcn3-compute3-1.redhat.local           | :-)   | UP    |
| Open vSwitch agent | dcn1-compute1-1.redhat.local           | :-)   | UP    |
| Open vSwitch agent | central-computehci0-0.redhat.local     | :-)   | UP    |
| DHCP agent         | central-controller0-1.redhat.local     | :-)   | UP    |
| L3 agent           | central-controller0-2.redhat.local     | :-)   | UP    |
| Metadata agent     | central-controller0-1.redhat.local     | :-)   | UP    |
| Open vSwitch agent | dcn2-computescaleout2-0.redhat.local   | :-)   | UP    |
| Open vSwitch agent | dcn2-computehci2-5.redhat.local        | :-)   | UP    |
| Open vSwitch agent | central-computehci0-2.redhat.local     | :-)   | UP    |
| DHCP agent         | central-controller0-0.redhat.local     | :-)   | UP    |
| Open vSwitch agent | central-controller0-1.redhat.local     | :-)   | UP    |
| Open vSwitch agent | dcn2-computehci2-0.redhat.local        | :-)   | UP    |
| Open vSwitch agent | dcn1-compute1-0.redhat.local           | :-)   | UP    |
...
```

3. Ceph クラスターのステータスを確認します。

   a. SSH を使用して新しい DistributedComputeHCI ノードに接続し、Ceph クラスターのステータスを確認します。

   ```
   [root@dcn2-computehci2-5 ~]# podman exec -it ceph-mon-dcn2-computehci2-5 \
   ceph -s -c /etc/ceph/dcn2.conf
   ```

   b. 新規ノードに ceph mon サービスと ceph mgr サービスの両方が存在することを確認します。

   ```
   services:
      mon: 3 daemons, quorum dcn2-computehci2-2,dcn2-computehci2-0,dcn2-
   computehci2-5 (age 3d)
      mgr: dcn2-computehci2-2(active, since 3d), standbys: dcn2-computehci2-0, dcn2-
   computehci2-5
      osd: 20 osds: 20 up (since 3d), 20 in (since 3d)
   ```

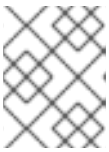
   c. ceph osds のステータスを 'ceph osd tree' で確認します。新規ノードの osds すべてが STATUS up にあることを確認します。

   ```
   [root@dcn2-computehci2-5 ~]# podman exec -it ceph-mon-dcn2-computehci2-5 ceph
   osd tree -c /etc/ceph/dcn2.conf
   ID CLASS WEIGHT  TYPE NAME                     STATUS REWEIGHT PRI-AFF
   -1       0.97595 root default
   -5       0.24399     host dcn2-computehci2-0
    0   hdd 0.04880         osd.0                 up  1.00000 1.00000
    4   hdd 0.04880         osd.4                 up  1.00000 1.00000
    8   hdd 0.04880         osd.8                 up  1.00000 1.00000
   13   hdd 0.04880         osd.13                up  1.00000 1.00000
   17   hdd 0.04880         osd.17                up  1.00000 1.00000
   -9       0.24399     host dcn2-computehci2-2
    3   hdd 0.04880         osd.3                 up  1.00000 1.00000
    5   hdd 0.04880         osd.5                 up  1.00000 1.00000
   10   hdd 0.04880         osd.10                up  1.00000 1.00000
   14   hdd 0.04880         osd.14                up  1.00000 1.00000
   19   hdd 0.04880         osd.19                up  1.00000 1.00000
   -3       0.24399     host dcn2-computehci2-5
    1   hdd 0.04880         osd.1                 up  1.00000 1.00000
    7   hdd 0.04880         osd.7                 up  1.00000 1.00000
   ```

```
 11   hdd 0.04880         osd.11                      up  1.00000 1.00000
 15   hdd 0.04880         osd.15                      up  1.00000 1.00000
 18   hdd 0.04880         osd.18                      up  1.00000 1.00000
 -7       0.24399     host dcn2-computehciscaleout2-0
  2   hdd 0.04880         osd.2                       up  1.00000 1.00000
  6   hdd 0.04880         osd.6                       up  1.00000 1.00000
  9   hdd 0.04880         osd.9                       up  1.00000 1.00000
 12   hdd 0.04880         osd.12                      up  1.00000 1.00000
 16   hdd 0.04880         osd.16                      up  1.00000 1.00000
```

4. 新しい DistributedComputeHCI ノードの **cinder-volume** サービスがステータス 'enabled' にあり、状態が 'up' であることを確認します。

```
(central) [stack@site-undercloud-0 ~]$ openstack volume service list --service cinder-volume
-c Binary -c Host -c Zone -c Status -c State
+--------------+---------------------------------+------------+---------+-------+
| Binary       | Host                            | Zone       | Status  | State |
+--------------+---------------------------------+------------+---------+-------+
| cinder-volume | hostgroup@tripleo_ceph          | az-central | enabled | up    |
| cinder-volume | dcn1-compute1-1@tripleo_ceph    | az-dcn1    | enabled | up    |
| cinder-volume | dcn1-compute1-0@tripleo_ceph    | az-dcn1    | enabled | up    |
| cinder-volume | dcn2-computehci2-0@tripleo_ceph | az-dcn2    | enabled | up    |
| cinder-volume | dcn2-computehci2-2@tripleo_ceph | az-dcn2    | enabled | up    |
| cinder-volume | dcn2-computehci2-5@tripleo_ceph | az-dcn2    | enabled | up    |
+--------------+---------------------------------+------------+---------+-------+
```



注記

cinder-volume サービスの状態が **down** の場合には、サービスはノードで起動していません。

5. ssh を使用して新しい DistributedComputeHCI ノードに接続し、Glance サービスのステータスを 'systemctl' で確認します。

```
[root@dcn2-computehci2-5 ~]# systemctl --type service | grep glance
  tripleo_glance_api.service                    loaded active     running       glance_api container
  tripleo_glance_api_healthcheck.service        loaded activating start   start glance_api
healthcheck
  tripleo_glance_api_tls_proxy.service          loaded active     running
glance_api_tls_proxy container
```

9.7. DISTRIBUTEDCOMPUTEHCI の状態ダウンのトラブルシューティング

代替ノードが EtcdInitialClusterState パラメーターの値を **existing** に設定せずにデプロイされている場合、**openstack volume service list** を実行すると、置き換えられたノードの **cinder-volume** サービスは **down** と表示されます。

手順

1. 置き換えノードにログインし、etcd サービスのログを確認します。**etcd** サービスがクラスター ID の不一致を報告していることが、**/var/log/containers/stdouts/etcd.log** のログファイルに表示されていることを確認します。

```
2022-04-06T18:00:11.834104130+00:00 stderr F 2022-04-06 18:00:11.834045 E | rafthttp:
request cluster ID mismatch (got 654f4cf0e2cfb9fd want 918b459b36fe2c0c)
```

2. **EtcdInitialClusterState** パラメーターをデプロイメントテンプレートの **existing** の値に設定し、デプロイメントスクリプトを再実行します。

3. SSH を使用して代替ノードに接続し、root で以下のコマンドを実行します。

```
[root@dcn2-computehci2-4 ~]# systemctl stop tripleo_etcd
[root@dcn2-computehci2-4 ~]# rm -rf /var/lib/etcd/*
[root@dcn2-computehci2-4 ~]# systemctl start tripleo_etcd
```

4. **/var/log/containers/stdouts/etcd.log** ログファイルを再チェックし、ノードが正常にクラスターに参加していることを確認します。

```
2022-04-06T18:24:22.130059875+00:00 stderr F 2022-04-06 18:24:22.129395 I |
etcdserver/membership: added member 96f61470cd1839e5 [https://dcn2-computehci2-
4.internalapi.redhat.local:2380] to cluster 654f4cf0e2cfb9fd
```

5. cinder-volume サービスの状態を確認し、**openstack volume service list** を実行する際に置き換えノードで **up** を読み取ることを確認します。

第10章 KEY MANAGER を含むデプロイ

Red Hat OpenStack Platform 16.1.2 リリース以前にエッジサイトをデプロイしている場合は、この機能を実装するのに roles.yaml を再生成する必要があります。機能を実装するには、DCN サイトのデプロイメントに使用する **roles.yaml** ファイルを再生成します。

```
$ openstack overcloud roles generate DistributedComputeHCI DistributedComputeHCIScaleOut -o
~/dcn0/roles_data.yaml
```

10.1. KEY MANAGER が設定されたエッジサイトのデプロイ

エッジサイトに Key Manager (barbican) サービスへのアクセスを含める場合、中央サイトに barbican を設定する必要があります。Barbican のインストールおよび設定の詳細は、Barbican のデプロイ を参照してください。

- **/usr/share/openstack-tripleo-heat-templates/environments/services/barbican-edge.yaml** を含めることで、DCN サイトからの barbican へのアクセスを設定することができます。

```
openstack overcloud deploy \
  --stack dcn0 \
  --templates /usr/share/openstack-tripleo-heat-templates/ \
  -r ~/dcn0/roles_data.yaml \
  ....
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/barbican-edge.yaml
```

第11章 GLANCE イメージの NOVA への事前キャッシュ

ローカルの一時ストレージを使用するように OpenStack Compute を設定する場合、インスタンスのデプロイメントを迅速化するために glance イメージがキャッシュされます。インスタンスに必要なイメージがまだキャッシュされていない場合は、インスタンスの作成時に Compute ノードのローカルディスクにダウンロードされます。

glance イメージのダウンロードプロセスに要する時間は、イメージのサイズおよび帯域幅やレイテンシー等のネットワーク特性によって変動します。

インスタンスの起動を試みる際にローカルの Ceph クラスターでイメージが利用できない場合は、以下のメッセージと共にインスタンスの起動に失敗します。

> Build of instance 3c04e982-c1d1-4364-b6bd-f876e399325b aborted: Image 20c5ff9d-5f54-4b74-830f-88e78b9999ed is unacceptable: No image locations are accessible

Compute サービスのログには以下のメッセージが記録されます。

> 'Image %s is not on my ceph and [workarounds]/ never_download_image_if_on_rbd=True; refusing to fetch and upload.',

インスタンスの起動に失敗する原因は、**nova.conf** 設定ファイルの **never_download_image_if_on_rbd** パラメーターです。DCN デプロイメントの場合、このパラメーターはデフォルトでは **true** に設定されています。**dcn-storage.yaml** ファイルの heat パラメーター **NovaDisableImageDownloadToRbd** を使用して、この値を制御することができます。

オーバークラウドのデプロイ前に **NovaDisableImageDownloadToRbd** の値を **false** に設定した場合の動作は、以下のようになります。

- イメージがローカルで利用できない場合、Compute サービス (nova) は **central** サイトで利用可能なイメージを自動的にストリーミングします。
- glance イメージからの COW コピーは使用されません。
- イメージを使用するインスタンスの数により、Compute (nova) ストレージに同じイメージのコピーが複数含まれる場合があります。
- **central** サイトへの WAN リンクと nova ストレージプールの両方が飽和状態になる可能性があります。

Red Hat では、この値を true に設定したままにし、インスタンスの起動前に必要なイメージがローカルで利用できるようにすることを推奨します。イメージをエッジサイトで利用できるようにする方法については、「新規サイトへのイメージのコピー」を参照してください。

ローカルにあるイメージに関して、**tripleo_nova_image_cache.yml** Ansible Playbook を使用して、共通的に使用されるイメージや今後デプロイする可能性の高いイメージを事前キャッシュして、仮想マシンの作成を迅速化することができます。

11.1. TRIPLEO_NOVA_IMAGE_CACHE.YML ANSIBLE PLAYBOOK の実行

前提条件

- シェル環境での正しい API への認証用クレデンシャル

各手順で提供されるコマンドの前に、正しい認証ファイルが読み込まれている必要があります。

手順

1. オーバークラウドスタックの Ansible インベントリーディレクトリーを作成します。

```
$ mkdir inventories

$ find ~/overcloud-deploy/*/config-download \
  -name tripleo-ansible-inventory.yaml |\
  while read f; do cp $f inventories/$(basename $(dirname $f)).yaml; done
```

2. 事前キャッシュするイメージの ID リストを作成します。

   a. 利用可能なイメージの完全なリストを取得します。

```
$ source centralrc

$ openstack image list
+--------------------------------------+---------+--------+
| ID                                   | Name    | Status |
+--------------------------------------+---------+--------+
| 07bc2424-753b-4f65-9da5-5a99d8383fe6 | image_0 | active |
| d5187afa-c821-4f22-aa4b-4e76382bef86 | image_1 | active |
+--------------------------------------+---------+--------+
```

   b. **nova_cache_args.yml** という名前で Ansible Playbook の引数ファイルを作成し、事前キャッシュするイメージの ID を追加します。

```
---
tripleo_nova_image_cache_images:
  - id: 07bc2424-753b-4f65-9da5-5a99d8383fe6
  - id: d5187afa-c821-4f22-aa4b-4e76382bef86
```

3. **tripleo_nova_image_cache.yml** Ansible Playbook を実行します。

```
$ source centralrc

$ ansible-playbook -i inventories \
--extra-vars "@nova_cache_args.yml" \
/usr/share/ansible/tripleo-playbooks/tripleo_nova_image_cache.yml
```

11.2. パフォーマンスに関する考慮事項

Ansible の **forks** パラメーターを使用して、同時にダウンロードするイメージの数を指定することができます。このパラメーターのデフォルト値は **5** です。**forks** パラメーターの値を増やして、このイメージの配布に要する時間を短縮することができます。ただし、ネットワークおよび glance-api の負荷が増えることとバランスを取る必要があります。

以下のように、**--forks** パラメーターを使用して同時実行を調整します。

```
ansible-playbook -i inventory.yaml \
--forks 10 \
--extra-vars "@nova_cache_args.yml" \
```

```
/usr/share/ansible/tripleo-playbooks/tripleo_nova_image_cache.yml
```

11.3. DCN サイトへのイメージ配布の最適化

glance イメージの配布にプロキシーを使用して、WAN トラフィックを軽減することができます。プロキシーを設定した場合の動作は、以下のようになります。

- glance イメージは、プロキシーとして機能する 1 台の Compute ノードにダウンロードされます。
- プロキシーは、glance イメージをインベントリー内の他の Compute ノードに再配布します。

Ansible の引数ファイル **nova_cache_args.yml** に以下のパラメーターを追加して、プロキシーノードを設定することができます。

tripleo_nova_image_cache_use_proxy パラメーターを **true** に設定して、イメージキャッシュプロキシーを有効にします。

イメージプロキシーは、セキュアなコピー **scp** を使用して、イメージをインベントリー内の他のノードに配布します。DCN サイト間の WAN 等レイテンシーが高いネットワークを通じて配布する場合は、SCP は効率的ではありません。Red Hat では、Playbook のターゲットを 1 つの DCN サイト (1 つのスタックに対応) に制限することを推奨します。

tripleo_nova_image_cache_proxy_hostname パラメーターを使用して、イメージキャッシュプロキシーを選択します。デフォルトのプロキシーは、Ansible インベントリーファイルで最初に定義されているコンピュートノードです。**tripleo_nova_image_cache_plan** パラメーターを使用して、Playbook のインベントリーを 1 つのサイトに制限します。

```
tripleo_nova_image_cache_use_proxy: true
tripleo_nova_image_cache_proxy_hostname: dcn0-novacompute-1
tripleo_nova_image_cache_plan: dcn0
```

11.4. NOVA-CACHE クリーンアップの設定

バックグラウンドプロセスが定期的に実行され、以下の両方の条件を満たすイメージが nova キャッシュから削除されます。

- イメージがインスタンスによって使用されていない。
- イメージの経過時間が nova パラメーター **remove_unused_original_minimum_age_seconds** の設定値を超えている。

remove_unused_original_minimum_age_seconds パラメーターのデフォルト値は **86400** です。値は秒単位で表され、これは 24 時間に相当します。初回デプロイメント時またはクラウドのスタック更新時に、tripleo-heat-templates パラメーター **NovaImageCachTTL** を使用してこの値を制御することができます。

```
parameter_defaults:
  NovaImageCacheTTL: 604800 # Default to 7 days for all compute roles
  Compute2Parameters:
    NovaImageCacheTTL: 1209600 # Override to 14 days for the Compute2 compute role
```

Playbook により Compute ノードにすでに存在するイメージを事前キャッシュすると、Ansible は変更を報告せず、イメージの経過時間は 0 にリセットされます。**NovaImageCacheTTL** パラメーターの値よりも頻繁に Ansible のプレイを実行し、イメージのキャッシュを維持します。

第12章 DCN への TLS-E の適用

分散コンピュートノードインフラストラクチャー用に設計されたクラウドで、TLS (Transport Layer Security) を有効にすることができます。パブリックアクセスだけに TLS を有効にするか、TLS-e としてすべてのネットワークで TLS を有効にすることができます。後者の場合、すべての内部および外部データフローで暗号化を行うことができます。

エッジサイトにはパブリックエンドポイントがないため、エッジスタックでパブリックアクセスを有効にすることはできません。パブリックアクセスの TLS の詳細は、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 を参照してください。

12.1. TLS-E を設定した分散コンピュートノードアーキテクチャーのデプロイ

前提条件

Red Hat Identity Manager (IdM) と共に Red Hat OpenStack Platform (RHOSP) 分散コンピュートノードアーキテクチャーで TLS-e を設定する場合には、Red Hat Identity Manager 用にデプロイされた Red Hat Enterprise Linux のバージョンに基づいて、以下のアクションを実行します。

Red Hat Enterprise Linux 8.4

1. Red Hat Identity Management ノードで、**ipa-ext.conf** ファイルの ACL に対して信頼されるサブネットを許可します。

```
acl "trusted_network" {
  localnets;
  localhost;
  192.168.24.0/24;
  192.168.25.0/24;
};
```

1. **/etc/named/ipa-options-ext.conf** ファイルで、再帰とクエリーキャッシュを許可します。

```
allow-recursion { trusted_network; };
allow-query-cache { trusted_network; };
```

2. named-pkcs11 サービスを再起動します。

```
systemctl restart named-pkcs11
```

Red Hat Enterprise Linux 8.2

Red Hat Enterprise Linux (RHEL) 8.2 に Red Hat Identity Manager (IdM) がある場合は、Red Hat Enterprise Linux をアップグレードしてから、RHEL 8.4 の指示に従ってください。

Red Hat Enterprise Linux 7.x

Red Hat Enterprise Linux (RHEL) 7.x に Red Hat Identity Manager (IdM) がある場合は、ドメイン名のアクセス制御手順 (ACI) を手動で追加する必要があります。たとえば、ドメイン名が **redhat.local** の場合は、Red Hat Identity Manager で以下のコマンドを実行して ACI を設定します。

```
ADMIN_PASSWORD=redhat_01
DOMAIN_LEVEL_1=local
DOMAIN_LEVEL_2=redhat
```

```
cat << EOF | ldapmodify -x -D "cn=Directory Manager" -w ${ADMIN_PASSWORD}
dn: cn=dns,dc=${DOMAIN_LEVEL_2},dc=${DOMAIN_LEVEL_1}
changetype: modify
add: aci
aci: (targetattr = "aaaarecord || arecord || cnamerecord || idnsname || objectclass || ptrrecord")
(targetfilter = "(&(objectclass=idnsrecord)(|(aaaarecord=)(arecord=)(cnamerecord=)(ptrrecord=)
(idnsZoneActive=TRUE)))")(version 3.0; acl "Allow hosts to read DNS A/AAA/CNAME/PTR records";
allow (read,search,compare) userdn =
"ldap:///fqdn=*,cn=computers,cn=accounts,dc=${DOMAIN_LEVEL_2},dc=${DOMAIN_LEVEL_1}";)
EOF
```

中央サイトとエッジロケーションの設計の違いのため、エッジスタックには以下のファイルを含めないでください。

tls-everywhere-endpoints-dns.yaml

エッジサイトではこのファイルは無視され、このファイルで設定されるエンドポイントは中央スタックからエクスポートされるエンドポイントによってオーバーライドされます。

haproxy-public-tls-certmonger.yaml

エッジサイトにはパブリックエンドポイントがないため、このファイルはデプロイメント失敗の原因になります。

関連情報

- Ansible を使用した TLS-e の実装

第13章 外部アクセス用 CEPH キーの作成

警告

この機能のコンテンツは、このリリースで **ドキュメンテーションプレビュー** として提供されているため、Red Hat によって完全に検証されているわけではありません。テスト用にのみ使用し、実稼働環境では使用しないでください。

Ceph ストレージへの外部アクセスとは、ローカルではない任意のサイトからの Ceph へのアクセスを指します。中央サイトにとってはエッジサイトの Ceph ストレージが外部にあたるのとまったく同じように、中央サイトの Ceph ストレージは、エッジ (DCN) サイトからは外部に該当します。

Ceph ストレージと共に中央サイトまたは DCN サイトをデプロイする場合、ローカルアクセスと外部アクセスの両方にデフォルトの **openstack** キーリングを使用するオプションが可能です。あるいは、ローカル以外のサイトからのアクセス用に別の鍵を作成することができます。

外部サイトへのアクセスに追加の Ceph キーを使用する場合は、それぞれのキーの名前を同じにする必要があります。以降の例では、鍵の名前を **external** としています。

ローカル以外のサイトからのアクセスに別の鍵を使用すると、セキュリティーが向上します。ローカルアクセスを中断すること無く、セキュリティーイベントに対応して外部アクセス用の鍵を無効にして再発行することができます。ただし、外部アクセスに別の鍵を使用すると、アベイラビリティーゾーンをまたがるバックアップやオフラインボリューム移行など、一部の機能を利用することができなくなります。セキュリティー対応のニーズと必要な機能セットの間でバランスを取る必要があります。

デフォルトでは、中央サイトおよびすべての DCN サイトの鍵は共有されます。

13.1. 外部アクセス用 CEPH キーの作成

ローカル以外のサイトからのアクセス用に **external** 鍵を作成するには、以下の手順を実施します。

Process

1. 外部アクセス用の Ceph キーを作成します。この鍵の取り扱いには注意が必要です。以下のコマンドを使用して鍵を生成することができます。

```
python3 -c 'import os,struct,time,base64; key = os.urandom(16) ; \
header = struct.pack("<hiih", 1, int(time.time()), 0, len(key)) ; \
print(base64.b64encode(header + key).decode())'
```

2. デプロイするスタックのディレクトリーにおいて、以下のような内容で **ceph_keys.yaml** 環境ファイルを作成します。鍵には、前のステップのコマンド出力を使用します。

```
parameter_defaults:
  CephExtraKeys:
    - name: "client.external"
      caps:
        mgr: "allow *"
        mon: "profile rbd"
```

```
    osd: "profile rbd pool=vms, profile rbd pool=volumes, profile rbd pool=images"
  key: "AQD29WteAAAAABAAphgOjFD7nyjdYe8Lz0mQ5Q=="
  mode: "0600"
```

3. サイトのデプロイメントに **ceph_keys.yaml** 環境ファイルを追加します。たとえば、**ceph_keys.yaml** 環境ファイルを指定して中央サイトをデプロイするには、以下のようなコマンドを実行します。

```
overcloud deploy \
      --stack central \
      --templates /usr/share/openstack-tripleo-heat-templates/ \
      ....
      -e ~/central/ceph_keys.yaml
```

13.2. 外部 CEPH キーの使用

すでにデプロイされている鍵だけを使用することができます。**external** 鍵と共にサイトをデプロイする際の詳細は、「外部アクセス用 Ceph キーの作成」を参照してください。この手順は、中央サイトとエッジサイトの両方で実施する必要があります。

- 中央サイトの提供する **external** 鍵を使用するエッジサイトをデプロイする場合は、以下の手順を実施します。

  1. エッジサイト用の **dcn_ceph_external.yaml** 環境ファイルを作成します。**cephx-key-client-name** オプションを追加して、含めるべきデプロイした鍵を指定する必要があります。

```
sudo -E openstack overcloud export ceph \
--stack central \
--cephx-key-client-name external \
--output-file ~/dcn-common/dcn_ceph_external.yaml
```

  2. エッジサイトが中央サイトの Ceph クラスターにアクセスできるように、**dcn_ceph_external.yaml** ファイルを追加します。**ceph_keys.yaml** ファイルを追加して、エッジサイトの Ceph クラスター用外部鍵をデプロイします。

- エッジサイトのデプロイ後に中央サイトを更新する場合は、中央サイトが dcn **external** 鍵を使用するようにします。

  1. **CephClientUserName** パラメーターがエクスポートされる鍵と一致するようにします。以降の例に示すように、**external** の名前を使用している場合は、以下のような内容で **glance_update.yaml** を作成します。

```
parameter_defaults:
  GlanceEnabledImportMethods: web-download,copy-image
  GlanceBackend: rbd
  GlanceStoreDescription: 'central rbd glance store'
  CephClusterName: central
  GlanceBackendID: central
  GlanceMultistoreConfig:
  dcn0:
    GlanceBackend: rbd
   GlanceStoreDescription: 'dcn0 rbd glance store'
   CephClientUserName: 'external'
   CephClusterName: dcn0
```

```
    GlanceBackendID: dcn0
  dcn1:
    GlanceBackend: rbd
    GlanceStoreDescription: 'dcn1 rbd glance store'
    CephClientUserName: 'external'
    CephClusterName: dcn1
    GlanceBackendID: dcn1
```

2. **openstack overcloud export ceph** コマンドを使用する際に、中央サイトから DCN エッジサイトへのアクセス用に **external** 鍵を追加します。そのためには、**--stack** 引数にスタックのコンマ区切りリストを指定し、**cephx-key-client-name** オプションを指定する必要があります。

```
sudo -E openstack overcloud export ceph \
--stack dcn0,dcn1,dcn2 \
--cephx-key-client-name external \
--output-file ~/central/dcn_ceph_external.yaml
```

3. 元のテンプレートを使用して中央サイトを再デプロイする際に、新たに作成した **dcn_ceph_external.yaml** および **glance_update.yaml** ファイルを追加します。

```
openstack overcloud deploy \
      --stack central \
      --templates /usr/share/openstack-tripleo-heat-templates/ \
      -r ~/central/central_roles.yaml \
    ...
      -e /usr/share/openstack-tripleo-heat-
templates/environments/cephadm/cephadm.yaml \
      -e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
      -e ~/central/central-images-env.yaml \
      -e ~/central/role-counts.yaml \
      -e ~/central/site-name.yaml
      -e ~/central/ceph.yaml \
      -e ~/central/ceph_keys.yaml \
      -e ~/central/glance.yaml \
      -e ~/central/dcn_ceph_external.yaml
```

付録A デプロイメントの移行オプション

本項では、DCN ストレージの検証と、アーキテクチャーの移行または変更に関するトピックについて説明します。

A.1. エッジストレージの検証

中央サイトおよびエッジサイトのデプロイメントが機能していることを確認するには、glance マルチストアおよびインスタンスの作成をテストします。

ローカルのファイルシステム上または Web サーバーで利用可能なイメージを glance にインポートすることができます。

注記

中央サイトにイメージを使用するインスタンスがない場合でも、必ず中央サイトにイメージのコピーを保存してください。

前提条件

1. **glance stores-info** コマンドを使用して、Image サービスを通じて利用可能なストアを確認します。以下の例では、central、dcn1、および dcn2 の 3 つのストアが利用可能です。これらは、それぞれ中央サイトおよびエッジサイトの glance ストアに対応します。

```
$ glance stores-info
+----------+----------------------------------------------------------------------------------+
| Property | Value                                                                            |
+----------+----------------------------------------------------------------------------------+
| stores   | [{"default": "true", "id": "central", "description": "central rbd glance         |
|          | store"}, {"id": "dcn0", "description": "dcn0 rbd glance store"},                  |
|          | {"id": "dcn1", "description": "dcn1 rbd glance store"}]                           |
+----------+----------------------------------------------------------------------------------+
```

A.1.1. ローカルファイルからのインポート

まず中央サイトのストアにイメージをアップロードし、続いてそれをリモートサイトにコピーする必要があります。

1. イメージのファイルが RAW 形式であることを確認します。イメージが raw 形式でなければ、イメージを Image サービスにインポートする前に変換する必要があります。

```
file cirros-0.5.1-x86_64-disk.img
cirros-0.5.1-x86_64-disk.img: QEMU QCOW2 Image (v3), 117440512 bytes

qemu-img convert -f qcow2 -O raw cirros-0.5.1-x86_64-disk.img cirros-0.5.1-x86_64-
disk.raw
```

```
Import the image into the default back end at the central site:
```

```
glance image-create \
--disk-format raw --container-format bare \
--name cirros --file cirros-0.5.1-x86_64-disk.raw \
--store central
```

-

A.1.2. Web サーバーからのイメージのインポート

イメージが Web サーバーでホストされている場合は、**GlanceImageImportPlugins** パラメーターを使用して複数のストアにイメージをアップロードすることができます。

この手順では、デフォルトのイメージ変換プラグインが glance で有効になっていることを前提としています。この機能により、QCOW2 ファイル形式が Ceph RBD に最適な RAW イメージに自動的に変換されます。**glance image-show ID | grep disk_format** を実行して、glance イメージが RAW 形式であることを確認することができます。

手順

1. **glance** コマンドの **image-create-via-import** パラメーターを使用して、Web サーバーからイメージをインポートします。**--stores** パラメーターを使用します。

```
# glance image-create-via-import \
--disk-format qcow2 \
--container-format bare \
--name cirros \
--uri http://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-disk.img \
--import-method web-download \
--stores central,dcn1
```

この例では、qcow2 cirros イメージが公式の Cirros サイトからダウンロードされ、glance により RAW に変換され、**--stores** パラメーターで指定した中央サイトおよびエッジサイト 1 にインポートされます。

あるいは、**--stores** を **--all-stores True** に置き換えて、すべてのストアにイメージをアップロードすることができます。

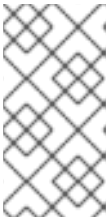
A.1.3. 新規サイトへのイメージのコピー

既存のイメージを中央サイトからエッジサイトにコピーすることができます。これにより、新たに構築されたサイトにおいて、以前作成したイメージにアクセスすることができます。

1. コピーの操作には、glance イメージの UUID を使用します。

```
ID=$(openstack image show cirros -c id -f value)

glance image-import $ID --stores dcn0,dcn1 --import-method copy-image
```



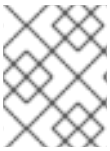
注記

この例では、**--stores** オプションにより、**cirros** イメージを中央サイトからエッジサイト dcn1 および dcn2 にコピーすることを指定しています。あるいは、**--all-stores True** オプションを使用して、現在イメージがアップロードされていないすべてのストアにイメージをアップロードすることができます。

2. イメージが各ストアにコピーされていることを確認します。**stores** キー (属性マッピングの最後の項目) が **central,dcn0,dcn1** に設定されている点に注意してください。

```
$ openstack image show $ID | grep properties
| properties      | direct_url=rbd://d25504ce-459f-432d-b6fa-
```

79854d786f2b/images/8083c7e7-32d8-4f7a-b1da-0ed7884f1076/snap, locations=**[{u'url**: u'rbd://d25504ce-459f-432d-b6fa-79854d786f2b/images/8083c7e7-32d8-4f7a-b1da-0ed7884f1076/snap', u'metadata': {u'store': u'central'}}, {u'url': u'rbd://0c10d6b5-a455-4c4d-bd53-8f2b9357c3c7/images/8083c7e7-32d8-4f7a-b1da-0ed7884f1076/snap', u'metadata': {u'store': u'dcn0'}}, {u'url': u'rbd://8649d6c3-dcb3-4aae-8c19-8c2fe5a853ac/images/8083c7e7-32d8-4f7a-b1da-0ed7884f1076/snap', u'metadata': {u'store': u'dcn1'}}]**, os_glance_failed_import=**', os_glance_importing_to_stores=**', os_hash_algo='sha512**, os_hash_value=**b795f047a1b10ba0b7c95b43b2a481a59289dc4cf2e49845e60b194a911819d3ada03767bbba4143b44c93fd7f66c96c5a621e28dff51d1196dae64974ce240e**, os_hidden=**False**, stores=**central,dcn0,dcn1** |



注記

中央サイトにイメージを使用する仮想マシンがない場合でも、必ず中央サイトにイメージのコピーを保存してください。

A.1.4. エッジサイトのインスタンスがイメージベースのボリュームからブートできることの確認

エッジサイトでイメージを使用して、永続ルートボリュームを作成することができます。

手順

1. ボリュームとして作成するイメージの ID を把握し、その ID を **openstack volume create** コマンドに渡します。

```
IMG_ID=$(openstack image show cirros -c id -f value)
openstack volume create --size 8 --availability-zone dcn0 pet-volume-dcn0 --image $IMG_ID
```

2. 新たに作成したボリュームのボリューム ID を把握し、それを **openstack server create** コマンドに渡します。

```
VOL_ID=$(openstack volume show -f value -c id pet-volume-dcn0)
openstack server create --flavor tiny --key-name dcn0-key --network dcn0-network --security-group basic --availability-zone dcn0 --volume $VOL_ID pet-server-dcn0
```

3. dcn0 エッジサイトの ceph-mon コンテナー内で rbd コマンドを実行してボリュームプールのリストを表示し、ボリュームがイメージベースであることを確認できます。

```
$ sudo podman exec ceph-mon-$HOSTNAME rbd --cluster dcn0 -p volumes ls -l
NAME                                       SIZE  PARENT                                     FMT PROT LOCK
volume-28c6fc32-047b-4306-ad2d-de2be02716b7 8 GiB images/8083c7e7-32d8-4f7a-b1da-0ed7884f1076@snap   2      excl
$
```

4. インスタンスのルートボリュームの cinder スナップショットを作成できることを確認します。クリーンなスナップショットを作成するために、サーバーを停止してデータが休止状態になるようにします。インスタンスが停止している場合ボリュームのステータスは **in-use** のままなので、**--force** オプションを使用します。

```
openstack server stop pet-server-dcn0
openstack volume snapshot create pet-volume-dcn0-snap --volume $VOL_ID --force
openstack server start pet-server-dcn0
```

5. dcn0 Ceph クラスターのボリュームプールの内容をリスト表示し、新たに作成したスナップショットを表示します。

```
$ sudo podman exec ceph-mon-$HOSTNAME rbd --cluster dcn0 -p volumes ls -l
NAME                                                                SIZE  PARENT
FMT PROT LOCK
volume-28c6fc32-047b-4306-ad2d-de2be02716b7                                  8 GiB
images/8083c7e7-32d8-4f7a-b1da-0ed7884f1076@snap   2      excl
volume-28c6fc32-047b-4306-ad2d-de2be02716b7@snapshot-a1ca8602-6819-45b4-a228-
b4cd3e5adf60 8 GiB images/8083c7e7-32d8-4f7a-b1da-0ed7884f1076@snap   2 yes
```

A.1.5. イメージのスナップショットを作成しサイト間でコピーできることの確認

1. dcn0 サイトで新規イメージを作成できることを確認します。クリーンなスナップショットを作成するために、サーバーを停止してデータが休止状態になるようにします。

```
NOVA_ID=$(openstack server show pet-server-dcn0 -f value -c id)
openstack server stop $NOVA_ID
openstack server image create --name cirros-snapshot $NOVA_ID
openstack server start $NOVA_ID
```

2. **dcn0** エッジサイトから glance のデフォルトバックエンドであるハブサイトに、イメージをコピーして戻します。

```
IMAGE_ID=$(openstack image show cirros-snapshot -f value -c id)
glance image-import $IMAGE_ID --stores central --import-method copy-image
```

glance マルチストア操作の詳細は、複数のストアに対応した Image サービス を参照してください。

A.2. スパイン/リーフ型デプロイメントへの移行

既存ネットワーク設定の既存クラウドを、スパイン/リーフ型アーキテクチャーのクラウドに移行することができます。そのためには、以下の条件が満たされている必要があります。

- すべてのベアメタルポートで、**physical-network** 属性の値が **ctlplane** に設定されている。
- undercloud.conf に追加されたパラメーター **enable_routed_networks** が **true** に設定され、続いてアンダークラウドのインストールコマンド **openstack undercloud install** が再実行される。

アンダークラウドが再デプロイされると、オーバークラウドは 1 つのリーフ **leaf0** が設定されたスパイン/リーフとみなされます。以下の手順で、さらにプロビジョニングリーフをデプロイメントに追加することができます。

1. アンダークラウドでのルーティング対応スパイン/リーフの設定 に示すように、必要なサブネットを undercloud.conf に追加します。
2. アンダークラウドのインストールコマンド **openstack undercloud install** を再実行します。
3. 必要なネットワークおよびロールを、それぞれオーバークラウドのテンプレート **network_data.yaml** および **roles_data.yaml** にさらに追加します。

注記

ネットワーク設定ファイルで **{{network.name}}InterfaceRoutes** パラメーターを使用している場合は、**NetworkDeploymentActions** パラメーターに **UPDATE** の値が含まれるようにする必要があります。

```
NetworkDeploymentActions: ['CREATE','UPDATE'])
```

4. 最後に、クラウドデプロイメントに該当するすべての heat テンプレートが含まれるオーバークラウドのインストールスクリプトを再実行します。

A.3. マルチスタックデプロイメントへの移行

既存のデプロイメントを中央サイトとして扱い、さらにエッジサイトを追加して、単一スタックのデプロイメントからマルチスタックのデプロイメントに移行することができます。

既存のスタックを分割することはできません。必要に応じて、既存のスタックをスケールダウンしてコンピュートノードを削除することができます。その後、これらのコンピュートノードをエッジサイトに追加することができます。

注記

すべてのコンピュートノードが削除されると、このアクションによりワークロードが中断します。

A.4. エッジサイト間のバックアップおよびリストア

エッジサイトの分散コンピュートノード (DCN) アーキテクチャーおよびアベイラビリティーゾーン間で、Block Storage サービス (cinder) ボリュームをバックアップしてリストアすることができます。**cinder-backup** サービスは中央のアベイラビリティーゾーン (AZ) で実行され、バックアップは中央の AZ に保存されます。Block Storage サービスは、DCN サイトにバックアップを保存しません。

前提条件

- オプションの Block Storage バックアップサービスをデプロイします。詳細は、**Block Storage ボリュームのバックアップ** の Block Storage バックアップサービスのデプロイ を参照してください。
- Block Storage (cinder) REST API マイクロバージョン 3.51 以降。
- すべてのサイトは共通の **openstack** cephx クライアント名を使用する必要があります。詳細は、**分散コンピュートノード (DCN) アーキテクチャーのデプロイ** の 外部アクセス用 Ceph キーの作成 を参照してください。

手順

1. 最初の DCN サイトのボリュームのバックアップを作成します。

```
$ cinder --os-volume-api-version 3.51 backup-create --name <volume_backup> --availability-zone <az_central> <edge_volume>
```

- **<volume_backup>** をボリュームバックアップの名前に置き換えます。

- **<az_central>** を、**cinder-backup** サービスをホストする中央アベイラビリティーゾーンの名前に置き換えます。
- **<edge_volume>** をバックアップするボリュームの名前に置き換えます。

注記

Ceph キーリングに問題がある場合には、**cinder-backup** コンテナーを再起動して、キーリングがホストからコンテナーに正常にコピーされるようにする必要がある場合があります。

2. 2 番目の DCN サイトの新規ボリュームにバックアップを復元します。

```
$ cinder --os-volume-api-version 3.51 create --availability-zone <az_2> --name
<new_volume> --backup-id <volume_backup> <volume_size>
```

- **<az_2>** を、バックアップを復元するアベイラビリティーゾーンの名前に置き換えます。
- **<new_volume>** を新規ボリュームの名前に置き換えます。
- **<volume_backup>** を、前のステップで作成したボリュームバックアップの名前に置き換えます。
- **<volume_size>** を、元のボリュームのサイズと同じまたはそれ以上の値に置き換えます (GB 単位)。

A.5. DCN 環境でのオーバークラウドの導入と準備

オーバークラウドを導入するには、次のタスクを実行する必要があります。

- 各サイトは、中央の場所から始めて、1 つずつ個別に完全にアップグレードされます。
- 中央ロケーションスタック用に、ネットワークとホストのプロビジョニング設定のエクスポートをオーバークラウドに採用します。```suggestion:-0+0
- 新しいコンテナーと追加の互換性設定を定義します。

導入後、次のタスクを実行するアップグレード準備スクリプトを実行する必要があります。

- オーバークラウドのプランを OpenStack Platform 17.1 に更新する。
- アップグレードに向けてノードを準備する。

このアップグレード手順の所要時間と影響については、アップグレードの所要時間と影響 を参照してください。

前提条件

- すべてのノードが **ACTIVE** 状態になっている。

```
$ openstack baremetal node list
```

いずれかのノードが **MAINTENANCE** 状態にある場合は、そのノードを **ACTIVE** に設定します。

```
$ openstack baremetal node maintenance unset <node_uuid>
```

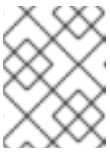
- **<node_uuid>** をノードの UUID に置き換えます。

手順

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. **stackrc** アンダークラウド認証情報ファイルを入手します。

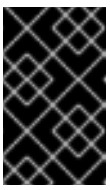
```
$ source ~/stackrc
```

3. アンダークラウドのアップグレード中にエクスポートされた以下のファイルに、オーバークラウドのアップグレードで想定される設定が含まれていることを確認します。**~/overcloud-deploy** ディレクトリーには以下のファイルがあります。
   - **tripleo-<stack>-passwords.yaml**
   - **tripleo-<stack>-network-data.yaml**
   - **tripleo-<stack>-virtual-ips.yaml**
   - **tripleo-<stack>-baremetal-deployment.yaml**



注記

ファイルがアンダークラウドのアップグレード後に生成されなかった場合は、Red Hat サポートにお問い合わせください。



重要

マルチセル環境をお使いの場合は、マルチセル環境でのオーバークラウドの導入 を参照し、ファイルを各セルスタックにコピーする例を確認してください。

4. メインスタックで、**passwords.yaml** ファイルを **~/overcloud-deploy/$(<stack>)** ディレクトリーにコピーします。環境内の各スタックでこの手順を繰り返します。

```
$ cp ~/overcloud-deploy/<stack>/tripleo-<stack>-passwords.yaml ~/overcloud-
deploy/<stack>/<stack>-passwords.yaml
```

   - **<stack>** は、スタックの名前に置き換えます。

5. 準備と導入を中央の場所で実行する場合は、**network-data.yaml** ファイルをスタックユーザーのホームディレクトリーにコピーし、ネットワークをデプロイします。中央の場所に対してのみこれを実行します。

```
$ cp /home/stack/overcloud-deploy/central/tripleo-central-network-data.yaml ~/
$ mkdir /home/stack/overcloud_adopt
$ openstack overcloud network provision --debug \
--output /home/stack/overcloud_adopt/generated-networks-deployed.yaml tripleo-central-
network-data.yaml
```

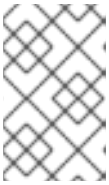
詳細は、**director を使用した Red Hat OpenStack Platform のインストールと管理**の オーバークラウドのプロビジョニングとデプロイ を参照してください。

6. 中央の場所で準備と導入を実行する場合は、**virtual-ips.yaml** ファイルをスタックユーザーのホームディレクトリーにコピーし、ネットワーク仮想 IP をプロビジョニングします。中央の場所に対してのみこれを実行します。

```
$ cp /home/stack/overcloud-deploy/central/tripleo-central-virtual-ips.yaml ~/
$ openstack overcloud network vip provision --debug \
--stack <stack> --output \
/home/stack/overcloud_adopt/generated-vip-deployed.yaml tripleo-central-virtual-ips.yaml
```

7. メインスタックで、**baremetal-deployment.yaml** ファイルをスタックユーザーのホームディレクトリーにコピーし、オーバークラウドノードをプロビジョニングします。環境内の各スタックでこの手順を繰り返します。

```
$ cp ~/overcloud-deploy/<stack>/tripleo-<stack>-baremetal-deployment.yaml ~/
$ openstack overcloud node provision --debug --stack <stack> \
--output /home/stack/overcloud_adopt/baremetal-central-deployment.yaml \
tripleo-<stack>-baremetal-deployment.yaml
```



注記

これはオーバークラウド導入の最終ステップです。オーバークラウドの導入が完了するまでに 10 分以上かかる場合は、Red Hat サポートにお問い合わせください。

8. コンテナーを準備するには、以下の手順を実行します。

   a. アンダークラウドのアップグレードに使用した **containers-prepare-parameter.yaml** ファイルをバックアップします。

```
$ cp containers-prepare-parameter.yaml \
containers-prepare-parameter.yaml.orig
```

   b. スクリプトを実行して **containers-prepare-parameter.yaml** ファイルを更新する前に、以下の環境変数を定義します。

   - **NAMESPACE**: UBI9 イメージの名前空間。たとえば、**NAMESPACE='"namespace":"example.redhat.com:5002",'** など。
   - **EL8_NAMESPACE**: UBI8 イメージの名前空間。
   - **NEUTRON_DRIVER**: 使用する OpenStack Networking (neutron) コンテナーを定義するために使用するドライバー。元のスタックのデプロイに使用したコンテナーのタイプに設定します。たとえば、OVN ベースのコンテナーを使用するには、**NEUTRON_DRIVER='"neutron_driver":"ovn",'** に設定します。
   - **EL8_TAGS**: UBI8 イメージのタグ (例: **EL8_TAGS='"tag":"17.1",'**)。
     - **17.1** は、コンテンツビューで使用するタグに置き換えます。
   - **EL9_TAGS**: UBI9 イメージのタグ (例: **EL9_TAGS='"tag":"17.1",'**)。
     - **17.1** は、コンテンツビューで使用するタグに置き換えます。

tag パラメーターの詳細は、**Red Hat OpenStack Platform デプロイメントのカスタマイズ** の コンテナーイメージ準備のパラメーター を参照してください。

- **CONTROL_PLANE_ROLES**: **--role** オプションを使用したコントロールプレーンロールのリスト (例: **--role ControllerOpenstack, --role Database, --role Messaging, --role Networker, --role CephStorage**)。環境内のコントロールプレーンのロールのリストを表示するには、以下のコマンドを実行します。

```
$ export STACK=<stack> \
$ sudo awk '/tripleo_role_name/ {print "--role " $2}' \
/var/lib/mistral/${STACK}/tripleo-ansible-inventory.yaml \
| grep -vi compute
```

  - **<stack>** は、スタックの名前に置き換えます。

- **COMPUTE_ROLES**: **--role** オプションを使用したコンピュートロールのリスト (**--Compute-1** など)。環境内のコンピュートロールのリストを表示するには、以下のコマンドを実行します。

```
$ sudo awk '/tripleo_role_name/ {print "--role " $2}' \
/var/lib/mistral/${STACK}/tripleo-ansible-inventory.yaml \
| grep -i compute
```

- **CEPH_OVERRIDE**: Red Hat Ceph Storage をデプロイした場合は、Red Hat Ceph Storage 5 コンテナーイメージを指定します。以下に例を示します。
CEPH_OVERRIDE='"ceph_image":"rhceph-5-rhel8","ceph_tag":"<latest>",'

  - **<latest>** は、最新の **ceph_tag** バージョン (**5-499** など) に置き換えます。
以下は、**containers-prepare-parameter.yaml** ファイル設定の例です。

```
NAMESPACE='"namespace":"registry.redhat.io/rhosp-rhel9",'
EL8_NAMESPACE='"namespace":"registry.redhat.io/rhosp-rhel8",'
NEUTRON_DRIVER='"neutron_driver":"ovn",'
EL8_TAGS='"tag":"17.1",'
EL9_TAGS='"tag":"17.1",'
CONTROL_PLANE_ROLES="--role Controller"
COMPUTE_ROLES="--role Compute"
CEPH_TAGS='"ceph_tag":"5",'
```

c. 以下のスクリプトを実行して、**containers-prepare-parameter.yaml** ファイルを更新します。

警告

Red Hat Ceph Storage をデプロイした場合は、次のコマンドを実行する前に、**CEPH_OVERRIDE** 環境変数が正しい値に設定されていることを確認してください。これを行わないと、Red Hat Ceph Storage のアップグレード時に問題が発生します。

```
$ python3 /usr/share/openstack-tripleo-heat-templates/tools/multi-rhel-container-image-prepare.py \
    ${COMPUTE_ROLES} \
    ${CONTROL_PLANE_ROLES} \
    --enable-multi-rhel \
    --excludes collectd \
    --excludes nova-libvirt \
    --minor-override "{${EL8_TAGS}${EL8_NAMESPACE}${CEPH_OVERRIDE}${NEUTRON_DRIVER}\"no_tag\":\"not_used\"}" \
    --major-override "{${EL9_TAGS}${NAMESPACE}${CEPH_OVERRIDE}${NEUTRON_DRIVER}\"no_tag\":\"not_used\"}" \
    --output-env-file \
   /home/stack/containers-prepare-parameter.yaml
```

multi-rhel-container-image-prepare.py スクリプトは、次のパラメーターをサポートしています。

--output-env-file

デフォルトの **ContainerImagePrepare** 値を含む環境ファイルを書き込みます。

--local-push-destination

ローカルレジストリーへのアップロードをトリガーします。

--enable-registry-login

コンテナーをプルする前に、システムがリモートレジストリーへのログインを試行できるようにするフラグを有効にします。このフラグは、**--local-push-destination** が使用されておらず、ターゲットシステムにリモートレジストリーへのネットワーク接続がある場合に使用します。このフラグは、リモートレジストリーへのネットワーク接続がない可能性があるオーバークラウドには使用しないでください。

--enable-multi-rhel

multi-rhel を有効にします。

--excludes

除外するサービスをリストします。

--major-override

メジャーリリースのオーバーライドパラメーターをリストします。

--minor-override

マイナーリリースのオーバーライドパラメーターをリストします。

--role

ロールのリスト。

--role-file

role_data.yaml ファイル。

d. Red Hat Ceph Storage をデプロイした場合は、**containers-prepare-parameter.yaml** ファイルを開いて、Red Hat Ceph Storage 5 コンテナーイメージが指定されていること、および Red Hat Ceph Storage 6 コンテナーイメージへの参照がないことを確認します。

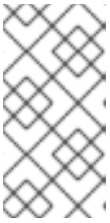
9. director でデプロイされた Red Hat Ceph Storage デプロイメントがある場合は、**ceph_params.yaml** というファイルを作成し、次の内容を含めます。

```
parameter_defaults:
```

```
CephSpecFqdn: true
CephConfigPath: "/etc/ceph"
CephAnsibleRepo: "rhceph-5-tools-for-rhel-8-x86_64-rpms"
DeployedCeph: true
```

重要

RHOSP のアップグレード完了後に **ceph_params.yaml** ファイルを削除しないでください。このファイルは、director でデプロイされた Red Hat Ceph Storage 環境に存在する必要があります。さらに、**openstack overcloud deploy** を実行するときは、**-e ceph_params.yaml** を指定するなどして、常に **ceph_params.yaml** ファイルを含める必要があります。



注記

Red Hat Ceph Storage デプロイメントに短縮名が含まれている場合は、**CephSpecFqdn** パラメーターを **false** に設定する必要があります。**true** に設定すると、短縮名とドメイン名の両方を使用してインベントリーが生成され、Red Hat Ceph Storage のアップグレードが失敗します。

10. テンプレートディレクトリーに **upgrades-environment.yaml** という環境ファイルを作成し、以下の内容を含めます。

```
parameter_defaults:
  ExtraConfig:
    nova::workarounds::disable_compute_service_check_for_ffu: true
  DnsServers: ["<dns_servers>"]
  DockerInsecureRegistryAddress: <undercloud_FQDN>
  UpgradeInitCommand: |
    sudo subscription-manager repos --disable=*
      if $( grep -q  9.2  /etc/os-release )
      then
        sudo subscription-manager repos --enable=rhel-9-for-x86_64-baseos-eus-rpms --
enable=rhel-9-for-x86_64-appstream-eus-rpms --enable=rhel-9-for-x86_64-highavailability-
eus-rpms --enable=openstack-17.1-for-rhel-9-x86_64-rpms --enable=fast-datapath-for-rhel-9-
x86_64-rpms
        sudo podman ps | grep -q ceph && subscription-manager repos --enable=rhceph-5-
tools-for-rhel-9-x86_64-rpms
        sudo subscription-manager release --set=9.2
      else
        sudo subscription-manager repos --enable=rhel-8-for-x86_64-baseos-tus-rpms --
enable=rhel-8-for-x86_64-appstream-tus-rpms --enable=rhel-8-for-x86_64-highavailability-
tus-rpms --enable=openstack-17.1-for-rhel-8-x86_64-rpms --enable=fast-datapath-for-rhel-8-
x86_64-rpms
        sudo podman ps | grep -q ceph && subscription-manager repos --enable=rhceph-5-
tools-for-rhel-8-x86_64-rpms
        sudo subscription-manager release --set=8.4
      fi

      if $(sudo podman ps | grep -q ceph )
      then
        sudo dnf -y install cephadm
      fi
```

- **<dns_servers>** を、DNS サーバーの IP アドレスのコンマ区切りのリスト (**["10.0.0.36", "10.0.0.37"]** など) に置き換えます。
- **<undercloud_FQDN>** をアンダークラウドホストの完全修飾ドメイン名 (FQDN) に置き換えます (例: **"undercloud-0.ctlplane.redhat.local:8787")**。
  環境ファイルで設定できるアップグレードパラメーターの詳細は、アップグレードパラメーター を参照してください。

11. エッジロケーションで準備と導入を実行する場合は、**AuthCloudName** パラメーターを中央ロケーションの名前に設定します。

```
parameter_defaults:
  AuthCloudName: central
```

12. 複数の Image サービス (glance) ストアがデプロイされている場合は、copy-image の Image サービス API ポリシーをすべてのルールを許可するように設定します。

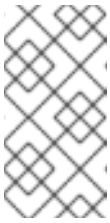
```
parameter_defaults:
    GlanceApiPolicies: {glance-copy_image: {key 'copy-image', value: ""}}
```

13. アンダークラウドで、テンプレートディレクトリーに **overcloud_upgrade_prepare.sh** というファイルを作成します。

- このファイルは、環境内のスタックごとに作成する必要があります。このファイルには、オーバークラウドのデプロイファイルの元の内容と、使用中の環境に関連する環境ファイルが含まれています。
- DCN エッジロケーション用に **overcloud_upgrade_prepare.sh** を作成する場合は、次のテンプレートを含める必要があります。
  - エクスポートされた中央サイトパラメーターを含む環境テンプレート。このファイルは **/home/stack/overcloud-deploy/centra/central-export.yaml** にあります。
  - **generated-networks-deployed.yaml**、中央の場所で **openstack overcloud network provision** コマンドを実行した結果のファイル。
  - **generated-vip-deployed.yaml**、中央の場所で **openstack overcloud network vip provision** コマンドを実行した結果のファイル。以下に例を示します。

```
#!/bin/bash
openstack overcloud upgrade prepare --yes \
  --timeout 460 \
  --templates /usr/share/openstack-tripleo-heat-templates \
  --ntp-server 192.168.24.1 \
  --stack <stack> \
  -r /home/stack/roles_data.yaml \
  -e /home/stack/templates/internal.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovs.yaml \
  -e /home/stack/templates/network/network-environment.yaml \
  -e /home/stack/templates/inject-trust-anchor.yaml \
  -e /home/stack/templates/hostnames.yml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml \
  -e /home/stack/templates/nodes_data.yaml \
  -e /home/stack/templates/debug.yaml \
```

```
  -e /home/stack/templates/firstboot.yaml \
  -e /home/stack/templates/upgrades-environment.yaml \
  -e /home/stack/overcloud-params.yaml \
  -e /home/stack/overcloud-deploy/<stack>/overcloud-network-environment.yaml \
  -e /home/stack/overcloud-adopt/<stack>-passwords.yaml \
  -e /home/stack/overcloud_adopt/<stack>-baremetal-deployment.yaml \
  -e /home/stack/overcloud_adopt/generated-networks-deployed.yaml \
  -e /home/stack/overcloud_adopt/generated-vip-deployed.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/nova-hw-machine-
type-upgrade.yaml \
  -e /home/stack/skip_rhel_release.yaml \
  -e ~/containers-prepare-parameter.yaml
```



注記

マルチセル環境をお使いの場合は、マルチセル環境でのオーバークラウドの導入 を参照し、セルスタックごとに **overcloud_upgrade_prepare.sh** ファイルを作成する例を確認してください。

a. 元の **network-environment.yaml** ファイル (**/home/stack/templates/network/network-environment.yaml**) で、**OS::TripleO::*::Net::SoftwareConfig** を指す resource_registry リソースをすべて削除します。

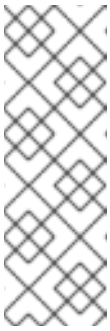
b. **overcloud_upgrade_prepare.sh** ファイルに、環境に関連する以下のオプションを含めます。

- アップグレード固有のパラメーターを持つ環境ファイル (**upgrades-environment.yaml**) (**-e**)
- 新しいコンテナーイメージの場所を定義した環境ファイル (**containers-prepare-parameter.yaml**) (**-e**)。多くの場合、アンダークラウドが使用する環境ファイルと同じファイルです。
- リリースパラメーターを持つ環境ファイル (**skip_rhel_release.yaml**) (-e)。
- デプロイメントに関連するカスタム設定環境ファイル (**-e**)
- 該当する場合は、**--roles-file** を使用してカスタムロール (**roles_data**) ファイルを指定します。
- Ceph デプロイメントの場合、Ceph パラメーターを持つ環境ファイル (**ceph_params.yaml**) (-e)。
- オーバークラウドの導入中に生成されたファイル (**network-deployed.yaml**、**vip-deployed.yaml**、**baremetal-deployment.yaml**) (-e)。
- 該当する場合、環境ファイル (**ipa-environment.yaml**) と IPA サービス (-e)。
- コンポーザブルネットワークを使用している場合は、**--network-file** を使用して (**network_data**) ファイルを指定します。

注記

オーバークラウドのデプロイファイルや **overcloud_upgrade_prepare.sh** ファイルに **network-isolation.yaml** ファイルを含めないでください。ネットワークの分離は **network_data.yaml** ファイルで定義します。

- カスタムのスタック名を使用する場合は、**--stack** オプションでその名前を渡します。



注記

環境内のすべての RHEL 8 コンピュートノードが RHEL 9 にアップグレードされるまで、テンプレートに **nova-hw-machine-type-upgrade.yaml** ファイルを含める必要があります。このファイルを除外すると、**/var/log/containers/nova** ディレクトリーの **nova_compute.log** にエラーが表示されます。すべての RHEL 8 コンピュートノードを RHEL 9 にアップグレードした後、このファイルを設定から削除してスタックを更新できます。

a. director でデプロイされた Red Hat Ceph Storage のユースケースでは、アップグレードするデプロイメントで、CephFS NFS を使用する Shared File Systems サービス (manila) を有効にしていた場合、**overcloud_upgrade_prepare.sh** スクリプトファイルの最後に追加の環境ファイルを指定する必要があります。環境ファイルは、スクリプトの前の方で指定されている別の環境ファイルをオーバーライドするため、スクリプトの最後に追加する必要があります。

```
-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/manila-cephfsganesha-config.yaml
```

b. 外部 Red Hat Ceph Storage のユースケースでは、アップグレードするデプロイメントで、CephFS NFS を使用する Shared File Systems サービス (manila) を有効にしていた場合、**overcloud_upgrade_prepare.sh** スクリプト内の関連する環境ファイルが tripleo ベースの **ceph-nfs** ロールを参照することを確認する必要があります。次の環境ファイルが存在する場合は、削除します。

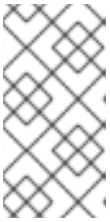
```
-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/manila-cephfsganesha-config.yaml
```

次の環境ファイルを追加します。

```
-e /usr/share/openstack-tripleo-heat-templates/environments/manila-cephfsganesha-config.yaml
```

14. 環境内のスタックごとにアップグレード準備スクリプトを実行します。

```
$ source stackrc
$ chmod 755 /home/stack/overcloud_upgrade_prepare.sh
$ sh /home/stack/overcloud_upgrade_prepare.sh
```



注記

マルチセル環境をお使いの場合は、各セルスタック用に作成した **overcloud_upgrade_prepare.sh** ファイルごとにスクリプトを実行する必要があります。例については、マルチセル環境でのオーバークラウドの導入 を参照してください。

15. アップグレードの準備が完了するまで待ちます。

16. コンテナーイメージをダウンロードします。

```
$ openstack overcloud external-upgrade run --stack <stack> --tags
container_image_prepare
```