



Red Hat OpenStack Platform 16.1

分散コンピューートノードおよびストレージのデプロイメント

Red Hat OpenStack Platform 分散コンピューートノードテクノロジーのデプロイ

Red Hat OpenStack Platform 16.1 分散コンピュートノードおよびストレージのデプロイメント

Red Hat OpenStack Platform 分散コンピュートノードテクノロジーのデプロイ

OpenStack Team
rhos-docs@redhat.com

法律上の通知

Copyright © 2023 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

概要

分散コンピューターノード (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	8
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	8
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	8
2.3. エッジサイトでのストレージトポロジーおよびロール	10
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. リーフネットワーク向けのフレーバーの作成とノードのタグ付け	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. 複数スタックでのネットワークリソースの再利用	23
4.5. MANAGENETWORKS を使用したネットワークリソースの再利用	23
4.6. UUID を使用したネットワークリソースの再利用	24
4.7. 個別の HEAT スタックの管理	25
4.8. コンテナイメージの取得	25
4.9. エッジサイト用の高速データパスロールの作成	26
第5章 中央サイトのインストール	28
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	28
5.2. ストレージが設定された中央サイトのデプロイ	30
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	35
6.1. ストレージを持たないエッジノードのデプロイ	35
第7章 エッジサイトでのストレージのデプロイ	38
7.1. ストレージが設定されたエッジサイトのデプロイ	38
7.2. 追加の分散コンピュートノードサイトの作成	42
7.3. 中央サイトの更新	44
7.4. DCN への RED HAT CEPH STORAGE DASHBOARD のデプロイ	46
第8章 KEY MANAGER を含むデプロイ	49
8.1. KEY MANAGER が設定されたエッジサイトのデプロイ	49
第9章 GLANCE イメージの NOVA への事前キャッシュ	50
9.1. TRIPLEO_NOVA_IMAGE_CACHE.YML ANSIBLE PLAYBOOK の実行	50
9.2. パフォーマンスに関する考慮事項	51
9.3. DCN サイトへのイメージ配布の最適化	52

9.4. NOVA-CACHE クリーンアップの設定	52
第10章 DCN への TLS-E の適用	54
10.1. 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	54
第11章 外部アクセス用 CEPH キーの作成	56
11.1. 外部アクセス用 CEPH キーの作成	56
11.2. 外部 CEPH キーの使用	57
付録A デプロイメントの移行オプション	59
A.1. エッジストレージの検証	59
A.2. スパイン/リーフ型デプロイメントへの移行	62
A.3. マルチスタックデプロイメントへの移行	63
A.4. エッジサイト間のバックアップおよびリストア	63

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 プロパティにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、[Red Hat CTO である Chris Wright のメッセージ](#) を参照してください。:leveloffset: +0

RED HAT ドキュメントへのフィードバック (英語のみ)

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

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

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

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

第1章 DCN について

分散コンピューティングノード (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ティング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。

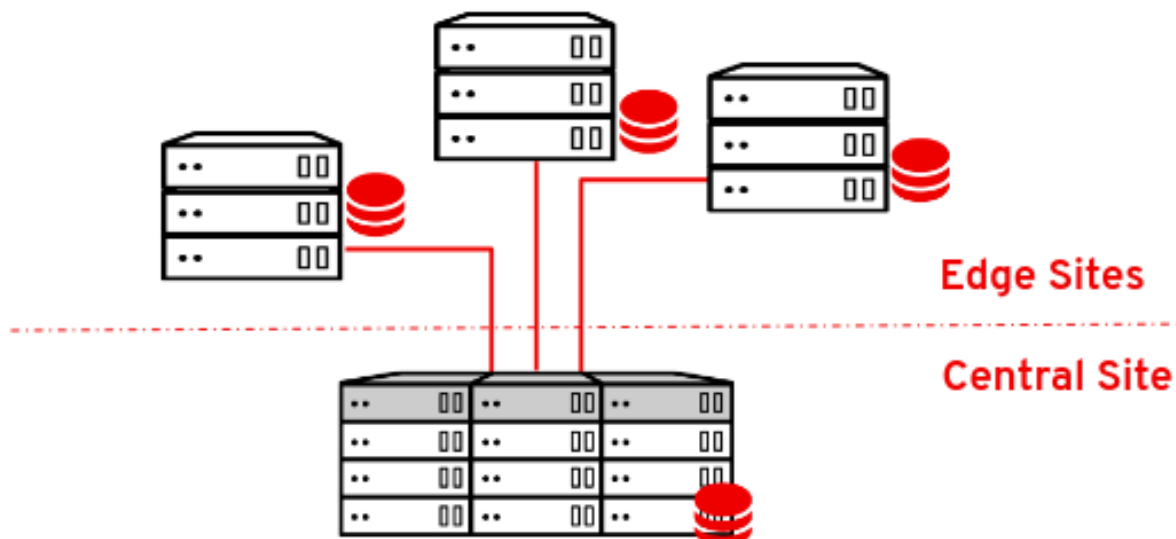
中央サイトは任意のロールで設定することができますが、最低でも3つのコントローラーが必要です。コンピューティングノードは、中央サイト以外にエッジサイトに設定することができます。


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。



 Dedicated (to-site) Ceph Cluster

1.1. 分散コンピューティング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、分散コンピューティングノード (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	オプション
Red Hat Enterprise Linux	8	いいえ
Red Hat OpenStack Platform	16.1	いいえ
Red Hat Ceph Storage	4	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、コンピュートノードまたは 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、[9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のアーキテクチャーでは、中央サイトはエッジロケーションを管理するコントロールノードと共にデプロイされます。エッジロケーションをデプロイする場合には、コンピュートノードのみをデプロイし、エッジサイトのアーキテクチャーは Red Hat OpenStack Platform の従来のデプロイメントとは異なります。エッジロケーション:

- 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 分散コンピューターノード (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。

- 分散コンピューターノードアーキテクチャー上での Red Hat OpenStack Platform 13 から 16 への Fast Forward Upgrade (FFU)
- エッジサイトにおける非ハイパーコンバージドストレージノード
- 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。
- サイト間でのボリュームの移行または種別変更
- 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、または中央の場所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) に存在する必要があります。
- エッジサイトでインスタンスを作成する前に、そのエッジサイトにイメージのローカルコピーが必要です。
- 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。

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。

- Octavia
- DPDK ノード上の DHCP
- TC Flower ハードウェアオフロードの contrack

ML2/OVN メカニズムドライバー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一緒に使用することは、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。



注記

ML2/OVN メカニズムドライバーは、DCN 環境外では完全にサポートされます。

以下のネットワークテクノロジーと ML2/OVS の組み合わせがサポートされます。

- DPDK ノード上の DHCP を使用しない DPDK
- SR-IOV
- contrack を使用しない TC Flower ハードウェアオフロード
- Neutron アベイラビリティゾーン (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 サイトごとの AZ)
- ルーティング対応プロバイダーネットワーク

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す。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、エッジサイトのコンピュートノードは、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コンピュート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。コンピュートノードと共に集中 neutron コントローラーをデプロイする場合には、neutron では特定のコンピュート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、コンピュートノードは他のコンピュート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コンピュートノードが、他のすべてのコンピュート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、すべてのコンピュート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。

2.3. エッジサイトでのストレージトポロジーおよびロール

分散コンピュートノードのアーキテクチャーと共に Red Hat OpenStack Platform をデプロイする場合には、エッジサイトにストレージが必要かどうかを決定する必要があります。ストレージおよびパフォーマンスのニーズに応じて、以下の 3 つの設定の 1 つを使用して各サイトをデプロイできます。すべてのエッジサイトの設定が同一でなければならない訳ではありません。

エッジサイトにブロックストレージをデプロイしない場合は、本ドキュメントの「[ストレージを持たないエッジノードのデプロイ](#)」セクションの手順に従う必要があります。エッジサイトにブロックストレージがない場合:

- Swift が Glance のバックエンドとして使用されます。
- エッジサイトのコンピュートノードは、イメージをキャッシュすることしかできません。
- Cinder 等のボリュームサービスは、エッジサイトでは利用することができません。

エッジサイトにストレージをデプロイする場合には、その場所に関わらず中央サイトにもブロックストレージをデプロイする必要があります。本ドキュメントの「[ストレージが設定された中央サイトのデプロイ](#)」の章に記載する手順に従ってください。エッジサイトにブロックストレージがある場合:

- Ceph RBD が Glance のバックエンドとして使用されます。
- イメージはエッジサイトに保管されます。
- Cinder ボリュームサービスは、Ceph RBD ドライバーを介して利用することができます。

デプロイメントに必要なロールは、エッジサイトにブロックストレージをデプロイするかどうかによって異なります。

- **エッジサイトにブロックストレージが不要な場合:**

コンピュート

ブロックストレージなしでエッジロケーションをデプロイする場合は、従来の **compute** ロールを使用する必要があります。

- **エッジサイトにブロックストレージが必要な場合:**

DistributedComputeHCI

このロールには、以下のサービスが含まれます。

- デフォルトの Compute サービス
- Block Storage (cinder) ボリュームサービス
- Ceph Mon
- Ceph Mgr
- Ceph OSD
- GlanceApiEdge
- etcd

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。

DistributedComputeHCIScaleOut

このロールには **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ルアップ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 **HProxyEdge** サービスも含まれています。

DistributedComputeScaleOut

エッジサイトでストレージ以外のコンピュートリソースをスケールアップする場合は、**DistributedComputeScaleOut** ロールを使用することができます。

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是一般的です。**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. リーフネットワーク向けのフレーバーの作成とノードのタグ付け

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. 各カスタムロール用のフレーバーを作成します。

```
$ ROLES="control compute_leaf0 compute_leaf1 compute_leaf2 ceph-storage_leaf0 ceph-storage_leaf1 ceph-storage_leaf2"
$ for ROLE in $ROLES; do openstack flavor create --id auto --ram <ram_size_mb> --disk <disk_size_gb> --vcpus <no_vcpus> $ROLE ; done
$ for ROLE in $ROLES; do openstack flavor set --property "cpu_arch"="x86_64" --property "capabilities:boot_option"="local" --property resources:DISK_GB='0' --property resources:MEMORY_MB='0' --property resources:VCPU='0' $ROLE ; done
```

- **<ram_size_mb>** をベアメタルノードの RAM (MB 単位) に置き換えます。
 - **<disk_size_gb>** をベアメタルノード上のディスク容量 (GB 単位) に置き換えます。
 - **<no_vcpus>** をベアメタルノードの CPU 数に置き換えます。
3. ノードリストを取得して UUID を把握します。

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

4. カスタムリソースクラスを使用して、各ベアメタルノードをリーフネットワークおよびロールにタグ付けします。

```
(undercloud)$ openstack baremetal node set \
--resource-class baremetal.LEAF-ROLE <node>
```

<node> をベアメタルノードの ID に置き換えてください。

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
```

5. 各リーフネットワークロールフレーバーをカスタムリソースクラスに関連付けます。

```
(undercloud)$ openstack flavor set \
--property resources:CUSTOM_BAREMETAL_LEAF_ROLE=1 \
<custom_role>
```

Bare Metal Provisioning サービスノードのリソースクラスに対応するカスタムリソースクラスの名前を指定するには、リソースクラスを大文字に変換し、それぞれの句読点をアンダースコアに置き換え、**CUSTOM_** の接頭辞を追加します。



注記

フレーバーが要求できるのは、ベアメタルリソースクラスの1つのインスタンスだけです。

6. **node-info.yaml** ファイルで、各カスタムリーフロールに使用するフレーバーおよび各カスタムリーフロールに割り当てるノード数を指定します。たとえば、以下の設定では、使用するフレーバーと、カスタムのリーフロール **compute_leaf0**、**compute_leaf1**、**compute_leaf2**、**ceph-storage_leaf0**、**ceph-storage_leaf1**、および **ceph-storage_leaf2** に割り当てるノード数を指定します。

```
parameter_defaults:
  OvercloudControllerFlavor: control
  OvercloudComputeLeaf0Flavor: compute_leaf0
  OvercloudComputeLeaf1Flavor: compute_leaf1
  OvercloudComputeLeaf2Flavor: compute_leaf2
  OvercloudCephStorageLeaf0Flavor: ceph-storage_leaf0
  OvercloudCephStorageLeaf1Flavor: ceph-storage_leaf1
  OvercloudCephStorageLeaf2Flavor: ceph-storage_leaf2
  ControllerLeaf0Count: 3
  ComputeLeaf0Count: 3
  ComputeLeaf1Count: 3
  ComputeLeaf2Count: 3
  CephStorageLeaf0Count: 3
  CephStorageLeaf1Count: 3
  CephStorageLeaf2Count: 3
```

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ポロジが使用されている。

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. source コマンドでアンダークラウドの認証情報ファイルを読み込みます。

```
$ 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。

- 稼働状態にある Red Hat OpenStack Platform 16 アンダークラウド
- Ceph Storage ユーザー: Red Hat Ceph Storage 4 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 コンピュートノード
- すべての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、分散コンピュータノード (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。



注記

スパイン/リーフ型ネットワークを使用する場合は、特定の形式を使用して **Storage** および **StorageMgmt** ネットワークを定義する必要があります。これにより、ceph-ansible はそれらのネットワークを使用するように Ceph を正しく設定することができます。**Storage** および **StorageMgmt** ネットワークをオーバーライド値として定義し、値を一重引用符で囲みます。以下の例では、ストレージネットワーク (**public_network**) はコンマで区切られた 2 つのサブネットにまたがり、一重引用符で囲まれています。

```
CephAnsibleExtraConfig:
  public_network: '172.23.1.0/24,172.23.2.0/24'
```

4.4. 複数スタックでのネットワークリソースの再利用

複数のスタックが同じネットワークリソース (仮想 IP アドレスやサブネット等) を使用するように設定することができます。**ManageNetworks** 設定または **external_resource_*** フィールドのいずれかを使用して、スタック間でネットワークリソースを複製することができます。



注記

external_resource_* フィールドを使用している場合は、**ManageNetworks** 設定を使用しないでください。

スタック間でネットワークを再利用しない場合は、**network_data.yaml** で定義される各ネットワークには、デプロイされるすべてのスタックに渡って一意の名前を指定する必要があります。たとえば、スタック間でネットワークを共有する意図がない限り、ネットワーク名 **internal_api** をスタック間で再利用することはできません。ネットワークに異なる名前および **name_lower** 属性 (例: **InternalApiCompute0** および **internal_api_compute_0**) を設定します。

4.5. MANAGENETWORKS を使用したネットワークリソースの再利用

ManageNetworks 設定を使用すると、複数のスタックで同じ **network_data.yaml** ファイルを使用することができ、設定はグローバルにすべてのネットワークリソースに適用されます。**network_data.yaml** ファイルは、スタックが使用するネットワークリソースを定義します。

```
- name: StorageBackup
  vip: true
  name_lower: storage_backup
  ip_subnet: '172.21.1.0/24'
  allocation_pools: [{'start': '171.21.1.4', 'end': '172.21.1.250'}]
  gateway_ip: '172.21.1.1'
```

ManageNetworks を false に設定すると、ノードはすでに **central** スタックに作成されている既存のネットワークを使用します。

新規スタックが既存のネットワークリソースを管理しないように、以下のシーケンスを使用します。

手順

1. **ManageNetworks: true** と設定するか未設定のままにして、中央スタックをデプロイします。
2. **ManageNetworks: false** の設定で、追加のスタックをデプロイします。

新規ネットワークリソースを追加する場合、たとえばスパイン/リーフ型デプロイメントに新しいリーフを追加する場合は、新たな **network_data.yaml** で中央スタックを更新する必要があります。これは、中央スタックが引き続きネットワークリソースを所有および管理するためです。中央スタックでネットワークリソースが利用可能になったら、それらを使用するように追加のスタックをデプロイすることができます。

4.6. UUID を使用したネットワークリソースの再利用

スタック間でのネットワークの再利用をより厳密に制御する必要がある場合は、**network_data.yaml** ファイルでリソース (ネットワーク、サブネット、セグメント、仮想 IP 等) の **external_resource_*** フィールドを使用することができます。これらのリソースは外部管理と識別され、heat はそれらのリソースに関する作成、更新、または削除を一切行いません。

network_data.yaml ファイルに、必要な各ネットワーク定義のエントリーを追加します。これで、リソースを別のスタックでのデプロイメントに利用できるようになります。

```
external_resource_network_id: Existing Network UUID
external_resource_subnet_id: Existing Subnet UUID
external_resource_segment_id: Existing Segment UUID
external_resource_vip_id: Existing VIP UUID
```

以下の例では、コントロールプレーンスタックからの **internal_api** ネットワークを別のスタックで再利用します。

手順

1. 関連するネットワークリソースの UUID を把握します。

```
$ openstack network show internal_api -c id -f value
$ openstack subnet show internal_api_subnet -c id -f value
$ openstack port show internal_api_virtual_ip -c id -f value
```

2. 上記のコマンドの出力に表示される値を保存し、それらを別のスタックの **network_data.yaml** ファイルの **internal_api** ネットワークのネットワーク定義に追加します。

```
- name: InternalApi
```

```
external_resource_network_id: 93861871-7814-4dbc-9e6c-7f51496b43af
external_resource_subnet_id: c85c8670-51c1-4b17-a580-1cfb4344de27
external_resource_vip_id: 8bb9d96f-72bf-4964-a05c-5d3fed203eb7
name_lower: internal_api
vip: true
ip_subnet: '172.16.2.0/24'
allocation_pools: [{'start': '172.16.2.4', 'end': '172.16.2.250'}]
ipv6_subnet: 'fd00:fd00:fd00:2000::/64'
ipv6_allocation_pools: [{'start': 'fd00:fd00:fd00:2000::10', 'end':
'fd00:fd00:fd00:2000:ffff:ffff:ffff:ffff'}]
mtu: 1400
```

4.7. 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
$ touch dcn-common/central-export.yaml
```

central-export.yaml ファイルは、後で **openstack overcloud export** コマンドで作成されます。本セクションで説明するすべての DCN デプロイメントがこのファイルを使用する必要があるため、このファイルは **dcn-common** ディレクトリー内に作成されます。

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。

4.8. コンテナイメージの取得

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

ナーイメーが含まれるようにする必要があります。

詳細については、[コンテナイメージの準備](#) 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-4-rhel8
      ceph_tag: latest
      name_prefix: openstack-
      namespace: registry.redhat.io/rhosp16-rhel8
      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。

4.9. 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。詳しくは、[Installing the undercloud](#) 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. DDPK 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 DDPK 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。

関連情報

- [カスタムロールの作成](#)
- [サポートされるカスタムロール](#)

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」](#)

Ceph Storage を使用してエッジサイトをデプロイすることを計画している場合は、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](#)
- [「ストレージが設定されたエッジサイトのデプロイ」](#)

第5章 中央サイトのインストール

分散コンピューティングノード (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。

- コンピューティングノードあり/なし
- Red Hat Ceph Storage あり/なし

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。

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ティング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コンピューティングノードは必要ありません。したがって、コンピューティングノード数を **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。

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. 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。

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

2. 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
  ControllerCount: 3
```



```

ComputeCount: 0
OvercloudControllerFlavor: baremetal
OvercloudComputeFlavor: baremetal
ControllerSchedulerHints:
  'capabilities:node': '0-controller-%index%'
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、コンピュート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。中央の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。

3. 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 Controller \
-o ~/central/control_plane_roles.yaml

```

4. 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

```

5. **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 アベイラビリティゾーンと Cinder ストレージアベイラビリティ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
EOF

```

6. 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。

```
#!/bin/bash

source ~/stackrc
time openstack overcloud deploy \
--stack central \
--templates /usr/share/openstack-tripleo-heat-templates/\
-e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
-e ~/central/containers-env-file.yaml \
-e ~/central/overrides.yaml \
-e ~/central/site-name.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。詳しくは、[スパイン/リーフ型ネットワーク](#) 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

- ハブ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 クラスタ用ハードウェア
- ハイパーコンバージドアーキテクチャーでエッジサイトをデプロイする必要があります。
- ハブ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 サービスサーバー用ハードウェア

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

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

マルチストアの Image サービスを使用する 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. 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。

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

2. 利用可能なハードウェアに該当する設定パラメーターおよび Ceph クラスターの名前を設定します。詳しくは、[カスタム設定を使用した Ceph の設定](#) に関するドキュメントを参照してください。

```
cat > /home/stack/central/ceph.yaml << EOF
parameter_defaults:
  CephClusterName: central
  CephAnsibleDisksConfig:
    osd_scenario: lvm
    osd_objectstore: bluestore
  devices:
    - /dev/sda
    - /dev/sdb
  CephPoolDefaultSize: 3
  CephPoolDefaultPgNum: 128

EOF
```

3. 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 ~/central/central_roles.yaml

cat > /home/stack/central/role-counts.yaml << EOF
parameter_defaults:
  ControllerCount: 3
  ComputeCount: 2
  CephStorage: 3
EOF
```

4. 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
```

5. **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 アベイラビリティゾーンと Cinder ストレージアベイラビリティ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
```

6. 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
```

7. その他すべてのテンプレートを準備したら、**central** スタックをデプロイします。

```
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/ceph-ansible/ceph-
  ansible.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/glance.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。詳しくは、[スパイン/リーフ型ネットワーク](#) を参照してください。

ceph-ansible.yaml ファイルのパラメーターは、以下のように設定されます。

- NovaEnableRbdBackend: true
- GlanceBackend: rbd

これらの設定を同時に使用すると、heat により glance.conf のパラメーター **image_import_plugins** の値には **image_conversion** が設定されます。これにより、**glance image-create-via-import --disk-format qcow2...** 等のコマンドで QCOW2 イメージが自動的に変換されます。

これは Ceph RBD に最適な設定です。イメージの変換を無効にするには、**GlanceImageImportPlugin** パラメーターを使用します。

```
parameter_defaults:
  GlanceImageImportPlugin: []
```

5.3. 外部 CEPH の統合

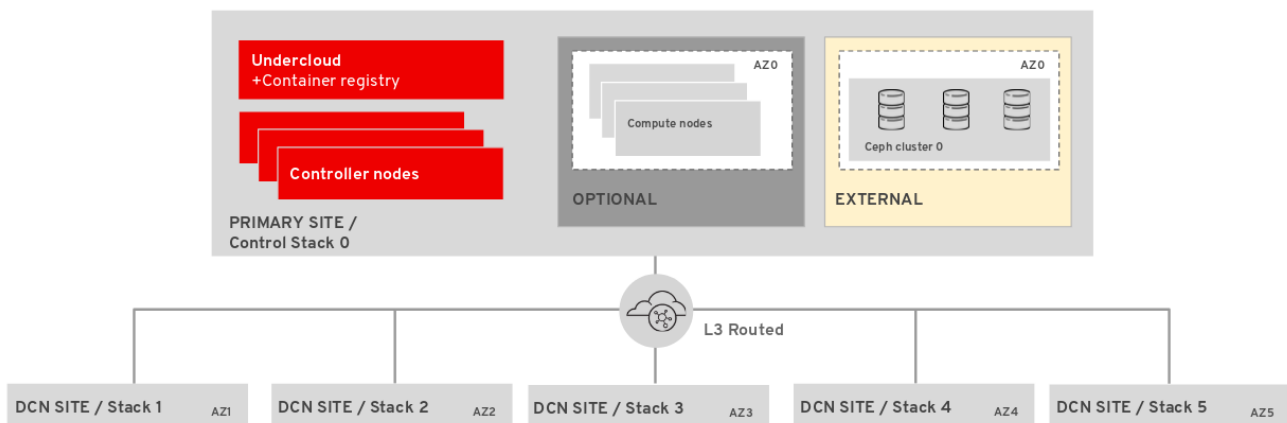
分散コンピュートノード (DCN) アーキテクチャーの中央サイトをデプロイし、事前にデプロイした Red Hat Ceph Storage ソリューションを統合することができます。

前提条件

- 中央サイトおよび各アベイラビリティ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
- ハイパーコンバージドアーキテクチャーでエッジサイトをデプロイする必要があります。
- 中央サイトおよび各アベイラビリティ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 サービスサーバー用ハードウェア

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)



[オーバークラウドと既存の Red Hat Ceph クラスターの統合](#) に記載されているプロセスに従って、中央の場所をインストールして既存の Red Hat Ceph Storage ソリューションと統合することができます。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. 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。

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

2. Red Hat OpenStack Platform director が管理するロールを使用して、中央の場所のロールを生成します。外部 Ceph と統合する場合は、Ceph ロールを使用しないでください。

```
cat > /home/stack/central/role-counts.yaml << EOF
parameter_defaults:
  ControllerCount: 3
  ComputeCount: 2
EOF
```

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. `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 アベイラビリティゾーンと Cinder ストレージアベイラビリティ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
```

5. 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
```

6. Ceph が Red Hat OpenStack Platform director を使用せずにデプロイされる場合には、**`ceph-ansible.yaml`** 環境ファイルを使用しないでください。代わりに **`ceph-ansible-external.yaml`** 環境ファイルを使用してください。

```
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/ceph-ansible/ceph-ansible-external.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/glance.yaml
```

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 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. ストレージを持たないエッジノードのデプロイ

コントロールプレーンに中央サイトを使用するエッジコンピュートノードをデプロイすることができます。以下の手順で、デプロイメントに新たな DCN スタックを追加し、既存の heat スタックからの設定を再利用して新たな環境ファイルを作成する方法について説明します。元の heat スタックにより、中央のデータセンター内にオーバークラウドがデプロイされます。リモートサイトにコンピュートノードをデプロイするために、追加の heat スタックを作成します。

6.1.1. 分散コンピュートノードの環境ファイルの設定

手順

1. DCN サイトに必要な設定ファイルを生成します。

```
openstack overcloud export \
--config-download-dir /var/lib/mistral/central \
--stack central --output-file ~/dcn-common/central-export.yaml
```



重要

この手順で新しい **central-export.yaml** 環境ファイルを作成し、オーバークラウドからの **plan-environment.yaml** ファイルのパスワードを使用します。**central-export.yaml** ファイルには、取り扱いに注意を要するセキュリティーデータが含まれます。セキュリティー向上のために、必要がなくなったらファイルを削除してください。



重要

--config-download-dir オプション用のディレクトリーを指定する場合、デプロイメント時に director が **/var/lib/mistral** に作成する中央のハブの Ansible 設定を使用します。**openstack overcloud config download** コマンドで手動で生成する Ansible 設定は使用しないでください。手動で生成した設定には、デプロイメント操作時にのみ作成される特定のファイルは含まれません。

2. 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。

```
openstack overcloud roles generate Compute -o ~/dcn0/dcn0_roles.yaml
```

注記

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、作成したロールファイルを編集して **NeutronDhcpAgent** および **NeutronMetadataAgent** ロールを含める必要があります。

```
...
- 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。

6.1.2. DCN サイトへのコンピュートノードのデプロイ

以下の手順では、**Compute** ロールを使用して、コンピュートノードを **dcn0** という名前のアベイラビリティゾーン (AZ) にデプロイします。分散コンピュートノード (DC) のコンテキストでは、このロールはストレージのないサイトに使用されます。

手順

1. `dcn0/overrides.yaml` で分散コンピュート (DCN) サイトのオーバーライドを確認します。

```
parameter_defaults:
  ComputeCount: 3
  ComputeFlavor: baremetal
  ComputeSchedulerHints:
    'capabilities:node': '0-compute-%index%'
  NovaAZAttach: false
```

2. 以下の内容で、`~/dcn0` ディレクトリー内に **site-name.yaml** という名前の新たなファイルを作成します。

```
resource_registry:
  OS::TripleO::Services::NovaAZConfig: /usr/share/openstack-tripleo-heat-templates/deployment/nova/nova-az-config.yaml
parameter_defaults:
  NovaComputeAvailabilityZone: dcn0
  RootStackName: dcn0
```

3. DCN サイトのコンテナイメージを取得します。

```
sudo openstack tripleo container image prepare \
--environment-directory dcn0 \
-r ~/dcn0/roles_data.yaml \
```



```
-e ~/dcn-common/central-export.yaml \  
-e ~/containers-prepare-parameter.yaml \  
--output-env-file ~/dcn0/dcn0-images-env.yaml
```

4. dcn0 の deploy.sh デプロイメントスクリプトを実行します。

```
#!/bin/bash  
STACK=dcn0  
source ~/stackrc  
time openstack overcloud deploy \  
  --stack $STACK \  
  --templates /usr/share/openstack-tripleo-heat-templates/ \  
  -e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \  
  -e ~/dcn-common/central-export.yaml \  
  -e ~/dcn0/dcn0-images-env.yaml \  
  -e ~/dcn0/site-name.yaml \  
  -e ~/dcn0/overrides.yaml
```

network_data.yaml ファイルへの編集が必要な追加のエッジサイトをデプロイする場合は、中央サイトでスタックの更新を実行する必要があります。



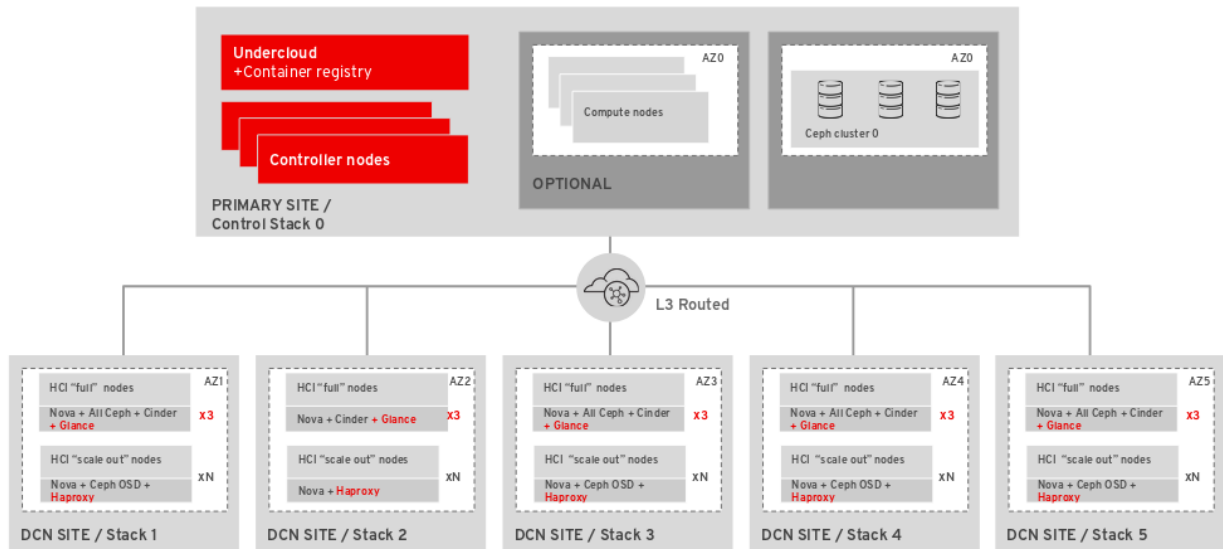
注記

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。詳しくは、[スパイン/リーフ型ネットワーク](#) を参照してください。

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。

OSP 16.1 DCN Architecture



7.1. ストレージが設定されたエッジサイトのデプロイ

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、[スパイン/リーフ型ネットワーク](#) を参照してください。

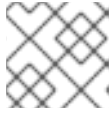
glance イメージをサイト間で移動することができるように、中央サイトと各エッジサイトのストレージネットワーク間に接続が必要です。

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。

手順

1. **central** スタックからスタック情報をエクスポートします。以下のコマンドを実行する前に、**central** スタックをデプロイする必要があります。

```
openstack overcloud export \
  --config-download-dir /var/lib/mistral/central/ \
  --stack central \
  --output-file ~/dcn-common/central-export.yaml
```



注記

config-download-dir のデフォルト値は `/var/lib/mistral/<stack>/` です。

2. **central_ceph_external.yaml** ファイルを作成します。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るので、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あります。

```
sudo -E openstack overcloud export ceph \
--stack central \
--config-download-dir /var/lib/mistral \
--output-file ~/dcn-common/central_ceph_external.yaml
```

Ceph が Red Hat OpenStack Platform director を使用せずにデプロイされる場合には、**openstack overcloud export ceph** コマンドを実行できません。**central_ceph_external.yaml** ファイルを手動で作成します。

```
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+OTz5uhV7ltLdwUw==
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についての詳しい情報は、[カスタム環境ファイルの作成](#) を参照してください。

- Image サービス設定のオーバーライド用に `~/dcn0/glance.yaml` ファイルを作成します。

```
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'
      CephClientUserName: 'openstack'
      CephClusterName: central
```

- 利用可能なハードウェアに該当する設定パラメーターで **ceph.yaml** ファイルを設定します。

```
cat > /home/stack/dcn0/ceph.yaml << EOF
parameter_defaults:
  CephClusterName: dcn0
  CephAnsibleDisksConfig:
    osd_scenario: lvm
    osd_objectstore: bluestore
  devices:
    - /dev/sda
    - /dev/sdb
  CephPoolDefaultSize: 3
  CephPoolDefaultPgNum: 128
EOF
```

詳しくは、[Ceph Storage ノードのディスクレイアウトのマッピング](#) を参照してください。

- 実際の環境要件に適した以下のパラメーターが含まれるファイルを使用して、システムのチューニングを実施します。

```
cat > /home/stack/dcn0/tuning.yaml << EOF
parameter_defaults:
  CephAnsibleExtraConfig:
    is_hci: true
  CephConfigOverrides:
    osd_recovery_op_priority: 3
    osd_recovery_max_active: 3
    osd_max_backfills: 1
  ## Set relative to your hardware:
  # DistributedComputeHCIParameters:
  # NovaReservedHostMemory: 181000
  # DistributedComputeHCIExtraConfig:
  # nova::cpu_allocation_ratio: 8.2
EOF
```

- パラメーター **CephAnsibleExtraConfig** の値の設定については、[Setting ceph-ansible group variables](#) を参照してください。

- パラメーター **CephConfigOverrides** の値の設定の詳細については、[Customizing the Ceph Storage cluster](#) を参照してください。
6. **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 アベイラビリティゾーンと Cinder ストレージアベイラビリティゾーンが一致している必要があります。ストレージと共にエッジサイトをデプロイする場合は、**CinderVolumeCluster** パラメーターを含めます。このパラメーターは、エッジサイトで必要な cinder-volume をアクティブ/アクティブ設定としてデプロイする場合に使用されます。ベストプラクティスとしては、Cinder クラスタ名をアベイラビリティゾーンと一致するように設定します。

```
cat > /home/stack/central/site-name.yaml << EOF
parameter_defaults:
  ...
  NovaComputeAvailabilityZone: dcn0
  NovaCrossAZAttach: false
  CinderStorageAvailabilityZone: dcn0
  CinderVolumeCluster: dcn0
```

7. dcn0 のデプロイメントに使用する **roles.yaml** ファイルを生成します。以下に例を示します。

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

8. それぞれのロールに必要な値で **~/dcn0/roles-counts.yaml** ファイルを作成して、各ロールに番号システムを設定します。

ハイパーコンバージドインフラストラクチャー (HCI) を使用する場合は、Ceph Mon および **GlanceApiEdge** サービスの要件を満たすために、3 台のノードを DistributedComputeHCICount ロールに割り当てる必要があります。

```
parameter_defaults:
  ControllerCount: 0
  ComputeCount: 0
  DistributedComputeHCICount: 3
  DistributedComputeHCIScaleOutCount: 1 # Optional
  DistributedComputeScaleOutCount: 1 # Optional
```

9. エッジサイトのコンテナイメージを取得します。

```
sudo openstack tripleo container image prepare \
--environment-directory dcn0 \
-r ~/dcn0/roles_data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml \
\
...
-e /home/stack/dcn-common/central-export.yaml \
-e /home/stack/containers-prepare-parameter.yaml \
--output-env-file ~/dcn0/dcn0-images-env.yaml
```



注記

openstack tripleo container image prepare コマンドに、デプロイメントに使用するすべての環境ファイルを追加する必要があります。

10. エッジサイトをデプロイします。

```

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/ceph-ansible/ceph-
ansible.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/dcn-hci.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
  -e ~/dcn0/dcn0-images-env.yaml \
  ....
  -e ~/dcn-common/central-export.yaml \
  -e ~/dcn-common/central_ceph_external.yaml \
  -e ~/dcn0/dcn_ceph_keys.yaml \
  -e ~/dcn0/role-counts.yaml \
  -e ~/dcn0/ceph.yaml \
  -e ~/dcn0/site-name.yaml \
  -e ~/dcn0/tuning.yaml \
  -e ~/dcn0/glance.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。詳しくは、[スパイン/リーフ型ネットワーク](#) を参照してください。

7.2. 追加の分散コンピュートノードサイトの作成

新しい分散コンピュートノード (DCN) サイトには、アンダークラウド上に独自の YAML ファイルのディレクトリーがあります。詳細は、「[個別の heat スタックの管理](#)」を参照してください。以下の手順で、コマンドの例を説明します。

手順

1. アンダークラウドの stack ユーザーとして、**dcn9** 用に新規ディレクトリーを作成します。

```

$ cd ~
$ mkdir dcn9

```

2. 既存の **dcn0** テンプレートを新しいディレクトリーにコピーし、**dcn0** の文字列を **dcn9** に置き換えます。

```

$ cp dcn0/ceph.yaml dcn9/ceph.yaml
$ sed s/dcn0/dcn9/g -i dcn9/ceph.yaml
$ cp dcn0/overrides.yaml dcn9/overrides.yaml
$ sed s/dcn0/dcn9/g -i dcn9/overrides.yaml
$ sed s/"0-ceph-%index%"/"9-ceph-%index%"/g -i dcn9/overrides.yaml
$ cp dcn0/deploy.sh dcn9/deploy.sh
$ sed s/dcn0/dcn9/g -i dcn9/deploy.sh

```

3. **dcn9** ディレクトリーのファイルを調べ、要件を満たしていることを確認します。

4. `undercloud.conf` を編集して、新しいリーフを追加します。以下の例では、`leaf9` を `undercloud.conf` に追加します。

```
[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

...
[leaf9]
cidr = 192.168.19.0/24
dhcp_start = 192.168.19.10
dhcp_end = 192.168.19.90
inspection_iprange = 192.168.19.100,192.168.19.190
gateway = 192.168.10.1
masquerade = False
```

5. `openstack undercloud install` コマンドを再度実行して、環境設定を更新します。
6. オーバークラウドのテンプレートで、**NetworkDeploymentActions** パラメーターの値を `["CREATE"]` の値から `["CREATE", "UPDATE"]` の値に更新します。このパラメーターが現在テンプレートに含まれていない場合は、環境ファイルのいずれかに追加するか、新しい環境ファイルを作成します。

```
cat > /home/stack/central/network-environment.yaml << EOF
parameter_defaults:
  NetworkDeploymentActions: ["CREATE", "UPDATE"]
EOF
```

7. 中央サイトのデプロイスクリプトを実行します。中央サイトの初回デプロイ時に使用した全テンプレートと、新たに作成または編集した `network-environment.yaml` ファイルを含めます。

```
openstack overcloud deploy \
  --stack central \
  --templates /usr/share/openstack-tripleo-heat-templates/ \
  -r ~/central/roles_data.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/dcn-hci.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
  -e ~/central/dcn9-images-env.yaml \
  ....
  -e ~/dcn-common/central-export.yaml \
  -e ~/dcn-common/central_ceph_external.yaml \
  -e ~/central/dcn_ceph_keys.yaml \
  -e ~/central/role-counts.yaml \
  -e ~/central/ceph.yaml \
  -e ~/central/site-name.yaml \
  -e ~/central/tuning.yaml \
  -e ~/central/glance.yaml
```

8. ノードが利用可能で、**Provisioning state** の状態にあることを確認します。

```
$ openstack baremetal node list
```

9. ノードが利用可能な場合は、適切なすべてのテンプレートで新しいエッジサイトをデプロイします。

```
openstack overcloud deploy \
  --stack dcn9 \
  --templates /usr/share/openstack-tripleo-heat-templates/\
  -r ~/dcn9/roles_data.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/dcn-hci.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
  -e ~/dcn9/dcn9-images-env.yaml \
  ....
  -e ~/dcn-common/central-export.yaml \
  -e ~/dcn-common/central_ceph_external.yaml \
  -e ~/dcn9/dcn_ceph_keys.yaml \
  -e ~/dcn9/role-counts.yaml \
  -e ~/dcn9/ceph.yaml \
  -e ~/dcn9/site-name.yaml \
  -e ~/dcn9/tuning.yaml \
  -e ~/dcn9/glance.yaml
```

10. エッジ間の直接通信でロケーションをデプロイしている場合は、各エッジサイトを再デプロイして、ルートを更新し、新しいロケーションとの通信を確立する必要があります。

7.3. 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。

```

sudo -E openstack overcloud export ceph \
--stack dcn0,dcn1 \
--config-download-dir /var/lib/mistral \
--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。

```

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/ceph-ansible/ceph-
  ansible.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

```

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

```

7.3.1. 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

- 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。
- 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。
- 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>
```

7.4. 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。

- エッジサイトをデプロイするためのテンプレートで、**ManageNetworks** パラメーターの値が **false** となるようにしてください。**ManageNetworks** を **false** に設定すると、エッジサイトは中央スタックですでに作成された既存のネットワークを使用します。

```
parameter_defaults:
  ManageNetworks: false
```

- 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。

7.4.1. 仮想 IP 用のコンポーザブルネットワークの作成

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/ceph-ansible/ceph-ansible.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/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. **/var/lib/mistral/<stackname>/ceph-ansible/group_vars/all.yaml** から、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。
2. 選択したスタックに固有のインベントリ **/var/lib/mistral/<stackname>/ceph-ansible/inventory.yaml** で、DistributedComputeHCI ロールホストのリストを見つけ、**storage_ip** の値を3つすべて保存します。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 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
```

8.1. 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、[Deploying Barbican](#) 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9章 GLANCE イメージの NOVA への事前キャッシュ

ローカルの一時的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コンピュート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-hci.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。

9.1. 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. スタック用の Ansible インベントリーファイルを作成します。複数のスタックをコンマ区切りリストで指定して、複数サイトのイメージをキャッシュすることができます。

```
$ source stackrc

$ tripleo-ansible-inventory --plan central,dcn0,dcn1 \
--static-yaml-inventory inventory.yaml
```

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 inventory.yaml \
--extra-vars "@nova_cache_args.yml" \
/usr/share/ansible/tripleo-playbooks/tripleo_nova_image_cache.yml
```

9.2. 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
```

9.3. 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、プロキシとして機能する 1 台のコンピュートノードにダウンロードされます。
- プロキシは、glance イメージをインベントリー内の他のコンピュート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を有効にします。

イメージプロキシは、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を選択します。デフォルトのプロキシは、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
```

9.4. 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行われ、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 **NovalImageCacheTTL** を使用してこの値を制御することができます。

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


Playbook によりコンピュータ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。**NovalmImageCacheTTL** 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10章 DCN への TLS-E の適用

分散コンピューティング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。

10.1. TLS-E を設定した分散コンピューティング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) 分散コンピューティング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
```

手順

分散コンピュートノード (DCN) アーキテクチャーで TLS-e を実装する場合は、従来の **novajoin** 手法ではなく Ansible ベースの **tripleo-ipa** 手法を使用する必要があります。 **tripleo-ipa** を使用した TLS-e のデプロイに関する詳細は、 [Ansible を使用した TLS-e の実装](#) を参照してください。

tripleo-ipa を使用して DCN アーキテクチャーに TLS-e をデプロイするには、以下の手順も完了する必要があります。

1. エッジサイトにストレージをデプロイする場合は、エッジスタック用に変更した tripleo heat テンプレートに以下のパラメーターを追加します。

```
TEMPLATES=/usr/share/openstack-tripleo-heat-templates

resource_registry:
  OS::TripleO::Services::IpaClient:
    ${TEMPLATES}/deployment/ipa/ipaservices-baremetal-ansible.yaml

parameter_defaults:
  EnableEtcInternalTLS: true
```

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。

第11章 外部アクセス用 CEPH キーの作成

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。

11.1. 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
```

11.2. 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 \
--config-download-dir /var/lib/mistral \
--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 \  
--config-download-dir /var/lib/mistral \  
--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/ceph-ansible/ceph-  
ansible.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。

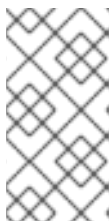
あるいは、**--stores** を **--all-stores True** に置き換えて、すべてのストアにイメージをアップロードすることができます。

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=b795f047a1b10ba0b7c95b43b2a481a59289dc4cf2e49845e60b194a91181
9d3ada03767bbba4143b44c93fd7f66c96c5a621e28dff51d1196dae64974ce240e,
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については、[Image service with multiple stores](#) 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'])
```

- 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。

単一スタックからマルチスタックに移行する機能は、本リリースでは [テクノロジープレビュー](#) であるため、Red Hat では全面的にはサポートしていません。これは、テスト用途にのみご利用いただく機能で、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。

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

- 中央サイトが、`/usr/share/openstack-tripleo-heat-templates/environments` にある `cinder-backup.yaml` 環境ファイルでデプロイされている。詳しくは、[Block Storage backup service deployment](#) を参照してください。
- Block Storage サービス (cinder) CLI が利用できる。
- 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、[外部アクセス用 Ceph キーの作成](#) を参照してください。

手順

- 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** コンテナを再起動して、キーリングがホストからコンテナに正常にコピーされるようにする必要があります。

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)。