Red Hat OpenStack Platform 16.0

個別の heat スタックを使用した分散コンピュートノードのデプロイ

個別の heat スタックを使用して、Red Hat Openstack Platform を管理する

Last Updated: 2023-02-18

Red Hat OpenStack Platform 16.0 個別の heat スタックを使用した分散コンピュートノードのデプロイ

個別の heat スタックを使用して、Red Hat Openstack Platform を管理する

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Deploying_Distributed_Compute_Nodes_with_Separate_Heat_Stacks.ent file |.

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

個別の heat スタックを使用して Red Hat OpenStack Platform をデプロイし、director が実行する管理操作を分離することができます。この手法を使用して、コントローラーノードを更新せずにコンピュートノードをスケーリングしたり、複数の Red Hat Ceph Storage クラスターをデプロイしたり、分散コンピュートノード (DCN) サイト内で中央データセンターへのネットワークおよび管理の依存関係を減らしたりすることができます。本ガイドでは、Red Hat OpenStack Platform デプロイメントで異なるノード種別を管理するために、個別の heat スタックを設定する方法について説明します。

目次

第1章 個別の HEAT スタックを使用したデプロイメントの作成

Red Hat OpenStack Platform 環境で個別の heat スタックを使用する場合、director が実行する管理操作を分離することができます。たとえば、コントロールプレーンスタックが管理するコントローラーノードを更新せずに、コンピュートノードをスケーリングすることができます。この手法を使用して、複数の Red Hat Ceph Storage クラスターをデプロイすることもできます。

1.1. 個別の HEAT スタックの使用

通常の Red Hat OpenStack Platform のデプロイメントでは、単一の heat スタックはコントロールプレーン (コントローラー) を含むすべてのノードを管理します。個別の heat スタックを使用して、以前のアーキテクチャー制約に対応することができるようになりました。

- 異なるノード種別には、個別の heat スタックを使用します。たとえば、コントロールプレーン、コンピュートノード、および HCI ノードは、それぞれ独自のスタックで管理できます。これにより、コントロールプレーンに影響を与えずにコンピュートスタックを変更またはスケーリングできます。
- 同じサイトで個別の heat スタックを使用して、複数の Ceph クラスターをデプロイすることができます。
- 同じデータセンター内の異なるアベイラビリティーゾーン (AZ) には、個別の heat スタックを使用することができます。
- 分散コンピュートノード (DCN) アーキテクチャーを使用する Red Hat OpenStack Platform のデプロイには、個別の heat スタックが必要です。これにより、中央データセンターへのネットワークおよび管理の依存関係を減らすことができます。このアーキテクチャーの各エッジサイトには、コンピュートノードとストレージノードの両方からの独自の AZ も必要です。
  この機能は、本リリースでは **テクノロジープレビュー** として提供しているため、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。

1.2. 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個別の heat スタックに分割することはできません。
- 分散コンピュートノード (DCN) サイトの場合: ハイパーコンバージドインフラストラクチャー (HCI) コンピュートノードまたは標準のコンピュートノードとして機能できる 3 つのノード。
- それぞれの追加の DCN サイト: 3 つの HCI コンピュートノードまたは Ceph ノード。
- すべてノードは事前にプロビジョニングされているか、または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。

1.3. 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。

- Image サービス (glance) のマルチストアは現在利用できませんが、今後のリリースでご利用いただける予定です。本書の例では、Block Storage (cinder) は Ceph Storage を使用する唯一のサービスです。

- スパイン/リーフ型ネットワーク: 本ガイドの例は、ルーティング要件を示していません。ルーティング要件は、ほとんどの分散コンピュートノード (DCN) デプロイメントで確認することができます。

- Ironic DHCP リレー: 本ガイドには、DHCP リレーと共に Ironic を設定する方法は含まれません。

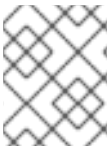
- Pacemaker を使用しない Block Storage (cinder) のアクティブ/アクティブは、テクノロジープレビューとしてのみ利用可能です。

- DCN HCI ノードは、テクノロジープレビューとしてのみ利用可能です。

第2章 個別 HEAT スタックのデプロイメントの設計

個別の heat スタック内でデプロイメントを分割するには、まずコントロールプレーンと共に単一のオーバークラウドをデプロイする必要があります。その後、分散コンピュートノード (DCN) サイト向けに個別のスタックを作成することができます。以下の例は、異なるノード種別の個別スタックを示しています。

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



注記

管理を簡素化するには、各スタックに個別のアベイラビリティーゾーン (AZ) を作成します。

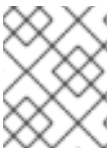
注記

スパイン/リーフ型ネットワークを使用する場合には、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。**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**) はカンマで区切られた 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'
```

2.1. 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。

2.1.1. 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'
```

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. 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。

2.1.2. 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:fffe'}]
  mtu: 1400
```

2.2. サービスの配置

この設定では、それぞれの分散コンピュートノード (DCN) サイトは、Compute および Block Storage (cinder) 用の独自のアベイラビリティーゾーン (AZ) 内にデプロイされます。

- Cinder: 各 DCN サイトは Block Storage AZ を使用して **cinder-volume** サービスを実行します。**cinder-volume** サービスは、今後の更新においてアクティブ/アクティブ設定をサポートすることが想定されます。

- glance: Image サービス (glance) は、中央サイトの Object Storage (swift) バックエンドを使用します。DCN サイト AZ で作成されたすべてのコンピュートインスタンスは、**HTTP GET** を使用して中央サイトからイメージを取得します。今後のリリースでは、Image サービスは中央サイトおよび DCN サイトで Ceph RBD バックエンドを使用します。その後、イメージを中央サイトから DCN サイトへ転送することができます。したがって、DCN の場所で COW ブートできます。

- Ceph: このアーキテクチャーでは、Ceph は中央サイトでは実行されません。その代わりに、各 DCN サイトは、HCI を使用するコンピュートノードと共存する独自の Ceph クラスターを実行します。Ceph バックエンドは、Block Storage ボリュームにのみ使用されます。

2.3. 個別の HEAT スタックの管理

本ガイドの手順では、3 つの heat スタック（ **central**、**dcn0**、および **dcn1** ）をデプロイす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/control-plane-export.yaml
```

control-plane-export.yaml ファイルは、後に **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
```

注記

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。

第3章 コンテナーイメージの取得

個別の heat スタックによるデプロイメントに必要なコンテナーイメージを取得するには、以下の手順およびサンプルファイルのコンテンツを使用ます。詳細については、「Preparing container images」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:
  NeutronMechanismDrivers: ovn
  ContainerImagePrepare:
  - push_destination: 192.168.24.1:8787
    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. 環境ファイルを ~/**containers-env-file.yaml** として生成します。

```
$ openstack tripleo container image prepare -e containers.yaml --output-env-file
~/containers-env-file.yaml
```

中央コントローラーの デプロイ および HCI ノードの分散コンピュートノード(DCN)サイトへのデプロイ では、作成される ~/**containers-env-file.yaml** ファイルはオーバークラウドのデプロイメント手順の一部として含まれます。

第4章 中央コントローラーのデプロイ

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トノードは必要ありません。したがって、コンピュートノード数を **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

1. 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
  OvercloudControlFlavor: baremetal
  OvercloudComputeFlavor: baremetal
  ControllerSchedulerHints:
    'capabilities:node': '0-controller-%index%'
  GlanceBackend: swift
```

   - **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。Red Hat は、マルチバックエンドの glance サポートが利用可能になるまで、Image サービスがこの設定で Ceph を使用しないことをお勧めします。
     この構成は、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。
   - それぞれの DCN には、独自の Object Storage ボリュームサービスがあります。つまり、DCN 上の Ceph ボリュームサービスがローカルの Ceph クラスターを使用するため、ユーザーが中央ノードから異なるアベイラビリティーゾーンに Object Storage ボリュームをスケジュールできることを意味します。

注記

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

2. 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

STACK=central
```

```
source ~/stackrc
time openstack overcloud deploy \
    --stack $STACK \
    --templates /usr/share/openstack-tripleo-heat-templates/ \
    -e /usr/share/openstack-tripleo-heat-templates/environments/podman.yaml \
    -e ~/containers-env-file.yaml \
    -e ~/central/overrides.yaml
```

第5章 HCI ノードのデプロイ

DCN サイトの場合は、単一ノードで Compute および Ceph Storage を使用するハイパーコンバージドインフラストラクチャー (HCI) スタックをデプロイすることができます。たとえば、以下の図には、**dcn0** と **dcn1** という名前の 2 つの DCN スタックが示されています。各 DCN スタックは、それぞれ独自のアベイラビリティーゾーン(AZ)にあります。それぞれの DCN スタックには、独自の Ceph クラスターおよび Compute サービスがあります。

「Configuring the distributed compute node (DCN) environment files 」および「Deploying HCI nodes to the distributed compute node (DCN) site」の手順では、このデプロイメント方法を説明しています。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。この手順の例では、最初の heat スタックは中央データセンター内にオーバークラウドをデプロイします。その後、別の heat スタックが作成され、コンピュートノードのバッチをリモートの物理的な場所にデプロイします。

5.1. 分散コンピュートノードの環境ファイルの設定

以下の手順では、中央サイトのメタデータを取得してから、分散コンピュートノード (DCN) サイトに必要な設定ファイルを生成します。

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、**control-plane** 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/control-plane-export.yaml \
```

重要

以下の手順では、新しい **control-plane-export.yaml** 環境ファイルを作成し、オーバークラウドの **plan-environment.yaml** のパスワードを使用します。**control-plane-export.yaml** ファイルには、機密の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。

5.2. 分散コンピュートノードサイトへの HCI ノードのデプロイ

この手順では、**DistributedComputeHCI** ロールを使用して HCI ノードを **dcn0** という名前のアベイラビリティーゾーン(AZ)にデプロイします。このロールは、分散コンピュート HCI ノード専用に使用されます。

注記

CephMon は HCI ノードで実行され、中央のコントローラーノードでは実行できません。また、中央コントローラーノードは、Ceph なしでデプロイされます。

手順

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

```
parameter_defaults:
  DistributedComputeHCICount: 3
  DistributedComputeHCIFlavor: baremetal
  DistributedComputeHCISchedulerHints:
    'capabilities:node': '0-ceph-%index%'
  CinderStorageAvailabilityZone: dcn0
  NovaAZAttach: false
```

2. **dcn0/ceph.yaml** で提案されている Ceph 設定を確認します。

```
parameter_defaults:
  CephClusterName: dcn0
  NovaEnableRbdBackend: false
  CephAnsiblePlaybookVerbosity: 3
  CephPoolDefaultPgNum: 256
  CephPoolDefaultSize: 3
  CephAnsibleDisksConfig:
    osd_scenario: lvm
    osd_objectstore: bluestore
    devices:
      - /dev/sda
      - /dev/sdb
      - /dev/sdc
      - /dev/sdd
      - /dev/sde
      - /dev/sdf
      - /dev/sdg
      - /dev/sdh
      - /dev/sdi
```

```
      - /dev/sdj
      - /dev/sdk
      - /dev/sdl

## Everything below this line is for HCI Tuning
  CephAnsibleExtraConfig:
    ceph_osd_docker_cpu_limit: 1
    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
```

以下のパラメーターの値を、お使いの環境に適した値に置き換えます。詳細は、「Deploying an overcloud with containerized Red Hat Ceph」および「Hyperconverged Infrastructure」ガイドを参照してください。

- **CephAnsibleExtraConfig**
- **DistributedComputeHCIParameters**
- **CephPoolDefaultPgNum**
- **CephPoolDefaultSize**
- **DistributedComputeHCIExtraConfig**

3. 以下の内容で **nova-az.yaml** という名前の新規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: central
```

オーバークラウドが中央デプロイメントによって作成された **centralrc** ファイルに記載されているエンドポイントにアクセスできると、このコマンドにより **dcn0** という名前の AZ が作成され、デプロイメント時に新しい HCI コンピュートノードがその AZ に追加されます。

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
if [[ ! -e distributed_compute_hci.yaml ]]; then
    openstack overcloud roles generate DistributedComputeHCI -o
distributed_compute_hci.yaml
fi
time openstack overcloud deploy \
    --stack $STACK \
```

```
    --templates /usr/share/openstack-tripleo-heat-templates/ \
    -r distributed_compute_hci.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/disable-telemetry.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/podman.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/cinder-volume-active-
active.yaml \
    -e ~/dcn-common/control-plane-export.yaml \
    -e ~/containers-env-file.yaml \
    -e ceph.yaml \
    -e nova-az.yaml \
    -e overrides.yaml
```

オーバークラウドのデプロイメントが完了したら、*6章 デプロイメント後の設定* を参照してデプロイメント後の設定手順を実施し、確認します。

第6章 デプロイメント後の設定

オーバークラウドのデプロイメントが完了したら、以下の手順を実施して機能を検証します。

手順

1. アベイラビリティーゾーンにテストインスタンスを作成します。この例では、新規のインスタンスは分散コンピュートノード (DCN) で実行されます。特定の AZ は、**--availability-zone** パラメーターを使用してターゲットを持ちます。

```
$ openstack server create --flavor m1.tiny --image cirros --network private --security-group basic dcn-instance --availability-zone dcn0
```

2. 最初のアベイラビリティーゾーンにボリュームを作成します。このボリュームは、**dcn0** ノードで実行している cinder のアクティブ/アクティブサービスを使用します。

```
$ openstack volume create --size 1 --availability-zone dcn0 myvol
```

注記

この手順は、**CinderStorageAvailabilityZone** で定義される cinder アベイラビリティーゾーンの設定によって異なります。詳細は、『**Storage Guide**』の「Deploying availability zones」を参照してください。

2 つの異なる HCI スタックが作成され、それぞれに Ceph クラスターがデプロイされています。HCI の詳細は、『ハイパーコンバージドインフラストラクチャーガイド』を参照してください。

6.1. コンテナーの健全性の確認

コンテナーが正常に機能していることを確認します。

手順

1. SSH を使用して Ceph MON サービスを実行しているノードにログインします。
2. 以下のコマンドを実行して、コンテナーの正常性を表示します。

```
$ podman exec ceph-mon-$HOSTNAME ceph -s --cluster CLUSTERNAME
```

CLUSTERNAME は、クラスター名に置き換えます（例： **dcn0** ）。デフォルト値は **ceph** です。

3. クラスターの健全性ステータスが **HEALTH_OK** であること、およびすべての OSD が **up** の状態にあることを確認します。

第7章 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、**dcn1** 用に新規ディレクトリーを作成します。

```
$ cd ~
$ mkdir dcn1
```

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 **dcn1** に置き換えます。

```
$ cp dcn0/ceph.yaml dcn1/ceph.yaml
$ sed s/dcn0/dcn1/g -i dcn1/ceph.yaml
$ cp dcn0/overrides.yaml dcn1/overrides.yaml
$ sed s/dcn0/dcn1/g -i dcn1/overrides.yaml
$ sed s/"0-ceph-%index%"/"1-ceph-%index%"/g -i dcn1/overrides.yaml
$ cp dcn0/deploy.sh dcn1/deploy.sh
$ sed s/dcn0/dcn1/g -i dcn1/deploy.sh
```

3. **dcn1** 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. 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
```

5. ノードが利用可能な状態になったら、**dcn1** サイト用の **deploy.sh** を実行します。

```
$ bash dcn1/deploy.sh
```