Load Balancer as a Service (LBaaS)

The Networking service offers a load balancer feature called “LBaaS v2” through the `neutron-lbaas` service plug-in.

LBaaS v2 adds the concept of listeners to the LBaaS v1 load balancers. LBaaS v2 allows you to configure multiple listener ports on a single load balancer IP address.

There are two reference implementations of LBaaS v2. The one is an agent based implementation with HAProxy. The agents handle the HAProxy configuration and manage the HAProxy daemon. Another LBaaS v2 implementation, Octavia, has a separate API and separate worker processes that build load balancers within virtual machines on hypervisors that are managed by the Compute service. You do not need an agent for Octavia.

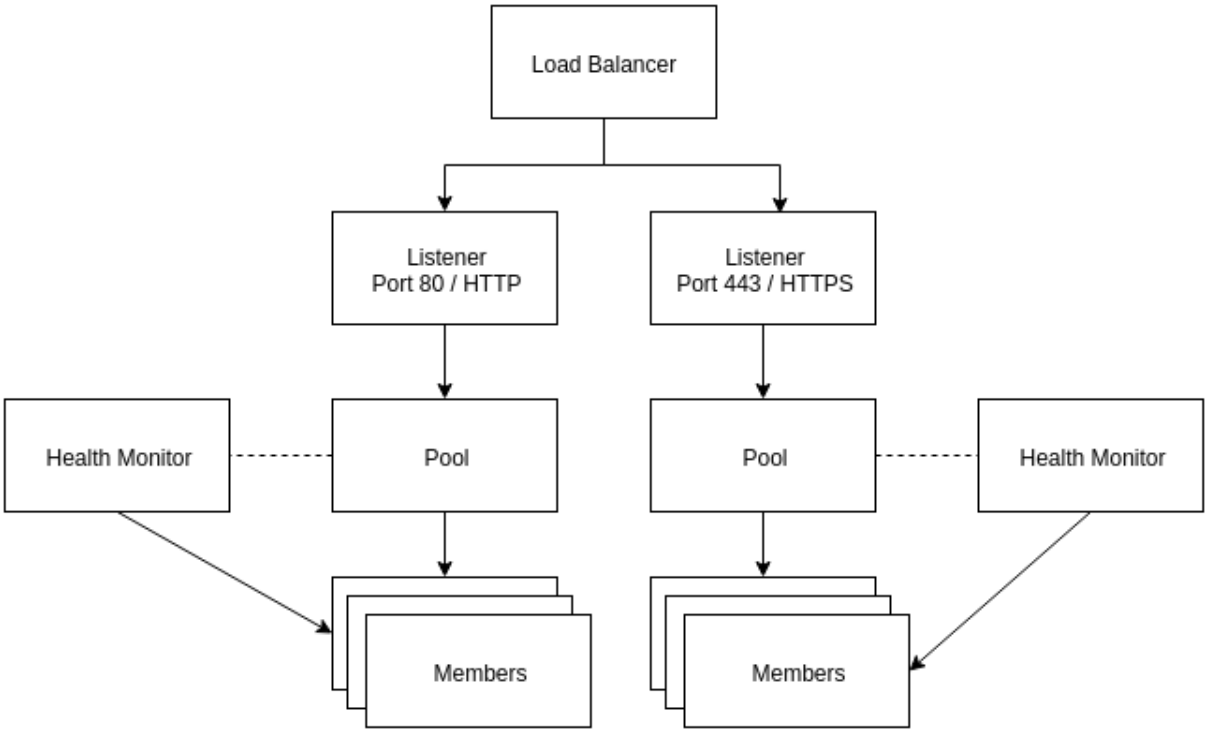
Note: LBaaS v1 was removed in the Newton release. These links provide more details about how LBaaS v1 works and how to configure it:

- Load-Balancer-as-a-Service (LBaaS) overview
- Basic Load-Balancer-as-a-Service operations

Warning: Currently, no migration path exists between v1 and v2 load balancers. If you choose to switch from v1 to v2, you must recreate all load balancers, pools, and health monitors.

LBaaS v2 Concepts

LBaaS v2 has several new concepts to understand:



Load balancer The load balancer occupies a neutron network port and has an IP address assigned from a subnet.

Listener Load balancers can listen for requests on multiple ports. Each one of those ports is specified by a listener.

Pool A pool holds a list of members that serve content through the load balancer.

Member Members are servers that serve traffic behind a load balancer. Each member is specified by the IP address and port that it uses to serve traffic.

Health monitor Members may go offline from time to time and health monitors divert traffic away from members that are not responding properly. Health monitors are associated with pools.

LBaaS v2 has multiple implementations via different service plug-ins. The two most common implementations use either an agent or the Octavia services. Both implementations use the LBaaS v2 API.

Configurations

Configuring LBaaS v2 with an agent

1. Add the LBaaS v2 service plug-in to the`service_plugins` configuration directive in `/etc/neutron/neutron.conf`. The plug-in list is comma-separated:

```
service_plugins = [existing service plugins],neutron_lbaas.services.loadbalancer.
,→plugin.LoadBalancerPluginv2
```

2. Add the LBaaS v2 service provider to the `service_provider` configuration directive within the `[service_providers]` section in `/etc/neutron/neutron_lbaas.conf`:

```
service_provider = LOADBALANCERV2:Haproxy:neutron_lbaas.drivers.haproxy.plugin_driver.
,→HaproxyOnHostPluginDriver:default
```

If you have existing service providers for other networking service plug-ins, such as VPNaaS or FWaaS, add the `service_provider` line shown above in the `[service_providers]` section as a separate line. These configuration directives are repeatable and are not comma-separated.

3. Select the driver that manages virtual interfaces in `/etc/neutron/lbaas_agent.ini`:

```
[DEFAULT]
interface_driver = INTERFACE_DRIVER
```

Replace `INTERFACE_DRIVER` with the interface driver that the layer-2 agent in your environment uses. For example, `openvswitch` for Open vSwitch or `linuxbridge` for Linux bridge.

4. Run the `neutron-lbaas` database migration:

```
neutron-db-manage --subproject neutron-lbaas upgrade head
```

5. If you have deployed LBaaS v1, **stop the LBaaS v1 agent now**. The v1 and v2 agents **cannot** run simultaneously.

6. Start the LBaaS v2 agent:

```
neutron-lbaasv2-agent \
--config-file /etc/neutron/neutron.conf \
--config-file /etc/neutron/lbaas_agent.ini
```

7. Restart the Network service to activate the new configuration. You are now ready to create load balancers with the LBaaS v2 agent.

Configuring LBaaS v2 with Octavia

Octavia provides additional capabilities for load balancers, including using a compute driver to build instances that operate as load balancers. The Hands on Lab - Install and Configure OpenStack Octavia session at the OpenStack Summit in Tokyo provides an overview of Octavia.

The DevStack documentation offers a simple method to deploy Octavia and test the service with redundant load balancer instances. If you already have Octavia installed and configured within your environment, you can configure the Network service to use Octavia:

1. Add the LBaaS v2 service plug-in to the`service_plugins` configuration directive in `/etc/neutron/neutron.conf`. The plug-in list is comma-separated:

```
service_plugins = [existing service plugins],neutron_lbaas.services.loadbalancer.
,→plugin.LoadBalancerPluginv2
```

2. Add the Octavia service provider to the `service_provider` configuration directive within the `[service_providers]` section in `/etc/neutron/neutron_lbaas.conf`:

```
service_provider = LOADBALANCERV2:Octavia:neutron_lbaas.drivers.octavia.driver.
,→OctaviaDriver:default
```

Ensure that the LBaaS v1 and v2 service providers are removed from the `[service_providers]` section. They are not used with Octavia. **Verify that all LBaaS agents are stopped.**

3. Restart the Network service to activate the new configuration. You are now ready to create and manage load balancers with Octavia.

Add LBaaS panels to Dashboard

The Dashboard panels for managing LBaaS v2 are available starting with the Mitaka release.

1. Clone the neutron-lbaas-dashboard repository and check out the release branch that matches the installed version of Dashboard:

```
$ git clone https://git.openstack.org/openstack/neutron-lbaas-dashboard
$ cd neutron-lbaas-dashboard
$ git checkout OPENSTACK_RELEASE
```

2. Install the Dashboard panel plug-in:

```
$ python setup.py install
```

3. Copy the `_1481_project_ng_loadbalancersv2_panel.py` file from the `neutron-lbaas-dashboard/enabled` directory into the Dashboard `openstack_dashboard/local/enabled` directory.

   This step ensures that Dashboard can find the plug-in when it enumerates all of its available panels.

4. Enable the plug-in in Dashboard by editing the `local_settings.py` file and setting `enable_lb` to `True` in the `OPENSTACK_NEUTRON_NETWORK` dictionary.

5. If Dashboard is configured to compress static files for better performance (usually set through `COMPRESS_OFFLINE` in `local_settings.py`), optimize the static files again:

```
$ ./manage.py collectstatic
$ ./manage.py compress
```

6. Restart Apache to activate the new panel:

```
$ sudo service apache2 restart
```

To find the panel, click on *Project* in Dashboard, then click the *Network* drop-down menu and select *Load Balancers*.

LBaaS v2 operations

The same neutron commands are used for LBaaS v2 with an agent or with Octavia.

Building an LBaaS v2 load balancer

1. Start by creating a load balancer on a network. In this example, the `private` network is an isolated network with two web server instances:

```
$ neutron lbaas-loadbalancer-create --name test-lb private-subnet
```

2. You can view the load balancer status and IP address with the **`neutron lbaas-loadbalancer-show`** command:

```
$ neutron lbaas-loadbalancer-show test-lb
+---------------------+------------------------------------------------+
| Field               | Value                                          |
```

```
+---------------------+----------------------------------------------+
| admin_state_up      | True                                         |
| description         |                                              |
| id                  | 7780f9dd-e5dd-43a9-af81-0d2d1bd9c386         |
| listeners           | {"id": "23442d6a-4d82-40ee-8d08-243750dbc191"} |
|                     | {"id": "7e0d084d-6d67-47e6-9f77-0115e6cf9ba8"} |
| name                | test-lb                                      |
| operating_status    | ONLINE                                       |
| provider            | haproxy                                      |
| provisioning_status | ACTIVE                                       |
| tenant_id           | fbfce4cb346c4f9097a977c54904cafd             |
| vip_address         | 192.0.2.22                                   |
| vip_port_id         | 9f8f8a75-a731-4a34-b622-864907e1d556         |
| vip_subnet_id       | f1e7827d-1bfe-40b6-b8f0-2d9fd946f59b         |
+---------------------+----------------------------------------------+
```

3. Update the security group to allow traffic to reach the new load balancer. Create a new security group along with ingress rules to allow traffic into the new load balancer. The neutron port for the load balancer is shown as `vip_port_id` above.

   Create a security group and rules to allow TCP port 80, TCP port 443, and all ICMP traffic:

```
$ neutron security-group-create lbaas
$ neutron security-group-rule-create \
  --direction ingress \
  --protocol tcp \
  --port-range-min 80 \
  --port-range-max 80 \
  --remote-ip-prefix 0.0.0.0/0 \
  lbaas
$ neutron security-group-rule-create \
  --direction ingress \
  --protocol tcp \
  --port-range-min 443 \
  --port-range-max 443 \
  --remote-ip-prefix 0.0.0.0/0 \
  lbaas
$ neutron security-group-rule-create \
  --direction ingress \
  --protocol icmp \
  lbaas
```

   Apply the security group to the load balancer's network port using `vip_port_id` from the **`neutron lbaas-loadbalancer-show`** command:

```
$ neutron port-update \
  --security-group lbaas \
  9f8f8a75-a731-4a34-b622-864907e1d556
```

Adding an HTTP listener

1. With the load balancer online, you can add a listener for plaintext HTTP traffic on port 80:

```
$ neutron lbaas-listener-create \
  --name test-lb-http \
```

```
  --loadbalancer test-lb \
  --protocol HTTP \
  --protocol-port 80
```

This load balancer is active and ready to serve traffic on `192.0.2.22`.

2. Verify that the load balancer is responding to pings before moving further:

```
$ ping -c 4 192.0.2.22
PING 192.0.2.22 (192.0.2.22) 56(84) bytes of data.
64 bytes from 192.0.2.22: icmp_seq=1 ttl=62 time=0.410 ms
64 bytes from 192.0.2.22: icmp_seq=2 ttl=62 time=0.407 ms
64 bytes from 192.0.2.22: icmp_seq=3 ttl=62 time=0.396 ms
64 bytes from 192.0.2.22: icmp_seq=4 ttl=62 time=0.397 ms

--- 192.0.2.22 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2997ms
rtt min/avg/max/mdev = 0.396/0.402/0.410/0.020 ms
```

3. You can begin building a pool and adding members to the pool to serve HTTP content on port 80. For this example, the web servers are `192.0.2.16` and `192.0.2.17`:

```
$ neutron lbaas-pool-create \
  --name test-lb-pool-http \
  --lb-algorithm ROUND_ROBIN \
  --listener test-lb-http \
  --protocol HTTP
$ neutron lbaas-member-create \
  --name test-lb-http-member-1 \
  --subnet private-subnet \
  --address 192.0.2.16 \
  --protocol-port 80 \
  test-lb-pool-http
$ neutron lbaas-member-create \
  --name test-lb-http-member-2 \
  --subnet private-subnet \
  --address 192.0.2.17 \
  --protocol-port 80 \
  test-lb-pool-http
```

4. You can use `curl` to verify connectivity through the load balancers to your web servers:

```
$ curl 192.0.2.22
web2
$ curl 192.0.2.22
web1
$ curl 192.0.2.22
web2
$ curl 192.0.2.22
web1
```

In this example, the load balancer uses the round robin algorithm and the traffic alternates between the web servers on the backend.

5. You can add a health monitor so that unresponsive servers are removed from the pool:

```
$ neutron lbaas-healthmonitor-create \
  --name test-lb-http-monitor \
  --delay 5 \
  --max-retries 2 \
  --timeout 10 \
  --type HTTP \
  --pool test-lb-pool-http
```

In this example, the health monitor removes the server from the pool if it fails a health check at two five-second intervals. When the server recovers and begins responding to health checks again, it is added to the pool once again.

Adding an HTTPS listener

You can add another listener on port 443 for HTTPS traffic.LBaaS v2 offers SSL/TLS termination at the load balancer, but this example takes a simpler approach and allows encrypted connections to terminate at each member server.

1. Start by creating a listener, attaching a pool, and then adding members:

```
$ neutron lbaas-listener-create \
  --name test-lb-https \
  --loadbalancer test-lb \
  --protocol HTTPS \
  --protocol-port 443
$ neutron lbaas-pool-create \
  --name test-lb-pool-https \
  --lb-algorithm LEAST_CONNECTIONS \
  --listener test-lb-https \
  --protocol HTTPS
$ neutron lbaas-member-create \
  --name test-lb-https-member-1 \
  --subnet private-subnet \
  --address 192.0.2.16 \
  --protocol-port 443 \
  test-lb-pool-https
$ neutron lbaas-member-create \
  --name test-lb-https-member-2 \
  --subnet private-subnet \
  --address 192.0.2.17 \
  --protocol-port 443 \
  test-lb-pool-https
```

2. You can also add a health monitor for the HTTPS pool:

```
$ neutron lbaas-healthmonitor-create \
  --name test-lb-https-monitor \
  --delay 5 \
  --max-retries 2 \
  --timeout 10 \
  --type HTTPS \
  --pool test-lb-pool-https
```

The load balancer now handles traffic on ports 80 and 443.

Associating a floating IP address

Load balancers that are deployed on a public or provider network that are accessible to external clients do not need a floating IP address assigned. External clients can directly access the virtual IP address (VIP) of those load balancers.

However, load balancers deployed onto private or isolated networks need a floating IP address assigned if they must be accessible to external clients. To complete this step, you must have a router between the private and public networks and an available floating IP address.

You can use the **`neutron lbaas-loadbalancer-show`** command from the beginning of this section to locate the `vip_port_id`. The `vip_port_id` is the ID of the network port that is assigned to the load balancer. You can associate a free floating IP address to the load balancer using **`neutron floatingip-associate`**:

```
$ neutron floatingip-associate FLOATINGIP_ID LOAD_BALANCER_PORT_ID
```

Setting quotas for LBaaS v2

Quotas are available for limiting the number of load balancers and load balancer pools. By default, both quotas are set to 10.

You can adjust quotas using the **`neutron quota-update`** command:

```
$ neutron quota-update --tenant-id TENANT_UUID --loadbalancer 25
$ neutron quota-update --tenant-id TENANT_UUID --pool 50
```

A setting of `-1` disables the quota for a tenant.

Retrieving load balancer statistics

The LBaaS v2 agent collects four types of statistics for each load balancer every six seconds. Users can query these statistics with the **`neutron lbaas-loadbalancer-stats`** command:

```
$ neutron lbaas-loadbalancer-stats test-lb
+--------------------+----------+
| Field              | Value    |
+--------------------+----------+
| active_connections | 0        |
| bytes_in           | 40264557 |
| bytes_out          | 71701666 |
| total_connections  | 384601   |
+--------------------+----------+
```

The `active_connections` count is the total number of connections that were active at the time the agent polled the load balancer. The other three statistics are cumulative since the load balancer was last started. For example, if the load balancer restarts due to a system error or a configuration change, these statistics will be reset.