The LMA collector plugin for Fuel Documentation

Release 0.7.0

Mirantis Inc.

November 03, 2015

Contents

CHAPTER 1

Introduction

The Mirantis OpenStack LMA (Logging, Monitoring and Alerting) Toolchain is comprised of a collection of open-source tools to help you monitor and diagnose problems in your OpenStack environment. These tools are packaged and delivered as Fuel plugins you can install from within the graphic user interface of Fuel starting with Mirantis OpenStack version 6.1.

From a high level view, the LMA Toolchain includes:

- The LMA Collector (or just the Collector) to gather all operational data that we think are relevant to increase the **operational visibility** over your OpenStack environment. Those data are collected from a variety of sources including the log messages, collectd, and the OpenStack notifications bus
- Pluggable external systems we call **satellite clusters** which can take action on the data received from the Collectors running on the OpenStack nodes.

The Collector is best described as a **pluggable message processing and routing pipeline**. Its core components are :

- Collectd that is bundled with a collection of monitoring plugins. Many of them are purpose-built for OpenStack.
- Heka which is the cornerstone component of the Collector.
- A collection of Heka plugins written in Lua to decode, process and encode the data to be sent to external systems.

The primary function of the Collector is to transform the acquired raw operational data into an internal message representation that is based on the Heka message structure. that can be further exploited to, for example, detect anomalies or create new metric messages.

The satellite clusters delivered as part of the LMA Toolchain starting with Mirantis OpenStack 6.1 include:

- Elasticsearch, a powerful open source search server based on Lucene and analytics engine that makes data like log messages and notifications easy to explore and analyse.
- InfluxDB, an open-source and distributed time-series database to store and search metrics.

By combining Elasticsearch with Kibana, the LMA Toolchain provides an effective way to search and correlate all service-affecting events that occurred in the system for root cause analysis.

Likewise, by combining InfluxDB with Grafana, the LMA Toolchain brings you insightful metrics analytics to visualise how OpenStack behaves over time. This includes metrics for the OpenStack services status and a variety of resource usage and performance indicators. The ability to visualise time-series over a period of time that can vary from 5 minutes to the last 30 days helps anticipating failure conditions and plan capacity ahead of time to cope with a changing demand.

Furthermore, the LMA Toolchain has been designed with the dual objective to be both insightful and adaptive.

It is, for example, quite possible (without any code change) to integrate the Collector with an external monitoring application like Nagios. This could simply be done through enabling the Nagios output plugin of Heka for a subset of

messages matching the message matcher syntax of the output plugin. You should probably not modify the configuration of the LMA Collector manually but apply any configuration change to the Puppet manifests that are shipped with the LMA Collector plugin for Fuel. Many other integration combinations are possible thanks to the extreme flexibility of Heka.

We recommend you to read the Heka documentation to become more familiar with that technology.

The rest of this document is organised in several chapters that will take you through a description of the internal message structure for the categories of operational data that are handled by the LMA Toolchain.

Table of Contents

2.1 Common Message Format

Heka turns the incoming data into Heka messages [1] with a well-defined format which is described below.

- **Timestamp** (number), the timestamp of the message (in nanoseconds since the Epoch).
- **Logger** (string), the datasource from the Heka's standpoint.
- **Type** (string), the type of message.
- **Hostname** (string), the name of the host that emitted the message.
- **Severity** (number), severity level as defined by the Syslog RFC 5424.
- **Payload** (string), the input data in most cases.
- **Pid** (number), the Process ID that generated the message.
- **Fields**, array of Field structures (see below).

2.1.1 Field Format

Every message (either originating from logs, metrics or notifications) is populated with a set of predefined fields:

Attributes in **bold** are always present in the messages while attributes in *italic* are optional.

- **deployment_mode** (string), the deployment mode of the Fuel environment (either 'multinode' or 'ha_compact').
- **deployment_id** (number), the deployment identifier of the Fuel environment.
- **openstack_region** (string), the name of the OpenStack region.
- **openstack_release** (string), the name of the OpenStack release.
- **openstack_roles** (string), a comma-separated list of the node's roles (eg 'controller', 'compute,cinder').
- *environment_label* (string), the label assigned to the OpenStack environment.

Note: All date/time fields represented as string are formatted according to the RFC3339 document.

[1] Heka message structure

2.2 Log Messages

The Heka collector service is configured to tail the following log files:

- System logs.
  - `/var/log/syslog`
  - `/var/log/messages`
  - `/var/log/debug`
  - `/var/log/auth.log`
  - `/var/log/cron.log`
  - `/var/log/daemon.log`
  - `/var/log/kern.log`
  - `/var/log/pacemaker.log`
- MySQL server logs (for controller nodes).
- RabbitMQ server logs (for controller nodes).
- Pacemaker logs (for controller nodes).
- OpenStack logs.
- Open vSwitch logs (all nodes).
  - `/var/log/openvswitch/ovsdb-server.log`
  - `/var/log/openvswitch/ovs-vswitchd.log`

2.2.1 Log Messages Format

In addition to the common *Common Message Format*, log-based messages have additional properties.

Attributes in **bold** are always present in the messages while attributes in *italic* are optional.

- **Logger** (string), `system.<service>`, `mysql` or `openstack.<service>`.
- **Type** (string), always `log`.
- **Fields**
  - **severity_label** (string), the textual representation of the severity level.
  - *programname* (string), the application name for Syslog-based messages.
  - *syslogfacility* (number), the Syslog facility for Syslog-based messages.
  - *http_method* (string), the HTTP method (for instance 'GET').
  - *http_client_ip_address* (string), the IP address of the client that originated the HTTP request.
  - *http_response_size* (number), the size of the HTTP response (in bytes).
  - *http_response_time* (number), the HTTP response time (in seconds).
  - *http_status* (string), the HTTP response status.
  - *http_url* (string), the requested HTTP URL.
  - *http_version* (string), the HTTP version (eg '1.1).

  - *request_id* (string), the UUID of the OpenStack request to which the message applies.
  - *tenant_id* (string), the UUID of the OpenStack tenant to which the message applies.
  - *user_id* (string), the UUID of the OpenStack user to which the message applies.

2.3 Notification Messages

OpenStack services can be configured to send notifications on the message bus about the executing task or the state of the cloud resources [2]. These notifications are received by the LMA collector service and turned into Heka messages.

2.3.1 Notification Messages Format

In addition to the common *Common Message Format*, notification-based messages have additional properties.

Attributes in **bold** are always present in the messages while attributes in *italic* are optional.

- **Logger** (string), the OpenStack service that emitted the notification, (eg, `nova`).
- **Payload** (string), the payload of the OpenStack notification.
- **Hostname** (string), the name of the host that originated the notification.
- **Type** (string), always `notification`.
- **Fields**
  - **hostname** (string), the name of the host that originated the notification.
  - **publisher** (string), the name of the underlying service that emitted the notification (eg, `scheduler`).
  - **severity_label** (string), the textual representation of the severity level.
  - **event_type** (string), the notification's type (eg `compute.instance.create.end`).
  - *tenant_id* (string), the UUID of the OpenStack tenant to which the message applies.
  - *user_id* (string), the UUID of the OpenStack user to which the message applies.
  - *instance_id* (string), the UUID of the virtual instance to which the message applies.
  - *image_name* (string), the image used by the image.
  - *display_name* (string), the visible name of the resource.
  - *instance_type* (string), the type of instance (eg `m1.small`).
  - *availability_zone* (string), the availability zone of the instance.
  - *vcpus* (number), the number of VCPU provisioned for the instance.
  - *memory_mb* (number), the amount of RAM provisioned for the instance.
  - *disk_gb* (number), the disk space provisioned for the instance.
  - *old_state* (string), the previous state of the instance (eg `building`).
  - *state* (string), the state of the instance (eg `active`).
  - *old_task_state* (string), the previous task state for the instance (eg `block_device_mapping`).
  - *new_task_state* (string), the new task state for the instance (eg `spawning`).

[2] OpenStack notifications

  - *created_at* (string): the date of creation of the instance.
  - *launched_at* (string): the date when the instance was effectively launched.
  - *deleted_at* (string): the date of deletion of the instance.
  - *terminated_at* (string): the date when the instance was effectively terminated.

2.4 Metric Messages

Metrics are extracted from several sources:

- Data received from collectd.
- Log messages processed by the collector service.
- OpenStack notifications processed by the collector service.

2.4.1 Metric Messages Format

In addition to the common *Common Message Format*, metric messages have additional properties.

Attributes in **bold** are always present in the messages while attributes in *italic* are optional.

- **Logger** (string), the datasource from the Heka's standpoint, it can be `collectd`, `notification_processor` or `http_log_parser`.
- **Type** (string), either `metric` or `heka.sandbox.metric` (for metrics derived from other messages).
- **Severity** (number), it is always equal to 6 (eg INFO).
- **Fields**
- **name** (string), the name of the metric. See *List of metrics* for the current metrics names that are emitted.
- **value** (number), the value associated to the metric.
- **type** (string), the metric's type, either `gauge` (a value that can go up or down), `counter` (an always increasing value) or `derive` (a per-second rate).
- **source** (string), the source from where the metric comes from, it can be the name of the collectd plugin, `<service>-api` for HTTP response metrics.
- **hostname** (string), the name of the host to which the metric applies. It may be different from the `Hostname` value. For instance when the metric is extracted from an OpenStack notification, `Hostname` is the host that captured the notification and `Fields[hostname]` is the host that emitted the notification.
- *device* (string), the name of the physical device. For instance `sda` or `eth0`.
- *interval* (number), the interval at which the metric is emitted (for `collectd` metrics).
- *tenant_id* (string), the UUID of the OpenStack tenant to which the metric applies.
- *user_id* (string), the UUID of the OpenStack user to which the metric applies.

2.4.2 List of metrics

This is the list of metrics that are emitted by the LMA collector service. They are listed by category then by metric name.

System

CPU

- `cpu.<cpu number>.idle`, percentage of CPU time spent in the idle task.
- `cpu.<cpu number>.interrupt`, percentage of CPU time spent servicing interrupts.
- `cpu.<cpu number>.nice`, percentage of CPU time spent in user mode with low priority (nice).
- `cpu.<cpu number>.softirq`, percentage of CPU time spent servicing soft interrupts.
- `cpu.<cpu number>.steal`, percentage of CPU time spent in other operating systems.
- `cpu.<cpu number>.system`, percentage of CPU time spent in system mode.
- `cpu.<cpu numbner>.user`, percentage of CPU time spent in user mode.
- `cpu.<cpu number>.wait`, percentage of CPU time spent waiting for I/O operations to complete.

`<cpu number>` expands to 0, 1, 2, and so on.

Disk

- `disk.<disk device>.disk_merged.read`, the number of read operations per second that could be merged with already queued operations.
- `disk.<disk device>.disk_merged.write`, the number of write operations per second that could be merged with already queued operations.
- `disk.<disk device>.disk_octets.read`, the number of octets (bytes) read per second.
- `disk.<disk device>.disk_octets.write`, the number of octets (bytes) written per second.
- `disk.<disk device>.disk_ops.read`, the number of read operations per second.
- `disk.<disk device>.disk_ops.write`, the number of write operations per second.
- `disk.<disk device>.disk_time.read`, the average time for a read operation to complete in the last interval.
- `disk.<disk device>.disk_time.write`, the average time for a write operation to complete in the last interval.

`<disk device>` expands to 'sda', 'sdb' and so on.

File system

- `fs.<mount point>.inodes.free`, the number of free inodes on the file system.
- `fs.<mount point>.inodes.reserved`, the number of reserved inodes.
- `fs.<mount point>.inodes.used`, the number of used inodes.
- `fs.<mount point>.space.free`, the number of free bytes.
- `fs.<mount point>.space.reserved`, the number of reserved bytes.
- `fs.<mount point>.space.used`, the number of used bytes.

`<mount point>` expands to 'root' for '/', 'boot' for '/boot', 'var-lib' for '/var/lib' and so on.

System load

- `load.longterm`, the system load average over the last 15 minutes.
- `load.midterm`, the system load average over the last 5 minutes.
- `load.shortterm`, the system load averge over the last minute.

Memory

- `memory.buffered`, the amount of memory (in bytes) which is buffered.
- `memory.cached`, the amount of memory (in bytes) which is cached.
- `memory.free`, the amount of memory (in bytes) which is free.
- `memory.used`, the amount of memory (in bytes) which is used.

Network

- `net.<interface>.if_errors.rx`, the number of errors per second detected when receiving from the interface.
- `net.<interface>.if_errors.tx`, the number of errors per second detected when transmitting from the interace.
- `net.<interface>.if_octets.rx`, the number of octets (bytes) received per second by the interace.
- `net.<interface>.if_octets.tx`, the number of octets (bytes) transmitted per second by the interface.
- `net.<interface>.if_packets.rx`, the number of packets received per second by the interace.
- `net.<interface>.if_packets.tx`, the number of packets transmitted per second by the interface.

`<interface>` expands to the interface name, eg 'br-mgmt', 'br-storage' and so on.

Processes

- `processes.fork_rate`, the number of processes forked per second.
- `processes.state.blocked`, the number of processes in blocked state.
- `processes.state.paging`, the number of processes in paging state.
- `processes.state.running`, the number of processes in running state.
- `processes.state.sleeping`, the number of processes in sleeping state.
- `processes.state.stopped`, the number of processes in stopped state.
- `processes.state.zombies`, the number of processes in zombie state.

Swap

- `swap.cached`, the amount of cached memory (in bytes) which is in the swap.
- `swap.free`, the amount of free memory (in bytes) which is in the swap.
- `swap.used`, the amount of used memory (in bytes) which is in the swap.

- `swap_io.in`, the number of swap pages written per second.
- `swap_io.out`, the number of swap pages read per second.

Users

- `users`, the number of users currently logged-in.

Apache

- `apache.status`, the status of the Apache service, 1 if it is responsive, 0 otherwise.
- `apache.bytes`, the number of bytes per second transmitted by the server.
- `apache.requests`, the number of requests processed per second.
- `apache.connections`, the current number of active connections.
- `apache.idle_workers`, the current number of idle workers.
- `apache.workers.<state>`, the current number of workers by state.

`<state>` is one of `closing`, `dnslookup`, `finishing`, `idle_cleanup`, `keepalive`, `logging`, `open`, `reading`, `sending`, `starting`, `waiting`.

MySQL

Service

- `mysql`, the status of the MySQL service, 1 if it is responsive, 0 otherwise.

Commands

The **mysql_commands.** metrics report how many times per second each statement has been executed.

- `mysql_commands.admin_commands`, the number of ADMIN statements.
- `mysql_commands.change_db`, the number of USE statements.
- `mysql_commands.commit`, the number of COMMIT statements.
- `mysql_commands.flush`, the number of FLUSH statements.
- `mysql_commands.insert`, the number of INSERT statements.
- `mysql_commands.rollback`, the number of ROLLBACK statements.
- `mysql_commands.select`, the number of SELECT statements.
- `mysql_commands.set_option`, the number of SET statements.
- `mysql_commands.show_collations`, the number of SHOW COLLATION statements.
- `mysql_commands.show_databases`, the number of SHOW DATABASES statements.
- `mysql_commands.show_fields`, the number of SHOW FIELDS statements.
- `mysql_commands.show_master_status`, the number of SHOW MASTER STATUS statements.
- `mysql_commands.show_status`, the number of SHOW STATUS statements.

- mysql_commands.show_tables, the number of SHOW TABLES statements.
- mysql_commands.show_variables, the number of SHOW VARIABLES statements.
- mysql_commands.show_warnings, the number of SHOW WARNINGS statements.
- mysql_commands.update, the number of UPDATE statements.

Handlers

The **mysql_handler.** metrics report how many times per second each handler has been executed.

- mysql_handler.commit, the number of internal COMMIT statements.
- mysql_handler.delete, the number of internal DELETE statements.
- mysql_handler.external_lock, the number of external locks.
- mysql_handler.read_first, the number of times the first entry in an index was read.
- mysql_handler.read_key, the number of requests to read a row based on a key.
- mysql_handler.read_next, the number of requests to read the next row in key order.
- mysql_handler.read_prev, the number of requests to read the previous row in key order.
- mysql_handler.read_rnd, the number of requests to read a row based on a fixed position.
- mysql_handler.read_rnd_next, the number of requests to read the next row in the data file.
- mysql_handler.rollback, the number of requests for a storage engine to perform rollback operation.
- mysql_handler.update, the number of requests to update a row in a table.
- mysql_handler.write, the number of requests to insert a row in a table.

Locks

- mysql_locks.immediate, the number of times per second the requests for table locks could be granted immediately.
- mysql_locks.waited, the number of times per second the requests for table locks had to wait.

Network

- mysql_octets.rx, the number of bytes received per second by the server.
- mysql_octets.tx, the number of bytes sent per second by the server.

Query cache

Note: These metrics are not available if your environment is set to HA.

- mysql_qcache.hits, the number of query cache hits per second.
- mysql_qcache.inserts, the number of queries added to the query cache per second.
- mysql_qcache.lowmem_prunes, the number of queries that were deleted from the query cache per second because of low memory.

- `mysql_qcache.not_cached`, the number of noncached queries per second.
- `mysql_qcache.queries_in_cache`, the number of queries registered in the query cache per second.

Threads

The **mysql_threads.created** metric is reported as a per-second rate.

- `mysql_threads.cached`, the number of threads in the thread cache.
- `mysql_threads.connected`, the number of currently open connections.
- `mysql_threads.running`, the number of threads that are not sleeping.
- `mysql_threads.created`, the number of threads created per second to handle connections.

Cluster

These metrics are collected with statement 'SHOW STATUS'. see Percona documentation for further details.

- `mysql.cluster.size`, current number of nodes in the cluster.
- `mysql.cluster.status`, `1` when the node is 'Primary', `2` if 'Non-Primary' and `3` if 'Disconnected'.
- `mysql.cluster.connected`, `1` when the node is connected to the cluster, `0` otherwise.
- `mysql.cluster.ready`, `1` when the node is ready to accept queries, `0` otherwise.
- `mysql.cluster.local_commits`, number of writesets commited on the node.
- `mysql.cluster.received_bytes`, total size in bytes of writesets received from other nodes.
- `mysql.cluster.received`, total number of writesets received from other nodes.
- `mysql.cluster.replicated_bytes` total size in bytes of writesets sent to other nodes.
- `mysql.cluster.replicated`, total number of writesets sent to other nodes.
- `mysql.cluster.local_cert_failures`, number of writesets that failed the certification test.
- `mysql.cluster.local_send_queue`, the number of writesets waiting to be sent.
- `mysql.cluster.local_recv_queue`, the number of writesets waiting to be applied.

Slow Queries

This metric is collected with statement 'SHOW STATUS where Variable_name = 'Slow_queries'.

- `mysql.slow_queries`, number of queries that have taken more than X seconds, depending of the MySQL configuration parameter 'long_query_time' (10s per default)

RabbitMQ

Service

- `rabbitmq.status`, the status of the RabbitMQ service, 1 if it is responsive, 0 otherwise.

Cluster

- `rabbitmq.connections`, Number of connections.
- `rabbitmq.consumers`, Number of consumers.
- `rabbitmq.exchanges`, Number of exchanges.
- `rabbitmq.memory`, Bytes of memory consumed by the Erlang process associated with all queues, including stack, heap and internal structures.
- `rabbitmq.messages`, Total number of messages which are ready to be consumed or not yet acknowledged.
- `rabbitmq.total_nodes`, Number of nodes in the cluster.
- `rabbitmq.running_nodes`, Number of running nodes in the cluster.
- `rabbitmq.queues`, Number of queues.

Queues

- `rabbitmq.<name_of_the_queue>.consumers`, Number of consumers.
- `rabbitmq.<name_of_the_queue>.memory`, Bytes of memory consumed by the Erlang process associated with the queue, including stack, heap and internal structures.
- `rabbitmq.<name_of_the_queue>.messages`, Number of messages which are ready to be consumed or not yet acknowledged.

HAProxy

Server

- `haproxy.status`, the status of the HAProxy service, 1 if it is responsive, 0 otherwise.
- `haproxy.connections`, number of current connections.
- `haproxy.ssl_connections`, number of current SSL connections.
- `haproxy.pipes_free`, number of free pipes.
- `haproxy.pipes_used`, number of used pipes.
- `haproxy.run_queue`, number of connections waiting in the queue.
- `haproxy.tasks`, number of tasks.
- `haproxy.uptime`, HAProxy server uptime in seconds.

Frontends

Metrics per frontend:

- `haproxy.frontend.<frontend>.bytes_in`, number of bytes received by the frontend.
- `haproxy.frontend.<frontend>.bytes_out`, number of bytes transmitted by the frontend.
- `haproxy.frontend.<frontend>.denied_requests`, number of denied requests.
- `haproxy.frontend.<frontend>.denied_responses`, number of denied responses.

- haproxy.frontend.<frontend>.error_requests, number of error requests.
- haproxy.frontend.<frontend>.response_1xx, number of HTTP responses with 1xx code.
- haproxy.frontend.<frontend>.response_2xx, number of HTTP responses with 2xx code.
- haproxy.frontend.<frontend>.response_3xx, number of HTTP responses with 3xx code.
- haproxy.frontend.<frontend>.response_4xx, number of HTTP responses with 4xx code.
- haproxy.frontend.<frontend>.response_5xx, number of HTTP responses with 5xx code.
- haproxy.frontend.<frontend>.response_other, number of HTTP responses with other code.
- haproxy.frontend.<frontend>.session_current, number of current sessions.
- haproxy.frontend.<frontend>.session_total, cumulative of total number of session.
- haproxy.frontend.bytes_in, total number of bytes received by all frontends.
- haproxy.frontend.bytes_out, total number of bytes transmitted by all frontends.
- haproxy.frontend.session_current, total number of current sessions for all frontends.

Backends

Metrics per backends:

- haproxy.backend.<backend>.bytes_in, number of bytes received by the backend.
- haproxy.backend.<backend>.bytes_out, number of bytes transmitted by the backend.
- haproxy.backend.<backend>.denied_requests, number of denied requests.
- haproxy.backend.<backend>.denied_responses, number of denied responses.
- haproxy.backend.<backend>.downtime, total downtime in second.
- haproxy.backend.<backend>.status, the backend status where values 0 and 1 represent respectively DOWN and UP.
- haproxy.backend.<backend>.error_connection, number of error connections.
- haproxy.backend.<backend>.error_responses, number of error responses.
- haproxy.backend.<backend>.queue_current, number of requests in queue.
- haproxy.backend.<backend>.redistributed, number of times a request was redispatched to another server.
- haproxy.backend.<backend>.response_1xx, number of HTTP responses with 1xx code.
- haproxy.backend.<backend>.response_2xx, number of HTTP responses with 2xx code.
- haproxy.backend.<backend>.response_3xx, number of HTTP responses with 3xx code.
- haproxy.backend.<backend>.response_4xx, number of HTTP responses with 4xx code.
- haproxy.backend.<backend>.response_5xx, number of HTTP responses with 5xx code.
- haproxy.backend.<backend>.response_other, number of HTTP responses with other code.
- haproxy.backend.<backend>.retries, number of times a connection to a server was retried.
- haproxy.backend.<backend>.servers.down, number of servers which are down.
- haproxy.backend.<backend>.servers.up, number of servers which are up.

- haproxy.backend.<backend>.session_current, number of current sessions.
- haproxy.backend.<backend>.session_total, cumulative number of sessions.
- haproxy.backend.bytes_in, total number of bytes received by all backends.
- haproxy.backend.bytes_out, total number of bytes transmitted by all backends.
- haproxy.backend.queue_current, total number of requests in queue for all backends.
- haproxy.backend.session_current, total number of current sessions for all backends.
- haproxy.backend.error_responses, total number of error responses for all backends.

Where frontend and backend are one of:

- cinder-api
- glance-api
- glance-registry-api
- heat-api
- heat-cfn-api
- heat-cloudwatch-api
- horizon-web
- keystone-public-api
- keystone-admin-api
- mysqld-tcp
- murano-api
- neutron-api
- nova-api
- nova-ec2-api
- nova-metadata-api
- nova-novncproxy-websocket
- sahara-api
- swift-api

Memcached

- memcached.status, the status of the memcached service, 1 if it is responsive, 0 otherwise.
- memcached.command.flush, cumulative number of flush reqs.
- memcached.command.get, cumulative number of retrieval reqs.
- memcached.command.set, cumulative number of storage reqs.
- memcached.command.touch, cumulative number of touch reqs.
- memcached.connections.current, number of open connections.
- memcached.items.current, current number of items stored.
- memcached.octets.rx, total number of bytes read by this server from network.

- `memcached.octets.tx`, total number of bytes sent by this server to network.
- `memcached.ops.decr_hits`, number of successful decr reqs.
- `memcached.ops.decr_misses`, number of decr reqs against missing keys.
- `memcached.ops.evictions`, number of valid items removed from cache to free memory for new items.
- `memcached.ops.hits`, number of keys that have been requested.
- `memcached.ops.incr_hits`, number of successful incr reqs.
- `memcached.ops.incr_misses`, number of successful incr reqs.
- `memcached.ops.misses`, number of items that have been requested and not found.
- `memcached.df.cache.used`, current number of bytes used to store items.
- `memcached.df.cache.free`, current number of free bytes to store items.
- `memcached.percent.hitratio`, percentage of get command hits (in cache).

See memcached documentation for further details.

OpenStack

Service checks

- `openstack.<service>.check_api`, the service's API status, 1 if it is responsive, 0 otherwise.

`<service>` is one of the following services with their respective resource checks:

- 'nova': '/'
- 'cinder': '/'
- 'cinder-v2': '/'
- 'glance': '/'
- 'heat': '/'
- 'keystone': '/'
- 'neutron': '/'
- 'ceilometer': '/v2/capabilities'
- 'swift': '/healthcheck'
- 'swift-s3': '/healthcheck'

Note: All checks are performed without authentication except for Ceilometer.

Compute

These metrics are emitted per compute node.

- `openstack.nova.instance_creation_time`, the time (in seconds) it took to launch a new instance.
- `openstack.nova.instance_state.<state>`, the number of instances which entered this state (always 1).

These metrics are retrieved from the Nova API and represent the aggregated values across all compute nodes.

- `openstack.nova.total_free_disk`, the total amount of disk space (in GB) available for new instances.
- `openstack.nova.total_used_disk`, the total amount of disk space (in GB) used by the instances.
- `openstack.nova.total_free_ram`, the total amount of memory (in MB) available for new instances.
- `openstack.nova.total_used_ram`, the total amount of memory (in MB) used by the instances.
- `openstack.nova.total_free_vcpus`, the total number of virtual CPU available for new instances.
- `openstack.nova.total_used_vcpus`, the total number of virtual CPU used by the instances.
- `openstack.nova.total_running_instances`, the total number of running instances.
- `openstack.nova.total_running_tasks`, the total number of tasks currently executed.

These metrics are retrieved from the Nova API.

- `openstack.nova.instances.<state>`, the number of instances by state.

`<state>` is one of 'active', 'deleted', 'error', 'paused', 'resumed', 'rescued', 'resized', 'shelved_offloaded' or 'suspended'.

These metrics are retrieved from the Nova database.

- `openstack.nova.services.<service>.<service_state>`, the total number of Nova services by state.

`<service>` is one of service is one of 'compute', 'conductor', 'scheduler', 'cert' or 'consoleauth'.

`<service_state>` is one of 'up', 'down' or 'disabled'.

The following status metrics are computed from other metrics including: *service checks*, *HAproxy servers* and *service states*.

Their value is one of '0' (ok), '1' (degraded), '2' (down) or '3' (unknown).

- `openstack.nova.services.<service>.status`, status of Nova services computed from `openstack.nova.services.<service>.<service_state>`.
- `openstack.nova.api.<backend>.status`, status of the API services located behind the HAProxy load-balancer, computed from `haproxy.backend.nova-*.servers.(up|down)`.
- `openstack.nova.status`, the general status of the Nova service which is computed using the previous metrics and the `openstack.nova.check_api` metric.

Identity

These metrics are retrieved from the Keystone API.

- `openstack.keystone.roles`, the total number of roles.
- `openstack.keystone.tenants.<state>`, the number of tenants by state.
- `openstack.keystone.users.<state>`, the number of users by state.

`<state>` is one of 'disabled' or 'enabled'.

The following status metrics are computed from other metrics: *service checks* and *HAproxy servers*.

Their value is one of '0' (ok), '1' (degraded), '2' (down) or '3' (unknown).

- `openstack.keystone.api.<backend>.status`, status of the API services located behind the HAProxy load-balancer, computed from `haproxy.backend.keystone-*.servers.(up|down)`.

- `openstack.keystone.status`, the general status of the Keystone service which is computed using the previous metric and the `openstack.keystone.check_api` metric.

Volume

These metrics are emitted per volume node.

- `openstack.cinder.volume_creation_time`, the time (in seconds) it took to create a new volume.

Note: When using Ceph as the backend storage for volumes, the `hostname` value is always set to `rbd`.

These metrics are retrieved from the Cinder API.

- `openstack.cinder.volumes.<state>`, the number of volumes by state.
- `openstack.cinder.snapshots.<state>`, the number of snapshots by state.
- `openstack.cinder.volumes_size.<state>`, the total size (in bytes) of volumes by state.
- `openstack.cinder.snapshots_size.<state>`, the total size (in bytes) of snapshots by state.

`<state>` is one of 'available', 'creating', 'attaching', 'in-use', 'deleting', 'backing-up', 'restoring-backup', 'error', 'error_deleting', 'error_restoring', 'error_extending'.

These metrics are retrieved from the Cinder database.

- `openstack.cinder.services.<service>.<service_state>`, the total number of Cinder services by state.

`<service>` is one of service is one of 'volume', 'backup', 'scheduler'.

`<service_state>` is one of 'up', 'down' or 'disabled'.

The following status metrics are computed from other metrics including: *service checks*, *HAproxy servers* and *service states*.

Their value is one of '0' (ok), '1' (degraded), '2' (down) or '3' (unknown).

- `openstack.cinder.services.<service>.status`, status of Cinder services computed from `openstack.cinder.services.<service>.<service_state>`.
- `openstack.cinder.api.<backend>.status`, status of the API services located behind the HAProxy load-balancer, computed from `haproxy.backend.cinder-api.servers.(up|down)`.
- `openstack.cinder.status`, the general status of the Cinder service which is computed using the previous metrics and the `openstack.cinder.check_api` metric.

Image

These metrics are retrieved from the Glance API.

- `openstack.glance.images.public.<state>`, the number of public images by state.
- `openstack.glance.images.private.<state>`, the number of private images by state.
- `openstack.glance.snapshots.public.<state>`, the number of public snapshot images by state.
- `openstack.glance.snapshots.private.<state>`, the number of private snapshot images by state.
- `openstack.glance.images_size.public.<state>`, the total size (in bytes) of public images by state.

- `openstack.glance.images_size.private.<state>`, the total size (in bytes) of private images by state.
- `openstack.glance.snapshots_size.public.<state>`, the total size (in bytes) of public snapshots by state.
- `openstack.glance.snapshots_size.private.<state>`, the total size (in bytes) of private snapshots by state.

`<state>` is one of 'queued', 'saving', 'active', 'killed', 'deleted', 'pending_delete'.

The following status metrics are computed from other metrics including: *service checks* and *HAproxy servers*.

Their value is one of '0' (ok), '1' (degraded), '2' (down) or '3' (unknown).

- `openstack.glance.api.<backend>.status`, status of the API services located behind the HAProxy load-balancer, computed from `haproxy.backend.glance-*.servers.(up|down)`.
- `openstack.glance.status`, the general status of the Glance service which is computed using the previous metric and the `openstack.glance.check_api` metric.

Network

These metrics are retrieved from the Neutron API.

- `openstack.neutron.agents`, the total number of Neutron agents.
- `openstack.neutron.networks.<state>`, the number of virtual networks by state.
- `openstack.neutron.networks`, the total number of virtual networks.
- `openstack.neutron.subnets`, the number of virtual subnets.
- `openstack.neutron.ports.<owner>.<state>`, the number of virtual ports by owner and state.
- `openstack.neutron.ports`, the total number of virtual ports.
- `openstack.neutron.routers.<state>`, the number of virtual routers by state.
- `openstack.neutron.routers`, the total number of virtual routers.
- `openstack.neutron.floatingips.free`, the number of floating IP addresses which aren't associated.
- `openstack.neutron.floatingips.associated`, the number of floating IP addresses which are associated.
- `openstack.neutron.floatingips`, the total number of floating IP addresses.

`<state>` is one of 'active', 'build', 'down' or 'error'.

`<owner>` is one of 'compute', 'dhcp', 'floatingip', 'floatingip_agent_gateway', 'router_interface', 'router_gateway', 'router_ha_interface', 'router_interface_distributed' or 'router_centralized_snat'.

These metrics are retrieved from the Neutron database.

- `openstack.neutron.agents.<agent_type>.<agent_state>`, the total number of Neutron agents by agent type and state.

`<agent_type>` is one of 'dhcp', 'l3', 'metadata' or 'openvswitch'.

`<agent_state>` is one of 'up', 'down' or 'disabled'.

The following status metrics are computed from other metrics including: *service checks*, *HAproxy servers* and *agent states*.

Their value is one of '0' (ok), '1' (degraded), '2' (down) or '3' (unknown).

- `openstack.neutron.agents.<agent_type>.status`, status of Neutron services computed from metric `openstack.neutron.agents.<agent_type>.<agent_state>`.
- `openstack.neutron.api.neutron.status`, status f the API services located behind the HAProxy load-balancer, computed from `haproxy.backend.neutron.servers.(up|down)`.
- `openstack.neutron.status`, the general status of the Neutron service which is computed using the previous metrics and the `openstack.neutron.check_api` metric.

API response times

- `openstack.<service>.http.<HTTP method>.<HTTP status>`, the time (in second) it took to serve the HTTP request.

`<service>` is one of 'cinder', 'glance', 'heat' 'keystone', 'neutron' or 'nova'.

`<HTTP method>` is the HTTP method name, eg 'GET', 'POST' and so on.

`<HTTP status>` is a 3-digit string representing the HTTP response code, eg '200', '404' and so on.

Ceph

All metrics are prefixed by `ceph.cluster-<name>` with `<name>` is *ceph* by default.

See cluster monitoring and RADOS monitoring for further details.

Cluster

- `health`, the health status of the entire cluster where values `1`, `2` , `3` represent respectively `OK`, `WARNING` and `ERROR`.
- `monitor`, number of ceph-mon processes. .
- `quorum`, number of quorum members.

Pools

- `pool.<name>.bytes_used`, amount of data stored in bytes per pool.
- `pool.<name>.max_avail`, available size in bytes per pool.
- `pool.<name>.objects`, number of objects per pool.
- `pool.<name>.read_bytes_sec`, number of bytes read by second per pool.
- `pool.<name>.write_bytes_sec`, number of bytes written by second per pool.
- `pool.<name>.op_per_sec`, number of operations per second per pool.
- `pool.<name>.size`, number of data replications per pool.
- `pool.<name>.pg_num`, number of placement groups per pool.
- `pool.total_bytes`, total number of bytes for all pools.
- `pool.total_used_bytes`, total used size in bytes by all pools.
- `pool.total_avail_bytes`, total available size in bytes for all pools.
- `pool.total_number`, total number of pools.

<name> is the name of the Ceph pool.

Placement Groups

- pg.total, total number of placement groups.
- pg.state.<state>, number of placement groups by state.
- pg.bytes_avail, available size in bytes.
- pg.bytes_total, cluster total size in bytes.
- pg.bytes_used, data stored size in bytes.
- pg.data_bytes, stored data size in bytes before it is replicated, cloned or snapshotted.

<state> is a combination separated by + of 2 or more states of this list: creating, active, clean, down, replay, splitting, scrubbing, degraded, inconsistent, peering, repair, recovering, recovery_wait, backfill, backfill-wait, backfill_toofull, incomplete, stale, remapped.

OSD Daemons

- osd.up, number of OSD daemons UP.
- osd.down, number of OSD daemons DOWN.
- osd.in, number of OSD daemons IN.
- osd.out, number of OSD daemons OUT.
- osd.<id>.used, data stored size in bytes.
- osd.<id>.total, total size in bytes.
- osd.<id>.apply_latency, apply latency in ms.
- osd.<id>.commit_latency, commit latency in ms.

<id> is the OSD numeric identifier.

OSD Performance

All the following metrics are retrieved per OSD daemon from the corresponding socket /var/run/ceph/ceph-osd.<ID>.asok by issuing the command perf dump.

Note: These metrics are not collected when a node has both the ceph-osd and controller roles.

See OSD performance counters for further details.

- osd-<id>.osd.recovery_ops, number of recovery operations in progress.
- osd-<id>.osd.op_wip, number of replication operations currently being processed (primary).
- osd-<id>.osd.op, number of client operations.
- osd-<id>.osd.op_in_bytes, number of bytes received from clients for write operations.
- osd-<id>.osd.op_out_bytes, number of bytes sent to clients for read operations.
- osd-<id>.osd.op_latency, average latency in ms for client operations (including queue time).

- `osd-<id>.osd.op_process_latency`, average latency in ms for client operations (excluding queue time).
- `osd-<id>.osd.op_r`, number of client read operations.
- `osd-<id>.osd.op_r_out_bytes`, number of bytes sent to clients for read operations.
- `osd-<id>.osd.op_r_latency`, average latency in ms for read operation (including queue time).
- `osd-<id>.osd.op_r_process_latency`, average latency in ms for read operation (excluding queue time).
- `osd-<id>.osd.op_w`, number of client write operations.
- `osd-<id>.osd.op_w_in_bytes`, number of bytes received from clients for write operations.
- `osd-<id>.osd.op_w_rlat`, average latency in ms for write operations with readable/applied.
- `osd-<id>.osd.op_w_latency`, average latency in ms for write operations (including queue time).
- `osd-<id>.osd.op_w_process_latency`, average latency in ms for write operation (excluding queue time).
- `osd-<id>.osd.op_rw`, number of client read-modify-write operations.
- `osd-<id>.osd.op_rw_in_bytes`, number of bytes per second received from clients for read-modify-write operations.
- `osd-<id>.osd.op_rw_out_bytes`, number of bytes per second sent to clients for read-modify-write operations.
- `osd-<id>.osd.op_rw_rlat`, average latency in ms for read-modify-write operations with readable/applied.
- `osd-<id>.osd.op_rw_latency`, average latency in ms for read-modify-write operations (including queue time).
- `osd-<id>.osd.op_rw_process_latency`, average latency in ms for read-modify-write operations (excluding queue time).

`<id>` is the OSD numeric identifier.

Pacemaker

Resource location

- `pacemaker.resource.<resource-name>.active`, `1` when the resource is located on the host reporting the metric, `0` otherwise.

`<resource-name>` is one of 'vip__public', 'vip__management', 'vip__public_vrouter' or 'vip__management_vrouter'.

2.5 Supported Ouptputs

The LMA collector can forward part or all of the processed Heka messages to any kind of external system, provided that the system supports a protocol-based interface such as HTTP, SMTP or AMQP.

The supported backends are described hereunder.

2.5.1 Elasticsearch

The LMA collector is able to send *Log Messages* and *Notification Messages* to Elasticsearch.

There is one index per day and per type of message:

- Index for log messages is `log-<YYYY-MM-DD>`.
- Index for notification messages is `notification-<YYYY-MM-DD>`.

2.5.2 InfluxDB

The LMA collector is able to send *Metric Messages* to InfluxDB.

Metrics are stored in individual series per metric name and hostname. The name of the serie is encoded as `<Fields[hostname]>.<Fields[name]>`.

CHAPTER 3

Indices and Tables

- genindex
- modindex
- search