



# Red Hat OpenStack Platform 16.1

## コンピュータセルでのデプロイメントのスケール 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



# Red Hat OpenStack Platform 16.1 コンピュートセルでのデプロイメントの 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

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<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

Java<sup>®</sup> is a registered trademark of Oracle and/or its affiliates.

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

MySQL<sup>®</sup> is a registered trademark of MySQL AB in the United States, the European Union and other countries.

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

All other trademarks are the property of their respective owners.

## 概要

本ガイドでは、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ター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について説明します。

## 目次

多様性を受け入れるオープンソースの強化 .....	3
RED HAT ドキュメントへのフィードバック (英語のみ) .....	4
<b>第1章 マルチセルオーバークラウドのデプロイメント .....</b>	<b>5</b>
1.1. 前提条件 .....	5
1.2. グローバルなコンポーネントおよびサービス .....	5
1.3. セル固有のコンポーネントおよびサービス .....	6
1.4. セルデプロイメントのアーキテクチャー .....	6
1.5. マルチセルのデプロイメントで考慮すべき事項 .....	7
<b>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 .....</b>	<b>9</b>
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 .....	9
2.2. セルのロールファイルの作成 .....	10
2.3. CELLCONTROLLER ロールのホストの指定 .....	10
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 .....	11
2.5. 次のステップ .....	12
<b>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 .....</b>	<b>13</b>
3.1. 前提条件 .....	13
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 .....	13
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 .....	14
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 .....	15
3.5. セルロールのホストの指定 .....	16
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 .....	18
3.7. デプロイメント後の新規セルサブネットの追加 .....	19
3.8. 次のステップ .....	20
<b>第4章 COMPUTE サービス内のセルの作成および管理 .....</b>	<b>21</b>
4.1. 前提条件 .....	21
4.2. COMPUTE サービス内でのセルの作成 .....	21
4.3. セルへのコンピュートノードの追加 .....	22
4.4. セルのアベイラビリティゾーンの作成 .....	23
4.5. セルからのコンピュートノードの削除 .....	24
4.6. セルの削除 .....	25



## 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。

## 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 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タ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タ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、多数のコンピュータ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ルアップする場合は、デフォルトのセルにコンピュータ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タノードを追加して、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タノードを追加する。
4. 各コンピュータセルをアベイラビリティ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、コンピュータ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タ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

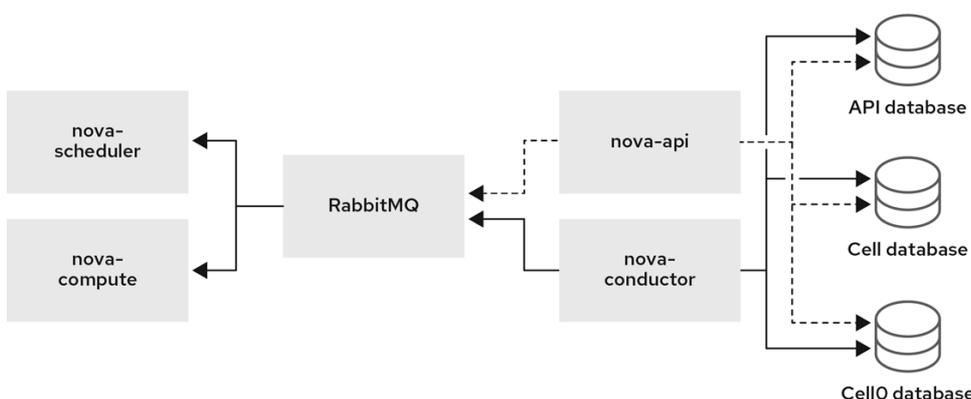
セル内の各サービス間の通信、およびグローバルなサービスとの通信のために、すべてのサービスによって使用されるメッセージングサービス

## 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。



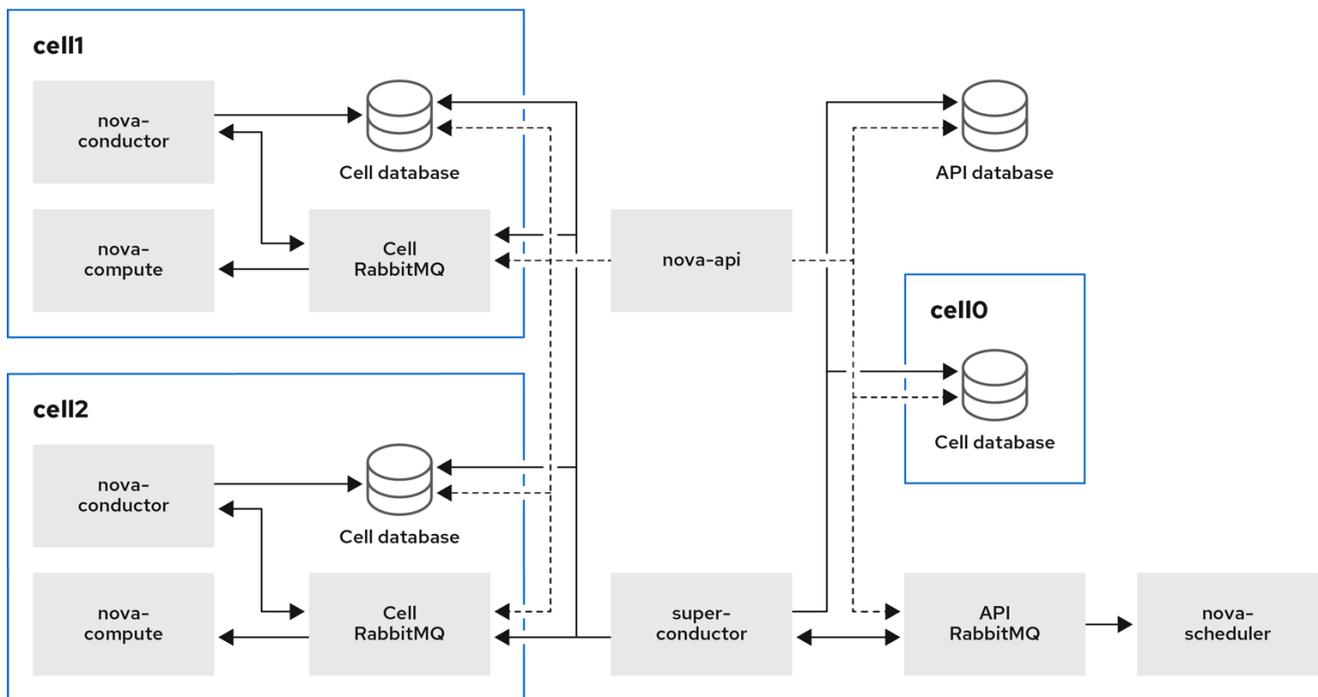
81\_OpenStack\_0420

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

最初のセルでは Compute サービスを実行しないでください。default という名前の。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

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を設定する必要があります。各コンソールプロキシサーバーは、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(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。セルのコンピュータ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** ロール用にセルコントローラーフレーバーを設定します。



### 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、カスタムロール用のフレーバーを設定する必要もあります。

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)$ sudo --preserve-env openstack overcloud cell export \
--output-file common/default_cell_export.yaml
```

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** ロールのノードをタグ付けするためのフレーバーおよびリソースクラスを設定する必要があります。以下の手順では、**CellController** ロール用にフレーバーおよびベアメタルリソースクラスを作成します。

### ヒント

複数のセル環境にカスタムロールを作成している場合は、以下の手順に従ってカスタムロール用のフレーバーおよびリソースクラスを設定します。そのためには、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ください。

### 手順

1. セルコントローラーノード用の **cellcontroller** オーバークラウドフレーバーを作成します。

```
(undercloud)$ openstack flavor create --id auto \
--ram <ram_size_mb> --disk <disk_size_gb> \
--vcpus <no_vcpus> cellcontroller
```

- **<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。



### 注記

これらの属性は、インスタンスのスケジューリングには使用されません。ただし Compute スケジューラーは、ディスク容量を使用してルートパーティションのサイズを決定します。

2. ノードリストを取得して UUID を把握します。

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

3. カスタムのセルコントローラークラスで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をタグ付けします。

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

<node> をベアメタルノードの ID に置き換えてください。

4. **cellcontroller** フレーバーをカスタムセルコントローラーリソースクラスに関連付けます。

```
(undercloud)$ openstack flavor set \
--property resources:CUSTOM_BAREMETAL_CELL_CONTROLLER=1 \
cellcontroller
```

Bare Metal 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。

5. 以下のフレーバー属性を設定して、Compute スケジューラーがインスタンスのスケジューリングにベアメタルフレーバー属性を使用するのを防ぎます。

```
(undercloud)$ openstack flavor set \
--property resources:VCPU=0 --property resources:MEMORY_MB=0 \
--property resources:DISK_GB=0 cellcontroller
```

## 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

各セルスタックで、オーバークラウドスタックのネットワークを使用して、セルを追加のセルとして特定する必要があります。また、ノードのフレーバーおよびセル内のコントローラーノードの数も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。

```
resource_registry:
  OS::TripleO::Network::Ports::OVNDBsVipPort: /usr/share/openstack-tripleo-heat-templates/network/ports/noop.yaml
  OS::TripleO::Network::Ports::RedisVipPort: /usr/share/openstack-tripleo-heat-templates/network/ports/noop.yaml
```

```
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

# Map the flavors to use for the CellController and Compute roles
OvercloudCellControllerFlavor: cellcontroller
OvercloudComputeFlavor: compute

# Number of controllers/computes in the cell
CellControllerCount: 3
ComputeCount: 1

# Node names must be unique across all cells
ComputeHostnameFormat: 'cell1-compute-%index%'
CellControllerHostnameFormat: 'cell1-cellcontroller-%index%'

```

- セルにネットワークリソースを割り当て、セルをネットワークに登録するには、各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::CellController::Net::SoftwareConfig: /home/stack/templates/nic-
  configs/cellcontroller.yaml
  OS::TripleO::Compute::Net::SoftwareConfig: /home/stack/templates/nic-
  configs/compute.yaml

```

- 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] \
-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

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ター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フレーバーを設定します。



### 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。お使いの環境に複数のコントロールプレーンネットワーク (スパイン/リーフネットワーク環境など) がある場合には、各リーフネットワーク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フレーバーも作成する必要があります。詳細は、[Creating flavors and tagging nodes for leaf networks](#) 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についての詳しい情報は、[オーバークラウドの高度なカスタマイズのカスタム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 \
-n /home/stack/common/network_data_routed_multi_cell.yaml \
-e [your environment files]
```

### 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)$ sudo --preserve-env openstack overcloud cell export \
--output-file common/default_cell_export.yaml
```

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、Advanced Overcloud Customization の [Designing 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、cell ロールのノードをタグ付けするためのフレーバーおよびリソースクラスを設定する必要があります。**cellcontrollercell1** ロール用のフレーバーおよびベアメタルリソースクラスを作成するには、以下の手順を実施します。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。

## 手順

1. **cell1** コントローラーノード用の **cellcontrollercell1** オーバークラウドフレーバーを作成します。

```
(undercloud)$ openstack flavor create --id auto \
--ram <ram_size_mb> --disk <disk_size_gb> \
--vcpus <no_vcpus> cellcontrollercell1
```

- **<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。



### 注記

これらの属性は、インスタンスのスケジューリングには使用されません。ただし Compute スケジューラーは、ディスク容量を使用してルートパーティションのサイズを決定します。

2. ノードリストを取得して UUID を把握します。

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

3. カスタムのセルコントローラークラスで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をタグ付けします。

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

**<node>** をベアメタルノードの ID に置き換えてください。

4. **cellcontrollercell1** フレーバーをカスタムセルコントローラーリソースクラスに関連付けます。

```
(undercloud)$ openstack flavor set \
--property resources:CUSTOM_BAREMETAL_CELL_CONTROLLER=1 \
cellcontrollercell1
```

Bare Metal 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。

- 以下のフレーバー属性を設定して、Compute スケジューラーがインスタンスのスケジューリングにベアメタルフレーバー属性を使用するのを防ぎます。

```
(undercloud)$ openstack flavor set \
  --property resources:VCPU=0 --property resources:MEMORY_MB=0 \
  --property resources:DISK_GB=0 cellcontrollercell1
```

### 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

- 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。
- 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
  OS::TripleO::Network::Ports::OVNDBsVipPort: /usr/share/openstack-tripleo-heat-
  templates/network/ports/noop.yaml
  OS::TripleO::Network::Ports::RedisVipPort: /usr/share/openstack-tripleo-heat-
  templates/network/ports/noop.yaml

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

  # Map the flavors to use for the CellController and Compute roles
  OvercloudCellControllerCell1 Flavor: cellcontrollercell1
  OvercloudComputeCell1 Flavor: computecell1

  # Number of controllers/computes in the cell
  CellControllerCell1 Count: 3
  ComputeCell1 Count: 1
```

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] \
-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:**NetworkDeploymentActions** を次のデプロイメントの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を作成します。



#### 注記

作成および起動するセルごとに、このプロセスを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。コミュニティのドキュメントはそのままの状態を提供され、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=$(openstack server list -f value -c Networks --name overcloud-controller-0 | sed 's/ctlplane=//')
$ CELL_CTRL_IP=$(openstack server list -f value -c Networks --name cell1-cellcontroller-0 | sed 's/ctlplane=//')
```

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=$(ssh heat-admin@${CELL_CTRL_IP} \
egrep cell1.*\.internalapi /etc/hosts)
(undercloud)$ ansible -i /usr/bin/tripleo-ansible-inventory \
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 heat-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 heat-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、ホストの検出コマンドを入力してコンピュートホストを公開してセルに割り当てます。

```
$ CTRL_IP=$(openstack server list -f value -c Networks --name overcloud-controller-0 | sed 's/ctlplane=//')
$ ssh heat-admin@${CTRL_IP} sudo podman exec -i -u root nova_api \
nova-manage cell_v2 discover_hosts --by-service --verbose
```

3. 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 heat-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ゾーンの作成

各セルにアベイラビリティゾーン (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ス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>** はメタデータのキー/値のペアに置き換えてください。キー/値の属性は、必要だけ追加することができます。
  - **<aggregate\_name>** をアベイラビリティ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 heat-admin@${CTRL_IP} sudo podman exec -i -u root nova_api \
nova-manage cell_v2 list_hosts
```

- セルに割り当てられたコンピューターノードをセルのアベイラビリティゾーンに追加します。

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

- **<aggregate\_name>** は、コンピューターノードを追加するアベイラビリティ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、アベイラビリティゾーンに追加するコンピューター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およびアベイラビリティゾーンについての情報は、[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ターノードの削除

セルからコンピューター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

- セル内のコンピューター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。

- グローバルコントローラーのいずれかで、セルから全コンピューターノードを削除します。

```
$ CTRL_IP=$(openstack server list -f value -c Networks --name overcloud-controller-0 | sed 's/ctlplane=//')
```

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

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

- 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トノードの削除](#)で説明するように、まず初めにセルからインスタンスおよびコンピュートノードをすべて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

- グローバルコントローラーのいずれかで、セルを削除します。

```
$ CTRL_IP=$(openstack server list -f value -c Networks --name overcloud-controller-0 | sed 's/ctlplane=//')

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

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

- 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 && openstack \
overcloud plan delete <stack_name>
```



### 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。