OVN Octavia Provider Documentation

Release 2.2.1.dev5

OpenStack Foundation

Feb 09, 2024

CONTENTS

CHAPTER

ONE

OVN AS A PROVIDER DRIVER FOR OCTAVIA

Octavia has integrated support for provider drivers where any third party Load Balancer driver can be integrated with Octavia. Functionality related to this has been developed in OVN and now OVN can now be supported as a provider driver for Octavia.

The OVN Provider driver has a few advantages when used as a provider driver for Octavia over Amphora, like:

- OVN can be deployed without VMs, so there is no additional overhead as is required currently in Octavia when using the default Amphora driver.
- OVN Load Balancers can be deployed faster than default Load Balancers in Octavia (which use Amphora currently) because of no additional deployment requirement.
- Since OVN supports virtual networking for both VMs and containers, OVN as a Load Balancer driver can be used succesfully with Kuryr Kubernetes[1].

1.1 Limitations of the OVN Provider Driver

OVN has its own set of limitations when considered as an Load Balancer driver. These include:

- OVN currently supports TCP, UDP and SCTP, so Layer-7 based load balancing is not possible with OVN.
- Currently, the OVN Provider Driver supports a 1:1 protocol mapping between Listeners and associated Pools, i.e. a Listener which can handle TCP protocols can only be used with pools associated to the TCP protocol. Pools handling UDP protocols cannot be linked with TCP based Listeners. This limitation will be handled in an upcoming core OVN release.
- IPv6 support is not tested by Tempest.
- Mixed IPv4 and IPv6 members are not supported.
- Only the SOURCE_IP_PORT load balancing algorithm is supported, others like ROUND_ROBIN and LEAST_CONNECTIONS are not currently supported.
- Octavia flavors are not supported.

1.2 Creating an OVN based Load Balancer

The OVN provider driver can be tested out on DevStack using the configuration options in:

```
#
# Sample DevStack local.conf.
#
# This sample file is intended to be used for your typical DevStack␣
↪environment
# that's running all of OpenStack on a single host.
#
# It will enable the use of OVN as Octavia's Provider driver.
#

[[local|localrc]]

DATABASE_PASSWORD=password
RABBIT_PASSWORD=password
SERVICE_PASSWORD=password
SERVICE_TOKEN=password
ADMIN_PASSWORD=password
SERVICE_PASSWORD=$ADMIN_PASSWORD

# Logging
# -------

# By default ``stack.sh`` output only goes to the terminal where it runs.  It␣
↪can
# be configured to additionally log to a file by setting ``LOGFILE`` to the␣
↪full
# path of the destination log file.  A timestamp will be appended to the␣
↪given name.
LOGFILE=$DEST/logs/stack.sh.log

# Old log files are automatically removed after 7 days to keep things neat. ␣
↪Change
# the number of days by setting ``LOGDAYS``.
LOGDAYS=2

# Nova logs will be colorized if ``SYSLOG`` is not set; turn this off by␣
↪setting
# ``LOG_COLOR`` false.
#LOG_COLOR=False

# Enable OVN
Q_AGENT=ovn
Q_ML2_PLUGIN_MECHANISM_DRIVERS=ovn,logger
Q_ML2_PLUGIN_TYPE_DRIVERS=local,flat,vlan,geneve
Q_ML2_TENANT_NETWORK_TYPE="geneve"
```

(continues on next page)

(continued from previous page)

```
# Enable OVN services
enable_service ovn-northd
enable_service ovn-controller
enable_service q-ovn-metadata-agent

# Use Neutron
enable_service q-svc

# Disable Neutron agents not used with OVN.
disable_service q-agt
disable_service q-l3
disable_service q-dhcp
disable_service q-meta

# Enable services, these services depend on neutron plugin.
enable_plugin neutron https://opendev.org/openstack/neutron
enable_service q-trunk
enable_service q-dns
#enable_service q-qos

# Enable octavia tempest plugin tests
enable_plugin octavia-tempest-plugin https://opendev.org/openstack/octavia-
↪tempest-plugin
disable_service horizon

# Cinder (OpenStack Block Storage) is disabled by default to speed up
# DevStack a bit. You may enable it here if you would like to use it.
disable_service cinder c-sch c-api c-vol

# A UUID to uniquely identify this system.  If one is not specified, a random
# one will be generated and saved in the file 'ovn-uuid' for re-use in future
# DevStack runs.
#OVN_UUID=

# If using the OVN native layer-3 service, choose a router scheduler to
# manage the distribution of router gateways on hypervisors/chassis.
# Default value is leastloaded.
#OVN_L3_SCHEDULER=leastloaded

# The DevStack plugin defaults to using the ovn branch from the official ovs
# repo.  You can optionally use a different one.  For example, you may want to
# use the latest patches in blp's ovn branch (and see OVN_BUILD_FROM_SOURCE):
#OVN_REPO=https://github.com/blp/ovs-reviews.git
#OVN_BRANCH=ovn

# NOTE: When specifying the branch, as shown above, you must also enable this!
# By default, OVN will be installed from packages. In order to build OVN from
# source, set OVN_BUILD_FROM_SOURCE=True
#OVN_BUILD_FROM_SOURCE=False
```

(continues on next page)

(continued from previous page)

```
# If the admin wants to enable this chassis to host gateway routers for
# external connectivity, then set ENABLE_CHASSIS_AS_GW to True.
# Then devstack will set ovn-cms-options with enable-chassis-as-gw
# in Open_vSwitch table's external_ids column.
# If this option is not set on any chassis, all the of them with bridge
# mappings configured will be eligible to host a gateway.
ENABLE_CHASSIS_AS_GW=True

# If you wish to use the provider network for public access to the cloud,
# set the following
#Q_USE_PROVIDERNET_FOR_PUBLIC=True

# Create public bridge
OVN_L3_CREATE_PUBLIC_NETWORK=True

# This needs to be equalized with Neutron devstack
PUBLIC_NETWORK_GATEWAY="172.24.4.1"

# Octavia configuration
OCTAVIA_NODE="api"
DISABLE_AMP_IMAGE_BUILD=True
enable_plugin barbican https://opendev.org/openstack/barbican
enable_plugin octavia https://opendev.org/openstack/octavia
enable_plugin octavia-dashboard https://opendev.org/openstack/octavia-
↪dashboard
LIBS_FROM_GIT+=python-octaviaclient
enable_service octavia
enable_service o-api
enable_service o-hk
enable_service o-da
disable_service o-cw
disable_service o-hm

# OVN octavia provider plugin
enable_plugin ovn-octavia-provider https://opendev.org/openstack/ovn-octavia-
↪provider

[[post-config|$NOVA_CONF]]
[scheduler]
discover_hosts_in_cells_interval = 2
```

Kindly note that the configuration allows the user to create Load Balancers of both Amphora and OVN types.

Once the DevStack run is complete, the user can create a load balancer in Openstack:

```
$ openstack loadbalancer create --vip-network-id public --provider ovn
+---------------------+--------------------------------------+
| Field               | Value                                |
```

(continues on next page)

(continued from previous page)

```
+---------------------+--------------------------------------+
| admin_state_up      | True                                 |
| created_at          | 2018-12-13T09:08:14                  |
| description         |                                      |
| flavor              |                                      |
| id                  | 94e7c431-912b-496c-a247-d52875d44ac7 |
| listeners           |                                      |
| name                |                                      |
| operating_status    | OFFLINE                              |
| pools               |                                      |
| project_id          | af820b57868c4864957d523fb32ccfba     |
| provider            | ovn                                  |
| provisioning_status | PENDING_CREATE                       |
| updated_at          | None                                 |
| vip_address         | 172.24.4.9                           |
| vip_network_id      | ee97665d-69d0-4995-a275-27855359956a |
| vip_port_id         | c98e52d0-5965-4b22-8a17-a374f4399193 |
| vip_qos_policy_id   | None                                 |
| vip_subnet_id       | 3eed0c05-6527-400e-bb80-df6e59d248f1 |
+---------------------+--------------------------------------+
```

The user can see the different types of loadbalancers with their associated providers as below:

```
+--------------------------------------+------+----------------------------------
↪---+-------------+----------------------+----------+
| id                                   | name | project_id                       ␣
↪   | vip_address | provisioning_status | provider |
+--------------------------------------+------+----------------------------------
↪---+-------------+----------------------+----------+
| c5f2070c-d51d-46f0-bec6-dd05e7c19370 |      |␣
↪af820b57868c4864957d523fb32ccfba | 172.24.4.10 | ACTIVE              |␣
↪amphora  |
| 94e7c431-912b-496c-a247-d52875d44ac7 |      |␣
↪af820b57868c4864957d523fb32ccfba | 172.24.4.9  | ACTIVE              | ovn ␣
↪    |
+--------------------------------------+------+----------------------------------
↪---+-------------+----------------------+----------+
```

Now we can see that OVN will show the load balancer in its *loadbalancer* table:

```
$ ovn-nbctl list load_balancer
_uuid               : c72de15e-5c2e-4c1b-a21b-8e9a6721193c
external_ids        : {enabled=True,
                       lr_ref="neutron-3d2a873b-b5b4-4d14-ac24-47a835fd47b2",
                       ls_refs="{\"neutron-ee97665d-69d0-4995-a275-
↪27855359956a\": 1}",
                       "neutron:vip"="172.24.4.9",
                       "neutron:vip_port_id"="c98e52d0-5965-4b22-8a17-
↪a374f4399193"}
name                : "94e7c431-912b-496c-a247-d52875d44ac7"
```

(continues on next page)

(continued from previous page)

```
protocol            : tcp
vips                : {}
```

Next, a Listener can be created for the associated Load Balancer:

```
$ openstack loadbalancer listener create --protocol TCP --protocol-port /
  64015 94e7c431-912b-496c-a247-d52875d44ac7
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| connection_limit          | -1                                   |
| created_at                | 2018-12-13T09:14:51                  |
| default_pool_id           | None                                 |
| default_tls_container_ref | None                                 |
| description               |                                      |
| id                        | 21e77cde-854f-4c3e-bd8c-9536ae0443bc |
| insert_headers            | None                                 |
| l7policies                |                                      |
| loadbalancers             | 94e7c431-912b-496c-a247-d52875d44ac7 |
| name                      |                                      |
| operating_status          | OFFLINE                              |
| project_id                | af820b57868c4864957d523fb32ccfba     |
| protocol                  | TCP                                  |
| protocol_port             | 64015                                |
| provisioning_status       | PENDING_CREATE                       |
| sni_container_refs        | []                                   |
| timeout_client_data       | 50000                                |
| timeout_member_connect    | 5000                                 |
| timeout_member_data       | 50000                                |
| timeout_tcp_inspect       | 0                                    |
| updated_at                | None                                 |
+---------------------------+--------------------------------------+
```

OVN updates the Listener information in the Load Balancer table:

```
$ ovn-nbctl list load_balancer
_uuid               : c72de15e-5c2e-4c1b-a21b-8e9a6721193c
external_ids        : {enabled=True, "listener_21e77cde-854f-4c3e-bd8c-
↪9536ae0443bc"="64015:", lr_ref="neutron-3d2a873b-b5b4-4d14-ac24-47a835fd47b2
↪", ls_refs="{\"neutron-ee97665d-69d0-4995-a275-27855359956a\": 1}",
↪"neutron:vip"="172.24.4.9", "neutron:vip_port_id"="c98e52d0-5965-4b22-8a17-
↪a374f4399193"}
name                : "94e7c431-912b-496c-a247-d52875d44ac7"
protocol            : tcp
vips                : {}
```

Next, a Pool is associated with the Listener:

```
$ openstack loadbalancer pool create --protocol TCP --lb-algorithm /
```

(continues on next page)

(continued from previous page)

```
SOURCE_IP_PORT --listener 21e77cde-854f-4c3e-bd8c-9536ae0443bc
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| admin_state_up      | True                                 |
| created_at          | 2018-12-13T09:21:37                  |
| description         |                                      |
| healthmonitor_id    |                                      |
| id                  | 898be8a2-5185-4f3b-8658-a56457f595a9 |
| lb_algorithm        | SOURCE_IP_PORT                       |
| listeners           | 21e77cde-854f-4c3e-bd8c-9536ae0443bc |
| loadbalancers       | 94e7c431-912b-496c-a247-d52875d44ac7 |
| members             |                                      |
| name                |                                      |
| operating_status    | OFFLINE                              |
| project_id          | af820b57868c4864957d523fb32ccfba     |
| protocol            | TCP                                  |
| provisioning_status | PENDING_CREATE                       |
| session_persistence | None                                 |
| updated_at          | None                                 |
+---------------------+--------------------------------------+
```

OVNs Load Balancer table is modified as below:

```
$ ovn-nbctl list load_balancer
_uuid               : c72de15e-5c2e-4c1b-a21b-8e9a6721193c
external_ids        : {enabled=True, "listener_21e77cde-854f-4c3e-bd8c-
↪9536ae0443bc"="64015:", lr_ref="neutron-3d2a873b-b5b4-4d14-ac24-47a835fd47b2
↪", ls_refs="{\"neutron-ee97665d-69d0-4995-a275-27855359956a\": 1}",
↪"neutron:vip"="172.24.4.9", "neutron:vip_port_id"="c98e52d0-5965-4b22-8a17-
↪a374f4399193", "pool_898be8a2-5185-4f3b-8658-a56457f595a9"=""}
name                : "94e7c431-912b-496c-a247-d52875d44ac7"
protocol            : tcp
vips                : {}
```

Lastly, when a member is created, OVNs Load Balancer table is complete:

```
$ openstack loadbalancer member create --address 10.10.10.10 /
--protocol-port 63015 898be8a2-5185-4f3b-8658-a56457f595a9
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| address             | 10.10.10.10                          |
| admin_state_up      | True                                 |
| created_at          | 2018-12-13T09:26:05                  |
| id                  | adf55e70-3d50-4e62-99fd-dd77eababb1c |
| name                |                                      |
| operating_status    | NO_MONITOR                           |
| project_id          | af820b57868c4864957d523fb32ccfba     |
| protocol_port       | 63015                                |
```

(continues on next page)

(continued from previous page)

```
| provisioning_status | PENDING_CREATE                       |
| subnet_id           | None                                 |
| updated_at          | None                                 |
| weight              | 1                                    |
| monitor_port        | None                                 |
| monitor_address     | None                                 |
| backup              | False                                |
+---------------------+--------------------------------------+
$ ovn-nbctl list load_balancer
_uuid               : c72de15e-5c2e-4c1b-a21b-8e9a6721193c
external_ids        : {enabled=True, "listener_21e77cde-854f-4c3e-bd8c-
↪9536ae0443bc"="64015:pool_898be8a2-5185-4f3b-8658-a56457f595a9", lr_ref=
↪"neutron-3d2a873b-b5b4-4d14-ac24-47a835fd47b2", ls_refs="{\"neutron-
↪ee97665d-69d0-4995-a275-27855359956a\": 1}", "neutron:vip"="172.24.4.9",
↪"neutron:vip_port_id"="c98e52d0-5965-4b22-8a17-a374f4399193", "pool_
↪898be8a2-5185-4f3b-8658-a56457f595a9"="member_adf55e70-3d50-4e62-99fd-
↪dd77eababb1c_10.10.10.10:63015"}
name                : "94e7c431-912b-496c-a247-d52875d44ac7"
protocol            : tcp
vips                : {"172.24.4.9:64015"="10.10.10.10:63015"}
```

[1]: https://docs.openstack.org/kuryr-kubernetes/latest/installation/services.html

CHAPTER

TWO

CONTRIBUTOR DOCUMENTATION

2.1 OpenStack LoadBalancer API and OVN

2.1.1 Introduction

Load balancing is essential for enabling simple or automatic delivery scaling and availability since application delivery, scaling and availability are considered vital features of any cloud. Octavia is an open source, operator-scale load balancing solution designed to work with OpenStack.

The purpose of this document is to propose a design for how we can use OVN as the backend for OpenStacks LoadBalancer API provided by Octavia.

2.1.2 Octavia LoadBalancers Today

A Detailed design analysis of Octavia is available here:

https://docs.openstack.org/octavia/latest/contributor/design/version0.5/component-design.html

Currently, Octavia uses the built-in Amphorae driver to fulfill the Loadbalancing requests in Openstack. Amphorae can be a Virtual machine, container, dedicated hardware, appliance or device that actually performs the task of load balancing in the Octavia system. More specifically, an amphora takes requests from clients on the front-end and distributes these to back-end systems. Amphorae communicates with its controllers over the LoadBalancers network through a driver interface on the controller.

Amphorae needs a placeholder, such as a separate VM/Container for deployment, so that it can handle the LoadBalancers requests. Along with this, it also needs a separate network (termed as lb-mgmt-network) which handles all Amphorae requests.

Amphorae has the capability to handle L4 (TCP/UDP) as well as L7 (HTTP) LoadBalancer requests and provides monitoring features using HealthMonitors.

2.1.3 Octavia with OVN

The OVN native LoadBalancer currently supports L4 protocols, with support for L7 protocols aimed for future releases. It does not need any extra hardware/VM/Container for deployment, which is a major positive point when compared with Amphorae. Also, it does not need any special network to handle the LoadBalancers requests as they are taken care by OpenFlow rules directly. And, though OVN does not have support for TLS, it is in development and once implemented can be integrated with Octavia.

This following section details how OVN can be used as an Octavia driver.

Overview of Proposed Approach

The OVN Driver for Octavia runs under the scope of Octavia. The Octavia API receives and forwards calls to the OVN Driver.

Step 1 - Creating a LoadBalancer

The Octavia API receives and issues a LoadBalancer creation request on a network to the OVN Provider driver. The OVN driver creates a LoadBalancer in the OVN NorthBound DB and asynchronously updates the Octavia DB with the status response. A VIP port is created in Neutron when the LoadBalancer creation is complete. The VIP information however is not updated in the NorthBound DB until the Members are associated with the LoadBalancers Pool.

Step 2 - Creating LoadBalancer entities (Pools, Listeners, Members)

Once a LoadBalancer is created by OVN in its NorthBound DB, users can now create Pools, Listeners and Members associated with the LoadBalancer using the Octavia API. With the creation of each entity, the LoadBalancers *external_ids* column in the NorthBound DB will be updated and corresponding Logical and Openflow rules will be added for handling them.

Step 3 - LoadBalancer request processing

When a user sends a request to the VIP IP address, the OVN pipeline takes care of load balancing the VIP request to one of the backend members. More information about this can be found in the ovn-northd man pages.

OVN LoadBalancer Driver Logic

- On startup: Open and maintain a connection to the OVN Northbound DB (using the ovsdbapp library). On first connection, and anytime a reconnect happens:
  - Do a full sync.
- Register a callback when a new interface is added or deleted from a router or switch. The LogicalSwitchPortUpdateEvent and LogicalRouterPortEvent are registered to process these events.
- When a new LoadBalancer L1 is created, create a Row in OVNs `Load_Balancer` table and update its entries for name and network references. If the network on which the LoadBalancer is created is associated with a router, say R1, then add the router reference to the LoadBalancers *external_ids* and associate the LoadBalancer to the router. Also associate the LoadBalancer L1 with all those networks which have an interface on the router R1. This is required so that Logical Flows for inter-network communication while using the LoadBalancer L1 is possible. Also, during this time, a new port is created via Neutron which acts as a VIP Port. The information of this new port is not visible in OVNs NorthBound DB until a member is added to the LoadBalancer.
- If a new network interface is added to the router R1 described above, all the LoadBalancers on that network are associated with the router R1 and all the LoadBalancers on the router are associated with the new network.
- If a network interface is removed from the router R1, then all the LoadBalancers which have been solely created on that network (identified using the *ls_ref* attribute in the LoadBalancers *external_ids*) are removed from the router. Similarly, those LoadBalancers which are associated with the network but not actually created on that network are removed from the network.
- A LoadBalancer can either be deleted with all its children entities using the *cascade* option, or its members/pools/listeners can be individually deleted. When the LoadBalancer is deleted, its references and associations from all networks and routers are removed. This might change in the

future once the association of LoadBalancers with networks/routers are changed to *weak* from *strong* [3]. Also the VIP port is deleted when the LoadBalancer is deleted.

OVN LoadBalancer at work

OVN Northbound schema [5] has a table to store LoadBalancers. The table looks like:

```
"Load_Balancer": {
    "columns": {
        "name": {"type": "string"},
        "vips": {
            "type": {"key": "string", "value": "string",
                     "min": 0, "max": "unlimited"}},
        "protocol": {
            "type": {"key": {"type": "string",
                             "enum": ["set", ["tcp", "udp"]]},
                             "min": 0, "max": 1}},
        "external_ids": {
            "type": {"key": "string", "value": "string",
                     "min": 0, "max": "unlimited"}}},
        "isRoot": true},
```

There is a `load_balancer` column in the Logical_Switch table (which corresponds to a Neutron network) as well as the Logical_Router table (which corresponds to a Neutron router) referring back to the Load_Balancer table.

The OVN driver updates the OVN Northbound DB. When a LoadBalancer is created, a row in this table is created. When the listeners and members are added, the vips column and the Logical_Switchs `load_balancer` column are updated accordingly.

The ovn-northd service, which monitors for changes to the OVN Northbound DB, generates OVN logical flows to enable load balancing, and ovn-controller running on each compute node translates the logical flows into actual OpenFlow rules.

The status of each entity in the Octavia DB is managed according to [4]

Below are a few examples on what happens when LoadBalancer commands are executed and what changes in the Load_Balancer Northbound DB table.

1. Create a LoadBalancer:

```
$ openstack loadbalancer create --provider ovn --vip-subnet-id=private lb1

$ ovn-nbctl list load_balancer
_uuid         : 9dd65bae-2501-43f2-b34e-38a9cb7e4251
external_ids  : {
    lr_ref="neutron-52b6299c-6e38-4226-a275-77370296f257",
    ls_refs="{\"neutron-2526c68a-5a9e-484c-8e00-0716388f6563\": 1}",
    neutron:vip="10.0.0.10",
    neutron:vip_port_id="2526c68a-5a9e-484c-8e00-0716388f6563"}
name          : "973a201a-8787-4f6e-9b8f-ab9f93c31f44"
protocol      : []
vips          : {}
```

2. Create a pool:

```
$ openstack loadbalancer pool create --name p1 --loadbalancer lb1
--protocol TCP --lb-algorithm SOURCE_IP_PORT

$ ovn-nbctl list load_balancer
_uuid         : 9dd65bae-2501-43f2-b34e-38a9cb7e4251
external_ids  : {
    lr_ref="neutron-52b6299c-6e38-4226-a275-77370296f257",
    ls_refs="{\"neutron-2526c68a-5a9e-484c-8e00-0716388f6563\": 1}",
    "pool_f2ddf7a6-4047-4cc9-97be-1d1a6c47ece9"="", neutron:vip="10.0.0.10
↪",
    neutron:vip_port_id="2526c68a-5a9e-484c-8e00-0716388f6563"}
name          : "973a201a-8787-4f6e-9b8f-ab9f93c31f44"
protocol      : []
vips          : {}
```

3. Create a member:

```
$ openstack loadbalancer member create --address 10.0.0.107
 --subnet-id 2d54ec67-c589-473b-bc67-41f3d1331fef --protocol-port 80 p1

$ ovn-nbctl list load_balancer
_uuid         : 9dd65bae-2501-43f2-b34e-38a9cb7e4251
external_ids  : {
    lr_ref="neutron-52b6299c-6e38-4226-a275-77370296f257",
    ls_refs="{\"neutron-2526c68a-5a9e-484c-8e00-0716388f6563\": 2}",
    "pool_f2ddf7a6-4047-4cc9-97be-1d1a6c47ece9"=
    "member_579c0c9f-d37d-4ba5-beed-cabf6331032d_10.0.0.107:80",
    neutron:vip="10.0.0.10",
    neutron:vip_port_id="2526c68a-5a9e-484c-8e00-0716388f6563"}
name          : "973a201a-8787-4f6e-9b8f-ab9f93c31f44"
protocol      : []
vips          : {}
```

4. Create another member:

```
$ openstack loadbalancer member create --address 20.0.0.107
 --subnet-id c2e2da10-1217-4fe2-837a-1c45da587df7 --protocol-port 80 p1

$ ovn-nbctl list load_balancer
_uuid         : 9dd65bae-2501-43f2-b34e-38a9cb7e4251
external_ids  : {
    lr_ref="neutron-52b6299c-6e38-4226-a275-77370296f257",
    ls_refs="{\"neutron-2526c68a-5a9e-484c-8e00-0716388f6563\": 2,
          \"neutron-12c42705-3e15-4e2d-8fc0-070d1b80b9ef\": 1}",
    "pool_f2ddf7a6-4047-4cc9-97be-1d1a6c47ece9"=
    "member_579c0c9f-d37d-4ba5-beed-cabf6331032d_10.0.0.107:80,
     member_d100f2ed-9b55-4083-be78-7f203d095561_20.0.0.107:80",
    neutron:vip="10.0.0.10",
    neutron:vip_port_id="2526c68a-5a9e-484c-8e00-0716388f6563"}
```

(continues on next page)

(continued from previous page)

```
name          : "973a201a-8787-4f6e-9b8f-ab9f93c31f44"
protocol      : []
vips          : {}
```

5. Create a listener:

```
$ openstack loadbalancer listener create --name l1 --protocol TCP
 --protocol-port 82 --default-pool p1 lb1

$ ovn-nbctl list load_balancer
_uuid         : 9dd65bae-2501-43f2-b34e-38a9cb7e4251
external_ids  : {
    lr_ref="neutron-52b6299c-6e38-4226-a275-77370296f257",
    ls_refs="{\"neutron-2526c68a-5a9e-484c-8e00-0716388f6563\": 2,
              \"neutron-12c42705-3e15-4e2d-8fc0-070d1b80b9ef\": 1}",
    "pool_f2ddf7a6-4047-4cc9-97be-1d1a6c47ece9"="10.0.0.107:80,20.0.0.
↪107:80",
    "listener_12345678-2501-43f2-b34e-38a9cb7e4132"=
        "82:pool_f2ddf7a6-4047-4cc9-97be-1d1a6c47ece9",
    neutron:vip="10.0.0.10",
    neutron:vip_port_id="2526c68a-5a9e-484c-8e00-0716388f6563"}
name          : "973a201a-8787-4f6e-9b8f-ab9f93c31f44"
protocol      : []
vips          : {"10.0.0.10:82"="10.0.0.107:80,20.0.0.107:80"}
```

As explained earlier in the design section:

- If a network N1 has a LoadBalancer LB1 associated to it and one of its interfaces is added to a router R1, LB1 is associated with R1 as well.
- If a network N2 has a LoadBalancer LB2 and one of its interfaces is added to the router R1, then R1 will have both LoadBalancers LB1 and LB2. N1 and N2 will also have both the LoadBalancers associated to them. However, kindly note that although network N1 would have both LB1 and LB2 LoadBalancers associated with it, only LB1 would be the LoadBalancer which has a direct reference to the network N1, since LB1 was created on N1. This is visible in the `ls_ref` key of the `external_ids` column in LB1s entry in the `load_balancer` table.
- If a network N3 is added to the router R1, N3 will also have both LoadBalancers (LB1, LB2) associated to it.
- If the interface to network N2 is removed from R1, network N2 will now only have LB2 associated with it. Networks N1 and N3 and router R1 will have LoadBalancer LB1 associated with them.

2.1.4 Limitations

The Following actions are not supported by the OVN Provider Driver:

- Creating a LoadBalancer/Listener/Pool with an L7 Protocol
- Currently only one algorithm is supported for pool management (Source IP Port)

The following issue exists with OVNs integration with Octavia:

- If creation/deletion of a LoadBalancer, Listener, Pool or Member fails, then the corresponding object will remain in the DB in a PENDING_* state.

2.1.5 Support Matrix

A detailed matrix of the operations supported by OVN Provider driver in Octavia can be found in https://docs.openstack.org/octavia/latest/user/feature-classification/index.html

2.1.6 Other References

[1] Octavia API: https://docs.openstack.org/api-ref/load-balancer/v2/

[2] Octavia Glossary: https://docs.openstack.org/octavia/latest/reference/glossary.html

[3] https://github.com/openvswitch/ovs/commit/612f80fa8ebf88dad2e204364c6c02b451dca36c

[4] https://docs.openstack.org/api-ref/load-balancer/v2/index.html#status-codes

[5] https://github.com/openvswitch/ovs/blob/d1b235d7a6246e00d4afc359071d3b6b3ed244c3/ovn/ovn-nb.ovsschema#L117