Difference between RackSpace Cloud LoadBalancers API and OpenStack API Proposal

This note describes the edits made to the Rackspace Cloud Load Balancers API to produce a core specification API that can be implemented by several solutions.

Section 2.4.1

- The exact criteria used by passive monitoring for verifying the condition of a node has been removed. Different vendors and implementers support and set up different criteria specific to the technology used behind the loadbalancers.

Section 2.4.2

- Removed the sentence:

  "One of the advantages of active health monitoring is that it does not require active transactions to be processed by the load balancer to determine whether or not a node is suitable for handling traffic."

Because passive (default) monitoring might also use synthetic transactions in some solutions (like Citrix NetScaler), so this is not a difference between active and passive monitoring in all solutions.

- Replaced the sentence:

  "When both active and passive monitoring are enabled, the decisions made by the active monitor override inferences made by the passive monitoring system"

  By:

  "When active monitoring is enabled, it takes over the monitoring of the node, and passive monitoring is disabled. Conversely, when active monitoring configuration is removed by the user, passive monitoring is re-enabled for the nodes of the load balancer."

Section 2.6

- Replaced "Apache-style logs" by "logs" as each provider might offer different formats for logs.
- Removed "Logs are delivered to the account every hour" as the way and period of delivering logs to a user differs between solutions and implementations.

Section 3

- Removed this section as this should apply to all OpenStack APIs (OpenStack Compute API, OpenStack Storage API). The OpenStack community hasn't decided on the general pattern that all OpenStack services should adhere to. Currently this section is found in all RackSpace Cloud APIs (Cloud Servers, Cloud Files and Cloud Load Balancers).

- May need to revisit this decision if a common framework that implements this part can be used for all OpenStack services.

Section 4.1

- Removed the paragraph:

  > "Load balancers which have been deleted will be shown in this list for at least 90 days after deletion. A deleted load balancer is immutable and irrecoverable. Only a limited set of attributes (id,name,status, created, and updated) will be returned in the response object."

  This is an operational matter for the cloud provider whether to allow access to deleted objects for a certain period of time after deletion.

Section 4.2

- Removed Atom as a support format for the API. The core API supports XML and JSON format. Atom support can be considered by extensions to this API.

- Removed the sentence

  > This operation is not capable of returning details for a load balancer which has been deleted.

  See comment on section 4.1 above.

- Changed the API to accept and return only one Virtual IP per load balancer. Not all solutions support multiple VIPs per a load balancer. Vendors and cloud providers that support more than virtual IP per load balancer can add this in their own extensions.

Section 4.3

- Removed the note about the load balancer's name maximum length, as this limit varies between implementations.

- Changed the payload of Request to include only one “virtualIP” element instead of a list (see comments in section 4.2).

- Replace paragraph:

  > “In order to conserve IPv4 address space, Rackspace highly recommends sharing Virtual IPs between your load balancers. If you have at least one load balancer, you may create subsequent load balancers that share a single virtual IP by issuing a POST and supplying a virtual Ip ID instead of a type. Additionally, this feature is highly desirable if you wish to load balance both an unsecured and secure protocol using one IP / DNS name (for example, HTTP and HTTPS).”

  With:

  > “If you have at least one load balancer, you may create subsequent load balancers that share a single virtual IP by issuing a POST and supplying a virtual IP address instead of a type. Additionally, this feature is highly desirable if you wish to load balance both an unsecured and secure protocol using one IP / DNS name (for example, HTTP and HTTPS).”

- Removed the “id” attribute for a virtualIP element (the server doesn’t return one). When a user wants to use an existing VIP, it can be referenced by its address value. Changed the example of creating a load balancer with a shared IP accordingly.

Section 4.5

- Changed the following sentence in the second paragraph:

  > “All nodes have an associated status that indicates whether the node is online, offline, or draining.”

  To

  > “All nodes have an associated status that indicates whether the node is online or offline.”

  Draining mode for nodes is not supported by all solutions.
- Removed the following sentence in the same paragraph:

"A node in DRAINING status represents a node that stops the traffic manager from sending any additional new connections to the node, but honors established sessions. If the traffic manager receives a request and session persistence requires that the node is used, the traffic manager will use it."

- Replaced the following paragraph

  > "If the WEIGHTED_ROUND_ROBIN load balancer algorithm mode is selected, then the caller should assign the relevant weights to the node as part of the weight attribute of the node element. When the algorithm of the load balancer is changed to WEIGHTED_ROUND_ROBIN and the nodes do not already have an assigned weight, the service will automatically set the weight to "1" for all nodes."

  With

  > "The caller can assign weights to the nodes as part of the weight attribute of the node element. When the nodes do not already have an assigned weight, the service will automatically set the weight to "1" for all nodes."

  The reason for the change is that not all solutions make a distinction between "weighted" version of algorithms and "non-weighted" ones. For example, in Citrix NetScaler product, LB algorithms support weights. There is no version of the algorithms that doesn't allow weights to be set on nodes.

- Changed the sentence

  > "When a node is added, it is assigned a unique identifier that can be used for mutating operations such as changing the port and condition or removing it"

  With

  > "When a node is added, it is assigned a unique identifier that can be used for mutating operations such as changing the ~~port and~~ condition or removing it"

  as some solutions do not support changes to all the attributes of a load balancer after creation.

- Removed the sentence

  > "Every load balancer is dual-homed on both the public Internet and ServiceNet; as a result, nodes can either be internal ServiceNet addresses or addresses on the public Internet."

  As the way the load-balancer is setup is a deployment issue for the cloud provider or the solution used.

- Separated the operations List, Get, Add, Update and Remove in their own sections each with its possible error codes.

- Changed the example of “List Nodes” so that the response payload is a list of one node (rather that a node) as a list operation should always return a list.

- Removed from the Note on page 22, the following sentence:

  “Additionally, a load balancer supports a maximum of 25 nodes”

  As this should be a configurable settings and would vary from one deployment to another.

- Corrected the payload of “Update Node” request in JSON to the following:

  { “condition”:”DISABLED”}

Section 4.6

- Edited a sentence in the first paragraph as follows:

  A caller can ~~subscribe to event notifications through the notification service or~~ poll the load balancer with its ID to wait for the changes to be applied and the load balancer to return to an ACTIVE status.

  Not all solutions will provide a notification service for changes.

- Removed the ability to update protocol and port of a load balancer as not all solutions support this:

  “This operation allows the caller to change one or more of the following attributes:

  - name
  - algorithm
  - ~~protocol~~
  - ~~port~~

  “

- For the same reason, extended the note in page 24 as follows:

  “The load balancer's ID**, port, protocol** and status are immutable attributes and cannot be modified by the caller. Supplying an unsupported attribute will result in a 400 (badRequest) fault.”

Section 4.7

- This section was removed since the proposal assumes one virtual IP per load balancer only. Also virtual IPs do not have an ID, they are reference by their address value. As a consequence, there is no need for virtual IP management.

Section 4.8

- Edited the first paragraph as follows:

  "The load balancer usage reports provide a view of all transfer~~, average number of connections and number of virtual IPs~~ associated with the load balancing service. ~~Current usage represents all usage recorded within the last 24 hours.~~ Values for both incomingTransfer and outgoingTransfer are expressed in bytes transferred. ~~The optional startTime and endTime parameters can be used to filter all usage. If the startTime parameter is supplied, but the endTime parameter is not then all usage beginning with the startTime will be provided. Likewise, if the endTime parameter is supplied but the startTime parameter is not then all usage will be returned up to the endTime specified.~~"

  Usage data would provide as a minimum the amount of data in and out. Any other usage statistics will vary depending on the solution used, and should therefore be supported in extensions.

- Removed the Note about keeping usage data for 90 days as this is a deployment-specific policy.

- Changed the examples accordingly to only return incomingTransfer and outgoingTransfer values.

- Removed the example of AccountBilling on page 29, as we consider this beyond the scope of this specification.

Section 4.9

- Removed this section from this specification, as we consider access control to be a useful feature for all OpenStack services and not only LB service. An OpenStack Access Control/Firewall service could be specified separately.

Section 4.10: Active Health Monitoring

- Removed the sentence

"Only one health monitor is allowed to be enabled on a load balancer at a time."

Because extensions to this API might want to support multiple monitors on a load balancer. It would be too restrictive for the core API to impose this constraint. If an extension supports multiple monitors per load balancer, then this API would act on the 1st monitor.

Section 4.10.1: Connection Monitor

- Clarified the meaning the "delay" parameter of monitors as follows:

  "delay: The minimum number of seconds to wait before ~~executing the health monitor~~ **checking the health of a node after it is put into OFFLINE status**."

Section 4.10.2: HTTP & HTTPS Monitor

- Removed repeat of the explanation of the 3 Connect parameters (delay, timeout and attemptsBeforeDeactivation) in this section.

- To allow for other solutions to support this API, the following changes were made to the parameters of these monitors:

  - Added a new parameter to this monitor called "method":
    method: The HTTP method (GET or HEAD) that will be used in the sample request.

  - Removed the parameters "StatusRegex". A health check is considered successful if the response has a status code in the 2xx or 3xx ranges.

  - Removed the parameter "bodyRegex"

  "StatusRegex" and "bodyRegex" (as well as other more sophisticated filtering mechanisms) could be supported in extensions to this API.

- Changed the examples in this section to take into account the changes mentioned above.

Section 4.11: Session Persistence

- Changed the JSON response body examples to be consistent with the remaining examples to the following

  {“persistenceType”:”HTTP_COOKIE”}

Section 4.12: Connection Logging

- Changed the JSON response body examples to be consistent with the remaining examples to the following

  {“enabled”:”true”}

Section 4.13: Connection Throttling

- Removed 2 parameters (minConnections and maxConnections) from the connection throttling configuration, as not all solutions provide these parameters:

  ~~minConnection: Allow at least this number of connections per IP address before applying throttling restrictions.~~

  ~~maxConnections: Maximum number of connection to allow for a single IP address.~~

- Changed maxConnectionRate to maxRequestRate (number of requests received in the time interval).

- Clarified the meaning of the remaining 2 parameters as follows:

  maxRequestRate: Maximum number of requests allowed from ~~a single IP address~~ **one client source IP address** in the defined rateInterval.

  rateInterval: Frequency (in seconds) at which the maxConnectionRate is assessed. For example, a maxConnectionRate of 30 with a rateInterval of 60 would allow a maximum of 30 connections per minute ~~for~~ **from** a single **source** IP address.

- Adjusted the examples in this section accordingly

Section 4.14: Load Balancing Protocols

- Removed the following protocols from the core specification: IMAPv4, POP3, LDAP, LDAPS, IMAPS, POP3S, SMTP.
    - **Note**: these protocols could be replaced in the core specification by a generic "TCP" protocol type if agreed.

Section 4.15: Load Balancing Algorithms

- Removed RANDOM from the list of supported algorithms in the core specification. Extended specifications can add support for RANDOM as well as other LB algorithms.

- Removed the sentence

    "The default algorithm for newly created load balancers is RANDOM, which can be overridden at creation time or changed after the load balancer has been initially provisioned."

  Not all solutions support the RANDOM algorithm. Therefore it cannot be the default. The core specification requires that an algorithm is always specified. Extensions may relax this by introducing a default.

- Removed the distinction between "WEIGHTED" and "non-WEIGHTED" algorithms. The core specification assumes that algorithms by default support setting "weights" on nodes. Therefore the Load Balancing Algorithms table 4.4 is changed to the following:

Name	Description
LEAST_CONNECTIONS	The node with the lowest number of connections will receive requests. Each connection will be assigned to a node based on the number of concurrent connections to the node and its weight.
ROUND_ROBIN	Connections are routed to each of the backend servers in turn, with different proportions of traffic being direct to a node depending on its weight.

Where "LEAST_CONNECTIONS" and "ROUND_ROBIN" are changed to weighted algorithms, and there would therefore be no need for having "WEIGHTED_LEAST_CONNECTIONS" and "WEIGHTED_ROUND_ROBIN" in the specification.

Section 4.16: Load Balancer status

- Removed references to “contact support” as the action to be taken by the user is better left to be specified by the service provider

Other

- Added a section in the core specification (section 5) that allows vendors and the OpenStack community to extend the API in a compatible manner. This section gives examples about how to discover the existence of extensions and how to make use of them without breaking compatibility with the core specification.

Section 4.17: Node condition

- Removed the “DRAINING” condition as this is not a widely supported feature.

- For the “DISABLED” condition. Removed the sentence “Existing connections are forcibly terminated” from the description, as this varies between implementations, whether existing connections are kept or forcibly terminated.