Red Hat OpenStack Platform 16.2

Load Balancing-as-a-Service を提供する octavia の使用

Octavia の管理および octavia を使用してデータプレーン全体でネットワークトラフィックの負荷を分散する方法。

Last Updated: 2022-05-25

Red Hat OpenStack Platform 16.2 Load Balancing-as-a-Service を提供する octavia の使用

Octavia の管理および octavia を使用してデータプレーン全体でネットワークトラフィックの負荷を分散する方法。

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

Red Hat OpenStack Platform (RHOSP) Load-balancing サービス (octavia) をインストール、設定、操作、トラブルシューティング、およびアップグレードします。

目次

はじめに

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の提供

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。

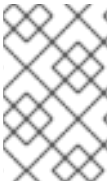
ドキュメントへのダイレクトフィードバック (DDF) 機能の使用 (英語版のみ)

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 LOAD-BALANCING サービスの概要

Load-balancing サービス (octavia) は、Red Hat OpenStack Platform (RHOSP) のデプロイメントに対して、Load Balancing-as-a-Service (LBaaS) API バージョン 2 の実装を提供します。Load-balancing サービスは、複数の仮想マシン、コンテナー、またはベアメタルサーバーを管理します。amphora と総称して、オンデマンドで起動します。オンデマンドの水平スケーリングを提供する機能により、Load-balancing サービスは RHOSP の大規模なエンタープライズデプロイメントに適した完全機能のロードバランサーになります。



注記

Red Hat は、Neutron-LBaaS から Load-balancing サービスへの移行パスをサポートしません。サポート対象外のオープンソースツールを使用できます。たとえば、GitHub で nlbaas2octavia-lb-replicator を検索します。

- 「Load-balancing サービスのコンポーネント」
- 「Load-balancing サービスのオブジェクトモデル」
- 「Red Hat OpenStack Platform での負荷分散の使用」

1.1. LOAD-BALANCING サービスのコンポーネント

Red Hat OpenStack Platform (RHOSP) Load-balancing サービス (octavia) は、コンピュートノード上で実行される **amphora** と呼ばれる仮想マシンインスタンスのセットを使用します。Load-balancing サービスコントローラーは、負荷分散管理ネットワーク (**lb-mgmt-net**) を使用して amphora と通信します。

octavia を使用する場合は、フローティング IP (FIP) を必要としないロードバランサー仮想 IP (VIP) を作成できます。FIP を使用しないことには、ロードバランサーによってパフォーマンスが向上するという利点があります。

図1.1 Load-balancing サービスのコンポーネント

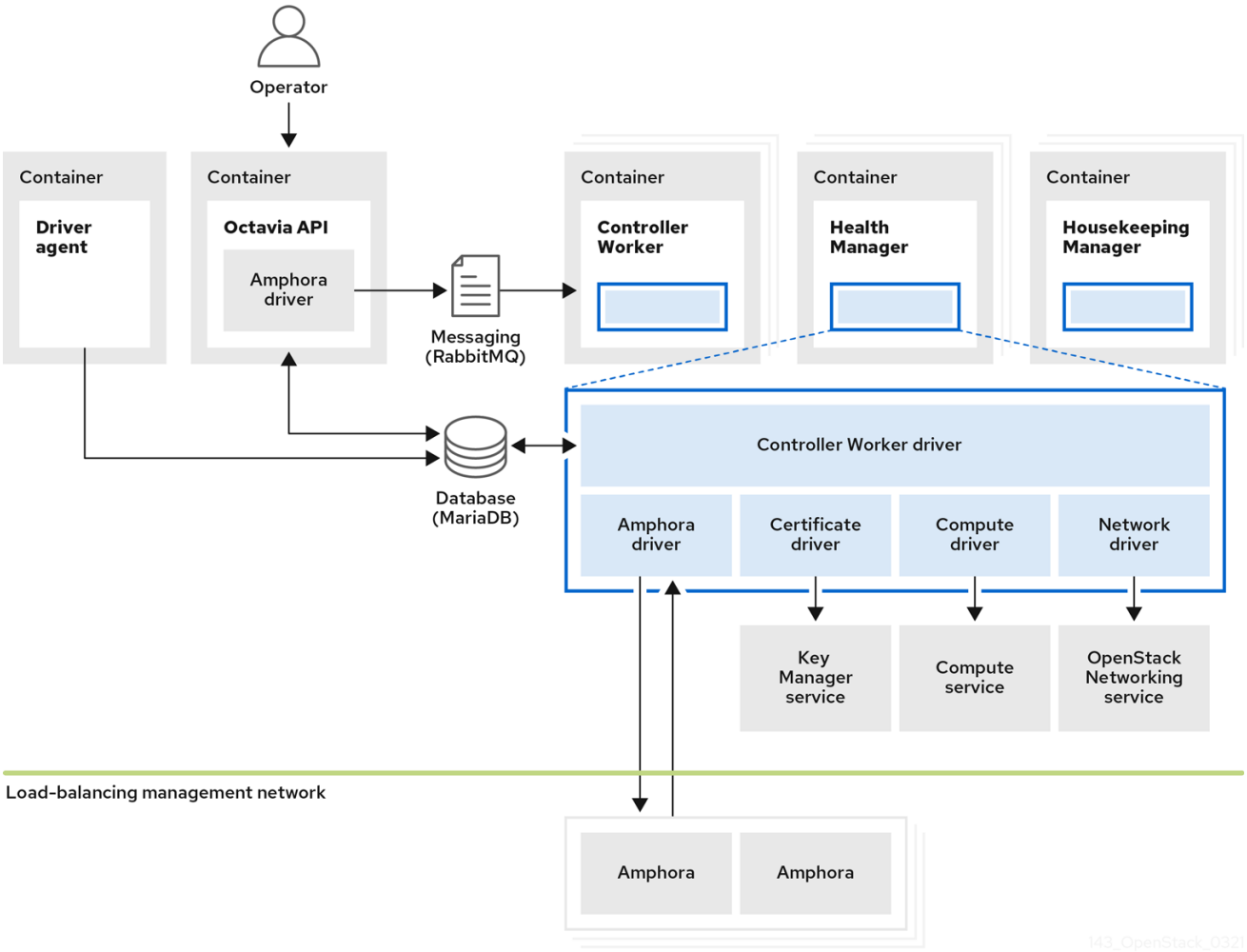


図 1.1 Load-balancing サービスのコンポーネントは、Networking API サーバーと同じノード上でホストされます。デフォルトでは、コントローラーノード上にあります。Load-balancing サービスは、以下のコンポーネントで構成されます。

octavia API (octavia_api コンテナー)

ユーザーが octavia と対話するための REST API を提供します。

コントローラーワーカー (octavia_worker コンテナー)

負荷分散管理ネットワークを通じて、設定および設定の更新を amphora に送信します。

ヘルスマネージャー (octavia_health_manager コンテナー)

個々の amphora の正常性を監視し、amphora に障害が発生した場合にフェイルオーバーイベントを処理します。

ハウスキーピングマネージャー (octavia_housekeeping コンテナー)

削除したデータベースレコードをクリーンアップし、amphora 証明書のローテーションを管理します。

ドライバーエージェント (octavia_driver_agentコンテナー)

OVN などの他のプロバイダードライバーをサポートします。

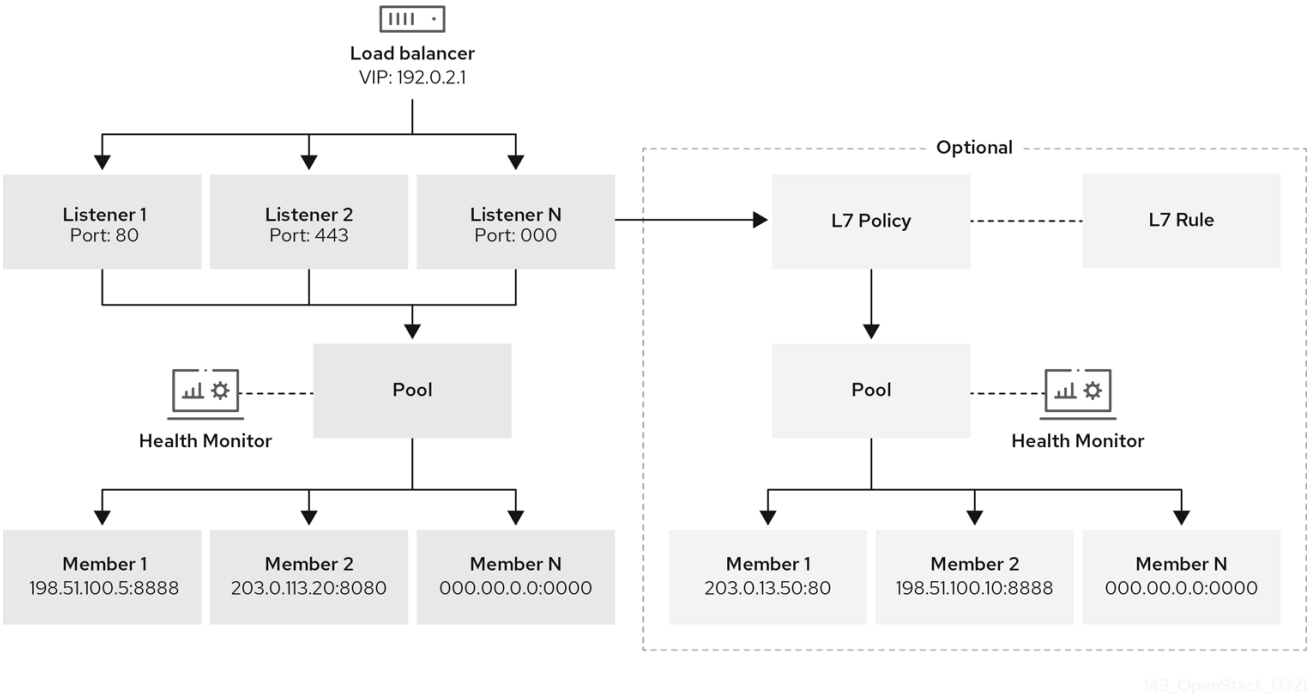
Amphora

負荷分散を実行します。amphora は、通常コンピュートノード上で実行されるインスタンスで、リスナー、プール、ヘルスモニター、L7 ポリシー、メンバーの設定に応じた負荷分散パラメーターにより設定されます。amphora はヘルスマネージャーに定期的なハートビートを送信します。

1.2. LOAD-BALANCING サービスのオブジェクトモデル

Red Hat OpenStack Platform (RHOSP) Load-balancing サービス (octavia) は、通常の負荷分散オブジェクトモデルを使用します。

図1.2 Load-balancing サービスオブジェクトモデルダイアグラム



ロードバランサー

負荷分散エンティティーを表す最上位の API オブジェクト。仮想 IP アドレスは、ロードバランサーの作成時に割り当てられます。amphora プロバイダーを使用してロードバランサー 1 つまたは複数の amphora インスタンスを作成する場合には、1 つまたは複数のコンピュートノードで起動します。

リスナー

ロードバランサーがリッスンするポート（HTTP の場合は TCP ポート 80 など）。

ヘルスモニター

各バックエンドメンバーサーバーで定期的なヘルスチェックを実行して失敗したサーバーを事前に検出し、プールから一時的に削除します。

プール

ロードバランサーからのクライアント要求を処理するメンバーのグループ。API を使用してプールを複数のリスナーに関連付けることができます。L7 ポリシーでプールを共有することができます。

メンバー

では、バックエンドインスタンスまたはサービスに接続する方法を説明します。この説明は、バックエンドメンバーが利用できる IP アドレスとネットワークポートで構成されます。

L7 ルール

L7 ポリシーが接続に適用されるかどうかを判断するレイヤー 7(L7)条件を定義します。

L7 ポリシー

リスナーに関連付けられた L7 ルールのコレクション。バックエンドプールに関連付けることもできます。ポリシーは、ポリシー内のすべてのルールが true の場合、ロードバランサーが実行するアクションを記述します。

関連情報

- 「Load-balancing サービスのコンポーネント」

1.3. RED HAT OPENSTACK PLATFORM での負荷分散の使用

負荷分散は、クラウドデプロイメントにおけるシンプルまたは自動配信のスケーリングおよび可用性を可能にする上で不可欠です。Load-balancing サービス (octavia) は、他の Red Hat OpenStack Platform (RHOSP) サービスに依存します。

- **Compute サービス (nova):** Load-balancing サービスの仮想マシンインスタンス (amphora) のライフサイクルを管理し、オンデマンドでコンピュートリソースを作成します。
- **Networking サービス (neutron):**amphora、テナント環境、および外部ネットワークとの間のネットワーク接続用。
- **Key Manager サービス (barbican)**: TLS セッションの終了がリスナーに設定されている場合に、TLS 証明書および認証情報を管理します。
- **Identity サービス (keystone):** octavia API への認証要求、および Load-balancing サービスが他の RHOSP サービスに対して認証を行う場合。
- **Image サービス (glance)**: amphora 仮想マシンイメージを保存します。
- **Common Libraries (oslo):** Load-balancing サービスコントローラーコンポーネント間の通信、標準の OpenStack フレームワーク内で機能する Load-balancing サービス、およびプロジェクトコード構造の確認を行います。
- **Taskflow**: 共通ライブラリーの一部で、Load-balancing サービスは、バックエンドサービスの設定および管理のオーケストレーション時にこのジョブフローシステムを使用します。

Load-balancing サービスは、ドライバーインターフェースを介して他の RHOSP サービスと対話します。このドライバーインターフェースは、外部コンポーネントが機能的に同等のサービスに置き換える必要がある場合に、Load-balancing サービスを大幅に再構築しないようにします。

第2章 LOAD-BALANCING サービスの実装に関する考慮事項

使用するプロバイダーを選択する、または高可用性環境を実装するかどうかなど、Red Hat OpenStack Platform (RHOSP) Load-balancing サービス (octavia) をデプロイする予定の場合には、いくつかの決定を行う必要があります。

- 「Load-balancing サービスプロバイダードライバー」
- 「Load-balancing サービス (octavia) 機能のサポートマトリックス」
- 「Load-balancing サービスのソフトウェア要件」
- 「アンダークラウドでの Load-balancing サービスの前提条件」
- 「Load-balancing サービスインスタンスのアクティブ/スタンバイトポロジーの基本」
- 「Load-balancing サービスデプロイメント後の操作」

2.1. LOAD-BALANCING サービスプロバイダードライバー

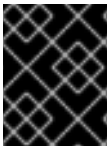
Red Hat OpenStack Platform (RHOSP) Load-balancing サービス (octavia) では、Octavia v2 API を使用して複数のプロバイダードライバーを有効にすることができます。1 つのプロバイダードライバーを使用するか、複数のプロバイダードライバーを同時に使用するかを選択することができます。

RHOSP では、amphora と Open Virtual Network (OVN) という 2 つの負荷分散プロバイダーを利用することができます。

デフォルトの amphora は、コンピュート環境でスケーリングする機能セットを備えた高可用性ロードバランサーです。このため、amphora は大規模なデプロイメントに適しています。

OVN 負荷分散プロバイダーは、基本的な機能セットを持つ軽量ロードバランサーです。OVN は、通常、east-west のレイヤー 4 ネットワークトラフィック用です。OVN は迅速にプロビジョニングを行い、amphora 等のフル機能の負荷分散プロバイダーよりも少ないリソースを消費します。

OVN メカニズムドライバー(ML2/OVN)と共に neutron Modular Layer 2 プラグインを使用する RHOSP デプロイメントでは、RHOSP director は追加のインストールや設定なしに Load-balancing サービスの OVN プロバイダードライバーを自動的に有効にします。



重要

本項の情報は、特に特に記載がない限り、amphora の負荷分散プロバイダーにのみ適用されます。

関連情報

- 「Load-balancing サービス (octavia) 機能のサポートマトリックス」

2.2. LOAD-BALANCING サービス (OCTAVIA) 機能のサポートマトリックス

Red Hat OpenStack Platform (RHOSP) Load-balancing サービス (octavia) では、amphora と Open Virtual Network (OVN) という 2 つの負荷分散プロバイダーを利用することができます。

Amphora はフル機能の負荷分散プロバイダーで、別個の haproxy 仮想マシンと追加のレイテンシーホップが必要になります。

OVN はすべてのノードで実行され、別の仮想マシンや追加のホップは必要ありません。ただし、OVN では、amphora よりも負荷分散機能がはるかに少なくなります。

以下の表には、Red Hat OpenStack Platform(RHOSP)16.2 がサポートする octavia の機能および機能に対するメンテナンスリリースのサポートをまとめています。

注記

この機能が一覧にない場合、RHOSP 16.2 はこの機能をサポートしません。

表2.1 Load-balancing サービス (octavia) 機能のサポートマトリックス

機能	RHOSP 16.2 のサポートレベル	
	amphora プロバイダー	OVN プロバイダー
ML2/OVS L3 HA	フルサポート	サポートなし
ML2/OVS DVR	フルサポート	サポートなし
ML2/OVS L3 HA とコンポーザブルネットワークノードの組み合わせ[1]	フルサポート	サポートなし
ML2/OVS DVR とコンポーザブルネットワークノードの組み合わせ[1]	フルサポート	サポートなし
ML2/OVN L3 HA	フルサポート	フルサポート
ML2/OVN DVR	フルサポート	フルサポート
ヘルスモニター	フルサポート	サポートなし
amphora アクティブ/スタンバイ	フルサポート: 16.1 以降	サポートなし
HTTPS 終端ロードバランサー (barbican を使用)	フルサポート: 16.0.1 以降	サポートなし
amphora 予備プール	テクノロジープレビューとしてのみ提供	サポートなし
UDP	フルサポート: 16.1 以降	フルサポート
バックアップメンバー	テクノロジープレビューとしてのみ提供	サポートなし

プロバイダーフレームワーク	テクノロジープレビューとしてのみ提供	サポートなし
TLS クライアント認証	テクノロジープレビューとしてのみ提供	サポートなし
TLS バックエンド暗号化	テクノロジープレビューとしてのみ提供	サポートなし
octavia フレーバー	フルサポート	サポートなし
オブジェクトタグ: API のみ	フルサポート: 16.1 以降	サポートなし
リスナー API タイムアウト	フルサポート	サポートなし
ログのオフロード	フルサポート: 16.1.2 以降	サポートなし
仮想 IP アクセス制御リスト	フルサポート	サポートなし
ボリュームベースの amphora	サポートなし	サポートなし

[1] ネットワークノードと OVS、メタデータ、DHCP、L3、および octavia (ワーカー、ヘルスモニター、およびハウスキーピング) の組み合わせ

関連情報

- 「Load-balancing サービスプロバイダードライバー」

2.3. LOAD-BALANCING サービスのソフトウェア要件

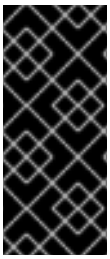
Red Hat OpenStack Platform (RHOSP) Load-balancing サービス (octavia) には、以下の OpenStack コアコンポーネントの設定が必要です。

- Compute (nova)
- OpenStack Networking (neutron)
- Image (glance)
- Identity (keystone)
- RabbitMQ
- MySQL

2.4. アンダークラウドでの LOAD-BALANCING サービスの前提条件

Red Hat OpenStack Platform (RHOSP) Load-balancing サービス (octavia) には、RHOSP アンダークラウドに対する以下の要件があります。

- アンダークラウドの正常なインストール。
- アンダークラウドに存在する Load-balancing サービス
- コンテナーベースのオーバークラウドデプロイメントプラン
- コントローラーノードに設定された負荷分散サービスコンポーネント。



重要

既存のオーバークラウドデプロイメントで Load-balancing サービスを有効にする場合には、アンダークラウドを準備する必要があります。準備を行わないと、Load-balancing サービスが動作していない状態でオーバークラウドのインストールは成功と報告されます。アンダークラウドを準備するには、**コンテナー化されたサービスへの移行**を参照してください。

2.5. LOAD-BALANCING サービスインスタンスのアクティブ/スタンバイトポロジーの基本

Red Hat OpenStack Platform (RHOSP) Load-balancing サービス (octavia) をデプロイする場合には、ユーザーがロードバランサーを作成する際に、デフォルトで高可用性であるかどうかを決定することができます。ユーザーに選択肢を与えたい場合は、RHOSP のデプロイメント後に、高可用性ロードバランサーを作成するための Load-balancing サービスフレーバー、およびスタンドアロンロードバランサーを作成するためのフレーバーを作成します。

デフォルトでは、amphora プロバイダードライバーは単一の Load-balancing サービス (amphora) インスタンストポロジーに対して設定され、高可用性 (HA) のサポートは限定的です。ただし、アクティブ/スタンバイのトポロジーを実装すると、Load-balancing サービスインスタンスを高可用性の構成にすることが可能です。

このトポロジーでは、Load-balancing サービスは各ロードバランサーに対してアクティブおよびスタンバイ状態のインスタンスを起動し、ロードバランサー間でセッションの永続性を維持します。アクティブなインスタンスが正常でなくなると、インスタンスは自動的にスタンバイ状態のインスタンスにフェイルオーバーし、アクティブにします。Load-balancing サービスヘルスマネージャーは、障害が発生したインスタンスを自動的に再構築します。

関連情報

- 「Load-balancing サービスインスタンスでのアクティブ/スタンバイトポロジーの有効化」

2.6. LOAD-BALANCING サービスデプロイメント後の操作

Red Hat OpenStack Platform (RHOSP) は、Load-balancing サービス (octavia) のデプロイ後のステップを簡素化するためのワークフロータスクを提供しています。このワークフローは、Ansible Playbook のセットを実行して、オーバークラウドのデプロイの最終段階として、以下のデプロイ後のステップを提供します。

- 証明書と鍵を設定する。
- amphora および Load-balancing サービスコントローラーワーカーおよびヘルスマネージャーの間の負荷分散管理ネットワークを設定します。

amphora イメージ

事前にプロビジョニングされたサーバーでは、Load-balancing サービスをデプロイする前に、アンダークラウドに amphora イメージをインストールする必要があります。

```
$ sudo dnf install octavia-amphora-image-x86_64.noarch
```

事前にプロビジョニングされていないサーバーでは、RHOSP director はデフォルトの amphora イメージを自動的にダウンロードして、オーバークラウドの Image サービス (glance) にアップロードし、Load-balancing サービスがその amphora イメージを使用するように設定します。スタックの更新またはアップグレード中に、director はこのイメージを最新の amphora イメージに更新します。

注記

カスタムの amphora イメージはサポートされていません。

関連情報

- 「Load-balancing サービスのデプロイ」

第3章 LOAD-BALANCING サービスのセキュリティー保護

Red Hat OpenStack Load-balancing サービス (octavia) のさまざまなコンポーネント間の通信を保護するには、TLS 暗号化プロトコルと公開鍵暗号を使用します。

- 「Load-balancing サービスにおける双方向 TLS 認証」
- 「Load-balancing サービスの証明書ライフサイクル」
- 「Load-balancing サービスの証明書および鍵の設定」

3.1. LOAD-BALANCING サービスにおける双方向 TLS 認証

Red Hat OpenStack Platform (RHOSP) Load-balancing サービス (octavia) のコントローラープロセスは、TLS 接続を介して Load-balancing サービスインスタンス (amphorae) と通信します。Load-balancing サービスは、双方向 TLS 認証を使用して、両側が信頼されていることを検証します。

注記

これは、TLS ハンドシェイクの完全なプロセスを単純化します。TLS ハンドシェイクプロセスの詳細は、TLS 1.3 RFC 8446 を参照してください。

双方向 TLS 認証には、2 つのフェーズがあります。**フェーズ 1** では、Load-balancing サービスのワーカープロセスなどの Controlller プロセスが Load-balancing サービスインスタンスに接続し、インスタンスがそのサーバー証明書を Controller に提示します。次に、Contoller は Controller に保存されているサーバー認証局 (CA) 証明書に対して検証します。提示された証明書がサーバー CA 証明書に対して検証されている場合は、接続はフェーズ 2 に進みます。

フェーズ 2 では、Controller はクライアント証明書を Load-balancing サービスインスタンスに提示します。次に、インスタンスはインスタンス内に保存されているクライアント CA 証明書に対して証明書を検証します。この証明書が正常に検証されると、残りの TLS ハンドシェイクは、Controller と Load-balancing サービスインスタンス間にセキュアな通信チャネルを確立します。

関連情報

- 「Load-balancing サービスの証明書および鍵の設定」

3.2. LOAD-BALANCING サービスの証明書ライフサイクル

Red Hat OpenStack Platform (RHOSP) Load-balancing サービス (octavia) コントローラーは、サーバーの認証局の証明書およびキーを使用して、それぞれの Load-balancing サービスインスタンス (amphora) の証明書を一意に生成します。

Load-balancing サービスハウスキーピングコントローラープロセスは、これらのサーバー証明書の有効期限が近づくと証明書を自動的にローテーションします。

Load-balancing サービスコントローラープロセスは、クライアント証明書を使用します。これらの TLS 証明書を管理する人の運用者は、証明書がクラウドコントロールプレーンで使用されるため、通常、長い有効期限を付与します。

関連情報

- 「Load-balancing サービスの証明書および鍵の設定」

3.3. LOAD-BALANCING サービスの証明書および鍵の設定

Red Hat OpenStack Platform (RHOSP) director を設定して証明書およびキーを生成するか、独自の証明書を指定することができます。必要なプライベート認証局を自動的に作成し、必要な証明書を発行するように director を設定します。これらの証明書は Load-balancing サービス (octavia) の内部通信専用で、ユーザーに公開されません。

重要

RHOSP director は証明書およびキーを生成し、有効期限が切れる前にそれらを自動的に更新します。独自の証明書を使用する場合は、証明書を更新することを覚えている必要があります。

注記

手動で生成された証明書から自動生成された証明書への切り替えは、RHOSP director ではサポートされません。ただし、Controller ノード上の既存の証明書 (**/var/lib/config-data/puppet-generated/octavia/etc/octavia/certs** ディレクトリー内) を削除して、オーバークラウドを更新することで、証明書を強制的に再作成することができます。

独自の証明書および鍵を使用する必要がある場合は、以下の手順を実行します。

前提条件

- 「Load-balancing サービスのデフォルト設定の変更」を読んで理解している (「関連資料」のリンクを参照してください)。

手順

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. source コマンドでアンダークラウドの認証情報ファイルを読み込みます。

```
$ source ~/stackrc
```

3. YAML カスタム環境ファイルを作成します。

   例

```
$ vi /home/stack/templates/my-octavia-environment.yaml
```

4. YAML 環境ファイルに以下のパラメーターを追加し、ご自分のサイトに適した値を設定します。

   - **OctaviaCaCert**:
     Octavia が証明書を生成するために使用する CA の証明書。
   - **OctaviaCaKey**:
     生成された証明書の署名に使用するプライベート CA 鍵
   - **OctaviaCaKeyPassphrase**:
     上記のプライベート CA 鍵で使用するパスフレーズ
   - **OctaviaClientCert**:

コントローラー用に octavia CA が発行するクライアント証明書および暗号化されていない鍵

- **OctaviaGenerateCerts**:
  証明書および鍵の自動生成の有効 (true) または無効 (false) を director に指示するブール値

重要

OctaviaGenerateCerts を false に設定する必要があります。

例

```
parameter_defaults:
  OctaviaCaCert: |
    -----BEGIN CERTIFICATE-----

MIIDgzCCAmugAwIBAgIJAKk46qw6ncJaMA0GCSqGSIb3DQEBCwUAMFgxCzAJBgNV
    [snip]
    sFW3S2roS4X0Af/kSSD8mlBBTFTCMBAj6rtLBKLaQbIxEpIzrgvp
    -----END CERTIFICATE-----

  OctaviaCaKey: |
    -----BEGIN RSA PRIVATE KEY-----
    Proc-Type: 4,ENCRYPTED
    [snip]
    -----END RSA PRIVATE KEY-----[

  OctaviaClientCert: |
    -----BEGIN CERTIFICATE-----

MIIDmjCCAoKgAwIBAgIBATANBgkqhkiG9w0BAQsFADBcMQswCQYDVQQGEwJVUzEP

    [snip]
    270l5ILSnfejLxDH+vI=
    -----END CERTIFICATE-----
    -----BEGIN PRIVATE KEY-----

MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQDU771O8MTQV8RY

    [snip]
    KfrjE3UqTF+ZaaIQaz3yayXW
    -----END PRIVATE KEY-----

  OctaviaCaKeyPassphrase:
    b28c519a-5880-4e5e-89bf-c042fc75225d

  OctaviaGenerateCerts: false
  [rest of file snipped]
```

5. コア heat テンプレート、環境ファイル、およびこの新しいカスタム環境ファイルを指定して、**openstack overcloud deploy** コマンドを実行します。

重要

後で実行される環境ファイルで定義されているパラメーターとリソースが優先されることになるため、環境ファイルの順序は重要となります。

例

```
$ openstack overcloud deploy --templates \
-e <your_environment_files> \
-e /usr/share/openstack-tripleo-heat-templates/environments/services/octavia.yaml \
-e /home/stack/templates/my-octavia-environment.yaml
```

関連情報

- 「Load-balancing サービスのデフォルト設定の変更」
- 『Advanced Overcloud Customization』の「Environment Files」
- 『オーバークラウドの高度 **なカスタマイズ**』 の「オーバークラウド作成時の環境ファイル の追加」

第4章 LOAD-BALANCING サービスのインストールおよび設定

RHOSP director を使用して Red Hat OpenStack Platform (RHOSP) Load-balancing サービス (octavia) をデプロイする場合、その仮想マシンインスタンス (amphora) を高可用性にすることができます。また、Load-balancing サービスの設定変更を行う場合には、director も使用します。

- 「Load-balancing サービスのデプロイ」
- 「Load-balancing サービスインスタンスでのアクティブ/スタンバイトポロジーの有効化」
- 「Load-balancing サービスのデフォルト設定の変更」

4.1. LOAD-BALANCING サービスのデプロイ

Red Hat OpenStack Platform (RHOSP) director を使用して、Load-balancing サービス (octavia) をデプロイします。director は、環境のプランのセットである Orchestration サービス (heat)テンプレートを使用します。アンダークラウドはこれらのプランをインポートし、Load-balancing サービスおよび RHOSP 環境を作成する手順に従います。

前提条件

- お使いの環境が octavia のイメージにアクセスできるようにしている。

手順

- コア heat テンプレート、環境ファイル、および **octavia.yaml** heat テンプレートを指定してデプロイメントコマンドを実行します。

  例

  ```
  $ openstack overcloud deploy --templates \
  -e <your_environment_files> \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/octavia.yaml
  ```

  注記

  director は、スタックの更新またはアップグレード中に amphora を最新の amphora イメージに更新します。

関連情報

- **『Director Installation and Usage』**の「Deployment command options」

4.2. LOAD-BALANCING サービスインスタンスでのアクティブ/スタンバイトポロジーの有効化

Red Hat OpenStack Platform (RHOSP) director を使用してアクティブ/スタンバイトポロジーを実装すると、Load-balancing サービスインスタンス (amphorae) を高可用性にすることができます。director は、環境のプランのセットである Orchestration サービス (heat)テンプレートを使用します。アンダークラウドはこれらのプランをインポートし、Load-balancing サービスおよび RHOSP 環境を作成する手順に従います。

前提条件

- Compute サービスに対して非アフィニティーが有効であることを確認します。これがデフォルトです。
- 最小 3 台のコンピュートノードホスト:
  - 異なるホストに amphora を配置する 2 台のコンピュートノードホスト (Compute の非アフィニティー)。
  - 問題が発生した場合に、アクティブスタンバイロードバランサーを正常にフェイルオーバーする 3 番目のホスト。

手順

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. source コマンドでアンダークラウドの認証情報ファイルを読み込みます。

```
$ source ~/stackrc
```

3. カスタム YAML 環境ファイルを作成します。

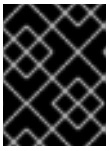
   例

```
$ vi /home/stack/templates/my-octavia-environment.yaml
```

4. カスタム環境ファイルに、以下のパラメーターを追加します。

```
parameter_defaults:
  OctaviaLoadBalancerTopology: "ACTIVE_STANDBY"
```

5. コア heat テンプレート、環境ファイル、およびこの新しいカスタム環境ファイルを指定して、deployment コマンドを実行します。



重要

後で実行される環境ファイルで定義されているパラメーターとリソースが優先されることになるため、環境ファイルの順序は重要となります。

   例

```
$ openstack overcloud deploy --templates \
-e <your_environment_files> \
-e /usr/share/openstack-tripleo-heat-templates/environments/services/octavia.yaml \
-e /home/stack/templates/my-octavia-environment.yaml
```

検証手順

- デプロイメントが完了してロードバランサーを作成したら、以下のコマンドを実行します。

```
$ source overcloudrc
$ openstack loadbalancer amphora list
```

Load-balancing サービスインスタンスの高可用性設定に成功すると、2 つのインスタンス (amphora) に関する出力が表示され、**role** の **SINGLE** は表示されなくなります。

関連情報

- **『Advanced Overcloud Customization』** の「Environment files」
- 『オーバークラウドの高度 **なカスタマイズ』** の「オーバークラウド作成時の環境ファイル の追加」

4.3. LOAD-BALANCING サービスのデフォルト設定の変更

Red Hat OpenStack Platform (RHOSP) director を使用して、Load-balancing サービス (octavia) に設定変更を行います。director は、環境のプランのセットである Orchestration サービス (heat)テンプレートを使用します。アンダークラウドはこれらのプランをインポートし、Load-balancing サービスおよび RHOSP 環境を作成する手順に従います。

前提条件

- アンダークラウドで以下のファイルを参照して、director が Load-balancing サービスをデプロイするのにすでに使用している RHOSP Orchestration サービス (heat) パラメーターを決定します。

```
/usr/share/openstack-tripleo-heat-templates/deployment/octavia/octavia-deployment-config.j2.yaml
```

- 変更するパラメーターを決定します。
  以下にいくつか例を示します。
  - **OctaviaControlNetwork**
    ロードバランサー管理ネットワークに使用される neutron ネットワークの名前
  - **OctaviaControlSubnetCidr**
    amphora 管理サブネット用のサブネット (CIDR 形式)
  - **OctaviaMgmtPortDevName**
    octavia ワーカー/ヘルスマネージャーと amphora マシン間の通信に使用される octavia 管理ネットワークインターフェースの名前

手順

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. source コマンドでアンダークラウドの認証情報ファイルを読み込みます。

```
$ source ~/stackrc
```

3. カスタム YAML 環境ファイルを作成します。

   例

```
$ vi /home/stack/templates/my-octavia-environment.yaml
```

4. ご自分の環境ファイルには、**parameter_defaults** というキーワードを含める必要があります。**parameter_defaults** のキーワードの後に、ご自分のパラメーター/値ペアを定義します。

例

```
parameter_defaults:
  OctaviaMgmtPortDevName: "o-hm0"
  OctaviaControlNetwork: 'lb-mgmt-net'
  OctaviaControlSubnet: 'lb-mgmt-subnet'
  OctaviaControlSecurityGroup: 'lb-mgmt-sec-group'
  OctaviaControlSubnetCidr: '172.24.0.0/16'
  OctaviaControlSubnetGateway: '172.24.0.1'
  OctaviaControlSubnetPoolStart: '172.24.0.2'
  OctaviaControlSubnetPoolEnd: '172.24.255.254'
```

5. コア heat テンプレート、環境ファイル、およびこの新しいカスタム環境ファイルを指定して、deployment コマンドを実行します。

重要

後で実行される環境ファイルで定義されているパラメーターとリソースが優先されることになるため、環境ファイルの順序は重要となります。

例

```
$ openstack overcloud deploy --templates \
-e <your_environment_files> \
-e /usr/share/openstack-tripleo-heat-templates/environments/services/octavia.yaml \
-e /home/stack/templates/my-octavia-environment.yaml
```

関連情報

- 『Advanced Overcloud Customization』の「Environment files」
- 『オーバークラウドの高度 **なカスタマイズ**』 の「オーバークラウド作成時の環境ファイル の追加」

第5章 LOAD-BALANCING サービスインスタンスのログの管理

テナントフローのロギングを有効にするか、amphora ローカルファイルシステムへのログインを抑制することができます。また、コンテナーのセットの syslog レシーバーに管理またはテナントフローログを転送したり、選択したエンドポイントで他の syslog レシーバーに転送することもできます。

さらに、syslog ファシリティー値を設定する、テナントフローログのフォーマットを変更する、カーネル等のソースや cron からのログを含めるように管理ログの範囲を拡張する、等のさまざまなロギング機能を制御できます。

- 「Load-balancing サービスインスタンス (amphora) ログのオフロードの基本」
- 「Load-balancing サービスインスタンスの管理ログのオフロードの有効化」
- 「Load-balancing サービスインスタンスのテナントフローログのオフロードの有効化」
- 「Load-balancing サービスインスタンスのテナントフローのロギングの無効化」
- 「Load-balancing サービスインスタンスのローカルログストレージの無効化」
- 「Load-balancing サービスインスタンスのロギング用 heat パラメーター」
- 「Load-balancing サービスインスタンスのテナントフローログ形式」

5.1. LOAD-BALANCING サービスインスタンス (AMPHORA) ログのオフロードの基本

デフォルトでは、Load-balancing サービスインスタンス (amphora) は、ローカルマシンの systemd ジャーナルにログを保存します。ただし、amphora がログを syslog レシーバーにオフロードするように指定して、管理とテナント両方のトラフィックフローのログを集約することができます。ログのオフロードにより、管理者はログを 1 カ所で管理し、amphora のローテーション後もログを維持することができます。

関連情報

- 「Load-balancing サービスインスタンスのロギング用 heat パラメーター」
- 「Load-balancing サービスインスタンスのテナントフローログ形式」

5.2. LOAD-BALANCING サービスインスタンスの管理ログのオフロードの有効化

デフォルトでは、Load-balancing サービスインスタンス (amphora) は、ローカルマシンの systemd ジャーナルにログを保存します。ただし、amphora がログを syslog レシーバーにオフロードするように指定して、管理ログを集約することができます。ログのオフロードにより、管理者はログを 1 カ所で管理し、amphora のローテーション後もログを維持することができます。

手順

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. source コマンドでアンダークラウドの認証情報ファイルを読み込みます。

```
$ source ~/stackrc
```

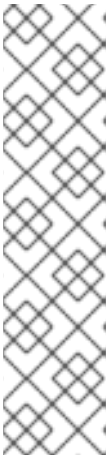
3. カスタム YAML 環境ファイルを作成します。

例

```
$ vi /home/stack/templates/my-octavia-environment.yaml
```

4. YAML 環境ファイルの **parameter_defaults** セクションで、**OctaviaLogOffload** を **true** に設定します。

```
parameter_defaults:
  OctaviaLogOffload: true
  ...
```



注記

OctaviaAdminLogFacility パラメーターで別の値を指定しない限り、デフォルトでは、amphora は syslog ファシリティーの値に **local1** を使用して管理ログをオフロードします。

例

```
parameter_defaults:
  OctaviaLogOffload: true
  OctaviaAdminLogFacility: 2
  ...
```

5. amphora は、haproxy 管理ログ、keepalived、amphora エージェントログなどのロードバランサー関連の管理ログのみを転送します。カーネル、システム、およびセキュリティーログ等の amphora からの すべての 管理ログを送信するように amphora を設定する場合には、**OctaviaForwardAllLogs** を **true** に設定します。

例

```
parameter_defaults:
  OctaviaLogOffload: true
  OctaviaForwardAllLogs: true
  ...
```

6. amphora は、ログメッセージをリッスンする syslog レシーバーが含まれる、Orchestration サービス (heat) で定義されたデフォルトコンテナーのセットを使用します。異なるエンドポイントのセットを使用する場合は、**OctaviaAdminLogTargets** パラメーターでそれらを指定することができます。

```
OctaviaAdminLogTargets: <ip_address>:<port>[, <ip_address>:<port>]
```

例

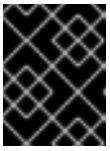
```
parameter_defaults:
  OctaviaLogOffload: true
  OctaviaAdminLogTargets: 192.0.2.1:10514, 2001:db8:1::10:10514
  ...
```

7. デフォルトでは、ログオフロードを有効にすると、テナントフローログもオフロードされます。
テナントフローログのオフロードを無効にする場合は、**OctaviaConnectionLogging** を **false** に設定します。

例

```
parameter_defaults:
  OctaviaLogOffload: true
  OctaviaConnectionLogging: false
  ...
```

8. コア heat テンプレート、環境ファイル、およびこの新しいカスタム環境ファイルを指定して、deployment コマンドを実行します。



重要

後で実行される環境ファイルで定義されているパラメーターとリソースが優先されることになるため、環境ファイルの順序は重要となります。

例

```
$ openstack overcloud deploy --templates \
-e [your-environment-files] \
-e /usr/share/openstack-tripleo-heat-templates/environments/services/octavia.yaml \
-e /home/stack/templates/my-octavia-environment.yaml
```

検証手順

- **OctaviaAdminLogTargets** または **OctaviaTenantLogTargets** で特定のエンドポイントを指定しない限り、amphora は RHOSP コントローラー内の他の RHOSP ログと同じ場所 (**/var/log/containers/octavia/**) にログをオフロードします。
- 適切な場所を確認して、以下のログファイルが存在することを確認します。
  - **octavia-amphora.log**: 管理ログのログファイル
  - (有効な場合) **octavia-tenant-traffic.log**: テナントトラフィックフローログのログファイル

関連情報

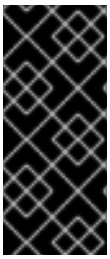
- 「Load-balancing サービスインスタンスのロギング用 heat パラメーター」
- 『Advanced Overcloud Customization』の「Environment files」
- 『オーバークラウドの高度なカスタマイズ』の「オーバークラウド作成時の環境ファイルの追加」

5.3. LOAD-BALANCING サービスインスタンスのテナントフローログのオフロードの有効化

デフォルトでは、Load-balancing サービスインスタンス (amphora) は、ローカルマシンの systemd ジャーナルにログを保存します。amphora がテナントフローログ用に十分なディスク領域が含まれるエ

ンドポイント上の syslog レシーバーにログをオフロードすることを指定できます。テナントフローログは、テナント接続の数によってサイズが増大する場合があります。

管理ログのオフロードが有効な場合に、Load-balancing サービスインスタンスのテナントフローログのオフロードは、自動的に有効になります。管理ログのオフロードが有効で、テナントフローログのオフロードが無効になるのは、**OctaviaConnectionLogging** パラメーターが **false** に設定されている場合だけです。



重要

テナントフローロギングは、ロードバランサーが受信する接続の数によって、多数の syslog メッセージを生成する場合があります。テナントフローロギングは、ロードバランサーへの接続ごとに 1 つのログエントリーを生成します。ログボリュームを監視し、ロードバランサーが管理する接続の数に基づいて syslog レシーバーを適切に設定します。

手順

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. source コマンドでアンダークラウドの認証情報ファイルを読み込みます。

```
$ source ~/stackrc
```

3. **OctaviaConnectionLogging** パラメーターが設定されている環境ファイルを見つけます。

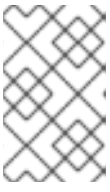
```
$ grep -rl OctaviaConnectionLogging /home/stack/templates/
```

4. ファイルが見つからない場合は、環境ファイルを作成します。

```
$ vi /home/stack/templates/my-octavia-environment.yaml
```

5. **OctaviaLogOffload** パラメーターおよび **OctaviaConnectionLogging** パラメーターを環境ファイルの **parameter_defaults** セクションに追加し、値を **true** に設定します。

```
parameter_defaults:
  OctaviaLogOffload: true
  OctaviaConnectionLogging: true
  ...
```



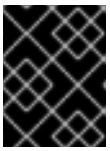
注記

amphora は、**OctaviaTenantLogFacility** パラメーターを使用して 別の値を指定しない限り、テナントフローログをオフロードするために **local0** の syslog ファシリティーのデフォルト値を使用します。

6. オプション: amphora は、ログメッセージをリッスンする syslog レシーバーを含むデフォルトコンテナーのセットを使用します。admin パラメーターおよびテナントログのエンドポイントは、**OctaviaAdminLogTargets** および **OctaviaTenantLogTargets** パラメーターを使用して変更することができます。

```
OctaviaAdminLogTargets: <ip-address>:<port>[, <ip-address>:<port>]
OctaviaTenantLogTargets: <ip-address>:<port>[, <ip-address>:<port>]
```

7. コア heat テンプレート、環境ファイル、修正したカスタム環境ファイルを指定してデプロイメントコマンドを実行します。



重要

後で実行される環境ファイルで定義されているパラメーターとリソースが優先されることになるため、環境ファイルの順序は重要となります。

```
$ openstack overcloud deploy --templates \
-e <your_environment_files> \
-e /usr/share/openstack-tripleo-heat-templates/environments/services/octavia.yaml \
-e /home/stack/templates/my-octavia-environment.yaml
```

検証

- **OctaviaAdminLogTargets** または **OctaviaTenantLogTargets** で特定のエンドポイントを指定しない限り、amphora は RHOSP コントローラー内の他の RHOSP ログと同じ場所 (**/var/log/containers/octavia/**) にログをオフロードします。
- 適切な場所を確認して、以下のログファイルが存在することを確認します。
  - **octavia-amphora.log**: 管理ログのログファイル
  - **octavia-tenant-traffic.log**: テナントトラフィックフローログのログファイル

関連情報

- 「Load-balancing サービスインスタンスのロギング用 heat パラメーター」
- 『Advanced Overcloud Customization』の「Environment files」
- 『オーバークラウドの高度 **なカスタマイズ**』 の「オーバークラウド作成時の環境ファイル の追加」
- 「Load-balancing サービスインスタンスのテナントフローログ形式」

5.4. LOAD-BALANCING サービスインスタンスのテナントフローのロギングの無効化

管理ログのオフロードを有効にすると、Load-balancing サービスインスタンス (amphorae) のテナントフローログのオフロードは自動的に有効になります。

管理ログのオフロードを有効にし、テナントフローのロギングを無効にするには、**OctaviaConnectionLogging** パラメーターを **false** に設定する必要があります。

OctaviaConnectionLogging パラメーターが **false** の場合、amphora は amphora 内のディスクにテナントフローログを書き込みせず、別の場所でリッスンする syslog レシーバーにログをオフロードします。

手順

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. source コマンドでアンダークラウドの認証情報ファイルを読み込みます。

```
$ source ~/stackrc
```

3. amphora ロギングが設定されている YAML カスタム環境ファイルを見つけます。

例

```
$ grep -rl OctaviaLogOffload /home/stack/templates/
```

4. カスタム環境ファイルの **parameter_defaults** セクションで、**OctaviaConnectionLogging** を **false** に設定します。

例

```
parameter_defaults:
  OctaviaLogOffload: true
  OctaviaConnectionLogging: false
  ...
```

5. コア heat テンプレート、環境ファイル、および **OctaviaConnectionLogging** を **true** に設定するカスタム環境ファイルを指定してデプロイメントコマンドを実行します。

重要

後で実行される環境ファイルで定義されているパラメーターとリソースが優先されることになるため、環境ファイルの順序は重要となります。

例

```
$ openstack overcloud deploy --templates \
-e [your-environment-files] \
-e /usr/share/openstack-tripleo-heat-templates/environments/services/octavia.yaml \
-e /home/stack/templates/my-octavia-environment.yaml
```

検証手順

- **OctaviaAdminLogTargets** または **OctaviaTenantLogTargets** で特定のエンドポイントを指定しない限り、amphora は RHOSP コントローラー内の他の RHOSP ログと同じ場所 (**/var/log/containers/octavia/**) にログをオフロードします。
- 適切な場所を確認して、**octavia-tenant-traffic.log** が **存在しない** ことを確認します。

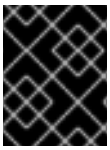
関連情報

- 『Advanced Overcloud Customization』の「Environment files」
- 『オーバークラウドの高度 **なカスタマイズ**』の「オーバークラウド作成時の環境ファイルの追加」

5.5. LOAD-BALANCING サービスインスタンスのローカルログストレージの無効化

管理者およびテナントフローログをオフロードするように Load-balancing サービスインスタンス

(amphorae)を設定している場合でも、amphora は引き続きこれらのログを amphora 内のディスクに書き込みます。ロードバランサーのパフォーマンスを向上させるために、ローカルでロギングを停止することができます。



重要

ロギングをローカルで無効にする場合には、カーネル、システム、セキュリティーロギングなど、amphora のすべてのログストレージも無効にしてください。

注記

ローカルログストレージを無効にし、**OctaviaLogOffload** パラメーターが false に設定されている場合、負荷分散のパフォーマンスを強化するために **OctaviaConnectionLogging** を false に設定するようにしてください。

手順

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. source コマンドでアンダークラウドの認証情報ファイルを読み込みます。

```
$ source ~/stackrc
```

3. カスタム YAML 環境ファイルを作成します。

例

```
$ vi /home/stack/templates/my-octavia-environment.yaml
```

4. **parameter_defaults** の下にある環境ファイルで、**OctaviaDisableLocalLogStorage** を **true** に設定します。

```
parameter_defaults:
  OctaviaDisableLocalLogStorage: true
  ...
```

5. コア heat テンプレート、環境ファイル、およびこの新しいカスタム環境ファイルを指定して、deployment コマンドを実行します。

重要

後で実行される環境ファイルで定義されているパラメーターとリソースが優先されることになるため、環境ファイルの順序は重要となります。

例

```
$ openstack overcloud deploy --templates \
-e <your_environment_files> \
-e /usr/share/openstack-tripleo-heat-templates/environments/services/octavia.yaml \
-e /home/stack/templates/my-octavia-environment.yaml
```

検証手順

- amphora インスタンスで、ログファイルが書き込まれる場所を確認し、新しいログファイルが書き込まれていないことを確認します。

関連情報

- 『Advanced Overcloud Customization』の「Environment Files」
- 『オーバークラウドの高度なカスタマイズ』の「オーバークラウド作成時の環境ファイルの追加」

5.6. LOAD-BALANCING サービスインスタンスのロギング用 HEAT パラメーター

Load-balancing サービスインスタンス (amphora) のロギングを設定する場合は、ロギングを制御する 1 つまたは複数の Orchestration サービス (heat) パラメーターに値を設定し、**openstack overcloud deploy** コマンドを実行します。

amphora ロギング用の heat パラメーターにより、ログオフロードの有効化、ログをオフロードするカスタムエンドポイントの定義、ログの syslog ファシリティー値の設定などの機能を制御できます。

表5.1 すべてのログの heat パラメーター

パラメーター	デフォルト	説明
OctaviaLogOffload	**false**	**true** の場合、インスタンスはログをオフロードします。エンドポイントが指定されていない場合、デフォルトでは、インスタンスはそのログを他の RHOSP ログと同じ場所 (**/var/log/containers/octavia/**) にオフロードします。
OctaviaDisableLocalLogStorage	**false**	**true** の場合、インスタンスはインスタンスのホストのファイルシステムにログを格納しません。これには、すべてのカーネル、システム、およびセキュリティログが含まれます。
OctaviaForwardAllLogs	**false**	**true** の場合、インスタンスはすべてのログメッセージを管理ログのエンドポイントに転送します。これには、cron やカーネルログなどの負荷分散に関連しないログも含まれます。 インスタンスが **OctaviaForwardAllLogs** を認識するには、**OctaviaLogOffload** も有効にする必要があります。

表5.2 管理ロギング用の heat パラメーター

パラメーター	デフォルト	説明

パラメーター	デフォルト	説明
OctaviaAdminLogTargets	値なし。	管理ログメッセージを受信する syslog エンドポイントのコンマ区切りリスト (<host>:<port>) これらのエンドポイントは、指定されたポートでログメッセージをリッスンするプロセスを実行しているコンテナー、仮想マシン、または物理ホストになります。 **OctaviaAdminLogTargets** が存在しない場合、インスタンスは RHOSP director で定義されたコンテナーセットの他の RHOSP ログと同じ場所 (**/var/log/containers/octavia/**) にログをオフロードします。
OctaviaAdminLogFacility	**1**	管理ログメッセージに使用する syslog "LOG_LOCAL" ファシリティーである 0 から 7 の間の数字

表5.3 テナントフローロギング用の heat パラメーター

パラメーター	デフォルト	説明
OctaviaConnectionLogging	**true**	**true** の場合、テナント接続フローがログに記録されます。 **OctaviaConnectionLogging** が false の場合、amphora は **OctaviaLogOffload** 設定に関係なく、テナントの接続を停止します。**OctaviaConnectionLogging** はローカルのテナントフローログのストレージを無効にし、ログのオフロードが有効な場合、テナントフローログを転送しません。
OctaviaTenantLogTargets	値なし。	テナントトラフィックフローログメッセージを受信する syslog エンドポイントのコンマ区切りリスト (<host>:<port>) これらのエンドポイントは、指定されたポートでログメッセージをリッスンするプロセスを実行しているコンテナー、仮想マシン、または物理ホストになります。 **OctaviaTenantLogTargets** が存在しない場合、インスタンスは RHOSP director で定義されたコンテナーセットの他の RHOSP ログと同じ場所 (**/var/log/containers/octavia/**) にログをオフロードします。
OctaviaTenantLogFacility	**0**	テナントトラフィックフローログメッセージに使用する syslog "LOG_LOCAL" ファシリティーである 0 から 7 の間の数字

パラメーター	デフォルト	説明
OctaviaUserLogFormat	**"{{ '{{' }} project_id {{ '}}' }} {{ '{{' }} lb_id {{ '}}' }} %f %ci %cp %t %{+Q}r %ST %B %U % [ssl_c_verify] %{+Q} [ssl_c_s_dn] %b %s %Tt %tsc"**	テナントトラフィックフローログの形式 英数字は特定の octavia フィールドを表し、中括弧 ({}) は置換変数です。

関連情報

- 『オーバークラウドの高度 **なカスタマイズ**』 の「オーバークラウド作成時の環境ファイル の追加」

5.7. LOAD-BALANCING サービスインスタンスのテナントフローログ形式

Load-balancing サービスインスタンス (amphorae) のテナントフローがログを記録するログ形式は HAProxy ログ形式です。2 つの例外は、**project_id** と **lb_id** 変数で、その値は amphora プロバイダードライバーによって提供されます。

例

rsyslog を syslog レシーバーとして使用するログエントリーの例を以下に示します。

```
Jun 12 00:44:13 amphora-3e0239c3-5496-4215-b76c-6abbe18de573 haproxy[1644]: 5408b89aa45b48c69a53dca1aaec58db fd8f23df-960b-4b12-ba62-2b1dff661ee7 261ecfc2-9e8e-4bba-9ec2-3c903459a895 172.24.4.1 41152 12/Jun/2019:00:44:13.030 "GET / HTTP/1.1" 200 76 73 - "" e37e0e04-68a3-435b-876c-cffe4f2138a4 6f2720b3-27dc-4496-9039-1aafe2fee105 4 --
```

備考

- ハイフン (-) は、不明または接続に該当しない任意のフィールドを示します。
- 上記のサンプルログエントリーの接頭辞は rsyslog レシーバーに由来するもので、amphora からの syslog メッセージの一部ではありません。

  ```
  Jun 12 00:44:13 amphora-3e0239c3-5496-4215-b76c-6abbe18de573 haproxy[1644]:”
  ```

デフォルト

デフォルトの amphora テナントフローログの形式は以下のとおりです。

```
`"{{ '{{' }} project_id {{ '}}' }} {{ '{{' }} lb_id {{ '}}' }} %f %ci %cp %t %{+Q}r %ST %B %U %[ssl_c_verify] %{+Q}[ssl_c_s_dn] %b %s %Tt %tsc"`
```

形式の説明は、以下の表を参照してください。

表5.4 テナントフローログ形式の変数定義のデータ変数

変数	型	フィールド名
{{project_id}}	UUID	プロジェクト ID (amphora プロバイダードライバーからの置換変数)
{{lb_id}}	UUID	ロードバランサー ID (amphora プロバイダードライバーからの置換変数)
%f	string	**frontend_name**
%ci	IP アドレス	**client_ip**
%cp	numeric	**client_port**
%t	date	**date_time**
%ST	numeric	**status_code**
%B	numeric	**bytes_read**
%U	numeric	**bytes_uploaded**
%ssl_c_verify	ブール値	**client_certificate_verify** (0 または 1)
%ssl_c_s_dn	string	**client_certificate_distinguised_name**
%b	string	**pool_id**
%s	string	**member_id**
%Tt	numeric	**processing_time** (ミリ秒)
%tsc	string	**termination_state** (cookie ステータスあり)

関連情報

- HAProxy ドキュメント の「Custom log format」

第6章 LOAD-BALANCING サービスフレーバーの設定

Load-balancing サービス (octavia) **フレーバー** は、作成するプロバイダー設定オプションのセットです。ユーザーがロードバランサーを要求する場合、定義されたフレーバーのいずれかを使用してロードバランサーを構築するように指定できます。各負荷分散プロバイダードライバー用にフレーバーを定義して、該当するプロバイダーの一意の機能を公開します。

新しい Load-balancing サービスフレーバーを作成するには、以下の手順を実施します。

1. フレーバーで設定する負荷分散プロバイダーの機能を決定する。
2. 選択したフレーバー機能でフレーバープロファイルを作成する。
3. フレーバーを作成する。
   - 「Load-balancing サービスプロバイダー機能の一覧」
   - 「フレーバープロファイルの定義」
   - 「Load-balancing サービスフレーバーの作成」

6.1. LOAD-BALANCING サービスプロバイダー機能の一覧

それぞれの Load-balancing サービス (octavia) プロバイダードライバーが公開する機能の一覧を確認することができます。

前提条件

- OpenStack の管理者権限を持っている必要があります。

手順

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. source コマンドでアンダークラウドの認証情報ファイルを読み込みます。

```
$ source ~/stackrc
```

3. 各ドライバーの機能を一覧表示します。

```
$ openstack loadbalancer provider capability list <provider>
```

<provider> をプロバイダーの名前または UUID に置き換えます。

例

```
$ openstack loadbalancer provider capability list amphora
```

コマンド出力には、プロバイダーがサポートするすべての機能が一覧表示されます。

出力例

```
+---------------------+-------------------------------------------------+
| name                | description                                     |
```

```
+----------------------+-----------------------------------------------+
| loadbalancer_topology | The load balancer topology. One of: SINGLE - One  |
|                      | amphora per load balancer. ACTIVE_STANDBY - Two   |
|                      | amphora per load balancer.                        |
| ...                  | ...                                           |
+----------------------+-----------------------------------------------+
```

4. 作成するフレーバーに追加する機能の名前をメモします。

関連情報

- 「フレーバープロファイルの定義」
- **Command Line Interface Reference** の loadbalancer コマンドを参照してください。

6.2. フレーバープロファイルの定義

Load-balancing サービス (octavia) フレーバープロファイルには、プロバイダードライバー名と機能の一覧が含まれます。フレーバープロファイルを使用して、ユーザーがロードバランサーを作成するために指定するフレーバーを作成します。

前提条件

- OpenStack の管理者権限を持っている必要があります。
- どの負荷分散プロバイダーがフレーバープロファイルに追加するかを把握しておく必要があります。

手順

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. source コマンドでアンダークラウドの認証情報ファイルを読み込みます。

```
$ source ~/stackrc
```

3. フレーバープロファイルを作成します。

```
$ openstack loadbalancer flavorprofile create --name <profile_name> --provider
<provider_name> --flavor-data '{"<capability>": "<value>"}'
```

例

```
$ openstack loadbalancer flavorprofile create --name amphora-single-profile --provider
amphora --flavor-data '{"loadbalancer_topology": "SINGLE"}'
```

出力例

```
+---------------+--------------------------------------+
| Field         | Value                                |
+---------------+--------------------------------------+
| id            | 72b53ac2-b191-48eb-8f73-ed012caca23a |
| name          | amphora-single-profile               |
```

```
| provider_name | amphora                          |
| flavor_data   | {"loadbalancer_topology": "SINGLE"} |
+---------------+----------------------------------+
```

以下の例では、amphora プロバイダー用にフレーバープロファイルが作成されます。このプロファイルがフレーバーで指定されている場合には、フレーバーを使用して作成するロードバランサーは単一の amphora ロードバランサーです。

検証手順

- フレーバープロファイルの作成時に、Load-balancing サービスはフレーバーの値をプロバイダーに検証し、プロバイダーが指定した機能をサポートできるようにします。

関連情報

- 「Load-balancing サービスフレーバーの作成」
- **Command Line Interface Reference** の loadbalancer flavorprofile create コマンドを参照してください。

6.3. LOAD-BALANCING サービスフレーバーの作成

フレーバープロファイルを使用して、Load-balancing サービス (octavia) 用にユーザー向けのフレーバーを作成します。フレーバーに割り当てる名前は、ユーザーがロードバランサーの作成時に指定する値です。

前提条件

- OpenStack の管理者権限を持っている必要があります。
- フレーバープロファイルを作成している必要があります。

手順

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. source コマンドでアンダークラウドの認証情報ファイルを読み込みます。

```
$ source ~/stackrc
```

3. フレーバーを作成します。

```
$ openstack loadbalancer flavor create --name <flavor_name> \
--flavorprofile <flavor-profile> --description "<string>"
```

ヒント

ユーザーは提供するフレーバーの機能を理解できるように、詳細な説明を指定します。

例

```
$ openstack loadbalancer flavor create --name standalone-lb --flavorprofile amphora-single-profile --description "A non-high availability load balancer for testing."
```

出力例

```
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| id                | 25cda2d8-f735-4744-b936-d30405c05359 |
| name              | standalone-lb                        |
| flavor_profile_id | 72b53ac2-b191-48eb-8f73-ed012caca23a |
| enabled           | True                                 |
| description       | A non-high availability load         |
|                   | balancer for testing.                |
+-------------------+--------------------------------------+
```

この例では、フレーバーが定義されている。このフレーバーを指定すると、Load-balancing サービスインスタンス (amphora) を使用するロードバランサーが作成され、高可用性はありません。

注記

無効にしたフレーバーはユーザーが引き続き表示されますが、ユーザーは、無効にしたフレーバーを使用してロードバランサーを作成することはできません。

関連情報

- 「フレーバープロファイルの定義」
- Command Line Interface Reference の loadbalancer flavor create

第7章 LOAD-BALANCING サービスの監視

負荷分散の動作を維持するには、ロードバランサー管理ネットワークを使用し、負荷分散のヘルスモニターを作成、変更、および削除できます。

7.1. 負荷分散管理ネットワーク

Red Hat OpenStack Platform (RHOSP) Load-balancing サービス (octavia) は、**ロードバランサー管理ネットワーク** と呼ばれるプロジェクトネットワークを介してロードバランサーを監視します。Load-balancing サービスを実行するホストには、ロードバランサー管理ネットワークに接続するためのインターフェースが必要です。サポートされるインターフェース設定は、neutron Modular Layer 2 プラグインと Open Virtual Network メカニズムドライバーの組み合わせ (ML2/OVN) または Open vSwitch メカニズムドライバーの組み合わせ (ML2/OVS) で機能します。他のメカニズムドライバーを使用するインターフェースの使用はテストされていません。

デプロイメント時に作成されるデフォルトのインターフェースは、デフォルトの統合ブリッジ **br-int** 上の内部 Open vSwitch (OVS) ポートです。これらのインターフェースを、ロードバランサー管理ネットワークに割り当てられた実際の Networking サービス (neutron) ポートに関連付ける必要があります。

デフォルトでは、デフォルトインターフェースの名前は、**o-hm0** です。これらは、Load-balancing サービスホストの標準のインターフェース設定ファイルで定義されます。RHOSP director は、デプロイメント時に Networking サービスポートおよび各 Load-balancing サービスホストのインターフェースを自動的に設定します。ポート情報とテンプレートは、以下を含むインターフェース設定ファイルの作成に使用されます。

- IP およびネットマスクを含む IP ネットワークアドレス情報
- MTU 設定
- MAC アドレス
- Networking サービスのポート ID

デフォルトの OVS の場合、Networking サービスのポート ID は追加データを OVS ポートに登録するのに使用されます。Networking サービスは、このインターフェースをポートに属するものとして認識し、ロードバランサー管理ネットワーク上で通信できるように OVS を設定します。

デフォルトでは、RHOSP は、Load-balancing サービスコントローラーが TCP ポート 9443 で仮想マシンインスタンス (amphora) と通信できるようにするセキュリティーグループおよびファイアウォールルールを設定し、amphora からのハートビートメッセージが UDP ポート 5555 のコントローラーに到達できるようにします。メカニズムドライバーによっては、負荷分散サービスとロードバランサー間の通信を可能にするために、追加または代替要件が必要になる場合があります。

7.2. LOAD-BALANCING サービスインスタンスの監視

Load-balancing サービス (octavia) は、負荷分散インスタンス (amphora) を監視し、amphora に異常が発生した場合にフェイルオーバーを開始して、置き換えます。フェイルオーバーが発生すると、Load-balancing サービスは **/var/log/containers/octavia** のコントローラー上の対応するヘルスマネージャーのログにフェイルオーバーを記録します。

ログ解析を使用して、フェイルオーバーの傾向を監視し、早い段階で問題に対処します。Networking サービス (neutron) の接続の問題、サービス拒否攻撃、および Compute サービス (nova) の異常などの問題により、ロードバランサーのフェイルオーバーの頻度が上がります。

7.3. LOAD-BALANCING サービスプールメンバーの監視

Load-balancing サービス (octavia) は、ベースとなる負荷分散サブシステムからの健全性情報を使用して、負荷分散プールのメンバーの健全性を判断します。健全性情報は Load-balancing サービスのデータベースにストリーミングされ、ステータスツリーまたは他の API メソッドにより利用できるようになります。重要なアプリケーションの場合、定期的な間隔で健全性情報をポーリングする必要があります。

7.4. ロードバランサーのプロビジョニングステータスの監視

ロードバランサーのプロビジョニングステータスをモニターし、プロビジョニングのステータスが **ERROR** の場合にはアラートを送信することができます。アプリケーションがプールに定期的な変更を加え、いくつかの **PENDING** ステージに入ったときにトリガーするように、アラートを構成しないでください。

ロードバランサーオブジェクトのプロビジョニングステータスは、コンタクトして作成、更新、および削除要求を正常にプロビジョニングするコントロールプレーンの性能を反映しています。ロードバランサーオブジェクトの操作ステータスは、ロードバランサーの現在の機能を報告します。

たとえば、ロードバランサーのプロビジョニングステータスが **ERROR** で、操作ステータスが **ONLINE** となる場合あります。これは、最後に要求されたロードバランサー設定への更新が正常に完了しないという、Networking (neutron) の障害により生じる可能性があります。この場合、ロードバランサーはロードバランサー経由でトラフィックの処理が継続しますが、最新の設定の更新が適用されていない可能性があります。

7.5. ロードバランサー機能の監視

ロードバランサーとその子オブジェクトの動作ステータスを監視できます。

また、ロードバランサーリスナーに接続し、クラウド外から監視する外部モニタリングサービスを使用することもできます。外部のモニタリングサービスでは、ルーターの失敗、ネットワーク接続の問題など、Load-balancing サービス (octavia) 外にロードバランサーの機能に影響を与える可能性のある障害が発生しているかどうかを示します。

7.6. LOAD-BALANCING サービスヘルスモニターの概要

Load-balancing サービス (octavia) のヘルスモニターは、各バックエンドメンバーサーバーで定期的なヘルスチェックを実行するプロセスで、障害が発生したサーバーを事前検出し、それらをプールから一時的に除外します。

ヘルスモニターが障害が発生したサーバーを検出すると、サーバーをプールから除外し、メンバーを **ERROR** とマークします。サーバーを修正して再度稼動状態にすると、ヘルスモニターはメンバーのステータスを **ERROR** から **ONLINE** に自動的に変更し、トラフィックをこれに渡すことを再開します。

実稼働環境のロードバランサーでは、ヘルスモニターを常に使用します。ヘルスモニターがない場合は、失敗したサーバーはプールから削除されません。これにより、Web クライアントのサービスの中断が発生する可能性があります。

以下に示すように、ヘルスモニターにはいくつかの種別があります。

HTTP

デフォルトでは、アプリケーションサーバーの / パスを調べます。

HTTPS

HTTP ヘルスモニターとまったく同じように動作しますが、TLS バックエンドサーバーが対象です。
サーバーがクライアント証明書の検証を実行する場合、HAProxy には有効な証明書がありません。このような場合、TLS-HELLO ヘルスモニタリングが代替手段です。

TLS-HELLO

バックエンドサーバーが SSLv3-client hello メッセージに応答するようにします。
TLS-HELLO ヘルスモニターは、ステータスコードやボディーの内容などの他のヘルスメトリックを確認しません。

PING

定期的に ICMP ping リクエストをバックエンドサーバーに送信します。
これらのヘルスチェックに合格するように、PING を許可するようにバックエンドサーバーを構成する必要があります。

重要

PING ヘルスモニターは、メンバーに到達可能で ICMP エコーリクエストに応答するかどうかを確認するだけです。PING ヘルスモニターは、インスタンスで実行されるアプリケーションが正常であるかどうかを確認しません。ICMP エコーリクエストが有効なヘルスチェックである場合にのみ、PING ヘルスモニターを使用します。

TCP

バックエンドサーバープロトコルポートへの TCP 接続を開きます。
TCP アプリケーションは TCP 接続を開き、TCP ハンドシェイクの後にデータを送信せずに接続を閉じる必要があります。

UDP-CONNECT

基本的な UDP ポート接続を実行します。
Destination Unreachable (ICMP タイプ 3) がメンバーサーバーで有効化されていない場合、またはセキュリティールールによってブロックされている場合には、UDP-CONNECT ヘルスモニターが正しく動作しないことがあります。この場合、メンバーサーバーは、実際にダウンしている時に **ONLINE** の稼働ステータスとしてマークされる可能性があります。

7.7. LOAD-BALANCING サービスヘルスモニターの作成

負荷分散サービス (octavia) ヘルスモニターを使用して、ユーザーのサービスの中断を回避します。各バックエンドメンバーサーバーで定期的なヘルスチェックを実行するプロセスで、障害が発生したサーバーを事前検出し、それらをプールから一時的に除外します。

手順

1. 認証情報ファイルに source を実行します。

   例

   ```
   $ source ~/overcloudrc
   ```

2. お使いのサイトに適した引数値を使用して、**openstack loadbalancer healthmonitor create** コマンドを実行します。

   - すべてのヘルスモニタータイプには、次の構成可能な引数が必要です。

     <pool>
     監視対象のバックエンドメンバーサーバーのプールの名前または ID。

     --type
     ヘルスモニターのタイプ。**HTTP**、**HTTPS**、**PING**、**SCTP**、**TCP**、**TLS-HELLO**、または **UDP-CONNECT** のいずれか。

     --delay
     ヘルスチェックの間隔 (秒単位)。

     --timeout
     指定したヘルスチェックが完了するまで待機する時間 (秒単位)。**timeout** は、常に **delay** よりも小さくなければなりません。

     --max-retries
     バックエンドサーバーが停止しているとみなされるまでに失敗する必要のあるヘルスチェックの数。また、障害が発生したバックエンドサーバーが再度稼働中とみなされるために成功しなければならないヘルスチェックの数。

   - さらに、HTTP ヘルスモニタータイプには、デフォルトで設定されている次の引数も必要です。

     --url-path
     バックエンドサーバーから取得される URL のパスの部分。デフォルトでは、これは / です。

     --http-method
     url_path を取得するために使用される HTTP メソッド。デフォルトでは、これは **GET** だけです。

     --expected-codes
     ヘルスチェックの成功を表す HTTP ステータスコードの一覧。デフォルトでは、これは **200** です。

例

```
$ openstack loadbalancer healthmonitor create --name my-health-monitor --delay 10 --max-retries 4 --timeout 5 --type TCP lb-pool-1
```

-

検証

- **openstack loadbalancer healthmonitor list** コマンドを実行し、ヘルスモニターが実行していることを確認します。

関連情報

- Command Line Interface Referenceの loadbalancer healthmonitor create

7.8. LOAD-BALANCING サービスヘルスモニターの変更

メンバーにプローブを送信する間隔、接続タイムアウト間隔、要求の HTTP メソッドなどを変更する場合は、負荷分散サービス (octavia) ヘルスモニターの構成を変更できます。

手順

1. 認証情報ファイルに source を実行します。

   例

   ```
   $ source ~/overcloudrc
   ```

2. ヘルスモニター(**my-health-monitor**)を変更します。
   この例では、ユーザーは、ヘルスモニターがメンバーにプローブを送信するまで待機する時間を秒単位で変更しています。

   例

   ```
   $ openstack loadbalancer healthmonitor set my_health_monitor --delay 600
   ```

検証

- **openstack loadbalancer healthmonitor show** コマンドを実行して、構成の変更を確認します。

  ```
  $ openstack loadbalancer healthmonitor show my_health_monitor
  ```

関連情報

- Command Line Interface Referenceの loadbalancer healthmonitor
- Command Line Interface Referenceの loadbalancer healthmonitor show

7.9. LOAD-BALANCING サービスヘルスモニターの削除

負荷分散サービス (octavia) ヘルスモニターを削除できます。

ヒント

ヘルスモニターを削除する代わりに、**openstack loadbalancer healthmonitor set --disable** コマンドを使用して無効にすることもできます。

手順

1. 認証情報ファイルに source を実行します。

   例

   ```
   $ source ~/overcloudrc
   ```

2. ヘルスモニター(**my-health-monitor**)を削除します。

   例

   ```
   $ openstack loadbalancer healthmonitor delete my-health-monitor
   ```

検証

- **openstack loadbalancer healthmonitor list** コマンドを実行して、削除したヘルスモニターが存在しないことを確認します。

関連情報

- Command Line Interface Reference の loadbalancer healthmonitor delete

7.10. LOAD-BALANCING サービス HTTP ヘルスモニターのベストプラクティス

Web アプリケーションでヘルスチェックを生成するコードを作成するときは、次のベストプラクティスを使用してください。

- ヘルスモニター **url-path** には、読み込む認証は必要ありません。
- **expected-codes** を代わりに指定しない限り、デフォルトでは、ヘルスモニター **url-path** はサーバーが正常であることを示すために **HTTP 200 OK** ステータスコードを返します。
- ヘルスチェックは、アプリケーションが完全に正常であることを確認するために、十分な内部チェックを実行します。アプリケーションについて、以下の条件を満たしていることを確認します。
  - 必要なデータベースまたは他の外部ストレージ接続が稼働している。
  - アプリケーションを実行するサーバーで読み込みが許可されている。
  - お使いのサイトがメンテナンスモードにない。
  - アプリケーションに固有のテストが機能する。
- ヘルスチェックによって生成されるページのサイズは小さくする必要があります。
  - 1 秒未満の間隔で戻ります。
  - アプリケーションサーバーに大きな負荷は発生させません。
- ヘルスチェックによって生成されたページはキャッシュされませんが、ヘルスチェックを実行するコードはキャッシュされたデータを参照する場合があります。

たとえば、cron を使用してさらに広範なヘルスチェックを実行し、結果をディスクに保存すると便利です。ヘルスモニターの **url-path** でページを生成するコードは、実行するテストにこの cron ジョブの結果を組み込みます。

- Load-balancing サービスは、返された HTTP ステータスコードのみを処理し、ヘルスチェックが頻繁に実行されるため、**HEAD** または **OPTIONS** HTTP メソッドを使用してページ全体の処理を省略できます。

第8章 セキュアではない HTTP 用ロードバランサーの作成

非セキュア HTTP ネットワークトラフィック用に次のロードバランサーを作成できます。

- 「ヘルスモニターを使用した HTTP ロードバランサーの作成」
- 「フローティング IP を使用する HTTP ロードバランサーの作成」
- 「セッション永続性による HTTP ロードバランサーの作成」

8.1. ヘルスモニターを使用した HTTP ロードバランサーの作成

Red Hat OpenStack Platform Networking Service (neutron) フローティング IP と互換性のないネットワークの場合、安全でない HTTP アプリケーションのネットワークトラフィックを管理するためのロードバランサーを作成します。ヘルスモニターを作成して、バックエンドメンバーが引き続き利用できるようにします。

前提条件

- TCP ポート 80 でセキュアではない HTTP アプリケーションをホストするバックエンドサーバーが含まれるプライベートサブネット
- プライベートサブネット上のバックエンドサーバーは、URL パス / でヘルスチェックを使用して構成されます。
- インターネットから到達できる共有外部 (パブリック) サブネット。

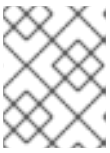
手順

1. 認証情報ファイルに source を実行します。

   例

   ```
   $ source ~/overcloudrc
   ```

2. パブリックサブネット (**public-subnet**) にロードバランサー (**lb1**) を作成します。

   注記

   丸かっこ内の値は、この手順のコマンド例で使用されるサンプルの値です。これらのサンプル値を、実際のサイトに適した値に置き換えてください。

   例

   ```
   $ openstack loadbalancer create --name lb1 --vip-subnet-id public_subnet
   ```

3. ロードバランサーの状態を確認します。

   例

   ```
   $ openstack loadbalancer show lb1
   ```

4. 次の手順に進む前に、**provisioning_status** が **ACTIVE** であることを確認してください。

5. ポート (**80**) にリスナー (**listener1**) を作成します。

   例

   ```
   $ openstack loadbalancer listener create --name listener1 --protocol HTTP --protocol-port 80
   lb1
   ```

6. リスナーの状態を確認します。

   例

   ```
   $ openstack loadbalancer listener show listener1
   ```

   次の手順に進む前に、ステータスが **ACTIVE** であることを確認してください。

7. リスナーのデフォルトプール (**pool1**) を作成します。

   例

   ```
   $ openstack loadbalancer pool create --name pool1 --lb-algorithm ROUND_ROBIN --listener
   listener1 --protocol HTTP
   ```

8. バックエンドサーバーに接続するプール (**pool1**) にヘルスモニターを作成し、パス (/) をテストします。

   例

   ```
   $ openstack loadbalancer healthmonitor create --delay 5 --max-retries 4 --timeout 10 --type
   HTTP --url-path / pool1
   ```

9. プライベートサブネット (**private_subnet**) のロードバランサーメンバー (**192.0.2.10** および **192.0.2.11**) をデフォルトのプールに追加します。

   例

   ```
   $ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.10 --
   protocol-port 80 pool1
   $ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.11 --
   protocol-port 80 pool1
   ```

検証

1. ロードバランサー (lb1) の設定を表示および確認します。

   例

   ```
   $ openstack loadbalancer show lb1
   ```

   出力例

   ```
   +---------------------+-------------------------------------+
   | Field               | Value                               |
   ```

```
+---------------------+--------------------------------------+
| admin_state_up      | True                                 |
| created_at          | 2022-01-15T11:11:09                  |
| description         |                                      |
| flavor              |                                      |
| id                  | 788fe121-3dec-4e1b-8360-4020642238b0 |
| listeners           | 09f28053-fde8-4c78-88b9-0f191d84120e |
| name                | lb1                                  |
| operating_status    | ONLINE                               |
| pools               | 627842b3-eed8-4f5f-9f4a-01a738e64d6a |
| project_id          | dda678ca5b1241e7ad7bf7eb211a2fd7     |
| provider            | amphora                              |
| provisioning_status | ACTIVE                               |
| updated_at          | 2022-01-15T11:12:13                  |
| vip_address         | 198.51.100.12                        |
| vip_network_id      | 9bca13be-f18d-49a5-a83d-9d487827fd16 |
| vip_port_id         | 69a85edd-5b1c-458f-96f2-b4552b15b8e6 |
| vip_qos_policy_id   | None                                 |
| vip_subnet_id       | 5bd7334b-49b3-4849-b3a2-b0b83852dba1 |
+---------------------+--------------------------------------+
```

2. ヘルスモニターが存在し正常に機能する場合は、各メンバーのステータスを確認することができます。
動作中のメンバー (**b85c807e-4d7c-4cbd-b725-5e8afddf80d2**) の **operating_status** の値は **ONLINE** です。

例

```
$ openstack loadbalancer member show pool1 b85c807e-4d7c-4cbd-b725-5e8afddf80d2
```

出力例

```
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| address             | 192.0.2.10                           |
| admin_state_up      | True                                 |
| created_at          | 2022-01-15T11:16:23                  |
| id                  | b85c807e-4d7c-4cbd-b725-5e8afddf80d2 |
| name                |                                      |
| operating_status    | ONLINE                               |
| project_id          | dda678ca5b1241e7ad7bf7eb211a2fd7     |
| protocol_port       | 80                                   |
| provisioning_status | ACTIVE                               |
| subnet_id           | 5bd7334b-49b3-4849-b3a2-b0b83852dba1 |
| updated_at          | 2022-01-15T11:20:45                  |
| weight              | 1                                    |
| monitor_port        | None                                 |
| monitor_address     | None                                 |
| backup              | False                                |
+---------------------+--------------------------------------+
```

関連情報

- Command Line Interface ReferenceのloadbalancerCommand Line Interface Referenceのloadbalancer

8.2. フローティング IP を使用する HTTP ロードバランサーの作成

セキュアでない HTTP アプリケーションのネットワークトラフィックを管理するには、Floating IP に依存する仮想 IP (VIP) によるロードバランサーを作成することができます。Floating IP を使用する利点は、割り当てられた IP の制御を維持することです。このことは、ロードバランサーを移動、破棄、または再作成する場合に必要です。バックエンドメンバーを利用できる状態に保つためのヘルスモニターも作成するのがベストプラクティスです。

注記

Floating IP は IPv6 ネットワークでは機能しません。

前提条件

- TCP ポート 80 でセキュアではない HTTP アプリケーションをホストするバックエンドサーバーが含まれるプライベートサブネット
- バックエンドサーバーは、URL パス / でヘルスチェックを使用して構成されます。
- ロードバランサー VIP で使用するフローティング IP。
- フローティング IP に使用するためにインターネットから到達できる Red Hat OpenStack Platform Networking サービス (neutron) 共有外部 (パブリック) サブネット。

手順

1. 認証情報ファイルに source を実行します。

   例

   ```
   $ source ~/overcloudrc
   ```

2. プライベートサブネット (**private_subnet**) にロードバランサー (**lb1**) を作成します。

   注記

   丸かっこ内の値は、この手順のコマンド例で使用されるサンプルの値です。これらのサンプル値を、実際のサイトに適した値に置き換えてください。

   例

   ```
   $ openstack loadbalancer create --name lb1 --vip-subnet-id private_subnet
   ```

3. **load_balancer_vip_port_id** の値に注意してください。これは、後のステップで指定する必要があるためです。

4. ロードバランサーの状態を確認します。

   例

   ```
   $ openstack loadbalancer show lb1
   ```

5. 次の手順に進む前に、**provisioning_status** が **ACTIVE** であることを確認してください。
6. ポート (**80**) にリスナー (**listener1**) を作成します。

   例

   ```
   $ openstack loadbalancer listener create --name listener1 --protocol HTTP --protocol-port 80 lb1
   ```

7. リスナーのデフォルトプール (**pool1**) を作成します。

   例

   ```
   $ openstack loadbalancer pool create --name pool1 --lb-algorithm ROUND_ROBIN --listener listener1 --protocol HTTP
   ```

8. バックエンドサーバーに接続するプール (**pool1**) にヘルスモニターを作成し、パス (/) をテストします。

   例

   ```
   $ openstack loadbalancer healthmonitor create --delay 5 --max-retries 4 --timeout 10 --type HTTP --url-path / pool1
   ```

9. プライベートサブネットのロードバランサーメンバー (**192.0.2.10** および **192.0.2.11**) をデフォルトのプールに追加します。

   例

   ```
   $ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.10 --protocol-port 80 pool1
   $ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.11 --protocol-port 80 pool1
   ```

10. 共有外部サブネット (**public**) に Floating IP アドレスを作成します。

    例

    ```
    $ openstack floating ip create public
    ```

11. 後のステップで指定する必要があるため、**floating_ip_address** の値に注意してください。
12. このフローティングIP (**203.0.113.0**) をロードバランサー **vip_port_id** (**69a85edd-5b1c-458f-96f2-b4552b15b8e6**) に関連付けます。

    例

    ```
    $ openstack floating ip set --port 69a85edd-5b1c-458f-96f2-b4552b15b8e6 203.0.113.0
    ```

検証

1. HTTP トラフィックが Floating IP (**203.0.113.0**) を使用してロードバランサー全体に流れることを確認します。

   例

   ```
   $ curl -v http://203.0.113.0 --insecure
   ```

   出力例

   ```
   * About to connect() to 203.0.113.0 port 80 (#0)
   *   Trying 203.0.113.0...
   * Connected to 203.0.113.0 (203.0.113.0) port 80 (#0)
   > GET / HTTP/1.1
   > User-Agent: curl/7.29.0
   > Host: 203.0.113.0
   > Accept: */*
   >
   < HTTP/1.1 200 OK
   < Content-Length: 30
   <
   * Connection #0 to host 203.0.113.0 left intact
   ```

2. ヘルスモニターが存在し正常に機能する場合は、各メンバーのステータスを確認することができます。
   動作中のメンバー (**b85c807e-4d7c-4cbd-b725-5e8afddf80d2**) の **operating_status** の値は **ONLINE** です。

   例

   ```
   $ openstack loadbalancer member show pool1 b85c807e-4d7c-4cbd-b725-5e8afddf80d2
   ```

   出力例

   ```
   +---------------------+--------------------------------------+
   | Field               | Value                                |
   +---------------------+--------------------------------------+
   | address             | 192.0.02.10                          |
   | admin_state_up      | True                                 |
   | created_at          | 2022-01-15T11:11:23                  |
   | id                  | b85c807e-4d7c-4cbd-b725-5e8afddf80d2 |
   | name                |                                      |
   | operating_status    | ONLINE                               |
   | project_id          | dda678ca5b1241e7ad7bf7eb211a2fd7     |
   | protocol_port       | 80                                   |
   | provisioning_status | ACTIVE                               |
   | subnet_id           | 5bd7334b-49b3-4849-b3a2-b0b83852dba1 |
   | updated_at          | 2022-01-15T11:28:42                  |
   | weight              | 1                                    |
   | monitor_port        | None                                 |
   | monitor_address     | None                                 |
   | backup              | False                                |
   +---------------------+--------------------------------------+
   ```

関連情報

- Command Line Interface ReferenceのloadbalancerCommand Line Interface Reference
- Command Line Interface Reference の floating

8.3. セッション永続性による HTTP ロードバランサーの作成

セキュアでない HTTP アプリケーションのネットワークトラフィックを管理するには、セッション永続性を追跡するロードバランサーを作成することができます。これにより、リクエスト受け取ると、ロードバランサーは、同じクライアントからの後続のリクエストを同じバックエンドサーバーに転送します。セッションの永続性は、時間とメモリーを節約することで負荷分散を最適化します。

前提条件

- TCP ポート 80 でセキュアではない HTTP アプリケーションをホストするバックエンドサーバーが含まれるプライベートサブネット
- バックエンドサーバーは、URL パス / でヘルスチェックを使用して構成されます。
- インターネットから到達できる共有外部 (パブリック) サブネット。
- ネットワークトラフィックの負荷分散を行うセキュアではない Web アプリケーションで、クッキーが有効になっている。

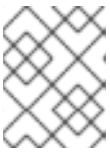
手順

1. 認証情報ファイルに source を実行します。

   例

   ```
   $ source ~/overcloudrc
   ```

2. パブリックサブネット (**public-subnet**) にロードバランサー (**lb1**) を作成します。

   注記

   丸かっこ内の値は、この手順のコマンド例で使用されるサンプルの値です。これらのサンプル値を、実際のサイトに適した値に置き換えてください。

   例

   ```
   $ openstack loadbalancer create --name lb1 --vip-subnet-id public_subnet
   ```

3. ロードバランサーの状態を確認します。

   例

   ```
   $ openstack loadbalancer show lb1
   ```

4. 次の手順に進む前に、**provisioning_status** が **ACTIVE** であることを確認してください。

5. ポート (**80**) にリスナー (**listener1**) を作成します。

例

```
$ openstack loadbalancer listener create --name listener1 --protocol HTTP --protocol-port 80
lb1
```

6. クッキーのセッション永続性 (**PHPSESSIONID**) を定義するリスナーのデフォルトプール (**pool1**) を作成します。

例

```
$ openstack loadbalancer pool create --name pool1 --lb-algorithm ROUND_ROBIN --listener
listener1 --protocol HTTP --session-persistence
type=APP_COOKIE,cookie_name=PHPSESSIONID
```

7. バックエンドサーバーに接続するプール (**pool1**) にヘルスモニターを作成し、パス (/) をテストします。

例

```
$ openstack loadbalancer healthmonitor create --delay 5 --max-retries 4 --timeout 10 --type
HTTP --url-path / pool1
```

8. プライベートサブネット (**private_subnet**) のロードバランサーメンバー (**192.0.2.10** および **192.0.2.11**) をデフォルトのプールに追加します。

例

```
$ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.10 --
protocol-port 80 pool1
$ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.11 --
protocol-port 80 pool1
```

検証

1. ロードバランサー (lb1) の設定を表示および確認します。

例

```
$ openstack loadbalancer show lb1
```

出力例

```
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| admin_state_up      | True                                 |
| created_at          | 2022-01-15T11:11:58                  |
| description         |                                      |
| flavor              |                                      |
| id                  | 788fe121-3dec-4e1b-8360-4020642238b0 |
| listeners           | 09f28053-fde8-4c78-88b9-0f191d84120e |
| name                | lb1                                  |
| operating_status    | ONLINE                               |
```

```
| pools               | 627842b3-eed8-4f5f-9f4a-01a738e64d6a |
| project_id          | dda678ca5b1241e7ad7bf7eb211a2fd7     |
| provider            | amphora                              |
| provisioning_status | ACTIVE                               |
| updated_at          | 2022-01-15T11:28:42                  |
| vip_address         | 198.51.100.22                        |
| vip_network_id      | 9bca13be-f18d-49a5-a83d-9d487827fd16 |
| vip_port_id         | 69a85edd-5b1c-458f-96f2-b4552b15b8e6 |
| vip_qos_policy_id   | None                                 |
| vip_subnet_id       | 5bd7334b-49b3-4849-b3a2-b0b83852dba1 |
+---------------------+--------------------------------------+
```

2. ヘルスモニターが存在し正常に機能する場合は、各メンバーのステータスを確認することができます。
動作中のメンバー (**b85c807e-4d7c-4cbd-b725-5e8afddf80d2**) の **operating_status** の値は **ONLINE** です。

例

```
$ openstack loadbalancer member show pool1 b85c807e-4d7c-4cbd-b725-5e8afddf80d2
```

出力例

```
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| address             | 192.0.02.10                          |
| admin_state_up      | True                                 |
| created_at          | 2022-01-15T11:11:23                  |
| id                  | b85c807e-4d7c-4cbd-b725-5e8afddf80d2 |
| name                |                                      |
| operating_status    | ONLINE                               |
| project_id          | dda678ca5b1241e7ad7bf7eb211a2fd7     |
| protocol_port       | 80                                   |
| provisioning_status | ACTIVE                               |
| subnet_id           | 5bd7334b-49b3-4849-b3a2-b0b83852dba1 |
| updated_at          | 2022-01-15T11:28:42                  |
| weight              | 1                                    |
| monitor_port        | None                                 |
| monitor_address     | None                                 |
| backup              | False                                |
+---------------------+--------------------------------------+
```

関連情報

- Command Line Interface Referenceのloadbalancer

第9章 セキュアな HTTP 用ロードバランサーの作成

セキュアな HTTP (HTTPS) ネットワークトラフィックを管理するために、さまざまな種別のロードバランサーを作成することができます。

9.1. HTTPS 非終端ロードバランサーの概要

HTTPS 非終端ロードバランサーは、一般的な TCP ロードバランサーのように効果的に機能します。ロードバランサーは、Web クライアントからの未加工の TCP トラフィックを、HTTPS 接続が Web クライアントで終端するバックエンドサーバーに転送します。終端ではない HTTPS ロードバランサーはレイヤー 7 機能などの高度なロードバランサー機能をサポートしていませんが、証明書とキー自体を管理することにより、ロードバランサーのリソース使用率を低下させます。

9.2. HTTPS 非終端ロードバランサーの作成

アプリケーションがバックエンドメンバーサーバーで終端する HTTPS トラフィックを必要とする場合に (一般的に **HTTPS パススルー** と呼ばれます)、ロードバランサーリスナーに HTTPS プロトコルを使用できます。

前提条件

- TCP ポート 443 で TLS 暗号化 Web アプリケーションが設定された HTTPS アプリケーションをホストするバックエンドサーバーが含まれるプライベートサブネット
- バックエンドサーバーは、URL パス / でヘルスチェックを使用して構成されます。
- インターネットから到達できる共有外部 (パブリック) サブネット。

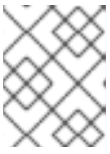
手順

1. 認証情報ファイルに source を実行します。

   例

   ```
   $ source ~/overcloudrc
   ```

2. パブリックサブネット (**public-subnet**) にロードバランサー (**lb1**) を作成します。



注記

丸かっこ内の値は、この手順のコマンド例で使用されるサンプルの値です。これらのサンプル値を、実際のサイトに適した値に置き換えてください。

例

```
$ openstack loadbalancer create --name lb1 --vip-subnet-id public_subnet
```

3. ロードバランサーの状態を監視します。

例

```
$ openstack loadbalancer show lb1
```

4. 次の手順に進む前に、**provisioning_status** が **ACTIVE** であることを確認してください。
5. ポート (**443**) にリスナー (**listener1**) を作成します。

例

```
$ openstack loadbalancer listener create --name listener1 --protocol HTTPS --protocol-port
443 lb1
```

6. リスナーのデフォルトプール (**pool1**) を作成します。

例

```
$ openstack loadbalancer pool create --name pool1 --lb-algorithm ROUND_ROBIN --listener
listener1 --protocol HTTPS
```

7. バックエンドサーバーに接続するプール (**pool1**) にヘルスモニターを作成し、パス (/) をテストします。

例

```
$ openstack loadbalancer healthmonitor create --delay 5 --max-retries 4 --timeout 10 --type
TLS-HELLO --url-path / pool1
```

8. プライベートサブネット (**private_subnet**) のロードバランサーメンバー (**192.0.2.10** および **192.0.2.11**) をデフォルトのプールに追加します。

例

```
$ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.10 --
protocol-port 443 pool1
$ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.11 --
protocol-port 443 pool1
```

検証

1. ロードバランサー (**lb1**) の設定を表示して確認します。

例

```
$ openstack loadbalancer show lb1
```

出力例

```
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| admin_state_up      | True                                 |
| created_at          | 2022-01-15T11:11:09                  |
| description         |                                      |
| flavor              |                                      |
| id                  | 788fe121-3dec-4e1b-8360-4020642238b0 |
| listeners           | 09f28053-fde8-4c78-88b9-0f191d84120e |
| name                | lb1                                  |
| operating_status    | ONLINE                               |
| pools               | 627842b3-eed8-4f5f-9f4a-01a738e64d6a |
| project_id          | dda678ca5b1241e7ad7bf7eb211a2fd7     |
| provider            | amphora                              |
| provisioning_status | ACTIVE                               |
| updated_at          | 2022-01-15T11:12:42                  |
| vip_address         | 198.51.100.11                        |
| vip_network_id      | 9bca13be-f18d-49a5-a83d-9d487827fd16 |
| vip_port_id         | 69a85edd-5b1c-458f-96f2-b4552b15b8e6 |
| vip_qos_policy_id   | None                                 |
| vip_subnet_id       | 5bd7334b-49b3-4849-b3a2-b0b83852dba1 |
+---------------------+--------------------------------------+
```

2. ヘルスモニターが存在し正常に機能する場合は、各メンバーのステータスを確認することができます。
動作中のメンバー (**b85c807e-4d7c-4cbd-b725-5e8afddf80d2**) の **operating_status** の値は **ONLINE** です。

例

```
$ openstack loadbalancer member show pool1 b85c807e-4d7c-4cbd-b725-5e8afddf80d2
```

出力例

```
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| address             | 192.0.2.10                           |
| admin_state_up      | True                                 |
| created_at          | 2022-01-15T11:11:09                  |
| id                  | b85c807e-4d7c-4cbd-b725-5e8afddf80d2 |
| name                |                                      |
| operating_status    | ONLINE                               |
| project_id          | dda678ca5b1241e7ad7bf7eb211a2fd7     |
| protocol_port       | 443                                  |
| provisioning_status | ACTIVE                               |
| subnet_id           | 5bd7334b-49b3-4849-b3a2-b0b83852dba1 |
| updated_at          | 2022-01-15T11:12:42                  |
| weight              | 1                                    |
| monitor_port        | None                                 |
```

```
| monitor_address     | None                                 |
| backup              | False                                |
+---------------------+--------------------------------------+
```

関連情報

- 『Manage Secrets with OpenStack Key Manager』
- **Command Line Interface Reference**の loadbalancer

9.3. TLS 終端 HTTPS ロードバランサーについて

TLS 終端 HTTPS ロードバランサーが実装されると、Web クライアントは Transport Layer Security (TLS) プロトコルを介してロードバランサーと通信します。ロードバランサーは TLS セッションを終端し、復号化されたリクエストをバックエンドサーバーに転送します。ロードバランサーで TLS セッションを終端することで、CPU 負荷の高い暗号化操作をロードバランサーにオフロードし、これによりロードバランサーはレイヤー 7 インスペクション等の高度な機能を使用することができます。

9.4. TLS 終端 HTTPS ロードバランサーの作成

TLS 終端 HTTPS ロードバランサーを使用することで、CPU 負荷の高い暗号化操作をロードバランサーにオフロードし、これによりロードバランサーはレイヤー 7 インスペクション等の高度な機能を使用することができます。バックエンドメンバーを利用できる状態に保つためのヘルスモニターも作成するのがベストプラクティスです。

前提条件

- TCP ポート 80 でセキュアではない HTTP アプリケーションをホストするバックエンドサーバーが含まれるプライベートサブネット
- バックエンドサーバーは、URL パス / でヘルスチェックを使用して構成されます。
- インターネットから到達できる共有外部 (パブリック) サブネット。
- TLS 公開鍵暗号化は、次の特性で構成されています。
  - ロードバランサーの仮想 IP アドレス (例: **www.example.com**) に割り当てられた DNS 名用に、TLS 証明書、鍵、および中間証明書チェーンが外部認証局 (CA) から取得される。
  - 証明書、鍵、および中間証明書チェーンが、現在のディレクトリー内の個別ファイルに保存される。
  - 鍵および証明書は PEM 形式でエンコードされる。
  - 鍵はパスフレーズで暗号化されない。
  - 中間証明書チェーンには PEM 形式でエンコードされた複数の証明書が含まれ、チェーンを形成する。
- Key Manager サービス (barbican) を使用するように Load-balancing サービス (octavia) が設定されている。詳しくは、**『Manage Secrets with OpenStack Key Manager』**を参照してください。

手順

1. 鍵 (**server.key**)、証明書 (**server.crt**)、および中間証明書チェーン (**ca-chain.crt**) を 1 つの PKCS12 ファイル (**server.p12**) に組み合わせます。

注記

丸かっこ内の値は、この手順のコマンド例で使用されるサンプルの値です。これらのサンプル値を、実際のサイトに適した値に置き換えてください。

例

```
$ openssl pkcs12 -export -inkey server.key -in server.crt -certfile ca-chain.crt -passout pass: -out server.p12
```

注記

PKCS12 ファイルを保護する場合は、以下の手順は機能しません。

2. 認証情報ファイルに source を実行します。

例

```
$ source ~/overcloudrc
```

3. Key Manager サービスを使用して、PKCS12 ファイルのシークレットリソース (**tls_secret1**) を作成します。

例

```
$ openstack secret store --name='tls_secret1' -t 'application/octet-stream' -e 'base64' --payload="$(base64 < server.p12)"
```

4. パブリックサブネット (**public_subnet**) にロードバランサー (**lb1**) を作成します。

例

```
$ openstack loadbalancer create --name lb1 --vip-subnet-id public_subnet
```

5. ロードバランサーの状態を監視します。

例

```
$ openstack loadbalancer show lb1
```

6. 次の手順に進む前に、**provisioning_status** が **ACTIVE** であることを確認してください。

7. **TERMINATED_HTTPS** リスナー (**listener1**) を作成し、リスナーのデフォルト TLS コンテナーとしてシークレットリソースを参照します。

例

```
$ openstack loadbalancer listener create --protocol-port 443 --protocol
TERMINATED_HTTPS --name listener1 --default-tls-container=$(openstack secret list | awk
'/ tls_secret1 / {print $2}') lb1
```

8. プール (**pool1**) を作成し、リスナーのデフォルトプールに設定します。

   例

```
$ openstack loadbalancer pool create --name pool1 --lb-algorithm ROUND_ROBIN --listener
listener1 --protocol HTTP
```

9. バックエンドサーバーに接続するプール (**pool1**) にヘルスモニターを作成し、パス (/) をテストします。

   例

```
$ openstack loadbalancer healthmonitor create --delay 5 --max-retries 4 --timeout 10 --type
HTTP --url-path / pool1
```

10. プライベートサブネット (**private_subnet**) 上のセキュアではない HTTP バックエンドサーバー (**192.0.2.10** および **192.0.2.11**) をプールに追加します。

    例

```
$ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.10 --
protocol-port 80 pool1
$ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.11 --
protocol-port 80 pool1
```

検証

1. ロードバランサー (**lb1**) の設定を表示して確認します。

   例

```
$ openstack loadbalancer show lb1
```

   出力例

```
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| admin_state_up      | True                                 |
| created_at          | 2022-01-15T11:11:09                  |
| description         |                                      |
| flavor              |                                      |
| id                  | 788fe121-3dec-4e1b-8360-4020642238b0 |
| listeners           | 09f28053-fde8-4c78-88b9-0f191d84120e |
| name                | lb1                                  |
| operating_status    | ONLINE                               |
| pools               | 627842b3-eed8-4f5f-9f4a-01a738e64d6a |
| project_id          | dda678ca5b1241e7ad7bf7eb211a2fd7     |
| provider            | amphora                              |
```

```
| provisioning_status | ACTIVE                               |
| updated_at          | 2022-01-15T11:12:42                  |
| vip_address         | 198.51.100.11                        |
| vip_network_id      | 9bca13be-f18d-49a5-a83d-9d487827fd16 |
| vip_port_id         | 69a85edd-5b1c-458f-96f2-b4552b15b8e6 |
| vip_qos_policy_id   | None                                 |
| vip_subnet_id       | 5bd7334b-49b3-4849-b3a2-b0b83852dba1 |
+---------------------+--------------------------------------+
```

2. ヘルスモニターが存在し正常に機能する場合は、各メンバーのステータスを確認することができます。
動作中のメンバー (**b85c807e-4d7c-4cbd-b725-5e8afddf80d2**) の **operating_status** の値は **ONLINE** です。

例

```
$ openstack loadbalancer member show pool1 b85c807e-4d7c-4cbd-b725-5e8afddf80d2
```

出力例

```
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| address             | 192.0.2.10                           |
| admin_state_up      | True                                 |
| created_at          | 2022-01-15T11:11:09                  |
| id                  | b85c807e-4d7c-4cbd-b725-5e8afddf80d2 |
| name                |                                      |
| operating_status    | ONLINE                               |
| project_id          | dda678ca5b1241e7ad7bf7eb211a2fd7     |
| protocol_port       | 80                                   |
| provisioning_status | ACTIVE                               |
| subnet_id           | 5bd7334b-49b3-4849-b3a2-b0b83852dba1 |
| updated_at          | 2022-01-15T11:12:42                  |
| weight              | 1                                    |
| monitor_port        | None                                 |
| monitor_address     | None                                 |
| backup              | False                                |
+---------------------+--------------------------------------+
```

関連情報

- 『Manage Secrets with OpenStack Key Manager』
- **Command Line Interface Reference**の loadbalancer

9.5. SNI を使用した TLS 終端 HTTPS ロードバランサーの作成

Server Name Indication (SNI) テクノロジーを使用する TLS 終端 HTTPS ロードバランサーでは、単一のリスナーに複数の TLS 証明書を含めることができ、ロードバランサーは、共有 IP の使用時に提示する証明書を認識することができます。バックエンドメンバーを利用できる状態に保つためのヘルスモニターも作成するのがベストプラクティスです。

前提条件

- TCP ポート 80 でセキュアではない HTTP アプリケーションをホストするバックエンドサーバーが含まれるプライベートサブネット
- バックエンドサーバーは、URL パス / でヘルスチェックを使用して構成されます。
- インターネットから到達できる共有外部 (パブリック) サブネット。
- TLS 公開鍵暗号化は、次の特性で構成されています。
  - ロードバランサーの仮想 IP アドレス (例: **www.example.com** および **www2.example.com**) に割り当てられた DNS 名用に、複数の TLS 証明書、鍵、および中間証明書チェーンが外部認証局 (CA) から取得される。
  - 鍵および証明書は PEM 形式でエンコードされる。
  - 鍵はパスフレーズで暗号化されない。
- Key Manager サービス (barbican) を使用するように Load-balancing サービス (octavia) が設定されている。詳しくは、**『Manage Secrets with OpenStack Key Manager』**を参照してください。

手順

1. SNI 一覧の TLS 証明書ごとに、鍵 (**server.key**)、証明書 (**server.crt**)、および中間証明書チェーン (**ca-chain.crt**) を 1 つの PKCS12 ファイル (**server.p12**) に組み合わせます。
以下の例では、それぞれの証明書 (**www.example.com** および **www2.example.com**) 用に、2 つの PKCS12 ファイル (**server.p12** および **server2.p12**) を作成します。

注記

丸かっこ内の値は、この手順のコマンド例で使用されるサンプルの値です。これらのサンプル値を、実際のサイトに適した値に置き換えてください。

```
$ openssl pkcs12 -export -inkey server.key -in server.crt -certfile ca-chain.crt -passout pass:
-out server.p12

$ openssl pkcs12 -export -inkey server2.key -in server2.crt -certfile ca-chain2.crt -passout
pass: -out server2.p12
```

2. 認証情報ファイルに source を実行します。

例

```
$ source ~/overcloudrc
```

3. Key Manager サービスを使用して、PKCS12 ファイルのシークレットリソース (**tls_secret1** および **tls_secret2**) を作成します。

```
$ openstack secret store --name='tls_secret1' -t 'application/octet-stream' -e 'base64' --
payload="$(base64 < server.p12)"
$ openstack secret store --name='tls_secret2' -t 'application/octet-stream' -e 'base64' --
payload="$(base64 < server2.p12)"
```

4. パブリックサブネット (**public_subnet**) にロードバランサー (**lb1**) を作成します。

```
$ openstack loadbalancer create --name lb1 --vip-subnet-id public_subnet
```

5. ロードバランサーの状態を監視します。

例

```
$ openstack loadbalancer show lb1
```

6. 次の手順に進む前に、**provisioning_status** が **ACTIVE** であることを確認してください。

7. TERMINATED_HTTPS リスナー (**listener1**) を作成し、SNI を使用して両方のシークレットリソースを参照します。
リスナーのデフォルト TLS コンテナーとして **tls_secret1** を参照します。)

```
$ openstack loadbalancer listener create --protocol-port 443 \
--protocol TERMINATED_HTTPS --name listener1 \
--default-tls-container=$(openstack secret list | awk '/ tls_secret1 / {print $2}') \
--sni-container-refs $(openstack secret list | awk '/ tls_secret1 / {print $2}') \
$(openstack secret list | awk '/ tls_secret2 / {print $2}') -- lb1
```

8. プール (**pool1**) を作成し、リスナーのデフォルトプールに設定します。

```
$ openstack loadbalancer pool create --name pool1 --lb-algorithm ROUND_ROBIN --listener
listener1 --protocol HTTP
```

9. バックエンドサーバーに接続するプール (**pool1**) にヘルスモニターを作成し、パス (/) をテストします。

例

```
$ openstack loadbalancer healthmonitor create --delay 5 --max-retries 4 --timeout 10 --type
HTTP --url-path / pool1
```

10. プライベートサブネット (**private_subnet**) 上のセキュアではない HTTP バックエンドサーバー (**192.0.2.10** および **192.0.2.11**) をプールに追加します。

```
$ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.10 --
protocol-port 80 pool1
$ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.11 --
protocol-port 80 pool1
```

検証

1. ロードバランサー (**lb1**) の設定を表示して確認します。

例

```
$ openstack loadbalancer show lb1
```

出力例

```
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| admin_state_up      | True                                 |
| created_at          | 2022-01-15T11:11:09                  |
| description         |                                      |
| flavor              |                                      |
| id                  | 788fe121-3dec-4e1b-8360-4020642238b0 |
| listeners           | 09f28053-fde8-4c78-88b9-0f191d84120e |
| name                | lb1                                  |
| operating_status    | ONLINE                               |
| pools               | 627842b3-eed8-4f5f-9f4a-01a738e64d6a |
| project_id          | dda678ca5b1241e7ad7bf7eb211a2fd7     |
| provider            | amphora                              |
| provisioning_status | ACTIVE                               |
| updated_at          | 2022-01-15T11:12:42                  |
| vip_address         | 198.51.100.11                        |
| vip_network_id      | 9bca13be-f18d-49a5-a83d-9d487827fd16 |
| vip_port_id         | 69a85edd-5b1c-458f-96f2-b4552b15b8e6 |
| vip_qos_policy_id   | None                                 |
| vip_subnet_id       | 5bd7334b-49b3-4849-b3a2-b0b83852dba1 |
+---------------------+--------------------------------------+
```

2. ヘルスモニターが存在し正常に機能する場合は、各メンバーのステータスを確認することができます。
   動作中のメンバー (**b85c807e-4d7c-4cbd-b725-5e8afddf80d2**) の **operating_status** の値は **ONLINE** です。

   例

```
$ openstack loadbalancer member show pool1 b85c807e-4d7c-4cbd-b725-5e8afddf80d2
```

   出力例

```
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| address             | 192.0.2.10                           |
| admin_state_up      | True                                 |
| created_at          | 2022-01-15T11:11:09                  |
| id                  | b85c807e-4d7c-4cbd-b725-5e8afddf80d2 |
| name                |                                      |
| operating_status    | ONLINE                               |
| project_id          | dda678ca5b1241e7ad7bf7eb211a2fd7     |
| protocol_port       | 80                                   |
| provisioning_status | ACTIVE                               |
| subnet_id           | 5bd7334b-49b3-4849-b3a2-b0b83852dba1 |
| updated_at          | 2022-01-15T11:12:42                  |
| weight              | 1                                    |
| monitor_port        | None                                 |
| monitor_address     | None                                 |
| backup              | False                                |
+---------------------+--------------------------------------+
```

関連情報

- 『Manage Secrets with OpenStack Key Manager』
- Command Line Interface ReferenceのloadbalancerX

9.6. 同じ IP およびバックエンドでの HTTP および TLS 終端 HTTPS 負荷分散の作成

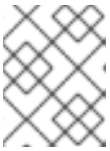
クライアントがセキュアなプロトコルまたはセキュアではない HTTP プロトコルで接続されているかどうかにかかわらず、まったく同じコンテンツで Web クライアントに応答する場合に、同じロードバランサーおよび同じ IP アドレスにセキュアではないリスナーと TLS 終端 HTTPS リスナーを設定できます。バックエンドメンバーを利用できる状態に保つためのヘルスモニターも作成するのがベストプラクティスです。

前提条件

- TCP ポート 80 でセキュアではない HTTP アプリケーションをホストするバックエンドサーバーが含まれるプライベートサブネット
- バックエンドサーバーは、URL パス / でヘルスチェックを使用して構成されます。
- インターネットから到達できる共有外部 (パブリック) サブネット。
- TLS 公開鍵暗号化は、次の特性で構成されています。
  - ロードバランサーの仮想 IP アドレス (例: www.example.com) に割り当てられた DNS 名用に、TLS 証明書、鍵、およびオプションの中間証明書チェーンが外部認証局 (CA) から取得される。
  - 証明書、鍵、および中間証明書チェーンが、現在のディレクトリー内の個別ファイルに保存される。
  - 鍵および証明書は PEM 形式でエンコードされる。
  - 鍵はパスフレーズで暗号化されない。
  - 中間証明書チェーンには PEM 形式でエンコードされた複数の証明書が含まれ、チェーンを形成する。
- Key Manager サービス (barbican) を使用するように負荷分散サービス(octavia)を構成しました。詳しくは、**『Manage Secrets with OpenStack Key Manager』**を参照してください。
- セキュアではない HTTP リスナーが、HTTPS TLS 終端ロードバランサーと同じプールで設定されている。

手順

1. 鍵 (**server.key**)、証明書 (**server.crt**)、および中間証明書チェーン (**ca-chain.crt**) を 1 つの PKCS12 ファイル (**server.p12**) に組み合わせます。



注記

丸かっこ内の値は、この手順のコマンド例で使用されるサンプルの値です。これらのサンプル値を、実際のサイトに適した値に置き換えてください。

```
$ openssl pkcs12 -export -inkey server.key -in server.crt -certfile ca-chain.crt -passout pass:
-out server.p12
```

2. 認証情報ファイルに source を実行します。

例

```
$ source ~/overcloudrc
```

3. Key Manager サービスを使用して、PKCS12 ファイルのシークレットリソース (**tls_secret1**) を作成します。

```
$ openstack secret store --name='tls_secret1' -t 'application/octet-stream' -e 'base64' --
payload="$(base64 < server.p12)"
```

4. パブリックサブネット (**public_subnet**) にロードバランサー (**lb1**) を作成します。

```
$ openstack loadbalancer create --name lb1 --vip-subnet-id public_subnet
```

5. ロードバランサーの状態を監視します。

例

```
$ openstack loadbalancer show lb1
```

6. 次の手順に進む前に、**provisioning_status** が **ACTIVE** であることを確認してください。

7. TERMINATED_HTTPS リスナー (**listener1**) を作成し、リスナーのデフォルト TLS コンテナーとしてシークレットリソースを参照します。

```
$ openstack loadbalancer listener create --protocol-port 443 --protocol
TERMINATED_HTTPS --name listener1 --default-tls-container=$(openstack secret list | awk
'/ tls_secret1 / {print $2}') lb1
```

8. プール (**pool1**) を作成し、リスナーのデフォルトプールに設定します。

```
$ openstack loadbalancer pool create --name pool1 --lb-algorithm ROUND_ROBIN --listener
listener1 --protocol HTTP
```

9. バックエンドサーバーに接続するプール (**pool1**) にヘルスモニターを作成し、パス (/) をテストします。

例

```
$ openstack loadbalancer healthmonitor create --delay 5 --max-retries 4 --timeout 10 --type
HTTP --url-path / pool1
```

10. プライベートサブネット (**private_subnet**) 上のセキュアではない HTTP バックエンドサーバー (**192.0.2.10** および **192.0.2.11**) をプールに追加します。

```
$ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.10 --
protocol-port 80 pool1
```

```
$ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.11 --
protocol-port 80 pool1
```

11. セキュアではない HTTP リスナー (**listener2**) を作成し、そのデフォルトのプールをセキュアなリスナーと同じプールに設定します。

```
$ openstack loadbalancer listener create --protocol-port 80 --protocol HTTP --name listener2
--default-pool pool1 lb1
```

検証

1. ロードバランサー (**lb1**) の設定を表示して確認します。

例

```
$ openstack loadbalancer show lb1
```

出力例

```
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| admin_state_up      | True                                 |
| created_at          | 2022-01-15T11:11:09                  |
| description         |                                      |
| flavor              |                                      |
| id                  | 788fe121-3dec-4e1b-8360-4020642238b0 |
| listeners           | 09f28053-fde8-4c78-88b9-0f191d84120e |
| name                | lb1                                  |
| operating_status    | ONLINE                               |
| pools               | 627842b3-eed8-4f5f-9f4a-01a738e64d6a |
| project_id          | dda678ca5b1241e7ad7bf7eb211a2fd7     |
| provider            | amphora                              |
| provisioning_status | ACTIVE                               |
| updated_at          | 2022-01-15T11:12:42                  |
| vip_address         | 198.51.100.11                        |
| vip_network_id      | 9bca13be-f18d-49a5-a83d-9d487827fd16 |
| vip_port_id         | 69a85edd-5b1c-458f-96f2-b4552b15b8e6 |
| vip_qos_policy_id   | None                                 |
| vip_subnet_id       | 5bd7334b-49b3-4849-b3a2-b0b83852dba1 |
+---------------------+--------------------------------------+
```

2. ヘルスモニターが存在し正常に機能する場合は、各メンバーのステータスを確認することができます。
動作中のメンバー (**b85c807e-4d7c-4cbd-b725-5e8afddf80d2**) の **operating_status** の値は **ONLINE** です。

例

```
$ openstack loadbalancer member show pool1 b85c807e-4d7c-4cbd-b725-5e8afddf80d2
```

出力例

```
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| address             | 192.0.2.10                           |
| admin_state_up      | True                                 |
| created_at          | 2022-01-15T11:11:09                  |
| id                  | b85c807e-4d7c-4cbd-b725-5e8afddf80d2 |
| name                |                                      |
| operating_status    | ONLINE                               |
| project_id          | dda678ca5b1241e7ad7bf7eb211a2fd7     |
| protocol_port       | 80                                   |
| provisioning_status | ACTIVE                               |
| subnet_id           | 5bd7334b-49b3-4849-b3a2-b0b83852dba1 |
| updated_at          | 2022-01-15T11:12:42                  |
| weight              | 1                                    |
| monitor_port        | None                                 |
| monitor_address     | None                                 |
| backup              | False                                |
+---------------------+--------------------------------------+
```

関連情報

- 『Manage Secrets with OpenStack Key Manager』
- **Command Line Interface Reference**の loadbalancer

第10章 他の種別のロードバランサーの作成

Load-balancing サービス (octavia) を使用して、管理する HTTP 以外のネットワークトラフィックの種別に一致するロードバランサーの種別を作成します。

10.1. TCP ロードバランサーの作成

HTTP 以外、TCP ベースのサービスおよびアプリケーションのネットワークトラフィックを管理する必要がある場合は、ロードバランサーを作成できます。バックエンドメンバーを利用できる状態に保つためのヘルスモニターも作成するのがベストプラクティスです。

前提条件

- 特定の TCP ポートでカスタムアプリケーションをホストするバックエンドサーバーが含まれるプライベートサブネット
- バックエンドサーバーは、URL パス / でヘルスチェックを使用して構成されます。
- インターネットから到達できる共有外部 (パブリック) サブネット。

手順

1. 認証情報ファイルに source を実行します。

   例

   ```
   $ source ~/overcloudrc
   ```

2. パブリックサブネット (**public_subnet**) にロードバランサー (**lb1**) を作成します。

   注記

   丸かっこ内の値は、この手順のコマンド例で使用されるサンプルの値です。これらのサンプル値を、実際のサイトに適した値に置き換えてください。

   例

   ```
   $ openstack loadbalancer create --name lb1 --vip-subnet-id public_subnet
   ```

3. ロードバランサーの状態を確認します。

   例

```
$ openstack loadbalancer show lb1
```

4. 次の手順に進む前に、**provisioning_status** が **ACTIVE** であることを確認してください。
5. カスタムアプリケーションが設定された指定のポート (**23456**) で **TCP** リスナー (**listener1**) を作成します。

   例

```
$ openstack loadbalancer listener create --name listener1 --protocol TCP --protocol-port 23456 lb1
```

6. プール (**pool1**) を作成し、リスナーのデフォルトプールに設定します。

   例

```
$ openstack loadbalancer pool create --name pool1 --lb-algorithm ROUND_ROBIN --listener listener1 --protocol TCP
```

7. バックエンドサーバーに接続するプール (**pool1**) にヘルスモニターを作成し、TCP サービスポートを確認します。

   例

```
$ openstack loadbalancer healthmonitor create --delay 5 --max-retries 4 --timeout 10 --type TCP pool1
```

8. プライベートサブネット (**private_subnet**) 上のバックエンドサーバー (**192.0.2.10** および **192.0.2.11**) をプールに追加します。

   例

```
$ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.10 --protocol-port 80 pool1
$ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.11 --protocol-port 80 pool1
```

検証

1. ロードバランサー (**lb1**) の設定を表示して確認します。

   例

```
$ openstack loadbalancer show lb1
```

   出力例

```
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| admin_state_up      | True                                 |
| created_at          | 2022-01-15T11:11:09                  |
```

```
| description         |                                      |
| flavor              |                                      |
| id                  | 788fe121-3dec-4e1b-8360-4020642238b0 |
| listeners           | 09f28053-fde8-4c78-88b9-0f191d84120e |
| name                | lb1                                  |
| operating_status    | ONLINE                               |
| pools               | 627842b3-eed8-4f5f-9f4a-01a738e64d6a |
| project_id          | dda678ca5b1241e7ad7bf7eb211a2fd7     |
| provider            | amphora                              |
| provisioning_status | ACTIVE                               |
| updated_at          | 2022-01-15T11:12:42                  |
| vip_address         | 198.51.100.11                        |
| vip_network_id      | 9bca13be-f18d-49a5-a83d-9d487827fd16 |
| vip_port_id         | 69a85edd-5b1c-458f-96f2-b4552b15b8e6 |
| vip_qos_policy_id   | None                                 |
| vip_subnet_id       | 5bd7334b-49b3-4849-b3a2-b0b83852dba1 |
+---------------------+--------------------------------------+
```

2. ヘルスモニターが存在し正常に機能する場合は、各メンバーのステータスを確認することができます。次のコマンドを使用して、メンバー ID を取得します。

例

```
$ openstack loadbalancer member list pool1
```

動作中のメンバー (**b85c807e-4d7c-4cbd-b725-5e8afddf80d2**) の **operating_status** の値は **ONLINE** です。

例

```
$ openstack loadbalancer member show pool1 b85c807e-4d7c-4cbd-b725-5e8afddf80d2
```

出力例

```
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| address             | 192.0.2.10                           |
| admin_state_up      | True                                 |
| created_at          | 2022-01-15T11:11:09                  |
| id                  | b85c807e-4d7c-4cbd-b725-5e8afddf80d2 |
| name                |                                      |
| operating_status    | ONLINE                               |
| project_id          | dda678ca5b1241e7ad7bf7eb211a2fd7     |
| protocol_port       | 80                                   |
| provisioning_status | ACTIVE                               |
| subnet_id           | 5bd7334b-49b3-4849-b3a2-b0b83852dba1 |
| updated_at          | 2022-01-15T11:12:42                  |
| weight              | 1                                    |
| monitor_port        | None                                 |
| monitor_address     | None                                 |
| backup              | False                                |
+---------------------+--------------------------------------+
```

関連情報

- Command Line Interface ReferenceのloadbalancerCommand Line Interface Reference

10.2. ヘルスモニターを使用した UDP ロードバランサーの作成

UDP ポート上のネットワークトラフィックを管理する必要がある場合が、ロードバランサーを作成することができます。バックエンドメンバーを利用できる状態に保つためのヘルスモニターも作成するのがベストプラクティスです。

前提条件

- UDP ポートを使用するように設定された 1 つまたは複数のアプリケーションをホストするバックエンドサーバーが含まれるプライベートサブネット
- インターネットから到達できる共有外部 (パブリック) サブネット。
- バックエンドサーバーは、UDP ヘルスチェックで構成されます。
- ICMP Destination Unreachable メッセージ (ICMP タイプ 3) をブロックするセキュリティールールはありません。

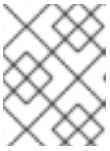
手順

1. 認証情報ファイルに source を実行します。

   例

   ```
   $ source ~/overcloudrc
   ```

2. プライベートサブネット (**private_subnet**) にロードバランサー (**lb1**) を作成します。

   注記

   丸かっこ内の値は、この手順のコマンド例で使用されるサンプルの値です。これらのサンプル値を、実際のサイトに適した値に置き換えてください。

   例

   ```
   $ openstack loadbalancer create --name lb1 --vip-subnet-id private_subnet
   ```

3. ロードバランサーの状態を確認します。

   例

   ```
   $ openstack loadbalancer show lb1
   ```

4. 次の手順に進む前に、**provisioning_status** が **ACTIVE** であることを確認してください。
5. ポート (**1234**) にリスナー (**listener1**) を作成します。

   例

```
$ openstack loadbalancer listener create --name listener1 --protocol UDP --protocol-port
1234 lb1
```

6. リスナーのデフォルトプール (**pool1**) を作成します。

例

```
$ openstack loadbalancer pool create --name pool1 --lb-algorithm ROUND_ROBIN --listener
listener1 --protocol UDP
```

7. UDP (**UDP-CONNECT**) を使用するバックエンドサーバーに接続するプール (**pool1**) にヘルスモニターを作成します。

例

```
$ openstack loadbalancer healthmonitor create --delay 5 --max-retries 2 --timeout 3 --type
UDP-CONNECT pool1
```

8. プライベートサブネット (**private_subnet**) のロードバランサーメンバー (**192.0.2.10** および **192.0.2.11**) をデフォルトのプールに追加します。

例

```
$ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.10 --
protocol-port 1234 pool1
$ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.11 --
protocol-port 1234 pool1
```

検証

1. ロードバランサー (**lb1**) の設定を表示して確認します。

例

```
$ openstack loadbalancer show lb1
```

出力例

```
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| admin_state_up      | True                                 |
| created_at          | 2022-01-15T11:11:09                  |
| description         |                                      |
| flavor              |                                      |
| id                  | 788fe121-3dec-4e1b-8360-4020642238b0 |
| listeners           | 09f28053-fde8-4c78-88b9-0f191d84120e |
| name                | lb1                                  |
| operating_status    | ONLINE                               |
| pools               | 627842b3-eed8-4f5f-9f4a-01a738e64d6a |
| project_id          | dda678ca5b1241e7ad7bf7eb211a2fd7     |
| provider            | amphora                              |
| provisioning_status | ACTIVE                               |
```

```
| updated_at          | 2022-01-15T11:12:42                  |
| vip_address         | 198.51.100.11                        |
| vip_network_id      | 9bca13be-f18d-49a5-a83d-9d487827fd16 |
| vip_port_id         | 69a85edd-5b1c-458f-96f2-b4552b15b8e6 |
| vip_qos_policy_id   | None                                 |
| vip_subnet_id       | 5bd7334b-49b3-4849-b3a2-b0b83852dba1 |
+---------------------+--------------------------------------+
```

2. ヘルスモニターが存在し正常に機能する場合は、各メンバーのステータスを確認することができます。次のコマンドを使用して、メンバー ID を取得します。

例

```
$ openstack loadbalancer member list pool1
```

動作中のメンバー (**b85c807e-4d7c-4cbd-b725-5e8afddf80d2**) の **operating_status** の値は **ONLINE** です。

例

```
$ openstack loadbalancer member show pool1 b85c807e-4d7c-4cbd-b725-5e8afddf80d2
```

出力例

```
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| address             | 192.0.2.10                           |
| admin_state_up      | True                                 |
| created_at          | 2022-01-15T11:11:09                  |
| id                  | b85c807e-4d7c-4cbd-b725-5e8afddf80d2 |
| name                |                                      |
| operating_status    | ONLINE                               |
| project_id          | dda678ca5b1241e7ad7bf7eb211a2fd7     |
| protocol_port       | 1234                                 |
| provisioning_status | ACTIVE                               |
| subnet_id           | 5bd7334b-49b3-4849-b3a2-b0b83852dba1 |
| updated_at          | 2022-01-15T11:12:42                  |
| weight              | 1                                    |
| monitor_port        | None                                 |
| monitor_address     | None                                 |
| backup              | False                                |
+---------------------+--------------------------------------+
```

関連情報

- Command Line Interface Referenceのloadbalancer

10.3. QOS ルールが適用されるロードバランサーの作成

Red Hat OpenStack Platform (RHOSP) ネットワーキングサービス (neutron) の Quality of Service (QoS) ポリシーを、ロードバランサーを使用する仮想 IP アドレス (VIP) に適用できます。これにより、QoS ポリシーを使用して、ロードバランサーが管理することのできる送受信ネットワークトラ

フィックを制限することができます。バックエンドメンバーを利用できる状態に保つためのヘルスモニターも作成するのがベストプラクティスです。

前提条件

- TCP ポート 80 で HTTP アプリケーションが設定されたバックエンドサーバーが含まれるプライベートサブネット
- バックエンドサーバーは、URL パス / でヘルスチェックを使用して構成されます。
- インターネットから到達できる共有外部 (パブリック) サブネット。
- RHOSP ネットワークサービス用に作成された帯域幅制限ルールを含む QoS ポリシー。

手順

1. 認証情報ファイルに source を実行します。

   例

   ```
   $ source ~/overcloudrc
   ```

2. 最大 1024 kbps および最大バーストレートが 1024 kb のネットワーク帯域幅 QoS ポリシー (**qos_policy_bandwidth**) を作成します。

   注記

   丸かっこ内の値は、この手順のコマンド例で使用されるサンプルの値です。これらのサンプル値を、実際のサイトに適した値に置き換えてください。

   例

   ```
   $ openstack network qos policy create qos_policy_bandwidth
   $ openstack network qos rule create --type bandwidth-limit --max-kbps 1024 --max-burst-kbits 1024 qos-policy-bandwidth
   ```

3. QoS ポリシー (**qos-policy-bandwidth**) を使用してパブリックサブネット (**public_subnet**) にロードバランサー (**lb1**) を作成します。

   例

   ```
   $ openstack loadbalancer create --name lb1 --vip-subnet-id public_subnet --vip-qos-policy-id qos-policy-bandwidth
   ```

4. ロードバランサーの状態を確認します。

   例

   ```
   $ openstack loadbalancer show lb1
   ```

5. 次の手順に進む前に、**provisioning_status** が **ACTIVE** であることを確認してください。

6. ポート (**80**) にリスナー (**listener1**) を作成します。

例

```
$ openstack loadbalancer listener create --name listener1 --protocol HTTP --protocol-port 80
lb1
```

7. リスナーのデフォルトプール (**pool1**) を作成します。

例

```
$ openstack loadbalancer pool create --name pool1 --lb-algorithm ROUND_ROBIN --listener
listener1 --protocol HTTP
```

8. バックエンドサーバーに接続するプールにヘルスモニターを作成し、パス (/) をテストします。

例

```
$ openstack loadbalancer healthmonitor create --delay 5 --max-retries 4 --timeout 10 --type
HTTP --url-path / pool1
```

9. プライベートサブネット (**private_subnet**) のロードバランサーメンバー (**192.0.2.10** および **192.0.2.11**) をデフォルトのプールに追加します。

例

```
$ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.10 --
protocol-port 80 pool1

$ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.11 --
protocol-port 80 pool1
```

検証

1. リスナー (**listener1**) の設定を表示して確認します。

例

```
$ openstack loadbalancer list
```

出力例

```
+--------------------+--------------------------------------+
| Field              | Value                                |
+--------------------+--------------------------------------+
| admin_state_up     | True                                 |
| created_at         | 2022-01-15T11:11:09                  |
| description        |                                      |
| flavor             |                                      |
| id                 | 788fe121-3dec-4e1b-8360-4020642238b0 |
| listeners          | 09f28053-fde8-4c78-88b9-0f191d84120e |
| name               | lb1                                  |
| operating_status   | ONLINE                               |
| pools              | 627842b3-eed8-4f5f-9f4a-01a738e64d6a |
| project_id         | dda678ca5b1241e7ad7bf7eb211a2fd7     |
```

```
| provider            | amphora                              |
| provisioning_status | ACTIVE                               |
| updated_at          | 2022-01-15T11:12:42                  |
| vip_address         | 198.51.100.11                        |
| vip_network_id      | 9bca13be-f18d-49a5-a83d-9d487827fd16 |
| vip_port_id         | 69a85edd-5b1c-458f-96f2-b4552b15b8e6 |
| vip_qos_policy_id   | cdfc3398-997b-46eb-9db1-ebbd88f7de05 |
| vip_subnet_id       | 5bd7334b-49b3-4849-b3a2-b0b83852dba1 |
+---------------------+--------------------------------------+
```

この例では、パラメーター **vip_qos_policy_id** にポリシー ID が含まれます。

関連情報

- Command Line Interface ReferenceのloadbalancerCommand Line Interface Reference

10.4. アクセス制御リストを使用したロードバランサーの作成

アクセス制御リスト (ACL) を作成し、リスナーへの受信トラフィックを、許可されたソース IP アドレスのセットに制限することができます。それ意外の受信トラフィックは、すべて拒否されます。バックエンドメンバーを利用できる状態に保つためのヘルスモニターも作成するのがベストプラクティスです。

前提条件

- TCP ポート 80 でカスタムアプリケーションが設定されたバックエンドサーバーが含まれるプライベートサブネット
- バックエンドサーバーは、URL パス / でヘルスチェックを使用して構成されます。
- インターネットから到達できる共有外部 (パブリック) サブネット。

手順

1. 認証情報ファイルに source を実行します。

   例

   ```
   $ source ~/overcloudrc
   ```

2. パブリックサブネット (**public_subnet**) にロードバランサー (**lb1**) を作成します。

   注記

   丸かっこ内の値は、この手順のコマンド例で使用されるサンプルの値です。これらのサンプル値を、実際のサイトに適した値に置き換えてください。

   例

   ```
   $ openstack loadbalancer create --name lb1 --vip-subnet-id public_subnet
   ```

3. ロードバランサーの状態を確認します。

例

```
$ openstack loadbalancer show lb1
```

4. 次の手順に進む前に、**provisioning_status** が **ACTIVE** であることを確認してください。
5. 許可される CIDR (**192.0.2.0/24** および **198.51.100.0/24**) でリスナー (**listener1**) を作成します。

例

```
$ openstack loadbalancer listener create --name listener1 --protocol TCP --protocol-port 80 --allowed-cidr 192.0.2.0/24 --allowed-cidr 198.51.100.0/24 lb1
```

6. リスナーのデフォルトプール (**pool1**) を作成します。

例

```
$ openstack loadbalancer pool create --name pool1 --lb-algorithm ROUND_ROBIN --listener listener1 --protocol TCP
```

7. バックエンドサーバーに接続するプールにヘルスモニターを作成し、パス (/) をテストします。

例

```
$ openstack loadbalancer healthmonitor create --delay 5 --max-retries 4 --timeout 10 --type HTTP --url-path / pool1
```

8. プライベートサブネット (**private_subnet**) のロードバランサーメンバー (**192.0.2.10** および **192.0.2.11**) をデフォルトのプールに追加します。

例

```
$ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.10 --protocol-port 80 pool1

$ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.11 --protocol-port 80 pool1
```

検証

1. リスナー (**listener1**) の設定を表示して確認します。

例

```
$ openstack loadbalancer listener show listener1
```

出力例

```
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
```

```
| connection_limit           | -1                                   |
| created_at                 | 2022-01-15T11:11:09                  |
| default_pool_id            | None                                 |
| default_tls_container_ref  | None                                 |
| description                |                                      |
| id                         | d26ba156-03c3-4051-86e8-f8997a202d8e |
| insert_headers             | None                                 |
| l7policies                 |                                      |
| loadbalancers              | 2281487a-54b9-4c2a-8d95-37262ec679d6 |
| name                       | listener1                            |
| operating_status           | ONLINE                               |
| project_id                 | 308ca9f600064f2a8b3be2d57227ef8f     |
| protocol                   | TCP                                  |
| protocol_port              | 80                                   |
| provisioning_status        | ACTIVE                               |
| sni_container_refs         | []                                   |
| timeout_client_data        | 50000                                |
| timeout_member_connect     | 5000                                 |
| timeout_member_data        | 50000                                |
| timeout_tcp_inspect        | 0                                    |
| updated_at                 | 2022-01-15T11:12:42                  |
| client_ca_tls_container_ref | None                                |
| client_authentication      | NONE                                 |
| client_crl_container_ref   | None                                 |
| allowed_cidrs              | 192.0.2.0/24                         |
|                            | 198.51.100.0/24                      |
+----------------------------+--------------------------------------+
```

この例では、パラメーター **allowed_cidrs** は、192.0.2.0/24 および 198.51.100.0/24 からのトラフィックだけを許可するように設定します。

2. ロードバランサーが保護されていることを確認するには、**allowed_cidrs** の一覧に記載されていない CIDR のクライアントからリスナーへの要求を確認します。要求は成功しないはずです。

出力例

```
curl: (7) Failed to connect to 203.0.113.226 port 80: Connection timed out
curl: (7) Failed to connect to 203.0.113.226 port 80: Connection timed out
curl: (7) Failed to connect to 203.0.113.226 port 80: Connection timed out
curl: (7) Failed to connect to 203.0.113.226 port 80: Connection timed out
```

関連情報

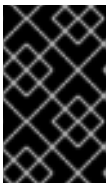
- Command Line Interface ReferenceのloadbalancerCommand Line Interface Referenceのloadbalancer

10.5. OVN ロードバランサーの作成

Red Hat OpenStack Platform (RHOSP) クライアントを使用して、RHOSP デプロイメントのネットワークトラフィックを管理するロードバランサーを作成できます。RHOSP Load-Balancing サービスは、neutron Modular Layer 2 プラグインと Open Virtual Network メカニズムドライバーの組み合わせ (ML2/OVN) をサポートします。

前提条件

- ML2/OVN プロバイダードライバーがデプロイされている必要があります。



重要

OVN プロバイダーは、レイヤー 4 TCP および UDP ネットワークトラフィック、ならびに **SOURCE_IP_PORT** ロードバランサーアルゴリズムだけをサポートします。OVN プロバイダーはヘルスモニタリングをサポートしません。

- 特定の TCP ポートでカスタムアプリケーションをホストするバックエンドサーバーが含まれるプライベートサブネット
- インターネットから到達できる共有外部 (パブリック) サブネット。

手順

1. 認証情報ファイルに source を実行します。

   例

   ```
   $ source ~/overcloudrc
   ```

2. **--provider ovn** 引数を使用して、プライベートサブネット (**private_subnet**) にロードバランサー (**lb1**) を作成します。

注記

丸かっこ内の値は、この手順のコマンド例で使用されるサンプルの値です。これらのサンプル値を、実際のサイトに適した値に置き換えてください。

   例

   ```
   $ openstack loadbalancer create --name lb1 --provider ovn --vip-subnet-id private_subnet
   ```

3. ロードバランサーの状態を確認します。

   ```
   $ openstack loadbalancer show lb1
   ```

4. 次の手順に進む前に、**provisioning_status** が **ACTIVE** であることを確認してください。
5. カスタムアプリケーションが設定された指定のポート (**80**) でプロトコル (**tcp**) を使用するリスナー (**listener1**) を作成します。

注記

OVN プロバイダーは、レイヤー 4 TCP および UDP ネットワークトラフィックだけをサポートします。

   例

   ```
   $ openstack loadbalancer listener create --name listener1 --protocol tcp --protocol-port 80
   lb1
   ```

6. リスナーのデフォルトプール (**pool1**) を作成します。

注記

OVN 用にサポートされる唯一の負荷分散アルゴリズムは **SOURCE_IP_PORT** です。

例

```
$ openstack loadbalancer pool create --name pool1 --lb-algorithm SOURCE_IP_PORT --
listener listener1 --protocol tcp
```

重要

OVN は負荷分散のヘルスモニター機能をサポートしません。

7. プライベートサブネット (**private_subnet**) 上のバックエンドサーバー (**192.0.2.10** および **192.0.2.11**) をプールに追加します。

例

```
$ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.10 --
protocol-port 80 pool1
$ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.11 --
protocol-port 80 pool1
```

検証

1. ロードバランサー (**lb1**) の設定を表示して確認します。

例

```
$ openstack loadbalancer show lb1
```

出力例

```
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| admin_state_up      | True                                 |
| created_at          | 2022-01-15T11:11:09                  |
| description         |                                      |
| flavor              |                                      |
| id                  | 788fe121-3dec-4e1b-8360-4020642238b0 |
| listeners           | 09f28053-fde8-4c78-88b9-0f191d84120e |
| name                | lb1                                  |
| operating_status    | ONLINE                               |
| pools               | 627842b3-eed8-4f5f-9f4a-01a738e64d6a |
| project_id          | dda678ca5b1241e7ad7bf7eb211a2fd7     |
| provider            | ovn                                  |
| provisioning_status | ACTIVE                               |
| updated_at          | 2022-01-15T11:12:42                  |
```

```
| vip_address         | 198.51.100.11                        |
| vip_network_id      | 9bca13be-f18d-49a5-a83d-9d487827fd16 |
| vip_port_id         | 69a85edd-5b1c-458f-96f2-b4552b15b8e6 |
| vip_qos_policy_id   | None                                 |
| vip_subnet_id       | 5bd7334b-49b3-4849-b3a2-b0b83852dba1 |
+---------------------+--------------------------------------+
```

2. リスナーの情報を表示するには、**openstack loadbalancer listener show** コマンドを実行します。

例

```
$ openstack loadbalancer listener show listener1
```

出力例

```
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| connection_limit          | -1                                   |
| created_at                | 2022-01-15T11:13:52                  |
| default_pool_id           | a5034e7a-7ddf-416f-9c42-866863def1f2 |
| default_tls_container_ref | None                                 |
| description               |                                      |
| id                        | a101caba-5573-4153-ade9-4ea63153b164 |
| insert_headers            | None                                 |
| l7policies                |                                      |
| loadbalancers             | 653b8d79-e8a4-4ddc-81b4-e3e6b42a2fe3 |
| name                      | listener1                            |
| operating_status          | ONLINE                               |
| project_id                | 7982a874623944d2a1b54fac9fe46f0b     |
| protocol                  | TCP                                  |
| protocol_port             | 64015                                |
| provisioning_status       | ACTIVE                               |
| sni_container_refs        | []                                   |
| timeout_client_data       | 50000                                |
| timeout_member_connect    | 5000                                 |
| timeout_member_data       | 50000                                |
| timeout_tcp_inspect       | 0                                    |
| updated_at                | 2022-01-15T11:15:17                  |
| client_ca_tls_container_ref | None                               |
| client_authentication     | NONE                                 |
| client_crl_container_ref  | None                                 |
| allowed_cidrs             | None                                 |
+---------------------------+--------------------------------------+
```

3. プール (**pool1**) とロードバランサーのメンバーを表示するには、**openstack loadbalancer pool show** コマンドを実行します。

例

```
$ openstack loadbalancer pool show pool1
```

出力例

```
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| admin_state_up      | True                                 |
| created_at          | 2022-01-15T11:17:34                  |
| description         |                                      |
| healthmonitor_id    |                                      |
| id                  | a5034e7a-7ddf-416f-9c42-866863def1f2 |
| lb_algorithm        | SOURCE_IP_PORT                       |
| listeners           | a101caba-5573-4153-ade9-4ea63153b164 |
| loadbalancers       | 653b8d79-e8a4-4ddc-81b4-e3e6b42a2fe3 |
| members             | 90d69170-2f73-4bfd-ad31-896191088f59 |
| name                | pool1                                |
| operating_status    | ONLINE                               |
| project_id          | 7982a874623944d2a1b54fac9fe46f0b     |
| protocol            | TCP                                  |
| provisioning_status | ACTIVE                               |
| session_persistence | None                                 |
| updated_at          | 2022-01-15T11:18:59                  |
| tls_container_ref   | None                                 |
| ca_tls_container_ref | None                                |
| crl_container_ref   | None                                 |
| tls_enabled         | False                                |
+---------------------+--------------------------------------+
```

関連情報

- Command Line Interface ReferenceのloadbalancerAPI

第11章 レイヤー 7 負荷分散の実装

レイヤー 7 ポリシーと共に Red Hat OpenStack Platform Load-balancing サービス (octavia) を使用して、ビジネスニーズに応じた複数の条件により、HTTP リクエストを特定のアプリケーションサーバープールにリダイレクトすることができます。

- 「レイヤー 7 の負荷分散について」
- 「Load-balancing サービスでのレイヤー 7 負荷分散」
- 「レイヤー 7 負荷分散ルール」
- 「レイヤー 7 負荷分散ルールの種別」
- 「レイヤー 7 負荷分散ルール比較の種別」
- 「レイヤー 7 負荷分散ルールの結果の反転」
- 「レイヤー 7 負荷分散ポリシー」
- 「レイヤー 7 負荷分散ポリシーのロジック」
- 「レイヤー 7 負荷分散ポリシーのアクション」
- 「レイヤー 7 負荷分散ポリシーの位置」
- 「セキュアではない HTTP リクエストのセキュアな HTTP へのリダイレクト」
- 「開始パスに基づくリクエストのプールへのリダイレクト」
- 「特定プールへのサブドメインリクエストの送信」
- 「ホスト名末尾に基づくリクエストの特定プールへの送信」
- 「ブラウザークッキーが存在しないことに基づくリクエストの特定プールへの送信」
- 「ブラウザークッキーが存在しないことまたは無効なクッキー値に基づくリクエストの特定プールへの送信」
- 「名前がホスト名およびパスと一致するリクエストのプールへの送信」
- 「cookie を使用して既存の本番サイトで AB テストを構成」

11.1. レイヤー 7 の負荷分散について

レイヤー 7 (L7) の負荷分散は、Open Systems Interconnection (OSI) モデルからその名前を取ります。ロードバランサーは、レイヤー 7 (アプリケーション) データに基づいてリクエストをバックエンドアプリケーションサーバープールに分散します。**リクエストスイッチング**、**アプリケーションロードバランシング**、および **コンテンツベースのルーティング**、**スイッチング**、または **バランシング** は、すべて L7 負荷分散を意味する用語です。Red Hat OpenStack Platform Load-balancing サービス (octavia) は、L7 負荷分散の堅牢なサポートを提供します。

注記

UDP ロードバランサーで L7 ポリシーおよびルールを作成することはできません。

L7 ロードバランサーは、数多くのバックエンドプールの代わりにリクエストを受け入れ、アプリケーションデータを使用してそれぞれのリクエストを処理するプールを決定するポリシーに基づいてこれらのリクエストを分散するリスナーで構成されます。これにより、アプリケーションインフラストラクチャーを、特定の種別のコンテンツに対応できるように具体的に調整および最適化することができます。たとえば、バックエンドサーバーの 1 つのグループ (プール) をイメージのみに対応するように調整し、別のグループを PHP や ASP などのサーバー側のスクリプト言語の実行用に、さらに別のグループを HTML、CSS、JavaScript などの静的コンテンツ用に調整することができます。

下位レベルの負荷分散と異なり、L7 負荷分散機能では、負荷分散サービスの背後にあるすべてのプールが同じコンテンツを持つ必要はありません。L7 ロードバランサーは、アプリケーションメッセージ内の URI、ホスト、HTTP ヘッダー、およびその他のデータに基づいてリクエストを送信できます。

11.2. LOAD-BALANCING サービスでのレイヤー 7 負荷分散

レイヤー 7 (L7) の負荷分散は、適切に定義された任意の L7 アプリケーションインターフェースに対して実装できますが、Red Hat OpenStack Platform Load-balancing サービス (octavia) の L7 機能は、HTTP および TERMINATED_HTTPS プロトコルとその意味のみを参照します。

Neutron LBaaS と負荷分散サービスは、L7 負荷分散のロジックに L7 ルールとポリシーを使用します。L7 ルールは、1 つの単純な論理テストで、true または false に評価します。L7 ポリシーは、L7 ルールのコレクションであり、ポリシーに関連付けられているすべてのルールが一致した場合に実行する定義済みのアクションです。

11.3. レイヤー 7 負荷分散ルール

Red Hat OpenStack Platform Load-balancing サービス (octavia) の場合、レイヤー 7 (L7) 負荷分散ルールは、true または false を返す単一の単純な論理テストです。これは、ルールの種別、比較の種別、値、およびルール種別に応じて使用されるオプションのキーで構成されます。L7 ルールは、常に L7 ポリシーに関連付ける必要があります。

注記

UDP ロードバランサーで L7 ポリシーおよびルールを作成することはできません。

関連情報

- 「レイヤー 7 負荷分散ルールの種別」

11.4. レイヤー 7 負荷分散ルールの種別

Red Hat OpenStack Platform Load-balancing サービス (octavia) には、以下の種別のレイヤー 7 負荷分散ルールがあります。

- **HOST_NAME**: ルールは、リクエスト内の HTTP /1.1 ホスト名をルール内の value パラメーターと比較します。
- **PATH**: ルールは、HTTP URI のパス部分を、ルールの値パラメーターと比較します。
- **FILE_TYPE**: ルールは、URI の最後の部分を、ルールの値パラメーターと比較します (例: txt、jpg など)。
- **HEADER**: ルールはキーパラメーターで定義されるヘッダーを検索し、それをルールの値パラメーターと比較します。

- **COOKIE**: ルールはキーパラメーターで命名されるクッキーを検索し、それをルールの値パラメーターと比較します。
- **SSL_CONN_HAS_CERT**: クライアントが TLS クライアント認証用の証明書を提示した場合、ルールは一致します。これは、証明書が有効であることを意味するものではありません。
- **SSL_VERIFY_RESULT**: このルールは、TLS クライアント認証証明書の検証結果を照合します。ゼロ (**0**) の値は、証明書が正常に検証されたことを意味します。ゼロより大きい値は、証明書が検証に失敗したことを意味します。この値は、**openssl-verify** 結果コードに従います。
- **SSL_DN_FIELD**: ルールはキーパラメーターで定義される **Distinguished Name** を検索し、それをルールの値パラメーターと比較します。

関連情報

- 「レイヤー 7 負荷分散ルール」

11.5. レイヤー 7 負荷分散ルール比較の種別

Red Hat OpenStack Platform Load-balancing サービス (octavia) の場合、指定された種別のレイヤー 7 負荷分散ルールは常に比較を行います。負荷分散サービスは、次のタイプの比較をサポートします。すべてのルールタイプがすべての比較タイプをサポートしているわけではありません。

- **REGEX**: perl 種別の正規表現の照合
- **STARTS_WITH**: 文字列で始まる
- **ENDS_WITH**: 文字列で終わる
- **CONTAINS**: 文字列が含まれる
- **EQUAL_TO**: 文字列が等しい

関連情報

- 「レイヤー 7 負荷分散ルールの種別」

11.6. レイヤー 7 負荷分散ルールの結果の反転

一部のポリシーが必要とし、Red Hat OpenStack プラットフォームの負荷分散サービス (octavia) が使用するロジックをより完全に表現するために、レイヤー 7 の負荷分散ルールの結果を反転させることができます。指定されたルールの invert パラメーターが true の場合、その比較の結果は反転されます。

たとえば、**equal to** ルールを反転すると、実質的には **not equal to** ルールになります。**regex** ルールを反転すると、指定された正規表現が一致しない場合にだけ **true** が返されます。

関連情報

- 「レイヤー 7 負荷分散ポリシー」

11.7. レイヤー 7 負荷分散ポリシー

Red Hat OpenStack Platform Load-balancing サービス (octavia) の場合、レイヤー 7 (L7) 負荷分散ポリシーは、リスナーと関連付けられた L7 ルールの集合です。また、バックエンドプールとも関連付け

られる場合があります。ポリシーは、ポリシー内のすべてのルールが true の場合、ロードバランサーが実行するアクションです。

注記

UDP ロードバランサーで L7 ポリシーおよびルールを作成することはできません。

関連情報

- 「レイヤー 7 負荷分散ルール」

11.8. レイヤー 7 負荷分散ポリシーのロジック

Red Hat OpenStack Platform Load-balancing サービス (octavia) の場合、レイヤー 7 負荷分散ポリシーのロジックは非常にシンプルです。指定されたポリシーに関連付けられたすべてのルールは、論理的に AND で結合されます。ポリシーとマッチするためには、リクエストはすべてのポリシールールとマッチする必要があります。

ルール間で論理 OR 演算を表現する必要がある場合は、同じアクションで複数のポリシーを作成するか、より複雑な正規表現を作成します)。

関連情報

- 「レイヤー 7 負荷分散ルール」

11.9. レイヤー 7 負荷分散ポリシーのアクション

レイヤー 7 負荷分散ポリシーが指定のリクエストとマッチする場合は、そのポリシーのアクションが実行されます。L7 ポリシーが実行する可能性のあるアクションは次のとおりです。

- **REJECT**: リクエストは適切な応答コードと共に拒否され、いずれのバックエンドプールにも転送されません。
- **REDIRECT_TO_URL**: リクエストは、redirect_url パラメーターに定義された URL に HTTP リダイレクトが送信されます。
- **REDIRECT_PREFIX**: このポリシーにマッチするリクエストは、このプレフィックス URL にリダイレクトされます。
- **REDIRECT_TO_POOL**: リクエストは、L7 ポリシーに関連付けられたバックエンドプールに転送されます。

関連情報

- 「レイヤー 7 負荷分散ポリシー」

11.10. レイヤー 7 負荷分散ポリシーの位置

Red Hat OpenStack Platform (RHOSP) Load-balancing サービス (octavia) の場合には、複数のレイヤー 7 (L7) 負荷分散ポリシーがリスナーに関連付けられると、ポリシーの位置パラメーターの値が重要になります。位置パラメーターは、L7 ポリシーが評価される順序を決定する際に使用されます。ポリシーの位置は、次の方法でリスナーの動作に影響を与えます。

- 負荷分散サービス (haproxy amphora) の参照実装では、HAProxy はポリシーのアクションに関する以下の順序を強制します。
  - **REJECT** ポリシーは、他のすべてのポリシーよりも優先されます。
  - **REDIRECT_TO_URL** ポリシーは、REDIRECT_TO_POOL ポリシーよりも優先されます。
  - **REDIRECT_TO_POOL** ポリシーは、上記のすべての後で、ポリシーの位置が指定する順序でのみ評価されます。
- L7 ポリシーは、位置属性で定義されるように特定の順序で評価されます。指定のリクエストとマッチする最初のポリシーのアクションが実行されます。
- いずれのポリシーも指定のリクエストにマッチしない場合、リクエストはリスナーのデフォルトプール (存在する場合) にルーティングされます。リスナーにデフォルトのプールがない場合は、エラー 503 を返します。
- ポリシーの位置の番号は、**1** から始まります。
- 既存ポリシーの位置に一致する位置で新しいポリシーが作成されると、新しいポリシーが指定の位置に挿入されます。
- 位置を指定せずに新しいポリシーが作成されるか、または一覧にすでにあるポリシーの番号よりも大きい位置を指定すると、新しいポリシーはただ一覧に追加されます。
- ポリシーが一覧に挿入、削除、または追加されると、ポリシーの位置の値は数字を飛ばさずに **1** から並べ替えられます。たとえば、ポリシー A、B、および C の位置の値がそれぞれ **1**、**2**、および **3** の場合、一覧からポリシー B を削除すると、ポリシー C の位置は **2** になります。

関連情報

- 「レイヤー 7 負荷分散ポリシー」

11.11. セキュアではない HTTP リクエストのセキュアな HTTP へのリダイレクト

レイヤー 7 (L7) ポリシーと共に Red Hat OpenStack Platform (RHOSP) Load-balancing サービス (octavia) を使用して、セキュアではない TCP ポート上で受信した HTTP リクエストをセキュアな TCP ポートにリダイレクトすることができます。

この例では、セキュアではない TCP ポート 80 に到達するすべての HTTP リクエストは、セキュアな TCP ポート 443 にリダイレクトされます。

前提条件

- リスナー (**listener1**) およびプール (**pool1**) を持つ TLS 終端 HTTPS ロードバランサー (**lb1**)。詳細は、Creating a TLS-terminated HTTPS load balancer を参照してください。

手順

1. 認証情報ファイルに source を実行します。

   例

   ```
   $ source ~/overcloudrc
   ```

2. ロードバランサー (**lb1**) のポート (**80**) に HTTP リスナー (**http_listener**) を作成します。

注記

丸かっこ内の値は、この手順のコマンド例で使用されるサンプルの値です。これらのサンプル値を、実際のサイトに適した値に置き換えてください。

例

```
$ openstack loadbalancer listener create --name http_listener --protocol HTTP --protocol-port
80 lb1
```

3. リスナー (**http_listener**) に L7 ポリシー (**policy1**) を作成します。ポリシーには、アクション (**REDIRECT_TO_URL**) が含まれ、URL (**https://www.example.com/**) を示す必要があります。

例

```
$ openstack loadbalancer l7policy create --action REDIRECT_PREFIX --redirect-prefix
https://www.example.com/ --name policy1 http_listener
```

4. すべてのリクエストにマッチする L7 ルールを、ポリシー (**policy1**) に追加します。

例

```
$ openstack loadbalancer l7rule create --compare-type STARTS_WITH --type PATH --value
/ policy1
```

検証

1. **openstack loadbalancer l7policy list** コマンドを実行し、ポリシー **policy1** が存在することを確認します。

2. **openstack loadbalancer l7rule list <l7policy>** コマンドを実行し、**compare_type** が **STARTS_WITH** のルールが存在することを確認します。

例

```
$ openstack loadbalancer l7rule list policy1
```

関連情報

- **Command Line Interface Reference** の loadbalancer listener create
- **Command Line Interface Reference** の loadbalancer l7policy create
- **Command Line Interface Reference** の loadbalancer l7rule create

11.12. 開始パスに基づくリクエストのプールへのリダイレクト

Red Hat OpenStack Platform (RHOSP) Load-balancing サービス (octavia) を使用して、HTTP リクエストをサーバーの別のプールにリダイレクトすることができます。リクエストの URL の 1 つ以上の開始パスに一致するようにレイヤー 7 (L7) ポリシーを定義できます。

以下の例では、**/js** または **/images** で始まる URL が含まれるリクエストは、すべて静的コンテンツサーバーの別のプールにリダイレクトされます。

前提条件

- リスナー (**listener1**) およびプール (**pool1**) を持つ HTTPS ロードバランサー (**lb1**)。詳細は、Creating an HTTP load balancer with a health monitor を参照してください。

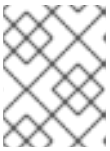
手順

1. 認証情報ファイルに source を実行します。

   例

   ```
   $ source ~/overcloudrc
   ```

2. ロードバランサー (**lb1**) に 2 番目のプール (**static_pool**) を作成します。

   注記

   丸かっこ内の値は、この手順のコマンド例で使用されるサンプルの値です。これらのサンプル値を、実際のサイトに適した値に置き換えてください。

   例

   ```
   $ openstack loadbalancer pool create --lb-algorithm ROUND_ROBIN --loadbalancer lb1 --
   name static_pool --protocol HTTP
   ```

3. プライベートサブネット (**private_subnet**) のロードバランサーメンバー (**192.0.2.10** および **192.0.2.11**) をプール (**static_pool**) に追加します。

   例

   ```
   $ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.10 --
   protocol-port 80 static_pool
   $ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.11 --
   protocol-port 80 static_pool
   ```

4. リスナー (**listener1**) に L7 ポリシー (**policy1**) を作成します。ポリシーには、アクション (**REDIRECT_TO_POOL**) を追加し、プール (**static_pool**) を示す必要があります。

   例

   ```
   $ openstack loadbalancer l7policy create --action REDIRECT_TO_POOL --redirect-pool
   static_pool --name policy1 listener1
   ```

5. リクエストパスの先頭に **/js** を探す L7 ルールを、ポリシーに追加します。

   例

   ```
   $ openstack loadbalancer l7rule create --compare-type STARTS_WITH --type PATH --value
   /js policy1
   ```

6. アクション (**REDIRECT_TO_POOL**) で L7 ポリシー (**policy2**) を作成し、プールに示したリスナー (**listener1**) を追加します。

   例

   ```
   $ openstack loadbalancer l7policy create --action REDIRECT_TO_POOL --redirect-pool
   static_pool --name policy2 listener1
   ```

7. リクエストパスの先頭に **/images** を探す L7 ルールを、ポリシーに追加します。

   例

   ```
   $ openstack loadbalancer l7rule create --compare-type STARTS_WITH --type PATH --value
   /images policy2
   ```

検証

1. **openstack loadbalancer l7policy list** コマンドを実行し、ポリシー **policy1** および **policy2** が存在することを確認します。

2. **openstack loadbalancer l7rule list <l7policy>** コマンドを実行し、それぞれのポリシーごとに **compare_type** が **STARTS_WITH** のルールが存在することを確認します。

   例

   ```
   $ openstack loadbalancer l7rule list policy1
   $ openstack loadbalancer l7rule list policy2
   ```

関連情報

- Command Line Interface Reference の loadbalancer pool create
- Command Line Interface Reference の loadbalancer member create
- Command Line Interface Reference の loadbalancer l7policy create
- Command Line Interface Reference の loadbalancer l7rule create

11.13. 特定プールへのサブドメインリクエストの送信

レイヤー 7 (L7) ポリシーと共に Red Hat OpenStack Platform (RHOSP) Load-balancing サービス (octavia) を使用して、特定の HTTP/1.1 ホスト名が含まれるリクエストをアプリケーションサーバーの異なるプールにリダイレクトすることができます。

以下の例では、HTTP/1.1 ホスト名 **www2.example.com** が含まれるリクエストは、すべてアプリケーションサーバーの別のプール **pool2** にリダイレクトされます。

前提条件

- リスナー (**listener1**) およびプール (**pool1**) を持つ HTTPS ロードバランサー (**lb1**)。詳細は、Creating an HTTP load balancer with a health monitor を参照してください。

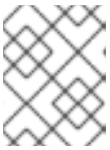
手順

1. 認証情報ファイルに source を実行します。

   例

   ```
   $ source ~/overcloudrc
   ```

2. ロードバランサー (**lb1**) に 2 つ目のプール (**pool2**) を作成します。

   注記

   丸かっこ内の値は、この手順のコマンド例で使用されるサンプルの値です。これらのサンプル値を、実際のサイトに適した値に置き換えてください。

   例

   ```
   $ openstack loadbalancer pool create --lb-algorithm ROUND_ROBIN --loadbalancer lb1 --
   name pool2 --protocol HTTP
   ```

3. リスナー (**listener1**) に L7 ポリシー (**policy1**) を作成します。ポリシーには、アクション (**REDIRECT_TO_POOL**) を追加し、プール (**pool2**) を示す必要があります。

   例

   ```
   $ openstack loadbalancer l7policy create --action REDIRECT_TO_POOL --redirect-pool
   pool2 --name policy1 listener1
   ```

4. HTTP/1.1 ホスト名 www2.example.com を使用するすべてのリクエストを 2 番目のプール (**pool2**) に送信するポリシーに、L7 ルールを追加します。

   例

   ```
   $ openstack loadbalancer l7rule create --compare-type EQUAL_TO --type HOST_NAME --
   value www2.example.com policy1
   ```

検証

1. **openstack loadbalancer l7policy list** コマンドを実行し、ポリシー **policy1** が存在することを確認します。

2. **openstack loadbalancer l7rule list <l7policy>** コマンドを実行し、ポリシーに **compare_type** が **EQUAL_TO** のルールが存在することを確認します。

   例

   ```
   $ openstack loadbalancer l7rule list policy1
   ```

関連情報

- Command Line Interface Reference の loadbalancer pool create
- Command Line Interface Reference の loadbalancer l7policy create

- **Command Line Interface Reference** の loadbalancer l7rule create

11.14. ホスト名末尾に基づくリクエストの特定プールへの送信

レイヤー 7 (L7) ポリシーと共に Red Hat OpenStack Platform (RHOSP) Load-balancing サービス (octavia) を使用して、特定の文字列で終わる HTTP/1.1 ホスト名が含まれるリクエストをアプリケーションサーバーの異なるプールにリダイレクトすることができます。

以下の例では、**.example.com** で終わる HTTP/1.1 ホスト名が含まれるリクエストは、すべてアプリケーションサーバーの別のプール **pool2** にリダイレクトされます。

前提条件

- リスナー (**listener1**) およびプール (**pool1**) を持つ HTTPS ロードバランサー (**lb1**)。詳細は、Creating an HTTP load balancer with a health monitor を参照してください。

手順

1. 認証情報ファイルに source を実行します。

   例

   ```
   $ source ~/overcloudrc
   ```

2. ロードバランサー (**lb1**) に 2 つ目のプール (**pool2**) を作成します。

   注記

   丸かっこ内の値は、この手順のコマンド例で使用されるサンプルの値です。これらのサンプル値を、実際のサイトに適した値に置き換えてください。

   例

   ```
   $ openstack loadbalancer pool create --lb-algorithm ROUND_ROBIN --loadbalancer lb1 --name pool2 --protocol HTTP
   ```

3. リスナー (**listener1**) に L7 ポリシー (**policy1**) を作成します。ポリシーには、アクション (**REDIRECT_TO_POOL**) を追加し、プール (**pool2**) を示す必要があります。

   例

   ```
   $ openstack loadbalancer l7policy create --action REDIRECT_TO_POOL --redirect-pool pool2 --name policy1 listener1
   ```

4. HTTP/1.1 ホスト名 (**www2.example.com**) を使用するすべてのリクエストを 2 番目のプール (**pool2**) に送信するポリシーに、L7 ルールを追加します。

   例

   ```
   $ openstack loadbalancer l7rule create --compare-type ENDS_WITH --type HOST_NAME --value .example.com policy1
   ```

検証

1. **openstack loadbalancer l7policy list** コマンドを実行し、ポリシー **policy1** が存在することを確認します。
2. **openstack loadbalancer l7rule list <l7policy>** コマンドを実行し、ポリシーに **compare_type** が **EQUAL_TO** のルールが存在することを確認します。

   例

   ```
   $ openstack loadbalancer l7rule list policy1
   ```

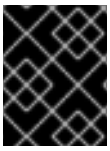
関連情報

- **Command Line Interface Reference** の loadbalancer pool create
- **Command Line Interface Reference** の loadbalancer l7policy create
- **Command Line Interface Reference** の loadbalancer l7rule create

11.15. ブラウザークッキーが存在しないことに基づくリクエストの特定プールへの送信

Red Hat OpenStack Platform (RHOSP) Load-balancing サービス (octavia) を使用して、認証されていない Web クライアントリクエストを、1 つまたは複数の認証サーバーが含まれる異なるプールにリダイレクトすることができます。レイヤー 7 (L7) ポリシーは、受信リクエストに認証クッキーがないかどうかを判断します。

以下の例では、ブラウザークッキー **auth_token** がないすべての Web クライアントリクエストは、認証サーバーが含まれる別のプールにリダイレクトされます。



重要

以下の手順では、ブラウザークッキーを使用した L7 アプリケーションルーティングを実行する例を説明し、セキュリティーの懸念に対処しません。

前提条件

- リスナー (**listener1**) およびプール (**pool1**) を持つ TLS 終端 HTTPS ロードバランサー (**lb1**)。詳細は、Creating a TLS-terminated HTTPS load balancer を参照してください。
- セキュアな認証サーバーが Web ユーザーを認証する 2 番目の RHOSP Networking (neutron) サブネット

手順

1. 認証情報ファイルに source を実行します。

   例

   ```
   $ source ~/overcloudrc
   ```

2. ロードバランサー (**lb1**) に 2 つ目のプール (**login_pool**) を作成します。

注記

丸かっこ内の値は、この手順のコマンド例で使用されるサンプルの値です。これらのサンプル値を、実際のサイトに適した値に置き換えてください。

例

```
$ openstack loadbalancer pool create --lb-algorithm ROUND_ROBIN --loadbalancer lb1 --
name login_pool --protocol HTTP
```

3. 認証サブネット (**secure_subnet**) 上のセキュアな認証サーバー (**192.0.2.10**) を、メンバーとして 2 番目のプールに追加します。

例

```
$ openstack loadbalancer member create --address 192.0.2.10 --protocol-port 80 --subnet-id
secure_subnet login_pool
```

4. リスナー (**listener1**) に L7 ポリシー (**policy1**) を作成します。ポリシーには、アクション (**REDIRECT_TO_POOL**) を追加し、2 つ目のプール (**login_pool**) を参照する必要があります。

例

```
$ openstack loadbalancer l7policy create --action REDIRECT_TO_POOL --redirect-pool
login_pool --name policy1 listener1
```

5. 任意の値のブラウザークッキー (**auth_token**) を探しクッキーが存在しない場合にマッチする L7 ルールを、ポリシー (**policy1**) に追加します。

例

```
$ openstack loadbalancer l7rule create --compare-type REGEX --key auth_token --type
COOKIE --value '.*' --invert policy1
```

検証

1. **openstack loadbalancer l7policy list** コマンドを実行し、ポリシー **policy1** が存在することを確認します。

2. **openstack loadbalancer l7rule list <l7policy>** コマンドを実行し、**compare_type** が **STARTS_WITH** のルールが存在することを確認します。

例

```
$ openstack loadbalancer l7rule list policy1
```

関連情報

- Command Line Interface Reference の loadbalancer pool create
- Command Line Interface Reference の loadbalancer member create
- Command Line Interface Reference の loadbalancer l7policy create

- **Command Line Interface Reference** の loadbalancer l7rule create

11.16. ブラウザークッキーが存在しないことまたは無効なクッキー値に基づくリクエストの特定プールへの送信

Red Hat OpenStack Platform (RHOSP) Load-balancing サービス (octavia) を使用して、認証されていない Web クライアントリクエストを、1 つまたは複数の認証サーバーが含まれる異なるプールにリダイレクトすることができます。レイヤー 7 (L7) ポリシーは、受信リクエストに認証クッキーがないかどうか、または特定の値を持つ認証クッキーが含まれるかどうかを判断します。

以下の例では、ブラウザークッキー **auth_token** がないか、値が **INVALID** である **auth_token** を持つすべての Web クライアントリクエストは、認証サーバーが含まれる別のプールにリダイレクトされます。

重要

以下の手順では、ブラウザークッキーを使用した L7 アプリケーションルーティングを実行する例を説明し、セキュリティーの懸念に対処しません。

前提条件

- リスナー (**listener1**) およびプール (**pool1**) を持つ TLS 終端 HTTPS ロードバランサー (**lb1**)。詳細は、Creating a TLS-terminated HTTPS load balancer を参照してください。
- セキュアな認証サーバーが Web ユーザーを認証する 2 番目の RHOSP Networking (neutron) サブネット

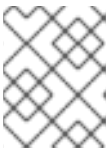
手順

1. 認証情報ファイルに source を実行します。

   例

   ```
   $ source ~/overcloudrc
   ```

2. ロードバランサー (**lb1**) に 2 つ目のプール (**login_pool**) を作成します。

   注記

   丸かっこ内の値は、この手順のコマンド例で使用されるサンプルの値です。これらのサンプル値を、実際のサイトに適した値に置き換えてください。

   例

   ```
   $ openstack loadbalancer pool create --lb-algorithm ROUND_ROBIN --loadbalancer lb1 --
   name login_pool --protocol HTTP
   ```

3. 認証サブネット (**secure_subnet**) 上のセキュアな認証サーバー (**192.0.2.10**) を、メンバーとして 2 番目のプールに追加します。

   例

```
$ openstack loadbalancer member create --address 192.0.2.10 --protocol-port 80 --subnet-id
secure_subnet login_pool
```

4. リスナー (**listener1**) に L7 ポリシー (**policy1**) を作成します。ポリシーには、アクション (**REDIRECT_TO_POOL**) を追加し、2 つ目のプール (**login_pool**) を参照する必要があります。

   例

```
$ openstack loadbalancer l7policy create --action REDIRECT_TO_POOL --redirect-pool
login_pool --name policy1 listener1
```

5. 任意の値のブラウザークッキー (**auth_token**) を探しクッキーが存在しない場合にマッチする L7 ルールを、ポリシー (**policy1**) に追加します。

   例

```
$ openstack loadbalancer l7rule create --compare-type REGEX --key auth_token --type
COOKIE --value '.*' --invert policy1
```

6. リスナー (**listener1**) に 2 つ目の L7 ポリシー (**policy2**) を作成します。ポリシーには、アクション (**REDIRECT_TO_POOL**) を追加し、2 つ目のプール (**login_pool**) を参照する必要があります。

   例

```
$ openstack loadbalancer l7policy create --action REDIRECT_TO_POOL --redirect-pool
login_pool --name policy2 listener1
```

7. ブラウザークッキー (**auth_token**) を検索しクッキーの値が文字列 **INVALID** に等しい場合にマッチする L7 ルールを、2 番目のポリシー (**policy2**) に追加します。

   例

```
$ openstack loadbalancer l7rule create --compare-type EQUAL_TO --key auth_token --type
COOKIE --value INVALID policy2
```

検証

1. **openstack loadbalancer l7policy list** コマンドを実行し、ポリシー **policy1** および **policy2** が存在することを確認します。

2. **openstack loadbalancer l7rule list <l7policy>** コマンドを実行し、**policy1** に **compare_type** が **REGEX** のルールが存在し、**policy2** に **compare_type** が **EQUAL_TO** のルールが存在することを確認します。

   例

```
$ openstack loadbalancer l7rule list policy1
$ openstack loadbalancer l7rule list policy2
```

関連情報

- **Command Line Interface Reference** の loadbalancer pool create
- **Command Line Interface Reference** の loadbalancer member create
- **Command Line Interface Reference** の loadbalancer l7policy create
- **Command Line Interface Reference** の loadbalancer l7rule create

11.17. 名前がホスト名およびパスと一致するリクエストのプールへの送信

Red Hat OpenStack Platform (RHOSP) Load-balancing サービス (octavia) を使用して、特定の条件に一致する Web クライアントリクエストをアプリケーションサーバーの別のプールにリダイレクトすることができます。ビジネスロジックの条件は、事前定義されたホスト名およびリクエストパスを照合しようとするレイヤー 7 (L7) ポリシーで実行されます。

以下の例では、ホスト名 **api.example.com** に一致するか、リクエストパスの先頭が **/api** であるすべての Web クライアントリクエストは、別のプール **api_pool** にリダイレクトされます。

前提条件

- リスナー (**listener1**) およびプール (**pool1**) を持つ HTTPS ロードバランサー (**lb1**)。詳細は、Creating an HTTP load balancer with a health monitor を参照してください。

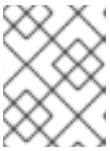
手順

1. 認証情報ファイルに source を実行します。

   例

   ```
   $ source ~/overcloudrc
   ```

2. ロードバランサー (**lb1**) に 2 番目のプール (**api_pool**) を作成します。

   注記

   丸かっこ内の値は、この手順のコマンド例で使用されるサンプルの値です。これらのサンプル値を、実際のサイトに適した値に置き換えてください。

   例

   ```
   $ openstack loadbalancer pool create --lb-algorithm ROUND_ROBIN --loadbalancer lb1 --
   name api_pool --protocol HTTP
   ```

3. プライベートサブネット (**private_subnet**) のロードバランサーメンバー (**192.0.2.10** および **192.0.2.11**) をプール (**static_pool**) に追加します。

   例

   ```
   $ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.10 --
   protocol-port 80 static_pool
   $ openstack loadbalancer member create --subnet-id private_subnet --address 192.0.2.11 --
   protocol-port 80 static_pool
   ```

4. リスナー (**listener1**) に L7 ポリシー (**policy1**) を作成します。ポリシーには、アクション (**REDIRECT_TO_POOL**) を追加し、プール (**api_pool**) を示す必要があります。

   例

   ```
   $ openstack loadbalancer l7policy create --action REDIRECT_TO_POOL --redirect-pool
   api_pool --name policy1 listener1
   ```

5. ホスト名 **api.example.com** にマッチする L7 ルールを、ポリシーに追加します。

   例

   ```
   $ openstack loadbalancer l7rule create --compare-type EQUAL_TO --type HOST_NAME --
   value api.example.com policy1
   ```

6. リクエスパスの最初の **/api** にマッチする 2 番目の L7 ルールを、ポリシーに追加します。このルールは、最初のルールと論理的に AND で結合されます。

   例

   ```
   $ openstack loadbalancer l7rule create --compare-type STARTS_WITH --type PATH --value
   /api policy1
   ```

検証

1. **openstack loadbalancer l7policy list** コマンドを実行し、ポリシー **policy1** が存在することを確認します。
2. **openstack loadbalancer l7rule list <l7policy>** コマンドを実行し、**policy1** に **compare_type** が **STARTS_WITH** および **STARTS_WITH** であるルールが共に存在することを確認します。

   例

   ```
   $ openstack loadbalancer l7rule list policy1
   $ openstack loadbalancer l7rule list policy2
   ```

関連情報

- Command Line Interface Reference の loadbalancer pool create
- Command Line Interface Reference の loadbalancer member create
- Command Line Interface Reference の loadbalancer l7policy create
- Command Line Interface Reference の loadbalancer l7rule create

11.18. COOKIE を使用して既存の本番サイトで AB テストを構成

レイヤー 7 (L7) ポリシーと共に Red Hat OpenStack Platform (RHOSP) Load-balancing サービス (octavia) を使用して、実稼働環境 Web サイトの A-B テスト (または分割テスト) を設定できます。

以下の例では、Web サイトの「B」バージョンにルーティングする Web クライアントは、プール (**pool1**) のメンバーサーバーによってクッキー **site_version** を **B** に設定します。

前提条件

- 2 つの実稼働環境用 Web サイト (サイト A とサイト B)
- HTTP ロードバランサーが、「開始パスに基づくリクエストのプールへのリダイレクト」の手順に従って設定されている。 必要な設定の概要は以下のとおりです。
  - ロードバランサー (**lb1**) のリスナー (**listener1**)。
  - **/js** または **/images** で始まる URL を持つ HTTP リクエストは、プール (**static_pool**) に送信される。
  - 他のすべてのリクエストは、リスナーのデフォルトプール (**pool1**) に送信される。
  - 設定についての詳細は、「開始パスに基づくリクエストのプールへのリダイレクト」 を参照してください。

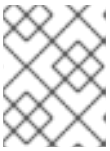
手順

1. 認証情報ファイルに source を実行します。

   例

   ```
   $ source ~/overcloudrc
   ```

2. ロードバランサー (**lb1**) に 3 番目のプール (**pool_B**) を作成します。

   注記

   丸かっこ内の値は、この手順のコマンド例で使用されるサンプルの値です。これらのサンプル値を、実際のサイトに適した値に置き換えてください。

   例

   ```
   $ openstack loadbalancer pool create --lb-algorithm ROUND_ROBIN --loadbalancer lb1 --
   name pool_B --protocol HTTP
   ```

3. プライベートサブネット (**private_subnet**) のロードバランサーメンバー (**192.0.2.50** および **192.0.2.51**) をプール (**pool_B**) に追加します。

   例

   ```
   $ openstack loadbalancer member create --address 192.0.2.50 --protocol-port 80 --subnet-id
   private_subnet pool_B
   $ openstack loadbalancer member create --address 192.0.2.51 --protocol-port 80 --subnet-id
   private_subnet pool_B
   ```

4. ロードバランサー (**lb1**) に 4 番目のプール (**static_pool_B**) を作成します。

   例

   ```
   $ openstack loadbalancer pool create --lb-algorithm ROUND_ROBIN --loadbalancer lb1 --
   name static_pool_B --protocol HTTP
   ```

5. プライベートサブネット (**private_subnet**) のロードバランサーメンバー (**192.0.2.100** および **192.0.2.101**) をプール (**static_pool_B**) に追加します。

   例

   ```
   $ openstack loadbalancer member create --address 192.0.2.100 --protocol-port 80 --subnet-id private_subnet static_pool_B
   $ openstack loadbalancer member create --address 192.0.2.101 --protocol-port 80 --subnet-id private_subnet static_pool_B
   ```

6. リスナー (**listener1**) に L7 ポリシー (**policy2**) を作成します。ポリシーには、アクション (**REDIRECT_TO_POOL**) を追加し、プール (**static_pool_B**) を示す必要があります。**1** の位置にポリシーを挿入します。

   例

   ```
   $ openstack loadbalancer l7policy create --action REDIRECT_TO_POOL --redirect-pool static_pool_B --name policy2 --position 1 listener1
   ```

7. リクエストパスの最初の **/js** または **/images** にマッチする正規表現を使用する L7 ルールを、ポリシー (**policy2**) に追加します。

   例

   ```
   $ openstack loadbalancer l7rule create --compare-type REGEX --type PATH --value '^/(js|images)' policy2
   ```

8. 文字列 (**B**) のクッキー (**site_version**) にマッチする 2 番目の L7 ルールを、ポリシー (**policy2**) に追加します。

   例

   ```
   $ openstack loadbalancer l7rule create --compare-type EQUAL_TO --key site_version --type COOKIE --value B policy2
   ```

9. リスナー (**listener1**) に L7 ポリシー (**policy3**) を作成します。ポリシーには、アクション (**REDIRECT_TO_POOL**) を追加し、プール (**pool_B**) を示す必要があります。**2** の位置にポリシーを挿入します。

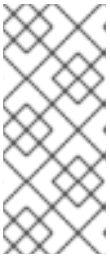
   例

   ```
   $ openstack loadbalancer l7policy create --action REDIRECT_TO_POOL --redirect-pool pool_B --name policy3 --position 2 listener1
   ```

10. 文字列 (**B**) のクッキー (**site_version**) にマッチする L7 ルールを、ポリシー (**policy3**) に追加します。

    例

    ```
    $ openstack loadbalancer l7rule create --compare-type EQUAL_TO --key site_version --type COOKIE --value B policy3
    ```



注記

最も厳しいルールを持つ L7 ポリシーが低い位置に割り当てられることが重要です。これは、すべてのルールが True と評価される最初のポリシーが、アクションが実行されるポリシーになるからです。この手順では、誤ったプールにリクエストが送信されないように、**policy3** の前に **policy2** が評価される必要があります。

検証

1. **openstack loadbalancer l7policy list** コマンドを実行し、ポリシー **policy2** および **policy3** が存在することを確認します。
2. **openstack loadbalancer l7rule list <l7policy>** コマンドを実行し、それぞれのポリシーごとに **compare_type** が **STARTS_WITH** のルールが存在することを確認します。

   例

```
$ openstack loadbalancer l7rule list policy2
$ openstack loadbalancer l7rule list policy3
```

関連情報

- Command Line Interface Reference の loadbalancer pool create
- Command Line Interface Reference の loadbalancer member create
- Command Line Interface Reference の loadbalancer l7policy create
- Command Line Interface Reference の loadbalancer l7rule create

第12章 LOAD-BALANCING サービスの更新およびアップグレード

定期的に更新およびアップグレードを実施すると、最新の Red Hat OpenStack Platform Load-balancing サービス機能を利用できます。また、更新およびアップグレードを頻繁に行わないことによって生じる長期に及ぶ問題の発生を防ぐことができます。

- 「Load-balancing サービスの更新およびアップグレード」
- 「実行中の Load-balancing サービスインスタンスの更新」

12.1. LOAD-BALANCING サービスの更新およびアップグレード

Load-balancing サービス (octavia) は、Red Hat OpenStack Platform (RHOSP) の更新またはアップグレードの一部です。

前提条件

- アップグレード中に負荷分散サービスコントロールプレーンが完全には機能しなくなるので、アップグレードを実行するためのメンテナンス期間がスケジュールされていること。

手順

1. **『Red Hat OpenStack Platform の最新状態の維持』**に記載の手順に従って、RHOSP の更新を実行します。
2. メンテナンスリリースを適用した後に新機能を使用する必要がある場合は、実行中の amphora をローテーションして最新の amphora イメージに更新します。

関連情報

- 『Red Hat OpenStack Platform の最新状態の維持』
- 「実行中の Load-balancing サービスインスタンスの更新」
- 「Load-balancing サービス (octavia) 機能のサポートマトリックス」

12.2. 実行中の LOAD-BALANCING サービスインスタンスの更新

定期的に、実行中の Load-balancing サービスインスタンス (amphora) をより新しいイメージで更新することができます。たとえば、以下のイベント時に amphora インスタンスを更新する必要があります。

- Red Hat OpenStack Platform (RHOSP) の更新またはアップグレード
- システムへのセキュリティーアップデート
- ベースとなる仮想マシンのフレーバー変更

RHOSP の更新またはアップグレード時に、director は自動的にデフォルトの amphora イメージをダウンロードし、それをオーバークラウドの Image サービス (glance) にアップロードし、負荷分散サービス (octavia) が新しいイメージを使用するように設定します。ロードバランサーをフェイルオーバーするときは、負荷分散サービスに、新しい amphora イメージを使用するインスタンス (amphora) を強制的に開始させます。

前提条件

- amphora の新しいイメージ。これらは RHOSP の更新またはアップグレード時に利用可能です。

手順

1. 認証情報ファイルに source を実行します。

   例

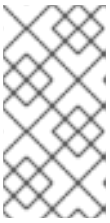
   ```
   $ source ~/overcloudrc
   ```

2. 更新するすべてのロードバランサーの ID を一覧表示します。

   ```
   $ openstack loadbalancer list -c id -f value
   ```

3. それぞれのロードバランサーをフェイルオーバーします。

   ```
   $ openstack loadbalancer failover <loadbalancer_id>
   ```

   注記

   ロードバランサーのフェイルオーバーを開始したら、システムの使用状況を監視し、必要に応じてフェイルオーバーを実行する速度を調整します。ロードバランサーのフェイルオーバーにより、新規仮想マシンおよびポートが作成されます。これにより、一時的に OpenStack Networking の負荷が高まる場合があります。

4. ロードバランサーのフェイルオーバーの状態を監視します。

   ```
   $ openstack loadbalancer show <loadbalancer_id>
   ```

   ロードバランサーのステータスが **ACTIVE** になれば、更新は完了です。

関連情報

- Command Line Interface Referenceのloadbalancer

第13章 LOAD-BALANCING サービスのトラブルシューティングおよびメンテナンス

Load-balancing サービス (octavia) の基本的なトラブルシューティングとメンテナンスは、ステータスを表示しインスタンスを移行するための OpenStack クライアントコマンドを熟知し、ログへのアクセス方法を理解することから始まります。より詳細なトラブルシューティングを行う必要がある場合は、1 つまたは複数の Load-balancing サービスインスタンス (amphora) に SSH 接続することができます。

- 「ロードバランサーの検証」
- 「Load-balancing サービスインスタンスの管理ログ」
- 「特定の Load-balancing サービスインスタンスの移行」
- 「SSH を使用した負荷分散インスタンスへの接続」
- 「リスナー統計の表示」
- 「リスナーリクエストエラーの解釈」

13.1. ロードバランサーの検証

ロードバランサーの show コマンドと list コマンドの出力を表示することで、負荷分散サービス (octavia) とそのさまざまなコンポーネントのトラブルシューティングを行うことができます。

手順

1. 認証情報ファイルに source を実行します。

   例

   ```
   $ source ~/overcloudrc
   ```

2. ロードバランサー (**lb1**) の設定を確認します。

   注記

   丸かっこ内の値は、この手順のコマンド例で使用されるサンプルの値です。これらのサンプル値を、実際のサイトに適した値に置き換えてください。

   例

   ```
   $ openstack loadbalancer show lb1
   ```

   出力例

   ```
   +---------------------+--------------------------------------+
   | Field               | Value                                |
   +---------------------+--------------------------------------+
   | admin_state_up      | True                                 |
   | created_at          | 2022-02-17T15:59:18                  |
   | description         |                                      |
   ```

```
| flavor_id           | None                                 |
| id                  | 265d0b71-c073-40f4-9718-8a182c6d53ca |
| listeners           | 5aaa67da-350d-4125-9022-238e0f7b7f6f |
| name                | lb1                                  |
| operating_status    | ONLINE                               |
| pools               | 48f6664c-b192-4763-846a-da568354da4a |
| project_id          | 52376c9c5c2e434283266ae7cacd3a9c     |
| provider            | amphora                              |
| provisioning_status | ACTIVE                               |
| updated_at          | 2022-02-17T16:01:21                  |
| vip_address         | 192.0.2.177                          |
| vip_network_id      | afeaf55e-7128-4dff-80e2-98f8d1f2f44c |
| vip_port_id         | 94a12275-1505-4cdc-80c9-4432767a980f |
| vip_qos_policy_id   | None                                 |
| vip_subnet_id       | 06ffa90e-2b86-4fe3-9731-c7839b0be6de |
+---------------------+--------------------------------------+
```

3. 前の手順の loadbalancer ID (**265d0b71-c073-40f4-9718-8a182c6d53ca**) を使用して、ロードバランサーに関連付けられている amphora の ID (**lb1**) を取得します。

例

```
$ openstack loadbalancer amphora list | grep 265d0b71-c073-40f4-9718-8a182c6d53ca
```

出力例

```
| 1afabefd-ba09-49e1-8c39-41770aa25070 | 265d0b71-c073-40f4-9718-8a182c6d53ca |
ALLOCATED | STANDALONE | 198.51.100.7  | 192.0.2.177   |
```

4. 前の手順の amphora ID (**1afabefd-ba09-49e1-8c39-41770aa25070**) を使用して、amphora 情報を表示します。

例

```
$ openstack loadbalancer amphora show 1afabefd-ba09-49e1-8c39-41770aa25070
```

出力例

```
+-----------------+--------------------------------------+
| Field           | Value                                |
+-----------------+--------------------------------------+
| id              | 1afabefd-ba09-49e1-8c39-41770aa25070 |
| loadbalancer_id | 265d0b71-c073-40f4-9718-8a182c6d53ca |
| compute_id      | ba9fc1c4-8aee-47ad-b47f-98f12ea7b200 |
| lb_network_ip   | 198.51.100.7                         |
| vrrp_ip         | 192.0.2.36                           |
| ha_ip           | 192.0.2.177                          |
| vrrp_port_id    | 07dcd894-487a-48dc-b0ec-7324fe5d2082 |
| ha_port_id      | 94a12275-1505-4cdc-80c9-4432767a980f |
| cert_expiration | 2022-03-19T15:59:23                  |
| cert_busy       | False                                |
| role            | STANDALONE                           |
| status          | ALLOCATED                            |
| vrrp_interface  | None                                 |
```

```
| vrrp_id         | 1                                    |
| vrrp_priority   | None                                 |
| cached_zone     | nova                                 |
| created_at      | 2022-02-17T15:59:22                  |
| updated_at      | 2022-02-17T16:00:50                  |
| image_id        | 53001253-5005-4891-bb61-8784ae85e962 |
| compute_flavor  | 65                                   |
+-----------------+--------------------------------------+
```

5. リスナー (**listener1**) の詳細を表示します。

例

```
$ openstack loadbalancer listener show listener1
```

出力例

```
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| connection_limit          | -1                                   |
| created_at                | 2022-02-17T16:00:59                  |
| default_pool_id           | 48f6664c-b192-4763-846a-da568354da4a |
| default_tls_container_ref | None                                 |
| description               |                                      |
| id                        | 5aaa67da-350d-4125-9022-238e0f7b7f6f |
| insert_headers            | None                                 |
| l7policies                |                                      |
| loadbalancers             | 265d0b71-c073-40f4-9718-8a182c6d53ca |
| name                      | listener1                            |
| operating_status          | ONLINE                               |
| project_id                | 52376c9c5c2e434283266ae7cacd3a9c     |
| protocol                  | HTTP                                 |
| protocol_port             | 80                                   |
| provisioning_status       | ACTIVE                               |
| sni_container_refs        | []                                   |
| timeout_client_data       | 50000                                |
| timeout_member_connect    | 5000                                 |
| timeout_member_data       | 50000                                |
| timeout_tcp_inspect       | 0                                    |
| updated_at                | 2022-02-17T16:01:21                  |
| client_ca_tls_container_ref | None                               |
| client_authentication     | NONE                                 |
| client_crl_container_ref  | None                                 |
| allowed_cidrs             | None                                 |
+---------------------------+--------------------------------------+
```

6. プール (**pool1**) とロードバランサーメンバーを表示します。

例

```
$ openstack loadbalancer pool show pool1
```

出力例

```
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| admin_state_up      | True                                 |
| created_at          | 2022-02-17T16:01:08                  |
| description         |                                      |
| healthmonitor_id    | 4b24180f-74c7-47d2-b0a2-4783ada9a4f0 |
| id                  | 48f6664c-b192-4763-846a-da568354da4a |
| lb_algorithm        | ROUND_ROBIN                          |
| listeners           | 5aaa67da-350d-4125-9022-238e0f7b7f6f |
| loadbalancers       | 265d0b71-c073-40f4-9718-8a182c6d53ca |
| members             | b92694bd-3407-461a-92f2-90fb2c4aedd1 |
|                     | 4ccdd1cf-736d-4b31-b67c-81d5f49e528d |
| name                | pool1                                |
| operating_status    | ONLINE                               |
| project_id          | 52376c9c5c2e434283266ae7cacd3a9c     |
| protocol            | HTTP                                 |
| provisioning_status | ACTIVE                               |
| session_persistence | None                                 |
| updated_at          | 2022-02-17T16:01:21                  |
| tls_container_ref   | None                                 |
| ca_tls_container_ref | None                                |
| crl_container_ref   | None                                 |
| tls_enabled         | False                                |
+---------------------+--------------------------------------+
```

7. ロードバランサーの VIP アドレス(**192.0.2.177**)に接続して、リスナーが **HTTPS** または **TERMINATED_HTTPS** プロトコルに設定されているロードバランサー全体に、HTTPS トラフィックが流れることを確認します。

ヒント

コマンド **openstack loadbalancer show <load_balancer_name>** を使用して、ロードバランサーの VIP アドレスを取得します。

例

```
$ curl -v https://192.0.2.177 --insecure
```

出力例

```
* About to connect() to 192.0.2.177 port 443 (#0)
*   Trying 192.0.2.177...
* Connected to 192.0.2.177 (192.0.2.177) port 443 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
* skipping SSL peer certificate verification
* SSL connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate:
*  subject: CN=www.example.com,O=Dis,L=Springfield,ST=Denial,C=US
*  start date: Jan 15 09:21:45 2021 GMT
*  expire date: Jan 15 09:21:45 2021 GMT
*  common name: www.example.com
```

```
*  issuer: CN=www.example.com,O=Dis,L=Springfield,ST=Denial,C=US
> GET / HTTP/1.1
> User-Agent: curl/7.29.0
> Host: 192.0.2.177
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Length: 30
<
* Connection #0 to host 192.0.2.177 left intact
```

関連情報

- **Command Line Interface Reference**のloadbalancer

13.2. LOAD-BALANCING サービスインスタンスの管理ログ

Load-balancing サービスインスタンス (amphora) の管理ログオフロード機能は、テナントフローログを除く amphora 内のすべてのシステムロギングを対象とします。管理ログが送信されるのと同じ syslog レシーバーにテナントフローログを送信できます。管理ログを処理するのと同じ syslog レシーバーにテナントフローログを送信できますが、テナントフローログを個別に設定する必要があります。

amphora は、メッセージを送信するアプリケーションのネイティブログ形式を使用して、すべての管理ログメッセージを送信します。amphora は、他の Red Hat OpenStack Platform (RHOSP) ログと同じ場所の RHOSP Controller ノードにログを記録します (**/var/log/containers/octavia/**)。

関連情報

- *5章Load-balancing サービスインスタンスのログの管理*

13.3. 特定の LOAD-BALANCING サービスインスタンスの移行

場合によっては、負荷分散サービスインスタンス (amphora) を移行する必要があります。たとえば、ホストがメンテナンスのためにシャットダウンされている場合

手順

1. 認証情報ファイルに source を実行します。

   例

   ```
   $ source ~/overcloudrc
   ```

2. 移行する amphora の ID を特定します。後のステップで ID を指定する必要があります。

   ```
   $ openstack loadbalancer amphora list
   ```

3. Compute スケジューラーサービスが退避しているコンピュートノードに新しい amphora をスケジュールしないようにするには、コンピュートノード (**compute-host-1**) を無効にします。

注記

丸かっこ内の値は、この手順のコマンド例で使用されるサンプルの値です。これらのサンプル値を、実際のサイトに適した値に置き換えてください。

例

```
$ openstack compute service set compute-host-1 nova-compute --disable
```

4. 取得した amphora ID (**ea17210a-1076-48ff-8a1f-ced49ccb5e53**) を使用して amphora をフェイルオーバーします。

例

```
$ openstack loadbalancer amphora failover ea17210a-1076-48ff-8a1f-ced49ccb5e53
```

関連情報

- **Command Line Interface Reference** の compute service set
- **Command Line Interface Reference**の loadbalancer

13.4. SSH を使用した負荷分散インスタンスへの接続

サービスの問題をトラブルシューティングするときは、SSH を使用して負荷分散サービスインスタンス (amphora) にログインします。

サービスの問題のトラブルシューティングを行う際に、Secure Shell (SSH) を使用して実行中の Load-balancing サービスインスタンス (amphora) にログインすると便利な場合があります。

前提条件

- 負荷分散サービス (octavia) の SSH 秘密鍵が必要です。
- ロードバランサーを作成する前に、ロードバランシングサービス構成で SSH を有効にする必要があります。

手順

1. ディレクターノードで、**ssh-agent** を起動し、ユーザー ID キーをエージェントに追加します。

```
$ eval `ssh-agent -s`
$ ssh-add
```

2. 認証情報ファイルに source を実行します。

例

```
$ source ~/overcloudrc
```

3. 接続する amphora の負荷分散管理ネットワーク (**lb_network_ip**) 上の IP アドレスを把握します。

```
$ openstack loadbalancer amphora list
```

4. SSH を使用して amphora に接続します。

```
$ ssh -A -t heat-admin@<controller_node_IP_address> ssh cloud-user@<lb_network_ip>
```

5. 終了したら、amphora への接続を閉じて、SSH エージェントを停止します。

```
$ exit
```

関連情報

- **Command Line Interface Reference**の**loadbalancer**

13.5. リスナー統計の表示

OpenStack クライアントを使用して、特定の Red Hat OpenStack Platform (RHOSP) ロードバランサーのリスナーに関する統計を取得できます。

- 現在のアクティブな接続 (**active_connections**)
- 受信バイト数の合計 (**bytes_in**)。
- 送信されたバイトの合計 (**bytes_out**)。
- 満たできなかった要求の合計 (**request_errors**)。
- 処理した合計接続数 (**total_connections**)

手順

1. 認証情報ファイルに source を実行します。

   例

```
$ source ~/overcloudrc
```

2. リスナー (**listener1**) の統計を表示します。

注記

丸かっこ内の値は、この手順のコマンド例で使用されるサンプルの値です。これらのサンプル値を、実際のサイトに適した値に置き換えてください。

例

```
$ openstack loadbalancer listener stats show listener1
```

ヒント

リスナーの名前が分からない場合は、**loadbalancer listener list** コマンドを実行します。

出力例

```
+--------------------+-------+
| Field              | Value |
+--------------------+-------+
| active_connections | 0     |
| bytes_in           | 0     |
| bytes_out          | 0     |
| request_errors     | 0     |
| total_connections  | 0     |
+--------------------+-------+
```

関連情報

- Command Line Interface Reference の loadbalancer listener stats show
- 「リスナーリクエストエラーの解釈」

13.6. リスナーリクエストエラーの解釈

特定の Red Hat OpenStack Platform (RHOSP) ロードバランサーのリスナーに関する統計を取得できます。詳細は、「リスナー統計の表示」 を参照してください。

RHOSP ロードバランサー (**request_errors**) が追跡する統計の 1 つが、ロードバランサーに接続するエンドユーザーからの要求で発生したエラーのみをカウントします。**request_errors** 変数は、メンバーサーバーによって報告されるエラーを測定しません。

たとえば、テナントが RHOSP Load-balancing サービス (octavia) を介して HTTP ステータスコード **400 (Bad Request)** を返す Web サーバーに接続する場合、このエラーは Load-balancing サービスによって収集されません。ロードバランサーは、データトラフィックの内容を検査しません。この例では、ロードバランサーはユーザーと Web サーバーと正しく情報を転送するため、このフローを成功として解釈します。

以下の条件により、**request_errors** 変数が増分する可能性があります。

- 要求を送信する前に、クライアントから早期の終了を行います。
- クライアントからエラーを読み取ります。
- クライアントのタイムアウト。
- クライアントは接続を閉じます。
- クライアントからのさまざまな不適切な要求。

関連情報

- Command Line Interface Reference の loadbalancer listener stats show
- 「リスナー統計の表示」