Red Hat OpenStack Platform 13

使用 Octavia 进行负载均衡即服务

Octavia 管理以及如何使用 octavia 在数据平面之间负载均衡网络流量。

Last Updated: 2023-04-20

Red Hat OpenStack Platform 13 使用 Octavia 进行负载均衡即服务

Octavia 管理以及如何使用 octavia 在数据平面之间负载均衡网络流量。

OpenStack Team
rhos-docs@redhat.com

法律通告

摘要

安装、配置、操作、故障排除和升级 Red Hat OpenStack Platform (RHOSP)负载均衡服务 (octavia)。

目录

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看 CTO Chris Wright 的信息。

对红帽文档提供反馈

我们感谢您对文档提供反馈信息。告诉我们我们如何使其更好。

使用直接文档反馈(DDF)功能

使用 **添加反馈** DDF 函数进行具体句子、段落或代码块的直接评论。

1. 以 *Multi-page HTML* 格式查看文档。
2. 确保您在文档右上角看到 **反馈** 按钮。
3. 用鼠标指针高亮显示您想评论的文本部分。
4. 单击**添加反馈**。
5. 将 **添加反馈** 字段填写您的意见。
6. 可选：添加您的电子邮件地址，以便文档团队可以与您联系以获取您的问题。
7. 点 **Submit**。

第 1 章 负载均衡服务简介

负载均衡服务(octavia)为 Red Hat OpenStack Platform (RHOSP)部署提供一个负载平衡即服务(LBaaS) API 版本 2 实施。负载平衡服务管理多个虚拟机、容器或裸机服务器- 称为 amphorae-，它按需启动。提供按需横向扩展功能，使负载平衡服务成为适合大型 RHOSP 企业部署的全功能负载均衡器。

注意

红帽不支持从 Neutron-LBaaS 到负载平衡服务的迁移路径。您可以使用一些不受支持的开源工具。例如，在 GitHub 上搜索 nlbaas2octavia-lb-replicator。

- 第 1.1 节 “负载均衡服务组件”
- 第 1.2 节 “负载均衡服务对象模型”
- 第 1.3 节 “在 Red Hat OpenStack Platform 中使用负载均衡”

1.1. 负载均衡服务组件

Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)使用一组虚拟机实例（称为 *amphorae* 驻留在 Compute 节点上）。负载平衡服务控制器通过负载平衡管理网络(**lb-mgmt-net**)与 amphorae 通信。

使用 octavia 时，您可以创建不需要浮动 IP (FIP)的负载均衡器虚拟 IP (VIP)。不使用 FIP 时具有通过负载平衡器提高性能的优势。

图 1.1. 负载均衡服务组件

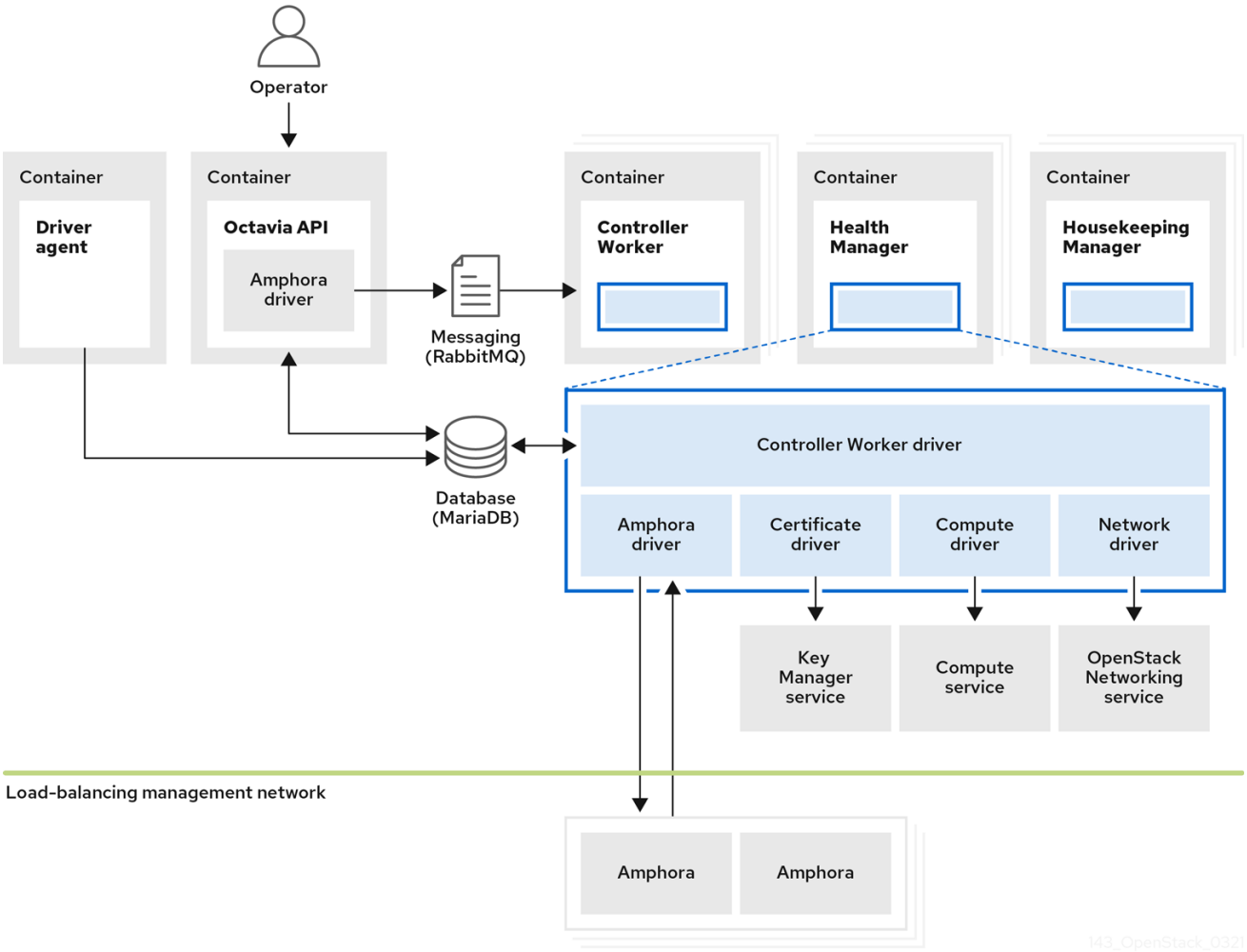


图 1.1 显示负载均衡服务的组件托管在与网络 API 服务器相同的节点上，默认情况下，位于 Controller 节点上。负载均衡服务由以下组件组成：

Octavia API (octavia_api 容器)

为用户提供 REST API 与 octavia 交互。

Controller Worker (octavia_worker 容器)

将配置和配置更新发送到负载平衡管理网络上的 amphorae。

健康管理器(octavia_health_manager 容器)

监控单独的 amphorae 的健康状况，并在 amphora 遇到故障时处理故障转移事件。

内部管理器(octavia_housekeeping 容器)

清理已删除的数据库记录，并管理 amphora 证书轮转。

驱动程序代理(octavia_driver_agent 容器)

支持其他提供程序驱动程序，如 OVN。

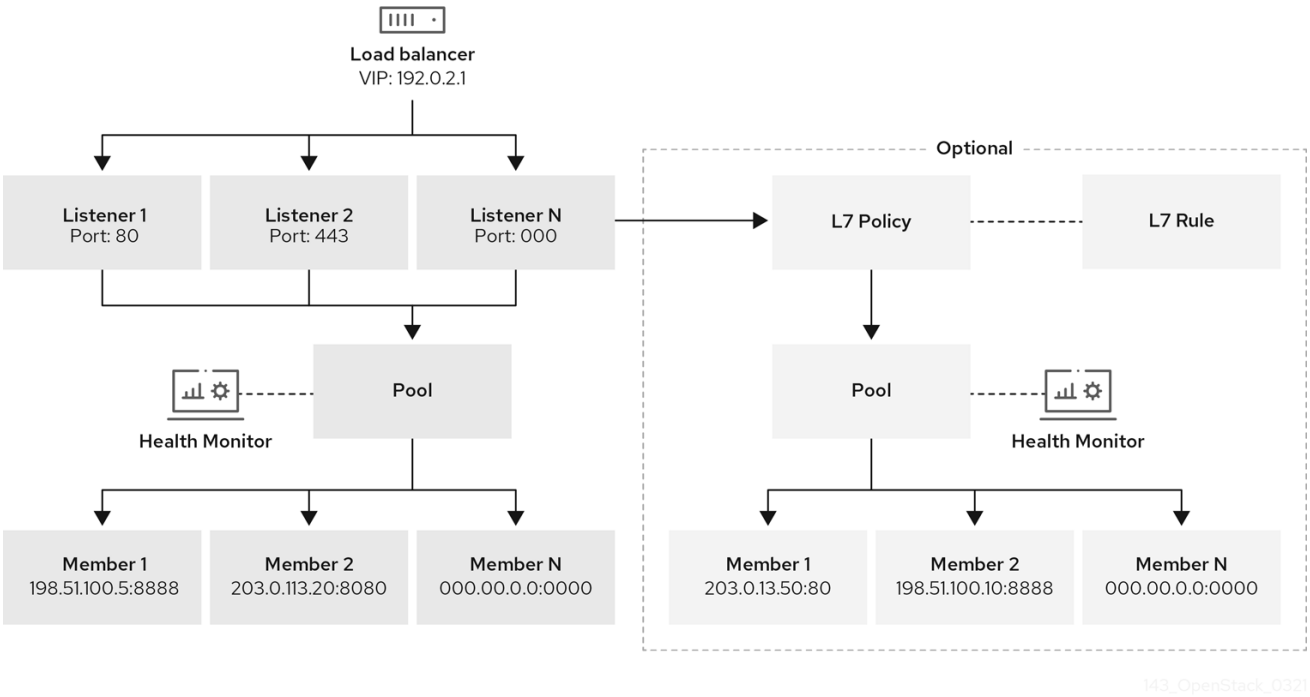
Amphora

执行负载平衡。Amphorae 通常是根据监听程序、池、运行状况监视器、L7 策略和成员配置在 Compute 节点上运行的实例。Amphorae 向 Health Manager 发送定期心跳。

1.2. 负载均衡服务对象模型

Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)使用典型的负载均衡对象模型。

图 1.2. 负载平衡服务对象模型图



负载均衡器

代表负载均衡实体的顶级 API 对象。创建负载均衡器时，会分配 VIP 地址。当您使用 amphora 供应商创建负载均衡器时，一个或多个 amphora 实例在一个或多个 Compute 节点上启动。

侦听器

负载均衡器侦听的端口，如 HTTP 的 TCP 端口 80。

健康监控

在每个后端成员服务器上执行定期健康检查的过程，以便预先检测失败的服务器并从池中临时删除它们。

pool

处理来自负载平衡器的客户端请求的一组成员。您可以使用 API 将池与多个监听程序关联。您可以使用 L7 策略共享池。

成员

介绍如何连接后端实例或服务。此描述由后端成员的 IP 地址和网络端口组成。

L7 规则

定义第 7 层(L7)条件，用于决定 L7 策略是否应用到连接。

L7 策略

与监听程序关联的 L7 规则集合，它可能与后端池关联。策略描述在策略中所有规则为 true 时负载均衡器执行的操作。

其他资源

- 第 1.1 节 “负载均衡服务组件”

1.3. 在 RED HAT OPENSTACK PLATFORM 中使用负载均衡

负载平衡对于为云部署启用简单或自动交付扩展和可用性至关重要。负载均衡服务(octavia)依赖于其他 Red Hat OpenStack Platform (RHOSP)服务：

- **计算服务(nova)** - 管理负载平衡服务虚拟机实例(amphora)生命周期，并按需创建计算资源。
- **网络服务(neutron)** - 对于 amphorae、租户环境和外部网络之间的网络连接。
- **Key Manager 服务(barbican)** - 在监听器上配置 TLS 会话终止时，用于管理 TLS 证书和密钥。
- **Identity Service (keystone)- 对 octavia API 的身份验证请求，以及用于与其他 RHOSP 服务进行身份验证** 的负载均衡服务。
- **Image 服务(glance)** - 用于存储 amphora 虚拟机镜像。
- **common Libraries (oslo)** - 用于负载平衡服务控制器组件之间的通信，使负载均衡服务在标准 OpenStack 框架内工作，并检查系统以及项目代码结构。
- **Taskflow** - Common Libraries 的一部分；负载平衡服务在编排后端服务配置和管理时使用此作业流系统。

负载均衡服务通过驱动程序接口与其他 RHOSP 服务交互。如果外部组件需要使用功能兼容的服务替换，驱动接口避免了负载平衡服务的主要重组。

第 2 章 实施负载均衡服务的注意事项

在计划部署 Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)时，您必须做出几个决策，例如选择要使用的供应商，还是要实施高度可用的环境：

- 第 2.1 节 “负载均衡服务供应商驱动程序”
- 第 2.2 节 “负载均衡服务(octavia)功能支持列表”
- 第 2.3 节 “负载平衡服务软件要求”
- 第 2.4 节 “undercloud 的负载均衡服务先决条件”
- 第 2.5 节 “负载均衡服务实例的活动拓扑的基础知识”
- 第 2.6 节 “负载均衡服务部署后步骤”

2.1. 负载均衡服务供应商驱动程序

Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)支持使用 Octavia v2 API 启用多个供应商驱动程序。您可以选择同时使用一个供应商驱动程序，或者同时使用多个供应商驱动程序。

RHOSP 提供两种负载平衡供应商，即 amphora 和 Open Virtual Network (OVN)。

Amphora，默认是高度可用的负载均衡器，它带有与计算环境扩展的功能集。因此，amphora 适用于大规模部署。

OVN 负载均衡供应商是一个轻量级负载均衡器，具有基本功能集。OVN 是东西层 4 层网络流量的典型。与功能齐全的负载均衡供应商（如 amphora）相比，OVN 快速启动并消耗资源更少。

在将 neutron Modular Layer 2 插件与 OVN 机制驱动程序(ML2/OVN)的 RHOSP 部署中，RHOSP director 会在负载均衡服务中自动启用 OVN 供应商驱动程序，而无需额外的安装或配置。

重要

本节中的信息仅适用于 amphora 负载均衡供应商，除非另有说明。

其他资源

- 第 2.2 节 “负载均衡服务(octavia)功能支持列表”

2.2. 负载均衡服务(OCTAVIA)功能支持列表

Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)提供两个负载均衡供应商 amphora 和 Open Virtual Network (OVN)。

Amphora 是一个功能齐全的负载均衡供应商，需要单独的 haproxy 虚拟机和额外的延迟跃点。

OVN 在每个节点上运行，不需要单独的虚拟机或额外的跃点。但是，OVN 的负载平衡功能比 amphora 少。

下表列出了 Red Hat OpenStack Platform (RHOSP) 13 支持的 octavia 的功能，以及该功能的维护支持。

注意

如果没有列出该功能，RHOSP 13 不支持该功能。

表 2.1. 负载均衡服务(octavia)功能支持列表

功能	RHOSP 13 版本中的支持级别	
	Amphora Provider	OVN Provider
ML2/OVS L3 HA	完全支持-13.0 以及所有维护版本	Not applicable
ML2/OVS DVR	完全支持-29 年 8 月 2018 维护支持版本	Not applicable
ML2/OVS L3 HA + composable network node [1]	完全支持-13 2019 年 3 月维护版本及更新版本	Not applicable
ML2/OVS DVR + composable network node [1]	完全支持-13 2019 年 3 月维护版本及更新版本	Not applicable
ML2/OVN L3 HA	完全支持-29 年 8 月 2018 维护支持版本	全面支持 2020 年 10 月 28 日
ML2/OVN DVR	完全支持-13 年 11 月 2018 年 11 月发布及更新版本	全面支持 2020 年 10 月 28 日
ML2/ODL	完全支持 2019 年 1 月发行版本及更新版本	Not applicable
Amphora active-standby	完全支持-28 October 2020 维护发行版本及更新版本	不支持
被终止的 HTTPS 负载均衡器	完全支持 2020 年 3 月 10 日及更新版本	不支持
Amphora备用池	技术预览只有 2019 年 4 月 30 日的发行版本及更新版本	Not applicable
UDP	全面支持 2020 年 10 月 28 日	全面支持 2020 年 10 月 28 日
类型(flavor)	技术预览 2020 年 10 月 28 日	Not applicable
使用 ACL 创建负载均衡器	技术预览 2020 年 10 月 28 日	Not applicable

[1]带有 OVS、元数据、DHCP、L3 和 Octavia 的网络节点(worker、健康监控、内务处理)。

其他资源

- 第 2.1 节 “负载均衡服务供应商驱动程序”

2.3. 负载平衡服务软件要求

Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)要求您配置以下核心 OpenStack 组件：

- 计算(nova)
- OpenStack Networking (neutron)
- 镜像(glance)
- Identity (keystone)
- RabbitMQ
- MySQL

2.4. UNDERCLOUD 的负载均衡服务先决条件

Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)有以下 RHOSP undercloud 的要求：

- 成功安装 undercloud。
- undercloud 上出现的负载均衡服务。
- 基于容器的 overcloud 部署计划。
- 在 Controller 节点上配置的负载均衡服务组件。

重要

如果要在现有 overcloud 部署中启用负载均衡服务，您必须准备 undercloud。如果不这样做，则 overcloud 安装会导致在没有负载均衡服务的情况下报告成功。要准备 undercloud，请参阅 *过渡到容器化服务* 指南。

2.5. 负载均衡服务实例的活动拓扑的基础知识

当您部署 Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)时，您可以决定是否在创建 Red Hat OpenStack Platform 时，负载均衡器会高度可用。如果要为用户提供选择，然后在 RHOSP 部署后创建一个负载均衡服务类别，用于创建高可用性负载均衡器和用于创建独立负载均衡器的类别。

默认情况下，amphora 提供者驱动程序是为单一负载均衡服务(amphora)实例拓扑配置，且对高可用性(HA)支持有限。但是，您可以在实现活跃和by拓扑时，使负载平衡服务实例具有高可用性。

在这个拓扑中，负载均衡服务为每个负载均衡器引导活跃和待机实例，并在每个负载均衡器之间维护会话持久性。如果活动实例变得不健康，实例将自动切换到待机实例，使其激活。负载均衡服务健康管理器会自动重建失败的实例。

其他资源

- 第 4.2 节 “为负载均衡服务实例启用主动拓扑”

2.6. 负载均衡服务部署后步骤

Red Hat OpenStack Platform (RHOSP)提供了一个工作流任务，以简化负载均衡服务(octavia)的部署后步骤。此工作流运行一组 Ansible playbook，以作为 overcloud 部署的最后一个阶段提供以下部署步骤：

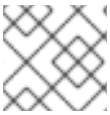
- 配置证书和密钥。
- 在 amphorae 和负载均衡服务控制器 worker 和健康管理器之间配置负载平衡管理网络。

Amphora 镜像

在预置备的服务器上，必须在部署负载均衡服务前在 undercloud 上安装 amphora 镜像：

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

在非预置备的服务器上，RHOSP director 会自动下载默认的 amphora 镜像，将其上传到 overcloud Image 服务(glance)，然后配置负载均衡服务来使用此 amphora 镜像。在堆栈更新或升级过程中，director 将此镜像更新为最新的 amphora 镜像。



注意

不支持自定义 amphora 镜像。

其他资源

- 第 4.1 节 “部署负载均衡服务”

第 3 章 保护负载均衡服务

要在 Red Hat OpenStack 负载均衡服务(octavia)的不同组件间安全通信，请使用 TLS 加密协议和公钥加密。

- 第 3.1 节 “负载均衡服务中的双向 TLS 身份验证”
- 第 3.2 节 “负载均衡服务的证书生命周期”
- 第 3.3 节 “配置负载平衡服务证书和密钥”

3.1. 负载均衡服务中的双向 TLS 身份验证

Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)的控制器进程通过 TLS 连接与负载均衡服务实例(amphorae)与负载均衡服务实例(amphorae)通信。负载均衡服务通过双向 TLS 身份验证验证两端是否受信任。

注意

这是完整的 TLS 握手过程的简化。有关 TLS 握手过程的更多信息，请参阅 TLS 1.3 RFC 8446。

双向 TLS 身份验证涉及两个阶段：*第一阶段*（如负载均衡服务 worker 进程）连接到负载均衡服务实例，实例为控制器提供其服务器证书。然后，控制器根据存储在控制器中的服务器证书颁发机构(CA)证书验证服务器证书。如果针对服务器 CA 证书验证了显示的证书，连接将进入阶段 2。

在两个阶段中，控制器将其客户端证书提供给负载平衡服务实例。然后，根据实例中存储的客户端 CA 证书验证证书。如果成功验证了这个证书，则 TLS 握手的其余部分将继续在控制器和负载均衡服务实例之间建立安全通信频道。

其他资源

- 第 3.3 节 “配置负载平衡服务证书和密钥”

3.2. 负载均衡服务的证书生命周期

Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)控制器使用服务器证书颁发机构证书和密钥来为每个负载均衡服务实例(amphora)唯一生成证书。

负载均衡服务内务处理控制器进程会在其接近过期日期时自动轮转这些服务器证书。

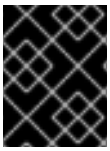
负载均衡服务控制器进程使用客户端证书。管理这些 TLS 证书的人类 operator 通常授予较长的过期周期，因为证书在云 control plane 上使用。

其他资源

- 第 3.3 节 “配置负载平衡服务证书和密钥”

3.3. 配置负载平衡服务证书和密钥

您可以配置 Red Hat OpenStack Platform (RHOSP) director 以生成证书和密钥，或者提供自己的内容。配置 director 以自动创建所需的私有证书颁发机构并发布所需证书。这些证书仅用于内部负载均衡服务(octavia)通信，而不向用户公开。



重要

RHOSP director 生成证书和密钥，并在它们过期前自动更新。如果使用自己的证书，您必须记得续订它们。

注意

RHOSP director 不支持从手动生成的证书切换到自动生成的证书。但是，您可以通过删除 **/var/lib/config-data/puppet-generated/octavia/etc/octavia/certs** 目录和更新 overcloud 中的 Controller 节点上的现有证书来强制重新创建证书。

如果您必须使用自己的证书和密钥，请完成以下步骤：

前提条件

- 读取和理解，"分离负载平衡服务默认设置"。（在"添加资源"中链接）

流程

1. 以 **stack** 用户身份登录 undercloud 主机。
2. 提供 undercloud 凭据文件：

```
$ source ~/stackrc
```

3. 创建 YAML 自定义环境文件。

   Example

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

4. 在 YAML 环境文件中，使用适合您的站点的值添加以下参数：

   - **OctaviaCaCert** ：
     Octavia 用来生成证书的 CA 的证书。
   - **OctaviaCaKey**:
     用于为生成的证书签名的私钥。
   - **OctaviaCaKeyPassphrase**:
     与以上私有 CA 密钥一起使用的密语。
   - **OctaviaClientCert** ：
     由 Octavia CA 为控制器发布的客户端证书和未加密密钥。
   - **Octavia 生成证书** ：
     指示 director 启用(true)或禁用(false)自动证书和密钥生成的布尔值。

     重要

     您必须将 **OctaviaGenerateCerts** 设置为 false。

     Example

```
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. 运行 **openstack overcloud deploy** 命令，并包含核心 heat 模板、环境文件和新的自定义环境文件。

重要

环境文件的顺序非常重要，因为后续环境文件中定义的参数和资源更为优先。

Example

```
$ 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
```

其他资源

- 第 4.3 节 “更改负载均衡服务默认设置”

- *高级 Overcloud 自定义指南中的* 环境文件 https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/13/html-single/advanced_overcloud_customization/index#sect-Environment_Files
- 在 *高级 Overcloud 自定义指南中* 包括 Overcloud 中的 环境文件

第 4 章 安装和配置负载均衡服务

当使用 RHOSP director 部署 Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)时，您可以决定其虚拟机实例(amphorae)具有高可用性。在您要对负载均衡服务进行配置更改时，也使用 director。

- 第 4.1 节 “部署负载均衡服务”
- 第 4.2 节 “为负载均衡服务实例启用主动拓扑”
- 第 4.3 节 “更改负载均衡服务默认设置”

4.1. 部署负载均衡服务

您可以使用 Red Hat OpenStack Platform (RHOSP) director 部署负载均衡服务(octavia)。director 使用编排服务(heat)模板，模板是适用于您环境的一组计划。undercloud 导入这些计划，并按照其说明创建负载平衡服务和 RHOSP 环境。

前提条件

- 确保您的环境可以访问 octavia 镜像。

流程

- 运行部署命令，包括核心 heat 模板、环境文件和 **octavia.yaml** heat 模板。

  Example

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

  注意

  director 会在堆栈更新或升级过程中将 amphorae 更新为最新的 amphora 镜像。

其他资源

- *Director 安装和使用* 指南中的 部署命令选项

4.2. 为负载均衡服务实例启用主动拓扑

在使用 Red Hat OpenStack Platform (RHOSP) director 实施 active-standby 拓扑时，您可以使负载均衡服务实例(amphorae)高度可用。director 使用编排服务(heat)模板，模板是适用于您环境的一组计划。undercloud 导入这些计划，并按照其说明创建负载平衡服务和 RHOSP 环境。

前提条件

- 确保为计算服务启用了反关联性。这是默认值。
- 最少三个 Compute 节点主机：
  - 两个计算节点主机，将 amphorae 放置到不同的主机上（计算反关联性）。

- 当出现问题时，第三个主机通过主动的负载平衡器成功失败。

流程

1. 以 **stack** 用户身份登录 undercloud 主机。
2. 提供 undercloud 凭据文件：

```
$ source ~/stackrc
```

3. 创建自定义 YAML 环境文件。

Example

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

4. 在自定义环境文件中，添加以下参数：

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

5. 运行部署命令，包括核心 heat 模板、环境文件和新的自定义环境文件。

重要

环境文件的顺序非常重要，因为后续环境文件中定义的参数和资源更为优先。

Example

```
$ 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
```

验证步骤

- 部署完成后，并创建了负载均衡器，运行以下命令：

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

如果您的负载均衡服务实例高可用性配置成功，您会看到两个实例(amphorae)的输出，且不会出现等于 **SINGLE 的角色**。

其他资源

- *高级 Overcloud 自定义指南中的* 环境文件 https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/13/html-single/advanced_overcloud_customization/index#sect-Environment_Files
- *在 高级 Overcloud 自定义指南中* 包括 Overcloud 中的 环境文件

4.3. 更改负载均衡服务默认设置

您可以使用 Red Hat OpenStack Platform (RHOSP) director 对负载均衡服务(octavia)进行配置更改。director 使用编排服务(heat)模板，模板是适用于您环境的一组计划。undercloud 导入这些计划，并按照其说明创建负载平衡服务和 RHOSP 环境。

前提条件

- 通过在 undercloud 上查询以下 文件，决定 director 已使用的 RHOSP 编排服务(heat)参数来部署负载平衡服务：

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

- 决定要修改的参数。
  以下是几个示例：
  - **OctaviaControlNetwork**
    用于负载均衡器管理网络的 neutron 网络的名称。
  - **OctaviaControlSubnetCidr**
    CIDR 表单中的 amphora 控制子网的子网。
  - **OctaviaMgmtPortDevName**
    用于 amphora 机器的 octavia worker/health-manager 间通信的 octavia 管理网络接口的名称。

流程

1. 以 **stack** 用户身份登录 undercloud 主机。
2. 提供 undercloud 凭据文件：

```
$ source ~/stackrc
```

3. 创建自定义 YAML 环境文件。

   Example

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

4. 您的环境文件必须包含关键字 **参数_defaults**。将参数值对放在 **parameter_defaults** 关键字后面。

   Example

```
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. 运行部署命令，包括核心 heat 模板、环境文件和新的自定义环境文件。

重要

环境文件的顺序非常重要，因为后续环境文件中定义的参数和资源更为优先。

Example

```
$ 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
```

其他资源

- *高级 Overcloud 自定义指南中的* 环境文件 https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/13/html-single/advanced_overcloud_customization/index#sect-Environment_Files
- 在 *高级 Overcloud 自定义指南中* 包括 Overcloud 中的 环境文件

第 5 章 配置负载均衡服务类型

负载均衡服务(octavia) 类别是您创建的供应商配置选项集合。当用户请求负载均衡器时，它们可以使用其中一个定义的类别指定负载均衡器。您可以为每个负载均衡供应商驱动程序定义一个类别，它会公开对应提供程序的唯一功能。

要创建新的负载均衡服务类别：

1. 决定您要在类别中配置的负载平衡提供程序的功能。
2. 使用您选择的类别功能创建 flavor 配置集。
3. 创建 类别。

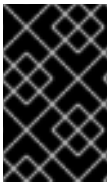
重要

本部分所含类别的主题在 Red Hat OpenStack Platform 13, 2020 年 10 月 28 日中 *技术预览*。有关技术预览功能的支持范围的更多信息，请参阅技术预览功能支持范围。

- 第 5.1 节 “列出负载均衡服务供应商功能”
- 第 5.2 节 “定义类别配置集”
- 第 5.3 节 “创建负载均衡服务类型”

5.1. 列出负载均衡服务供应商功能

您可以查看每个负载均衡服务(octavia)供应商驱动程序公开的功能列表。



重要

本部分描述的类别功能在 Red Hat OpenStack Platform 13、28 年 10 月 28 日及之后的 *技术预览* 中提供。有关技术预览功能的支持范围的更多信息，请参阅技术预览功能支持范围。

前提条件

- 您必须具有 OpenStack 管理员权限。

流程

1. 以 **stack** 用户身份登录 undercloud 主机。
2. 提供 undercloud 凭据文件：

```
$ source ~/stackrc
```

3. 列出每个驱动程序的功能：

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

将 **<provider** > 替换为供应商的名称或 UUID。

Example

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

命令输出列出了提供程序支持的所有功能。

输出示例

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

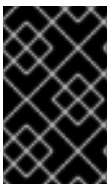
4. 记录您要包含在您要创建的类别中的功能名称。

其他资源

- 第 5.2 节 "定义类别配置集"
- *命令行界面参考中的*LoadBalancer 供应商功能列表

5.2. 定义类别配置集

负载均衡服务(octavia)类别配置集包含供应商驱动程序名称和功能列表。您可以使用类别配置集来创建用户指定的类别来创建负载均衡器。



重要

本部分描述的类别功能在 Red Hat OpenStack Platform 13、28 年 10 月 28 日及之后的 *技术预览* 中提供。有关技术预览功能的支持范围的更多信息，请参阅技术预览功能支持范围。

前提条件

- 您必须具有 OpenStack 管理员权限。
- 您必须知道哪些负载平衡供应商以及您要包含在类别配置集中的哪些功能。

流程

1. 以 **stack** 用户身份登录 undercloud 主机。
2. 提供 undercloud 凭据文件：

```
$ source ~/stackrc
```

3. 创建 flavor 配置集：

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

Example

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

输出示例

```
+---------------+--------------------------------------+
| Field         | Value                                |
+---------------+--------------------------------------+
| id            | 72b53ac2-b191-48eb-8f73-ed012caca23a |
| name          | amphora-single-profile               |
| provider_name | amphora                              |
| flavor_data   | {"loadbalancer_topology": "SINGLE"}  |
+---------------+--------------------------------------+
```

在本例中，为 amphora 提供者创建一个 flavor 配置集。在类别中指定此配置集时，用户使用该类别创建的负载均衡器是一个 amphora 负载均衡器。

验证步骤

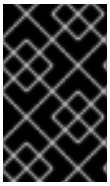
- 创建 flavor 配置集时，负载均衡服务使用提供程序验证类别值，以确保供应商支持您指定的功能。

其他资源

- 第 5.3 节 “创建负载均衡服务类型”
- 在 *命令行界面参考* 中创建 LoadBalancer 类别配置集

5.3. 创建负载均衡服务类型

您可以使用 flavor 配置集为负载均衡服务(octavia)创建面向用户的类别。您分配给该类别的名称是用户在创建负载均衡器时指定的值。



重要

本部分描述的类别功能在 Red Hat OpenStack Platform 13、28 年 10 月 28 日及之后的 *技术预览* 中提供。有关技术预览功能的支持范围的更多信息，请参阅技术预览功能支持范围。

前提条件

- 您必须具有 OpenStack 管理员权限。
- 您必须已创建了 flavor 配置集。

流程

1. 以 **stack** 用户身份登录 undercloud 主机。

2. 提供 undercloud 凭据文件：

```
$ source ~/stackrc
```

3. 创建类别：

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

提示

提供一个详细描述，方便用户了解您所提供的类别的功能。

Example

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

输出示例

```
+-------------------+--------------------------------------+
| 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.                |
+-------------------+--------------------------------------+
```

在本例中，定义了类别。当用户指定此类别时，它们会创建一个负载均衡器，它使用一个负载均衡服务实例(amphora)且不可用。

注意

禁用的类别仍对用户可见，但用户无法使用禁用的类别来创建负载均衡器。

其他资源

- 第 5.2 节 “定义类别配置集”
- 在 *命令行界面参考* 中创建 LoadBalancer 类别

第 6 章 监控负载均衡服务

要保持负载平衡操作，您可以使用负载均衡器管理网络，并创建、修改和删除负载平衡运行状况监视器。

6.1. 负载均衡管理网络

Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)通过项目网络（称为 *负载均衡器管理网络*）*监控负载均衡器*。运行负载均衡服务的主机必须具有接口才能连接到负载均衡器管理网络。支持的接口配置与 neutron Modular Layer 2 插件与 Open Virtual Network 机制驱动程序(ML2/OVN)或 Open vSwitch 机制驱动程序(ML2/OVS)一同使用。尚未测试使用带有其他机制驱动程序的接口。

在部署时创建的默认接口是默认集成网桥 **br-int** 上的内部 Open vSwitch (OVS)端口。您必须将这些接口与负载均衡器管理网络上分配的实际 Networking 服务(neutron)端口关联。

默认接口名为 **o-hm0**。它们通过负载平衡服务主机上的标准接口配置文件来定义。RHOSP director 自动配置网络服务端口和部署期间每个负载均衡服务主机的接口。端口信息和模板用于创建接口配置文件，包括：

- IP 网络地址信息，包括 IP 和子网掩码
- MTU 配置
- MAC 地址
- 网络服务端口 ID

在默认的 OVS 情形中，网络服务端口 ID 用于向 OVS 端口注册额外数据。网络服务将该接口识别为属于端口并配置 OVS，使它可以在负载平衡器管理网络上进行通信。

默认情况下，RHOSP 配置安全组和防火墙规则，它允许负载平衡服务控制器与 TCP 端口 9443 上的虚拟机实例(amphorae)进行通信，并允许来自 amphorae 的心跳消息到达 UDP 端口 5555 上的控制器。不同的机制驱动程序可能需要额外的或备用要求，以允许负载平衡服务和负载均衡器之间的通信。

6.2. 负载均衡服务实例监控

负载均衡服务(octavia)监控负载均衡实例(amphorae)，并在 amphorae 故障时启动故障切换和替换。每当发生故障转移时，负载均衡服务会在控制器的 **/var/log/containers/octavia** 中的对应健康状况管理器日志中记录故障转移。

使用日志分析功能监控故障转移趋势，以尽快解决问题。网络服务(neutron)连接问题、服务攻击和计算服务(nova)故障等问题往往导致负载均衡器的故障转移率更高。

6.3. 负载均衡服务池成员监控

负载均衡服务(octavia)使用底层负载均衡子系统中的健康信息来决定负载平衡池的成员的健康状况。健康信息流化到负载均衡服务数据库，并由状态树或其他 API 方法提供。对于关键应用程序，您必须定期轮询健康信息。

6.4. 负载均衡器置备状态监控

您可以监控负载平衡器的调配状态，并在置备状态是 **ERROR** 时发送警报。不要将警报配置为在应用程序对池进行常规更改并进入多个 **PENDING** 阶段时触发。

负载均衡器对象的置备状态反映了 control plane 无法联系并成功置备创建、更新和删除请求的功能。负载均衡器对象的操作状态报告负载均衡器的当前功能。

例如，负载均衡器的调配状态可能为 **ERROR**，但状态为 **ONLINE**。这可能是因为 Networking (neutron) 失败导致，因为最后请求到负载均衡器配置的更新无法成功完成。在本例中，负载均衡器继续通过负载均衡器处理流量，但可能还没有应用最新的配置更新。

6.5. 负载均衡器功能监控

您可以监控负载均衡器及其子对象的运行状态。

您还可以使用连接到负载均衡器监听程序的外部监控服务，并从云外监控它们。外部监控服务表示负载均衡服务(octavia)以外的失败可能会影响负载均衡器的功能，如路由器故障、网络连接问题等。

6.6. 关于负载均衡服务运行状况监控器

负载均衡服务(octavia)健康监控器是一个流程，它在每个后端成员服务器上定期进行健康检查来预检测故障服务器，并临时将它们从池中拉取。

如果运行状况监控器检测到失败的服务器，它将服务器从池中移除，并将成员标记为 **ERROR**。在更正了服务器并再次正常工作后，健康监控器会自动将成员的状态从 **ERROR** 改为 **ONLINE**，并恢复流量。

始终在生产负载均衡器中使用运行状况监视器。如果您没有运行状况监控器，则失败的服务器不会从池中移除。这可能会导致 Web 客户端的服务中断。

有多种健康监控器类型，如这里简略描述：

HTTP

默认情况下，探测应用服务器上的 / 路径。

HTTPS

完全类似于 HTTP 运行状况监视器，但使用 TLS 后端服务器。
如果服务器执行客户端证书验证，则 HAProxy 没有有效的证书。在这些情况下，TLS-HELLO 健康监控是备选的。

TLS-HELLO

确保后端服务器响应 SSLv3-client hello 消息。
TLS-HELLO 运行状况监控器不会检查任何其他健康指标，如状态代码或正文内容。

PING

将定期 ICMP ping 请求发送到后端服务器。
您必须将后端服务器配置为允许 PING，以便这些健康检查通过。

重要

只有成员可以访问并响应 ICMP echo 请求时，PING 运行状况监视器才会检查。PING 运行状况监视器不会检测实例上运行的应用程序是否健康。仅在 ICMP 回显请求为有效的健康检查的情况下，才使用 PING 运行状况监视器。

TCP

打开到后端服务器协议端口的 TCP 连接。
TCP 应用程序打开 TCP 连接，并在 TCP 握手后关闭连接，而不发送任何数据。

UDP-CONNECT

执行基本 UDP 端口连接。
如果在成员服务器上没有启用 Destination Unreachable (ICMP 类型 3)，或安全规则阻止，则 UDP-CONNECT 运行状况监视器可能无法正常工作。在这些情况下，成员服务器在实际停机时可能被标记为 **ONLINE** 操作状态。

6.7. 创建负载平衡服务运行状况监视器

使用负载均衡服务(octavia)健康监控器来避免用户的服务中断。运行状况监视器在每个后端服务器上运行定期健康检查，以预先检测失败的服务器，并临时从池中拉取服务器。

流程

1. 提供您的凭据文件。

   Example

   ```
   $ source ~/overcloudrc
   ```

2. 使用适合您的站点的参数值运行 **openstack loadbalancer healthmonitor create** 命令。

   - 所有运行状况监视器类型都需要以下可配置参数：

     <pool>

     要监控的后端成员服务器池的名称或 ID。

     --type

     运行状况监视器的类型。**HTTP**、**HTTPS**、**PING**、**SCTP**、**TCP**、**TLS-HELLO** 或 **UDP-CONNECT** 之一。

     --delay

     健康检查之间等待的秒数。

     --timeout

     等待任何给定健康检查完成的秒数。**超时** 必须始终小于 **延迟**。

--max-retries

后端服务器在被视为关闭前必须失败的健康检查数量。另外，必须再次考虑故障后端服务器的健康检查数量。

- 另外，HTTP 健康监控器类型还需要以下参数，默认为设置：

--url-path

应从后端服务器检索的 URL 的路径部分。默认为 /。

--http-method

用于检索 **url_path** 的 HTTP 方法。默认为 **GET**。

--expected-codes

表示正常健康检查的 HTTP 状态代码列表。默认值为 **200**。

Example

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

验证

- 运行 **openstack loadbalancer healthmonitor list** 命令，并验证您的运行状况监控器是否正在运行。

其他资源

- 在 *命令行界面参考* 中创建 LoadBalancer healthmonitor

6.8. 修改负载平衡服务运行状况监视器

当您要更改将探测发送到成员的时间间隔、连接超时间隔、请求的 HTTP 方法等时，您可以修改负载均衡服务(octavia)健康监控器的配置。

流程

1. 提供您的凭据文件。

   Example

   ```
   $ source ~/overcloudrc
   ```

2. 修改运行状况监视器(**my-health-monitor**)。
   在本例中，用户正在更改以秒为单位的时间，让健康监控器在向成员发送探测之间等待。

   Example

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

验证

- 运行 **openstack loadbalancer healthmonitor show** 命令以确认配置更改。

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

其他资源

- *命令行界面参考* 中设置的 LoadBalancer healthmonitor
- LoadBalancer healthmonitor 在 *命令行界面参考*中显示

6.9. 删除负载平衡服务运行状况监视器

您可以删除负载均衡服务(octavia)健康监控器。

提示

删除运行状况监控器的替代方法是使用 **openstack loadbalancer healthmonitor set --disable** 命令来禁用它。

流程

1. 提供您的凭据文件。

   Example

   ```
   $ source ~/overcloudrc
   ```

2. 删除运行状况监视器(**my-health-monitor**)。

   Example

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

验证

- 运行 **openstack loadbalancer healthmonitor list** 命令，以验证您删除的运行状况监控器不再存在。

其他资源

- *命令行界面参考中的*LoadBalancer healthmonitor delete

6.10. 负载平衡服务 HTTP 运行状况监视器的最佳实践

在 web 应用程序中编写生成健康检查的代码时，使用以下最佳实践：

- 健康监控器 **url-path** 不需要身份验证来加载。
- 默认情况下，健康监控器 **url-path** 返回 **HTTP 200 OK** 状态代码，以指示健康服务器，除非您指定了备用 **预期的代码**。
- 健康检查有足够的内部检查，以确保应用程序健康且不再运行。确保应用程序满足以下条件：
  - 任何所需的数据库或其他外部存储连接都已启动并在运行。

- 对于运行应用的服务器，可以接受负载。
- 您的站点不处于维护模式。
- 特定于应用程序的测试是可操作的。

- 由健康检查生成的页面的大小应该是小的：
  - 它以秒为单位返回。
  - 它不会在应用服务器上产生显著负载。
- 健康检查生成的页面不会被缓存，但运行健康检查的代码可能会引用缓存的数据。
  例如，您可能会发现使用 cron 运行更广泛的健康检查并将结果保存到磁盘中很有用。在健康监控器 **url-path** 生成页面的代码在它的测试中纳入此 cron 作业的结果。
- 由于负载均衡服务仅处理返回的 HTTP 状态代码，并且因为健康检查经常运行，因此您可以使用 **HEAD** 或 **OPTIONS** HTTP 方法跳过处理整个页面。

第 7 章 创建非安全 HTTP 负载均衡器

您可以为非安全 HTTP 网络流量创建以下负载均衡器：

- 第 7.1 节 “使用健康监控器创建 HTTP 负载均衡器”
- 第 7.2 节 “创建使用浮动 IP 的 HTTP 负载均衡器”
- 第 7.3 节 “使用会话持久性创建 HTTP 负载均衡器”

7.1. 使用健康监控器创建 HTTP 负载均衡器

对于与 Red Hat OpenStack Platform 网络服务(neutron)浮动 IP 不兼容的网络，请创建一个负载均衡器来管理非安全 HTTP 应用程序的网络流量。创建运行状况监视器，以确保您的后端成员仍然可用。

前提条件

- 专用子网，包含在 TCP 端口 80 上托管非安全 HTTP 应用程序的后端服务器。
- 专用子网上的后端服务器通过 URL 路径为健康检查配置。
- 您可以从互联网访问的共享外部（公共）子网。

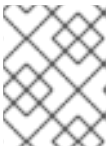
流程

1. 提供您的凭据文件。

   Example

   ```
   $ source ~/overcloudrc
   ```

2. 在公共子网(**public_subnet**)上创建负载均衡器(**lb1**)。

   注意

   括号中的值是本流程中示例命令中使用的示例值。将这些示例值替换为适合您的站点的值。

   Example

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

3. 验证负载均衡器的状态。

   Example

   ```
   $ openstack loadbalancer show lb1
   ```

4. 在继续下一步之前，请确保 **provisioning_status** 为 **ACTIVE**。

5. 在端口(80)上创建**侦听器（侦听器1**）。

   Example

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

6. 验证监听器的状态。

Example

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

在进入下一步之前，请确保状态是 **ACTIVE**。

7. 创建侦听器默认池(**池1**)。

Example

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

8. 在池(**pool1**)上创建运行状况监控器，它将连接到后端服务器并测试路径/。

Example

```
$ openstack loadbalancer healthmonitor create --delay 15 --max-retries 4 --timeout 10 --type
HTTP --url-path / pool1
```

9. 将专用子网(**private_subnet**)上的负载平衡器成员(**192.0.2.10 和 192.0.2.11)**添加到默认池。

Example

```
$ 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
```

验证

1. 查看并验证负载均衡器(lb1)设置：

Example

```
$ openstack loadbalancer show lb1
```

输出示例

```
+---------------------+--------------------------------------+
| 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. 当健康检查存在且正常工作时，您可以检查每个成员的状态。
一个正常工作的成员(**b85c807e-4d7c-4cbd-b725-5e8afddf80d2**)为其 **operating_status** 有一个 **ONLINE** 值。

Example

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

输出示例

```
+---------------------+--------------------------------------+
| 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                                |
+---------------------+--------------------------------------+
```

其他资源

- *命令行界面参考中的 LoadBalancer*

7.2. 创建使用浮动 IP 的 HTTP 负载均衡器

要管理非安全 HTTP 应用程序的网络流量，请使用依赖于浮动 IP 的虚拟 IP (VIP)创建一个负载均衡器。使用浮动 IP 的优点是您保留对分配的 IP 的控制，如果需要移动、销毁或重新创建负载均衡器，则需要这样做。最佳实践是创建运行状况监视器，以确保您的后端成员保持可用。

注意

浮动 IP 不适用于 IPv6 网络。

前提条件

- 专用子网，包含在 TCP 端口 80 上托管非安全 HTTP 应用程序的后端服务器。
- 后端服务器使用 URL 路径 / 的健康检查进行配置。
- 与负载均衡器 VIP 搭配使用的浮动 IP。
- 您可以从互联网访问的 Red Hat OpenStack Platform Networking 服务(neutron)共享的外部(public)子网，以用于浮动 IP。

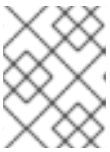
流程

1. 提供您的凭据文件。

   Example

   ```
   $ source ~/overcloudrc
   ```

2. 在专用子网(**private_subnet**)上创建负载均衡器(**lb1**)。

   注意

   括号中的值是本流程中示例命令中使用的示例值。将这些示例值替换为适合您的站点的值。

   Example

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

3. 注意 **load_balancer_vip_port_id** 的值，因为您必须在后续步骤中提供它。
4. 验证负载均衡器的状态。

   Example

   ```
   $ openstack loadbalancer show lb1
   ```

5. 在继续下一步之前，请确保 **provisioning_status** 为 **ACTIVE**。
6. 在端口(80)上创建侦听器（侦听器**1**）。

   Example

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

7. 创建侦听器默认池(**池1**)。

   Example

Example

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

8. 在池(**pool1**)上创建运行状况监控器，它将连接到后端服务器并测试路径/。

   Example

   ```
   $ openstack loadbalancer healthmonitor create --delay 15 --max-retries 4 --timeout 10 --type HTTP --url-path / pool1
   ```

9. 将专用子网上的负载平衡器成员(**192.0.2.10 和 192.0.2.11)**添加到默认池。

   Example

   ```
   $ 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. 在共享外部子网(**公共**)上创建一个浮动 IP 地址。

    Example

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

11. 请注意 **floating_ip_address** 的值，因为您必须在后续步骤中提供。

12. 将此浮动 IP (**203.0.113.0**)与负载均衡器 **vip_port_id** (**69a85edd-5b1c-458f-96f2-b4552b15b8e6**)关联。

    Example

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

验证

1. 使用浮动 IP (**203.0.113.**0)验证负载均衡器之间的 HTTP 流量流。

   Example

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

   输出示例

   ```
   * 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. 当健康检查存在且正常工作时，您可以检查每个成员的状态。
一个正常工作的成员(**b85c807e-4d7c-4cbd-b725-5e8afddf80d2**)为其 **operating_status** 有一个 **ONLINE** 值。

Example

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

输出示例

```
+---------------------+--------------------------------------+
| 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                                |
+---------------------+--------------------------------------+
```

其他资源

- *命令行界面参考中的 LoadBalancer*
- *命令行界面参考中的浮动信息*

7.3. 使用会话持久性创建 HTTP 负载均衡器

要管理非安全 HTTP 应用程序的网络流量，您可以创建跟踪会话持久性的负载均衡器。这样做可确保当请求存在时，负载均衡器会将来自同一客户端的后续请求定向到同一个后端服务器。会话持久性通过节省时间和内存优化负载平衡。

前提条件

- 专用子网，包含在 TCP 端口 80 上托管非安全 HTTP 应用程序的后端服务器。
- 后端服务器使用 URL 路径 / 的健康检查进行配置。

- 您可以从互联网访问的共享外部（公共）子网。
- 您进行负载平衡的网络流量启用了 Cookie 的非安全 Web 应用。

流程

1. 提供您的凭据文件。

   Example

   ```
   $ source ~/overcloudrc
   ```

2. 在公共子网(**public_subnet**)上创建负载均衡器(**lb1**)。

   注意

   括号中的值是本流程中示例命令中使用的示例值。将这些示例值替换为适合您的站点的值。

   Example

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

3. 验证负载均衡器的状态。

   Example

   ```
   $ openstack loadbalancer show lb1
   ```

4. 在继续下一步之前，请确保 **provisioning_status** 为 **ACTIVE**。

5. 在端口(80)上创建**侦听器（侦听器1**）。

   Example

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

6. 创建侦听器的默认池(**pool1**)，用于定义 Cookie (**PHPSESSIONID**)上的会话持久性。

   Example

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

7. 在池(**pool1**)上创建运行状况监控器，它将连接到后端服务器并测试路径/。

   Example

   ```
   $ openstack loadbalancer healthmonitor create --delay 15 --max-retries 4 --timeout 10 --type
   HTTP --url-path / pool1
   ```

8. 将专用子网(**private_subnet**)上的负载平衡器成员(**192.0.2.10 和 192.0.2.11)**添加到默认池。

Example

```
$ 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
```

验证

1. 查看并验证负载均衡器(lb1)设置：

Example

```
$ openstack loadbalancer show lb1
```

输出示例

```
+---------------------+--------------------------------------+
| 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. 当健康检查存在且正常工作时，您可以检查每个成员的状态。
一个正常工作的成员(**b85c807e-4d7c-4cbd-b725-5e8afddf80d2**)为其 **operating_status** 有一个 **ONLINE** 值。

Example

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

输出示例

```
+---------------------+--------------------------------------+
```

```
| 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                                |
+---------------------+--------------------------------------+
```

其他资源

- *命令行界面参考中的 LoadBalancer*

第 8 章 创建安全 HTTP 负载均衡器

您可以创建各种类型的负载均衡器来管理安全的 HTTP (HTTPS)网络流量。

8.1. 关于非最终 HTTPS 负载均衡器

非最终 HTTPS 负载均衡器的行为与通用 TCP 负载均衡器类似：负载均衡器将 web 客户端的原始 TCP 流量转发到使用 web 客户端终止 HTTPS 连接的后端服务器。虽然非最终的 HTTPS 负载均衡器不支持第 7 层功能等高级负载均衡器功能，但它们通过管理证书和密钥本身来降低负载均衡器资源利用率。

8.2. 创建非结尾的 HTTPS 负载均衡器

如果您的应用程序需要 HTTPS 流量在后端成员服务器上终止，通常通过 *HTTPS 传递*，您可以在负载均衡器监听器中使用 HTTPS 协议。

前提条件

- 私有子网，包含托管使用 TLS 加密 web 应用程序在 TCP 端口 443 上配置的 HTTPS 应用程序的后端服务器。
- 后端服务器使用 URL 路径 / 的健康检查进行配置。
- 您可以从互联网访问的共享外部（公共）子网。

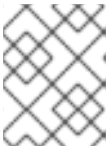
流程

1. 提供您的凭据文件。

   Example

   ```
   $ source ~/overcloudrc
   ```

2. 在公共子网(**public_subnet**)上创建负载均衡器(**lb1**)：

   注意

   括号中的值是本流程中示例命令中使用的示例值。将这些示例值替换为适合您的站点的值。

   Example

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

3. 监控负载均衡器的状态。

   Example

   ```
   $ openstack loadbalancer show lb1
   ```

4. 在继续下一步之前，请确保 **provisioning_status** 为 **ACTIVE**。

5. 在端口(**443**)上创建侦听器（侦听器**1**）。

   Example

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

6. 创建侦听器默认池(**池1**)。

   Example

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

7. 在池(**pool1**)上创建运行状况监控器，它将连接到后端服务器并测试路径/。

   Example

   ```
   $ openstack loadbalancer healthmonitor create --delay 15 --max-retries 4 --timeout 10 --type
   TLS-HELLO --url-path / pool1
   ```

8. 将专用子网(**private_subnet**)上的负载平衡器成员(**192.0.2.10 和 192.0.2.11)**添加到默认池。

   Example

   ```
   $ 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
   ```

验证

1. 查看并验证负载均衡器(**lb1**)设置。

   Example

   ```
   $ openstack loadbalancer show lb1
   ```

   输出示例

   ```
   +---------------------+--------------------------------------+
   | 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. 当健康检查存在且正常工作时，您可以检查每个成员的状态。
一个正常工作的成员(**b85c807e-4d7c-4cbd-b725-5e8afddf80d2**)为其 **operating_status** 有一个 **ONLINE** 值。

Example

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

输出示例

```
+---------------------+--------------------------------------+
| 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                                |
+---------------------+--------------------------------------+
```

其他资源

- 使用 OpenStack Key Manager 指南管理机密。

- *命令行界面参考中的 LoadBalancer*

8.3. 关于 TLS 终止的 HTTPS 负载均衡器

当实现 TLS 终止的 HTTPS 负载均衡器时，Web 客户端通过传输层安全(TLS)协议与负载均衡器通信。负载平衡器终止 TLS 会话，并将解密的请求转发到后端服务器。当您在负载均衡器上终止 TLS 会话时，您可以将 CPU 密集型加密操作卸载到负载均衡器，并允许负载均衡器使用第 7 层检查等高级功能。

8.4. 创建 TLS 终止的 HTTPS 负载均衡器

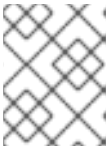
使用 TLS 终止的 HTTPS 负载均衡器时，您可以将 CPU 密集型加密操作卸载到负载均衡器，并允许负载均衡器使用第 7 层检查等高级功能。最佳实践是创建运行状况监视器，以确保您的后端成员保持可用。

前提条件

- 专用子网，包含在 TCP 端口 80 上托管非安全 HTTP 应用程序的后端服务器。
- 后端服务器使用 URL 路径 / 的健康检查进行配置。
- 您可以从互联网访问的共享外部（公共）子网。
- TLS 公钥加密配置具有以下特征：
  - TLS 证书、密钥和中间证书链是从分配给负载均衡器 VIP 地址的 DNS 名称的外部证书颁发机构(CA)获取，例如 **www.example.com**。
  - 证书、密钥和中间证书链位于当前目录中的单独文件中。
  - 密钥和证书是 PEM 编码的。
  - 密钥没有使用密码短语加密。
  - intermediate 证书链包含多个 PEM 编码并串联在一起的证书。
- 您必须将负载均衡服务(octavia)配置为使用 Key Manager 服务(barbican)。如需更多信息，*请参阅使用 OpenStack Key Manager 管理机密指南*。

流程

1. 将密钥(**server.key**)、证书(**server.crt**)和中间证书链(**ca-chain.crt**)合并到单个 PKCS12 文件中(**server.p12**)。



注意

括号中的值是本流程中示例命令中使用的示例值。将这些示例值替换为适合您的站点的值。

Example

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

注意

如果您密码保护 PKCS12 文件，则以下步骤无法正常工作。

2. 提供您的凭据文件。

Example

```
$ source ~/overcloudrc
```

3. 使用 Key Manager 服务为 PKCS12 文件创建 secret 资源(**tls_secret1**)。

Example

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

4. 在公共子网**public_subnet**上创建负载平衡器(**lb1**)。

Example

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

5. 监控负载均衡器的状态。

Example

```
$ openstack loadbalancer show lb1
```

6. 在继续下一步之前，请确保 **provisioning_status** 为 **ACTIVE**。

7. 创建 **TERMINATED_HTTPS** 侦听器（侦听器**1**），并将 secret 资源引用为监听程序的默认 TLS 容器。

Example

```
$ 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. 创建池(**pool1**)，并使其成为侦听器的默认池。

Example

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

9. 在池(**pool1**)上创建运行状况监控器，它将连接到后端服务器并测试路径/。

Example

```
$ openstack loadbalancer healthmonitor create --delay 15 --max-retries 4 --timeout 10 --type
HTTP --url-path / pool1
```

10. 将专用子网(**private_subnet**)上的非安全 HTTP 后端服务器(**192.0.2.10 和 192.0.2.11)**添加到池。

Example

```
$ 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
```

验证

1. 查看并验证负载均衡器(**lb1**)设置。

Example

```
$ openstack loadbalancer show lb1
```

输出示例

```
+---------------------+--------------------------------------+
| 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. 当健康检查存在且正常工作时，您可以检查每个成员的状态。
一个正常工作的成员(**b85c807e-4d7c-4cbd-b725-5e8afddf80d2**)为其 **operating_status** 有一个 **ONLINE** 值。

Example

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

输出示例

```
+---------------------+--------------------------------------+
| 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                                |
+---------------------+--------------------------------------+
```

其他资源

- 使用 OpenStack Key Manager 指南管理机密。
- *命令行界面参考中的 LoadBalancer*

8.5. 使用 SNI 创建 TLS 终止的 HTTPS 负载均衡器

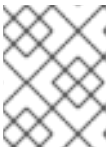
对于使用 Server Name Indication (SNI)技术的 TLSterminated HTTPS 负载均衡器，单个侦听器可以包含多个 TLS 证书，并启用负载均衡器来知道在使用共享 IP 时存在的证书。最佳实践是创建运行状况监视器，以确保您的后端成员保持可用。

前提条件

- 专用子网，包含在 TCP 端口 80 上托管非安全 HTTP 应用程序的后端服务器。
- 后端服务器使用 URL 路径 / 的健康检查进行配置。
- 您可以从互联网访问的共享外部（公共）子网。
- TLS 公钥加密配置具有以下特征：
  - 从分配给负载均衡器 VIP 地址的 DNS 名称从外部证书颁发机构(CA)获取多个 TLS 证书、密钥和中间证书链，例如 **www.example.com** 和 **www2.example.com**。
  - 密钥和证书经过 PEM 编码。
  - 密钥没有使用密码短语加密。
- 您必须将负载均衡服务(octavia)配置为使用 Key Manager 服务(barbican)。如需更多信息，*请参阅使用 OpenStack Key Manager 管理机密指南*。

流程

1. 对于 SNI 列表中的每个 TLS 证书，请将密钥(**server.key**)、证书(**server.crt**)和中间证书链(**ca-chain.crt**)组合为一个 PKCS12 文件(**server.p12**)。
在这个示例中，为每个证书创建两个 PKCS12 文件(**server.p12** 和 **server2.p12**)一(**www.example.com** 和 **www2.example.com)。**



注意

括号中的值是本流程中示例命令中使用的示例值。将这些示例值替换为适合您的站点的值。

```
$ 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. 提供您的凭据文件。

Example

```
$ source ~/overcloudrc
```

3. 使用 Key Manager 服务为 PKCS12 文件创建 secret 资源(**tls_secret1 和 tls_secret2)。**

```
$ 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. 在公共子网**public_subnet**上创建负载平衡器(**lb1**)。

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

5. 监控负载均衡器的状态。

Example

```
$ openstack loadbalancer show lb1
```

6. 在继续下一步之前，请确保 **provisioning_status** 为 **ACTIVE**。

7. 创建 TERMINATED_HTTPS**侦听器（侦听器1**），并使用 SNI 来引用机密资源。
（参考 **tls_secret1** 作为监听程序的默认 TLS 容器。）

```
$ 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. 创建池(**pool1**)，并使其成为侦听器的默认池。

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

9. 在池(**pool1**)上创建运行状况监控器，它将连接到后端服务器并测试路径/。

Example

```
$ openstack loadbalancer healthmonitor create --delay 15 --max-retries 4 --timeout 10 --type
HTTP --url-path / pool1
```

10. 将专用子网(**private_subnet**)上的非安全 HTTP 后端服务器(**192.0.2.10 和 192.0.2.11)**添加到池。

```
$ 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
```

验证

1. 查看并验证负载均衡器(**lb1**)设置。

Example

```
$ openstack loadbalancer show lb1
```

输出示例

```
+---------------------+--------------------------------------+
| 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. 当健康检查存在且正常工作时，您可以检查每个成员的状态。

一个正常工作的成员(**b85c807e-4d7c-4cbd-b725-5e8afddf80d2**)为其 **operating_status** 有一个 **ONLINE** 值。

Example

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

输出示例

```
+---------------------+--------------------------------------+
| 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                                |
+---------------------+--------------------------------------+
```

其他资源

- 使用 OpenStack Key Manager 指南管理机密。
- *命令行界面参考中的 LoadBalancer*

8.6. 在同一 IP 和后端中创建 HTTP 和 TLS 终止的 HTTPS 负载均衡

您可以在同一负载均衡器上配置非安全监听程序和 TLS 侦听器，当您要响应具有相同内容的 web 客户端时，无论客户端是否连接到安全或不安全 HTTP 协议时。最佳实践是创建运行状况监视器，以确保您的后端成员保持可用。

前提条件

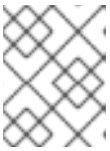
- 专用子网，包含在 TCP 端口 80 上托管非安全 HTTP 应用程序的后端服务器。
- 后端服务器使用 URL 路径 / 的健康检查进行配置。
- 您可以从互联网访问的共享外部（公共）子网。
- TLS 公钥加密配置具有以下特征：
  - TLS 证书、密钥和可选中间证书链是从分配给负载均衡器 VIP 地址的 DNS 名称的外部证书颁发机构(CA)获取（例如 www.example.com）。
  - 证书、密钥和中间证书链位于当前目录中的单独文件中。

- 密钥和证书是 PEM 编码的。
- 密钥没有使用密码短语加密。
- intermediate 证书链包含多个 PEM 编码并串联在一起的证书。

- 您已将负载均衡服务(octavia)配置为使用 Key Manager 服务(barbican)。如需更多信息，*请参阅使用 OpenStack Key Manager 管理机密指南*。
- 非安全 HTTP 侦听器配置有与 HTTPS TLS 终止负载均衡器相同的池。

流程

1. 将密钥(**server.key**)、证书(**server.crt**)和中间证书链(**ca-chain.crt**)合并到单个 PKCS12 文件中(**server.p12**)。



注意

括号中的值是本流程中示例命令中使用的示例值。将这些示例值替换为适合您的站点的值。

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

2. 提供您的凭据文件。

Example

```
$ source ~/overcloudrc
```

3. 使用 Key Manager 服务为 PKCS12 文件创建 secret 资源(**tls_secret1**)。

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

4. 在公共子网**public_subnet**上创建负载平衡器(**lb1**)。

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

5. 监控负载均衡器的状态。

Example

```
$ openstack loadbalancer show lb1
```

6. 在继续下一步之前，请确保 **provisioning_status** 为 **ACTIVE**。

7. 创建 TERMINATED_HTTPS**侦听器（侦听器1）**，并将 secret 资源引用为监听程序的默认 TLS 容器。

```
$ 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. 创建池(**pool1**)，并使其成为侦听器的默认池。

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

9. 在池(**pool1**)上创建运行状况监控器，它将连接到后端服务器并测试路径/：

Example

```
$ openstack loadbalancer healthmonitor create --delay 15 --max-retries 4 --timeout 10 --type
HTTP --url-path / pool1
```

10. 将专用子网(**private_subnet**)上的非安全 HTTP 后端服务器(**192.0.2.10 和 192.0.2.11)**添加到池。

```
$ 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. 创建非安全、HTTP侦**听器（侦听器2**），并使它的默认池与安全监听程序相同。

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

验证

1. 查看并验证负载均衡器(**lb1**)设置。

Example

```
$ openstack loadbalancer show lb1
```

输出示例

```
+---------------------+--------------------------------------+
| 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. 当健康检查存在且正常工作时，您可以检查每个成员的状态。
一个正常工作的成员(**b85c807e-4d7c-4cbd-b725-5e8afddf80d2**)为其 **operating_status** 有一个 **ONLINE** 值。

Example

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

输出示例

```
+---------------------+--------------------------------------+
| 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                                |
+---------------------+--------------------------------------+
```

其他资源

- 使用 OpenStack Key Manager 指南管理机密。
- *命令行界面参考中的 LoadBalancer*

第 9 章 创建其他类型的负载均衡器

您可以使用负载均衡服务(octavia)创建与您要管理的非 HTTP 网络流量类型的负载均衡器类型。

9.1. 创建 TCP 负载均衡器

当需要管理非 HTTP、基于 TCP 的服务和应用程序的网络流量时，您可以创建一个负载均衡器。最佳实践是创建运行状况监视器，以确保您的后端成员保持可用。

前提条件

- 专用子网，包含在特定 TCP 端口上托管自定义应用的后端服务器。
- 后端服务器使用 URL 路径 / 的健康检查进行配置。
- 您可以从互联网访问的共享外部（公共）子网。

流程

1. 提供您的凭据文件。

   Example

   ```
   $ source ~/overcloudrc
   ```

2. 在公共子网**public_subnet**上创建负载平衡器(**lb1**)。

   注意

   括号中的值是本流程中示例命令中使用的示例值。将这些示例值替换为适合您的站点的值。

   Example

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

3. 验证负载均衡器的状态。

   Example

   ```
   $ openstack loadbalancer show lb1
   ```

4. 在继续下一步之前，请确保 **provisioning_status** 为 **ACTIVE**。

5. 在配置了自定义应用程序的指定端口(**23456**)上创建 **TCP 侦听器（监听程序1**）。

Example

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

6. 创建池(**pool1**)，并使其成为侦听器的默认池。

Example

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

7. 在池(**pool1**)上创建运行状况监控器，它将连接到后端服务器并探测 TCP 服务端口。

Example

```
$ openstack loadbalancer healthmonitor create --delay 15 --max-retries 4 --timeout 10 --type
TCP pool1
```

8. 将专用子网(**private_subnet**)上的后端服务器(**192.0.2.10 和 192.0.2.11)**添加到池中。

Example

```
$ 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
```

验证

1. 查看并验证负载均衡器(**lb1**)设置。

Example

```
$ openstack loadbalancer show lb1
```

输出示例

```
+---------------------+--------------------------------------+
| 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. 当健康检查存在且正常工作时，您可以检查每个成员的状态。使用以下命令获取成员 ID：

Example

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

一个正常工作的成员(**b85c807e-4d7c-4cbd-b725-5e8afddf80d2**)为其 **operating_status** 有一个 **ONLINE** 值。

Example

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

输出示例

```
+---------------------+--------------------------------------+
| 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                                |
+---------------------+--------------------------------------+
```

其他资源

- *命令行界面 参考中的 LoadBalancer*

9.2. 使用健康监控器创建 UDP 负载均衡器

当您需要管理 UDP 端口上的网络流量时，您可以创建负载均衡器。最佳实践是创建运行状况监视器，以确保您的后端成员保持可用。

前提条件

- 包含托管配置为使用 UDP 端口的一个或多个应用的后端服务器。
- 您可以从互联网访问的共享外部（公共）子网。
- 后端服务器配置有 UDP 健康检查。
- 没有安全规则阻止 ICMP Destination Unreachable 消息(ICMP 类型 3)。

流程

1. 提供您的凭据文件。

   Example

   ```
   $ source ~/overcloudrc
   ```

2. 在专用子网(**private_subnet**)上创建负载均衡器(**lb1**)。

   注意

   括号中的值是本流程中示例命令中使用的示例值。将这些示例值替换为适合您的站点的值。

   Example

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

3. 验证负载均衡器的状态。

   Example

   ```
   $ openstack loadbalancer show lb1
   ```

4. 在继续下一步之前，请确保 **provisioning_status** 为 **ACTIVE**。

5. 在端口(**1234)上创建侦听器**（监听程序1）。

   Example

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

6. **创建侦听器默认池(池1)。**

Example

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

7. **在池(pool1)上创建运行状况监控器，它使用 UDP (UDP-CONNECT)连接到后端服务器。**

Example

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

8. **将专用子网(private_subnet)上的负载平衡器成员(192.0.2.10 和 192.0.2.11)添加到默认池。**

Example

```
$ 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
```

验证

1. **查看并验证负载均衡器(lb1)设置。**

Example

Example

```
$ openstack loadbalancer show lb1
```

输出示例

```
+---------------------+--------------------------------------+
| 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. **当健康检查存在且正常工作时，您可以检查每个成员的状态。使用以下命令获取成员 ID：**

Example

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

一个正常工作的成员(b85c807e-4d7c-4cbd-b725-5e8afddf80d2)为其 operating_status 有一个 ONLINE 值。

Example

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

输出示例

```
+---------------------+--------------------------------------+
| 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                                |
+---------------------+--------------------------------------+
```

其他资源

- *命令行界面 参考中的 LoadBalancer*

9.3. 创建 QOS 定义的负载均衡器

您可以将 Red Hat OpenStack Platform (RHOSP) Networking 服务(neutron)服务质量(QoS)策略应

用到使用负载均衡器的虚拟 IP 地址(VIP)。这样，您可以使用 QoS 策略来限制负载均衡器可以管理的进入或传出流量。最佳实践是创建运行状况监视器，以确保您的后端成员保持可用。

前提条件

- 专用子网，其中包含在 TCP 端口 80 上配置有 HTTP 应用程序的后端服务器。
- 后端服务器使用 URL 路径 / 的健康检查进行配置。
- 您可以从互联网访问的共享外部（公共）子网。
- 包含为 RHOSP 网络服务创建的带宽限制规则的 QoS 策略。

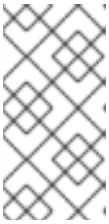
流程

1. 提供您的凭据文件。

   Example

   ```
   $ source ~/overcloudrc
   ```

2. 创建网络带宽 QoS 策略(qos_policy_bandwidth)，最大 1024 kbps 和最大突发速率 1024 kb。

   注意

   括号中的值是本流程中示例命令中使用的示例值。将这些示例值替换为适合您的站点的值。

   Example

```
$ 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. **使用 QoS 策略(qos-policy-bandwidth)在公共子网(public_subnet)上创建负载均衡器(lb1)。**

Example

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

4. **验证负载均衡器的状态。**

Example

```
$ openstack loadbalancer show lb1
```

5. **在继续下一步之前，请确保 provisioning_status 为 ACTIVE。**

6. **在端口(80)上创建侦听器（侦听器1）。**

Example

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

7. **创建侦听器默认池(池1)。**

Example

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

8. **在池中创建一个连接到后端服务器并测试路径/的运行状况监控器。**

Example

```
$ openstack loadbalancer healthmonitor create --delay 15 --max-retries 4 --timeout 10 --type
HTTP --url-path / pool1
```

9. **将专用子网(private_subnet)上的负载平衡器成员(192.0.2.10 和 192.0.2.11)添加到默认池。**

Example

```
$ 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
```

验证

1. **查看并验证监听器(侦听器1)设置。**

Example

```
$ openstack loadbalancer list
```

输出示例

```
+--------------------+--------------------------------------+
| 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 |
+--------------------+--------------------------------------+
```

在本例中，vip_qos_policy_id 中包含策略 ID。

其他资源

- *命令行界面 参考中的 LoadBalancer*

9.4. 使用访问控制列表创建负载均衡器

您可以创建一个访问控制列表(ACL)，将进入监听程序的流量限制为一组允许的源 IP 地址。任何其它传入的流量都会被拒绝。最佳实践是创建运行状况监视器，以确保您的后端成员保持可用。

前提条件

- 专用子网，其中包含在 TCP 端口 80 上配置自定义应用的后端服务器。
- 后端服务器使用 URL 路径 / 的健康检查进行配置。
- 您可以从互联网访问的共享外部（公共）子网。

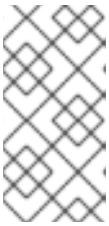
流程

1. 提供您的凭据文件。

Example

```
$ source ~/overcloudrc
```

2. 在公共子网public_subnet上创建负载平衡器(lb1)。



注意

括号中的值是本流程中示例命令中使用的示例值。将这些示例值替换为适合您的站点的值。

Example

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

-

3. 验证负载均衡器的状态。

Example

```
$ openstack loadbalancer show lb1
```

4. 在继续下一步之前，请确保 **provisioning_status** 为 **ACTIVE**。

5. 使用允许的 **CIDR (192.0.2.0/24** 和 **198.51.100.0/24)**创建侦听器（侦听器**1**）。

Example

```
$ 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. 创建侦听器默认池**(**池**1)**。

Example

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

7. **在池中创建一个连接到后端服务器并测试路径/的运行状况监控器。**

Example

```
$ openstack loadbalancer healthmonitor create --delay 15 --max-retries 4 --timeout 10 --type
HTTP --url-path / pool1
```

8. **将专用子网(private_subnet)上的负载平衡器成员(192.0.2.10 和 192.0.2.11)添加到默认池。**

Example

```
$ 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
```

验证

1. **查看和验证侦听器(侦听器1)设置。**

Example

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

输出示例

```
+----------------------------+--------------------------------------+
| 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                      |
+----------------------------+--------------------------------------+
```

在本例中，allow _cidrs 被设置为只允许来自 192.0.2.0/24 和 198.51.100.0/24 的流量。

2. **要验证负载均衡器是否安全，请确保来自 CIDR 不在 allowed_cidrs 列表中的客户端的请求，请求不会成功。**

输出示例

```
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
```

其他资源

- *命令行界面 参考中的 LoadBalancer*

9.5. 创建 OVN 负载均衡器

您可以使用 Red Hat OpenStack Platform (RHOSP)客户端创建一个负载均衡器，用于管理 RHOSP 部署中的网络流量。RHOSP 负载均衡服务支持使用 Open Virtual Network 机制驱动程序(ML2/OVN)的 neutron Modular Layer 2 插件。

前提条件

- ML2/OVN 供应商驱动程序必须部署。

重要

OVN 提供程序只支持第 4 层 TCP 和 UDP 网络流量和 SOURCE_IP_PORT 负载均衡器算法。OVN 供应商不支持健康监控。

- 专用子网，包含在特定 TCP 端口上托管自定义应用的后端服务器。
- 您可以从互联网访问的共享外部（公共）子网。

流程

1. 提供您的凭据文件。

Example

```
$ source ~/overcloudrc
```

2. 使用 **--provider ovn** 参数在专用子网(private_subnet)上创建负载均衡器(lb1)。

注意

括号中的值是本流程中示例命令中使用的示例值。将这些示例值替换为适合您的站点的值。

Example

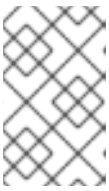
```
$ openstack loadbalancer create --name lb1 --provider ovn --vip-subnet-id private_subnet
```

3. 验证负载均衡器的状态。

```
$ openstack loadbalancer show lb1
```

4. 在继续下一步之前，请确保 **provisioning_status** 为 **ACTIVE**。

5. 创建使用配置了自定义应用的指定端口(tcp)的侦听器(listener1)。



注意

OVN 供应商只支持第 **4** 层 **TCP** 和 **UDP** 网络流量。

Example

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

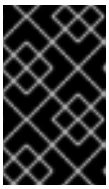
6. 创建侦听器默认池(池1)。

注意

OVN 唯一支持的负载平衡算法是 SOURCE_IP_PORT。

Example

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



重要

OVN 不支持健康检查功能进行负载平衡。

7. 将专用子网(private_subnet)上的后端服务器(192.0.2.10 和 192.0.2.11)添加到池中。

Example

```
$ 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
```

验证

1. 查看并验证负载均衡器(lb1)设置。

Example

```
$ openstack loadbalancer show lb1
```

输出示例

```
+---------------------+--------------------------------------+
| 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. 运行 **openstack loadbalancer 监听程序 show** 命令来查看监听程序详情。

Example

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

输出示例

```
+---------------------------+--------------------------------------+
| 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. 运行 **openstack loadbalancer pool show** 命令来查看池(pool1)和 load-balancer 成员。

Example

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

-

输出示例

```
+---------------------+--------------------------------------+
| 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                                |
+---------------------+--------------------------------------+
```

其他资源

- *命令行界面 参考中的 LoadBalancer*

第 10 章 实施第 7 层负载均衡

您可以使用带有第 7 层策略的 Red Hat OpenStack Platform 负载均衡服务(octavia)将 HTTP 请求重定向到特定的应用程序服务器池，以满足您的业务需求。

10.1. 关于第 7 层负载均衡

第 7 层(L7)负载平衡）从 Open Systems Interconnection (OSI)模型获取其名称，这表示负载均衡器根据第 7 层（应用程序）数据将请求分发到后端应用程序服务器池。以下是所有代表 L7 负载平衡的不同术语： *请求切换*、*应用程序负载平衡*，以及 *基于内容的路由*、*切换* 或 *平衡*。Red Hat OpenStack Platform 负载均衡服务(octavia)为 L7 负载均衡提供可靠的支持。

注意

您无法使用 UDP 负载均衡器创建 L7 策略和规则。

L7 负载平衡器由一个侦听器组成，后者代表多个后端池接受请求，并根据使用应用程序数据的策略分发这些请求，以确定哪个池服务任何给定请求。这允许对应用程序基础架构进行特别调整和优化，以提供特定类型的内容。例如，您可以调整一组后端服务器（池）来仅提供镜像；另一个用于执行 PHP 和 ASP 等服务器端脚本语言；另一个用于静态内容，如 HTML、CSS 和 JavaScript。

与较低级别的负载平衡不同，L7 负载平衡并不要求负载平衡服务背后的所有池都有相同的内容。L7 负载平衡器可以根据 URI、主机、HTTP 标头和其他数据直接请求。

10.2. 负载均衡服务中的第 7 层负载均衡

虽然您可以为任何明确的 L7 应用程序接口实施第 7 层(L7)负载均衡服务(octavia)的 L7 功能，但 Red Hat OpenStack Platform 负载均衡服务(octavia)仅适用于 HTTP 和 TERMINATED_HTTPS 协议及其语义。

Neutron LBaaS 和负载平衡服务将 L7 规则和策略用于 L7 负载平衡的逻辑。L7 规则是评估为 true 或 false 的单一简单逻辑测试。L7 策略是 L7 规则的集合，在与策略关联的所有规则匹配时定义采取的操作。

10.3. 第 7 层负载均衡规则

对于 Red Hat OpenStack Platform 负载均衡服务(octavia)，一个第 7 层(L7)负载平衡规则是一个单项简单的逻辑测试，返回 true 或 false。它由规则类型、比较类型、值以及根据规则类型使用的可选键组成。L7 规则必须始终与 L7 策略关联。

注意

您无法使用 UDP 负载均衡器创建 L7 策略和规则。

其他资源

- 第 10.4 节 “第 7 层负载均衡规则类型”

10.4. 第 7 层负载均衡规则类型

Red Hat OpenStack Platform 负载均衡服务(octavia)有以下第 7 层负载均衡规则类型：

- HOST_NAME ：规则将请求的 HTTP/1.1 主机名与规则中的 value 参数进行比较。
- PATH ：规则将 HTTP URI 的路径部分与规则中的 value 参数进行比较。
- FILE_TYPE ：规则将 URI 的最后一部分与规则中的 value 参数进行比较，如 txt、jpg 等。
- HEADER ：规则查找 key 参数中定义的标头，并将其与规则中的 value 参数进行比较。

- COOKIE ：规则查找由 key 参数命名的 Cookie，并将其与规则中的 value 参数进行比较。
- SSL_CONN_HAS_CERT ：如果客户端呈现了 TLS 客户端身份验证的证书，则规则匹配。这并不意味着该证书有效。
- SSL_VERIFY_RESULT ：此规则与 TLS 客户端身份验证证书验证结果匹配。值0表示证书已被成功验证。值大于零表示证书失败验证。这个值遵循 openssl-verify 结果代码。
- SSL_DN_FIELD ：规则查找 key 参数中定义的可辨识名称字段，并将其与规则中的 value 参数进行比较。

其他资源

- 第 10.3 节 “第 7 层负载均衡规则”

10.5. 第 7 层负载均衡规则比较类型

对于 Red Hat OpenStack Platform 负载均衡服务(octavia)，给定类型的第 7 层负载均衡规则始终执行比较。负载平衡服务支持以下类型比较：并非所有规则类型都支持所有比较类型：

- REGEX: Perl 类型正则表达式匹配
- STARTS_WITH: 字符串开头
- ENDS_WITH: 字符串结尾
- CONTAINS: String 包含
- EQUAL_TO ：字符串等于

其他资源

- 第 10.4 节 “第 7 层负载均衡规则类型”

10.6. 第 7 层负载均衡规则会导致版本

为了更加全面地表达一些策略需要并且 Red Hat OpenStack Platform 负载均衡服务(octavia)使用的逻辑，第 7 层负载均衡规则可以将其结果推断出来。如果给定规则的 invert 参数为 true，则其比较的结果会被颠倒。

例如，有效 等于 规则的负数将变为 不等于 规则。只有在指定正则表达式不匹配时才返回为 true。

其他资源

- 第 10.7 节 “第 7 层负载均衡策略”

10.7. 第 7 层负载均衡策略

对于 Red Hat OpenStack Platform 负载均衡服务(octavia)，第 7 层(L7)负载平衡策略是与监听程序关联的 L7 规则集合，它可能与后端池关联。如果策略中所有规则都为 true，则策略是负载均衡器执行的操作。

注意

您无法使用 UDP 负载均衡器创建 L7 策略和规则。

其他资源

- 第 10.3 节 “第 7 层负载均衡规则”

10.8. 第 7 层负载均衡策略逻辑

Red Hat OpenStack Platform 负载均衡服务(octavia)第 7 层负载均衡策略使用以下逻辑：与给定策略关联的所有规则在逻辑上被逻辑 AND。请求必须与所有策略规则匹配才能与策略匹配。

如果您需要在规则之间表达逻辑 OR 操作，请使用同一操作创建多个策略，或者生成更加详尽的正则表达式。

其他资源

- 第 10.3 节 “第 7 层负载均衡规则”

10.9. 第 7 层负载均衡策略操作

如果第 7 层负载均衡策略与给定请求匹配，则执行该策略操作。以下是 L7 策略可能需要执行的操作：

- REJECT ：请求被拒绝使用适当的响应代码，且不会转发到任何后端池。
- REDIRECT_TO_URL ：该请求将向 redirect_url 参数中定义的 URL 发送 HTTP 重定向。
- REDIRECT_PREFIX ：与此策略匹配的请求重定向至此前缀 URL。
- REDIRECT_TO_POOL ：此请求转发到与 L7 策略关联的后端池。

其他资源

- 第 10.7 节 “第 7 层负载均衡策略”

10.10. 第 7 层负载均衡策略位置

对于 Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)，当多个第 7 (L7)负载均衡策略与监听程序关联时，策略位置参数的值会变得非常重要。在确定评估了 L7 策略的顺序时使用 position 参数。策略位置会影响监听程序的行为：

- 在负载均衡服务(haproxy amphorae)的引用实现中，HAProxy 强制执行以下对策略操作的顺序：
  - REJECT 策略优先于所有其他策略。
  - REDIRECT_TO_URL 策略优先于 REDIRECT_TO_POOL 策略。

- REDIRECT_TO_POOL 策略仅在上述所有后评估，按照策略指定位置的顺序进行评估。

- L7 策略以特定顺序进行评估，按照位置属性定义，与给定请求匹配的第一个策略是其操作所遵循的。

- 如果没有策略与给定请求匹配，则请求将路由到监听程序的默认池（如果存在）。如果侦听器没有默认池，则返回 503 错误。

- 策略位置编号以1开头。

- 如果使用与现有策略匹配的位置创建新策略，则会在给定位置插入新策略。

- 如果创建新策略而不指定位置，或者指定超过列表中已超过策略数的位置，新策略将附加到列表中。

- 当策略插入、删除或附加到列表中时，策略位置值将从(1)中重新排序，而不会跳过数字。例如，如果策略 A、B 和 C 分别具有 1、2 和 3 的位置，如果您从列表中删除策略 B，则策略 C 的位置将变为 2。

其他资源

- 第 10.7 节 “第 7 层负载均衡策略”

10.11. 重定向未安全 HTTP 请求到安全 HTTP

您可以使用 Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)带有第 7 层(L7)策略，将在非安全 TCP 端口上收到的 HTTP 请求重定向到安全 TCP 端口。

在本例中，任何到达未安全 TCP 端口 80 的 HTTP 请求都会重定向到安全 TCP 端口 443。

前提条件

- 具有侦听器（侦听器1）和池（池1）的 TLS 终止的 HTTPS 负载均衡器(lb1)。如需更多信

息，请参阅 创建 TLS 终止的 HTTPS 负载均衡器。

流程

1. 提供您的凭据文件。

Example

```
$ source ~/overcloudrc
```

2. 在负载均衡器(lb1)端口(80)上创建 HTTP 侦听器(http_listener)。

注意

括号中的值是本流程中示例命令中使用的示例值。将这些示例值替换为适合您的站点的值。

Example

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

3. 在监听器(http_listener)上创建 L7 策略(policy1)。策略必须包含操作(REDIRECT_TO_URL)并指向 URL (https://www.example.com/)。

Example

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

4. **添加与策略中的所有请求匹配的 L7 规则(policy1)。**

Example

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

验证

1. 运行 **openstack loadbalancer l7policy list 命令，再验证策略 policy1 是否存在。**

2. 运行 **openstack loadbalancer l7rule list <l7policy& gt; 命令，并验证规则是否与 compare_type of STARTS_WITH 存在。**

Example

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

其他资源

- ***在 命令行界面参考 中创建 LoadBalancer 监听程序***
- ***在 命令行界面参考 中创建 LoadBalancer l7policy***
- ***在 命令行界面参考 中创建 LoadBalancer l7rule***

10.12. 根据到池的起始路径重定向请求

您可以使用 Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)将 HTTP 请求重定向到另一个服务器池。您可以定义第 7 层(L7)策略，以匹配请求的 URL 中的一个或多个启动路径。

在本例中，任何包含以 /js 或 /images 开头的 URL 的请求都会被重新定向到备用静态内容服务器池。

前提条件

- 具有侦听器（侦听器1）和池（池1）的 HTTP 负载均衡器(lb1)。如需更多信息，请参阅使用健康监控器创建 HTTP 负载均衡器。

流程

1. 提供您的凭据文件。

Example

```
$ source ~/overcloudrc
```

2. 在负载平衡器(lb1)上创建第二个池(static_pool)。

注意

括号中的值是本流程中示例命令中使用的示例值。将这些示例值替换为适合您的站点的值。

Example

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

3. **将专用子网(private_subnet)上的负载平衡器成员(192.0.2.10 和 192.0.2.11)添加到池(static_pool)：**

Example

```
$ 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. **在监听器（侦听器1）上创建 L7 策略(policy1)。策略必须包含操作(REDIRECT_TO_POOL)，并指向池(static_pool)。**

Example

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

5. **添加 L7 规则，用于在到策略请求路径的开头查找 /js。**

Example

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

6. **使用操作(REDIRECT_TO_POOL)创建 L7 策略(policy2)，再添加池中所指向的监听程序(listener1)。**

Example

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

7. **添加 L7 规则，该规则在请求路径开始时查找 /images。**

Example

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

验证

1. 运行 **openstack loadbalancer l7policy list** 命令，再验证策略、**policy1** 和 **policy2** 是否存在。

2. 运行 **openstack loadbalancer l7rule list <l7policy >** 命令，并验证每个对应策略是否存在带有 **compare_type** 为 **STARTS_WITH** 的规则。

Example

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

其他资源

- 在 *命令行界面参考* 中创建 LoadBalancer 池
- 在 *命令行界面参考* 中创建 LoadBalancer 成员
- 在 *命令行界面参考* 中创建 LoadBalancer l7policy
- 在 *命令行界面参考* 中创建 LoadBalancer l7rule

10.13. 将子域请求发送到特定池

您可以使用带有第 7 (L7)策略的 Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)策略将包含特定 HTTP/1.1 主机名的请求重定向到不同的应用程序服务器池。

在这个示例中，任何包含 HTTP/1.1 主机名 www2.example.com 的请求都会被重新定向到备用池应用程序服务器 pool2。

前提条件

- 具有侦听器（侦听器1）和池（池1）的 HTTP 负载均衡器(lb1)。如需更多信息，请参阅使用健康监控器创建 HTTP 负载均衡器。

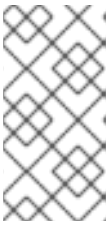
流程

1. 提供您的凭据文件。

   Example

   ```
   $ source ~/overcloudrc
   ```

2. **在负载平衡器(lb1)上创建第二个池(pool2)。**



注意

括号中的值是本流程中示例命令中使用的示例值。将这些示例值替换为适合您的站点的值。

Example

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

3. **在监听器（侦听器1）上创建 L7 策略(policy1)。策略必须包含操作(REDIRECT_TO_POOL)，并指向池(pool2)。**

Example

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

4. **在策略中添加 L7 规则，它使用 HTTP/1.1 主机名 www2.example.com 发送任何请求到第二个池(池2)。**

Example

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

验证

1. 运行 **openstack loadbalancer l7policy list** 命令，再验证策略 **policy1** 是否存在。

2. 运行 **openstack loadbalancer l7rule list <l7policy>** 命令，并验证策略是否存在带有 **compare_type** 的 **EQUAL_TO** 规则。

   Example

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

其他资源

- 在 *命令行界面参考* 中创建 **LoadBalancer** 池
- 在 *命令行界面参考* 中创建 **LoadBalancer l7policy**
- 在 *命令行界面参考* 中创建 **LoadBalancer l7rule**

10.14. 根据结束到特定池的主机名发送请求

您可以使用 **Red Hat OpenStack Platform (RHOSP)**负载均衡服务**(octavia)**及第 **7 (L7)**策略来重定向包含 **HTTP/1.1** 主机名的请求，该请求以特定字符串结束到不同的应用程序服务器池。

在本例中，任何以 **.example.com** 结尾的 **HTTP/1.1** 主机名的请求都会被重新定向到备用池应用程序服务器 **pool2**。

前提条件

-

具有侦听器（侦听器1）和池（池1）的 HTTP 负载均衡器(lb1)。如需更多信息，请参阅使用健康监控器创建 HTTP 负载均衡器

流程

1. 提供您的凭据文件。

Example

```
$ source ~/overcloudrc
```

2. 在负载平衡器(lb1)上创建第二个池(pool2)。

注意

括号中的值是本流程中示例命令中使用的示例值。将这些示例值替换为适合您的站点的值。

Example

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

3. 在监听器（侦听器1）上创建 L7 策略(policy1)。策略必须包含操作(REDIRECT_TO_POOL)，并指向池(pool2)。

Example

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

4. **在策略中添加 L7 规则，将任何使用 HTTP/1.1 主机名(www2.example.com)的请求发送到第二个池(pool2)。**

Example

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

验证

1. 运行 **openstack loadbalancer l7policy list** 命令，再验证策略 **policy1** 是否存在。

2. 运行 **openstack loadbalancer l7rule list <l7policy>** 命令，并验证策略是否存在带有 **compare_type** 的 **EQUAL_TO** 规则。

Example

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

其他资源

- 在 *命令行界面参考* 中创建 **LoadBalancer 池**
- 在 *命令行界面参考* 中创建 **LoadBalancer l7policy**

- 在 *命令行界面参考* 中创建 LoadBalancer l7rule

10.15. 将基于浏览器 COOKIE 的请求发送到特定的池

您可以使用 Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)将未经身份验证的 Web 客户端请求重定向到包含一个或多个身份验证服务器的不同池。第 7 层(L7)策略决定传入请求是否缺少身份验证 Cookie。

在本例中，任何缺少浏览器 cookie 的 Web 客户端请求 都会重新定向到包含身份验证服务器的备用池。

重要

此流程提供了如何使用浏览器 Cookie 执行 L7 应用程序路由的示例，且无法解决安全问题。

前提条件

- 具有侦听器（侦听器1）和池（池1）的 TLS 终止的 HTTPS 负载均衡器(lb1)。如需更多信息，请参阅 创建 TLS 终止的 HTTPS 负载均衡器。
- 第二个 RHOSP Networking 服务(neutron)子网，其中的安全身份验证服务器验证 Web 用户。

流程

1. 提供您的凭据文件。

   Example

   ```
   $ source ~/overcloudrc
   ```

2. **在负载平衡器(lb1)上创建第二个池(login_pool)。**

注意

括号中的值是本流程中示例命令中使用的示例值。将这些示例值替换为适合您的站点的值。

Example

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

3. **向第二个池添加身份验证子网(secure_subnet)上的安全身份验证服务器(192.0.2.10)的成员。**

Example

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

4. **在监听器（侦听器1）上创建 L7 策略(policy1)。策略必须包含操作(REDIRECT_TO_POOL)，并指向第二个池(login_pool)。**

Example

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

5. **向策略(policy1)添加 L7 规则，它使用任何值搜索浏览器 cookie (auth_token)，并在不存在 Cookie 时匹配。**

Example

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

验证

1. 运行 **openstack loadbalancer l7policy list 命令，再验证策略 policy1 是否存在。**

2. 运行 **openstack loadbalancer l7rule list <l7policy& gt; 命令，并验证存在带有 compare_type of REGEX 的规则。**

Example

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

其他资源

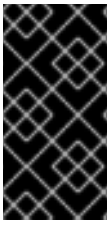
- ***在 命令行界面参考* 中创建 LoadBalancer 池**
- ***在 命令行界面参考* 中创建 LoadBalancer 成员**
- ***在 命令行界面参考* 中创建 LoadBalancer l7policy**

- 在 *命令行界面参考* 中创建 LoadBalancer l7rule

10.16. 根据浏览器的 COOKIE 或无效的 COOKIE 值发送到特定的池，将请求发送到特定的池

您可以使用 Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)将未经身份验证的 Web 客户端请求重定向到包含一个或多个身份验证服务器的不同池。第 7 层(L7)策略决定传入的请求是否缺少身份验证 Cookie，或包含具有特定值的身份验证 Cookie。

在本例中，任何缺少浏览器 cookie、auth_token 或具有 auth_token 的 Web 客户端请求都会重定向到包含身份验证服务器的备用池。



重要

此流程提供了如何使用浏览器 Cookie 执行 L7 应用程序路由的示例，且无法解决安全问题。

前提条件

- 具有侦听器（侦听器1）和池（池1）的 TLS 终止的 HTTPS 负载均衡器(lb1)。如需更多信息，请参阅 创建 TLS 终止的 HTTPS 负载均衡器。
- 第二个 RHOSP Networking 服务(neutron)子网，其中的安全身份验证服务器验证 Web 用户。

流程

1. 提供您的凭据文件。

   Example

   ```
   $ source ~/overcloudrc
   ```

2. **在负载平衡器(lb1)上创建第二个池(login_pool)。**

注意

括号中的值是本流程中示例命令中使用的示例值。将这些示例值替换为适合您的站点的值。

Example

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

3. **向第二个池添加身份验证子网(secure_subnet)上的安全身份验证服务器(192.0.2.10)的成员。**

Example

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

4. **在监听器（侦听器1）上创建 L7 策略(policy1)。策略必须包含操作(REDIRECT_TO_POOL)，并指向第二个池(login_pool)。**

Example

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

5. **向策略(policy1)添加 L7 规则，它使用任何值搜索浏览器 cookie (auth_token)，并在不存在 Cookie 时匹配。**

   Example

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

6. **在监听器（侦听器1）上创建第二个 L7 策略(policy2)。策略必须包含操作 (REDIRECT_TO_POOL)，并指向第二个池(login_pool)。**

   Example

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

7. **在第二个策略(policy2)中添加 L7 规则，用于搜索浏览器 cookie (auth_token)，并在 Cookie 值等于字符串 INVALID 时匹配。**

   Example

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

验证

1. 运行 **openstack loadbalancer l7policy list** 命令，再验证策略、**policy1** 和 **policy2** 是否存在。

2. 运行 **openstack loadbalancer l7rule list <l7policy >** 命令，再验证存在与 **policy1** 的 **compare_type** 为 **REGEX** 的规则，以及与 **policy2** 为 **EQUAL_TO** 的 **compare_type** 的规则存在。

Example

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

其他资源

- 在 *命令行界面参考* 中创建 LoadBalancer 池
- 在 *命令行界面参考* 中创建 LoadBalancer 成员
- 在 *命令行界面参考* 中创建 LoadBalancer l7policy
- 在 *命令行界面参考* 中创建 LoadBalancer l7rule

10.17. 将请求发送到名称与主机名和路径匹配的池

您可以使用 Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)将与特定条件匹配的 web 客户端请求重定向到另一个应用程序服务器池。业务逻辑标准通过第 7 层(L7)策略来执行，该政策会尝试与预定义的主机名和请求路径匹配。

在本例中，任何与 **hostname api.example.com** 匹配的 web 客户端请求，并在请求路径开始时具有 **/api** 重定向到另一个池 **api_pool**。

前提条件

- 具有侦听器（侦听器1）和池（池1）的 HTTP 负载均衡器(lb1)。如需更多信息，请参阅使用健康监控器创建 HTTP 负载均衡器。

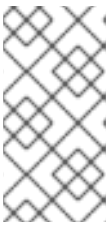
流程

1. 提供您的凭据文件。

   Example

   ```
   $ source ~/overcloudrc
   ```

2. 在负载平衡器(lb1)上创建第二个池(api_pool)。

   注意

   括号中的值是本流程中示例命令中使用的示例值。将这些示例值替换为适合您的站点的值。

   Example

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

3. 将专用子网(private_subnet)上的负载平衡器成员(192.0.2.10 和 192.0.2.11)添加到池(static_pool)：

   Example

```
$ 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. **在监听器（侦听器1）上创建 L7 策略(policy1)。策略必须包含操作(REDIRECT_TO_POOL)，并指向池(api_pool)。**

Example

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

5. **在与 hostname api.example.com 匹配的策略中添加 L7 规则。**

Example

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

6. **将第二个 L7 规则添加到与请求路径开头的 /api 匹配的策略。**

该规则通过第一个规则进行逻辑化。

Example

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

-

验证

1. 运行 **openstack loadbalancer l7policy list** 命令，再验证策略 **policy1** 是否存在。

2. 运行 **openstack loadbalancer l7rule list <l7policy& gt; 命令，并验证规则及 compare_type** 为 **EQUAL_TO 和 STARTS_WITH，分别适用于 policy1。**

Example

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

其他资源

- 在 *命令行界面参考* 中创建 **LoadBalancer 池**
- 在 *命令行界面参考* 中创建 **LoadBalancer 成员**
- 在 *命令行界面参考* 中创建 **LoadBalancer l7policy**
- 在 *命令行界面参考* 中创建 **LoadBalancer l7rule**

10.18. 使用 COOKIE 在现有生产站点上配置 A-B 测试

您可以使用带有第 7 (L7)策略的 Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)策略为生产环境网站配置 A-B 测试或分割测试。

在本例中，路由到网站的"B"版本的 Web 客户端，由池(pool1)中的成员服务器将 cookie site_version

设置为 **B**。

前提条件

- 两个生产网站（现场 **A** 和 **site B**）。
- 您已根据根据池启动路径"重定向请求的说明配置了 **HTTP** 负载均衡器"。 所需的配置概述如下：
  - 负载均衡器(lb1)上的侦听器（监听程序1）。
  - 以 **/js** 或 **/images** 开头的 **URL** 发送到池(static_pool)的 **HTTP** 请求。
  - 所有其他请求都发送到侦听器的默认池(池1)。
  - 有关配置的更多信息，请参阅 第 10.12 节 “根据到池的起始路径重定向请求”。

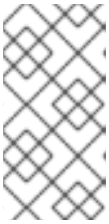
流程

1. 提供您的凭据文件。

   Example

   ```
   $ source ~/overcloudrc
   ```

2. 在负载平衡器(lb1)上创建第三个池(pool_B)。



注意

括号中的值是本流程中示例命令中使用的示例值。将这些示例值替换为适合您的站点的值。

Example

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

3. **将专用子网(private_subnet)上的负载平衡器成员(192.0.2.50 和 192.0.2.51)添加到池(pool_B)：**

Example

```
$ 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. **在负载平衡器(lb1)上创建第四个池(static_pool_B)。**

Example

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

5. **将专用子网(private_subnet)上的负载平衡器成员(192.0.2.100 和 192.0.2.101)添加到池**

(static_pool_B) :

Example

```
$ 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. **在监听器（侦听器1）上创建 L7 策略(policy2)。策略必须包含操作(REDIRECT_TO_POOL)，并指向池(static_pool_B)。在位置 1 插入策略。**

Example

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

7. **在策略(policy2)中添加 L7 规则，它使用正则表达式在请求路径开始时匹配 /js 或 /images。**

Example

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

8. **将第二个 L7 规则添加到与 Cookie (site_version)匹配的策略(policy2)到确切字符串(B)。**

Example

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

9. **在监听器（侦听器1）上创建 L7 策略(policy3)。策略必须包含操作(REDIRECT_TO_POOL)，并指向池(pool_B)。在位置 2 插入策略。**

Example

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

10. **将 L7 规则添加到与 Cookie (site_version)匹配的策略（策略3）到精确字符串(B)。**

Example

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

注意

为较低位置分配具有最具体规则的 L7 策略非常重要，因为规则对 True 做出的所有评估就是其操作的策略。在这一流程中，需要在 policy3 之前对 policy2 进行评估，以避免向不正确的池发送请求。

验证

1. **运行 openstack loadbalancer l7policy list 命令，再验证策略、policy2 和 policy3 是否存**

在。

2. 运行 **openstack loadbalancer l7rule list <l7policy >** 命令，并验证每个对应策略是否存在带有 **compare_type** 为 **STARTS_WITH** 的规则。

Example

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

其他资源

- 在 *命令行界面参考* 中创建 LoadBalancer 池
- 在 *命令行界面参考* 中创建 LoadBalancer 成员
- 在 *命令行界面参考* 中创建 LoadBalancer l7policy
- 在 *命令行界面参考* 中创建 LoadBalancer l7rule

第 11 章 更新和升级负载均衡服务

执行定期更新和升级，以便您可以使用最新的 Red Hat OpenStack Platform 负载均衡服务功能，并避免因更新和升级而造成的可能长度和问题的问题。

- 第 11.1 节 “更新和升级负载均衡服务”
- 第 11.2 节 “更新正在运行的负载均衡服务实例”

11.1. 更新和升级负载均衡服务

负载均衡服务(octavia)是 Red Hat OpenStack Platform (RHOSP)更新或升级的一部分。

前提条件

- 调度一个维护窗口来执行升级，因为在升级负载均衡服务 control plane 过程中无法完全正常工作。

流程

1. 执行 RHOSP 更新，如 *保持 Red Hat OpenStack Platform 更新* 指南中所述。
2. 应用维护发行后，如果您需要使用新功能，则轮转正在运行的 amphorae 以将它们更新为最新的 amphorara。

其他资源

- 保持 Red Hat OpenStack Platform 更新指南
- 第 11.2 节 “更新正在运行的负载均衡服务实例”
- 第 2.2 节 “负载均衡服务(octavia)功能支持列表”

11.2. 更新正在运行的负载均衡服务实例

您可以定期使用较新的镜像更新正在运行的负载均衡服务实例(amphora)。例如，您可能希望在以下事件期间更新 amphora 实例：

- Red Hat OpenStack Platform (RHOSP)的更新或升级。
- 系统安全更新。
- 对底层虚拟机的不同类别的更改。

在 RHOSP 更新或升级过程中，director 会自动下载默认的 amphora 镜像，将其上传到 overcloud 镜像服务(glance)，然后配置负载均衡服务(octavia)以使用新镜像。当您在负载均衡器上出现故障时，您可以强制负载平衡服务启动使用新 amphora 镜像的实例(amphora)。

前提条件

- amphora 的新镜像。它们在 RHOSP 更新或升级过程中可用。

流程

1. 提供您的凭据文件。

   Example

   ```
   $ source ~/overcloudrc
   ```

2. 列出您要更新的所有负载均衡器的 ID：

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

3. 每个负载均衡器的失败：

   -

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

注意

当您在负载均衡器上启动失败、监控系统使用率以及需要时，调整执行故障转移的速度。负载平衡器故障转移可创建新的虚拟机和端口，这可能会暂时增加 OpenStack 网络的负载。

4. 监控负载均衡器的失败状态：

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

当负载均衡器状态变为 **ACTIVE** 时，更新已完成。

其他资源

- *命令行界面 参考中的 LoadBalancer*

第 12 章 故障排除和维护负载平衡服务

对负载均衡服务(octavia)的基本故障排除和维护始于 OpenStack 客户端命令来显示状态和迁移实例，以及了解如何访问日志。如果您需要更加深入地进行故障排除，您可以 SSH 到一个或多个负载平衡服务实例(amphoe)。

12.1. 验证负载均衡器

您可以通过查看负载均衡器显示和列出命令的输出，对负载均衡服务(octavia)及其组件进行故障排除。

流程

1. 提供您的凭据文件。

   Example

   ```
   $ source ~/overcloudrc
   ```

2. 验证负载均衡器(lb1)设置。

注意

括号中的值是本流程中示例命令中使用的示例值。将这些示例值替换为适合您的站点的值。

Example

```
$ openstack loadbalancer show lb1
```

输出示例

```
+---------------------+--------------------------------------+
| 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. **使用 Loadbalancer ID (265d0b71-c073-40f4-9718-8a182c6d53ca)，获取与负载均衡器 (lb1)关联的 amphora 的 ID。**

Example

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

输出示例

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

4. **使用上一步中的 amphora ID (1afabefd-ba09-49e1-8c39-41770a25070)，查看 amphora 信息。**

Example

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

输出示例

```
+-----------------+--------------------------------------+
| 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. 查看侦听器(侦听器1)详细信息。

Example

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

输出示例

```
+-----------------------------+--------------------------------------+
| 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. **查看池(pool1)和 load-balancer 成员。**

Example

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

输出示例

```
+----------------------+--------------------------------------+
| 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. **通过连接到负载均衡器的 VIP 地址(192.0.2.177)，验证针对 HTTPS 或 TERMINATED_HTTPS 协议配置的负载均衡器的 HTTPS 流量流。**

提示

使用命令 openstack loadbalancer show <load_balancer_name> 获取负载均衡器 VIP 地址。

Example

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

输出示例

```
* 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
```

其他资源

- *命令行界面 参考中的 LoadBalancer*

12.2. 迁移特定负载均衡服务实例

在某些情况下，您必须迁移负载平衡服务实例(amphora)。例如，如果主机正在关闭以进行维护

流程

1. 提供您的凭据文件。

   Example

   ```
   $ source ~/overcloudrc
   ```

2. 找到您要迁移的 amphora 的 ID。后续步骤中需要提供 ID。

   ```
   $ openstack loadbalancer amphora list
   ```

3. 为防止计算调度程序服务将任何新的 amphorae 调度到正在撤离的 Compute 节点，请禁用计算节点(compute-host-1)。

注意

括号中的值是本流程中示例命令中使用的示例值。将这些示例值替换为适合您的站点的值。

Example

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

4. 使用您获取的 amphora ID (ea17210a-1076-48ff-8a1f-ced49ccb5e53)通过 amphora 故障切换。

Example

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

其他资源

- *命令行界面参考中的* 计算服务设置
- *命令行界面 参考中的 LoadBalancer*

12.3. 使用 SSH 连接到负载均衡实例

在对服务问题进行故障排除时，使用 SSH 登录负载平衡服务实例(amphorae)。

在对服务问题进行故障排除时，使用 Secure Shell (SSH)登录运行负载均衡服务实例(amphorae)会很有帮助。

前提条件

- 您必须具有负载均衡服务(octavia) SSH 私钥。

- 在创建负载均衡器前，您必须在负载均衡服务配置中启用 SSH。

流程

1. 在 director 节点上，启动 ssh-agent 并将用户身份密钥添加到代理中：

```
$ eval $(ssh-agent -s)
$ ssh-add
```

2. 提供您的凭据文件。

Example

```
$ source ~/overcloudrc
```

3. 确定您要连接的 amphora 的负载平衡管理网络(lb_network_ip)上的 IP 地址：

```
$ openstack loadbalancer amphora list
```

4. 使用 SSH 连接到 amphora：

```
$ ssh -A -t heat-admin@<controller_node_IP_address> ssh cloud-user@<lb_network_ip>
```

5. 完成后，关闭与 amphora 的连接并停止 SSH 代理：

```
$ exit
```

其他资源

- *命令行界面 参考中的 LoadBalancer*

12.4. 显示监听程序统计

使用 OpenStack 客户端，您可以获取有关特定 Red Hat OpenStack Platform (RHOSP)负载均衡器的监听程序的统计信息：

- 当前活动连接(active_connections)。
- 收到的字节总数(bytes_in)。
- 发送的总字节数(bytes_out)。
- 无法实现的请求总数(request_errors)。
- 已处理的连接总数(total_connections)。

流程

1. 提供您的凭据文件。

   Example

   ```
   $ source ~/overcloudrc
   ```

2. 查看监听器(监听器1)的统计数据。

   注意

   括号中的值是本流程中示例命令中使用的示例值。将这些示例值替换为适合您的站点的值。

   Example

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

提示

如果您不知道监听器的名称，请输入命令 **loadbalancer** 侦听器列表。

输出示例

```
+--------------------+-------+
| Field              | Value |
+--------------------+-------+
| active_connections | 0     |
| bytes_in           | 0     |
| bytes_out          | 0     |
| request_errors     | 0     |
| total_connections  | 0     |
+--------------------+-------+
```

其他资源

- LoadBalancer 监听程序统计显示在 *命令行界面参考*中
- 第 12.5 节 “解释监听程序请求错误”

12.5. 解释监听程序请求错误

您可以获取有关特定 Red Hat OpenStack Platform (RHOSP)负载均衡器的监听程序的统计信息。更多信息请参阅 第 12.4 节 “显示监听程序统计”。

RHOSP 负载均衡器 request_errors 跟踪的其中一个统计仅计算来自最终用户连接到负载均衡器时的请求中出现的错误。request_errors 变量不测量成员服务器报告的错误。

例如，如果租户通过 RHOSP 负载均衡服务(octavia)连接到返回 HTTP 状态代码 400 (Bad Request) 的 Web 服务器，则负载均衡服务不会收集此错误。负载平衡器不检查数据流量的内容。在本例中，负载均衡器将这一流解释为成功，因为它会正确在用户和 Web 服务器之间传输信息。

以下条件可能会导致 request_errors 变量递增：

- 在发送请求前，先从客户端终止。
- 从客户端读取错误。
- 客户端超时。
- 客户端关闭连接。
- 来自客户端的各种错误请求。

其他资源

- LoadBalancer 监听程序统计显示在 *命令行界面参考*中
- 第 12.4 节 “显示监听程序统计”