Red Hat OpenStack Platform 17.1

コンピュートセルを使用したデプロイメントのスケーリング

マルチセルオーバークラウドの作成および設定に関するガイド

Last Updated: 2024-08-16

Red Hat OpenStack Platform 17.1 コンピュートセルを使用したデプロイメントのスケーリング

マルチセルオーバークラウドの作成および設定に関するガイド

OpenStack Team
rhos-docs@redhat.com

法律上の通知

概要

このガイドでは、クラウド管理者は、大規模な Red Hat OpenStack Platform(RHOSP) デプロイメントのコンピュートノードをグループ化するためにセルを設定および管理するための概念と手順を説明します。

目次

多様性を受け入れるオープンソースの強化

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

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 マルチセルオーバークラウドのデプロイメント

セルを使用して、大規模なデプロイメントのコンピュートノードをグループに分割することができます。それぞれのグループは、メッセージキューおよびインスタンス情報が含まれる専用のデータベースを持ちます。

デフォルトでは、director はすべてのコンピュートノードを単一のセルとしてオーバークラウドをインストールします。このセルには、すべての Compute サービスおよびデータベースならびにすべてのインスタンスおよびインスタンスのメタデータが含まれます。大規模なデプロイメントでは、複数のセルでオーバークラウドをデプロイし、多数のコンピュートノードに対応することができます。新しいオーバークラウドをインストールする際に、またはその後の任意の時に、セルを環境に追加することができます。

マルチセルのデプロイメントでは、それぞれのセルはセル固有の Compute サービスおよびデータベースのスタンドアロンコピーを実行し、そのセル内のインスタンスに関するメタデータだけを保管します。グローバルな情報とセルのマッピングは、グローバルなコントローラーセルに保管されます。このセルは、いずれかのセルに障害が発生した場合のセキュリティーとリカバリーを提供します。

注意

既存のオーバークラウドにセルを追加する場合、デフォルトセルのコンダクターはスーパーコンダクターのロールも果たします。これは、デプロイメント内のセルとのコンダクターの通信およびオーバークラウドのパフォーマンスに悪影響を及ぼします。さらに、デフォルトのセルをオフラインにするとスーパーコンダクターもオフラインになり、オーバークラウドのデプロイメント全体が停止します。したがって、既存のオーバークラウドをスケーリングする場合は、デフォルトのセルにコンピュートノードを追加しないでください。その代わりに、作成する新しいセルにコンピュートノードを追加して、デフォルトのセルをスーパーコンダクターとして機能させます。

マルチセルオーバークラウドを作成するには、以下のタスクを実行する必要があります。

1. 複数のセルを処理するようにオーバークラウドを設定およびデプロイします。
2. デプロイメントに必要な新しいセルを作成およびプロビジョニングする。
3. それぞれのセルにコンピュートノードを追加する。
4. 各コンピュートセルをアベイラビリティーゾーンに追加します。

1.1. 前提条件

- 必要な数のコントローラーノードと共に基本的なオーバークラウドをデプロイしている。

1.2. グローバルなコンポーネントおよびサービス

以下のコンポーネントは、コンピュートセルの数にかかわらず、オーバークラウドごとに一度コントローラーセルにデプロイされます。

Compute API
ユーザーに外部 REST API を提供します。

Compute スケジューラー
インスタンスを割り当てるコンピュートノードを決定します。

Placement サービス
Compute リソースを監視し、インスタンスに割り当てます。

API データベース

Compute API および Compute スケジューラーサービスがインスタンスの場所に関する情報を追跡するのに使用されます。また、ビルドされているがスケジュールされていないインスタンスの一時的な場所を提供します。
マルチセルのデプロイメントでは、このデータベースには各セルのデータベース接続を指定するセルのマッピングも含まれます。

cell0 データベース

スケジュールに失敗したインスタンスに関する情報を保管する専用のデータベース

スーパーコンダクター

このサービスはマルチセルのデプロイメントにのみ存在し、グローバルなサービスと各コンピュートセル間の調整を行います。また、このサービスはスケジュールに失敗したインスタンスの情報を **cell0** データベースに送信します。

1.3. セル固有のコンポーネントおよびサービス

以下のコンポーネントは、それぞれのコンピュートセルにデプロイされます。

セルデータベース

インスタンスに関するほとんどの情報が含まれます。グローバル API、コンダクター、および Compute サービスによって使用されます。

コンダクター

データベースのクエリーおよび長時間実行されているグローバルなサービスからのタスクの調整を行い、コンピュートノードをデータベースへの直接アクセスから隔離します。

メッセージキュー

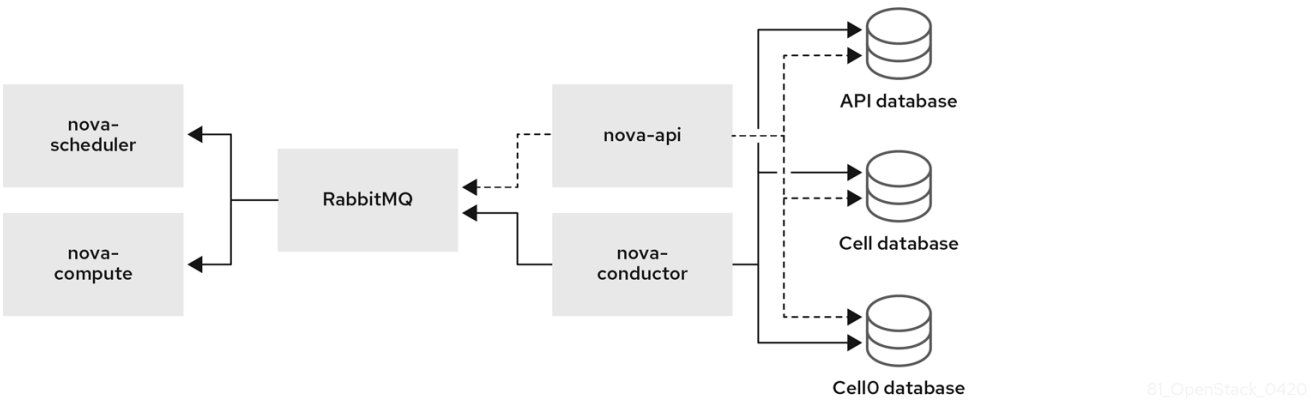
セル内の各サービス間の通信、およびグローバルなサービスとの通信のために、すべてのサービスによって使用されるメッセージングサービス

1.4. セルデプロイメントのアーキテクチャー

director がインストールするデフォルトのオーバークラウドには、すべてのコンピュートノードが単一のセルとして含まれます。以下のアーキテクチャー図に示すように、セルをさらに追加してオーバークラウドをスケーリングすることができます。

単一セルデプロイメントのアーキテクチャー

デフォルトの単一セルオーバークラウドの基本的な設定および対話の例を以下の図に示します。

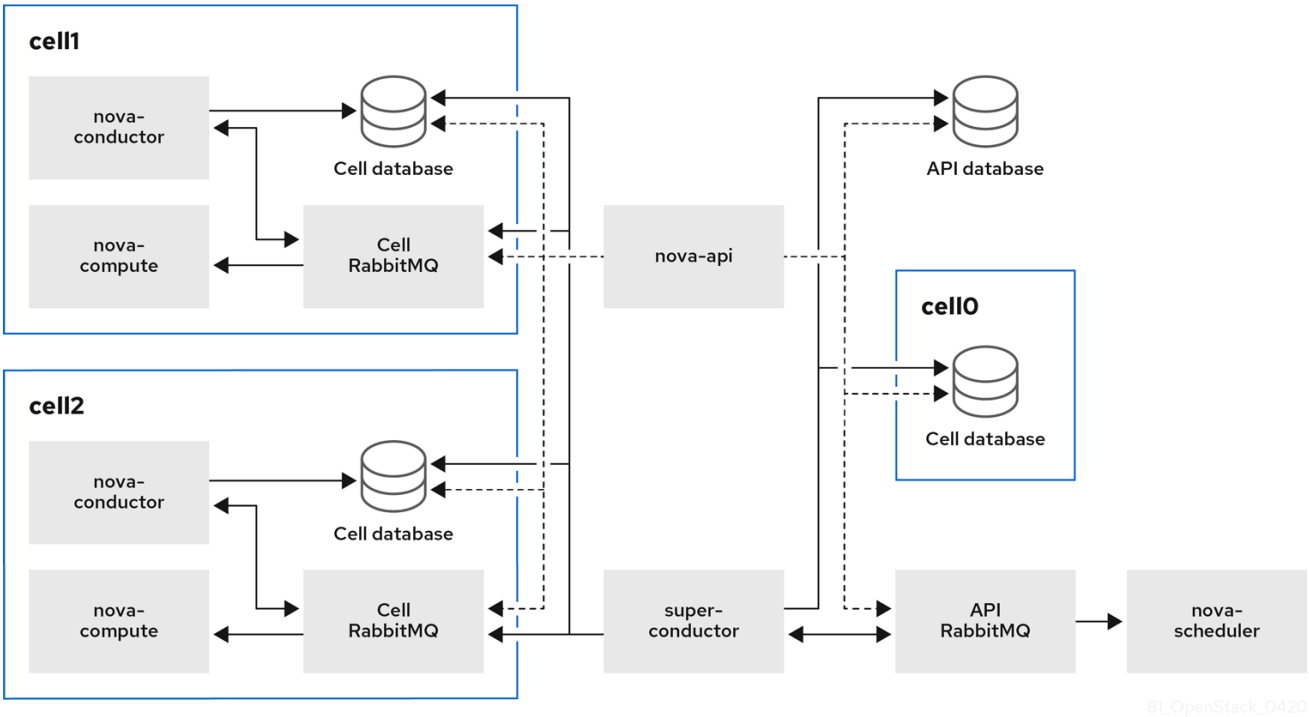


このデプロイメントでは、すべてのサービスが 1 つのコンダクターを使用して Compute API とコンピュートノード間の通信を行うように設定されます。また、1 つのデータベースに動作中の全インスタンスのデータが保管されます。

小規模なデプロイメントであれば、この設定で十分です。ただし、グローバルな API サービスまたはデータベースに障害が発生すると、高可用性の設定かどうかにかかわらず、Compute デプロイメント全体が情報を送受信できなくなります。

マルチセルデプロイメントのアーキテクチャー

カスタムのマルチセルオーバークラウドの基本的な設定および対話の例を以下の図に示します。



このデプロイメントでは、コンピュートノードは複数のセルに分割され、それぞれのセルは専用のコンダクター、データベース、およびメッセージキューを持ちます。グローバルなサービスは、スーパーコンダクターを使用して各セルと通信を行います。また、グローバルデータベースにはオーバークラウド全体に必要な情報だけが含まれます。

セルレベルのサービスは、グローバルなサービスに直接アクセスすることはできません。この分離により、セキュリティーが向上し、セルに障害が発生した場合のフェイルセーフ機能が得られます。

重要

"default" という名前の最初のセルでは、Compute サービスを実行しないでください。その代わりに、コンピュートノードが含まれる各新規セルを個別にデプロイします。

1.5. マルチセルのデプロイメントで考慮すべき事項

マルチセルのデプロイメントにおける最大のコンピュートノード数

最大のコンピュートノード数は、全セルの合計で 500 です。

セル間のインスタンスの移行

あるセル内のホストから別のセル内のホストにインスタンスを移行する操作は、サポートされていません。この制限は、以下の操作に影響を及ぼします。

- コールドマイグレーション
- ライブマイグレーション
- 復元
- サイズ変更
- 退避

サービスクォータ

Compute サービスのクォータは、データベースで静的に計算されるのではなく、リソースが消費されるそれぞれの時点で動的に計算されます。マルチセルのデプロイメントでは、アクセス不能なセルは使用状況に関する情報をリアルタイムで提供することができないため、セルが再びアクセス可能になるとクォータを超過する可能性があります。
Placement サービスおよび API データベースを使用して、障害の発生したセルまたはアクセス不能なセルに対応するようにクォータの算出を設定することができます。

API データベース

Compute API データベースは常にグローバルで (すべてのセルが対象)、セルごとに複製することはできません。

コンソールプロキシー

コンソールトークンの承認はセルのデータベースに保管されるため、セルごとにコンソールプロキシーを設定する必要があります。各コンソールプロキシーサーバーは、対応するセルのデータベースの **database.connection** の情報にアクセスする必要があります。

Compute メタデータ API

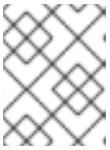
複数のセル環境のセルすべてに同じネットワークを使用する場合には、セル間をブリッジできるように Compute メタデータ API をグローバルに実行する必要があります。Compute メタデータ API がグローバルに実行される場合には、**api_database.connection** 情報へのアクセスが必要になります。
ルーティング対応のネットワークで複数のセル環境をデプロイする場合は、各セルで Compute メタデータ API を個別に実行して、パフォーマンスとデータの分離を改善する必要があります。
Compute メタデータ API が各セルで実行される場合、**neutron -metadata-agent** サービスは対応する **nova-api-metadata** サービスをポイントする必要があります。

パラメーター **NovaLocalMetadataPerCell** を使用して、Compute メタデータ API が実行される場所を制御します。

第2章 同じネットワークを使用するマルチセル環境の設定およびデプロイ

同じネットワークを使用して複数のセルを処理するように Red Hat OpenStack Platform (RHOSP) デプロイメントを設定するには、次のタスクを実行する必要があります。

1. オーバークラウドスタックのコントロールプレーンからパラメーター情報を抽出します。
2. セルのロールファイルを作成します。セルのコンピュートノードのデフォルトの **Compute** ロールと、セルコントローラーノード専用の **CellController** ロールを使用することができます。また、各セルスタックのカスタムロールなど、マルチセル環境で使用するカスタムロールを作成することもできます。カスタムロールの作成に関する詳しい情報は、オーバークラウドの高度なカスタマイズのコンポーザブルサービスとカスタムロールを参照してください。
3. **CellController** ロールのホストを指定します。



注記

マルチセル環境用のカスタムロールを作成した場合は、カスタムロールのホストも指定する必要があります。

4. 各セルを設定します。
5. それぞれのセルスタックをデプロイします。

2.1. オーバークラウドスタックコントロールプレーンからのパラメーター情報の抽出

基本的なオーバークラウドスタックの 1 つ (**default** という名前の最初のセルからパラメーター情報を抽出する)

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. **stackrc** ファイルを取得します。

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

3. オーバークラウドスタックの **default** セルから、マルチセルデプロイメントの新しい共通環境ファイルに、セルの設定およびパスワード情報をエクスポートします。

```
(undercloud)$ openstack overcloud cell export --control-plane-stack overcloud \
 -f --output-file common/default_cell_export.yaml \
 --working-dir /home/stack/overcloud-deploy/overcloud/
```

このコマンドは、**EndpointMap**、**HostsEntry**、**AllNodesConfig**、**GlobalConfig** パラメーター、およびパスワード情報を共通の環境ファイルにエクスポートします。

ヒント

環境ファイルがすでに存在する場合は、**--force-overwrite** または **-f** オプションを指定してコマンドを実行します。

2.2. セルのロールファイルの作成

スタックが同じネットワークを使用し、カスタムロールが必要ない場合、すべてのセルスタックで共通のセルロールファイルを作成することができます。

手順

- **Compute** ロールおよび **CellController** ロールが含まれる **cell_roles_data.yaml** という名前の新しいロールデータファイルを生成します。

```
(undercloud)$ openstack overcloud roles generate \
  --roles-path /usr/share/openstack-tripleo-heat-templates/roles \
  -o common/cell_roles_data.yaml Compute CellController
```

2.3. CELLCONTROLLER ロールのホストの指定

CellController ロールのベアメタルノードを指定するには、**CellController** ロールのノードにタグを付けるリソースクラスを使用してベアメタルノードを設定する必要があります。

ヒント

マルチセル環境用のカスタムロールを作成した場合は、この手順に従ってカスタムロールのリソースクラスを設定できます。設定するには、セルコントローラー名をカスタムロールの名前に置き換えます。

注記

以下の手順は、まだプロビジョニングされていない新しいオーバークラウドノードに適用されます。すでにプロビジョニングされている既存のオーバークラウドノードにリソースクラスを割り当てるには、オーバークラウドをスケールダウンしてノードのプロビジョニングを解除してから、オーバークラウドをスケールアップして、新しいリソースクラスの割り当てでノードを再プロビジョニングします。詳細は、オーバークラウドノードのスケーリング を参照してください。

手順

1. **CellController** ロールのベアメタルノードをノード定義テンプレート (**node.json** または **node.yaml)** に追加して、ベアメタルノードを登録します。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 オーバークラウドのノードの登録 を参照してください。

2. ノードのハードウェアを検査します。

```
(undercloud)$ openstack overcloud node introspect \
  --all-manageable --provide
```

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 ベアメタルノードハードウェアのインベントリーの作成 を参照してください。

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. セルコントローラーとして指定する各ベアメタルノードに、カスタムセルコントローラーリソースクラスのタグを付けます。

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

- **<node>** は、ベアメタルノードの名前または UUID に置き換えます。

5. **CellController** ロールをノード定義ファイル **overcloud-baremetal-deploy.yaml** に追加し、予測ノード配置、リソースクラス、ネットワークトポロジー、またはノードに割り当てるその他の属性を定義します。

```
- name: Controller
  count: 3
- name: Compute
  count: 3
  defaults:
    network_config:
      template: /home/stack/templates/nic-config/<cell_topology_file>
  instances:
  - hostname: cell1-compute-%index%
    name: computecell1
  - hostname: cell1-compute-%index%
    name: computecell2
  - hostname: cell1-compute-%index%
    name: computecell3
- name: CellController
  count: 1
  defaults:
    resource_class: baremetal.CELL-CONTROLLER
    network_config:
      template: /home/stack/templates/nic-config/<role_topology_file>
  instances:
  - hostname: cell1-cellcontroller-%index%
    name: cellcontroller
```

- **<cell_topology_file>** は、セルスタックに使用するネットワークトポロジーファイルの名前 (**compute.j2** など) に置き換えます。

- **<role_topology_file>** は、**CellController** ロールに使用するネットワークトポロジーファイルの名前 (**cell_controller_net_top.j2** など) に置き換えます。
  既存のネットワークトポロジーを再利用することも、ロールまたはセル用の新しいカスタムネットワークインターフェイステンプレートを作成することもできます。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 カスタムネットワークインターフェイステンプレート を参照してください。デフォルトのネットワーク定義設定を使用するには、ロール定義に **network_config** を含めないでください。

ノード定義ファイルでノード属性を設定するために使用できるプロパティーについて詳しくは、ベアメタルノードのプロビジョニング属性 を参照してください。ノード定義ファイルの例は、 ノード定義ファイルの例 を参照してください。

6. ロールの新しいノードをプロビジョニングします。

```
(undercloud)$ openstack overcloud node provision \
[--stack <stack>] \
```

```
[--network-config \]
--output <deployment_file> \
/home/stack/templates/overcloud-baremetal-deploy.yaml
```

- オプション: **<stack>** をベアメタルノードがプロビジョニングされるスタックの名前に置き換えます。デフォルトは **overcloud** です。
- オプション: **--network-config** オプションの引数を含めて、Ansible Playbook **cli-overcloud-node-network-config.yaml** にネットワーク定義を提供します。**network_config** プロパティーを使用してノード定義ファイルにネットワーク定義を定義していない場合は、デフォルトのネットワーク定義が使用されます。
- **<deployment_file>** は、デプロイメントコマンドに含めるために生成する heat 環境ファイルの名前に置き換えます (例 **:/home/stack/templates/overcloud-baremetal-deployed.yaml)**。

7. 別のターミナルでプロビジョニングの進捗をモニタリングします。プロビジョニングが成功すると、ノードの状態が **available** から **active** に変わります。

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

8. **--network-config** オプションを指定せずにプロビジョニングコマンドを実行した場合は、**network-environment.yaml** ファイルで **<Role>NetworkConfigTemplate** パラメーターを設定して、NIC テンプレートファイルを指すようにします。

```
parameter_defaults:
  ComputeNetworkConfigTemplate: /home/stack/templates/nic-configs/compute.j2
  CellControllerNetworkConfigTemplate: /home/stack/templates/nic-
configs/<role_topology_file>
  ControllerNetworkConfigTemplate: /home/stack/templates/nic-configs/controller.j2
```

- **<role_topology_file>** は、**CellController** ロールのネットワークトポロジーを含むファイルの名前 (**cell_controller_net_top.j2** など) に置き換えます。デフォルトのネットワークトポロジーを使用するには、**compute.j2** に設定します。

2.4. 同じネットワークを使用する各セルスタックの設定およびデプロイ

デプロイメント内の追加セルとしてセルを識別するために、各セルスタックを設定する必要があります。

手順

1. 新しいセル用に新しいディレクトリーを作成します。

```
(undercloud)$ mkdir cells
```

2. **cells** 固有のパラメーター (例: **/cells/cell1.yaml**) の各追加セルに、新しい環境ファイルを作成します。
3. 各環境ファイルに以下のパラメーターを追加して、デプロイメント内の各セルのパラメーター値を更新します。

```
parameter_defaults:
  # Disable network creation in order to use the `network_data.yaml` file from the overcloud
```

```
stack,
  # and create ports for the nodes in the separate stacks on the existing networks.
  ManageNetworks: false

  # Specify that this is an additional cell
  NovaAdditionalCell: True

  # The DNS names for the VIPs for the cell
  CloudName: cell1.ooo.test
  CloudNameInternal: cell1.internalapi.ooo.test
  CloudNameStorage: cell1.storage.ooo.test
  CloudNameStorageManagement: cell1.storagemgmt.ooo.test
  CloudNameCtlplane: cell1.ctlplane.ooo.test
```

4. その他の環境ファイルと共にこの環境ファイルをスタックに追加して、セルスタックをデプロイします。

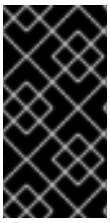
```
(undercloud)$ openstack overcloud deploy --templates \
 --stack cell1 \
 -e [your environment files] \
 -e /home/stack/templates/overcloud-baremetal-deployed.yaml \
 -e /home/stack/templates/overcloud-networks-deployed.yaml \
 -e /home/stack/templates/overcloud-vip-deployed.yaml \
 -r $HOME/common/cell_roles_data.yaml \
 -e $HOME/common/default_cell_export.yaml \
 -e $HOME/cells/cell1.yaml
```

すべてのセルスタックがデプロイされるまで、それぞれのセルスタックでこの手順を繰り返します。

2.5. 次のステップ

- Compute サービス内のセルの作成および管理

第3章 ルーティング対応のネットワークを使用するマルチセル環境の設定およびデプロイ



重要

このセクションの以下のコンテンツは、今回のリリースではテクノロジープレビューとしての利用となるため、Red Hat によって完全にはサポートされません。これは、テスト用途にのみご利用いただく機能です。実稼働環境にはデプロイしないでください。詳細は、テクノロジープレビューを参照してください。

ルーティング対応のネットワークで複数のセルを処理するように Red Hat OpenStack(RHOSP) デプロイメントを設定するには、以下のタスクを実行する必要があります。

1. オーバークラウドスタック上のセルネットワークルーティング用にコントロールプレーンを準備します。
2. オーバークラウドスタックのコントロールプレーンからパラメーター情報を抽出します。
3. セルスタックでセルネットワークルーティングを設定します。
4. それぞれのスタックにセルロールファイルを作成します。デフォルトの **Compute** ロールをセル内のコンピュートノードのベースとして使用し、専用の **CellController** ロールをセルコントローラーノードのベースとして使用することができます。マルチセル環境で使用するカスタムロールを作成することもできます。カスタムロールの作成に関する詳しい情報は、オーバークラウドの高度なカスタマイズのコンポーザブルサービスとカスタムロールを参照してください。
5. 作成するカスタムロールごとにホストを指定します。

注記

この手順は、単一のコントロールプレーンネットワークを持つ環境用のものです。スパイン/リーフ型ネットワーク環境など、環境に複数のコントロールプレーンネットワークがある場合は、各リーフにノードをタグ付けできるように、各リーフネットワークのロールごとにホストを指定する必要もあります。詳細は、リーフノードのロールの指定 を参照してください。

6. 各セルを設定します。
7. それぞれのセルスタックをデプロイします。

3.1. 前提条件

- アンダークラウドがルーティングされたネットワーク用に設定されていること。詳しくは、アンダークラウドでのルーティング対応のスパイン/リーフの設定を参照してください。

3.2. セルネットワークルーティング用のコントロールプレーンおよびデフォルトのセルの準備

オーバークラウドスタックの overcloud スタックでルートを設定し、セルと通信する必要があります。そのためには、メインのスタック内の全ネットワークおよびサブネットを定義するネットワークデータファイルを作成し、このファイルを使用して、オーバークラウドスタックとセルスタックの両方をデプ

ロイします。

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. **stackrc** ファイルを取得します。

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

3. 共通のスタック設定用に新規ディレクトリーを作成します。

```
(undercloud)$ mkdir common
```

4. デフォルトの **network_data_subnets_routed.yaml** ファイルを **common** ディレクトリーにコピーし、オーバークラウドスタック用のコンポーザブルネットワークを追加します。

```
(undercloud)$ cp /usr/share/openstack-tripleo-heat-
templates/network_data_subnets_routed.yaml
~/common/network_data_routed_multi_cell.yaml
```

コンポーザブルネットワークの詳細は、**director のインストールと使用方法** ガイドの コンポーザブルネットワーク を参照してください。

5. ネットワーク用の **/common/network_data_routed_multi_cell.yaml** の設定を更新し、簡単に識別できるようにセルのサブネット名を更新します。たとえば、**internal_api_leaf1** を **internal_api_cell1**に変更します。
6. 各ロールの NIC テンプレートのインターフェイスに、**<network_name>InterfaceRoutes** が含まれるようにします。以下に例を示します。

```
-
  type: vlan
  vlan_id:
    get_param: InternalApiNetworkVlanID
  addresses:
  -
    ip_netmask:
      get_param: InternalApiIpSubnet
  routes:
    get_param: InternalApiInterfaceRoutes
```

7. その他の環境ファイルと共に **network_data_routed_multi_cell.yaml** ファイルをオーバークラウドスタックに追加して、オーバークラウドをデプロイします。

```
(undercloud)$ openstack overcloud deploy --templates \
 --stack overcloud \
 -e [your environment files]
 -n /home/stack/common/network_data_routed_multi_cell.yaml \
 -e /home/stack/templates/overcloud-baremetal-deployed.yaml \
 -e /home/stack/templates/overcloud-networks-deployed.yaml \
 -e /home/stack/templates/overcloud-vip-deployed.yaml
```

3.3. オーバークラウドスタックコントロールプレーンからのパラメーター情報の抽出

基本的なオーバークラウドスタックの 1 つ (**default** という名前の最初のセルからパラメーター情報を抽出する)

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. **stackrc** ファイルを取得します。

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

3. オーバークラウドスタックの **default** セルから、マルチセルデプロイメントの新しい共通環境ファイルに、セルの設定およびパスワード情報をエクスポートします。

```
(undercloud)$ openstack overcloud cell export --control-plane-stack overcloud \
 -f --output-file common/default_cell_export.yaml \
 --working-dir /home/stack/overcloud-deploy/overcloud/
```

このコマンドは、**EndpointMap**、**HostsEntry**、**AllNodesConfig**、**GlobalConfig** パラメーター、およびパスワード情報を共通の環境ファイルにエクスポートします。

ヒント

環境ファイルがすでに存在する場合は、**--force-overwrite** または **-f** オプションを指定してコマンドを実行します。

3.4. ルーティング対応のネットワーク用のセルロールファイルの作成

それぞれのスタックが異なるネットワークを使用する場合は、カスタムセルロールが含まれるセルスタックごとにセルロールファイルを作成します。

注記

それぞれのカスタムロール用にフレーバーを作成する必要があります。詳細は、Designating hosts for cell roles を参照してください。

手順

1. cell スタックに必要なその他のロールに加えて、**CellController** ロールが含まれる新しいロールデータファイルを生成します。以下の例では、**CellController** ロールおよび **Compute** ロールが含まれるロールデータファイル **cell1_roles_data.yaml** を生成します。

```
(undercloud)$ openstack overcloud roles generate \
  --roles-path /usr/share/openstack-tripleo-heat-templates/roles \
  -o cell1/cell1_roles_data.yaml \
  Compute:ComputeCell1 \
  CellController:CellControllerCell1
```

2. 新規セルのロールファイルの各ロール定義に **HostnameFormatDefault** を追加します。

```
- name: ComputeCell1
  ...
  HostnameFormatDefault: '%stackname%-compute-cell1-%index%'
  ServicesDefault:
  ...
  networks:
  ...
- name: CellControllerCell1
  ...
  HostnameFormatDefault: '%stackname%-cellcontrol-cell1-%index%'
  ServicesDefault:
  ...
  networks:
  ...
```

3. Networking サービス (neutron) DHCP およびメタデータエージェントがない場合は、**ComputeCell1** ロールおよび **CellControllerCell1** ロールに追加します。

```
- name: ComputeCell1
  ...
  HostnameFormatDefault: '%stackname%-compute-cell1-%index%'
  ServicesDefault:
  - OS::TripleO::Services::NeutronDhcpAgent
  - OS::TripleO::Services::NeutronMetadataAgent
  ...
  networks:
  ...
- name: CellControllerCell1
  ...
  HostnameFormatDefault: '%stackname%-cellcontrol-cell1-%index%'
  ServicesDefault:
  - OS::TripleO::Services::NeutronDhcpAgent
  - OS::TripleO::Services::NeutronMetadataAgent
  ...
  networks:
  ...
```

4. **network_data_routed_multi_cell.yaml** に設定したサブネットを **ComputeCell1** および **CellControllerCell1** ロールに追加します。

```
- name: ComputeCell1
  ...
  networks:
    InternalApi:
      subnet: internal_api_subnet_cell1
    Tenant:
      subnet: tenant_subnet_cell1
    Storage:
      subnet: storage_subnet_cell1
...
- name: CellControllerCell1
  ...
  networks:
    External:
      subnet: external_subnet
```

```
InternalApi:
  subnet: internal_api_subnet_cell1
Storage:
  subnet: storage_subnet_cell1
StorageMgmt:
  subnet: storage_mgmt_subnet_cell1
Tenant:
  subnet: tenant_subnet_cell1
```

3.5. セルロールのホストの指定

セルロールのベアメタルノードを指定するには、セルロールのノードにタグを付けるために使用するリソースクラスを使用してベアメタルノードを設定する必要があります。**cellcontrollercell1** ロールのベアメタルリソースクラスを作成するには、次の手順を実行します。セルコントローラー名をカスタムロール名に置き換えて、それぞれのカスタムロールに対してこの手順を繰り返します。

注記

以下の手順は、まだプロビジョニングされていない新しいオーバークラウドノードに適用されます。すでにプロビジョニングされている既存のオーバークラウドノードにリソースクラスを割り当てるには、オーバークラウドをスケールダウンしてノードのプロビジョニングを解除してから、オーバークラウドをスケールアップして、新しいリソースクラスの割り当てでノードを再プロビジョニングします。詳細は、オーバークラウドノードのスケーリング を参照してください。

手順

1. **cellcontrollercell1** ロールのベアメタルノードをノード定義テンプレート (**node.json** または **node.yaml**) に追加して、ベアメタルノードを登録します。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 オーバークラウドのノードの登録 を参照してください。

2. ノードのハードウェアを検査します。

```
(undercloud)$ openstack overcloud node introspect \
 --all-manageable --provide
```

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 ベアメタルノードハードウェアのインベントリーの作成 を参照してください。

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. セルコントローラーとして指定する各ベアメタルノードに、カスタムセルコントローラーリソースクラスのタグを付けます。

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

- **<node>** は、ベアメタルノードの名前または UUID に置き換えます。

5. **cellcontrollercell1** ロールをノード定義ファイル **overcloud-baremetal-deploy.yaml** に追加し、予測ノード配置、リソースクラス、ネットワークトポロジー、またはノードに割り当てるその他の属性を定義します。

```
- name: cellcontrollercell1
  count: 1
  defaults:
    resource_class: baremetal.CELL1-CONTROLLER
    network_config:
      template: /home/stack/templates/nic-config/<role_topology_file>
  instances:
  - hostname: cell1-cellcontroller-%index%
    name: cell1controller
```

   - **<role_topology_file>** は、**cellcontrollercell1** ロールに使用するネットワークトポロジーファイルの名前 (**cell1_controller_net_top.j2** など) に置き換えます。既存のネットワークトポロジーを再利用することも、ロールまたはセル用の新しいカスタムネットワークインターフェイステンプレートを作成することもできます。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 カスタムネットワークインターフェイステンプレート を参照してください。デフォルトのネットワーク定義設定を使用するには、ロール定義に **network_config** を含めないでください。
     ノード定義ファイルでノード属性を設定するために使用できるプロパティーについて詳しくは、ベアメタルノードのプロビジョニング属性 を参照してください。ノード定義ファイルの例は、ノード定義ファイルの例 を参照してください。

6. ロールの新しいノードをプロビジョニングします。

```
(undercloud)$ openstack overcloud node provision \
[--stack <stack>] \
[--network-config \]
--output <deployment_file> \
/home/stack/templates/overcloud-baremetal-deploy.yaml
```

   - オプション: **<stack>** をベアメタルノードがプロビジョニングされるスタックの名前に置き換えます。デフォルトは **overcloud** です。

   - オプション: **--network-config** オプションの引数を含めて、Ansible Playbook **cli-overcloud-node-network-config.yaml** にネットワーク定義を提供します。**network_config** プロパティーを使用してノード定義ファイルにネットワーク定義を定義していない場合は、デフォルトのネットワーク定義が使用されます。

   - **<deployment_file>** は、デプロイメントコマンドに含めるために生成する heat 環境ファイルの名前に置き換えます (例 **:/home/stack/templates/overcloud-baremetal-deployed.yaml)**。

7. 別のターミナルでプロビジョニングの進捗をモニタリングします。プロビジョニングが成功すると、ノードの状態が **available** から **active** に変わります。

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

8. **--network-config** オプションを指定せずにプロビジョニングコマンドを実行した場合は、**network-environment.yaml** ファイルで **<Role>NetworkConfigTemplate** パラメーターを設定して、NIC テンプレートファイルを指すようにします。

```
parameter_defaults:
```

```
  ComputeNetworkConfigTemplate: /home/stack/templates/nic-configs/compute.j2
  CellControllerCell1NetworkConfigTemplate: /home/stack/templates/nic-configs/<role_topology_file>
  ControllerNetworkConfigTemplate: /home/stack/templates/nic-configs/controller.j2
```

- **<role_topology_file>** は、**cellcontrollercell1** ロールのネットワークトポロジーを含むファイルの名前 (**cell1_controller_net_top.j2** など) に置き換えます。デフォルトのネットワークトポロジーを使用するには、**controller.j2** に設定します。

3.6. ルーティング対応のネットワークを使用した各セルスタックの設定およびデプロイ

セルスタック (**cell1**)1 つを設定するには、以下の手順を実施します。セルスタックがすべてデプロイされるまで、デプロイする追加のセルスタックで手順を繰り返します。

手順

1. セル固有のパラメーター用に、セル固有のパラメーターの追加セル用に新たな環境ファイル (例: **/home/stack/cell1/cell1.yaml**) を作成します。
2. 環境ファイルに以下のパラメーターを追加します。

```
resource_registry:
  OS::TripleO::CellControllerCell1::Net::SoftwareConfig: /home/stack/templates/nic-configs/cellcontroller.yaml
  OS::TripleO::ComputeCell1::Net::SoftwareConfig: /home/stack/templates/nic-configs/compute.yaml

parameter_defaults:
  # Specify that this is an additional cell
  NovaAdditionalCell: True

  # Enable local metadata API for each cell
  NovaLocalMetadataPerCell: True

  #Disable network creation in order to use the `network_data.yaml` file from the overcloud stack,
  # and create ports for the nodes in the separate stacks on the existing networks.
  ManageNetworks: false

  # Specify that this is an additional cell
  NovaAdditionalCell: True

  # The DNS names for the VIPs for the cell
  CloudDomain: redhat.local
  CloudName: cell1.redhat.local
  CloudNameInternal: cell1.internalapi.redhat.local
  CloudNameStorage: cell1.storage.redhat.local
  CloudNameStorageManagement: cell1.storagemgmt.redhat.local
  CloudNameCtlplane: cell1.ctlplane.redhat.local
```

3. グローバルコントローラーではなく、各セルで Compute メタデータ API を実行するには、セル環境ファイルに以下のパラメーターを追加します。

```
parameter_defaults:
  NovaLocalMetadataPerCell: True
```

4. セルの仮想 IP アドレス (VIP) 情報をセル環境ファイルに追加します。

```
parameter_defaults:
  ...
  VipSubnetMap:
    InternalApi: internal_api_cell1
    Storage: storage_cell1
    StorageMgmt: storage_mgmt_cell1
    External: external_subnet
```

これにより、セルのコントローラーノードが接続されている L2 ネットワークセグメントに関連付けられた仮想 IP アドレスが作成されます。

5. その他の環境ファイルと共にこの環境ファイルをスタックに追加して、セルスタックをデプロイします。

```
(undercloud)$ openstack overcloud deploy --templates \
 --stack cell1 \
 -e [your environment files] \
 -e /home/stack/templates/overcloud-baremetal-deployed.yaml \
 -e /home/stack/templates/overcloud-networks-deployed.yaml \
 -e /home/stack/templates/overcloud-vip-deployed.yaml \
 -r /home/stack/cell1/cell1_roles_data.yaml \
 -n /home/stack/common/network_data_spine_leaf.yaml \
 -e /home/stack/common/default_cell_export.yaml \
 -e /home/stack/cell1/cell1.yaml
```

3.7. デプロイメント後の新規セルサブネットの追加

マルチセル環境をデプロイした後に、オーバークラウドスタックに新しいセルサブネットを追加するには、**NetworkDeploymentActions** の値を更新して **'UPDATE'** を追加する必要があります。

手順

1. オーバークラウドスタックの環境ファイルに以下の設定を追加して、新しいセルサブネットでネットワーク設定を更新します。

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

2. 新しいセルサブネットの設定を **/common/network_data_routed_multi_cell.yaml** に追加します。

3. オーバークラウドスタックをデプロイします。

```
(undercloud)$ openstack overcloud deploy --templates \
 --stack overcloud \
 -n /home/stack/common/network_data_routed_multi_cell.yaml \
 -e [your environment files]
```

4. オプション: 次のデプロイメントのために **NetworkDeploymentActions** をデフォルトにリセットします。

```
parameter_defaults:
  NetworkDeploymentActions: ['CREATE']
```

3.8. 次のステップ

- Compute サービス内のセルの作成および管理

第4章 COMPUTE サービス内のセルの作成および管理

セルスタックでオーバークラウドをデプロイしたら、Compute サービス内にセルを作成する必要があります。Compute サービス内にセルを作成するには、グローバル API データベースにセルとメッセージキューのマッピングのエントリーを作成します。次に、いずれかのコントローラーノードでセルホストに検出を実行することで、セルにコンピュートノードを追加できます。

セルを作成するには、以下のタスクを実行する必要があります。

1. **nova-manage** ユーティリティーを使用して、グローバル API データベースにセルおよびメッセージキューのマッピングレコードを作成します。
2. それぞれのセルにコンピュートノードを追加する。
3. 各セルにアベイラビリティーゾーンを作成します。
4. 各セルのすべてのコンピュートノードを、セルのアベイラビリティーゾーンに追加します。

4.1. 前提条件

- 複数のセルでオーバークラウドを設定およびデプロイしている。

4.2. COMPUTE サービス内でのセルの作成

新しいセルスタックでオーバークラウドをデプロイしたら、Compute サービス内にセルを作成する必要があります。Compute サービス内にセルを作成するには、グローバル API データベースにセルとメッセージキューのマッピングのエントリーを作成します。注記: このプロセスは、作成して起動するセルごとに繰り返す必要があります。Ansible Playbook で手順を自動化することができます。Ansible Playbook の例については、OpenStack コミュニティーのドキュメントの Create the cell and discover compute nodes (ansible playbook) セクションを参照してください。コミュニティーのドキュメントはそのままの状態で提供され、公式にはサポートされません。

手順

1. コントロールプレーンおよびセルコントローラーの IP アドレスを取得します。

```
$ CTRL_IP=$(ansible-inventory -i /home/stack/overcloud-deploy/overcloud/config-download/overcloud/tripleo-ansible-inventory.yaml --host <controller_node> | jq -r .ctlplane_ip)
$ CELL_CTRL_IP=$(ansible-inventory -i /home/stack/overcloud-deploy/cell1/config-download/cell1/tripleo-ansible-inventory.yaml --host <cell_controller_node> | jq -r .ctlplane_ip)
```

   - **<controller_node>** は、コントローラーノードの名前 (**controller-0** など) に置き換えます。
   - **<cell_controller_node>** は、セルコントローラーノードの名前 (**cell1-controller-0** など) に置き換えます。

2. すべてのコントローラーノードにセル情報を追加します。この情報は、アンダークラウドからセルのエンドポイントに接続するのに使用されます。以下の例では、接頭辞 **cell1** を使用してセルシステムだけを指定し、コントローラーシステムを除外しています。

```
(undercloud)$ CELL_INTERNALAPI_INFO=$(ssh tripleo-admin@${CELL_CTRL_IP} \
```

```
 egrep cell1.*\.internalapi /etc/hosts)
(undercloud)$ ansible -i /home/stack/overcloud-deploy/overcloud/config-
download/overcloud/tripleo-ansible-inventory.yaml \
 Controller -b -m lineinfile -a "dest=/etc/hosts line=\"$CELL_INTERNALAPI_INFO\""
```

3. **transport_url** パラメーターからコントローラーセルのメッセージキューのエンドポイントを、**database.connection** パラメーターからコントローラーセルのデータベース接続を、それぞれ取得します。

```
(undercloud)$ CELL_TRANSPORT_URL=$(ssh tripleo-admin@${CELL_CTRL_IP} \
 sudo crudini --get /var/lib/config-data/puppet-generated/nova/etc/nova/nova.conf \
 DEFAULT transport_url)
(undercloud)$ CELL_MYSQL_VIP=$(ssh tripleo-admin@${CELL_CTRL_IP} \
 sudo crudini --get /var/lib/config-data/puppet-generated/nova/etc/nova/nova.conf \
 database connection | awk -F[@/] '{print $4}')
```

4. グローバルコントローラーノードのいずれかにログインして、セルを作成します。

```
$ ssh tripleo-admin@${CTRL_IP} sudo podman \
 exec -i -u root nova_api \
 nova-manage cell_v2 create_cell --name cell1 \
 --database_connection "{scheme}://{username}:{password}@$CELL_MYSQL_VIP/nova?
{query}" \
 --transport-url "$CELL_TRANSPORT_URL"
```

5. セルが作成され、セルのリストに表示されることを確認します。

```
$ ssh tripleo-admin@${CTRL_IP} sudo podman \
 exec -i -u root nova_api \
 nova-manage cell_v2 list_cells --verbose
```

6. コントローラーノードで Compute サービスを再起動します。

```
$ ansible -i /usr/bin/tripleo-ansible-inventory Controller -b -a \
"systemctl restart tripleo_nova_api tripleo_nova_conductor tripleo_nova_scheduler"
```

7. セルコントローラーサービスがプロビジョニングされていることを確認します。

```
(overcloud)$ openstack compute service list -c Binary -c Host -c Status -c State
+----------------+-------------------------+---------+-------+
| Binary         | Host                    | Status  | State |
+----------------+-------------------------+---------+-------+
| nova-conductor | controller-0.ostest     | enabled | up    |
| nova-scheduler | controller-0.ostest     | enabled | up    |
| nova-conductor | cellcontroller-0.ostest | enabled | up    |
| nova-compute   | compute-0.ostest        | enabled | up    |
| nova-compute   | compute-1.ostest        | enabled | up    |
+----------------+-------------------------+---------+-------+
```

4.3. セルへのコンピュートノードの追加

いずれかのコントローラーノードでセルホストの検出を実行して、コンピュートノードを検出し、ノード間のマッピングで API データベースを更新します。

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. セルのコントロールプレーンの IP アドレスを取得します。

```
$ CTRL_IP=$(ansible-inventory -i /home/stack/overcloud-deploy/overcloud/config-download/overcloud/tripleo-ansible-inventory.yaml --host <controller_node> | jq -r .ctlplane_ip)
```

   - **<controller_node>** は、コントローラーノードの名前 (**controller-0** など) に置き換えます。

3. コンピューティングホストを公開してセルに割り当てます。

```
$ ssh tripleo-admin@${CTRL_IP} sudo podman exec -i -u root nova_api \
  nova-manage cell_v2 discover_hosts --by-service --verbose
```

4. コンピュートホストがセルに割り当てられたことを確認します。

```
$ ssh tripleo-admin@${CTRL_IP} sudo podman exec -i -u root nova_api \
  nova-manage cell_v2 list_hosts
```

4.4. セルのアベイラビリティーゾーンの作成

各セルに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。これにより、そのセル内のコンピュートノードで作成されたインスタンスが、同じセル内の他のコンピュートノードにのみ移行されるようにします。セル間でのインスタンの移行はサポートされていません。

セル AZ を作成したら、セル内の全コンピュートノードを AZ に追加する必要があります。デフォルトのセルは、コンピュートセルとは異なるアベイラビリティーゾーンに属している必要があります。

手順

1. source コマンドで **overcloudrc** ファイルを読み込みます。

```
(undercloud)$ source ~/overcloudrc
```

2. セルの AZ を作成します。

```
(overcloud)# openstack aggregate create \
 --zone <availability_zone> \
 <aggregate_name>
```

   - **<availability_zone>** をアベイラビリティーゾーンに割り当てる名前に置き換えてください。
   - **<aggregate_name>** をホストアグリゲートに割り当てる名前に置き換えてください。

3. オプション: アベイラビリティーゾーンにメタデータを追加します。

```
(overcloud)# openstack aggregate set --property <key=value> \
  <aggregate_name>
```

- **<key=value>** をメタデータのキー/値のペアに置き換えてください。キー/値の属性は、必要なだけ追加することができます。
- **<aggregate_name>** をアベイラビリティーゾーンホストアグリゲートの名前に置き換えてください。

4. セルに割り当てられたコンピュートノードのリストを取得します。

```
$ ssh tripleo-admin@${CTRL_IP} sudo podman exec -i -u root nova_api \
 nova-manage cell_v2 list_hosts
```

5. セルに割り当てられたコンピュートノードをセルのアベイラビリティーゾーンに追加します。

```
(overcloud)# openstack aggregate add host <aggregate_name> \
 <host_name>
```

- **<aggregate_name>** は、コンピュートノードを追加するアベイラビリティーゾーンホストアグリゲートの名前に置き換えてください。
- **<host_name>** は、アベイラビリティーゾーンに追加するコンピュートノードの名前に置き換えてください。

注記

- この段階ではセルが作成されていないため、**OS::TripleO::Services::NovaAZConfig** パラメーターを使用してデプロイメント時に AZ を自動的に作成することはできません。
- セル間でのインスタンスの移行はサポートされていません。インスタンスを別のセルに移動するには、インスタンスを古いセルから削除し、新しいセルに作成し直す必要があります。

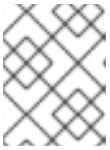
ホストアグリゲートとアベイラビリティーゾーンの詳細は、ホストアグリゲートの作成と管理 を参照してください。

4.5. セルからのコンピュートノードの削除

セルからコンピュートノードを削除するには、セルからインスタンスをすべて削除し、Placement データベースからホスト名を削除する必要があります。

手順

1. セル内のコンピュートノードからすべてのインスタンスを削除します。



注記

セル間でのインスタンスの移行はサポートされていません。インスタンスを削除して、別のセルに作成し直す必要があります。

2. グローバルコントローラーの 1 つで、セルからすべてのコンピューティングノードを削除します。

```
$ CTRL_IP=$(ansible-inventory -i /home/stack/overcloud-deploy/overcloud/config-
```

```
download/overcloud/tripleo-ansible-inventory.yaml --host <controller_node> | jq -r
.ctlplane_ip)

$ ssh tripleo-admin@${CTRL_IP} sudo podman  \
 exec -i -u root nova_api \
 nova-manage cell_v2 list_hosts

$ ssh tripleo-admin@${CTRL_IP} sudo podman  \
 exec -i -u root nova_api \
 nova-manage cell_v2 delete_host --cell_uuid <uuid> --host <compute>
```

- **<controller_node>** は、コントローラーノードの名前 (**controller-0** など) に置き換えます。

3. Placement サービスからセルのリソースプロバイダーを削除し、後で別のセルに同じホスト名のコンピュートノードを追加する場合には、ホスト名が利用可能になるようにします。

```
(undercloud)$ source ~/overcloudrc

(overcloud)$ openstack resource provider list
+--------------------------------------+---------------------------------------+------------+
| uuid                                 | name                                  | generation |
+--------------------------------------+---------------------------------------+------------+
| 9cd04a8b-5e6c-428e-a643-397c9bebcc16 | computecell1-novacompute-0.site1.test |
11 |
+--------------------------------------+---------------------------------------+------------+

(overcloud)$ openstack resource provider  \
 delete 9cd04a8b-5e6c-428e-a643-397c9bebcc16
```

4.6. セルの削除

セルを削除するには、セルからのコンピューティングノードの削除 で説明されているように、まずセルからすべてのインスタンスとコンピュートノードを削除する必要があります。続いて、セル自体とセルスタックを削除します。

手順

1. グローバルコントローラーの 1 つでセルを削除します。

```
$ CTRL_IP=$(ansible-inventory -i /home/stack/overcloud-deploy/overcloud/config-
download/overcloud/tripleo-ansible-inventory.yaml --host <controller_node> | jq -r
.ctlplane_ip)

$ ssh tripleo-admin@${CTRL_IP} sudo podman  \
 exec -i -u root nova_api \
 nova-manage cell_v2 list_cells

$ ssh tripleo-admin@${CTRL_IP} sudo podman  \
 exec -i -u root nova_api \
 nova-manage cell_v2 delete_cell --cell_uuid <uuid>
```

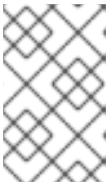
- **<controller_node>** は、コントローラーノードの名前 (**controller-0** など) に置き換えます。

2. エフェメラル Heat プロセスを開始し、Heat 環境をエクスポートします。

```
(undercloud)$ openstack tripleo launch heat --heat-dir /home/stack/overcloud-
deploy/cell1/heat-launcher --restore-db
(undercloud)$ export OS_CLOUD=heat
```

3. オーバークラウドからセルスタックを削除します。

```
$ openstack stack delete <stack name> --wait --yes
```



注記

コントローラーセルとコンピュートセル用に別個のセルスタックをデプロイした場合は、最初にコンピュートセルスタックを削除し、その後にコントローラーセルスタックを削除します。

4. セルスタックの削除が完了したら、アンダークラウドからエフェメラル Heat プロセスを削除します。

```
(undercloud)$ openstack tripleo launch heat --kill
```