Red Hat OpenStack Platform 11

オーバークラウド向けの外部のロードバランシング

OpenStack Platform 環境で外部のロードバランサーを使用するための設定

OpenStack Team

Red Hat OpenStack Platform 11 オーバークラウド向けの外部のロードバランシング

OpenStack Platform 環境で外部のロードバランサーを使用するための設定

OpenStack Team
rhos-docs@redhat.com

概要

本ガイドでは、Red Hat OpenStack Platform 環境で外部のロードバランサーをオーバークラウド向けに使うための設定について説明します。これには、ロードバランサーの設定ガイドラインや Re Hat OpenStack Platform director を使用したオーバークラウドの設定が含まれます。

目次

第1章 はじめに

Red Hat OpenStack Platform director は、**オーバークラウド** と呼ばれるクラウド環境を作成します。オーバークラウドには、異なる役割を果たす複数のノード種別が含まれます。これらのノード種別の中の 1 つに **コントローラー** ノードがあります。コントローラーは、オーバークラウドを管理する機能を果たし、特定の OpenStack コンポーネントを使用します。オーバークラウドは、複数のコントローラーをまとめて高可用性クラスターとして使用して、Openstack サービスの運用パフォーマンスが最大限となるようにします。また、このクラスターは、OpenStack サービスにアクセスするためのロードバランシング機能も提供し、コントローラーノードへのトラフィックを均等に分散して、各ノードのサーバーの過負荷を軽減します。

外部ロードバランサーを使用してこの分散を実行することも可能です。たとえば、組織は、独自のハードウェアベースのロードバランサーを使用してコントローラーノードへのトラフィックの分散を処理することができます。本ガイドでは、外部のロードバランサーの設定の定義とオーバークラウドの作成に役立つ必要な情報を提供します。これには、以下のプロセスを伴います。

1. **ロードバランサーのインストールと設定** 本ガイドでは、ロードバランシングとサービスの HAProxy オプションについて記載しています。設定値は、お使いの外部のロードバランサーに適した値に変更して適用してください。

2. **オーバークラウドの設定とデプロイ**: 本ガイドには、オーバークラウドと外部のロードバランサーを統合するのに役立つ、Heat テンプレートのパラメーターを記載しています。これには、主としてロードバランサーおよびノードとして使用するマシンの IP アドレスが必要です。また、本ガイドには、オーバクラウドのデプロイメントを開始するためのコマンドと、外部のロードバランサーを使用するための設定も記載しています。

1.1. オーバークラウドでのロードバランシングの使用

オーバークラウドは、**HAProxy** と呼ばれるオープンソースのツールを使用します。HAProxy は、OpenStack サービスを実行するコントローラーノードのトラフィックの負荷を分散します。**haproxy** パッケージには、haproxy systemd サービスから起動される**haproxy** デーモンがロギング機能およびサンプル設定とともに含まれています。ただし、オーバークラウドは高可用性リソース管理機能 (Pacemaker) も使用して HAProxy 自体を高可用性のサービスとして管理します。そのため、HAProxy は各コントローラーノードで実行され、各設定に定義されているルールに従ってトラフィックを分散します。

1.2. シナリオ例の定義

本書では、一例として以下のシナリオを使用しています。

- HAProxy を使用する外部のロードバランシングサーバー。フェデレーション対応の HAProxy サーバーの使用方法の実例を紹介しますこのサーバーの代わりにサポートされている他の外部のロードバランサーを使用することも可能です。
- OpenStack Platform director ノード 1 台
- オーバークラウドは、以下の要素で構成されます。
- 高可用性クラスター内のコントローラーノード 3 台
- コンピュートノード 1 台
- VLAN を使用したネットワークの分離

このシナリオでは、各ネットワークに以下の IP アドレス割り当てを使用します。

- 内部 API: 172.16.20.0/24
- テナント: 172.16.22.0/24
- ストレージ: 172.16.21.0/24
- ストレージ管理: 172.16.19.0/24
- 外部: 172.16.23.0/24

IP アドレスの範囲には、コントローラーノードに割り当てられる IP アドレスと、ロードバランサーが OpenStack サービスにバインディングするための仮想 IP アドレスが含まれます。

第2章 デフォルト設定の定義

外部のロードバランサーなしのオーバークラウドを作成/設定する際には、director は HAProxy が複数の OpenStack サービスにトラフィックを分散するように設定します。director はこの設定を各コントローラーノードの **/etc/haproxy/haproxy.conf** ファイルで提供します。デフォルトの設定には主に、グローバル、デフォルト、複数のサービスの 3 セクションがあります。

この後の数セクションでは、各設定セクションで使用されるデフォルトのパラメーターを考察します。これは、外部のロードバランサーの設定値の設定例を提供します。これらのパラメーターは、HAProxy の全パラーメーターの中のごく一部であることに注意してください。これらのパラメーターおよびその他のパラメーターについての詳しい情報はコントローラーノード (または **haproxy** パッケージがインストールされた任意のシステム) の **/usr/share/doc/haproxy-*/configuration.txt** にある、『HAProxy Configuration Manual』を参照してください。

2.1. グローバルの設定

```
global
  daemon
  group  haproxy
  log  /dev/log local0
  maxconn  10000
  pidfile  /var/run/haproxy.pid
  user  haproxy
```

このセクションでは、プロセス全体のパラメーターを定義します。これには、以下が含まれます。

- **daemon**: バックグラウンドプロセスとして実行します。
- **user haproxy**、**group haproxy**: プロセスを所有する Linux ユーザーとグループを定義します。
- **log**: 使用する syslog サーバーを定義します。
- **maxconn**: プロセスへの最大同時接続数を設定します。
- **pidfile**: プロセス ID に使用するファイルを設定します。

2.2. デフォルトの設定

```
defaults
  log  global
  mode  tcp
  retries  3
  timeout  http-request 10s
  timeout  queue 1m
  timeout  connect 10s
  timeout  client 1m
  timeout  server 1m
  timeout  check 10s
```

このセクションでは、各サービスのデフォルトのパラメーターセットを設定します。これには、以下が含まれます。

- **log**: サービスのログ記録を有効にします。**global** の値は、ロギング機能が**global** セクションの **log** パラメーターを使用することを意味します。

- **mode**: 使用するプロトコルを設定します。この場合は、デフォルトは TCP です。
- **retries**: 接続エラーと報告されるまでにサーバーで実行される再試行の回数を設定します。
- **timeout**: 特定の機能の最大待ち時間を設定します。たとえば、**timeout http-request** は HTTP 要求が完了するまでの最大の待ち時間を設定します。

2.3. SERVICES の設定

```
listen ceilometer
  bind 172.16.20.250:8777
  bind 172.16.23.250:8777
  server overcloud-controller-0 172.16.20.150:8777 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:8777 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:8777 check fall 5 inter
2000 rise 2
```

デフォルトのファイルには、複数のサービスの設定セクションがあります。各サービスの設定には、以下が含まれます。

- **listen**: 要求をリッスンするサービスの名前
- **bind**: サービスがリッスンする IP アドレスと TCP ポートの番号
- **server**: サービスを提供する各サーバーの名前、サーバーの IP アドレス、リッスンするポート、およびその他の情報

上記の例は、**ceilometer** サービスの HAProxy 設定を示しています。このサービスは、ceilometer サービスが提供される IP アドレスとポートを特定します (172.16.20.2500 および 172.16.23.250 上のポート 8777)。HAProxy は、これらのアドレスに対して送信された要求を **overcloud-controller-0** (172.16.20.150:8777)、**overcloud-controller-1** (172.16.20.151:8777)、**overcloud-controller-2** (172.16.0.152:8777) のいずれかに転送します。

また、この例の **server** パラメーターにより、以下の設定が有効になります。

- **check**: ヘルスチェックが有効になります。
- **fall 5**: ヘルスチェックに 5 回失敗すると、サービスは停止中と見なされます。
- **inter 2000**: ヘルスチェックの実行間隔が 2000 ミリ秒 (2 秒) に設定されます。
- **rise 2**: ヘルスチェックに 2 回成功すると、サーバーは稼働中と見なされます。

各サービスは、異なるアドレスにバインディングして、異なるネットワークトラフィックの種別を表します。また、一部のサービスには、追加の設定オプションが含まれます。次の章では、各特定サービスの設定を考察して、それらの詳細をお使いの外部のロードバランサーで再現できるようにします。

第3章 サービス設定のリファレンス

本章では、ロードバランシングを使用するオーバークラウド内の各特定サービスの設定について概説します。この設定は、お使いの外部のロードバランサーを設定するための指針として利用してください。これらのパラメーターおよびその他のパラメーターについての詳しい説明は、コントローラーノード (または **haproxy** パッケージがインストールされた任意のシステム) の **/usr/share/doc/haproxy-*/configuration.txt** にある『HAProxy Configuration Manual』を参照してください。

注記

大半のサービスは、デフォルトのヘルスチェック設定を使用します。

- ヘルスチェックの実行間隔が 2000 ミリ秒 (2 秒) に設定されます。
- ヘルスチェックに 2 回成功すると、サーバーは稼働中と見なされます。
- ヘルスチェックに 5 回失敗すると、サービスは停止中と見なされます。

各サービスは、デフォルトのヘルスチェックまたは追加のオプションを示すか、各サービスの **Other information** セクションに追加のオプションを示します。

3.1. AODH

ポート番号: 8042

バインド先: internal_api、external

ターゲットのネットワーク/サーバー: overcloud-controller-0、overcloud-controller-1、overcloud-controller-2 上の internal_api

その他の情報:

- 各ターゲットサーバーは、デフォルトのヘルスチェックを使用します。

HAProxy の例:

```
listen aodh
  bind 172.16.20.250:8042
  bind 172.16.23.250:8042
  server overcloud-controller-0 172.16.20.150:8042 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:8042 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:8042 check fall 5 inter
2000 rise 2
```

3.2. CEILOMETER

ポート番号: 8777

バインド先: internal_api、external

ターゲットのネットワーク/サーバー: overcloud-controller-0、overcloud-controller-1、overcloud-controller-2 上の internal_api

その他の情報:

- 各ターゲットサーバーは、デフォルトのヘルスチェックを使用します。

HAProxy の例:

```
listen ceilometer
  bind 172.16.20.250:8777
  bind 172.16.23.250:8777
  server overcloud-controller-0 172.16.20.150:8777 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:8777 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:8777 check fall 5 inter
2000 rise 2
```

3.3. CINDER

ポート番号: 8776

バインド先: internal_api、external

ターゲットのネットワーク/サーバー: overcloud-controller-0、overcloud-controller-1、overcloud-controller-2 上の internal_api

その他の情報:

- 各ターゲットサーバーは、デフォルトのヘルスチェックを使用します。

HAProxy の例:

```
listen cinder
  bind 172.16.20.250:8776
  bind 172.16.23.250:8776
  server overcloud-controller-0 172.16.20.150:8776 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:8776 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:8776 check fall 5 inter
2000 rise 2
```

3.4. GLANCE_API

ポート番号: 9292

バインド先: storage、external

ターゲットのネットワーク/サーバー: overcloud-controller-0、overcloud-controller-1、overcloud-controller-2 上のストレージ

その他の情報:

- 各ターゲットサーバーは、デフォルトのヘルスチェックを使用します。

HAProxy の例:

```
listen glance_api
  bind 172.16.23.250:9292
  bind 172.16.21.250:9292
  server overcloud-controller-0 172.16.21.150:9292 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.21.151:9292 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.21.152:9292 check fall 5 inter
2000 rise 2
```

3.5. GLANCE_REGISTRY

ポート番号: 9191

バインド先: internal_api

ターゲットのネットワーク/サーバー: overcloud-controller-0、overcloud-controller-1、overcloud-controller-2 上の internal_api

その他の情報:

- 各ターゲットサーバーは、デフォルトのヘルスチェックを使用します。

HAProxy の例:

```
listen glance_registry
  bind 172.16.20.250:9191
  server overcloud-controller-0 172.16.20.150:9191 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:9191 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:9191 check fall 5 inter
2000 rise 2
```

3.6. GNOCCHI

ポート番号: 8041

バインド先: internal_api、external

ターゲットのネットワーク/サーバー: overcloud-controller-0、overcloud-controller-1、overcloud-controller-2 上の internal_api

その他の情報:

- 各ターゲットサーバーは、デフォルトのヘルスチェックを使用します。

HAProxy の例:

```
listen gnocchi
  bind 172.16.20.250:8041
  bind 172.16.23.250:8041
  server overcloud-controller-0 172.16.20.150:8041 check fall 5 inter
```

```
2000 rise 2
  server overcloud-controller-1 172.16.20.151:8041 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:8041 check fall 5 inter
2000 rise 2
```

3.7. HEAT_API

ポート番号: 8004

バインド先: internal_api、external

ターゲットのネットワーク/サーバー: overcloud-controller-0、overcloud-controller-1、overcloud-controller-2 上の internal_api

その他の情報:

- 各ターゲットサーバーは、デフォルトのヘルスチェックを使用します。
- このサービスは、デフォルトの TCP モードの代わりに HTTP モードを使用します。

HAProxy の例:

```
listen heat_api
  bind 172.16.20.250:8004
  bind 172.16.23.250:8004
  mode http
  server overcloud-controller-0 172.16.20.150:8004 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:8004 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:8004 check fall 5 inter
2000 rise 2
```

3.8. HEAT_CFN

ポート番号: 8000

バインド先: internal_api、external

ターゲットのネットワーク/サーバー: overcloud-controller-0、overcloud-controller-1、overcloud-controller-2 上の internal_api

その他の情報:

- 各ターゲットサーバーは、デフォルトのヘルスチェックを使用します。

HAProxy の例:

```
listen heat_cfn
  bind 172.16.20.250:8000
  bind 172.16.23.250:8000
  server overcloud-controller-0 172.16.20.150:8000 check fall 5 inter
2000 rise 2
```

```
  server overcloud-controller-1 172.16.20.152:8000 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.151:8000 check fall 5 inter
2000 rise 2
```

3.9. HEAT_CLOUDWATCH

ポート番号: 8003

バインド先: internal_api、external

ターゲットのネットワーク/サーバー: overcloud-controller-0、overcloud-controller-1、overcloud-controller-2 上の internal_api

その他の情報:

- 各ターゲットサーバーは、デフォルトのヘルスチェックを使用します。

HAProxy の例:

```
listen heat_cloudwatch
  bind 172.16.20.250:8003
  bind 172.16.23.250:8003
  server overcloud-controller-0 172.16.20.150:8003 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:8003 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:8003 check fall 5 inter
2000 rise 2
```

3.10. HORIZON

ポート番号: 80

バインド先: internal_api、external

ターゲットのネットワーク/サーバー: overcloud-controller-0、overcloud-controller-1、overcloud-controller-2 上の internal_api

その他の情報:

- 各ターゲットサーバーは、デフォルトのヘルスチェックを使用します。
- このサービスは、デフォルトの TCP モードの代わりに HTTP モードを使用します。
- このサービスは、UI との対話に Cookie によるパーシステンスを使用します。

HAProxy の例:

```
listen horizon
  bind 172.16.20.250:80
  bind 172.16.23.250:80
  mode http
  cookie SERVERID insert indirect nocache
  server overcloud-controller-0 172.16.20.150:80 check fall 5 inter
2000 rise 2
```

```
  server overcloud-controller-1 172.16.20.151:80 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:80 check fall 5 inter
2000 rise 2
```

3.11. KEYSTONE_ADMIN

ポート番号: 35357

バインド先: internal_api、external

ターゲットのネットワーク/サーバー: overcloud-controller-0、overcloud-controller-1、overcloud-controller-2 上の internal_api

その他の情報:

- 各ターゲットサーバーは、デフォルトのヘルスチェックを使用します。

HAProxy の例:

```
listen keystone_admin
  bind 172.16.23.250:35357
  bind 172.16.20.250:35357
  server overcloud-controller-0 172.16.20.150:35357 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:35357 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:35357 check fall 5 inter
2000 rise 2
```

3.12. KEYSTONE_ADMIN_SSH

ポート番号: 22

バインド先: internal_api

ターゲットのネットワーク/サーバー: overcloud-controller-0、overcloud-controller-1、overcloud-controller-2 上の internal_api

その他の情報:

- 各ターゲットサーバーは、デフォルトのヘルスチェックを使用します。

HAProxy の例:

```
listen keystone_admin_ssh
  bind 172.16.20.250:22
  server overcloud-controller-0 172.16.20.150:22 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:22 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:22 check fall 5 inter
2000 rise 2
```

3.13. KEYSTONE_PUBLIC

3.13. KEYSTONE_PUBLIC

ポート番号: 5000

バインド先: internal_api、external

ターゲットのネットワーク/サーバー: overcloud-controller-0、overcloud-controller-1、overcloud-controller-2 上の internal_api

その他の情報:

- 各ターゲットサーバーは、デフォルトのヘルスチェックを使用します。

HAProxy の例:

```
listen keystone_public
  bind 172.16.20.250:5000
  bind 172.16.23.250:5000
  server overcloud-controller-0 172.16.20.150:5000 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:5000 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:5000 check fall 5 inter
2000 rise 2
```

3.14. MYSQL

ポート番号: 3306

バインド先: internal_api

ターゲットのネットワーク/サーバー: overcloud-controller-0、overcloud-controller-1、overcloud-controller-2 上の internal_api

その他の情報:

- 各ターゲットサーバーは、デフォルトのヘルスチェックを使用します。ただし、ヘルスチェックにはポート 9200 を使用します。
- このサービスは、一度に 1 サーバーに対してのみに負荷分散されます。
- バックアップ用でないサーバーが利用できない場合には、各サーバーはロードバランシングのみに使用されます。
- サーバーのステータスが down の場合には、全接続が直ちに切断されます。
- TCP キープアライブパケットの送信を両側で有効にします。
- サーバーの正常性をチェックする HTTP プロトコルを有効にします。
- IP アドレスを保管するスティッキネステーブルを設定します。これは、パーシステンスの維持に役立ちます。

重要

mysql サービスは、Galera を使用して高可用性のデータベースクラスターを提供します。Galera は **active/active** 構成をサポートしますが、ロックの競合を回避するためにロードバランサーによって有効化される **active/passive** 構成を使用することを推奨します。

HAProxy の例:

```
listen mysql
  bind 172.16.20.250:3306
  option tcpka
  option httpchk
  stick on dst
  stick-table type ip size 1000
  timeout client 0
  timeout server 0
  server overcloud-controller-0 172.16.20.150:3306 backup check fall 5
inter 2000 on-marked-down shutdown-sessions port 9200 rise 2
  server overcloud-controller-1 172.16.20.151:3306 backup check fall 5
inter 2000 on-marked-down shutdown-sessions port 9200 rise 2
  server overcloud-controller-2 172.16.20.152:3306 backup check fall 5
inter 2000 on-marked-down shutdown-sessions port 9200 rise 2
```

3.15. NEUTRON

ポート番号: 9696

バインド先: internal_api、external

ターゲットのネットワーク/サーバー: overcloud-controller-0、overcloud-controller-1、overcloud-controller-2 上の internal_api

その他の情報:

- 各ターゲットサーバーは、デフォルトのヘルスチェックを使用します。

HAProxy の例:

```
listen neutron
  bind 172.16.20.250:9696
  bind 172.16.23.250:9696
  server overcloud-controller-0 172.16.20.150:9696 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:9696 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:9696 check fall 5 inter
2000 rise 2
```

3.16. NOVA_EC2

ポート番号: 8773

バインド先: internal_api、external

ターゲットのネットワーク/サーバー: overcloud-controller-0、overcloud-controller-1、overcloud-controller-2 上の internal_api

その他の情報:

- 各ターゲットサーバーは、デフォルトのヘルスチェックを使用します。

HAProxy の例:

```
listen nova_ec2
  bind 172.16.20.250:8773
  bind 172.16.23.250:8773
  server overcloud-controller-0 172.16.20.150:8773 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:8773 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:8773 check fall 5 inter
2000 rise 2
```

3.17. NOVA_METADATA

ポート番号: 8775

バインド先: internal_api

ターゲットのネットワーク/サーバー: overcloud-controller-0、overcloud-controller-1、overcloud-controller-2 上の internal_api

その他の情報:

- 各ターゲットサーバーは、デフォルトのヘルスチェックを使用します。

HAProxy の例:

```
listen nova_metadata
  bind 172.16.20.250:8775
  server overcloud-controller-0 172.16.20.150:8775 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:8775 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:8775 check fall 5 inter
2000 rise 2
```

3.18. NOVA_NOVNCPROXY

ポート番号: 6080

バインド先: internal_api、external

ターゲットのネットワーク/サーバー: overcloud-controller-0、overcloud-controller-1、overcloud-controller-2 上の internal_api

その他の情報:

- 各ターゲットサーバーは、デフォルトのヘルスチェックを使用します。
- デフォルトの分散メソッドはラウンドロビン式ですが、このサービスには、**ソース** メソッドを使用します。このメソッドは、ソース IP アドレスをハッシュ化し、実行中のサーバーの加重値の和で除算します。これにより、要求を受信するサーバーを指定します。こうすることでサーバーが down または up のステータスに切り替わらない限りは、確実に同じクライアント IP アドレスが常に同じサーバーに到達するようにします。ハッシュ化により実行中のサーバーの数に変更があった場合には、バランサーは多数のクライアントを別のサーバーにリダイレクトします。

HAProxy の例:

```
listen nova_novncproxy
  bind 172.16.20.250:6080
  bind 172.16.23.250:6080
  balance source
  server overcloud-controller-0 172.16.20.150:6080 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:6080 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:6080 check fall 5 inter
2000 rise 2
```

3.19. NOVA_OSAPI

ポート番号: 8774

バインド先: internal_api、external

ターゲットのネットワーク/サーバー: overcloud-controller-0、overcloud-controller-1、overcloud-controller-2 上の internal_api

その他の情報:

- 各ターゲットサーバーは、デフォルトのヘルスチェックを使用します。

HAProxy の例:

```
listen nova_osapi
  bind 172.16.20.250:8774
  bind 172.16.23.250:8774
  server overcloud-controller-0 172.16.20.150:8774 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:8774 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:8774 check fall 5 inter
2000 rise 2
```

3.20. NOVA_PLACEMENT

ポート番号: 8778

バインド先: internal_api、external

ターゲットのネットワーク/サーバー: overcloud-controller-0、overcloud-controller-1、overcloud-controller-2 上の internal_api

その他の情報:

- 各ターゲットサーバーは、デフォルトのヘルスチェックを使用します。

HAProxy の例:

```
listen nova_osapi
  bind 172.16.20.250:8778
  bind 172.16.23.250:8778
  server overcloud-controller-0 172.16.20.150:8778 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:8778 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:8778 check fall 5 inter
2000 rise 2
```

3.21. PANKO

ポート番号: 8779

バインド先: internal_api、external

ターゲットのネットワーク/サーバー: overcloud-controller-0、overcloud-controller-1、overcloud-controller-2 上の internal_api

その他の情報:

- 各ターゲットサーバーは、デフォルトのヘルスチェックを使用します。

HAProxy の例:

```
listen nova_osapi
  bind 172.16.20.250:8779
  bind 172.16.23.250:8779
  server overcloud-controller-0 172.16.20.150:8779 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:8779 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:8779 check fall 5 inter
2000 rise 2
```

3.22. REDIS

ポート番号: 6379

バインド先: internal_api (redis サービスの IP)

ターゲットのネットワーク/サーバー: overcloud-controller-0、overcloud-controller-1、overcloud-controller-2 上の internal_api

その他の情報:

- 各ターゲットサーバーは、デフォルトのヘルスチェックを使用します。

- **tcp-check** send/expect シーケンスを使用してヘルスチェックを実行します。送信す文字列は「info\ replication\r\n」、応答は「role:master」です。
- Redis サービスは、認証にパスワードを使用します。たとえば、HAProxy 設定には、**tcp-check** に **AUTH** メソッドと Redis の管理パスワードを使用します。director は、通常無作為なパスワードを生成しますが、カスタムの Redis パスワードを定義することもできます。詳しくは、「ロードバランシングのオプションの設定」を参照してください。
- デフォルトのバランシングメソッドは、ラウンドロビン式ですが、このサービスには、**first** メソッドを使用します。この方法では、利用可能な接続スロットのある最初のサーバーが接続を受けます。

HAProxy の例:

```
listen redis
  bind 172.16.20.249:6379 transparent
  balance first
  option tcp-check
  tcp-check send AUTH\ p@55w0rd!\r\n
  tcp-check send PING\r\n
  tcp-check expect string +PONG
  tcp-check send info\ replication\r\n
  tcp-check expect string role:master
  tcp-check send QUIT\r\n
  tcp-check expect string +OK
  server overcloud-controller-0 172.16.20.150:6379 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:6379 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:6379 check fall 5 inter
2000 rise 2
```

3.23. SAHARA

ポート番号: 8386

バインド先: internal_api、external

ターゲットのネットワーク/サーバー: overcloud-controller-0、overcloud-controller-1、overcloud-controller-2 上の internal_api

その他の情報:

- 各ターゲットサーバーは、デフォルトのヘルスチェックを使用します。
- このサービスは、オプションのオーバークラウドサービスです。インストールするには、オーバークラウドのデプロイメントに **environments/services/sahara.yaml** 環境ファイルを追加してください。

HAProxy の例:

```
listen sahara
  bind 172.16.20.250:8386
  bind 172.16.23.250:8386
  server overcloud-controller-0 172.16.20.150:8386 check fall 5 inter
2000 rise 2
```

```
  server overcloud-controller-1 172.16.20.151:8386 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:8386 check fall 5 inter
2000 rise 2
```

3.24. SWIFT_PROXY_SERVER

ポート番号: 8080

バインド先: storage、external

ターゲットのネットワーク/サーバー: overcloud-controller-0、overcloud-controller-1、overcloud-controller-2 上のストレージ

その他の情報:

- 各ターゲットサーバーは、デフォルトのヘルスチェックを使用します。

HAProxy の例:

```
listen swift_proxy_server
  bind 172.16.23.250:8080
  bind 172.16.21.250:8080
  server overcloud-controller-0 172.16.21.150:8080 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.21.151:8080 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.21.152:8080 check fall 5 inter
2000 rise 2
```

第4章 オーバークラウドの設定

本項では、外部のロードバランサーを使用するオーバークラウドを作成するプロセスを順を追って説明します。これには、ノードの登録、ネットワークの設定、オーバークラウドの作成コマンドに必要な設定オプションが含まれます。

4.1. 環境の設定

このセクションでは、『Red Hat OpenStack Platform 11 director のインストールと使用方法』ガイドからプロセスを抜粋しています。

以下のワークフローを使用して環境を設定します。

- ノード定義のテンプレートを作成して director で空のノードを登録します。
- 全ノードのハードウェアを検査します。
- 手動でロールにノードをタグ付けします。
- フレーバーを作成してロールにタグ付けします。

4.1.1. stack ユーザーの初期化

stack ユーザーとして director ホストにログインし、以下のコマンドを実行して director の設定を初期化します。

```
$ source ~/stackrc
```

このコマンドでは、director の CLI ツールにアクセスする認証情報が含まれる環境変数を設定します。

4.1.2. ノードの登録

ノード定義のテンプレート (**instackenv.json**) は JSON ファイル形式で、ノード登録用のハードウェアおよび電源管理の情報が含まれています。以下に例を示します。

```
{
    "nodes":[
        {
            "mac":[
                "bb:bb:bb:bb:bb:bb"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"pxe_ipmitool",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.205"
        },
        {
            "mac":[
                "cc:cc:cc:cc:cc:cc"
```

```
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"pxe_ipmitool",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.206"
        },
        {
            "mac":[
                "dd:dd:dd:dd:dd:dd"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"pxe_ipmitool",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.207"
        },
        {
            "mac":[
                "ee:ee:ee:ee:ee:ee"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"pxe_ipmitool",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.208"
        }
    ]
}
```

テンプレートの作成後に、stack ユーザーのホームディレクトリー (**/home/stack/instackenv.json**) にファイルを保存してから、director にインポートします。これには、以下のコマンドを実行します。

```
$ openstack baremetal import --json ~/instackenv.json
```

このコマンドでテンプレートをインポートして、テンプレートから director に各ノードを登録します。

カーネルと ramdisk イメージを全ノードに割り当てます。

```
$ openstack baremetal configure boot
```

director でのノードの登録、設定が完了しました。

4.1.3. ノードのハードウェアの検査

ノードの登録後には、各ノードのハードウェア属性を検査します。各ノードのハードウェア属性を検査するには、以下のコマンドを実行します。

```
$ openstack baremetal introspection bulk start
```

重要

このプロセスが最後まで実行されて正常に終了したことを確認してください。ベアメタルの場合には、通常 15 分ほどかかります。

4.1.4. ノードの手動でのタグ付け

各ノードのハードウェアを登録、検査した後には、特定のプロファイルにノードをタグ付けします。これらのプロファイルタグによりノードとフレーバーが照合され、フレーバーがデプロイメントロールに割り当てられます。

ノード一覧を取得して UUID を識別します。

```
$ ironic node-list
```

特定のプロファイルにノードを手動でタグ付けする場合には、各ノードの properties/capabilities パラメーターに profile オプションを追加します。たとえば、2 台のノードをタグ付けしてコントローラープロファイルとコンピュートプロファイルをそれぞれ使用するには、以下のコマンドを実行します。

```
$ ironic node-update 1a4e30da-b6dc-499d-ba87-0bd8a3819bc0 add
properties/capabilities='profile:control,boot_option:local'
$ ironic node-update 6faba1a9-e2d8-4b7c-95a2-c7fbdc12129a add
properties/capabilities='profile:control,boot_option:local'
$ ironic node-update 5e3b2f50-fcd9-4404-b0a2-59d79924b38e add
properties/capabilities='profile:control,boot_option:local'
$ ironic node-update 58c3d07e-24f2-48a7-bbb6-6843f0e8ee13 add
properties/capabilities='profile:compute,boot_option:local'
```

profile:compute と **profile:control** オプションを追加することで、この 2 つのノードがそれぞれのプロファイルにタグ付けされます。

4.2. ネットワークの設定

本項では、オーバークラウドのネットワーク設定について考察します。これには、特定のネットワークトラフィックを使用するようにサービスを分離し、ロードバランシングオプションでオーバークラウドを設定する手順が含まれます。

4.2.1. ネットワークの分離

director は、分離されたオーバークラウドネットワークを設定する方法を提供します。つまり、オーバークラウド環境はネットワークトラフィック種別を異なるネットワークに分離して、個別のネットワークインターフェースまたはボンディングにネットワークトラフィックを割り当てます。分離されたネットワークを設定した後に、director は、OpenStack サービスが分離されたネットワークを使用するように設定します。分離されたネットワークが設定されていない場合には、サービスはすべて、プロビジョニングネットワーク上で実行されます。

まず最初に、オーバークラウドには、ネットワークインターフェースのテンプレートセットが必要です。これらのテンプレートをカスタマイズして、ロールごとにノードのインターフェースを設定します。このテンプレートは YAML 形式の標準の Heat テンプレートです。director にはテンプレートサンプルが含まれているので、すぐに使用を開始することができます。

- **/usr/share/openstack-tripleo-heat-templates/network/config/single-nic-vlans**: このディレクトリーには、ロールごとに VLAN が設定された単一 NIC のテンプレートが含まれます。
- **/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans**: ディレクトリーには、ロールごとにボンディングされた NIC の設定用のテンプレートが格納されます。

ネットワークインターフェースの設定に関する詳細は、『Red Hat OpenStack Platform 11 director のインストールと使用方法』ガイドを参照してください。

次に、ネットワーク環境ファイルを作成します。このファイルは、Heat の環境ファイルで、オーバークラウドのネットワーク環境を記述し、前のセクションのネットワークインターフェース設定テンプレートを参照します。また、IP アドレス範囲と合わせてネットワークのサブネットおよび VLAN を定義します。これらの値をローカルの環境用にカスタマイズします。

このシナリオでは、**/home/stack/network-environment.yaml** として保存した下記のネットワーク環境ファイルを使用します。

```
resource_registry:
  OS::TripleO::BlockStorage::Net::SoftwareConfig:
/home/stack/templates/my-overcloud/network/config/bond-with-
vlans/cinder-storage.yaml
  OS::TripleO::Compute::Net::SoftwareConfig: /home/stack/templates/my-
overcloud/network/config/bond-with-vlans/compute.yaml
  OS::TripleO::Controller::Net::SoftwareConfig:
/home/stack/templates/my-overcloud/network/config/bond-with-
vlans/controller.yaml
  OS::TripleO::ObjectStorage::Net::SoftwareConfig:
/home/stack/templates/my-overcloud/network/config/bond-with-
vlans/swift-storage.yaml
  OS::TripleO::CephStorage::Net::SoftwareConfig:
/home/stack/templates/my-overcloud/network/config/bond-with-vlans/ceph-
storage.yaml

parameter_defaults:
  InternalApiNetCidr: 172.16.20.0/24
  TenantNetCidr: - 172.16.22.0/24
  StorageNetCidr: 172.16.21.0/24
  StorageMgmtNetCidr: 172.16.19.0/24
  ExternalNetCidr: 172.16.23.0/24
  InternalApiAllocationPools: [{'start': '172.16.20.10', 'end':
'172.16.20.200'}]
  TenantAllocationPools: [{'start': '172.16.22.10', 'end':
'172.16.22.200'}]
  StorageAllocationPools: [{'start': '172.16.21.10', 'end':
'172.16.21.200'}]
  StorageMgmtAllocationPools: [{'start': '172.16.19.10', 'end':
'172.16.19.200'}]
  # Leave room for floating IPs in the External allocation pool
  ExternalAllocationPools: [{'start': '172.16.23.10', 'end':
'172.16.23.60'}]
  # Set to the router gateway on the external network
  ExternalInterfaceDefaultRoute: 172.16.23.1
```

```
  # Gateway router for the provisioning network (or Undercloud IP)
  ControlPlaneDefaultRoute: 192.0.2.254
  # The IP address of the EC2 metadata server. Generally the IP of the
Undercloud
  EC2MetadataIp: 192.0.2.1
  # Define the DNS servers (maximum 2) for the overcloud nodes
  DnsServers: ["8.8.8.8","8.8.4.4"]
  InternalApiNetworkVlanID: 201
  StorageNetworkVlanID: 202
  StorageMgmtNetworkVlanID: 203
  TenantNetworkVlanID: 204
  ExternalNetworkVlanID: 100
  # Set to "br-ex" if using floating IPs on native VLAN on bridge br-ex
  NeutronExternalNetworkBridge: "''"
  # Customize bonding options if required
  BondInterfaceOvsOptions:
    "bond_mode=balance-tcp lacp=active other-config:lacp-fallback-
ab=true"
```

ネットワーク環境の設定に関する詳細は、『Red Hat OpenStack Platform 11 director のインストールと使用方法』ガイドを参照してください。

director のホストが **keystone_admin_ssh** の仮想 IP に接続できるように、内部の API ネットワークにアクセスできることを確認してください。

4.2.2. ロードバランシングのオプションの設定

director は、HAProxy が内部で管理する代わりに、外部のロードバランサーが仮想 IP をホストするように設定されたオーバークラウドを作成することができます。この構成では、オーバークラウドのデプロイメントを開始する前に、外部のロードバランサーで、分離されたネットワーク 1 つにつき仮想 IP が 1 つと、Redis サービス用に 1 つ設定済みであることを前提とします。オーバークラウドノードの NIC の設定で許可されている場合には、全く同じ仮想 IP を使用することができます。

外部のロードバランサーは、前章の設定を使用して構成が済んでいます。これらの設定には、director がオーバークラウドノードに割り当て、サービスの設定に使用する IP が含まれます。

以下は、Heat 環境ファイル (**external-lb.yaml**) の例です。これには、外部のロードバランサーが使用するためのオーバークラウドの設定が含まれます。

```
parameter_defaults:
  ControlFixedIPs: [{'ip_address':'192.0.2.250'}]
  PublicVirtualFixedIPs: [{'ip_address':'172.16.23.250'}]
  InternalApiVirtualFixedIPs: [{'ip_address':'172.16.20.250'}]
  StorageVirtualFixedIPs: [{'ip_address':'172.16.21.250'}]
  StorageMgmtVirtualFixedIPs: [{'ip_address':'172.16.19.250'}]
  RedisVirtualFixedIPs: [{'ip_address':'172.16.20.249'}]
  # IPs assignments for the Overcloud Controller nodes. Ensure these
IPs are from each respective allocation pools defined in the network
environment file.
  ControllerIPs:
    external:
    - 172.16.23.150
    - 172.16.23.151
    - 172.16.23.152
    internal_api:
```

```
      - 172.16.20.150
      - 172.16.20.151
      - 172.16.20.152
      storage:
      - 172.16.21.150
      - 172.16.21.151
      - 172.16.21.152
      storage_mgmt:
      - 172.16.19.150
      - 172.16.19.151
      - 172.16.19.152
      tenant:
      - 172.16.22.150
      - 172.16.22.151
      - 172.16.22.152
      # CIDRs
      external_cidr: "24"
      internal_api_cidr: "24"
      storage_cidr: "24"
      storage_mgmt_cidr: "24"
      tenant_cidr: "24"
    RedisPassword: p@55w0rd!
    ServiceNetMap:
      NeutronTenantNetwork: tenant
      CeilometerApiNetwork: internal_api
      AodhApiNetwork: internal_api
      GnocchiApiNetwork: internal_api
      MongoDbNetwork: internal_api
      CinderApiNetwork: internal_api
      CinderIscsiNetwork: storage
      GlanceApiNetwork: storage
      GlanceRegistryNetwork: internal_api
      KeystoneAdminApiNetwork: internal_api
      KeystonePublicApiNetwork: internal_api
      NeutronApiNetwork: internal_api
      HeatApiNetwork: internal_api
      NovaApiNetwork: internal_api
      NovaMetadataNetwork: internal_api
      NovaVncProxyNetwork: internal_api
      SwiftMgmtNetwork: storage_mgmt
      SwiftProxyNetwork: storage
      HorizonNetwork: internal_api
      MemcachedNetwork: internal_api
      RabbitMqNetwork: internal_api
      RedisNetwork: internal_api
      MysqlNetwork: internal_api
      CephClusterNetwork: storage_mgmt
      CephPublicNetwork: storage
      ControllerHostnameResolveNetwork: internal_api
      ComputeHostnameResolveNetwork: internal_api
      BlockStorageHostnameResolveNetwork: internal_api
      ObjectStorageHostnameResolveNetwork: internal_api
      CephStorageHostnameResolveNetwork: storage
```

parameter_defaults セクションには、OpenStack 上の各ネットワークの 仮想 IP と IP の割り当てが含まれます。これらの設定は、ロードバランサー上の各サービスの IP 設定と一致する必要があ

ります。このセクションでは、Redis サービスの管理パスワード (**RedisPassword**) も定義します。また、このセクションには、OpenStack の各サービスを特定のネットワークにマッピングする **ServiceNetMap** パラメーターも含まれます。ロードバランシングの設定には、このサービスを再マッピングする必要があります。

4.3. ロードバランシング向けの SSL 設定

デフォルトでは、オーバークラウドはサービスに対して暗号化されていないエンドポイントを使用します。これは、オーバークラウドの設定には、パブリック API エンドポイントの SSL/TLS を有効化するために追加の環境ファイルが必要という意味です。

注記

外部のロードバランサーに、インストール済みの SSL 証明書と鍵のコピーがあることを確認します。

Heat テンプレートコレクションから **enable-tls.yaml** の環境ファイルをコピーします。

```
$ cp -r /usr/share/openstack-tripleo-heat-
templates/environments/enable-tls.yaml ~/templates/.
```

ファイルを編集して、以下のステップを実行します。

- **parameter_defaults**セクションから **SSLCertificate**、**SSLIntermediateCertificate**、**SSLKey** を削除します。
- **resource_registry** セクションを完全に削除します。
- これで、**parameter_defaults** には、**EndpointMap** パラメーターのみが残るはずです。**EndpointMap** には、HTTPS および HTTP の通信を使用するサービスのマッピングが含まれます。SSL 通信に DNS を使用する場合には、このセクションはデフォルトのままにしますが、SSL 証明書の共通名に IP アドレスを使用する場合には、**CLOUDNAME** をすべて **IP_ADDRESS** に置き換えます。このステップを実行するには、以下のコマンドを使用します。

  ```
  $ sed -i 's/CLOUDNAME/IP_ADDRESS/' ~/templates/enable-tls.yaml
  ```

  重要

  IP_ADDRESS または **CLOUDNAME** は、実際の値に置き換えないでください。Heat により、オーバークラウドの作成時にこれらの変数が適切な値に置き換えられます。

自己署名証明書を使用する場合または、証明書の署名者がオーバークラウドのイメージにあるデフォルトのトラストストアに含まれない場合には、証明書をオーバークラウドのイメージに注入します。Heat テンプレートコレクションから **inject-trust-anchor.yaml** 環境ファイルをコピーします。

```
$ cp -r /usr/share/openstack-tripleo-heat-
templates/environments/inject-trust-anchor.yaml ~/templates/.
```

このファイルを編集して、下記のパラメーターに以下の変更を加えます。

SSLRootCertificate

SSLRootCertificate パラメーターにルート認証局ファイルの内容をコピーします。以下に例を示します。

```
parameter_defaults:
  SSLRootCertificate: |
  -----BEGIN CERTIFICATE-----

MIIDgzCCAmugAwIBAgIJAKk46qw6ncJaMA0GCSqGSIb3DQEBCwUAMFgxCzAJBgNV
  ...
  sFW3S2roS4X0Af/kSSD8mlBBTFTCMBAj6rtLBKLaQbIxEpIzrgvp
  -----END CERTIFICATE-----
```

重要

この認証局のコンテンツで、新しく追加する行は、すべて同じレベルにインデントする必要があります。

OS::TripleO::NodeTLSCAData

OS::TripleO::NodeTLSCAData: のリソース URL を絶対 URL に変更します。

```
resource_registry:
     OS::TripleO::NodeTLSCAData: /usr/share/openstack-tripleo-
heat-templates/puppet/extraconfig/tls/ca-inject.yaml
```

DNS ホスト名を使用して SSL/TLS でオーバークラウドにアクセスする場合は、新しい環境ファイル (**~/templates/cloudname.yaml**) を作成して、オーバークラウドのエンドポイントのホスト名を定義します。以下のパラメーターを使用してください。

CloudName

オーバークラウドエンドポイントの DNS ホスト名

DnsServers

使用する DNS サーバー一覧。設定済みの DNS サーバーには、パブリック API の IP アドレスに一致する設定済みの CloudName へのエントリーが含まれていなければなりません。

このファイルの内容の例は以下のとおりです。

```
parameter_defaults:
 CloudName: overcloud.example.com
 DnsServers: 10.0.0.1
```

「オーバークラウドの作成」に記載のデプロイメントのコマンド (**openstack overcloud deploy**) は、**-e** オプションを使用して環境ファイルを追加します。以下の順番にこのセクションから環境ファイルを追加します。

- SSL/TLS を有効化する環境ファイル (**enable-tls.yaml**)
- DNS ホスト名を設定する環境ファイル (**cloudname.yaml**)
- ルート認証局を注入する環境ファイル (**inject-trust-anchor.yaml**)

例

```
$ openstack overcloud deploy --templates [...] -e
/home/stack/templates/enable-tls.yaml -e ~/templates/cloudname.yaml -e
~/templates/inject-trust-anchor.yaml
```

4.4. オーバークラウドの作成

外部のロードバランサーを使用するオーバークラウドを作成するには、**openstack overcloud deploy** コマンドに追加の引数を指定する必要があります。以下に例を示します。

```
$ openstack overcloud deploy --templates -e /usr/share/openstack-
tripleo-heat-templates/environments/network-isolation.yaml -e
~/network-environment.yaml  -e /usr/share/openstack-tripleo-heat-
templates/environments/external-loadbalancer-vip.yaml -e ~/external-
lb.yaml --control-scale 3 --compute-scale 1 --control-flavor control --
compute-flavor compute [ADDITIONAL OPTIONS]
```

上記のコマンドは、以下のオプションを使用します。

- **--templates**: デフォルトの Heat テンプレートコレクションからオーバークラウドを作成します。
- **-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml**: オーバークラウドデプロイメントに別の環境ファイルを追加します。この場合は、リソースのネットワーク分離の設定を初期化する環境ファイルです。
- **-e ~/network-environment.yaml**: オーバークラウドデプロイメントに追加の環境ファイルを追加します。この場合は、以前に作成したネットワーク環境ファイルです。
- **-e /usr/share/openstack-tripleo-heat-templates/environments/external-loadbalancer-vip.yaml**: オーバークラウドデプロイメントに追加の環境ファイルを追加します。この場合は、外部のロードバランシング設定を初期化する環境ファイルです。この環境ファイルは、ネットワーク設定ファイルの後に追加する必要がある点に注意してください。
- **-e ~/external-lb.yaml**: オーバークラウドデプロイメントに追加の環境ファイルを追加します。この場合は、外部のロードバランサー設定が記載された環境ファイルです。この環境ファイルは、ネットワーク設定ファイルの後に追加する必要がある点に注意してください。
- **--control-scale 3**: コントローラーノードを 3 つにスケーリングします。
- **--compute-scale 3**: コンピュートノードを 3 つにスケーリングします。
- **--control-flavor control**: コントローラーノードに特定のフレーバーを使用します。
- **--compute-flavor compute**: コンピュートノードに特定のフレーバーを使用します。

注記

オプションの完全な一覧を表示するには、以下のコマンドを実行します。

```
$ openstack help overcloud deploy
```

パラメーターの例については、『Red Hat OpenStack Platform 11 director インストールと使用方法』ガイドを参照してください。

オーバークラウドの作成プロセスが開始され、director によりノードがプロビジョニングされます。このプロセスは完了するまで多少時間がかかります。オーバークラウドの作成のステータスを確認するには、stack ユーザーとして別のターミナルを開き、以下を実行します。

```
$ source ~/stackrc
$ heat stack-list --show-nested
```

4.5. オーバークラウドへのアクセス

director は、director ホストからオーバークラウドに対話するための設定を行い、認証をサポートするスクリプトを作成して、stack ユーザーのホームディレクトリーにこのファイル (overcloudrc) を保存します。このファイルを使用するには、以下のコマンドを実行します。

```
$ source ~/overcloudrc
```

これで、director のホストの CLI からオーバークラウドと対話するために必要な環境変数が読み込まれます。director のホストとの対話に戻るには、以下のコマンドを実行します。

```
$ source ~/stackrc
```

4.6. オーバークラウドの設定の完了

これで、高度なオーバークラウドの作成が終了しました。

高可用性クラスターのフェンシングに関する情報は、『Red Hat OpenStack Platform 11 director のインストールと使用方法』ガイドを参照してください。

作成後の機能に関する情報は、『Red Hat OpenStack Platform 11 director のインストールと使用方法』ガイドを参照してください。

付録A デフォルトの HAPROXY 設定の例

以下は、オーバークラウドのコントローラーノード上の HAProxy 用のデフォルトの設定ファイルの例です。このファイルは、各コントローラーノード上の **/etc/haproxy/haproxy.conf** にあります。

```
global
  daemon
  group  haproxy
  log  /dev/log local0
  maxconn  10000
  pidfile  /var/run/haproxy.pid
  user  haproxy

defaults
  log  global
  mode  tcp
  retries  3
  timeout  http-request 10s
  timeout  queue 1m
  timeout  connect 10s
  timeout  client 1m
  timeout  server 1m
  timeout  check 10s

listen aodh
  bind 172.16.20.250:8042
  bind 172.16.20.250:8042
  mode http
  server overcloud-controller-0 172.16.20.150:8042 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:8042 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.252:8042 check fall 5 inter
2000 rise 2

listen ceilometer
  bind 172.16.20.250:8777
  bind 172.16.23.250:8777
  server overcloud-controller-0 172.16.20.150:8777 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:8777 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:8777 check fall 5 inter
2000 rise 2

listen cinder
  bind 172.16.20.250:8776
  bind 172.16.23.250:8776
  server overcloud-controller-0 172.16.20.150:8776 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:8776 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:8776 check fall 5 inter
2000 rise 2
```

```
listen glance_api
  bind 172.16.23.250:9292
  bind 172.16.21.250:9292
  server overcloud-controller-0 172.16.21.150:9292 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.21.151:9292 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.21.152:9292 check fall 5 inter
2000 rise 2

listen glance_registry
  bind 172.16.20.250:9191
  server overcloud-controller-0 172.16.20.150:9191 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:9191 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:9191 check fall 5 inter
2000 rise 2

listen gnocchi
  bind 172.16.23.250:8041
  bind 172.16.21.250:8041
  mode http
  server overcloud-controller-0 172.16.20.150:8041 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:8041 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:8041 check fall 5 inter
2000 rise 2

listen heat_api
  bind 172.16.20.250:8004
  bind 172.16.23.250:8004
  mode http
  server overcloud-controller-0 172.16.20.150:8004 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:8004 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:8004 check fall 5 inter
2000 rise 2

listen heat_cfn
  bind 172.16.20.250:8000
  bind 172.16.23.250:8000
  server overcloud-controller-0 172.16.20.150:8000 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.152:8000 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.151:8000 check fall 5 inter
2000 rise 2

listen heat_cloudwatch
  bind 172.16.20.250:8003
  bind 172.16.23.250:8003
  server overcloud-controller-0 172.16.20.150:8003 check fall 5 inter
```

```
2000 rise 2
  server overcloud-controller-1 172.16.20.151:8003 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:8003 check fall 5 inter
2000 rise 2

listen horizon
  bind 172.16.20.250:80
  bind 172.16.23.250:80
  mode http
  cookie SERVERID insert indirect nocache
  server overcloud-controller-0 172.16.20.150:80 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:80 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:80 check fall 5 inter
2000 rise 2

listen keystone_admin
  bind 172.16.23.250:35357
  bind 172.16.20.250:35357
  server overcloud-controller-0 172.16.20.150:35357 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:35357 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:35357 check fall 5 inter
2000 rise 2

listen keystone_admin_ssh
  bind 172.16.20.250:22
  server overcloud-controller-0 172.16.20.150:22 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:22 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:22 check fall 5 inter
2000 rise 2

listen keystone_public
  bind 172.16.20.250:5000
  bind 172.16.23.250:5000
  server overcloud-controller-0 172.16.20.150:5000 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:5000 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:5000 check fall 5 inter
2000 rise 2

listen mysql
  bind 172.16.20.250:3306
  option tcpka
  option httpchk
  stick on dst
  stick-table type ip size 1000
  timeout client 0
  timeout server 0
  server overcloud-controller-0 172.16.20.150:3306 backup check fall 5
```

```
inter 2000 on-marked-down shutdown-sessions port 9200 rise 2
  server overcloud-controller-1 172.16.20.151:3306 backup check fall 5
inter 2000 on-marked-down shutdown-sessions port 9200 rise 2
  server overcloud-controller-2 172.16.20.152:3306 backup check fall 5
inter 2000 on-marked-down shutdown-sessions port 9200 rise 2

listen neutron
  bind 172.16.20.250:9696
  bind 172.16.23.250:9696
  server overcloud-controller-0 172.16.20.150:9696 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:9696 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:9696 check fall 5 inter
2000 rise 2

listen nova_ec2
  bind 172.16.20.250:8773
  bind 172.16.23.250:8773
  server overcloud-controller-0 172.16.20.150:8773 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:8773 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:8773 check fall 5 inter
2000 rise 2

listen nova_metadata
  bind 172.16.20.250:8775
  server overcloud-controller-0 172.16.20.150:8775 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:8775 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:8775 check fall 5 inter
2000 rise 2

listen nova_novncproxy
  bind 172.16.20.250:6080
  bind 172.16.23.250:6080
  balance source
  server overcloud-controller-0 172.16.20.150:6080 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:6080 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:6080 check fall 5 inter
2000 rise 2

listen nova_osapi
  bind 172.16.20.250:8774
  bind 172.16.23.250:8774
  server overcloud-controller-0 172.16.20.150:8774 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:8774 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:8774 check fall 5 inter
2000 rise 2
```

```
listen nova_placement
  bind 172.16.20.250:8778
  bind 172.16.23.250:8778
  mode http
  server overcloud-controller-0 172.16.20.150:8778 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:8778 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:8778 check fall 5 inter
2000 rise 2

listen panko
  bind 172.16.20.250:8779 transparent
  bind 172.16.23.250:8779 transparent
  server overcloud-controller-0 172.16.20.150:8779 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:8779 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:8779 check fall 5 inter
2000 rise 2

listen redis
  bind 172.16.20.249:6379
  balance first
  option tcp-check
  tcp-check send AUTH\ p@55w0rd!\r\n
  tcp-check send PING\r\n
  tcp-check expect string +PONG
  tcp-check send info\ replication\r\n
  tcp-check expect string role:master
  tcp-check send QUIT\r\n
  tcp-check expect string +OK
  server overcloud-controller-0 172.16.20.150:6379 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.20.151:6379 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.20.152:6379 check fall 5 inter
2000 rise 2

listen swift_proxy_server
  bind 172.16.23.250:8080
  bind 172.16.21.250:8080
  server overcloud-controller-0 172.16.21.150:8080 check fall 5 inter
2000 rise 2
  server overcloud-controller-1 172.16.21.151:8080 check fall 5 inter
2000 rise 2
  server overcloud-controller-2 172.16.21.152:8080 check fall 5 inter
2000 rise 2
```