Red Hat OpenStack Platform 17.1

将负载均衡配置为服务

使用负载均衡服务(octavia)管理数据平面间的网络流量

Last Updated: 2024-05-23

Red Hat OpenStack Platform 17.1 将负载均衡配置为服务

使用负载均衡服务(octavia)管理数据平面间的网络流量

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。与我们分享您的成功秘诀。

在 JIRA 中提供文档反馈

使用 Create Issue 表单对文档提供反馈。JIRA 问题将在 Red Hat OpenStack Platform Jira 项目中创建，您可以在其中跟踪您的反馈进度。

1. 确保您已登录到 JIRA。如果您没有 JIRA 帐户，请创建一个帐户来提交反馈。
2. 点击以下链接打开 **Create Issue** 页面： Create Issue
3. 完成 **Summary** 和 **Description** 字段。在 **Description** 字段中，包含文档 URL、章节或章节号以及问题的详细描述。不要修改表单中的任何其他字段。
4. 点 **Create**。

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。提供按需扩展的功能，水平扩展使负载平衡服务成为适合大型 RHOSP 企业部署的功能齐全的负载均衡器。

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，称为位于 Compute 节点上的 *amphorae*。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 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

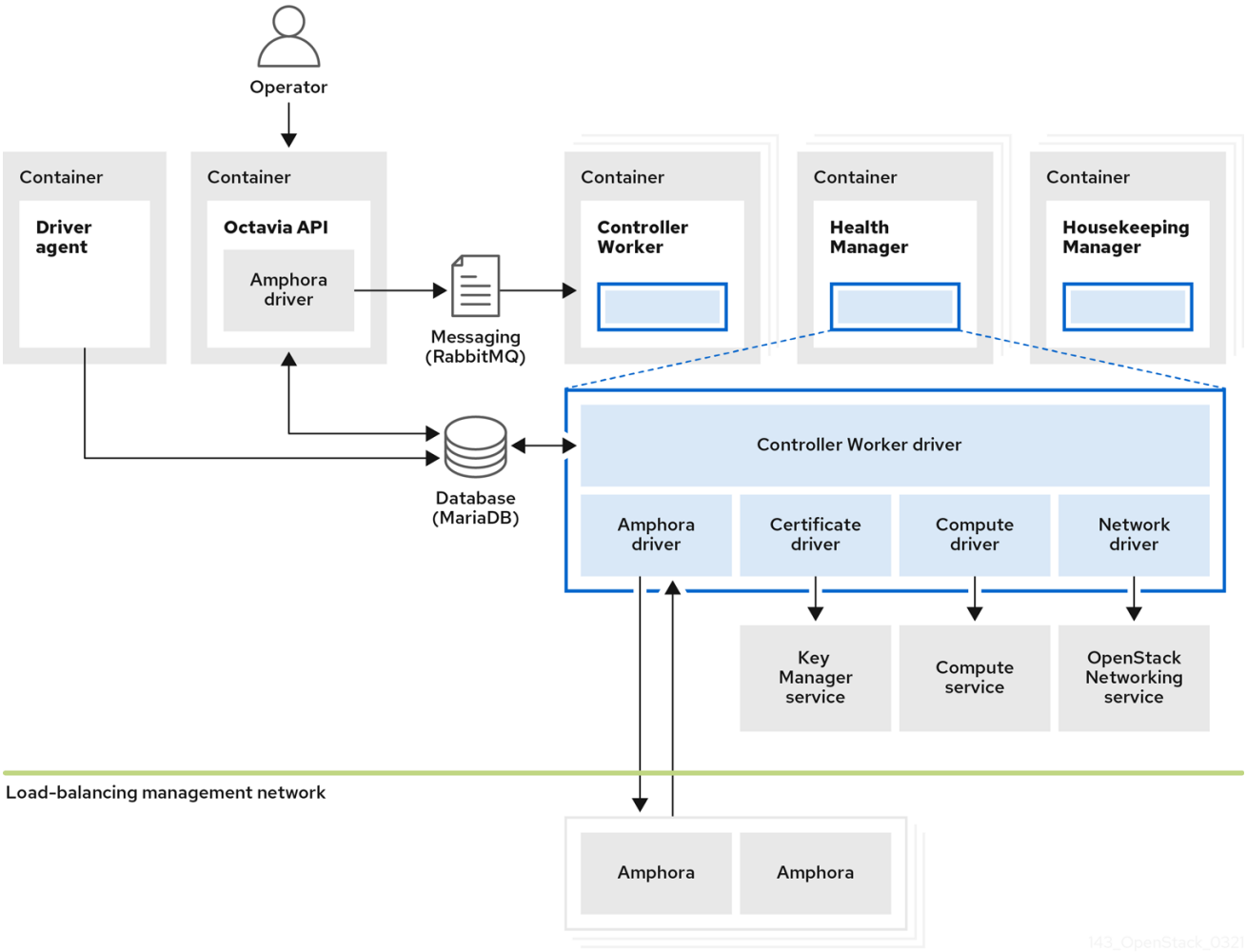


图 1.1 显示负载平衡服务的组件托管在与网络 API 服务器相同的节点上，默认为位于 Controller 节点上。负载均衡服务由以下组件组成：

Octavia API (octavia_api 容器)

为用户提供 REST API 与 octavia 交互。

Controller Worker (octavia_worker 容器)

在负载平衡管理网络上向 amphorae 发送配置和配置更新。

Health Manager (octavia_health_manager container)

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

housekeeping Manager (octavia_housekeeping container)

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 container)

支持其他供应商驱动程序，如 OVN。

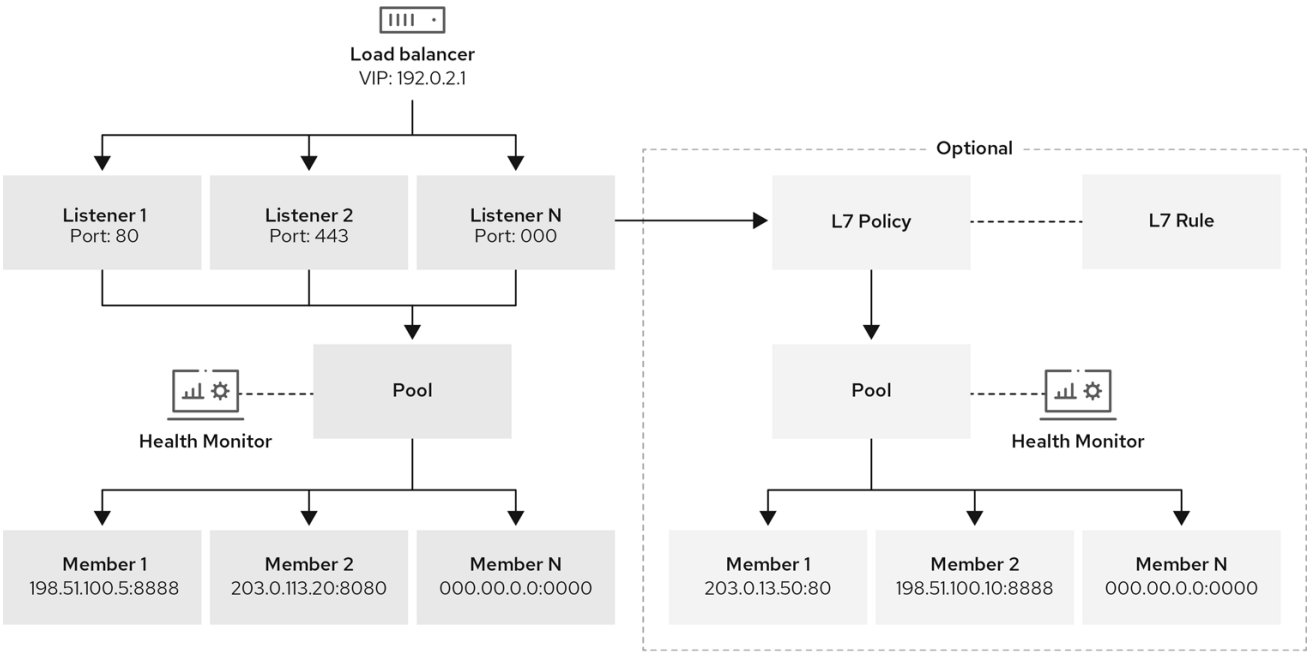
Amphora

执行负载平衡。amphorae 通常是在 Compute 节点上运行的实例，它根据监听器、池、运行状况监控、L7 策略和成员配置使用负载平衡参数进行配置。amphorae 将定期心跳发送到 Health Manager。

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. 负载均衡服务对象模型图



负载均衡器

代表负载平衡实体的顶级 API 对象。在创建负载均衡器时，VIP 地址会被分配。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。

listener

负载均衡器侦听的端口，例如，HTTP 的 TCP 端口 80。

健康监控器

在每个后端成员服务器上执行定期健康检查的进程，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 将池与多个监听器相关联。您可以使用 L7 策略共享池。

成员

描述如何连接到后端实例或服务。此描述由提供后端成员的 IP 地址和网络端口组成。

L7 规则

定义确定 L7 策略是否应用到连接的第 7 层(L7)条件。

L7 策略

与侦听器关联的 L7 规则集合，它们也可能具有与后端池的关联。policies 描述了如果策略中的所有规则都为 true，则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)** - 用于管理负载平衡服务虚拟机实例(amphora)生命周期，以及按需创建计算资源。
- **网络服务(neutron)** - 用于 amphorae、租户环境和外部网络之间的网络连接。
- **Key Manager 服务(barbican)** - 用于在侦听器上配置 TLS 会话终止时管理 TLS 证书和凭证。
- **Identity service (keystone)** - 对于对 octavia API 的身份验证请求，以及负载均衡服务以与其他 RHOSP 服务进行身份验证。
- **镜像服务(glance)** - 用于存储 amphora 虚拟机镜像。
- **通用库(oslo)** - 对于负载平衡服务控制器组件之间的通信，使负载平衡服务在标准的 OpenStack 框架内工作，并查看系统以及项目代码结构。
- **taskflow** - 属于通用库的一部分；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功能功能的服务替换，则驱动程序接口可避免重大重组负载平衡服务。

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，如选择使用哪个供应商还是要实现高可用性环境：

- 第 2.1 节 “负载均衡服务提供商驱动程序”
- 第 2.2 节 “负载均衡服务(octavia)功能支持列表”
- 第 2.3 节 “负载均衡服务软件要求”
- 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 active-standby 拓扑的基础知识”
- 第 2.6 节 “负载均衡服务的部署后步骤”

2.1. 负载均衡服务提供商驱动程序

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，或同时使用多个供应商驱动程序。

RHOSP 提供两个负载平衡提供程序，amphora 和 Open Virtual Network (OVN)。

Amphora （默认值）是一个高度可用的负载均衡器，它带有一个与您的计算环境扩展的功能集。因此，amphora 适用于大规模部署。

OVN 负载平衡提供程序是一个轻量级负载均衡器，它带有一个基本功能集。OVN 通常为 east-west，第 4 层网络流量。OVN 快速置备并消耗比功能齐全的负载平衡提供商（如 amphora）的资源更少。

在使用带有 OVN 机制驱动程序(ML2/OVN)的 neutron Modular Layer 2 插件的 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 负载均衡提供程序，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平衡供应商，需要单独的 haproxy 虚拟机和一个额外的延迟跃点。

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) 17.1 支持以及该功能的维护支持负载平衡服务中的功能。

注意

如果没有列出该功能，则 RHOSP 17.1 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 17.1 中的支持级别	
	Amphora Provider	OVN 供应商
ML2/OVS L3 HA	完全支持	不支持
ML2/OVS DVR	完全支持	不支持
ML2/OVS L3 HA + composable network node [1]	完全支持	不支持
ML2/OVS DVR + 可组合网络节点 [1]	完全支持	不支持
ML2/OVN L3 HA	完全支持	完全支持
ML2/OVN DVR	完全支持	完全支持
DPDK	不支持	不支持
SR-IOV	不支持	不支持
运行状况监视器	完全支持	不支持
Amphora active-standby	完全支持	不支持
终止 HTTPS 负载均衡器（使用 barbican）	完全支持	不支持
Amphora 备用池	仅限技术预览	不支持
UDP	完全支持	完全支持
备份成员	仅限技术预览	不支持
供应商框架	仅限技术预览	不支持
TLS 客户端身份验证	仅限技术预览	不支持
TLS 后端加密	仅限技术预览	不支持
Octavia 类型	完全支持	不支持

对象标签	完全支持	不支持
监听器 API 超时	完全支持	不支持
日志卸载	完全支持	不支持
VIP 访问控制列表	完全支持	不支持
availabilty 区域	完全支持	不支持
基于卷的 amphora	不支持	不支持

[1] 带有 OVS、元数据、DHCP、L3 和 Octavia (worker、运行状况监控和内务)的网络节点。

其他资源

- 第 2.1 节 “负载均衡服务提供商驱动程序”

2.3. 负载均衡服务软件要求

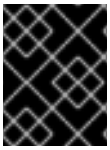
Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)需要配置以下核心 OpenStack 组件：

- Compute (nova)
- OpenStack Networking (neutron)
- Image (glance)
- 身份(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)对 RHOSP undercloud 有以下要求：

- 成功安装 undercloud。
- undercloud 上存在的负载平衡服务。
- 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 部署上启用负载平衡服务，您必须准备 undercloud。如果不这样做，则会导致 overcloud 安装被报告为成功，而无需运行负载平衡服务。

2.5. 负载均衡服务实例的 ACTIVE-STANDBY 拓扑的基础知识

当您部署 Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)时，您可以决定是否默认负载均衡器在用户创建它们时高度可用。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，以创建高度可用的负载均衡器和一个类别来创建独立负载均衡器。

默认情况下，为单一负载平衡服务(amphora)实例拓扑配置 amphora 供应商驱动程序，对高可用性(HA)的支持有限。但是，在实施 active-standby 拓扑时，您可以使负载平衡服务实例高度可用。

在这个拓扑中，负载均衡服务为每个负载均衡器引导一个主动和待机实例，并维护每个负载均衡器之间的会话持久性。如果活动实例变得不健康，实例会自动切换到待机实例，使它处于活动状态。负载均衡服务运行状况管理器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 active-standby 拓扑”

2.6. 负载均衡服务的部署后步骤

Red Hat OpenStack Platform (RHOSP)提供了一个工作流任务，可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，以提供以下部署后步骤作为 overcloud 部署的最后阶段：

- 配置证书和密钥。
- 在 amphorae 和 Load-balancing 服务 Controller worker 和健康管理器之间配置负载平衡管理网络。

Amphora 镜像

在部署负载均衡服务前，您必须在 undercloud 上安装 amphora 镜像：

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

在未预置备的服务器上，RHOSP director 会自动下载默认的 amphora 镜像，将其上传到 overcloud Image 服务(glance)，然后将负载平衡服务配置为使用此 amphora 镜像。在堆栈更新或升级过程中，director 将此镜像更新至最新的 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”

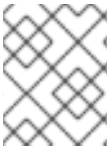
第 3 章 保护负载均衡服务

为了保护 Red Hat OpenStack 负载均衡服务(octavia)的不同组件之间的通信使用 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 “配置负载均衡服务证书和密钥”

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)通信。负载平衡服务使用双向 TLS 身份验证验证两端是否都信任。



注意

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

双向 TLS 身份验证涉及两个阶段。*阶段中，一个* Controller 进程（如负载均衡服务 worker 进程）连接到负载均衡服务实例，实例将其服务器证书提供给控制器。然后，控制器会根据控制器中存储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 *阶段，两个* Controller 将其客户端证书提供给负载均衡服务实例。然后，实例会根据存储在实例中的客户端 CA 证书来验证证书。如果成功验证此证书，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 “配置负载均衡服务证书和密钥”

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。

负载均衡服务内务控制器进程会在这些服务器证书接近其过期日期时自动轮转这些服务器证书。

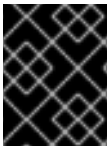
负载均衡服务控制器进程使用客户端证书。管理这些 TLS 证书的 human operator 通常授予很长时间，因为证书在云 control plane 上使用。

其他资源

- 第 3.3 节 “配置负载均衡服务证书和密钥”

3.3. 配置负载均衡服务证书和密钥

您可以配置 Red Hat OpenStack Platform (RHOSP) director 生成证书和密钥，也可以自行提供。配置 director 以自动创建所需的私有证书颁发机构并发布必要的证书。这些证书仅用于内部负载均衡服务(octavia)通信，且不会向用户公开。



重要

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

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** 目录中的现有证书来强制重新创建证书，并更新 overcloud。

如果您必须使用自己的证书和密钥，请完成以下步骤：

先决条件

- 阅读并理解，"Changing 负载平衡服务默认设置"。（"添加资源"中的链接"

流程

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

```
$ source ~/stackrc
```

3. 创建 YAML 自定义环境文件。

   示例

```
$ 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有 CA 密钥。
   - **OctaviaCaKeyPassphrase**:
     与上述私有 CA 密钥一起使用的密码短语。
   - **OctaviaClientCert**:
     Octavia CA 为控制器发布的客户端证书和未加密密钥。
   - **OctaviaGenerateCerts**:
     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。

     示例

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

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

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

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

MIIDmjCCAoKgAwIBAgIBATANBgkqhkiG9w0BAQsFADBcMQswCQYDVQQGEwJVUzEP

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

MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQDU771O8MTQV8RY

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

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

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

5. 运行 **openstack overcloud deploy** 命令，并包含核心 heat 模板、环境文件以及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具有优先权。

示例

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

其他资源

- 第 4.3 节 “更改负载均衡服务默认设置”

- *自定义 Red Hat OpenStack Platform 部署* 指南中的环境文件
  https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/17.1/html/customizing_your_red_hat_openstack_platform_deploy
  the-overcloud-with-the-orchestration-service#con_environment-files_understanding-heat-templates

- 在自定义 *Red Hat OpenStack Platform 部署* 指南中的 overcloud 创建中包括环境文件

第 4 章 安装和配置负载均衡服务

当您使用 RHOSP director 部署 Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)时，您可以决定使其虚拟机实例(amphorae)高度可用。当您想要对负载平衡服务进行配置更改时，您还可以使用 director。

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，它们是您的环境的一组计划。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。

先决条件

- 确保您的环境可以访问 octavia 镜像。

流程

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

  示例

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

注意

director 在堆栈更新或升级过程中将 amphorae 更新至最新的 amphora 镜像。

其他资源

- *使用 director 安装和管理 Red Hat OpenStack Platform* 指南中的 部署命令选项

4.2. 为负载均衡服务实例启用 ACTIVE-STANDBY 拓扑

在使用 Red Hat OpenStack Platform (RHOSP) director 实现主动拓扑时，您可以使负载平衡服务实例(amphorae)高度可用。director 使用编排服务(heat)模板，它们是您的环境的一组计划。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。

先决条件

- 确保为 Compute 服务启用反关联性。这是默认值。
- 至少三个 Compute 节点主机：
  - 两个 Compute 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，要成功故障转移到活跃负载均衡器的第三个主机。

流程

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

```
$ source ~/stackrc
```

3. 创建自定义 YAML 环境文件。

示例

```
$ 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，并包含核心 heat 模板、环境文件以及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具有优先权。

示例

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

验证

- 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，且不会发生等于 **SINGLE 的角色**。

其他资源

- *自定义 Red Hat OpenStack Platform 部署* 指南中的环境文件 https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/17.1/html/customizing_your_red_hat_openstack_platform_deploy the-overcloud-with-the-orchestration-service#con_environment-files_understanding-heat-templates

- 在自定义 *Red Hat OpenStack Platform 部署* 指南中的 overcloud 创建中包括环境文件

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，它们是您的环境的一组计划。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。

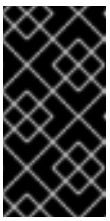
先决条件

- 通过咨询 undercloud 上的以下文件，确定 director 已用于部署负载平衡服务的 RHOSP 编配服务(heat)参数：

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

- 决定您要修改的参数。
  以下是几个示例：
  - **OctaviaControlNetwork**
    用于负载平衡管理网络的 neutron 网络的名称。
  - **OctaviaControlSubnetCidr**
    amphora 控制子网的子网，格式为 CIDR。

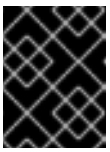
    您可以使用此参数将 amphora 控制子网指定为 IPv6 CIDR。在 **OctaviaControlSubnetCidr** 中使用 IPv6 CIDR 要求您在 **OctaviaControlSubnetPoolStart** 和 **OctaviaControlSubnetPoolEnd** 中设置 IPv6 地址。

    重要

    此功能将 IPv6 CIDR 用于负载平衡管理网络，在此发行版本中 *作为技术预览提供*，因此不受红帽完全支持。它只应用于测试，不应部署在生产环境中。

    有关技术预览功能的更多信息，请参阅覆盖范围详细信息。

    重要

    为 amphora 控制子网设置 CIDR 后，您无法修改这些值。这也意味着您无法从 IPv4 子网升级到 IPv6 子网。
  - **OctaviaMgmtPortDevName**
    用于带有 amphora 机器的 octavia worker/health-manager 间的 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 凭证文件：

```
$ source ~/stackrc
```

3. 创建自定义 YAML 环境文件。

示例

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

4. 您的环境文件必须包含关键字 **parameter_defaults**。将您的参数值对放在 **parameter_defaults** 关键字的后面。

示例

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

5. 运行部署命令，并包含核心 heat 模板、环境文件以及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具有优先权。

示例

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

其他资源

- *自定义 Red Hat OpenStack Platform 部署* 指南中的环境文件 https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/17.1/html/customizing_your_red_hat_openstack_platform_deploy the-overcloud-with-the-orchestration-service#con_environment-files_understanding-heat-templates
- 在自定义 *Red Hat OpenStack Platform 部署* 指南中的 overcloud 创建中包括环境文件

第 5 章 管理负载均衡服务实例日志

您可以启用租户流日志记录或阻止日志记录到 amphora 本地文件系统。您还可以将管理或租户流日志转发到一组容器中的 syslog 接收器，或者在您选择的端点处其他 syslog 接收器。

当您选择使用 TCP syslog 协议时，您可以在主端点失败时为管理和租户日志卸载指定一个或多个次要端点。

另外，您可以控制一系列其他日志功能，如设置 syslog facility 值、更改租户流日志格式，以及广泛的管理日志记录范围，使其包含来自内核和 cron 的源的日志。

5.1. 启用负载均衡服务实例管理日志卸载

默认情况下，负载均衡服务实例(amphorae)将日志存储在 systemd 日志中的本地机器上。但是，您可以指定 amphorae 卸载日志到 syslog 接收器，以汇总管理日志。日志卸载可让管理员进入日志的一个位置，并在轮转 amphorae 时保留日志。

流程

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

```
$ source ~/stackrc
```

3. 创建自定义 YAML 环境文件。

   示例

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

4. 在 **parameter_defaults** 下的 YAML 环境文件中，将 **OctaviaLogOffload** 设置为 **true**。

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

注意

amphorae 卸载管理日志默认使用 **local1** 的 syslog 工具值，除非您使用 **OctaviaAdminLogFacility** 参数指定另一个值。有效值为 0 iwl-wagon7。

示例

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

5. amphorae 仅转发与负载均衡器相关的管理日志，如 haproxy admin 日志、keepalived 和 amphora 代理日志。如果要配置 amphorae 从 amphorae 发送所有管理日志，如内核、系统和安全日志，请将 **OctaviaForwardAllLogs** 设置为 **true**。

   示例

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

6. 选择日志协议：UDP （默认）或 TCP。
   如果主端点失败，则 amphorae 只有在日志协议是 TCP 时将其日志发送到二级端点。

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

7. amphorae 使用由编排服务(heat)定义的一组默认容器，其中包含侦听日志消息的 syslog 接收器。如果要使用不同的端点集合，您可以使用 **OctaviaAdminLogTargets** 参数指定它们。
   为租户流日志卸载配置的端点可以是用于管理日志卸载的端点。

   另外，如果您的日志卸载协议是 TCP，如果第一个端点无法访问，则 amphorae 将按照您列出它们的顺序尝试额外的端点，直到连接成功为止。

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

   示例

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

8. 默认情况下，当您启用日志卸载时，租户流日志也会被卸载。
   如果要禁用租户流日志卸载，请将 **OctaviaConnectionLogging** 设置为 **false**。

   示例

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

9. 运行部署命令，并包含核心 heat 模板、环境文件以及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具有优先权。

示例

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

验证

- 除非使用 **OctaviaAdminLogTargets** 或 **OctaviaTenantLogTargets** 指定特定端点，否则 amphorae 卸载日志位于与其他 RHOSP 日志相同的位置(**/var/log/containers/octavia-amphorae/**)。
- 检查以下日志文件是否存在适当的位置：
  - **用于管理日志的 Octavia-amphora.log**-- 日志文件。
  - （如果启用） **octavia-tenant-traffic.log**-- Log 文件用于租户流量流日志。

其他资源

- 第 5.5 节 “负载均衡服务实例日志的 Heat 参数”
- *自定义 Red Hat OpenStack Platform 部署* 指南中的环境文件 https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/17.1/html/customizing_your_red_hat_openstack_platform_deploy the-overcloud-with-the-orchestration-service#con_environment-files_understanding-heat-templates
- 在自定义 *Red Hat OpenStack Platform 部署* 指南中的 overcloud 创建中包括环境文件

5.2. 为负载均衡服务实例启用租户流日志卸载

默认情况下，负载均衡服务实例(amphorae)将日志存储在 systemd 日志中的本地机器上。但是，您可以指定备用 syslog 接收器端点。由于租户流日志的大小会根据租户连接的数量而增加，因此请确保备用 syslog 接收器包含足够的磁盘空间。

默认情况下，在启用管理日志卸载时，会自动启用租户流日志卸载。要关闭租户流日志卸载，请将 **OctaviaConnectionLogging** 参数设置为 **false**。

重要

租户流日志记录可能会生成大量 syslog 信息，具体取决于接收负载均衡器的连接数量。租户流日志记录为每个与负载均衡器的连接生成一个日志条目。监控日志卷并根据负载均衡器管理的预期连接数量正确配置 syslog 接收器。

流程

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

```
$ source ~/stackrc
```

3. 找到设置 **OctaviaConnectionLogging** 参数的环境文件：

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

4. 如果没有找到该文件，请创建一个环境文件：

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

5. 将 **OctaviaLogOffload** 和 **OctaviaConnectionLogging** 参数添加到环境文件的 **parameter_defaults** 部分，并将值设为 **true** ：

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

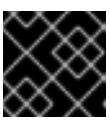
注意

amphorae 使用 syslog 工具默认值 **local0** 卸载租户流日志，除非您使用 **OctaviaTenantLogFacility** 参数指定另一个值。有效值为 0 iwl-wagon7。

6. 可选：要更改 amphorae 用于租户和管理日志卸载的默认端点，请分别使用 **OctaviaTenantLogTargets** 和 **OctaviaAdminLogTargets**。amphorae 使用一组默认容器，其中包含侦听日志消息的 syslog 接收器。
另外，如果您的日志卸载协议是 TCP，如果第一个端点无法访问，则 amphorae 将按照您列出它们的顺序尝试额外的端点，直到连接成功为止。

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

7. 运行部署命令，并包括核心 heat 模板、环境文件以及您修改的自定义环境文件。



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具有优先权。

```
$ 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

- 除非使用 **OctaviaAdminLogTargets** 或 **OctaviaTenantLogTargets** 指定特定端点，否则 amphorae 卸载日志位于与其他 RHOSP 日志相同的位置(**/var/log/containers/octavia-amphorae/**)。
- 检查以下日志文件是否存在适当的位置：
  - **用于管理日志的 Octavia-amphora.log**-- 日志文件。
  - **Octavia-tenant-traffic.log**-- Log 文件用于租户流量流日志。

其他资源

- 第 5.5 节 "负载均衡服务实例日志的 Heat 参数"
- *自定义 Red Hat OpenStack Platform 部署* 指南中的环境文件 https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/17.1/html/customizing_your_red_hat_openstack_platform_deploy the-overcloud-with-the-orchestration-service#con_environment-files_understanding-heat-templates
- 在自定义 *Red Hat OpenStack Platform 部署* 指南中的 overcloud 创建中包括环境文件
- 第 5.6 节 "负载均衡服务实例租户日志格式"

5.3. 禁用负载均衡服务实例租户流日志记录

当您启用管理日志卸载时，会自动启用负载平衡服务实例(amphorae)的租户流日志卸载。

要启用管理日志卸载并禁用租户流日志记录，您必须将 **OctaviaConnectionLogging** 参数设置为 **false**。

当 **OctaviaConnectionLogging** 参数为 **false** 时，amphorae 不将租户流日志写入 amphorae 中的磁盘，也不会将任何日志卸载到 syslog 接收器，侦听其他位置。

流程

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

```
$ source ~/stackrc
```

3. 找到配置 amphora 日志记录的 YAML 自定义环境文件。

   示例

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

4. 在自定义环境文件的 **parameter_defaults** 下，将 **OctaviaConnectionLogging** 设置为 **false**。

示例

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

5. 运行部署命令，并包含核心 heat 模板、环境文件以及您将 **OctaviaConnectionLogging** 设置为 **true** 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具有优先权。

示例

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

验证

- 除非使用 **OctaviaAdminLogTargets** 或 **OctaviaTenantLogTargets** 指定特定端点，否则 amphorae 卸载日志位于与其他 RHOSP 日志相同的位置(**/var/log/containers/octavia-amphorae/**)。
- 检查 **octavia-tenant-traffic.log** 的 *absence* 的适当位置。

其他资源

- *自定义 Red Hat OpenStack Platform 部署* 指南中的环境文件 https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/17.1/html/customizing_your_red_hat_openstack_platform_deploy the-overcloud-with-the-orchestration-service#con_environment-files_understanding-heat-templates
- 在自定义 *Red Hat OpenStack Platform 部署* 指南中的 overcloud 创建中包括环境文件

5.4. 禁用负载均衡服务实例本地日志存储

即使配置负载平衡服务实例(amphorae)以卸载管理和租户流日志，但 amphorae 将继续将这些日志写入 amphorae 中的磁盘。要提高负载均衡器的性能，您可以在本地停止日志记录。

重要

如果您在本地禁用日志记录，您还会禁用 amphora 中的所有日志存储，包括内核、系统和安全日志记录。

注意

如果您禁用了本地日志存储，并且 **OctaviaLogOffload** 参数设置为 false，请确保将 **OctaviaConnectionLogging** 设置为 false 以改进负载均衡性能。

流程

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

```
$ source ~/stackrc
```

3. 创建自定义 YAML 环境文件。

示例

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

4. 在 **parameter_defaults** 下的环境文件中，将 **OctaviaDisableLocalLogStorage** 设置为 **true**。

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

5. 运行部署命令，并包含核心 heat 模板、环境文件以及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具有优先权。

示例

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

验证

- 在 amphora 实例中，检查写入日志文件的位置，并验证没有写入新的日志文件。

其他资源

- *自定义 Red Hat OpenStack Platform 部署* 指南中的环境文件 https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/17.1/html/customizing_your_red_hat_openstack_platform_deploy the-overcloud-with-the-orchestration-service#con_environment-files_understanding-heat-templates
- 在自定义 *Red Hat OpenStack Platform 部署* 指南中的 overcloud 创建中包括环境文件

5.5. 负载均衡服务实例日志的 HEAT 参数

当您要配置负载平衡服务实例(amphora)日志记录时，您可以为控制日志记录并运行 **openstack overcloud deploy** 命令的一个或多个编排服务(heat)参数设置值。

这些用于 amphora 日志记录的 heat 参数允许您控制日志卸载等功能，定义自定义端点来卸载日志，为日志设置 syslog facility 值等。

表 5.1. 所有日志的 Heat 参数

参数	默认	描述
OctaviaLogOffload	**false**	为 **true** 时，实例会卸载其日志。如果没有指定端点，则默认情况下，实例会将其日志卸载到与其他 RHOSP 日志相同的位置(**/var/log/containers/octavia-amphorae/**)。
OctaviaDisableLocalLogStorage	**false**	为 **true** 时，实例不会将日志存储在实例主机文件系统中。这包括所有内核、系统和安全日志。
OctaviaForwardAllLogs	**false**	为 **true** 时，实例将所有日志消息转发到管理日志端点，包括 cron 和内核日志等非负载平衡相关日志。 对于识别 **OctaviaForwardAllLogs** 的实例，还必须启用 **OctaviaLogOffload**。

表 5.2. 用于 admin 日志记录的 Heat 参数

参数	默认	描述
OctaviaAdminLogTargets	无值。	以逗号分隔的 syslog 端点列表(<host>:<port>)来接收管理日志消息。 这些端点可以是运行在指定端口上侦听日志消息的进程的容器、虚拟机或物理主机。 当 **OctaviaAdminLogTargets** 不存在时，实例会将日志卸载到与其他 RHOSP 日志(**/var/log/containers/octavia-amphorae/**octavia-amphorae/)相同的位置。
OctaviaAdminLogFacility	**1**	0 到 7 之间的数字，它是用于管理日志消息的 syslog "LOG_LOCAL" 工具。

表 5.3. 租户流日志记录的 Heat 参数

参数	默认	描述

参数	默认	描述
OctaviaConnectionLogging	**true**	为 **true** 时，会记录租户连接流。 当 **OctaviaConnectionLogging** 为 false 时，无论 **OctaviaLogOffload** 设置如何，amphorae 会停止日志记录租户连接。**OctaviaConnectionLogging** 禁用本地租户流日志存储，如果启用了日志卸载，它不会转发租户流日志。
OctaviaTenantLogTargets	无值。	以逗号分隔的 syslog 端点列表(<host>:<port>)来接收租户流量流日志消息。 这些端点可以是运行在指定端口上侦听日志消息的进程的容器、虚拟机或物理主机。 当 **OctaviaTenantLogTargets** 不存在时，实例会将日志卸载到与其他 RHOSP 日志相同的位置(**/var/log/containers/octavia-amphorae/**)，这些容器由 RHOSP director 定义的容器。
OctaviaTenantLogFacility	**0**	0 到 7 之间的数字，即 syslog "LOG_LOCAL" 工具，用于租户流量流日志消息。
OctaviaUserLogFormat	**"{{ '{{' }} project_id {{ '}}' }} {{ '{{' }} lb_id {{ '}}' }} %f %ci %cp %t %{+Q}r %ST %B %U % [ssl_c_verify] %{+Q} [ssl_c_s_dn] %b %s %Tt %tsc"**	租户流量流日志的格式。 字母数字字符代表特定的 octavia 字段，大括号({})是替换变量。

其他资源

- 在自定义 *Red Hat OpenStack Platform 部署* 指南中的 overcloud 创建中包括环境文件

5.6. 负载均衡服务实例租户日志格式

负载均衡服务实例(amphorae)的租户流日志的日志格式是 HAProxy 日志格式。两个例外是 **project_id** 和 **lb_id** 变量，其值由 amphora 提供程序驱动程序提供。

示例

以下是 rsyslog 作为 syslog 接收器的示例日志条目：

> Jun 12 00:44:13 amphora-3e0239c3-5496-4215-b76c-6abbe18de573 haproxy[1644]: 5408b89aa45b48c69a53dca1aaec58db fd8f23df-960b-4b12-ba62-2b1dff661ee7 261ecfc2-9e8e-

> 4bba-9ec2-3c903459a895 172.24.4.1 41152 12/Jun/2019:00:44:13.030 "GET / HTTP/1.1" 200 76 73 - "" e37e0e04-68a3-435b-876c-cffe4f2138a4 6f2720b3-27dc-4496-9039-1aafe2fee105 4 --

备注

- hyphen (-)表示所有未知或不可用于连接的字段。
- 前面示例日志条目中的前缀源自 rsyslog 接收器，不是来自 amphora 的 syslog 消息的一部分：

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

默认

默认的 amphora 租户流日志格式为：

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

有关格式的描述，请参考下表。

表 5.4. 租户流日志格式变量定义的数据变量。

变量	类型	字段名称
{{project_id}}	UUID	项目 ID （来自 amphora 提供者驱动程序的子变量）
{{lb_id}}	UUID	负载均衡器 ID （来自 amphora 提供者驱动程序的子变量）
%f	字符串	**frontend_name**
%ci	IP 地址	**client_ip**
%cp	数字	**client_port**
%t	date	**date_time**
%ST	数字	**status_code**
%B	数字	**bytes_read**
%U	数字	**bytes_uploaded**
%ssl_c_verify	布尔值	**client_certificate_verify** (0 or 1)
%ssl_c_s_dn	字符串	**client_certificate_distinguised_name**
%b	字符串	**pool_id**
%s	字符串	**member_id**

变量	类型	字段名称
%Tt	数字	**processing_time** （毫秒）
%tsc	字符串	**termination_state** （具有 Cookie 状态）

其他资源

- *HAProxy 文档*中的自定义日志格式

第 6 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。当用户请求负载均衡器时，他们可以指定使用其中一个定义的类别构建负载均衡器。您可以为每个负载平衡供应商驱动程序定义类别，该驱动程序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型：

1. 决定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 配置文件。
3. 创建类别。

- 第 6.1 节 “列出负载均衡服务提供商功能”
- 第 6.2 节 “定义类别配置集”
- 第 6.3 节 “创建负载均衡服务类型”

6.1. 列出负载均衡服务提供商功能

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。

流程

1. 提供可让您使用 RHOSP admin 角色访问 overcloud 的凭据文件。

   示例

   ```
   $ source ~/my_overcloudrc
   ```

2. 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。

   示例

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

3. 请注意您要包含在您要创建的类别中的功能的名称。

其他资源

- 第 6.2 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

6.2. 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文件创建用户指定的类别，以创建负载平衡器。

先决条件

- 您必须知道哪个负载平衡提供程序及其要包含在类别配置文件中的功能。

流程

1. 提供可让您使用 RHOSP admin 角色访问 overcloud 的凭据文件。

   示例

   ```
   $ source ~/my_overcloudrc
   ```

2. 创建 flavor 配置集：

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

   示例

   ```
   $ 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 提供程序创建了一个类别配置文件。在类别中指定此配置文件时，用户使用该类别创建的负载平衡器是单个 amphora 负载平衡器。

验证

- 在创建类别配置集时，负载均衡服务会将类别值与供应商验证，以确保供应商可以支持您指定的功能。

其他资源

- 第 6.3 节 “创建负载均衡服务类型”
- *命令行界面参考中的*LoadBalancer flavorprofile create

6.3. 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)创建一个面向用户的类别。您分配给类别的名称是用户在创建负载均衡器时指定的值。

先决条件

- 您必须已创建了 flavor 配置集。

流程

1. 提供可让您使用 RHOSP admin 角色访问 overcloud 的凭据文件。

   示例

   ```
   $ source ~/my_overcloudrc
   ```

2. 创建类别：

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

   提示

   提供一个详细的描述，以便用户可以了解您提供的类别的功能。

   示例

   ```
   $ 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，他们会创建一个负载均衡器，它使用一个负载平衡服务实例(amphora)，且不具有高可用性。

注意

禁用的类别仍然对用户可见，但用户无法使用 disabled 类别来创建负载平衡器。

其他资源

- 第 6.2 节 “定义类别配置集”
- *命令行界面参考中的*LoadBalancer flavor create

第 7 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控器。

7.1. 负载均衡管理网络

Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)通过称为 *负载平衡管理网络的项目网络监控负载均衡器*。运行负载平衡服务的主机必须具有接口才能连接到负载平衡管理网络。支持的接口配置可用于带有 Open Virtual Network 机制驱动程序(ML2/OVN)或 Open vSwitch 机制驱动程序(ML2/OVS)的 neutron Modular Layer 2 插件。将接口与其他机制驱动程序搭配使用。

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。

默认接口默认命名为 **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 在部署过程中自动配置网络服务端口和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 端口注册额外的数据。网络服务将此接口识别为属于端口，并且配置 OVS，以便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，允许负载平衡服务控制器在 TCP 端口 9443 上与其虚拟机实例(amphorae)通信，并允许来自 amphorae 的 heartbeat 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来允许负载平衡服务和负载均衡器之间的通信。

7.2. 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 失败时启动故障转移和替换。每当发生故障转移时，负载均衡服务会在 **/var/log/containers/octavia** 中的控制器上记录相应的健康管理器日志中的故障切换。

使用日志分析来监控故障转移趋势，以便在早期解决问题。网络服务(neutron)连接问题、服务攻击和计算服务(nova)出现故障等问题通常会导致负载均衡器的故障转移率更高。

7.3. 负载均衡服务池成员监控

负载均衡服务(octavia)使用底层负载平衡子系统中的健康信息来确定负载平衡池的成员的健康状况。健康信息流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，您必须以固定间隔轮询健康信息。

7.4. 负载均衡器置备状态监控

您可以监控负载均衡器的置备状态，并在置备状态为 **ERROR** 时发送警报。当应用定期更改池并进入几个 **PENDING** 阶段时，不要配置警报来触发。

负载均衡器对象的置备状态反映了 control plane 能够联系并成功置备创建、更新和删除请求。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，负载均衡器可能会具有 **ERROR** 状态，但操作状态为 **ONLINE**。这可能是因为一个 Networking (neutron)失败，阻止最后一次请求更新负载均衡器配置成功完成。在这种情况下，负载均衡器将继续处理通过负载均衡器的流量，但可能还没有应用最新的配置更新。

7.5. 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操作状态。

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，并从云外部监控它们。外部监控服务指示负载平衡服务(octavia)以外的故障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。

7.6. 关于负载均衡服务运行状况监控器

负载均衡服务(octavia)运行状况监视器是一个进程，在每个后端成员服务器上定期进行健康检查，以预先检测失败的服务器，并临时从池中拉取它们。

如果运行状况监视器检测到失败的服务器，它将从池中移除服务器，并将成员标记为 **ERROR**。在解决了服务器并再次正常工作后，运行状况监视器会自动将成员的状态从 **ERROR** 改为 **ONLINE**，并恢复将流量传递给它。

在生产负载均衡器中始终使用运行状况监控器。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。

有几个类型的运行状况监视器，如这里所述：

HTTP

默认情况下，在应用服务器上探测 / 路径。

HTTPS

与 HTTP 运行状况监视器完全相同，但使用 TLS 后端服务器。
如果服务器执行客户端证书验证，HAProxy 没有有效的证书。在这些情况下，TLS-HELLO 健康监控是一个替代方案。

TLS-HELLO

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

PING

发送定期 ICMP ping 请求到后端服务器。
您必须配置后端服务器，以允许 PING，以便这些健康检查通过。

重要

PING 运行状况监视器仅检查成员是否可访问并响应 ICMP 回显请求。PING 运行状况监视器不会检测实例上运行的应用是否健康。仅在 ICMP 回显请求是有效的健康检查时使用 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，在 TCP 握手后关闭连接而不发送任何数据。

UDP-CONNECT

执行基本的 UDP 端口连接。
如果成员服务器上没有启用 Destination Unreachable (ICMP type 3)，或者安全规则阻断它，则 UDP-CONNECT 健康监控器可能无法正常工作。在这些情况下，成员服务器可能会在实际关闭时被标记为 **ONLINE** 操作状态。

7.7. 创建负载均衡服务运行状况监控器

使用负载均衡服务(octavia)运行状况监视器，以避免对用户造成服务中断。运行状况监视器在每个后端服务器上运行定期健康检查，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。

   示例

   ```
   $ source ~/overcloudrc
   ```

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

   - 所有运行状况监控类型都需要以下可配置的参数：

     <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，后端服务器必须通过的健康检查数量必须再次被视为 up。

- 另外，HTTP 运行状况监控类型还需要以下参数，该参数会被默认设置：

--url-path

应从后端服务器检索的 URL 的路径部分。默认情况下，这是 /。

--http-method

用于检索 **url_path** 的 HTTP 方法。默认情况下，这是 **GET**。

--expected-codes

指明 OK 健康检查的 HTTP 状态代码列表。默认情况下，这是 **200**。

示例

```
$ 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

- *命令行界面参考*中的 loadbalancer healthmonitor create 部分

7.8. 修改负载均衡服务运行状况监控器

当您要将探测发送到成员、连接超时间隔、请求的 HTTP 方法等时，您可以修改负载平衡服务(octavia)运行状况监视器的配置。

流程

1. 提供您的凭据文件。

示例

```
$ source ~/overcloudrc
```

2. 修改运行状况监控器(**my-health-monitor**)。
在本例中，用户更改健康检查向成员发送探测之间等待的时间（以秒为单位）。

示例

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

验证

- 运行 **openstack loadbalancer healthmonitor show** 命令确认您的配置更改。

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

其他资源

- *命令行界面参考* 中设置的 LoadBalancer healthmonitor
- *命令行界面参考*中的 loadbalancer healthmonitor show 部分

7.9. 删除负载均衡服务运行状况监控器

您可以删除负载均衡服务(octavia)运行状况监控器。

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。

   示例

```
$ source ~/overcloudrc
```

2. 删除运行状况监视器(**my-health-monitor**)。

   示例

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

验证

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

其他资源

- *命令行界面参考*中的 loadbalancer healthmonitor delete 部分

7.10. 负载均衡服务 HTTP 健康监视器的最佳实践

当您编写在 web 应用程序中生成健康检查的代码时，请使用以下最佳实践：

- 运行状况监控器 **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，以指示健康的服务器，除非您指定了备用 **expected-** code。

- 健康检查执行足够的内部检查，以确保应用程序健康且不再工作。确保满足应用程序的以下条件：
  - 任何所需的数据库或其他外部存储连接都已启动并运行。
  - 负载对于应用程序运行的服务器可以接受。
  - 您的站点不处于维护模式。
  - 特定于应用程序的测试可以正常工作。
- 健康检查生成的页面的大小应该小：
  - 它以亚秒的间隔返回。
  - 它不会降低应用服务器中的负载。
- 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时很有用。在健康监控器 **url-path** 中生成页面的代码将这个 cron 作业的结果合并到其执行的测试中。
- 因为负载均衡服务只处理返回的 HTTP 状态代码，并且由于健康检查会频繁运行，所以您可以使用 **HEAD** 或 **OPTIONS** HTTP 方法跳过整个页面。

第 8 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 8.1 节 “使用运行状况监控器创建 HTTP 负载均衡器”
- 第 8.2 节 “创建使用浮动 IP 的 HTTP 负载均衡器”
- 第 8.3 节 “使用会话持久性创建 HTTP 负载均衡器”

8.1. 使用运行状况监控器创建 HTTP 负载均衡器

对于与 Red Hat OpenStack Platform Networking 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。创建一个运行状况监控器，以确保后端成员保持可用。

先决条件

- 可以从互联网访问的共享外部(public)子网。

流程

1. 提供您的凭据文件。

   示例

   ```
   $ source ~/overcloudrc
   ```

2. 在公共子网(**public_subnet**)上创建一个负载均衡器(**lb1**)。

   注意

   括号中的值是此流程中的示例命令中使用的示例值。使用适合您的站点的值替换这些示例值。

   示例

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

3. 在端口 (**80**) 上创建一个监听器 (**listener1**)。

   示例

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

4. 验证监听器的状态。

   示例

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

在继续下一步之前，请确保状态为 **ACTIVE**。

5. 创建侦听器默认池(**pool1**)。

示例

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

6. 在池(**pool1**)上创建一个类型为(**HTTP**)的运行状况监视器(**healthmon1**)，以连接到后端服务器并测试路径(/)。
建议使用健康检查，但不是必需的。如果没有定义运行状况监控器，则假定成员服务器为 **ONLINE**。

示例

```
$ openstack loadbalancer healthmonitor create --name healthmon1 \
--delay 15 --max-retries 4 --timeout 10 --type HTTP --url-path / pool1
```

7. 在专用子网(**private_subnet**)上添加负载均衡器成员（**192.0.2.10** 和 **192.0.2.11**）到默认的池。

示例

在本例中，后端服务器 **192.0.2.10** 和 **192.0.2.11** 分别命名为 **member1** 和 **member2** ：

```
$ openstack loadbalancer member create --name member1 --subnet-id \
private_subnet --address 192.0.2.10 --protocol-port 80 pool1

$ openstack loadbalancer member create --name member2 --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：

示例

```
$ 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. 当健康监控器存在并正常运行时，您可以检查每个成员的状态。工作成员(**member1)**具有其 **operating_status** 的 **ONLINE** 值。

示例

```
$ openstack loadbalancer member show pool1 member1
```

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                | member1                              |
| 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

8.2. 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，您可以保留对分配的 IP 的控制，如果您需要移动、销毁或重新创建负载均衡器，这是必需的。最好还要创建一个运行状况监视器，以确保您的后端成员仍然可用。

注意

浮动 IP 无法使用 IPv6 网络。

先决条件

- 用于负载均衡器 VIP 的浮动 IP。
- 可以从互联网访问的 Red Hat OpenStack Platform Networking 服务(neutron)共享外部(public)子网，以用于浮动 IP。

流程

1. 提供您的凭据文件。

   示例

   ```
   $ source ~/overcloudrc
   ```

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

   注意

   括号中的值是此流程中的示例命令中使用的示例值。使用适合您的站点的值替换这些示例值。

   示例

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

3. 在第 2 步的输出中，记录 **load_balancer_vip_port_id** 的值，因为您必须在后续步骤中提供它。

4. 在端口 (**80**) 上创建一个监听器 (**listener1**)。

   示例

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

5. 创建侦听器默认池(**pool1**)。

   示例

   本例中的命令会创建一个 HTTP 池，它使用专用子网，其中包含在 TCP 端口 80 上托管非安全 HTTP 应用程序的后端服务器：

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

6. 在池(**pool1**)上创建一个类型为(**HTTP**)的运行状况监视器(**healthmon1**)，以连接到后端服务器并测试路径(/)。
   建议使用健康检查，但不是必需的。如果没有定义运行状况监控器，则假定成员服务器为 **ONLINE**。

   示例

```
$ openstack loadbalancer healthmonitor create --name healthmon1 \
--delay 15 --max-retries 4 --timeout 10 --type HTTP --url-path / pool1
```

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

   示例

   在本例中，后端服务器 **192.0.2.10** 和 **192.0.2.11** 分别命名为 **member1** 和 **member2** ：

```
$ openstack loadbalancer member create --name member1 --subnet-id \
private_subnet --address 192.0.2.10 --protocol-port 80 pool1

$ openstack loadbalancer member create --name member2 --subnet-id \
private_subnet --address 192.0.2.11 --protocol-port 80 pool1
```

8. 在共享外部子网(**公共**)上创建一个浮动 IP 地址。

   示例

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

9. 在 step 8 输出中记录 **floating_ip_address** 的值，因为您必须在后续步骤中提供它。

10. 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。

    示例

```
$ 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**)验证跨负载均衡器的 HTTP 流量流。

   示例

```
$ 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. 当健康监控器存在并正常运行时，您可以检查每个成员的状态。工作成员(**member1**)具有其 **operating_status** 的 **ONLINE** 值。

示例

```
$ openstack loadbalancer member show pool1 member1
```

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                | member1                              |
| 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 floating

8.3. 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进入时，负载均衡器会将来自同一客户端的后续请求定向到同一后端服务器。会话持久性通过节省时间和内存来优化负载平衡。

先决条件

- 可以从互联网访问的共享外部(public)子网。
- 您要负载平衡的网络流量的非安全 Web 应用程序启用了 Cookie。

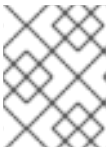
流程

1. 提供您的凭据文件。

示例

```
$ source ~/overcloudrc
```

2. 在公共子网(**public_subnet**)上创建一个负载均衡器(**lb1**)。



注意

括号中的值是此流程中的示例命令中使用的示例值。使用适合您的站点的值替换这些示例值。

示例

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

3. 在端口 (**80**) 上创建一个监听器 (**listener1**)。

示例

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

4. 创建监听器默认池(**pool1**)，用于定义 Cookie 上的会话持久性(**PHPSESSIONID**)。

示例

本例中的命令会创建一个 HTTP 池，它使用专用子网，其中包含在 TCP 端口 80 上托管非安全 HTTP 应用程序的后端服务器：

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

5. 在池(**pool1**)上创建一个类型为(**HTTP**)的运行状况监视器(**healthmon1**)，以连接到后端服务器并测试路径(/)。
建议使用健康检查，但不是必需的。如果没有定义运行状况监控器，则假定成员服务器为 **ONLINE**。

示例

```
$ openstack loadbalancer healthmonitor create --name healthmon1 \
--delay 15 --max-retries 4 --timeout 10 --type HTTP --url-path / pool1
```

6. 在专用子网(**private_subnet**)上添加负载均衡器成员（**192.0.2.10** 和 **192.0.2.11**）到默认的池。

示例

在本例中，后端服务器 **192.0.2.10** 和 **192.0.2.11** 分别命名为 **member1** 和 **member2** ：

```
$ openstack loadbalancer member create --name member1 --subnet-id \
private_subnet --address 192.0.2.10 --protocol-port 80 pool1

$ openstack loadbalancer member create --name member2 --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：

示例

```
$ 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. 当健康监控器存在并正常运行时，您可以检查每个成员的状态。工作成员(**member1)**具有其 **operating_status** 的 **ONLINE** 值。

示例

```
$ openstack loadbalancer member show pool1 member1
```

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                | member1                              |
| 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 9 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 HTTP (HTTPS)网络流量。

9.1. 关于非终止的 HTTPS 负载均衡器

非终止的 HTTPS 负载均衡器实际上类似于通用 TCP 负载均衡器：负载平衡器将来自 Web 客户端的原始 TCP 流量转发到 HTTPS 连接通过 Web 客户端终止的后端服务器。虽然非终止的 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进行低负载均衡器资源利用率。

9.2. 创建非终止的 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，通常称为 *HTTPS 传递*，您可以使用负载均衡器监听程序的 HTTPS 协议。

先决条件

- 可以从互联网访问的共享外部(public)子网。

流程

1. 提供您的凭据文件。

   示例

   ```
   $ source ~/overcloudrc
   ```

2. 在公共子网(**public_subnet**)上创建一个负载均衡器(**lb1**)。

   注意

   括号中的值是此流程中的示例命令中使用的示例值。使用适合您的站点的值替换这些示例值。

   示例

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

3. 在端口(**443**)上创建一个侦听器（监听程序**1**）。

   示例

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

4. 创建侦听器默认池(**pool1**)。

   示例

   本例中的命令会创建一个 HTTPS 池，它使用专用子网，其中包含在 TCP 端口 443 上托管使用 TLS 加密 Web 应用程序配置的 HTTPS 应用程序的后端服务器：

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

5. 在池(**pool1**)上创建一个运行状况监视器(**healthmon1**)，类型为(**TLS-HELLO**)，以连接到后端服务器并测试路径(/)。
   建议使用健康检查，但不是必需的。如果没有定义运行状况监控器，则假定成员服务器为 **ONLINE**。

   示例

   ```
   $ openstack loadbalancer healthmonitor create --name healthmon1 \
   --delay 15 --max-retries 4 --timeout 10 --type TLS-HELLO \
   --url-path / pool1
   ```

6. 在专用子网(**private_subnet**)上添加负载均衡器成员（**192.0.2.10** 和 **192.0.2.11**）到默认的池。

   示例

   在本例中，后端服务器 **192.0.2.10** 和 **192.0.2.11** 分别命名为 **member1** 和 **member2** ：

   ```
   $ openstack loadbalancer member create --name member1 --subnet-id \
   private_subnet --address 192.0.2.10 --protocol-port 443 pool1

   $ openstack loadbalancer member create --name member2 --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。

   示例

   ```
   $ 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. 当健康监控器存在并正常运行时，您可以检查每个成员的状态。

示例

工作成员(**member1)**具有其 **operating_status** 的 **ONLINE** 值。

```
$ openstack loadbalancer member show pool1 member1
```

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                | member1                              |
| 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

- *使用密钥管理器服务管理* secret 指南

- *命令行界面参考*中的 LoadBalancer

9.3. 关于 TLS 终止 HTTPS 负载均衡器

当实施 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。负载均衡器终止 TLS 会话，并将解密的请求转发到后端服务器。当您终止负载均衡器上的 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。

9.4. 创建 TLS 终止的 HTTPS 负载均衡器

当您使用 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。最好还要创建一个运行状况监视器，以确保您的后端成员仍然可用。

先决条件

- 可以从互联网访问的共享外部(public)子网。
- TLS 公钥加密配置有以下特征：
  - TLS 证书、密钥和中间证书链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独立文件中。
  - 密钥和证书是 PEM 编码的。
  - 中间证书链包含多个使用 PEM 编码和串联的证书。
- 您必须配置负载均衡服务(octavia)以使用 Key Manager 服务(barbican)。如需更多信息，*请参阅使用密钥管理器服务管理 secret* 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(**server.p12**)中。

注意

括号中的值是此流程中的示例命令中使用的示例值。使用适合您的站点的值替换这些示例值。

示例

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

注意

如果您的密码保护 PKCS12 文件，则以下步骤无法正常工作。

2. 提供您的凭据文件。

示例

```
$ source ~/overcloudrc
```

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

示例

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

4. 在公共子网(**public_subnet**)上创建一个负载平衡器(**lb1**)。

示例

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

5. 创建 **TERMINATED_HTTPS** 侦听器(**listener**1)，并将机密资源引用为监听器的默认 TLS 容器。

示例

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

6. 创建一个池(**pool1**)，并使它成为监听器的默认池。

示例

本例中的命令会创建一个 HTTP 池，它使用专用子网，其中包含在 TCP 端口 80 上托管非安全 HTTP 应用程序的后端服务器：

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

7. 在池(**pool1**)上创建一个类型为(**HTTP**)的运行状况监视器(**healthmon1**)，以连接到后端服务器并测试路径(/)。
建议使用健康检查，但不是必需的。如果没有定义运行状况监控器，则假定成员服务器为 **ONLINE**。

示例

```
$ openstack loadbalancer healthmonitor create --name healthmon1 \
--delay 15 --max-retries 4 --timeout 10 --type HTTP --url-path / pool1
```

8. 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。

示例

在本例中，后端服务器 **192.0.2.10** 和 **192.0.2.11** 分别命名为 **member1** 和 **member2** ：

```
$ openstack loadbalancer member create --name member1 --subnet-id \
private_subnet --address 192.0.2.10 --protocol-port 443 pool1

$ openstack loadbalancer member create --name member2 --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。

示例

```
$ 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. 当健康监控器存在并正常运行时，您可以检查每个成员的状态。

示例

```
$ openstack loadbalancer member show pool1 member1
```

工作成员(**member1)**在其 **operating_status** 具有 **ONLINE** 值：

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                | member1                           |
| 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

- *使用密钥管理器服务管理* secret 指南
- *命令行界面参考*中的 LoadBalancer

9.5. 使用 SNI 创建 TLS 终止 HTTPS 负载均衡器

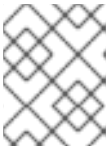
对于使用 Server Name Indication (SNI)技术的 TLS 终止 HTTPS 负载均衡器，单个侦听器可以包含多个 TLS 证书，并让负载均衡器知道使用共享 IP 时要存在哪些证书。最好还要创建一个运行状况监视器，以确保您的后端成员仍然可用。

先决条件

- 可以从互联网访问的共享外部(public)子网。
- TLS 公钥加密配置有以下特征：
  - 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**。
  - 密钥和证书是 PEM 编码的。
- 您必须配置负载均衡服务(octavia)以使用 Key Manager 服务(barbican)。如需更多信息，*请参阅使用密钥管理器服务管理 secret* 指南。

流程

1. 对于 SNI 列表中的每个 TLS 证书，将密钥(**server.key**)、证书(**server.crt**)和中间证书链(**ca-chain.crt**)合并到一个 PKCS12 文件(**server.p12**)中。
   这这个示例中，您创建了两个 PKCS12 文件 (**server.p12** 和 **server2.p12**)，分别作为 **www.example.com** 和 **www2.example.com** 的证书。



注意

括号中的值是此流程中的示例命令中使用的示例值。使用适合您的站点的值替换这些示例值。

示例

```
$ 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。

示例

```
$ source ~/overcloudrc
```

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

示例

```
$ 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**)上创建一个负载平衡器(**lb1**)。

示例

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

5. 创建 TERMINATED_HTTPS**侦听器(listener1),** 并使用 SNI 引用机密资源。（引用 **tls_secret1** 作为监听器的默认 TLS 容器。）

示例

```
$ openstack loadbalancer listener create  --name listener1 \
--protocol-port 443 --protocol TERMINATED_HTTPS \
--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
```

6. 创建一个池(**pool1**)，并使它成为监听器的默认池。

示例

本例中的命令会创建一个 HTTP 池，它使用专用子网，其中包含在 TCP 端口 80 上托管非安全 HTTP 应用程序的后端服务器：

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

7. 在池(**pool1**)上创建一个类型为(**HTTP**)的运行状况监视器(**healthmon1**)，以连接到后端服务器并测试路径(/)。
建议使用健康检查，但不是必需的。如果没有定义运行状况监控器，则假定成员服务器为 **ONLINE**。

示例

```
$ openstack loadbalancer healthmonitor create --name healthmon1 \
--delay 15 --max-retries 4 --timeout 10 --type HTTP --url-path / pool1
```

8. 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。

示例

在本例中，后端服务器 **192.0.2.10** 和 **192.0.2.11** 分别命名为 **member1** 和 **member2** ：

```
$ openstack loadbalancer member create --name member1 --subnet-id \
private_subnet --address 192.0.2.10 --protocol-port 443 pool1

$ openstack loadbalancer member create --name member2 --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。

示例

```
$ 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. 当健康监控器存在并正常运行时，您可以检查每个成员的状态。

示例

```
$ openstack loadbalancer member show pool1 member1
```

输出示例

工作成员(**member1**)在其 **operating_status** 具有 **ONLINE** 值：

```
+---------------------+--------------------------------------+
| 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                | member1                              |
| 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

- *使用密钥管理器服务管理* secret 指南
- *命令行界面参考*中的 LoadBalancer

9.6. 使用 HTTP/2 侦听器创建 TLS 终止负载均衡器

当您使用 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。添加 HTTP/2 侦听器后，您可以通过更快地加载页面来利用 HTTP/2 协议来提高性能。负载均衡器使用 Application-Layer Protocol Negotiation (ALPN) TLS 扩展将 HTTP/2 与客户端协商。

负载均衡服务(octavia)支持端到端 HTTP/2 流量，这意味着 HTTP2 流量不会由 HAProxy 从请求到达监听程序的点转换，直到响应从负载均衡器返回为止。要实现端到端的 HTTP/2 流量，您必须有一个带有后端重新加密的 HTTP 池：组成员侦听为 HTTPS 流量配置的安全端口和 Web 应用。

您可以将 HTTP/2 流量发送到 HTTP 池，而无需重新加密。在这种情况下，HAProxy 在到达池前转换流量，并在从负载均衡器返回前将响应转换为 HTTP/2。

红帽建议您创建一个运行状况监控器，以确保后端成员仍然可用。

注意

目前，负载均衡服务不支持对使用 HTTP/2 侦听的 TLS 终止负载均衡器进行健康监控。

先决条件

- TLS 公钥加密配置有以下特征：
  - TLS 证书、密钥和中间证书链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独立文件中。
  - 密钥和证书是 PEM 编码的。
  - 中间证书链包含多个使用 PEM 编码和串联的证书。
- 您必须配置负载均衡服务(octavia)以使用 Key Manager 服务(barbican)。如需更多信息，*请参阅使用密钥管理器服务管理 secret* 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(**server.p12**)中。

注意

括号中的值是此流程中的示例命令中使用的示例值。使用适合您的站点的值替换这些示例值。

重要

当您创建 PKCS12 文件时，请不要密码保护该文件。

示例

在这个示例中，在没有密码的情况下创建 PKCS12 文件：

```
$ 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。

示例

```
$ source ~/overcloudrc
```

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

示例

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

4. 在公共子网(**public_subnet**)上创建一个负载平衡器(**lb1**)。

   示例

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

5. 创建 **TERMINATED_HTTPS** 侦听器(**listener1)，** 并执行以下操作：

   - 引用机密资源(**tls_secret1**)，作为监听器的默认 TLS 容器。
   - 设置 ALPN 协议(**h**2)。
   - 如果客户端不支持 HTTP/2 (**http/1.1**)，则设置回退协议。

     示例

     ```
     $ openstack loadbalancer listener create --name listener1 \
     --protocol-port 443 --protocol TERMINATED_HTTPS --alpn-protocol h2 \
     --alpn-protocol http/1.1 --default-tls-container=\
     $(openstack secret list | awk '/ tls_secret1 / {print $2}') lb1
     ```

6. 创建一个池(**pool1**)，并使它成为监听器的默认池。

   示例

   本例中的命令会创建一个 HTTP 池，其中包含在 TCP 端口 80 上托管使用 Web 应用程序配置的 HTTP 应用程序的后端服务器：

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

7. 在池(**pool1**)上创建一个类型为(**TCP**)的运行状况监视器(**healthmon1**)，以连接到后端服务器。建议使用健康检查，但不是必需的。如果没有定义运行状况监控器，则假定成员服务器为 **ONLINE**。

   示例

   ```
   $ openstack loadbalancer healthmonitor create --name healthmon1 \
   --delay 15  --max-retries 4 --timeout 10 --type TCP pool1
   ```

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

   示例

   在本例中，后端服务器 192.0.2.10 和 192.0.2.11 分别命名为 member1 和 member2 ：

```
$ openstack loadbalancer member create --name member1 --subnet-id \
private_subnet --address 192.0.2.10 --protocol-port 80 pool1

$ openstack loadbalancer member create --name member2 --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。

示例

```
$ openstack loadbalancer status show lb1
```

输出示例

```
{
   "loadbalancer": {
      "id": "936dad29-4c3f-4f24-84a8-c0e6f10ed810",
      "name": "lb1",
      "operating_status": "ONLINE",
      "provisioning_status": "ACTIVE",
      "listeners": [
         {
            "id": "708b82c6-8a6b-4ec1-ae53-e619769821d4",
            "name": "listener1",
            "operating_status": "ONLINE",
            "provisioning_status": "ACTIVE",
            "pools": [
               {
                  "id": "5ad7c678-23af-4422-8edb-ac3880bd888b",
                  "name": "pool1",
                  "provisioning_status": "ACTIVE",
                  "operating_status": "ONLINE",
                  "health_monitor": {
                     "id": "4ad786ef-6661-4e31-a325-eca07b2b3dd1",
                     "name": "healthmon1",
                     "type": "TCP",
                     "provisioning_status": "ACTIVE",
                     "operating_status": "ONLINE"
                  },
                  "members": [
                     {
```

```
                    "id": "facca0d3-61a7-4b46-85e8-da6994883647",
                    "name": "member1",
                    "operating_status": "ONLINE",
                    "provisioning_status": "ACTIVE",
                    "address": "192.0.2.10",
                    "protocol_port": 80
                },
                {
                    "id": "2b0d9e0b-8e0c-48b8-aa57-90b2fde2eae2",
                    "name": "member2",
                    "operating_status": "ONLINE",
                    "provisioning_status": "ACTIVE",
                    "address": "192.0.2.11",
                    "protocol_port": 80
                }
...
```

2. **当健康监控器存在并正常运行时，您可以检查每个成员的状态。**

示例

```
$ openstack loadbalancer member show pool1 member1
```

输出示例

工作成员(member1)在其 operating_status 具有 ONLINE 值：

```
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| address             | 192.0.2.10                           |
| admin_state_up      | True                                 |
| created_at          | 2023-11-16T20:08:01                  |
| id                  | facca0d3-61a7-4b46-85e8-da6994883647 |
| name                | member1                              |
| operating_status    | ONLINE                               |
| project_id          | 9b29c91f67314bd09eda9018616851cf     |
| protocol_port       | 80                                   |
| provisioning_status | ACTIVE                               |
| subnet_id           | 3b459c95-64d2-4cfa-b348-01aacc4b3fa9 |
| updated_at          | 2023-11-16T20:08:42                  |
```

```
| weight            | 1                                    |
| monitor_port      | None                                 |
| monitor_address   | None                                 |
| backup            | False                                |
| tags              |                                      |
+-------------------+--------------------------------------+
```

其他资源

- *使用密钥管理器服务管理* secret 指南
- *命令行界面参考*中的 LoadBalancer

9.7. 在同一 IP 和后端上创建 HTTP 和 TLS 终止 HTTPS 负载均衡

当您想响应具有相同内容的 Web 客户端时，您可以在同一负载均衡器上配置非安全监听程序和 TLS 终止的 HTTPS 侦听器，无论客户端是否使用安全或不安全的 HTTP 协议连接。最好还要创建一个运行状况监视器，以确保您的后端成员仍然可用。

先决条件

- 可以从互联网访问的共享外部(public)子网。
- TLS 公钥加密配置有以下特征：
  - 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独立文件中。
  - 密钥和证书是 PEM 编码的。
  - 中间证书链包含多个使用 PEM 编码和串联的证书。
- 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，*请参阅使用密钥管理器服务管理 secret* 指南。

- 非安全 HTTP 侦听器配置与 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(server.p12)中。

注意

括号中的值是此流程中的示例命令中使用的示例值。使用适合您的站点的值替换这些示例值。

示例

```
$ 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。

示例

```
$ source ~/overcloudrc
```

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

示例

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

4. **在公共子网(public_subnet)上创建一个负载平衡器(lb1)。**

示例

```
$ openstack loadbalancer create --name lb1 \
--vip-subnet-id external_subnet --wait
```

5. **创建 TERMINATED_HTTPS侦听器(listener1)，并将机密资源引用为监听器的默认 TLS 容器。**

示例

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

6. **创建一个池(pool1)，并使它成为监听器的默认池。**

示例

本例中的命令会创建一个 HTTP 池，它使用专用子网，其中包含在 TCP 端口 80 上托管非安全 HTTP 应用程序的后端服务器：

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

7. **在池(pool1)上创建一个类型为(HTTP)的运行状况监视器(healthmon1)，以连接到后端服务器并测试路径(/)。**

   建议使用健康检查，但不是必需的。如果没有定义运行状况监控器，则假定成员服务器为 ONLINE。

   示例

```
$ openstack loadbalancer healthmonitor create --name healthmon1 \
--delay 15 --max-retries 4 --timeout 10 --type HTTP --url-path / pool1
```

8. **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。**

   示例

   在本例中，后端服务器 192.0.2.10 和 192.0.2.11 分别命名为 member1 和 member2 ：

```
$ openstack loadbalancer member create --name member1 --subnet-id \
private_subnet --address 192.0.2.10 --protocol-port 443 pool1

$ openstack loadbalancer member create --name member2 --subnet-id \
private_subnet --address 192.0.2.11 --protocol-port 443 pool1
```

9. **创建非安全、HTTP 侦听器(listener2)，并使其成为默认池，与安全监听程序相同。**

   示例

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

验证

1. **查看并验证负载均衡器(lb1)设置。**

示例

```
$ 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. **当健康监控器存在并正常运行时，您可以检查每个成员的状态。**

示例

```
$ openstack loadbalancer member show pool1 member1
```

输出示例

工作成员(member1)在其 operating_status 具有 ONLINE 值：

```
+---------------------+--------------------------------------+
| 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                | member1                              |
| 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

- *使用密钥管理器服务管理* **secret 指南**
- *命令行界面参考*中的 **LoadBalancer**

第 10 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 网络流量匹配的负载均衡器类型。

- 第 10.1 节 “创建 TCP 负载均衡器”
- 第 10.2 节 “使用健康监控器创建 UDP 负载均衡器”
- 第 10.3 节 “创建 QoS-ruled 负载均衡器”
- 第 10.4 节 “使用访问控制列表创建负载均衡器”
- 第 10.5 节 “创建 OVN 负载均衡器”

10.1. 创建 TCP 负载均衡器

当您需要管理非 HTTP、基于 TCP 的服务和应用程序的网络流量时，您可以创建负载均衡器。最好还要创建一个运行状况监视器，以确保您的后端成员仍然可用。

先决条件

- 可以从互联网访问的共享外部(public)子网。

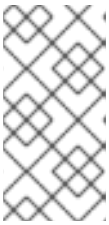
流程

1. 提供您的凭据文件。

   示例

   ```
   $ source ~/overcloudrc
   ```

2. **在公共子网(public_subnet)上创建一个负载平衡器(lb1)。**



注意

括号中的值是此流程中的示例命令中使用的示例值。使用适合您的站点的值替换这些示例值。

示例

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

3. **在指定端口 (23456) 中创建一个 TCP 监听器 (listener1)，自定义应用程序被配置。application is configured.**

示例

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

4. **创建一个池(pool1)，并使它成为监听器的默认池。**

示例

在本例中，会创建一个池，它使用一个私有子网，其中包含在特定 TCP 端口上托管自定义应用程序的后端服务器：

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

5. **在池(pool1)上创建一个运行状况监视器(healthmon1)，以连接到后端服务器并探测 TCP 服务端口。**

示例

建议使用健康检查，但不是必需的。如果没有定义运行状况监控器，则假定成员服务器为 ONLINE。

```
$ openstack loadbalancer healthmonitor create --name healthmon1 \
--delay 15 --max-retries 4 --timeout 10 --type TCP pool1
```

6. **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。**

示例

在本例中，后端服务器 192.0.2.10 和 192.0.2.11 分别命名为 member1 和 member2 ：

```
$ openstack loadbalancer member create --name member1 --subnet-id \
private_subnet --address 192.0.2.10 --protocol-port 443 pool1

$ openstack loadbalancer member create --name member2 --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。**

示例

```
$ 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. **当健康监控器存在并正常运行时，您可以检查每个成员的状态。使用以下命令获取成员 ID：**

示例

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

工作成员(member1)具有其 operating_status 的 ONLINE 值。

示例

```
$ openstack loadbalancer member show pool1 member1
```

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                | member1                              |
| 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

10.2. 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。最好还要创建一个运行状况监视器，以确保您的后端成员仍然可用。

先决条件

- 可以从互联网访问的共享外部(public)子网。
- 没有阻止 ICMP Destination Unreachable 消息(ICMP 类型 3)的安全规则。

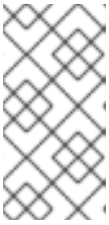
流程

1. **提供您的凭据文件。**

示例

```
$ source ~/overcloudrc
```

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



注意

括号中的值是此流程中的示例命令中使用的示例值。使用适合您的站点的值替换这些示例值。

示例

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

3. **在端口 (1234) 上创建一个监听器（listener1）。**

示例

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

4. 创建侦听器默认池(pool1)。

示例

本例中的命令会创建一个使用专用子网的池，其中包含托管一个或多个配置为使用 UDP 端口的后端服务器：

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

5. 在池(pool1)上创建一个运行状况监视器(healthmon1)，使用 UDP (UDP-CONNECT)连接到后端服务器。

建议使用健康检查，但不是必需的。如果没有定义运行状况监控器，则假定成员服务器为 ONLINE。

示例

```
$ openstack loadbalancer healthmonitor create --name healthmon1 \
--delay 5 --max-retries 2 --timeout 3 --type UDP-CONNECT pool1
```

6. 将专用子网上的后端服务器(192.0.2.10 和 192.0.2.11)添加到默认池。

示例

在本例中，后端服务器 192.0.2.10 和 192.0.2.11 分别命名为 member1 和 member2 ：

```
$ openstack loadbalancer member create --name member1 --subnet-id \
private_subnet --address 192.0.2.10 --protocol-port 1234 pool1

$ openstack loadbalancer member create --name member2 --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。

示例

```
$ 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. 当健康监控器存在并正常运行时，您可以检查每个成员的状态。

示例

```
$ openstack loadbalancer member show pool1 member1
```

工作成员(member1)具有其 operating_status 的 ONLINE 值。

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                | member1                              |
| 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

10.3. 创建 QOS-RULED 负载均衡器

您可以将 Red Hat OpenStack Platform (RHOSP)网络服务(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传入或传出流量。最好还要创建一个运行状况监视器，以确保您的后端成员仍然可用。

先决条件

- 可以从互联网访问的共享外部(public)子网。
- 包含为 RHOSP 网络服务创建的带宽限制规则的 QoS 策略。

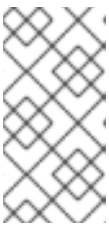
流程

1. 提供您的凭据文件。

   示例

   ```
   $ source ~/overcloudrc
   ```

2. 创建一个网络带宽 QoS 策略(qos_policy_bandwidth)，最大 1024 kbps，最大突发率 1024 kb。

   注意

   括号中的值是此流程中的示例命令中使用的示例值。使用适合您的站点的值替换这些示例值。

   示例

   ```
   $ openstack network qos policy create qos_policy_bandwidth

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

3. 使用 QoS 策略(qos-policy-bandwidth)在公共子网上创建负载均衡器(lb1)。

示例

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

4. **在端口 (80) 上创建一个监听器 (listener1)。**

示例

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

5. **创建侦听器默认池(pool1)。**

示例

本例中的命令会创建一个 HTTP 池，它使用包含在 TCP 端口 80 上托管 HTTP 应用程序的专用子网：

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

6. **在连接到后端服务器的池上创建运行状况监视器(healthmon1)，并测试路径(/)。**

建议使用健康检查，但不是必需的。如果没有定义运行状况监控器，则假定成员服务器为 ONLINE。

示例

```
$ openstack loadbalancer healthmonitor create --name healthmon1 \
--delay 15 --max-retries 4 --timeout 10 --type HTTP --url-path / \
pool1
```

7. **在专用子网(private_subnet)上添加负载均衡器成员（192.0.2.10 和 192.0.2.11）到默认的池。**

示例

在本例中，后端服务器 192.0.2.10 和 192.0.2.11 分别命名为 member1 和 member2 ：

```
$ openstack loadbalancer member create --name member1 --subnet-id \
private_subnet --address 192.0.2.10 --protocol-port 443 pool1

$ openstack loadbalancer member create --name member2 --subnet-id \
private_subnet --address 192.0.2.11 --protocol-port 443 pool1
```

验证

- **查看并验证监听器(listener1)设置。**

示例

```
$ 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，parameter vip_qos_policy_id，包含策略 ID。

其他资源

- *命令行界面参考*中的 LoadBalancer

10.4. 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。最好还要创建一个运行状况监视器，以确保您的后端成员仍然可用。

先决条件

- 可以从互联网访问的共享外部(public)子网。

流程

1. 提供您的凭据文件。

   示例

```
$ source ~/overcloudrc
```

2. **在公共子网(public_subnet)上创建一个负载平衡器(lb1)。**

注意

括号中的值是此流程中的示例命令中使用的示例值。使用适合您的站点的值替换这些示例值。

示例

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

3. **使用允许的 CIDR (192.0.2.0/24 和 198.51.100.0/24)创建监听器(listener1)。**

示例

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

4. **创建侦听器默认池(pool1)。**

示例

在本例中，会创建一个池，它使用专用子网，其中包含在 TCP 端口 80 上配置了自定义应用程序的后端服务器：

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

5. 在连接后端服务器的池上创建运行状况监控器，并测试路径(/)。

建议使用健康检查，但不是必需的。如果没有定义运行状况监控器，则假定成员服务器为 **ONLINE**。

示例

```
$ openstack loadbalancer healthmonitor create --name healthmon1 \
--delay 15 --max-retries 4 --timeout 10 --type HTTP --url-path / pool1
```

6. 在专用子网(private_subnet)上添加负载均衡器成员（192.0.2.10 和 192.0.2.11）到默认的池。

示例

在本例中，后端服务器 192.0.2.10 和 192.0.2.11 分别命名为 member1 和 member2 ：

```
$ 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(listener1)设置。

示例

```
$ 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，参数 allowed_cidrs 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，请确保从 CIDR 不在 allowed_cidrs 列表中的客户端向监听程序的请求；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

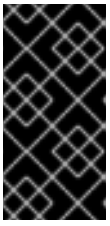
- *命令行界面参考*中的 LoadBalancer

10.5. 创建 OVN 负载均衡器

您可以使用 Red Hat OpenStack Platform (RHOSP)客户端创建一个管理 RHOSP 部署中网络流量的负载均衡器。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。

先决条件

- 必须部署 ML2/OVN 供应商驱动程序。



重要

OVN 提供程序仅支持第 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。

- 可以从互联网访问的共享外部(public)子网。

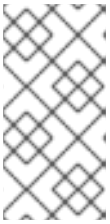
流程

1. 提供您的凭据文件。

示例

```
$ source ~/overcloudrc
```

2. 使用 **--provider ovn** 参数，在专用子网(private_subnet)上创建一个负载均衡器(lb1)。



注意

括号中的值是此流程中的示例命令中使用的示例值。使用适合您的站点的值替换这些示例值。

示例

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

3. 创建一个监听器(listener1)，它将使用配置自定义应用程序的指定端口（80）上的协议(tcp)。

注意

OVN 提供程序仅支持第 4 层 TCP 和 UDP 网络流量。

示例

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

4. 创建侦听器默认池(pool1)。

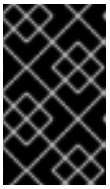
注意

OVN 唯一支持的负载平衡算法是 SOURCE_IP_PORT。

示例

本例中的命令会创建一个 HTTP 池，它使用专用子网，其中包含在特定 TCP 端口上托管自定义应用程序的后端服务器：

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



重要

OVN 不支持用于负载平衡的运行状况监控器功能。

5. **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。**

示例

在本例中，后端服务器 192.0.2.10 和 192.0.2.11 分别命名为 member1 和 member2 ：

```
$ openstack loadbalancer member create --name member1 --subnet-id \
private_subnet --address 192.0.2.10 --protocol-port 80 pool1

$ openstack loadbalancer member create --name member2 --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。**

示例

```
$ 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 listener show** 命令来查看监听程序详细信息。

示例

```
$ 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，以查看池(**pool1**)和 **load-balancer** 成员。

示例

```
$ 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

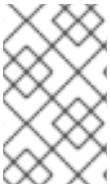
- *命令行界面参考*中的 **LoadBalancer**

第 11 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)来通过利用多个条件将 HTTP 请求重定向到特定应用服务器池，以满足您的业务需求。

11.1. 关于第 7 层负载均衡

第 7 层(L7)负载均衡从 Open Systems Interconnection (OSI)模型中提取其名称，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服务器池。以下是所有 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*、*交换* 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

您不能使用 UDP 负载均衡器创建 L7 策略和规则。

L7 负载均衡器包含一个侦听器，它代表多个后端池接受请求，并根据使用应用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专门调整和优化，以满足特定类型的内容。例如，您可以将一组后端服务器（池）配置为仅服务镜像；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。

与低级负载平衡不同，L7 负载均衡不要求负载平衡服务后面的所有池都具有相同的内容。L7 负载均衡器可以根据应用程序消息中的 URI、主机、HTTP 标头和其他数据直接请求。

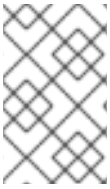
11.2. 负载均衡服务中的第 7 层负载均衡

虽然您可以为任何定义的 L7 应用程序接口实施第 7 层(L7)负载均衡），但 Red Hat OpenStack Platform 负载均衡服务(octavia)的 L7 功能只指向 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 规则是一个单一简单的逻辑测试，评估为 true 或 false。L7 策略是 L7 规则的集合，如果与策略关联的所有规则都匹配，则需要采取的已定义操作。

11.3. 第 7 层负载均衡规则

对于 Red Hat OpenStack Platform 负载均衡服务(octavia)，层 7 (L7)负载均衡规则是一个单一、简单的逻辑测试，返回 true 或 false。它由规则类型、比较类型、值和可选键组成，具体取决于规则类型。L7 规则必须始终与 L7 策略关联。



注意

您不能使用 UDP 负载均衡器创建 L7 策略和规则。

其他资源

- 第 11.4 节 “第 7 层负载均衡规则类型”

11.4. 第 7 层负载均衡规则类型

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

- HOST_NAME ：规则将请求中的 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 path 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，并将它与规则中的 value 参数进行比较。
- SSL_CONN_HAS_CERT ：如果客户端提供用于 TLS 客户端身份验证的证书，则规则匹配。这并不意味着证书有效。
- SSL_VERIFY_RESULT ：该规则与 TLS 客户端验证证书验证结果匹配。值为零(0)表示证书已被成功验证。大于零的值表示证书失败的验证。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 Distinguished Name 字段，并将它与规则中的 value 参数进行比较。

其他资源

- 第 11.3 节 “第 7 层负载均衡规则”

11.5. 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的比较：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: String starting with
- ENDS_WITH: String 结束
- CONTAINS: 字符串包含
- EQUAL_TO: 字符串等于

其他资源

- 第 11.4 节 “第 7 层负载均衡规则类型”

11.6. 第 7 层负载均衡规则会导致VERSION

要更全面地表达某些策略需要以及 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有其结果。如果给定规则的 invert 参数为 true，则其比较的结果会被反转。

例如，一个 反转等于 规则实际上 不再等于 规则。只有在给定的正则表达式不匹配时，inverted regex 规则才会返回 true。

其他资源

- 第 11.7 节 “第 7 层负载均衡策略”

11.7. 第 7 层负载均衡策略

对于 Red Hat OpenStack Platform 负载均衡服务(octavia)，层 7 (L7)负载均衡策略是与监听器关联的 L7 规则集合，它们可能还具有与后端池的关联。策略(policy)是策略中的所有规则为 true 执行的操作。

注意

您不能使用 UDP 负载均衡器创建 L7 策略和规则。

其他资源

- 第 11.3 节 “第 7 层负载均衡规则”

11.8. 第 7 层负载平衡策略逻辑

Red Hat OpenStack Platform 负载均衡服务(octavia)，第 7 层负载平衡策略使用以下逻辑：与给定策略关联的所有规则都逻辑上是 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，请使用相同操作或创建多个策略，生成更详细的正则表达式。

其他资源

- 第 11.3 节 “第 7 层负载均衡规则”

11.9. 第 7 层负载平衡策略操作

如果第 7 层负载平衡策略与给定请求匹配，则执行该策略操作。以下是 L7 策略可能会执行的操作：

- REJECT ：请求通过适当的响应代码被拒绝，且不会转发到任何后端池。
- REDIRECT_TO_URL ：请求将 HTTP 重定向发送到 redirect_url 参数中定义的 URL。
- REDIRECT_PREFIX: 匹配此策略的请求会被重定向到这个前缀 URL。
- REDIRECT_TO_POOL: 请求转发到与 L7 策略关联的后端池。

其他资源

- 第 11.7 节 “第 7 层负载均衡策略”

11.10. 第 7 层负载平衡策略位置

对于 Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)，当多个第 7 层(L7)负载均衡策略与监听器关联时，策略位置参数的值非常重要。position 参数用于决定评估 L7 策略的顺序。策略位置通过以下方法影响监听程序行为：

- 在负载均衡服务(haproxy amphorae)的引用实现中，HAProxy 强制执行以下有关策略操作的顺序：
  - 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 策略仅在以上所有之后评估，并根据策略指定位置的顺序进行评估。

- L7 策略以特定顺序评估，如 location 属性所定义，第一个与给定请求匹配的策略是其操作的操作。

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

- 策略位置编号以一(1)开头。

- 如果创建了具有与现有策略匹配的位置创建新策略，则新策略将插入到给定位置。

- 如果在没有指定位置的情况下创建新策略，或者指定大于列表中已存在的策略数量的位置，则新策略会附加到列表中。

- 当策略插入、删除或附加到列表中时，策略位置值会在不跳过数字的情况下从一个(1)重新排序。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，策略 C 的位置将变为 2。

其他资源

- 第 11.7 节 “第 7 层负载均衡策略”

11.11. 将非安全 HTTP 请求重定向到安全 HTTP

您可以使用带有第 7 层(L7)的 Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)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，任何到达不安全 TCP 端口 80 的 HTTP 请求都会被重定向到安全 TCP 端口 443。

先决条件

- 一个 TLS 终止的 HTTPS 负载均衡器 (lb1)，它有一个监听器 (listener1) 和一个池 (pool1)。

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。

示例

```
$ source ~/overcloudrc
```

2. 在负载平衡器(lb1)端口(80)上创建一个 HTTP 侦听器(http_listener)。

注意

括号中的值是此流程中的示例命令中使用的示例值。使用适合您的站点的值替换这些示例值。

示例

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

3. 在监听器(http_listener)上创建一个 L7 策略(policy1)。该策略必须包含操作(REDIRECT_TO_URL)并指向 URL (https://www.example.com/)。

示例

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

4. **添加与策略的所有请求匹配的 L7 规则(policy1)。**

示例

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

验证

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

2. 运行 **openstack loadbalancer l7rule list <l7policy& gt; 命令，并验证是否存在带有 STARTS_WITH 的 compare_type 规则。**

示例

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

其他资源

- ***命令行界面参考中的* loadbalancer listener create**
- ***命令行界面参考中的*LoadBalancer l7policy create**

- *命令行界面参考中的*LoadBalancer l7rule create

11.12. 根据到池的开始路径重定向请求

您可以使用 Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)将 HTTP 请求重定向到备用服务器池。您可以定义第 7 层(L7)策略，以匹配请求的 URL 中的一个或多个起始路径。

在本例中，任何包含以 /js 或 /images 开头的 URL 的请求都会被重定向到静态内容服务器的替代池。

先决条件

- HTTP load balancer (lb1) 有一个监听器 (listener1) 和一个池 (pool1)。如需更多信息，请参阅使用 健康监控器创建 HTTP 负载均衡器。

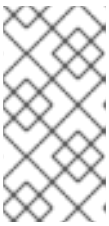
流程

1. 提供您的凭据文件。

   示例

   ```
   $ source ~/overcloudrc
   ```

2. 在负载均衡器(lb1)上创建第二个池(static_pool)。

   注意

   括号中的值是此流程中的示例命令中使用的示例值。使用适合您的站点的值替换这些示例值。

   示例

   -

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

3. **在专用子网(private_subnet)上添加负载均衡器成员（192.0.2.10 和 192.0.2.11）到池（static_pool）。**

示例

在本例中，后端服务器 192.0.2.10 和 192.0.2.11 分别命名为 member1 和 member2 ：

```
$ openstack loadbalancer member create --name member1 --subnet-id \
private_subnet --address 192.0.2.10 --protocol-port 80 static_pool

$ openstack loadbalancer member create --name member2 --subnet-id \
private_subnet --address 192.0.2.11 --protocol-port 80 static_pool
```

4. **在监听器 (listener1) 上创建一个 L7 策略 (policy1) 。策略必须包含操作(REDIRECT_TO_POOL)并指向池(static_pool)。**

示例

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

5. **添加 L7 规则，该规则在策略的请求路径的开头查找 /js。**

示例

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

6. **创建一个 L7 策略(policy2)，其操作(REDIRECT_TO_POOL)并添加指向池的监听程序(listener1)。**

示例

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

7. **添加 L7 规则，该规则在策略的请求路径的开头查找 /images。**

示例

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

验证

1. 运行 **openstack loadbalancer l7policy list 命令，并验证策略 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都有带有 STARTS_WITH 的 compare_type 规则。**

示例

```
$ openstack loadbalancer l7rule list policy1

$ openstack loadbalancer l7rule list policy2
```

其他资源

- *命令行界面参考中的*LoadBalancer pool create
- *命令行界面参考中的*LoadBalancer member create
- *命令行界面参考中的*LoadBalancer l7policy create
- *命令行界面参考中的*LoadBalancer l7rule create

11.13. 将子域请求发送到特定池

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

在这个示例中，包含 HTTP/1.1 主机名 www2.example.com 的任何请求都会被重定向到备用池应用服务器 pool2。

先决条件

- HTTP load balancer (lb1) 有一个监听器 (listener1) 和一个池 (pool1)。

  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。

   示例

```
$ source ~/overcloudrc
```

2. **在负载均衡器 (lb1) 上创建第二个池（pool2）。**

注意

括号中的值是此流程中的示例命令中使用的示例值。使用适合您的站点的值替换这些示例值。

示例

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

3. **在监听器 (listener1) 上创建一个 L7 策略 (policy1) 。该策略必须包含操作 (REDIRECT_TO_POOL)并指向池(pool2)。**

示例

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

4. **在策略中添加 L7 规则，该规则使用 HTTP/1.1 主机名 www2.example.com 发送任何请求到第二个池(pool2)。**

示例

```
$ 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，并验证策略 **policy1** 是否存在。

2. 运行 **openstack loadbalancer l7rule list <l7policy>** 命令，并验证策略存在一个带有 **compare_type** 的 **EQUAL_TO** 规则。

   示例

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

其他资源

- *命令行界面参考中的***LoadBalancer pool create**
- *命令行界面参考中的***LoadBalancer l7policy create**
- *命令行界面参考中的***LoadBalancer l7rule create**

11.14. 根据主机名向特定池发送请求

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

在本例中，任何包含以 .example.com 结尾的 HTTP/1.1 主机名的请求都会被重定向到备用池应用服务

器 pool2。

先决条件

- HTTP load balancer (lb1) 有一个监听器 (listener1) 和一个池 (pool1)。

  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。

   示例

   ```
   $ source ~/overcloudrc
   ```

2. 在负载均衡器 (lb1) 上创建第二个池（pool2）。

   注意

   括号中的值是此流程中的示例命令中使用的示例值。使用适合您的站点的值替换这些示例值。

   示例

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

3. 在监听器 (listener1) 上创建一个 L7 策略 (policy1) 。该策略必须包含操作 (REDIRECT_TO_POOL)并指向池(pool2)。

示例

```
$ openstack loadbalancer l7policy create --name policy1 \
--action REDIRECT_TO_POOL --redirect-pool pool2 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)。**

示例

```
$ 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，并验证策略 policy1 是否存在。**

2. 运行 **openstack loadbalancer l7rule list <l7policy> 命令，并验证策略存在一个带有 compare_type 的 EQUAL_TO 规则。**

示例

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

其他资源

- *命令行界面参考中的*LoadBalancer pool create

- *命令行界面参考中的*LoadBalancer l7policy create

- *命令行界面参考中的*LoadBalancer l7rule create

11.15. 根据没有浏览器 COOKIE 向特定池发送请求

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。

在本例中，任何缺少浏览器 cookie（auth_token） 的 Web 客户端请求被重定向到包含身份验证服务器的替代池。

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，且不会解决安全问题。

先决条件

- 一个 TLS 终止的 HTTPS 负载均衡器 (lb1)，它有一个监听器 (listener1) 和一个池 (pool1)。

  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 网络服务(neutron)子网，其安全身份验证服务器在其上验证 Web 用户。

流程

1. 提供您的凭据文件。

   示例

```
$ source ~/overcloudrc
```

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

注意

括号中的值是此流程中的示例命令中使用的示例值。使用适合您的站点的值替换这些示例值。

示例

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

3. **向第二个池中添加一个成员，在身份验证子网（secure_subnet)）中的身份验证服务器(192.0.2.10)。**

示例

在本例中，后端服务器 192.0.2.10 被命名为 member1 ：

```
$ openstack loadbalancer member create --name member1 \
--address 192.0.2.10 --protocol-port 80 --subnet-id secure_subnet \
login_pool
```

4. **在监听器 (listener1) 上创建一个 L7 策略 (policy1) 。该策略必须包含操作(REDIRECT_TO_POOL)，并指向第二个池(login_pool)。**

示例

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

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

示例

```
$ 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，并验证策略 policy1 是否存在。**

2. 运行 **openstack loadbalancer l7rule list <l7policy& gt; 命令，并验证是否存在带有 REGEX 的 compare_type 规则。**

示例

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

其他资源

- ***命令行界面参考中的*LoadBalancer pool create**
- ***命令行界面参考中的*LoadBalancer member create**

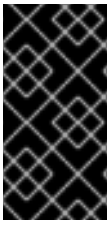
- *命令行界面参考中的*LoadBalancer l7policy create

- *命令行界面参考中的*LoadBalancer l7rule create

11.16. 根据没有浏览器 COOKIE 或无效 COOKIE 值向特定池发送请求

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，或者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 auth_token 的值为 INVALID 的 Web 客户端请求都会被重定向到包含身份验证服务器的替代池。



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，且不解决安全问题。

先决条件

- 一个 TLS 终止的 HTTPS 负载均衡器 (lb1)，它有一个监听器 (listener1) 和一个池 (pool1)。

  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 网络服务(neutron)子网，其安全身份验证服务器在其上验证 Web 用户。

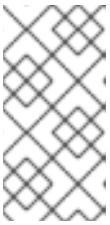
流程

1. 提供您的凭据文件。

   示例

```
$ source ~/overcloudrc
```

2. 在负载均衡器(lb1)上创建第二个池(login_pool)。



注意

括号中的值是此流程中的示例命令中使用的示例值。使用适合您的站点的值替换这些示例值。

示例

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

3. 向第二个池中添加一个成员，在身份验证子网（secure_subnet)）中的身份验证服务器(192.0.2.10)。

示例

在本例中，后端服务器 192.0.2.10 被命名为 member1 ：

```
$ openstack loadbalancer member create --name member1 \
--address 192.0.2.10 --protocol-port 80 --subnet-id secure_subnet \
login_pool
```

4. 在监听器 (listener1) 上创建一个 L7 策略 (policy1) 。该策略必须包含操作(REDIRECT_TO_POOL)，并指向第二个池(login_pool)。

示例

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

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

示例

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

6. **在监听器 (listener1) 上创建第二个 L7 策略（policy2）。该策略必须包含操作 (REDIRECT_TO_POOL)，并指向第二个池(login_pool)。**

示例

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

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

示例

```
$ 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，并验证策略 **policy1** 和 **policy2** 是否存在。

2. 运行 **openstack loadbalancer l7rule list <l7policy>** 命令，并验证 **policy1** 是否存在带有 **REGEX** 的 **compare_type** 规则，以及 **policy2** 的带有 **compare_type** 为 **EQUAL_TO** 的规则。

示例

```
$ openstack loadbalancer l7rule list policy1

$ openstack loadbalancer l7rule list policy2
```

其他资源

- *命令行界面参考中的***LoadBalancer pool create**
- *命令行界面参考中的***LoadBalancer member create**
- *命令行界面参考中的***LoadBalancer l7policy create**
- *命令行界面参考中的***LoadBalancer l7rule create**

11.17. 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应用程序服务器的替代池。业务逻辑标准通过第 **7** 层**(L7)**策略执行，该策略尝试与预定义的主机名和请求路径匹配。

在本例中，任何匹配主机名 **api.example.com** 的 **Web** 客户端请求，并在请求路径的开头具有 **/api** 重定向到备用池 **api_pool**。

先决条件

- HTTP load balancer (lb1) 有一个监听器 (listener1) 和一个池 (pool1)。如需更多信息，请参阅使用 健康监控器创建 HTTP 负载均衡器。

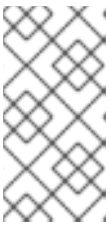
流程

1. 提供您的凭据文件。

   示例

   ```
   $ source ~/overcloudrc
   ```

2. 在负载均衡器(lb1)上创建第二个池(api_pool)。

   注意

   括号中的值是此流程中的示例命令中使用的示例值。使用适合您的站点的值替换这些示例值。

   示例

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

3. 在专用子网(private_subnet)上添加负载均衡器成员（192.0.2.10 和 192.0.2.11）到池（static_pool）。

   示例

   在本例中，后端服务器 192.0.2.10 和 192.0.2.11 分别命名为 member1 和 member2 ：

```
$ openstack loadbalancer member create --name member1 --subnet-id \
private_subnet --address 192.0.2.10 --protocol-port 80 static_pool

$ openstack loadbalancer member create --name member2 --subnet-id \
private_subnet --address 192.0.2.11 --protocol-port 80 static_pool
```

4. **在监听器 (listener1) 上创建一个 L7 策略 (policy1) 。策略必须包含操作 (REDIRECT_TO_POOL)并指向池(api_pool)。**

   示例

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

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

   示例

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

6. **向策略添加第二个 L7 规则，该策略与请求路径开头的 /api 匹配。**

   此规则通过第一个规则逻辑 AND。

   示例

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

验证

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

2. 运行 **openstack loadbalancer l7rule list <l7policy>** 命令，再验证两者都存在带有 **EQUAL_TO** 和 **STARTS_WITH** 的 **compare_type** 规则的 **policy1**。

   示例

```
$ openstack loadbalancer l7rule list policy1

$ openstack loadbalancer l7rule list policy2
```

其他资源

- *命令行界面参考中的*LoadBalancer pool create
- *命令行界面参考中的*LoadBalancer member create
- *命令行界面参考中的*LoadBalancer l7policy create
- *命令行界面参考中的*LoadBalancer l7rule create

11.18. 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 层(L7)的 Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)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，将池中成员服务器(pool1)中的 Cookie **site_version** 设置为 **B**。

先决条件

- 两个生产网站（站点 A 和网站 B）。
- 您已按照"基于到池的起始路径重定向请求"的说明配置了 HTTP 负载均衡器。所需的配置概述如下：
  - 在负载均衡器 (lb1) 中的监听器 (listener1)。
  - 带有以 /js 或 /images 开头的 URL 的 HTTP 请求发送到池(static_pool)。
  - 所有其他请求都发送到侦听器默认池(pool1)。
  - 有关配置的详情，请参考 第 11.12 节 “根据到池的开始路径重定向请求”。

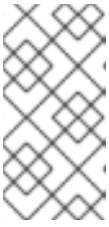
流程

1. 提供您的凭据文件。

   示例

   ```
   $ source ~/overcloudrc
   ```

2. 在负载均衡器(lb1)上创建第三个池(pool_B)。

   注意

   括号中的值是此流程中的示例命令中使用的示例值。使用适合您的站点的值替换这些示例值。

示例

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

3. **在专用子网(private_subnet)上添加负载均衡器成员（192.0.2.50 和 192.0.2.51）到池（pool_B）。**

示例

在本例中，后端服务器 192.0.2.50 和 192.0.2.51 分别命名为 member1 和 member2 ：

```
$ openstack loadbalancer member create --name member1 \
--address 192.0.2.50 --protocol-port 80 \
--subnet-id private_subnet pool_B

$ openstack loadbalancer member create --name member2 \
--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)。**

示例

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

5. **在专用子网(private_subnet)上添加负载均衡器成员（192.0.2.100 和 192.0.2.101）到池（static_pool_B）。**

示例

```
$ openstack loadbalancer member create --name member3 \
--address 192.0.2.100 --protocol-port 80 \
--subnet-id private_subnet static_pool_B

$ openstack loadbalancer member create --name member4 \
--address 192.0.2.101 --protocol-port 80 \
--subnet-id private_subnet static_pool_B
```

6. **在监听器 (listener1) 中创建一个 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 中插入策略。**

示例

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

7. **向策略（policy2）添加一个 L7 规则，它使用一个正则表达式匹配以 /js 或 /images 开始的请求路径。**

示例

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

8. **向策略添加第二个 L7 规则，该策略(policy2)与 Cookie (site_version)匹配到确切的字符串 (B)。**

示例

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

9. **在监听器 (listener1) 上创建一个 L7 策略(policy3)。该策略必须包含操作 (REDIRECT_TO_POOL)，并指向池(pool_B)。将策略插入到位置 2。**

示例

```
$ 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(policy3)到确切的字符串(B)。**

示例

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

注意

将带有最具体规则的 L7 策略分配给较低位置非常重要，因为第一个规则都评估为 True 的策略是其操作的策略。在此过程中，需要在 policy3 之前评估 policy2，以避免将请求发送到不正确的池。

验证

1. 运行 **openstack loadbalancer l7policy list** 命令，并验证策略 **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都有带有 STARTS_WITH 的 compare_type 规则。**

示例

```
$ openstack loadbalancer l7rule list policy2

$ openstack loadbalancer l7rule list policy3
```

其他资源

- ***命令行界面参考中的*LoadBalancer pool create**
- ***命令行界面参考中的*LoadBalancer member create**
- ***命令行界面参考中的*LoadBalancer l7policy create**
- ***命令行界面参考中的*LoadBalancer l7rule create**

第 12 章 使用标签对负载均衡服务对象进行分组

标签是您可以添加到 Red Hat OpenStack Platform 负载均衡服务(octavia)对象中的任意字符串，用于将它们分类到组中。标签不会影响负载平衡对象的功能：负载均衡器、监听程序、池、成员、运行状况监视器、规则和策略。您可以在创建对象时添加标签，或者在对象创建后添加或删除标签。

通过将特定标签与负载平衡对象关联，您可以运行 list 命令来过滤属于一个或多个组的对象。可以将对象过滤到一个或多个组中，可以是管理负载平衡服务资源的使用量、分配和维护的起点。自动配置管理工具也可以利用标记对象的功能。

本节中包含的主题有：

- 第 12.1 节 “在创建负载均衡服务对象时添加标签”
- 第 12.2 节 “在预先存在的负载均衡服务对象上添加或删除标签”
- 第 12.3 节 “使用标签过滤负载均衡服务对象”

12.1. 在创建负载均衡服务对象时添加标签

在创建 Red Hat OpenStack Load-balancing 服务(octavia)对象时，您可以添加您选择的标签。当标签存在时，您可以使用对应的 loadbalancer list 命令过滤负载均衡器、监听程序、池、成员、运行状况监视器、规则和策略。

流程

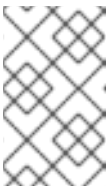
1. 提供您的凭据文件。

示例

```
$ source ~/overcloudrc
```

2. 当为对象使用正确的带有 **--tag <tag>** 选项的 **create** 命令创建负载均衡器时，为它添加一个标签（tag）：

- **openstack loadbalancer create --tag <tag> …**
- **OpenStack loadbalancer listener create --tag <tag> …**
- **openstack loadbalancer pool create --tag <tag> …**
- **OpenStack loadbalancer member create --tag <tag> …**
- **OpenStack loadbalancer healthmonitor create --tag <tag> …**
- **OpenStack loadbalancer l7policy create --tag <tag> …**
- **OpenStack loadbalancer l7rule create --tag <tag> …**



注意

标签可以是任意有效的 **unicode** 字符串，最大长度为 **255** 个字符。

示例 - 创建并标记负载均衡器

在这个示例中，负载均衡器 **lb1** 被创建并带有两个标签 **Finance** 和 **Sales**：

```
$ openstack loadbalancer create --name lb1 \
--vip-subnet-id public_subnet --tag Finance --tag Sales
```

注意

负载平衡服务对象可以具有一个或多个标签。对您要添加的每个额外标签重复 **--tag <tag>** 选项。

示例 - 创建和标记监听程序

在这个示例中，监听程序 **listener1** 被创建并带有一个标签 **Sales**：

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

示例 - 创建和标记池

在本例中，池 **pool1** 使用标签 **Sales** 创建：

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

示例 - 在池中创建成员并标记它

在本例中，成员 **192.0.2.10** 在 **pool1** 中创建，标签为 **Sales** ：

```
$ openstack loadbalancer member create --name member1 \
--subnet-id private_subnet --address 192.0.2.10 --protocol-port 80 \
--tag Sales pool1
```

示例 - 创建并标记运行状况监控器

在这个示例中，健康监控器 **healthmon1** 被创建，带有一个标签 **Sales**：

```
$ openstack loadbalancer healthmonitor create --name healthmon1 \
--delay 15 --max-retries 4 --timeout 10 --type HTTP --url-path / \
--tag Sales pool1
```

示例 - 创建并标记 L7 策略

在这个示例中，使用标签 **Sales** 创建 **L7** 策略 **policy1** ：

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

示例 - 创建和标记 L7 规则

在这个示例中，使用标签 **Sales** 创建 **L7** 规则 **rule1** ：

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

验证

- **确认已创建了对象，并包含您使用对象的相应 show 命令添加的标签。**

示例

在本例中，show 命令在 loadbalancer, lb1 上运行：

```
$ openstack loadbalancer show lb1
```

输出示例

```
+---------------------+--------------------------------------+
| admin_state_up      | True                                 |
| availability_zone   | None                                 |
| created_at          | 2022-08-26T19:34:15                  |
| description         |                                      |
| flavor_id           | None                                 |
| id                  | 7975374b-3367-4436-ab19-2d79d8c1f29b |
| listeners           |                                      |
| name                | lb1                                  |
| operating_status    | ONLINE                               |
| pools               |                                      |
| project_id          | 2eee3b86ca404cdd977281dac385fd4e     |
| provider            | amphora                              |
| provisioning_status | ACTIVE                               |
| updated_at          | 2022-08-30T13:30:17                  |
| vip_address         | 172.24.3.76                          |
| vip_network_id      | 4c241fc4-95eb-491a-affe-26c53a8805cd |
| vip_port_id         | 9978a598-cc34-47f7-ba28-49431d570fd1 |
| vip_qos_policy_id   | None                                 |
| vip_subnet_id       | e999d323-bd0f-4469-974f-7f66d427e507 |
| tags                | Finance                              |
|                     | Sales                                |
+---------------------+--------------------------------------+
```

- ***命令行界面参考*中的 loadbalancer create**

- *命令行界面参考*中的 loadbalancer show

12.2. 在预先存在的负载均衡服务对象上添加或删除标签

您可以在创建后在 Red Hat OpenStack 负载均衡服务(octavia)对象上添加和删除选择的标签。当标签存在时，您可以使用对应的 loadbalancer list 命令过滤负载均衡器、监听程序、池、成员、健康监控器、规则和策略。

流程

1. 提供您的凭据文件。

   示例

   ```
   $ source ~/overcloudrc
   ```

2. 执行以下操作之一：

   - 将 --tag <tag > 选项与对象的相应 set 命令一起使用，为已存在的负载均衡对象添加标签：
     - openstack loadbalancer set --tag <tag> <load_balancer_name_or_ID>
     - openstack loadbalancer listener set --tag <tag> <listener_name_or_ID>
     - openstack loadbalancer pool set --tag <tag> <pool_name_or_ID>
     - openstack loadbalancer member set --tag <tag> <pool_name_or_ID> <member_name_or_ID>
     -

```
openstack loadbalancer healthmonitor set --tag <tag>
<healthmon_name_or_ID>
```

- ```
  openstack loadbalancer l7policy set --tag <tag> <l7policy_name_or_ID>
  ```

- ```
  openstack loadbalancer l7rule set --tag <tag> <l7policy_name_or_ID>
  <l7rule_ID>
  ```

注意

标签可以是任意有效的 unicode 字符串，最大长度为 255 个字符。

示例

在这个示例中，标签 **Finance** 和 **Sales** 被添加的奥负载均衡器 **lb1**：

```
$ openstack loadbalancer set --tag Finance --tag Sales lb1
```

注意

负载平衡服务对象可以具有一个或多个标签。对您要添加的每个额外标签重复 **--tag <tag>** 选项。

- 使用带有相应 未设置 对象的 **--tag <tag>** 选项，从预先存在的负载均衡对象中删除标签：

  - ```
    openstack loadbalancer unset --tag <tag> <load_balancer_name_or_ID>
    ```

  - ```
    openstack loadbalancer listener unset --tag <tag> <listener_name_or_ID>
    ```

  - ```
    openstack loadbalancer pool unset --tag <tag> <pool_name_or_ID>
    ```

  - ```
    openstack loadbalancer member unset --tag <tag> <pool_name_or_ID>
    <member_name_or_ID>
    ```

- openstack loadbalancer healthmonitor unset --tag <tag> <healthmon_name_or_ID>
- openstack loadbalancer l7policy unset --tag <tag> <policy_name_or_ID>
- openstack loadbalancer l7rule unset --tag <tag> <policy_name_or_ID> <l7rule_ID>

示例

在本例中，标签 Sales 已从负载均衡器 lb1 中删除：

```
$ openstack loadbalancer unset --tag Sales lb1
```

- 在对象的相应 set 命令中使用 --no-tag 选项，从预先存在的负载均衡对象中删除所有标签：
  - openstack loadbalancer set --no-tag <load_balancer_name_or_ID>
  - openstack loadbalancer listener set --no-tag <listener_name_or_ID>
  - openstack loadbalancer pool set --no-tag <pool_name_or_ID>
  - openstack loadbalancer member set --no-tag <pool_name_or_ID> <member_name_or_ID>
  - openstack loadbalancer healthmonitor set --no-tag <healthmon_name_or_ID>
  - openstack loadbalancer l7policy set --no-tag <l7policy_name_or_ID>
  - openstack loadbalancer l7rule set --no-tag <l7policy_name_or_ID>

<l7rule_ID>

示例

在本例中，所有标签都从负载均衡器 lb1 中删除：

```
$ openstack loadbalancer set --no-tag lb1
```

验证

- **使用适当的 show 命令，确认您在负载平衡对象上添加或删除了一个或多个标签。**

示例

在本例中，show 命令在 loadbalancer, lb1 上运行：

```
$ openstack loadbalancer show lb1
```

输出示例

```
+---------------------+--------------------------------------+
| admin_state_up      | True                                 |
| availability_zone   | None                                 |
| created_at          | 2022-08-26T19:34:15                  |
| description         |                                      |
| flavor_id           | None                                 |
| id                  | 7975374b-3367-4436-ab19-2d79d8c1f29b |
| listeners           |                                      |
| name                | lb1                                  |
| operating_status    | ONLINE                               |
| pools               |                                      |
| project_id          | 2eee3b86ca404cdd977281dac385fd4e     |
| provider            | amphora                              |
| provisioning_status | ACTIVE                               |
| updated_at          | 2022-08-30T13:30:17                  |
| vip_address         | 172.24.3.76                          |
| vip_network_id      | 4c241fc4-95eb-491a-affe-26c53a8805cd |
| vip_port_id         | 9978a598-cc34-47f7-ba28-49431d570fd1 |
| vip_qos_policy_id   | None                                 |
| vip_subnet_id       | e999d323-bd0f-4469-974f-7f66d427e507 |
| tags                | Finance                              |
|                     | Sales                                |
+---------------------+--------------------------------------+
```

- *命令行界面参考*中的 loadbalancer set
- *命令行界面参考*中的 loadbalancer show

12.3. 使用标签过滤负载均衡服务对象

您可以使用 Red Hat OpenStack 负载均衡服务(octavia)创建对象列表。对于标记的对象，您可以创建过滤的列表：根据对象是否包含一个或多个指定标签，包含或排除对象的列表。能够过滤负载均衡器、监听程序、池、成员、健康监控器、规则和策略，可以是管理负载平衡服务资源的使用量、分配和维护的起点。

流程

1. 提供您的凭据文件。

   示例

   ```
   $ source ~/overcloudrc
   ```

2. 使用其中一个标签选项为对象运行适当的 loadbalancer list 命令来过滤您要列出的对象：

   表 12.1. 过滤对象的标签选项

在我的列表中，我希望...	例子

在我的列表中，我希望...	例子
包括匹配所有指定标签的对象。	**$ OpenStack loadbalancer list --tags Sales,Finance** **$ OpenStack loadbalancer listener list --tags Sales,Finance** **$ OpenStack loadbalancer l7pool list --tags Sales,Finance** **$ OpenStack loadbalancer member list --tags Sales,Finance pool1** **$ OpenStack loadbalancer healthmonitor list --tags Sales,Finance** **$ OpenStack loadbalancer l7policy list --tags Sales,Finance** **$ OpenStack loadbalancer l7rule list --tags Sales,Finance policy1**
包括与一个或多个指定标签匹配的对象。	**$ openstack loadbalancer list --any-tags Sales,Finance** **$ OpenStack loadbalancer listener list --any-tags Sales,Finance** **$ OpenStack loadbalancer l7pool list --any-tags Sales,Finance** **$ OpenStack loadbalancer member list --any-tags Sales,Finance pool1** **$ OpenStack loadbalancer healthmonitor list --any-tags Sales,Finance** **$ OpenStack loadbalancer l7policy list --any-tags Sales,Finance** **$ OpenStack loadbalancer l7rule list --any-tags Sales,Finance policy1**

在我的列表中，我希望...	例子
排除与所有指定标签匹配的对象。	**$ OpenStack loadbalancer list --not-tags Sales,Finance** **$ OpenStack loadbalancer listener list --not-tags Sales,Finance** **$ OpenStack loadbalancer l7pool list --not-tags Sales,Finance** **$ OpenStack loadbalancer member list --not-tags Sales,Finance pool1** **$ OpenStack loadbalancer healthmonitor list --not-tags Sales,Finance** **$ OpenStack loadbalancer l7policy list --not-tags Sales,Finance** **$ OpenStack loadbalancer l7rule list --not-tags Sales,Finance policy1**
排除与一个或多个指定标签匹配的对象。	**$ openstack loadbalancer list --not-any-tags Sales,Finance** **$ OpenStack loadbalancer listener list --not-any-tags Sales,Finance** **$ OpenStack loadbalancer l7pool list --not-any-tags Sales,Finance** **$ OpenStack loadbalancer member list --not-any-tags Sales,Finance pool1** **$ OpenStack loadbalancer healthmonitor list --not-any-tags Sales,Finance** **$ OpenStack loadbalancer l7policy list --not-any-tags Sales,Finance** **$ OpenStack loadbalancer l7rule list --not-any-tags Sales,Finance policy1**

注意

当指定多个标签时，请使用逗号分隔标签。

其他资源

- *命令行界面参考*中的 loadbalancer list

第 13 章 在边缘创建用于负载均衡网络流量的可用区

您可以使用 Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)在可用区中创建负载均衡器来提高流量吞吐量并缩短延迟。

本节中包含的主题有：

- 创建负载均衡服务可用区
- 在可用区中创建负载均衡器

13.1. 为负载均衡服务创建可用区

使用 Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)，RHOSP 管理员可以创建可用区(AZ)，使项目用户在分布式计算节点(DCN)环境中创建负载均衡器以增加流量吞吐量并降低延迟。

创建负载平衡服务 AZ 需要两个步骤：RHOSP 管理员必须首先创建一个 AZ 配置集，然后使用配置集创建用户可见的负载平衡服务 AZ。

AZ 配置集必须具有以下内容：

- Compute 服务(nova) AZ 的名称。
- 要使用的管理网络。

  每个 AZ 都有多个管理网络，一个唯一的网络。中央 AZ 使用现有的负载平衡管理网络 lb-mgmt-net，另外 AZ 使用其相应的网络，lb-mgmt-<AZ_name>-net，例如 lb-mgmt-az-dcn1-net、lb-mgmt-az-dcn2-net 等等。

先决条件

- 您必须有一个 DCN 环境，它已通过运行 octavia-dcn-deployment.yaml Ansible playbook 创建所需的网络资源。

如需更多信息，请参阅部署 *分布式 Compute 节点(DCN)* 架构指南中的 为负载均衡服务可用区创建网络资源。

- 您的负载均衡服务供应商驱动程序必须是 amphora。OVN 供应商驱动程序不支持 AZ。
- 您必须是一个具有 admin 角色的 RHOSP 用户。

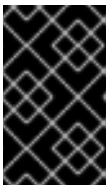
流程

1. 提供您的凭据文件。

示例

```
$ source ~/centralrc
```

2. 收集要用于命名负载平衡服务 AZ 的计算服务 AZ 的名称。



重要

您创建的负载均衡服务 AZ 的名称必须与计算服务 AZ 的名称匹配。

```
$ openstack availability zone list --compute
```

输出示例

```
+------------+-------------+
| Zone Name  | Zone Status |
+------------+-------------+
| az-central | available   |
| az-dcn1    | available   |
| az-dcn2    | available   |
| internal   | available   |
+------------+-------------+
```

3. **收集用于创建负载平衡服务 AZ 的管理网络的 ID：**

```
$ openstack network list -c Name -c ID
```

输出示例

```
+--------------------------------------+-------------------------+
| ID                                   | Name                    |
+--------------------------------------+-------------------------+
| 10458d6b-e7c9-436f-92d9-711677c9d9fd | lb-mgmt-az-dcn2-net     |
| 662a94f5-51eb-4a4c-86c4-52dcbf471ef9 | lb-mgmt-net             |
| 6b97ef58-2a25-4ea5-931f-b7c07cd09474 | lb-mgmt-backbone-net    |
| 99f4215b-fad8-432d-8444-1f894154dc30 | heat_tempestconf_network |
| a2884aaf-846c-4936-9982-3083f6a71d9b | lb-mgmt-az-dcn1-net     |
| d7f7de6c-0e84-49e2-9042-697fa85d2532 | public                  |
| e887a9f9-15f7-4854-a797-033cedbfe5f3 | public2                 |
+--------------------------------------+-------------------------+
```

4. **创建 AZ 配置集。重复此步骤，为您要创建的每个负载均衡服务 AZ 创建 AZ 配置集：**

```
$ openstack loadbalancer availabilityzoneprofile create \
--name <AZ_profile_name> --provider amphora --availability-zone-data '{"compute_zone": "
<compute_AZ_name>","management_network": "<lb_mgmt_AZ_net_UUID>"}'
```

示例 - 为 az-central创建配置集

在本例中，创建了一个 AZ 配置集(az_profile_central)，它使用管理网络(lb-mgmt-net)在 Compute AZ (az-central)中运行的 Compute 节点上：

```
$ openstack loadbalancer availabilityzoneprofile create \
--name az_profile_central --provider amphora --availability-zone-data \
'{"compute_zone": "az-central","management_network": \
"662a94f5-51eb-4a4c-86c4-52dcbf471ef9"}'
```

5. **重复步骤 4，为您要创建的每个负载均衡服务 AZ 创建 AZ 配置集。**

示例 - 为 az-dcn1创建配置文件

示例 - 为 az-dcn1创建配置文件

在本例中，会创建一个 AZ 配置集(az-profile-dcn1)，它使用管理网络(lb-mgmt-az-dcn1-net)在计算 AZ (az-dcn1)上运行的 Compute 节点上：

```
$ openstack loadbalancer availabilityzoneprofile create \
--name az-profile-dcn1 --provider amphora --availability-zone-data \
'{"compute-zone": "az-dcn1","management-network": \
"a2884aaf-846c-4936-9982-3083f6a71d9b"}'
```

示例 - 为 az-dcn2创建配置文件

在本例中，会创建一个 AZ 配置集(az-profile-dcn2)，它使用管理网络(lb-mgmt-az-dcn2-net)在计算 AZ (az-dcn2)中运行的 Compute 节点上：

```
$ openstack loadbalancer availabilityzoneprofile create \
--name az-profile-dcn2 --provider amphora --availability-zone-data \
'{"compute-zone": "az-dcn2","management-network": \
"10458d6b-e7c9-436f-92d9-711677c9d9fd"}'
```

6. **使用 AZ 配置集，创建一个负载均衡服务 AZ。为每个 AZ 使用适当的配置集，为任何其他 AZ 重复此步骤。**

示例 - 创建 AZ: az-central

在本例中，使用 AZ 配置集(az-profile-central)创建负载平衡服务 AZ (az-central)：

```
$ openstack loadbalancer availabilityzone create --name az-central \
--availabilityzoneprofile az-profile-central \
--description "AZ for Headquarters" --enable
```

示例 - 创建 AZ: az-dcn1

在本例中，使用 AZ 配置集(az-profile-az-dcn1)创建负载平衡服务 AZ (az-dcn1)：

```
$ openstack loadbalancer availabilityzone create --name az-dcn1 \
--availabilityzoneprofile az-profile-az-dcn1 \
--description "AZ for South Region" --enable
```

示例 - 创建 AZ: az-dcn2

在本例中，负载均衡服务 AZ (az-dcn2)使用 AZ 配置集(az-profile-az-dcn2)创建：

```
$ openstack loadbalancer availabilityzone create --name az-dcn2 \
--availabilityzoneprofile az-profile-az-dcn2 \
--description "AZ for North Region" --enable
```

验证

- **确认 AZ (az-central)已创建。对任何其他 AZ 重复此步骤，为每个 AZ 使用适当的名称。**

示例 - 验证 az-central

```
$ openstack loadbalancer availabilityzone show az-central
```

输出示例

```
+-------------------------------+--------------------------------------+
| Field                         | Value                                |
+-------------------------------+--------------------------------------+
| name                          | az-central                           |
| availability_zone_profile_id | 5ed25d22-52a5-48ad-85ec-255910791623 |
| enabled                       | True                                 |
| description                   | AZ for Headquarters                  |
+-------------------------------+--------------------------------------+
```

示例 - 验证 az-dcn1

```
$ openstack loadbalancer availabilityzone show az-dcn1
```

输出示例

```
+---------------------------+-------------------------------------+
| Field                     | Value                               |
+---------------------------+-------------------------------------+
| name                      | az-dcn1                             |
| availability_zone_profile_id | e0995a82-8e67-4cea-b32c-256cd61f9cf3 |
| enabled                   | True                                |
| description               | AZ for South Region                 |
+---------------------------+-------------------------------------+
```

示例 - 验证 az-dcn2

```
$ openstack loadbalancer availabilityzone show az-dcn2
```

输出示例

```
+---------------------------+-------------------------------------+
| Field                     | Value                               |
+---------------------------+-------------------------------------+
| name                      | az-dcn2                             |
| availability_zone_profile_id | 306a4725-7dac-4046-8f16-f2e668ee5a8d |
| enabled                   | True                                |
| description               | AZ for North Region                 |
+---------------------------+-------------------------------------+
```

后续步骤

- 在可用区中创建负载均衡器

其他资源

- *部署分布式 Compute 节点(DCN)架构指南中的*为 负载平衡服务可用区创建网络资源。

- *命令行界面参考中的*LoadBalancer availabilityzoneprofile create

- *命令行界面参考中的*LoadBalancer availabilityzone create

13.2. 在可用区中创建负载均衡器

使用 Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)，您可以在分布式计算节点(DCN)环境中在可用区(AZ)中创建负载均衡器，以增加流量吞吐量并降低延迟。

先决条件

- 您必须具有 RHOSP 管理员提供的负载均衡服务 AZ。

- 与负载均衡器关联的虚拟 IP (VIP)网络必须在负载均衡器所属的 AZ 中提供。

流程

1. 提供您的凭据文件。

   示例

   ```
   $ source ~/centralrc
   ```

2. 要为 DCN 环境创建负载均衡器，请使用 loadbalancer create 命令和 --availability-zone 选项并指定适当的 AZ。

   示例

   例如，要在可用区(az-central)上在公共子网(public_subnet)上创建非终止的 HTTPS 负载均衡器(lb1)，您需要输入以下命令：

```
$ openstack loadbalancer create --name lb1 --vip-subnet-id \
public_subnet --availability-zone az-central
```

3. 通过添加监听程序、池、运行状况监视器和负载均衡器成员继续创建负载均衡器。

验证

- 确认负载平衡器(lb1)是可用性区域的成员(az-central)。

示例

```
$ openstack loadbalancer show lb1
```

输出示例

```
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| admin_state_up      | True                                 |
| availability_zone   | az-central                           |
| created_at          | 2023-07-12T16:35:05                  |
| description         |                                      |
| flavor_id           | None                                 |
| id                  | 85c7e567-a0a7-4fcb-af89-a0bbc9abe3aa |
| listeners           |                                      |
| name                | lb1                                  |
| operating_status    | ONLINE                               |
| pools               |                                      |
| project_id          | d303d3bda9b34d73926dc46f4d0cb4bc     |
| provider            | amphora                              |
| provisioning_status | ACTIVE                               |
| updated_at          | 2023-07-12T16:36:45                  |
| vip_address         | 10.101.10.229                        |
| vip_network_id      | d7f7de6c-0e84-49e2-9042-697fa85d2532 |
| vip_port_id         | 7f916764-d171-4317-9c86-a1750a54b16e |
| vip_qos_policy_id   | None                                 |
| vip_subnet_id       | a421cbcf-c5db-4323-b7ab-1df20ee6acab |
| tags                |                                      |
+---------------------+--------------------------------------+
```

其他资源

- 为负载均衡服务创建可用区
- *命令行界面参考*中的 LoadBalancer

第 14 章 更新并升级负载均衡服务

执行常规更新和升级，以便您可以获取最新的 Red Hat OpenStack Platform 负载均衡服务(octavia)功能和程序错误修复。

- 第 14.1 节 “更新并升级负载均衡服务”
- 第 14.2 节 “更新正在运行的负载均衡服务实例”

14.1. 更新并升级负载均衡服务

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

先决条件

- 调度维护窗口来执行升级，因为升级负载均衡服务 control plane 无法正常工作。

流程

1. 执行 RHOSP 更新，如 *执行 Red Hat OpenStack Platform 的次要更新* 指南中所述。
2. 应用维护版本后，如果您需要使用新功能，请轮转运行 amphorae 将其更新至最新的 amphora 镜像。

其他资源

- *执行 Red Hat OpenStack Platform 的次要更新* 指南
- 第 14.2 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”

14.2. 更新正在运行的负载均衡服务实例

您可以使用较新的镜像定期更新正在运行的负载均衡服务实例(amphora)。例如，您可能想要在以下事件期间更新 amphora 实例：

- Red Hat OpenStack Platform (RHOSP)的更新或升级。
- 对您的系统进行安全更新。
- 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 Image 服务(glance)，然后配置负载均衡服务(octavia)以使用新镜像。当您故障转移负载均衡器时，您可以强制负载平衡服务启动使用新 amphora 镜像的实例(amphora)。

先决条件

- 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。

   示例

   ```
   $ source ~/overcloudrc
   ```

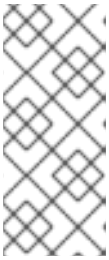
2. 列出您要更新的所有负载均衡器的 ID：

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

3. 对每个负载均衡器失败：

   -

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



注意

当您启动对负载均衡器的出现故障时，请根据需要监控系统利用率，请调整执行故障转移的速度。负载平衡器故障转移创建新的虚拟机和端口，这可能会临时增加 OpenStack 网络上的负载。

4. 监控负载均衡器失败的状态：

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

当负载均衡器状态为 ACTIVE 时，更新已完成。

其他资源

- *命令行界面参考*中的 LoadBalancer

第 15 章 故障排除和维护负载均衡服务

负载均衡服务(octavia)的基本故障排除和维护从熟悉 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。如果您需要对更多深度进行故障排除，您可以 SSH 到一个或多个负载平衡服务实例(amphorae)。

15.1. 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)及其各种组件进行故障排除。

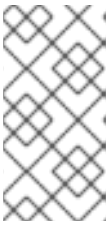
流程

1. 提供您的凭据文件。

   示例

   ```
   $ source ~/overcloudrc
   ```

2. 验证负载平衡器(lb1)设置。



注意

括号中的值是此流程中的示例命令中使用的示例值。使用适合您的站点的值替换这些示例值。

示例

```
$ 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.

使用上一步中的 **loadbalancer ID(265d0b71-c073-40f4-9718-8a182c6d53ca)**，获取与负载均衡器 **(lb1)** 关联的 **amphora** 的 **ID**。

示例

```
$ 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-ba09-49e1-8c39-41770aa25070)**，查看 **amphora** 信息。

示例

```
$ 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(listener1)详情。

示例

```
$ 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。**

示例

```
$ 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。

注意

为负载均衡器 VIP 实施的安全组只允许所需的协议和端口的数据流量。因此，您无法 ping 负载均衡器 VIP，因为 ICMP 流量被阻止。

示例

```
$ 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

15.2. 负载均衡服务实例管理日志

负载均衡服务实例(amphora)的管理日志卸载功能涵盖了 amphora 内的所有系统日志，但租户流日志除外。您可以将租户流日志发送到发送管理日志的同一 syslog 接收器。您可以将租户流日志发送到处理管理日志的同一 syslog 接收器，但必须单独配置租户流日志。

amphora 使用应用程序发送消息的原生日志格式来发送所有管理日志消息。amphorae 记录到与其他 RHOSP 日志相同的位置的 Red Hat OpenStack Platform (RHOSP) Controller 节点 (/var/log/containers/octavia-amphorae/)。

其他资源

- *第 5 章 管理负载均衡服务实例日志*

15.3. 迁移特定的负载均衡服务实例

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

流程

1. **提供您的凭据文件。**

示例

```
$ source ~/overcloudrc
```

2. **找到您要迁移的 amphora 的 ID。您需要在以后的步骤中提供 ID。**

```
$ openstack loadbalancer amphora list
```

3. **要防止计算调度程序服务将任何新的 amphorae 调度到要撤离的 Compute 节点，请禁用 Compute 节点(compute-host-1)。**

注意

括号中的值是此流程中的示例命令中使用的示例值。使用适合您的站点的值替换这些示例值。

示例

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

4. **使用您获取的 amphora ID (ea17210a-1076-48ff-8a1f-ced49ccb5e53)，在 amphora 上失败。**

示例

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

其他资源

- *命令行界面参考* 中设置的计算服务
- *命令行界面参考*中的 LoadBalancer

15.4. 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)登录运行负载平衡服务实例(amphorae)会很有帮助。

先决条件

- 您必须具有负载均衡服务(octavia) SSH 私钥。

流程

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

```
$ eval $(ssh-agent -s)
$ sudo -E ssh-add /etc/octavia/ssh/octavia_id_rsa
```

2. 提供您的凭据文件。

示例

```
$ source ~/overcloudrc
```

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

```
$ openstack loadbalancer amphora list
```

4. 使用 SSH 连接到 amphora：

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

5. 完成后，关闭到 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

15.5. 显示监听器统计

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

- 当前活跃的连接(active_connections)。
- 已接收的字节总数(在 中)。
- 发送的总字节数(bytes_out)。
- 无法实现的请求总数(request_errors)。
- 处理的连接总数(total_connections)。

流程

1. **提供您的凭据文件。**

示例

```
$ source ~/overcloudrc
```

2. 查看监听器(listener1)的统计信息。

注意

括号中的值是此流程中的示例命令中使用的示例值。使用适合您的站点的值替换这些示例值。

示例

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

提示

如果您不知道监听程序的名称，请输入命令 **loadbalancer listener list**。

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

- *命令行界面参考*中的 loadbalancer listener stats show 部分。
- 第 15.6 节 “解释监听程序请求错误”

15.6. 解释监听程序请求错误

您可以获取有关特定 Red Hat OpenStack Platform (RHOSP) loadbalancer 的监听程序的统计信息。更多信息请参阅 第 15.5 节 “显示监听器统计”。

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 变量不会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这个错误。loadBalancers 不会检查数据流量的内容。在本例中，loadbalancer 会将这个流解析为成功，因为它在用户和 Web 服务器之间进行正确传输信息。

以下条件可能会导致 request_errors 变量递增：

- 从客户端早期终止，在发送请求前。
- 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

- *命令行界面参考*中的 loadbalancer listener stats show 部分。
- 第 15.5 节 “显示监听器统计”