Red Hat OpenStack Platform 16.2

オーバークラウド向けの外部負荷分散

外部ロードバランサーを使用するように Red Hat OpenStack Platform 環境を設定する

Last Updated: 2022-04-21

Red Hat OpenStack Platform 16.2 オーバークラウド向けの外部負荷分散

外部ロードバランサーを使用するように Red Hat OpenStack Platform 環境を設定する

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

概要

オーバークラウドの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、Red Hat OpenStack Platform director を使用したオーバークラウドの設定に関するガイドラインが含まれます。

目次

前書き

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、弊社 の CTO、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)

弊社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連絡を取り問題についてお伺いできるように、ご自分のメールアドレスを追加します。
7. **送信** をクリックします。

第1章 外部のロードバランサーを使用するための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高可用性クラスターとして使用し、OpenStack サービスで動作する最大パフォーマンスを確保します。また、クラスターは OpenStack サービスの負荷分散機能も提供し、コントローラーノードに均等にトラフィックを分散し、各ノードのサーバーのオーバーロードを減らします。

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にトラフィックを負荷分散します。Haproxy **パッケージには**、着信トラフィックをリッスンする **haproxy** デーモンが含まれています。これには、ロギング機能とサンプル設定が含まれます。

オーバークラウドは、高可用性リソースマネージャーの 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、それぞれの設定で定義するルールの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、組織は専用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場合があります。外部ロードバランサーおよびオーバークラウド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て設定します。
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、ロードバランサーの IP アドレスと潜在的なノードの IP アドレスが必要です。

オーバークラウドが外部のロードバランサーを使用するように設定する前に、オーバークラウドで高可用性をデプロイし、実行するようにしてください。

1.1. 外部ロードバランサー用の環境の準備

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て、空のノードを director に登録します。次に、全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 で登録します。ノード定義のテンプレート **instackenv.json** は JSON ファイル形式で、ノードを登録するハードウェアおよび電源管理の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 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により、ノードがフレーバーに照合されます。次に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、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、お使いの環境に応じて以下の例を編集します。

```
{
    "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**)、続いて director にインポートして director にノードを登録します。

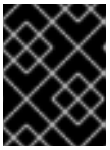
```
$ openstack overcloud node import ~/instackenv.json
```

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査します。

```
$ 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には、通常 15 分ほどかかります。

6. ノード一覧を取得して UUID を把握します。

```
$ 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、それぞれのプロファイルにノードをタグ付けします。

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

オーバークラウドの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、ローカル環境用の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ース設定テンプレート

を参照し、ネットワーク用のサブネットおよび 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. 各ロールにノードインターフェースを設定するには、以下のネットワークインターフェーステンプレートをカスタマイズします。
   - 各ロールに VLAN が設定された単一の NIC を設定するには、以下のディレクトリーで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
     ```

   - 各ロール向け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で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、環境に応じてコンテンツ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ーステンプレート
- 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してオーバークラウドをデプロイするには、必要な設定で新規の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、それぞれの分離ネットワーク上の 1 つの仮想 IP、Redis サービス用に 1 つ設定されます。オーバークラウドノードの NIC 設定が設定をサポートする場合、一部の仮想 IP は同一である場合があります。

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、環境に応じてコンテンツを編集します。

```
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 アドレスおよび IP 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各サービスで同じ IP 設定と一致する必要があります。
- **Parameter_defaults** セクションは、Redis サービス(RedisPassword)の管理パスワード **を定義し**、各 OpenStack サービスを特定のネットワークにマッピングします。負荷分散の設定には、このサービスの再マップが必要です。

1.4. 外部のロードバランシングでの 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、SSL がエンドポイントにアクセスできるように追加の環境ファイルを作成します。次に、外部の負荷分散サーバーに SSL 証明書とキー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

- パブリックエンドポイントへのアクセスに IP アドレスまたはドメイン名を使用している場合には、オーバークラウドのデプロイメントに含める以下の環境ファイルのいずれかを選択します。
  - ドメイン名サービス(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、**/usr/share/openstack-tripleo-heat-templates/environments/tls-endpoints-public-dns.yaml** ファイルを使用します。
  - IP アドレスでパブリックエンドポイントにアクセスするには、**/usr/share/openstack-tripleo-heat-templates/environments/tls-endpoints-public-ip.yaml** ファイルを使用します。

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トラストストアにない場合、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挿入します。

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

2. テキストエディターでファイルを開き、ルート認証局ファイルのコンテンツを **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

認証局のコンテンツには、新しい行すべてに同じインデントレベルが必要です。

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には、新しい環境ファイル **~/templates/cloudname.yaml** を作成し、以下のパラメーターに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、実際の環境の実際の値に置き換えます。

- **Cloud** Name: **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ます。
- **DnsServers**: 使用する DNS サーバーの一覧。設定済みの 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る必要があります。

1.5. 外部ロードバランサーを使用した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、**openstack overcloud deploy** を実行して、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追加します。

前提条件

- 環境は、外部ロードバランサー用に準備されます。環境の準備方法の詳細は、「外部ロードバランサー用の環境の準備」を参照してください。
- オーバークラウドネットワークが外部ロードバランサー用に設定されている。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の準備が整っている。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。
- SSL が外部の負荷分散用に設定されています。外部負荷分散用に SSL を設定する方法は、「外部のロードバランシングでの SSL の設定」を参照してください。

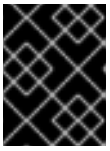
手順

1. 外部ロードバランサーのすべての環境ファイルおよび設定ファイルを指定して、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>
```

角括弧 **<>** の値を、環境で定義したファイルパスに置き換えます。



重要

以下の例に一覧表示されている順序で、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。

このコマンドにより、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**: 外部負荷分散の仮想 IP アドレス設定ファイル
- **external-lb.yaml**: 外部ロードバランサーの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、お使いの環境の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用の特定のフレーバーを使用します。
- SSL/TLS 環境ファイル：
  - **SSL/TLS エンドポイント環境ファイル**: パブリックの endpoinst 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 ホスト名環境ファイル：** DNS ホスト名を設定する環境ファイル。
  - **ルート証明書インジェクション環境ファイル**: ルート認証局を注入する環境ファイル。

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 はノードをプロビジョニングします。このプロセスは完了するまで多少時間がかかります。

2. オーバークラウドデプロイメントのステータスを表示するには、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： 外部の HAPROXY ロードバランサーを使用するオーバークラウド

この設定例は、フェデレーションされた HAProxy サーバーを使用して外部の負荷分散を提供するオーバークラウドを示しています。環境要件に応じて、別の外部ロードバランサーを選択できます。

設定例には、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と 1 つのコンピュートノードで構成される 3 つのコントローラーノードで構成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

以下の例では、各ネットワークに以下の IP アドレスの割り当てを使用します。

- Internal API **: 172.16.20.0/24**
- Tenant **: 172.16.22.0/24**
- Storage **: 172.16.21.0/24**
- ストレージ管理： **172.16.19.0/24**
- External **: 172.16.23.0/24**

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

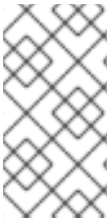
2.1. HAPROXY 設定ファイルの例

このサンプルファイルは内部 HAProxy 設定パラメーターを示しています。外部ロードバランサーの設定のベースとして、設定例パラメーターを使用できます。

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
- サービスの設定

Director は、コンテナー化されていない環境の各コントローラーノードの **/etc/haproxy/haproxy.conf** ファイルと、コンテナー化された環境の **/var/lib/config-data/puppet-generated/haproxy/etc/haproxy/haproxy.cfg** ファイルで設定を提供します。



注記

Global、default、および services 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 パラメーターの詳細は、**/usr/share/doc/haproxy-*/configuration.txt** の **HAProxy Configuration Manual**（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るシステム上にあります。

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、ロードバランサーのプロセス全体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できます。ご使用の環境に応じてパラメーター値を調整します。

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
```

以下の例は、次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

デフォルトの値設定パラメーター 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できます。ご使用の環境に応じてパラメーター値を調整します。

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
```

以下の例は、次のパラメーターを示しています。

- **ログ**: サービスのロギングを有効にします。**グローバル** 値は、ロギング機能がグローバル **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。この場合、デフォルトは TCP です。
- **retries**: 接続失敗を報告する前にサーバー上で実行する再試行回数を設定します。
- **timeout**: 特定の関数を待つ最大時間を設定します。たとえば、**timeout http-request** は、完全な HTTP 要求が完了するまで最大待機時間を設定します。

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

サービスレベルの設定パラメーター セクションは、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できます。ご使用の環境に基づいてパラメーター値を調整し、負荷分散する各サービスの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

以下の例では、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する必要があります。それぞれのサービス設定には、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** サービス名。
- **bind**: 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名、サーバーの IP アドレス、リッスンするポート、および接続パラメーター：
- **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 つのヘルスチェックが 2000 ミリ秒または 2 秒に設定される間隔。
- **rise 2**:（オプション）ヘルスチェックに 2 つの成功後、サービスは機能とみなされます。

Ceilometer **の例では**、このサービスは、ceilometer サービスが 172.16. **20.2500:8777 および 172.16. 23.250:8777** として提供される IP アドレスとポートを識別します。HAProxy は、これらのアドレスのリクエストを **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。

関連情報

- 「負荷分散機能を使用するサービスの設定パラメーター」

2.2. 負荷分散機能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サービスごとに、以下の例を参考にして外部ロードバランサーを設定します。ご使用の環境に基づいてパラメーター値を調整し、負荷分散する各サービスの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。

- 連続するヘルスチェックが 2000 ミリ秒または 2 秒に設定される間隔。
- ヘルスチェックに 2 回成功すると、サーバーは機能とみなされます。
- 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オプションを示します。

aodh

ポート番号： 8042

バインド： 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： 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： 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： 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
```

glance_registry

ポート番号： 9191

Bind to: 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： 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： 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： 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： 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： 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、URL との対話に Cookie 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： 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

Bind to: 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： 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

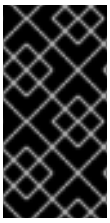
ポート番号： 3306

Bind to: 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、ヘルスチェックはポート 9200 を使用します。
- このサービスは、1度に 1 台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サーバーが利用できない場合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、すべての接続が即座に終了されます。
- 両側で TCP keepalive 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。
- 永続性を維持するのに役立つように、スティッキネステーブルを設定して IP アドレスを保存することができます。



重要

Mysql サービスは、Galera を使用して高可用性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ますが、ロックの競合を回避するには、ロードバランサーによって実施されるアクティブ/パッシブ構成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： 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： 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

Bind to: 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： 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、このサービスには **source** メソッドを使用します。この方法では、ソース IP アドレスをハッシュ化し、実行中のサーバーの合計ウェイトで割ります。このメソッドは、要求を受信するサーバーを指定し、サーバーが停止または起動しない限り、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できるようにします。実行中のサーバー数の変更が原因で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： 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： 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： 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

Bind to: 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 sequences 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 になります**。
- 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 設定では、AUTH メソッドと Redis 管理パスワードを指定して **tcp-check** を使用します。Director は通常無作為の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。
- デフォルトの分散方法は **ラウンドロビン** です。ただし、このサービスには **最初の** 方法を使用します。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： 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
```