Red Hat OpenStack Platform 16.1

オーバークラウド用の外部ロードバランサー

外部ロードバランサーを使用する Red Hat OpenStack Platform 環境の設定

Last Updated: 2023-11-07

Red Hat OpenStack Platform 16.1 オーバークラウド用の外部ロードバランサー

外部ロードバランサーを使用する Red Hat OpenStack Platform 環境の設定

OpenStack Team
rhos-docs@redhat.com

法律上の通知

概要

オーバークラウド用に外部ロードバランサーを使用するように Red Hat OpenStack Platform (RHOSP) 環境を設定します。これには、ロードバランサーの設定ガイドラインおよび Red 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

多様性を受け入れるオープンソースの強化

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

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 外部ロードバランサーを使用するオーバークラウドの設定

Red Hat OpenStack Platform (RHOSP) では、オーバークラウドは複数のコントローラーノードを組み合わせて高可用性クラスターとして使用し、OpenStack サービスのオペレーションパフォーマンスを最大限に保ちます。クラスターにより、OpenStack サービスの負荷分散が行われ、コントローラーノードに均等にトラフィックを分配して、各ノードのサーバーで過剰負荷を軽減します。

デフォルトでは、オーバークラウドは HAProxy と呼ばれるオープンソースツールを使用して負荷分散を管理します。HAProxy は、OpenStack サービスを実行するコントローラーノードへのトラフィックの負荷分散を行います。**haproxy** パッケージには、着信トラフィックをリッスンする **haproxy** デーモンが含まれ、ロギング機能とサンプル設定が含まれます。

また、オーバークラウドは高可用性リソースマネージャー Pacemaker を使用して、高可用性サービスとして HAProxy を制御します。つまり、HAProxy は各コントローラーノードで実行され、各設定で定義するルールセットに従ってトラフィックを分散します。

外部のロードバランサーを使用して、この負荷分散を実行することも可能です。たとえば、組織で、コントローラーノードへのトラフィックの分散処理に、専用のハードウェアベースのロードバランサーを使用する場合などです。外部ロードバランサーおよびオーバークラウドの作成の設定を定義するには、以下のプロセスを実施します。

1. 外部ロードバランサーをインストールし、設定します。
2. オーバークラウドを heat テンプレートパラメーターを使用して設定およびデプロイし、オーバークラウドを外部ロードバランサーと統合します。これには、ロードバランサーと潜在的なノードの IP アドレスが必要です。

外部ロードバランサーを使用するようにオーバークラウドを設定する前に、オーバークラウドに高可用性をデプロイして実行していることを確認してください。

1.1. 外部ロードバランサーに向けた環境の準備

外部ロードバランサー用に環境を準備するには、まずノード定義のテンプレートを作成し、空のノードを director に登録します。次に、すべてのノードのハードウェアを検査し、手動でノードをプロファイルにタグ付けします。

以下のワークフローを使用して、環境を準備します。

- ノード定義のテンプレートを作成し、空のノードを Red Hat OpenStack Platform director に登録します。ノード定義のテンプレート **instackenv.json** は JSON ファイル形式で、ノードを登録するためのハードウェアおよび電源管理情報が含まれています。
- 全ノードのハードウェアを検査します。これにより、すべてのノードが manageable の状態になります。
- 手動でノードをプロファイルにタグ付けします。これらのプロファイルタグは、ノードをフレーバーに一致させます。その後、フレーバーはデプロイメントロールに割り当てられます。

手順

1. director ホストに **stack** ユーザーとしてログインし、source コマンドで director の認証情報を読み込みます。

```
$ source ~/stackrc
```

2. ノード定義のテンプレート **instackenv.json** を作成し、以下のサンプルをコピーして実際の環境に応じて編集します。

```
{
    "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"
```

```
      }
    ]
}
```

3. **stack** ユーザーのホームディレクトリーにファイルを保存し (**/home/stack/instackenv.json**)、director にインポートして director にノードを登録します。

```
$ openstack overcloud node import ~/instackenv.json
```

4. カーネルと ramdisk イメージを全ノードに割り当てます。

```
$ openstack overcloud node configure
```

5. 各ノードのハードウェア属性を検証します。

```
$ openstack overcloud node introspect --all-manageable
```

重要

ノードは manageable の状態である必要があります。このプロセスを必ず完了させてください。ベアメタルノードの場合は、通常 15 分ほどかかります。

6. ノードのリストを取得して UUID を特定します。

```
$ openstack baremetal node list
```

7. 各ノードの **properties/capabilities** パラメーターに profile オプションを追加して、各ノードを特定のプロファイルに手動でタグ付けします。たとえば、3 つのノードが Controller プロファイルを使用し、1 つのノードが Compute プロファイルを使用するようにタグ付けするには、以下のコマンドを使用します。

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

profile:compute および **profile:control** オプションは、ノードをそれぞれのプロファイルにタグ付けします。

関連情報

- オーバークラウドのプランニング

1.2. 外部ロードバランサー用のオーバークラウドネットワークの設定

オーバークラウド用にネットワークを設定するには、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てから、ローカル環境用にネットワーク環境ファイルを設定します。このファイルは、オーバークラウドのネットワーク環境を記述し、ネットワークインターフェイス設定テンプレート

を参照して、ネットワークのサブネットと VLAN および IP アドレス範囲を定義する heat 環境ファイルです。

手順

1. 各ロールのノードインターフェイスを設定するには、以下のネットワークインターフェイステンプレートをカスタマイズします。
   - ロールごとに VLAN を持つ単一の NIC を設定するには、以下のディレクトリーのサンプルテンプレートを使用します。

     ```
     /usr/share/openstack-tripleo-heat-templates/network/config/single-nic-vlans
     ```

   - ロールごとにボンディングされた NIC を設定するには、以下のディレクトリーのサンプルテンプレートを使用します。

     ```
     /usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans
     ```

2. **/home/stack/network-environment.yaml** からファイルをコピーし、実際の環境に応じて設定を編集して、ネットワークの環境ファイルを作成します。

関連情報

- 基本的なネットワーク分離
- カスタムコンポーザブルネットワーク
- カスタムネットワークインターフェイステンプレート
- オーバークラウドネットワーク

1.3. 外部ロードバランサーの環境ファイルの作成

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デプロイするには、必要な設定で新しい環境ファイルを作成します。以下の例では、オーバークラウドのデプロイメントを開始する前に、複数の仮想 IP が外部ロードバランサーに設定されます (分離したネットワークごとに 1 つの仮想 IP、および Redis サービス用に 1 つの仮想 IP)。オーバークラウドノードの NIC 設定がその設定をサポートしている場合には、一部の仮想 IP が同一になる場合があります。

手順

- 以下のサンプル環境ファイル **external-lb.yaml** を使用して環境ファイルを作成し、実際の環境に基づいて設定を編集します。

```
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
```

注記

- **parameter_defaults** セクションには、各ネットワークの仮想 IP および IP の割り当てが含まれます。これらの設定は、ロードバランサーのサービスごとに同じ IP 設定と一致する必要があります。
- **parameter_defaults** セクションでは、Redis サービスの管理パスワードを定義し (RedisPassword)、各 OpenStack サービスを特定のネットワークにマッピングする **ServiceNetMap** パラメーターが含まれます。負荷分散設定には、このサービスの再マッピングが必要です。

1.4. 外部負荷分散のための SSL の設定

外部ロードバランサーの暗号化されたエンドポイントを設定するには、エンドポイントへのアクセスに SSL を有効にする追加の環境ファイルを作成し、外部の負荷分散サーバーに SSL 証明書および鍵のコピーをインストールします。デフォルトでは、オーバークラウドは暗号化されていないエンドポイントサービスを使用します。

前提条件

- IP アドレスまたはドメイン名を使用してパブリックエンドポイントにアクセスする場合は、以下の環境ファイルのいずれかを選択してオーバークラウドのデプロイメントにしていること。
  - ドメインネームサービス (DNS) を使用してパブリックエンドポイントにアクセスするには、ファイル **/usr/share/openstack-tripleo-heat-templates/environments/tls-endpoints-public-dns.yaml** を使用します。
  - IP アドレスを使用してパブリックエンドポイントにアクセスするには、ファイル **/usr/share/openstack-tripleo-heat-templates/environments/tls-endpoints-public-ip.yaml** を使用します。

手順

1. 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イメージのデフォルトのトラストストアにない場合は、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て、証明書をオーバークラウドイメージに注入します。

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

2. ファイルをテキストエディターで開き、ルート認証局ファイルの内容を **SSLRootCertificate** パラメーターにコピーします。

```
parameter_defaults:
  SSLRootCertificate: |
  -----BEGIN CERTIFICATE-----
  MIIDgzCCAmugAwIBAgIJAKk46qw6ncJaMA0GCSqGSIb3DQEBCwUAMFgxCzAJBgNV
  ...
  sFW3S2roS4X0Af/kSSD8mlBBTFTCMBAj6rtLBKLaQbIxEpIzrgvp
  -----END CERTIFICATE-----
```

重要

この認証局の内容に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。

3. **OS::TripleO::NodeTLSCAData:** パラメーターのリソース 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
```

4. オプション: 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、以下のパラメーターでオーバークラウドエンドポイントのホスト名を定義します。

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

以下の値を、環境の実際の値に置き換えます。

- **CloudName**: **overcloud.example.com** を、オーバークラウドエンドポイントの DNS ホスト名に置き換えてください。
- **DnsServers**: 使用する DNS サーバーのリスト。設定する DNS サーバーには、パブリック API の IP に一致する設定済みの **CloudName** のエントリーが含まれていなければなりません。

1.5. 外部ロードバランサーを使用するオーバークラウドのデプロイ

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デプロイするには、外部ロードバランサー用の追加の環境ファイルおよび設定ファイルを指定して、**openstack overcloud deploy** を実行します。

前提条件

- 外部ロードバランサー用に環境が準備されている。環境の準備方法の詳細は、「外部ロードバランサーに向けた環境の準備」を参照してください。
- 外部ロードバランサー用に、オーバークラウドのネットワークが設定されている。ネットワークの設定方法に関する情報は、「外部ロードバランサー用のオーバークラウドネットワークの設定」を参照してください。
- 外部ロードバランサーの環境ファイルが準備されている。環境ファイルの作成方法についての情報は、「外部ロードバランサーの環境ファイルの作成」を参照してください。
- 外部負荷分散用に、SSL が設定されている。外部負荷分散用に SSL を設定する方法は、「外部負荷分散のための SSL の設定」を参照してください。

手順

1. 外部ロードバランサー用のすべての環境ファイルおよび設定ファイルと共にオーバークラウドをデプロイします。

```
$ 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 /
  -e <SSL/TLS endpoint environment file> /
  -e <DNS hostname environment file> /
  -e <root certificate injection environment file> /
  -e <additional_options_if_needed>
```

山かっこ **<>** 内の値を、実際の環境用に定義したファイルパスに置き換えます。

重要

この例に記載されている順序で、ネットワーク環境ファイルをコマンドに追加する必要があります。

このコマンドには、以下の環境ファイルが含まれます。

- **network-isolation.yaml**: ネットワーク分離設定ファイル
- **network-environment.yaml**: ネットワーク設定ファイル。
- **external-loadbalancer-vip.yaml**: 外部負荷分散仮想 IP アドレス設定ファイル
- **external-lb.yaml**: 外部ロードバランサー設定ファイル。このファイルに以下のオプションを設定して、実際の環境用に値を調整することもできます。
  - **--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。
- SSL/TLS 環境ファイル:
  - **SSL/TLS endpoint environment file**: パブリックエンドポイントへの接続方法を定義する環境ファイル。**tls-endpoints-public-dns.yaml** または **tls-endpoints-public-ip.yaml** を使用します。
  - (オプション) **DNS hostname environment file**: DNS ホスト名を設定するための環境ファイル。
  - **Root certificate injection environment file**: ルート認証局を挿入するための環境ファイル。

オーバークラウドのデプロイメントプロセス中に、Red Hat OpenStack Platform 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。

2. オーバークラウドのデプロイメントステータスを確認するには、以下のコマンドを入力します。

```
$ source ~/stackrc
$ openstack stack list --nested
```

1.6. 関連情報

- Load balancing traffic with HAProxy

第2章 設定例: 外部 HAPROXY ロードバランサーを使用したオーバークラウド

以下の設定例は、外部負荷分散を提供するフェデレーションされた HAProxy サーバーを使用するオーバークラウドを示しています。使用している環境の要件に基づいて、別の外部ロードバランサーを選択できます。

設定例では、以下の要素が含まれます。

- HAProxy を実行する外部の負荷分散サーバー。
- 1 つの Red Hat OpenStack Platform (RHOSP) director ノード。
- 高可用性クラスター設定の 3 つのコントローラーノードおよび 1 つのコンピュートノードで設定されるオーバークラウド。
- VLAN を使用したネットワーク分離。

この例では、ネットワークごとに以下の IP アドレスの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**

これらの IP 範囲には、コントローラーノードの IP 割り当てと、ロードバランサーが OpenStack サービスにバインドする仮想 IP が含まれます。

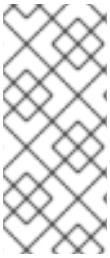
2.1. HAPROXY 設定ファイルの例

サンプルファイルは、内部の HAProxy 設定パラメーターを示しています。サンプルの設定パラメーターを、外部ロードバランサーを設定するためのベースとして使用することができます。

HAProxy 設定ファイルには、以下のセクションが含まれます。

- グローバル設定
- デフォルト設定
- サービス設定

director は、この設定を、コンテナー化されていない環境用に各コントローラーノードの **/etc/haproxy/haproxy.conf** ファイルで提供し、コンテナー化環境用に **/var/lib/config-data/puppet-generated/haproxy/etc/haproxy/haproxy.cfg** ファイルで提供します。



注記

グローバル、デフォルト、およびサービスパラメーターに加えて、他の HAProxy パラメーターも設定する必要があります。HAProxy パラメーターについての詳細は、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ている任意のシステムの **/usr/share/doc/haproxy-*/configuration.txt** にある **HAProxy 設定マニュアル**を参照してください。

HAProxy 設定ファイルの例

```
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
```

2.1.1. グローバル設定パラメーター: HAProxy 設定ファイルの例

グローバル設定パラメーターセクションは、ロードバランサーに関するプロセス全体のパラメーターセットを定義します。設定ファイルのパラメーター例を使用して、外部ロードバランサーを設定することができます。実際の環境に応じて、パラメーターの値を調整します。

グローバル設定パラメーター

```
global
  daemon
  user  haproxy
  group  haproxy
```

```
  log  /dev/log local0
  maxconn  10000
  pidfile  /var/run/haproxy.pid
```

この例は、以下のパラメーターを示しています。

- **daemon**: バックグラウンドプロセスとして実行します。
- **user haproxy** および **group haproxy**: プロセスを所有する Linux ユーザーおよび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**: プロセスへの同時接続の最大数を設定します。
- **pidfile**: プロセス ID に使用するファイルを設定します。

2.1.2. デフォルト値設定パラメーター: HAProxy 設定ファイルの例

デフォルト値の設定パラメーターセクションは、外部ロードバランサーサービスの実行時に使用するデフォルト値のセットを定義します。設定ファイルのパラメーター例を使用して、外部ロードバランサーを設定することができます。実際の環境に応じて、パラメーターの値を調整します。

デフォルト値の設定パラメーター

```
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
```

この例は、以下のパラメーターを示しています。

- **log**: サービスのロギングを有効にします。値 **global** 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定義します。ここでは、デフォルトは TCP です。
- **retries**: サーバーで実行するリトライ回数を設定します。これを超えると、接続の失敗が報告されます。
- **timeout**: 特定の機能を待機する最大の時間を設定します。たとえば、**timeout http-request** は、完全な HTTP 要求を待つ最大の時間を設定します。

2.1.3. サービスレベルの設定パラメーター: HAProxy 設定ファイルの例

サービスレベル設定パラメーターのセクションでは、特定の Red Hat OpenStack Platform (RHOSP) サービスへのトラフィックの負荷分散時に使用するパラメーターのセットを定義します。設定ファイルのパラメーター例を使用して、外部ロードバランサーを設定することができます。実際の環境に応じてパラメーターの値を調整し、負荷分散を行うサービスごとにセクションをコピーします。

サービスレベル設定パラメーター

以下の例は、**ceilometer** サービスの設定パラメーターを示しています。

```
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
```

負荷分散を行う各サービスは、設定ファイルのセクションに対応している必要があります。各サービスの設定には、以下のパラメーターが含まれます。

- **listen**: 要求をリッスンするサービスの名前。
- **listen**: サービスがリッスンする IP アドレスおよび TCP ポート番号。各サービスは、異なるネットワークトラフィック種別を表す異なるアドレスをバインドします。
- **server**: サービスを提供する各サーバーの名前、サーバーの IP アドレスおよびリッスンするポート、ならびに接続パラメーター。
- **check**: (オプション) ヘルスチェックを有効にします。
- **fall 5**: (オプション) ヘルスチェックに 5 回失敗すると、サービスはオフラインとみなされます。
- **inter 2000**: (オプション) 連続する 2 回のヘルスチェックの間隔を 2000 ミリ秒 (2 秒) に設定します。
- **rise 2**: (オプション) ヘルスチェックに 2 回成功すると、サービスは稼働状態とみなされます。

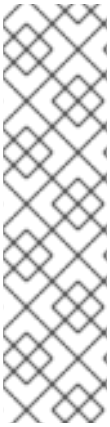
ceilometer の例では、サービスは ceilometer サービスが提供される IP アドレスとポートを **172.16.20.2500:8777** および **172.16.23.250:8777** と識別します。HAProxy は、これらのアドレスの要求を **overcloud-controller-0** (172.16.20.150:8777)、**overcloud-controller-1** (172.16.20.151:8777)、または **overcloud-controller-2** (172.16.0.152:8777) に転送します。

関連情報

- 「負荷分散を使用するサービスの設定パラメーター」

2.2. 負荷分散を使用するサービスの設定パラメーター

負荷分散を使用するオーバークラウドの各サービスについて、以下の例を参考にして外部のロードバランサーを設定します。実際の環境に応じてパラメーターの値を調整し、負荷分散を行うサービスごとにセクションをコピーします。



注記

ほとんどのサービスは、デフォルトのヘルスチェック設定を使用します。

- 連続する 2 回のヘルスチェックの間隔を 2000 ミリ秒 (2 秒) に設定します。
- ヘルスチェックに 2 回成功すると、サービスは稼働状態とみなされます。
- ヘルスチェック に 5 回失敗すると、サービスはオフラインとみなされます。

各サービスは、そのサービスの**その他の情報** セクションで、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します。

aodh

ポート番号: 8042

バインド先: internal_api、 external

ターゲットネットワークまたはサーバー: overcloud-controller-0、overcloud-controller-1、および overcloud-controller-2 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
```

ceilometer

ポート番号: 8777

バインド先: internal_api、 external

ターゲットネットワークまたはサーバー: overcloud-controller-0、overcloud-controller-1、および overcloud-controller-2 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
```

cinder

ポート番号: 8776

バインド先: internal_api、external

ターゲットネットワークまたはサーバー: overcloud-controller-0、overcloud-controller-1、および overcloud-controller-2 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
```

glance_api

ポート番号: 9292

バインド先: storage、external

ターゲットネットワークまたはサーバー: overcloud-controller-0、overcloud-controller-1、および overcloud-controller-2 の storage

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
```

glance_registry

ポート番号: 9191

バインド先: internal_api

ターゲットネットワークまたはサーバー: overcloud-controller-0、overcloud-controller-1、および overcloud-controller-2 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
```

gnocchi

ポート番号: 8041

バインド先: internal_api、external

ターゲットネットワークまたはサーバー: overcloud-controller-0、overcloud-controller-1、および overcloud-controller-2 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
```

heat_api

ポート番号: 8004

バインド先: internal_api、external

ターゲットネットワークまたはサーバー: overcloud-controller-0、overcloud-controller-1、および overcloud-controller-2 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
```

heat_cfn

ポート番号: 8000

バインド先: internal_api、external

ターゲットネットワークまたはサーバー: overcloud-controller-0、overcloud-controller-1、および overcloud-controller-2 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
```

heat_cloudwatch

ポート番号: 8003

バインド先: internal_api、external

ターゲットネットワークまたはサーバー: overcloud-controller-0、overcloud-controller-1、および overcloud-controller-2 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
```

horizon

ポート番号: 80

バインド先: internal_api、external

ターゲットネットワークまたはサーバー: overcloud-controller-0、overcloud-controller-1、および overcloud-controller-2 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 との対話にクッキーベースの永続性を使用します。

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
```

keystone_admin

ポート番号: 35357

バインド先: internal_api、external

ターゲットネットワークまたはサーバー: overcloud-controller-0、overcloud-controller-1、および overcloud-controller-2 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
```

keystone_admin_ssh

ポート番号: 22

バインド先: internal_api

ターゲットネットワークまたはサーバー: overcloud-controller-0、overcloud-controller-1、および overcloud-controller-2 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
```

keystone_public

ポート番号: 5000

バインド先: internal_api、external

ターゲットネットワークまたはサーバー: overcloud-controller-0、overcloud-controller-1、および overcloud-controller-2 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
```

mysql

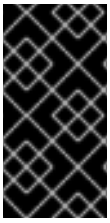
ポート番号: 3306

バインド先: internal_api

ターゲットネットワークまたはサーバー: overcloud-controller-0、overcloud-controller-1、および overcloud-controller-2 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 が使用されます。
- このサービスは、1度に1つのサーバーにのみ負荷分散されます。
- 各サーバーは、他のすべてのバックアップ以外のサーバーが使用できない場合にのみ、負荷分散で使用されます。
- サーバーがオフラインの場合、すべての接続は即時に終了します。
- 両側で TCP キープアライブパケットパケットの送信を有効にする必要があります。
- サーバーの正常性を確認するには、HTTP プロトコルを有効にする必要があります。
- IP アドレスを格納するためにスティッキネステーブルを設定すると、永続性を維持するのに役立ちます。



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 はアクティブ/アクティブ設定をサポートしていますが、ロック競合を回避するために、ロードバランサーにより強制されるアクティブ/パッシブ設定を使用する必要があります。

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
```

neutron

ポート番号: 9696

バインド先: internal_api、external

ターゲットネットワークまたはサーバー: overcloud-controller-0、overcloud-controller-1、および overcloud-controller-2 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
```

nova_ec2

ポート番号: 8773

バインド先: internal_api、external

ターゲットネットワークまたはサーバー: overcloud-controller-0、overcloud-controller-1、および overcloud-controller-2 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
```

nova_metadata

ポート番号: 8775

バインド先: internal_api

ターゲットネットワークまたはサーバー: overcloud-controller-0、overcloud-controller-1、および overcloud-controller-2 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
```

nova_novncproxy

ポート番号: 6080

バインド先: internal_api、external

ターゲットネットワークまたはサーバー: overcloud-controller-0、overcloud-controller-1、および overcloud-controller-2 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。
- デフォルトの負荷分散方法はラウンドロビンです。ただし、このサービスでは **source** メソッドが使用されます。このメソッドは、ソース IP アドレスをハッシュし、実行中のサーバーの重みの合計で除算します。このメソッドは、リクエストを受信するサーバーも指定し、サーバーが終了/起動しない限り、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。実行中のサーバー数が変更されたためにハッシュの結果が変更された場合、ロードバランサーは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
```

nova_osapi

ポート番号: 8774

バインド先: internal_api、external

ターゲットネットワークまたはサーバー: overcloud-controller-0、overcloud-controller-1、および overcloud-controller-2 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
```

nova_placement

ポート番号: 8778

バインド先: internal_api、external

ターゲットネットワークまたはサーバー: overcloud-controller-0、overcloud-controller-1、および overcloud-controller-2 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_placement
  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
```

panko

ポート番号: 8779

バインド先: internal_api、external

ターゲットネットワークまたはサーバー: overcloud-controller-0、overcloud-controller-1、および overcloud-controller-2 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 panko
  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
```

redis

ポート番号: 6379

バインド先: internal_api(redis サービス IP)

ターゲットネットワークまたはサーバー: overcloud-controller-0、overcloud-controller-1、および overcloud-controller-2 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。送信する文字列は **info\ replication\r\n** で、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 設定は、AUTH メソッドと Redis 管理パスワードによる **tcp-check** を使用します。director は通常、ランダムなパスワードを生成しますが、カスタムの Redis パスワードを定義できます。
- デフォルトの負荷分散方法は **ラウンドロビン** です。ただし、このサービスでは **first** メソッドが使用されます。これにより、利用可能な接続スロットがある最初のサーバーが接続を受け取るようになります。

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
```

swift_proxy_server

ポート番号: 8080

バインド先: storage、external

ターゲットネットワークまたはサーバー: overcloud-controller-0、overcloud-controller-1、および overcloud-controller-2 の storage

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
```