Abstract

A comprehensive guide to understanding the metrics and logs collected by Metrics Store.
# Table of Contents

## CHAPTER 1. INTRODUCTION

1.1. ACCESSING KIBANA .......................................................... 3

## CHAPTER 2. ANALYZING METRICS

2.1. USING DASHBOARDS .......................................................... 4
2.2. CREATING A NEW VISUALIZATION ........................................ 5
2.3. GRAPHIC USER INTERFACE ELEMENTS ................................ 6
2.4. USING THE VISUALIZATION EDITOR ..................................... 6
   2.4.1. Submitting Search Queries .......................................... 6
   2.4.2. Selecting Metrics and Aggregations ............................... 6
2.5. METRICS SCHEMA ............................................................. 7
   2.5.1. Aggregation Metrics ................................................... 7
   2.5.2. CPU Metrics ............................................................. 8
   2.5.3. CPU Load Average Metrics ......................................... 9
   2.5.4. Disk Consumption Metrics ......................................... 10
   2.5.5. Disk Operation Metrics ............................................. 11
   2.5.6. Entropy Metrics ....................................................... 13
   2.5.7. Network Interface Metrics ......................................... 13
   2.5.8. Memory Metrics ...................................................... 18
   2.5.9. NFS Metrics ............................................................. 19
   2.5.10. PostgreSQL Metrics ................................................. 20
   2.5.11. Process Metrics ..................................................... 22
   2.5.12. StatsD Metrics ....................................................... 24
   2.5.13. Swap Metrics ......................................................... 27
   2.5.14. Virtual Machine Metrics .......................................... 28
   2.5.15. Gauge and Derive Data Source Types ............................ 29
2.6. WORKING WITH METRICS STORE INDEXES ......................... 30

## CHAPTER 3. ANALYZING LOGS

3.1. GRAPHIC USER INTERFACE ELEMENTS ............................. 31
3.2. USING THE DISCOVER PAGE .............................................. 31
   3.2.1. Setting the Time Filter ............................................. 32
   3.2.2. Searching Your Data ................................................ 32
   3.2.3. Filtering By Field .................................................... 32
   3.2.4. Visualizing Log Data ............................................... 34
   3.2.5. Customizing the Documents Table ............................... 34

## CHAPTER 4. TROUBLESHOOTING

4.1. INFORMATION IS MISSING FROM KIBANA ......................... 35
4.2. EXTRACTING ELASTICSEARCH LOGS ................................. 35
4.3. SEARCHING ELASTICSEARCH LOGS .................................. 35
4.4. SEARCHING LOG RECORD RESULTS ................................. 36
CHAPTER 1. INTRODUCTION

The Metrics Store collects logs and metrics from Red Hat Virtualization. The data is transferred from Red Hat Virtualization to OpenShift where it is stored and aggregated in Elasticsearch and saved in indexes. Elasticsearch is a distributed, RESTful search and analytics engine that lets you perform and combine many types of searches.

Use Kibana to search, view, and interact with real-time data stored in Elasticsearch indexes. Kibana is an open source analytics and visualization platform designed to work with Elasticsearch. You can easily perform advanced data analysis and visualize your data in a variety of charts and tables.

Kibana enables you to view and preempt critical problems, track system usage and resources, and even plan for future growth.

1.1. ACCESSING KIBANA

1. Access Kibana at https://kibana.FQDN.

2. Log in by entering your username and password.
CHAPTER 2. ANALYZING METRICS

Kibana offers two ways of analyzing metrics:

- Build your own visualizations such as charts, graphs, and tables.
- Load and use predefined sets of visualizations

Red Hat suggests that you start off by using the predefined visualizations. Each set is known as a dashboard. Dashboards have the advantage of enabling you to quickly access a wide range of metrics while offering the flexibility of changing them to match your individual needs.

2.1. USING DASHBOARDS

A dashboard displays a set of saved visualizations. Dashboards have the advantage of enabling you to quickly access a wide range of metrics while offering the flexibility of changing them to match your individual needs.

You can use the Dashboard tab to create your own dashboards. Alternatively, Red Hat provides the following dashboard examples, which you can import into Kibana and use as is or customize to suit your specific needs:

- System dashboard
- Hosts dashboard
- VMs dashboard

Importing Dashboard Examples

**NOTE**

The dashboard examples are only available after completing the procedure for Deploying collectd and rsyslog.

1. Copy the `/etc/ovirt-engine-metrics/dashboards-examples` directory from the Manager virtual machine to your local machine.

2. Log in to the Kibana console using the URL (`https://kibana.example.com`) that you recorded during the Metrics Store installation process. Use the default `admin` user, and the password you defined during the installation.

3. Open Kibana and click the Management tab.

4. Click the Saved Objects tab.

5. Click Import and import Searches from your local copy of `/etc/ovirt-engine-metrics/dashboards-examples`.

6. Click Import and import Visualizations.

**NOTE**

If you see an error message while importing the visualizations, check your hosts to ensure that collectd and rsyslog are running without errors.
7. Click **Import** and import **Dashboards**.

   **NOTE**
   If you are logged in as the **admin** user, you may see a message regarding missing index patterns while importing the visualizations. Select the **project.** index pattern instead.

8. Select an index pattern that begins with **project.** in the **Index Patterns** pane and click the **Refresh field list** button.

9. Repeat for each index pattern that begins with **project.**.

   The imported dashboards are now stored in the system.

### Loading Saved Dashboards

Once you have created and saved a dashboard, or imported Red Hat’s sample dashboards, you can display them in the **Dashboard** tab:

1. Click the **Dashboards** tab to display a list of saved dashboards.

2. Click a saved dashboard to load it.

### 2.2. CREATING A NEW VISUALIZATION

Use the **Visualize** page to design data visualizations based on the metrics or log data collected by Metrics Store. You can save these visualizations, use them individually, or combine visualizations into a dashboard. A visualization can be based on one of the following data source types:

- A new interactive search
- A saved search
- An existing saved visualization

Visualizations are based on Elasticsearch’s **aggregation feature**.

#### Creating a New Visualization

Kibana guides you through the creation process with the help of a visualization wizard.

1. To start the new visualization wizard, click the **Visualize** tab.

2. In step 1, **Create a new visualization table**, select the type of visualization you want to create.

3. In step 2, **Select a search source**, select whether you want to create a new search or reuse a saved search:

   - To create a new search, select **From a new search** and enter the **indexes** to use as the source. Use `project.ovirt-logs` prefix for log data or `project.ovirt-metrics` prefix for metric data.

   - To create a visualization from a saved search, select **From a saved search** and enter the name of the search.
     The **visualization editor** appears.
2.3. GRAPHIC USER INTERFACE ELEMENTS

The visualization editor consists of three main areas:

1. The toolbar
2. The aggregation builder
3. The preview pane

Visualization Editor

2.4. USING THE VISUALIZATION EDITOR

Use the visualization editor to create visualizations by:

- Submitting search queries from the toolbar
- Selecting metrics and aggregations from the aggregation builder

2.4.1. Submitting Search Queries

Use the toolbar to perform search queries based on the Lucene query parser syntax. For a detailed explanation of this syntax, see Apache Lucene - Query Parser Syntax.

2.4.2. Selecting Metrics and Aggregations

Use the aggregation builder to define which metrics to display, how to aggregate the data, and how to group the results.

The aggregation builder performs two types of aggregations, metric and bucket, which differ depending on the type of visualization you are creating:

- Bar, line, or area chart visualizations use **metrics** for the y-axis and **buckets** for the x-axis, segment bar colors, and row/column splits.
- Pie charts use **metrics** for the slice size and **buckets** to define the number of slices.
To define a visualization from the aggregation bar:

1. Select the metric aggregation for your visualization’s y-axis from the Aggregation drop-down list in the metrics section, for example, count, average, sum, min, max, or unique count. For more information about how these aggregations are calculated, see Metrics Aggregation in the Elasticsearch Reference documentation.

2. Use the buckets area to select the aggregations for the visualization’s x-axis, color slices, and row/column splits:
   a. Use the Aggregation drop-down list to define how to aggregate the bucket. Common bucket aggregations include date histogram, range, terms, filters, and significant terms. The order in which you define the buckets determines the order in which they will be executed, so the first aggregation determines the data set for any subsequent aggregations. For more information, see Aggregation Builder in the Kibana documentation.
   b. Select the metric you want to display from the Field drop-down list. For details about each of the available metrics, see Metrics Schema.
   c. Select the required interval from the Interval field.

3. Click Apply Changes.

### 2.5. METRICS SCHEMA

The following sections describe the metrics that are available from the Field menu when creating visualizations.

**NOTE**

All metric values are collected at 10 second intervals.

#### 2.5.1. Aggregation Metrics

The Aggregation metric aggregates several values into one using aggregation functions such as sum, average, min, and max. It is used to provide a combined value for average and total CPU statistics.

The following table describes the aggregation metrics reported by the Aggregation plugin.

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>collectd.type_instance</th>
<th>Description</th>
</tr>
</thead>
</table>
### Additional Values

- **collectd.plugin**: Aggregation
- **collectd.type_instance**: cpu-average / cpu-sum
- **collectd.plugin_instance**:
- **collectd.type**: percent
- **ovirt.entity**: host
- **ovirt.cluster.name.raw**: *The cluster’s name*
- **ovirt.engine_fqdn.raw**: *The Manager’s FQDN*
- **hostname**: *The host’s FQDN*
- **ipaddr4**: *IP address*
- **interval**: 10
- **collectd.dstypes**: Gauge

## 2.5.2. CPU Metrics

CPU metrics display the amount of time spent by the hosts’ CPUs, as a percentage.

The following table describes CPU metrics as reported by the **CPU** plugin.

**Table 2.1. CPU Metrics**
### Additional Values

- **collectd.plugin**: CPU
- **collectd.plugin_instance**: The CPU’s number
- **collectd.type**: percent
- **ovirt.entity**: host
- **ovirt.cluster.name.raw**: The cluster’s name
- **ovirt.engine_fqdn.raw**: The Manager’s FQDN
- **hostname**: The host’s FQDN
- **ipaddr4**: IP address
- **interval**: 10
- **collectd.dstypes**: Gauge

### 2.5.3. CPU Load Average Metrics

CPU load represents CPU contention, that is, the average number of schedulable processes at any given time. This is reported as an average value for all CPU cores on the host. Each CPU core can only execute one process at a time. Therefore, a CPU load average above 1.0 indicates that the CPUs have more work than they can perform, and the system is overloaded.

CPU load is reported over short term (last one minute), medium term (last five minutes) and long term (last fifteen minutes). While it is normal for a host’s short term load average to exceed 1.0 (for a single CPU), sustained load average above 1.0 on a host may indicate a problem.

On multi-processor systems, the load is relative to the number of processor cores available. The “100% utilization” mark is 1.00 on a single-core, 2.00 on a dual-core, 4.00 on a quad-core system.

Red Hat recommends looking at CPU load in conjunction with [CPU Metrics](#).
The following table describes the CPU load metrics reported by the Load plugin.

### Table 2.2. CPU Load Average Metrics

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.load.load.longterm</td>
<td>Average number of schedulable processes per CPU core over the last 15 minutes. A value above 1.0 indicates the system was overloaded during the last 15 minutes.</td>
</tr>
<tr>
<td>collectd.load.load.midterm</td>
<td>Average number of schedulable processes per CPU core over the last five minutes. A value above 1.0 indicates the system was overloaded during the last 5 minutes.</td>
</tr>
<tr>
<td>collectd.load.load.shortterm</td>
<td>Average number of schedulable processes per CPU core over the last one minute. A value above 1.0 indicates the system was overloaded during the last minute.</td>
</tr>
</tbody>
</table>

**Additional Values**

- `collectd.plugin`: Load
- `collectd.type`: load
- `collectd.type_instance`: None
- `collectd.plugin_instance`: None
- `ovirt.entity`: host
- `ovirt.cluster.name.raw`: The cluster’s name
- `ovirt.engine_fqdn.raw`: The Manager’s FQDN
- `hostname`: The host’s FQDN
- `ipaddr4`: IP address
- `interval`: 10
- `collectd.dstypes`: Gauge

### 2.5.4. Disk Consumption Metrics

Disk consumption (DF) metrics enable you to monitor metrics about disk consumption, such as the used, reserved, and free space for each mounted file system.

The following table describes the disk consumption metrics reported by the DF plugin.
<table>
<thead>
<tr>
<th>Metric Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.df.df_complex</td>
<td>The amount of free, used, and reserved disk space, in bytes, on this file system.</td>
</tr>
<tr>
<td>collectd.df.percent_bytes</td>
<td>The amount of free, used, and reserved disk space, as a percentage of total disk space, on this file system.</td>
</tr>
</tbody>
</table>

**Additional Values**

- **collectd.plugin**: DF
- **collectd.type_instance**: free, used, reserved
- **collectd.plugin_instance**: A mounted partition
- **ovirt.entity**: host
- **ovirt.cluster.name.raw**: The cluster’s name
- **ovirt.engine_fqdn.raw**: The Manager’s FQDN
- **hostname**: The host’s FQDN
- **ipaddr4**: IP address
- **interval**: 10
- **collectd.dstypes**: Gauge

### 2.5.5. Disk Operation Metrics

Disk operation metrics are reported per physical disk on the host, and per partition.

The following table describes the disk operation metrics reported by the Disk plugin.

**Table 2.3. Disk Operation Metrics**

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>Description</th>
<th>collectd.dstypes</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.disk.disk_ops.read</td>
<td>The number of disk read operations.</td>
<td>Derive</td>
</tr>
<tr>
<td>collectd.disk.disk_ops.write</td>
<td>The number of disk write operations.</td>
<td>Derive</td>
</tr>
<tr>
<td>Metric Name</td>
<td>Description</td>
<td>collectd.dstypes</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>collectd.disk.disk_merged.read</td>
<td>The number of disk reads that have been merged into single physical disk access operations. In other words, this metric measures the number of instances in which one physical disk access served multiple disk reads. The higher the number, the better.</td>
<td>Derive</td>
</tr>
<tr>
<td>collectd.disk.disk_merged.write</td>
<td>The number of disk writes that were merged into single physical disk access operations. In other words, this metric measures the number of instances in which one physical disk access served multiple write operations. The higher the number, the better.</td>
<td>Derive</td>
</tr>
<tr>
<td>collectd.disk.disk_time.read</td>
<td>The average amount of time it took to do a read operation, in milliseconds.</td>
<td>Derive</td>
</tr>
<tr>
<td>collectd.disk.disk_time.write</td>
<td>The average amount of time it took to do a write operation, in milliseconds.</td>
<td>Derive</td>
</tr>
<tr>
<td>collectd.disk.pending_operations</td>
<td>The queue size of pending I/O operations.</td>
<td>Gauge</td>
</tr>
<tr>
<td>collectd.disk.disk_io_time.io_time</td>
<td>The time spent doing I/Os in milliseconds. This can be used as a device load percentage, where a value of 1 second of time spent represents a 100% load.</td>
<td>Derive</td>
</tr>
<tr>
<td>collectd.disk.disk_io_time.weighted_io_time</td>
<td>A measure of both I/O completion time and the backlog that may be accumulating.</td>
<td>Derive</td>
</tr>
</tbody>
</table>

### Additional Values

- **collectd.plugin**: Disk
- **collectd.type_instance**: None
- **collectd.plugin_instance**: The disk's name
- **ovirt.entity**: host
2.5.6. Entropy Metrics

Entropy metrics display the available entropy pool size on the host. Entropy is important for generating random numbers, which are used for encryption, authorization, and similar tasks.

The following table describes the entropy metrics reported by the Entropy plugin.

Table 2.4. Entropy Metrics

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.entropy.entropy</td>
<td>The entropy pool size, in bits, on the host.</td>
</tr>
</tbody>
</table>

Additional Values

- collectd.plugin: Entropy
- collectd.type_instance: None
- collectd.plugin_instance: None
- ovirt.entity: host
- ovirt.cluster.name.raw: The cluster’s name
- ovirt.engine_fqdn.raw: The Manager’s FQDN
- hostname: The host’s FQDN
- ipaddr4: IP address
- interval: 10
- collectd.dstypes: Gauge

2.5.7. Network Interface Metrics

The following types of metrics are reported from physical and virtual network interfaces on the host:

- Bytes (octets) transmitted and received (total, or per second)
- Packets transmitted and received (total, or per second)
- Interface errors (total, or per second)
The following table describes the network interface metrics reported by the **Interface** plugin.

**Table 2.5. Network Interface Metrics**

<table>
<thead>
<tr>
<th>collectd.type</th>
<th>Metric Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| if_octets     | collectd.interface.if_octets.rx | A count of the bytes received by the interface. You can view this metric as a Rate/sec or a cumulative count (Max):  
* Rate/sec: Provides the current traffic level on the interface in bytes/sec.  
* Max: Provides the cumulative count of bytes received. Note that since this metric is a cumulative counter, its value will periodically restart from zero when the maximum possible value of the counter is exceeded. |
| if_octets     | collectd.interface.if_octets.tx | A count of the bytes transmitted by the interface. You can view this metric as a Rate/sec or a cumulative count (Max):  
* Rate/sec: Provides the current traffic level on the interface in bytes/sec.  
* Max: Provides the cumulative count of bytes transmitted. Note that since this metric is a cumulative counter, its value will periodically restart from zero when the maximum possible value of the counter is exceeded. |
<table>
<thead>
<tr>
<th>collectd.type</th>
<th>Metric Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| if_packets    | collectd.interface.if_packets.rx | A count of the packets received by the interface. You can view this metric as a Rate/sec or a cumulative count (Max):  
* Rate/sec: Provides the current traffic level on the interface in bytes/sec.  
* Max: Provides the cumulative count of packets received. Note that since this metric is a cumulative counter, its value will periodically restart from zero when the maximum possible value of the counter is exceeded. |
| if_packets    | collectd.interface.if_packets.tx | A count of the packets transmitted by the interface. You can view this metric as a Rate/sec or a cumulative count (Max):  
* Rate/sec: Provides the current traffic level on the interface in packets/sec.  
* Max: Provides the cumulative count of packets transmitted. Note that since this metric is a cumulative counter, its value will periodically restart from zero when the maximum possible value of the counter is exceeded. |
<table>
<thead>
<tr>
<th>collectd.type</th>
<th>Metric Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>if_errors</td>
<td>collectd.interface.if_errors.rx</td>
<td>A count of errors received on the interface. You can view this metric as a Rate/sec or a cumulative count (Max).  * Rate/sec rollup provides the current rate of errors received on the interface in errors/sec.  * Max rollup provides the total number of errors received since the beginning. Note that since this is a cumulative counter, its value will periodically restart from zero when the maximum possible value of the counter is exceeded.</td>
</tr>
<tr>
<td>if_errors</td>
<td>collectd.interface.if_errors.tx</td>
<td>A count of errors transmitted on the interface. You can view this metric as a Rate/sec or a cumulative count (Max).  * Rate/sec rollup provides the current rate of errors transmitted on the interface in errors/sec.  * Max rollup provides the total number of errors transmitted since the beginning. Note that since this is a cumulative counter, its value will periodically restart from zero when the maximum possible value of the counter is exceeded.</td>
</tr>
</tbody>
</table>
### collectd.type

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>if_dropped</td>
<td>A count of dropped packets received on the interface. You can view this metric as a Rate/sec or a cumulative count (Max). * Rate/sec rollup provides the current rate of dropped packets received on the interface in packets/sec. * Max rollup provides the total number of dropped packets received since the beginning. Note that since this is a cumulative counter, its value will periodically restart from zero when the maximum possible value of the counter is exceeded.</td>
</tr>
<tr>
<td>if_dropped</td>
<td>A count of dropped packets transmitted on the interface. You can view this metric as a Rate/sec or a cumulative count (Max). * Rate/sec rollup provides the current rate of dropped packets transmitted on the interface in packets/sec. * Max rollup provides the total number of dropped packets transmitted since the beginning. Note that since this is a cumulative counter, its value will periodically restart from zero when the maximum possible value of the counter is exceeded.</td>
</tr>
</tbody>
</table>

### Additional Values

- **collectd.plugin**: Interface
- **collectd.type_instance**: None
- **collectd.plugin_instance**: The network’s name
- **ovirt.entity**: host
- **ovirt.cluster.name.raw**: The cluster’s name
2.5.8. Memory Metrics

Metrics collected about memory usage.

The following table describes the memory usage metrics reported by the Memory plugin.

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>collectd.type</th>
<th>collectd.type_instance</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.memory.memory</td>
<td>memory</td>
<td>used</td>
<td>The total amount of memory used.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>free</td>
<td>The total amount of unused memory.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cached</td>
<td>The amount of memory used for caching disk data for reads, memory-mapped files, or tmpfs data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>buffered</td>
<td>The amount of memory used for buffering, mostly for I/O operations.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>slab_recl</td>
<td>The amount of reclaimable memory used for slab kernel allocations.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>slab_unrecl</td>
<td>Amount of unreclaimable memory used for slab kernel allocations.</td>
</tr>
<tr>
<td>collectd.memory.percent</td>
<td>percent</td>
<td>used</td>
<td>The total amount of memory used, as a percentage.</td>
</tr>
<tr>
<td>Metric Name</td>
<td>collectd.type</td>
<td>collectd.type_instance</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------</td>
<td>------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>free</td>
<td></td>
<td></td>
<td>The total amount of unused memory, as a percentage.</td>
</tr>
<tr>
<td>cached</td>
<td></td>
<td></td>
<td>The amount of memory used for caching disk data for reads, memory-mapped files, or tmpfs data, as a percentage.</td>
</tr>
<tr>
<td>buffered</td>
<td></td>
<td></td>
<td>The amount of memory used for buffering I/O operations, as a percentage.</td>
</tr>
<tr>
<td>slab_recl</td>
<td></td>
<td></td>
<td>The amount of reclaimable memory used for slab kernel allocations, as a percentage.</td>
</tr>
<tr>
<td>slab_unrecl</td>
<td></td>
<td></td>
<td>The amount of unreclaimable memory used for slab kernel allocations, as a percentage.</td>
</tr>
</tbody>
</table>

**Additional Values**

- **collectd.plugin**: Memory
- **collectd.plugin_instance**: None
- **ovirt.entity**: Host
- **ovirt.cluster.name.raw**: *The cluster’s name*
- **ovirt.engine_fqdn.raw**: *The Manager’s FQDN*
- **hostname**: *The host’s FQDN*
- **ipaddr4**: *IP address*
- **interval**: 10
- **collectd.dstypes**: Gauge

### 2.5.9. NFS Metrics

NFS metrics enable you to analyze the use of NFS procedures.
The following table describes the NFS metrics reported by the **NFS** plugin.

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>collectd.type_instance</th>
<th>Description</th>
</tr>
</thead>
</table>

**Additional Values**

- **collectd.plugin**: NFS
- **collectd.plugin_instance**: *File system + server or client (for example: v3client)*
- **collectd.type**: nfs_procedure
- **ovirt.entity**: host
- **ovirt.cluster.name.raw**: *The cluster’s name*
- **ovirt.engine_fqdn.raw**: *The Manager’s FQDN*
- **hostname**: *The host’s FQDN*
- **ipaddr4**: *IP address*
- **interval**: 10
- **collectd.dstypes**: Derive

### 2.5.10. PostgreSQL Metrics

PostgreSQL data collected by executing SQL statements on a PostgreSQL database.

The following table describes the PostgreSQL metrics reported by the **PostgreSQL** plugin.

**Table 2.7. PostgreSQL Metrics**
<table>
<thead>
<tr>
<th>Metric Name</th>
<th><code>collectd.type_instance</code></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.postgresql.pg_numbakeends</td>
<td>N/A</td>
<td>How many server processes this database is using.</td>
</tr>
<tr>
<td>collectd.postgresql.pg_n_tup_g</td>
<td>live</td>
<td>The number of live rows in the database.</td>
</tr>
<tr>
<td></td>
<td>dead</td>
<td>The number of dead rows in the database. Rows that are deleted or obsoleted by an update are not physically removed from their table; they remain present as dead rows until a VACUUM is performed.</td>
</tr>
<tr>
<td>collectd.postgresql.pg_n_tup_c</td>
<td>del</td>
<td>The number of delete operations.</td>
</tr>
<tr>
<td></td>
<td>upd</td>
<td>The number of update operations.</td>
</tr>
<tr>
<td></td>
<td>hot_upd</td>
<td>The number of update operations that have been performed without requiring an index update.</td>
</tr>
<tr>
<td></td>
<td>ins</td>
<td>The number of insert operations.</td>
</tr>
<tr>
<td>collectd.postgresql.pg_xact</td>
<td>num_deadlocks</td>
<td>The number of deadlocks that have been detected by the database. Deadlocks are caused by two or more competing actions that are unable to finish because each is waiting for the other’s resources to be unlocked.</td>
</tr>
<tr>
<td>collectd.postgresql.pg_db_size</td>
<td>N/A</td>
<td>The size of the database on disk, in bytes.</td>
</tr>
<tr>
<td>collectd.postgresql.pg_blks</td>
<td>heap_read</td>
<td>How many disk blocks have been read.</td>
</tr>
<tr>
<td></td>
<td>heap_hit</td>
<td>How many read operations were served from the buffer in memory, so that a disk read was not necessary. This only includes hits in the PostgreSQL buffer cache, not the operating system’s file system cache.</td>
</tr>
<tr>
<td></td>
<td>idx_read</td>
<td>How many disk blocks have been read by index access operations.</td>
</tr>
</tbody>
</table>
### Additional Values

- **collectd.plugin**: Postgresql
- **collectd.plugin_instance**: Database’s Name
- **ovirt.entity**: engine
- **ovirt.cluster.name.raw**: The cluster’s name
- **ovirt.engine_fqdn.raw**: The Manager’s FQDN
- **hostname**: The host’s FQDN
- **ipaddr4**: IP address
- **interval**: 10
- **collectd.dstypes**: Gauge

#### 2.5.11. Process Metrics

The following table describes the process metrics reported by the **Processes** plugin.

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>collectd.type</th>
<th>Description</th>
<th>collectd.dstypes</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.processes.ps_state</td>
<td>ps_state</td>
<td>The number of processes in each state.</td>
<td>Gauge</td>
</tr>
<tr>
<td>collectd.processes.ps_disk_ops.read</td>
<td>ps_disk_ops</td>
<td>The process’s I/O read operations.</td>
<td>Derive</td>
</tr>
<tr>
<td>collectd.processes.ps_disk_ops.write</td>
<td>ps_disk_ops</td>
<td>The process’s I/O write operations.</td>
<td>Derive</td>
</tr>
<tr>
<td>Metric Name</td>
<td>collectd.type</td>
<td>Description</td>
<td>collectd.dstypes</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>collectd.processes.ps_vm</td>
<td>ps_vm</td>
<td>The total amount of memory including swap.</td>
<td>Gauge</td>
</tr>
<tr>
<td>collectd.processes.ps_rss</td>
<td>ps_rss</td>
<td>The amount of physical memory assigned to the process.</td>
<td>Gauge</td>
</tr>
<tr>
<td>collectd.processes.ps_data</td>
<td>ps_data</td>
<td></td>
<td>Gauge</td>
</tr>
<tr>
<td>collectd.processes.ps_code</td>
<td>ps_code</td>
<td></td>
<td>Gauge</td>
</tr>
<tr>
<td>collectd.processes.ps_stacksize</td>
<td>ps_stacksize</td>
<td></td>
<td>Gauge</td>
</tr>
<tr>
<td>collectd.processes.ps_cputime.syst</td>
<td>ps_cputime</td>
<td>The amount of time spent by the matching processes in kernel mode. The values are scaled to microseconds per second to match collectd’s numbers.</td>
<td>Derive</td>
</tr>
<tr>
<td>collectd.processes.ps_cputime.user</td>
<td>ps_cputime</td>
<td>The amount of time spent by the matching processes in user mode. The values are scaled to microseconds per second.</td>
<td>Derive</td>
</tr>
<tr>
<td>collectd.processes.ps_count.processes</td>
<td>ps_count</td>
<td>The number of processes for the defined process.</td>
<td>Gauge</td>
</tr>
<tr>
<td>collectd.processes.ps_count.threads</td>
<td>ps_count</td>
<td>The number of threads for the defined process.</td>
<td>Gauge</td>
</tr>
<tr>
<td>collectd.processes.ps_pagefaults.majfltadd</td>
<td>ps_pagefaults</td>
<td>The number of major page faults caused by the process.</td>
<td>Derive</td>
</tr>
<tr>
<td>collectd.processes.ps_pagefaults.minflt</td>
<td>ps_pagefaults</td>
<td>The number of major page faults caused by the process.</td>
<td>Derive</td>
</tr>
<tr>
<td>Metric Name</td>
<td>collectd.type</td>
<td>Description</td>
<td>collectd.dstypes</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>----------------</td>
<td>--------------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>collectd.processes.ps_disk_octets.write</td>
<td>ps_disk_octets</td>
<td>The process’s I/O write operations in transferred bytes.</td>
<td>Derive</td>
</tr>
<tr>
<td>collectd.processes.ps_disk_octets.read</td>
<td>ps_disk_octets</td>
<td>The process’s I/O read operations in transferred bytes.</td>
<td>Derive</td>
</tr>
<tr>
<td>collectd.processes.fork_rate</td>
<td>fork_rate</td>
<td>The system’s fork rate.</td>
<td>Derive</td>
</tr>
</tbody>
</table>

**Additional Values**

- **collectd.plugin**: Processes
- **collectd.type_instance**: N/A (except for collectd.processes.ps_state=running/ zombies/ stopped/ paging/ blocked/ sleeping)
- **ovirt.entity**: host
- **ovirt.cluster.name.raw**: The cluster’s name
- **ovirt.engine_fqdn.raw**: The Manager’s FQDN
- **hostname**: The host’s FQDN
- **ipaddr4**: IP address
- **interval**: 10

### 2.5.12. StatsD Metrics

The following table describes the StatsD metrics reported by the StatsD plugin.

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>collectd.type</th>
<th>collectd.type_instance</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.statsd.host_storage</td>
<td>host_storage</td>
<td>storage_uuid</td>
<td>The latency for writing to the storage domain.</td>
</tr>
<tr>
<td>collectd.statsd.vm_balloon_cur</td>
<td>vm_balloon_cur</td>
<td>N/A</td>
<td>The current amount of memory available to the guest virtual machine (in KB).</td>
</tr>
<tr>
<td>Metric Name</td>
<td>collectd.type</td>
<td>collectd.type_instance</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------</td>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>collectd.statsd.vm_balloon_max</td>
<td>vm_balloon_max</td>
<td>N/A</td>
<td>The maximum amount of memory available to the guest virtual machine (in KB).</td>
</tr>
<tr>
<td>collectd.statsd.vm_balloon_min</td>
<td>vm_balloon_min</td>
<td>N/A</td>
<td>The minimum amount of memory guaranteed to the guest virtual machine (in KB).</td>
</tr>
<tr>
<td>collectd.statsd.vm_balloon_target</td>
<td>vm_balloon_target</td>
<td>N/A</td>
<td>The amount of memory requested (in KB).</td>
</tr>
<tr>
<td>collectd.statsd.vm_cpu_sys</td>
<td>vm_cpu_sys</td>
<td>N/A</td>
<td>Ratio of non-guest virtual machine CPU time to total CPU time spent by QEMU.</td>
</tr>
<tr>
<td>collectd.statsd.vm_cpu_usage</td>
<td>vm_cpu_usage</td>
<td>N/A</td>
<td>Total CPU usage since VM start in (ns).</td>
</tr>
<tr>
<td>collectd.statsd.vm_cpu_user</td>
<td>vm_cpu_user</td>
<td>N/A</td>
<td>Ratio of guest virtual machine CPU time to total CPU time spent by QEMU.</td>
</tr>
<tr>
<td>collectd.statsd.vm_disk_apparent_size</td>
<td>vm_disk_apparent_size</td>
<td>disk name</td>
<td>The size of the disk (in bytes).</td>
</tr>
<tr>
<td>collectd.statsd.vm_disk_flush_latency</td>
<td>vm_disk_flush_latency</td>
<td>disk name</td>
<td>The virtual disk’s flush latency (in seconds).</td>
</tr>
<tr>
<td>collectd.statsd.vm_disk_read_bytes</td>
<td>vm_disk_read_bytes</td>
<td>disk name</td>
<td>The read rate from disk (in bytes per second).</td>
</tr>
<tr>
<td>collectd.statsd.vm_disk_read_latency</td>
<td>vm_disk_read_latency</td>
<td>disk name</td>
<td>The virtual disk’s read latency (in seconds).</td>
</tr>
<tr>
<td>collectd.statsd.vm_disk_read_ops</td>
<td>vm_disk_read_ops</td>
<td>disk name</td>
<td>The number of read operations since the virtual machine was started.</td>
</tr>
<tr>
<td>collectd.statsd.vm_disk_read_rate</td>
<td>vm_disk_read_rate</td>
<td>disk name</td>
<td>The virtual machine’s read activity rate (in bytes per second).</td>
</tr>
<tr>
<td>Metric Name</td>
<td>collectd.type</td>
<td>collectd.type_instance</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>--------------------</td>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>collectd.statsd.vm_disk_true_size</td>
<td>vm_disk_true_size</td>
<td>disk name</td>
<td>The amount of underlying storage allocated (in bytes).</td>
</tr>
<tr>
<td>collectd.statsd.vm_disk_write_latency</td>
<td>vm_disk_write_latency</td>
<td>disk name</td>
<td>The virtual disk’s write latency (in seconds).</td>
</tr>
<tr>
<td>collectd.statsd.vm_disk_write_ops</td>
<td>vm_disk_write_ops</td>
<td>disk name</td>
<td>The number of write operations since the virtual machine was started.</td>
</tr>
<tr>
<td>collectd.statsd.vm_disk_write_rate</td>
<td>vm_disk_write_rate</td>
<td>disk name</td>
<td>The virtual machine’s write activity rate (in bytes per second).</td>
</tr>
<tr>
<td>collectd.statsd.vm_nic_rx_bytes</td>
<td>vm_nic_rx_bytes</td>
<td>network name</td>
<td>The total number of incoming bytes.</td>
</tr>
<tr>
<td>collectd.statsd.vm_nic_rx_dropped</td>
<td>vm_nic_rx_dropped</td>
<td>network name</td>
<td>The number of incoming packets that have been dropped.</td>
</tr>
<tr>
<td>collectd.statsd.vm_nic_rx_errors</td>
<td>vm_nic_rx_errors</td>
<td>network name</td>
<td>The number of incoming packets that contained errors.</td>
</tr>
<tr>
<td>collectd.statsd.vm_nic_speed</td>
<td>vm_nic_speed</td>
<td>network name</td>
<td>The interface speed (in Mbps).</td>
</tr>
<tr>
<td>collectd.statsd.vm_nic_tx_bytes</td>
<td>vm_nic_tx_bytes</td>
<td>network name</td>
<td>The total number of outgoing bytes.</td>
</tr>
<tr>
<td>collectd.statsd.vm_nic_tx_dropped</td>
<td>vm_nic_tx_dropped</td>
<td>network name</td>
<td>The number of outgoing packets that were dropped.</td>
</tr>
<tr>
<td>collectd.statsd.vm_nic_errs</td>
<td>vm_nic_tx_errors</td>
<td>network name</td>
<td>The number of outgoing packets that contained errors.</td>
</tr>
</tbody>
</table>

**Additional Values**

- **collectd.plugin**: StatsD
- **collectd.plugin_instance**: The virtual machine’s name (except for `collectd.statsd.host_storage=N/A`)
2.5.13. Swap Metrics

Swap metrics enable you to view the amount of memory currently written onto the hard disk, in bytes, according to available, used, and cached swap space.

The following table describes the Swap metrics reported by the **Swap** plugin.

**Table 2.8. Swap Metrics**

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>collectd.type</th>
<th>collectd.type_instance</th>
<th>collectd.dstypes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.swap.swap</td>
<td>swap</td>
<td>used / free / cached</td>
<td>Gauge</td>
<td>The used, available, and cached swap space (in bytes).</td>
</tr>
<tr>
<td>collectd.swap.swap_io</td>
<td>swap_io</td>
<td>in / out</td>
<td>Derive</td>
<td>The number of swap pages written and read per second.</td>
</tr>
<tr>
<td>collectd.swap.percent</td>
<td>percent</td>
<td>used / free / cached</td>
<td>Gauge</td>
<td>The percentage of used, available, and cached swap space.</td>
</tr>
</tbody>
</table>

**Additional Fields**

- **collectd.plugin**: Swap
- **collectd.plugin_instance**: None
- **ovirt.entity**: host or Manager
- **ovirt.cluster.name.raw**: The cluster’s name
- **ovirt.engine_fqdn.raw**: The Manager’s FQDN
- **hostname**: The host’s FQDN

**CHAPTER 2. ANALYZING METRICS**

- **ovirt.entity**: vm (except for collectd.statsd.host_storage=host)
- **ovirt.cluster.name.raw**: The cluster’s name
- **ovirt.engine_fqdn.raw**: The Manager’s FQDN
- **hostname**: The host’s FQDN
- **ipaddr4**: IP address
- **interval**: 10
- **collectd.dstypes**: Gauge
2.5.14. Virtual Machine Metrics

The following table describes the virtual machine metrics reported by the Virt plugin.

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>collectd.type</th>
<th>collectd.type_instance</th>
<th>collectd.dstypes</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.virt.ps_cputime.syst</td>
<td>ps_cputime.syst</td>
<td>N/A</td>
<td>Derive</td>
</tr>
<tr>
<td>collectd.virt.percent</td>
<td>percent</td>
<td>virt_cpu_total</td>
<td>Gauge</td>
</tr>
<tr>
<td>collectd.virt.ps_cputime.user</td>
<td>ps_cputime.user</td>
<td>N/A</td>
<td>Derive</td>
</tr>
<tr>
<td>collectd.virt.virt_cpu_total</td>
<td>virt_cpu_total</td>
<td>CPU number</td>
<td>Derive</td>
</tr>
<tr>
<td>collectd.virt.vcpu</td>
<td>virt_vcpu</td>
<td>CPU number</td>
<td>Derive</td>
</tr>
<tr>
<td>collectd.virt.disk_octets.read</td>
<td>disk_octets.read</td>
<td>disk name</td>
<td>Gauge</td>
</tr>
<tr>
<td>collectd.virt.disk_ops.read</td>
<td>disk_ops.read</td>
<td>disk name</td>
<td>Gauge</td>
</tr>
<tr>
<td>collectd.virt.disk_octets.write</td>
<td>disk_octets.write</td>
<td>disk name</td>
<td>Gauge</td>
</tr>
<tr>
<td>collectd.virt.disk_ops.write</td>
<td>disk_ops.write</td>
<td>disk name</td>
<td>Gauge</td>
</tr>
<tr>
<td>collectd.virt.if_octets.rx</td>
<td>if_octets.rx</td>
<td>network name</td>
<td>Derive</td>
</tr>
<tr>
<td>collectd.virt.if_dropped.rx</td>
<td>if_dropped.rx</td>
<td>network name</td>
<td>Derive</td>
</tr>
<tr>
<td>collectd.virt.if_errors.rx</td>
<td>if_errors.rx</td>
<td>network name</td>
<td>Derive</td>
</tr>
<tr>
<td>collectd.virt.if_octets.tx</td>
<td>if_octets.tx</td>
<td>network name</td>
<td>Derive</td>
</tr>
<tr>
<td>collectd.virt.if_dropped.tx</td>
<td>if_dropped.tx</td>
<td>network name</td>
<td>Derive</td>
</tr>
<tr>
<td>collectd.virt.if_errors.tx</td>
<td>if_errors.tx</td>
<td>network name</td>
<td>Derive</td>
</tr>
</tbody>
</table>
### Metric Name | collectd.type | collectd.type_instance | collectd.dtypes
---|---|---|---
collectd.virt.if_packets.rx | if_packets.rx | network name | Derive
collectd.virt.if_packets.tx | if_packets.tx | network name | Derive
collectd.virt.memory | memory | rss / total /actual_balloon /available / unused / usable / last_update /major_fault / minor_fault / swap_in / swap_out | Gauge
collectd.virt.total_requests | total_requests | flush-DISK | Derive
collectd.virt.total_time_in_ms | total_time_in_ms | flush-DISK | Derive
collectd.virt.total_time_in_ms | total_time_in_ms | flush-DISK | Derive

### Additional Values
- **collectd.plugin**: virt
- **collectd.plugin_instance**: The virtual machine's name
- **ovirt.entity**: vm
- **ovirt.cluster.name.raw**: The cluster's name
- **ovirt.engine_fqdn.raw**: The Manager’s FQDN
- **hostname**: The host’s FQDN
- **ipaddr4**: IP address
- **interval**: 10

### 2.5.15. Gauge and Derive Data Source Types

Each metric includes a `collectd.dstypes` value that defines the data source’s type:

- **Gauge**: A gauge value is simply stored as-is and is used for values that may increase or decrease, such as the amount of memory used.

- **Derive**: These data sources assume that the change of the value is interesting, i.e., the derivative. Such data sources are very common for events that can be counted, for example the number of disk read operations. The total number of disk read operations is not interesting, but rather the change since the value was last read. The value is therefore converted to a rate using the following formula:
rate = value(new)-value(old)\
     time(new)-time(old)

**NOTE**

If value(new) is less than value (old), the resulting rate will be negative. If the minimum value to zero, such data points will be discarded.

### 2.6. WORKING WITH METRICS STORE INDEXES

Metrics Store creates the following two indexes per day:

- project.ovirt-metrics-<ovirt-env-name>.uuid.yyyy.mm.dd
- project.ovirt-logs-<ovirt-env-name>.uuid.yyyy.mm.dd

When using the **Discover** page, select the index named project.ovirt-logs-<ovirt-env-name>.uuid.

In the **Visualization** page select project.ovirt-metrics-<ovirt-env-name>.uuid for metrics data or project.ovirt-logs-<ovirt-env-name>.uuid for log data.
CHAPTER 3. ANALYZING LOGS

Use the Discover page to interactively explore the data collected from Red Hat Virtualization. Each set of results that is collected is referred to as a document. Documents are collected from the following log files:

- **engine.log** contains all Red Hat Virtualization Manager UI crashes, Active Directory lookups, database issues, and other events.
- **vdsm.log** is the log file for VDSM, the Manager’s agent on the virtualization host(s), and contains host-related events.

3.1. GRAPHIC USER INTERFACE ELEMENTS

The distribution of documents over time is displayed in a histogram at the top of the page. By default the information is grouped into 30 second intervals, but this can be changed by clicking the time drop-down list that appears above the histogram.

Figure 3.1. Histogram

The bottom of the page displays the documents in a table, sorted according to time.

Figure 3.2. Documents Table

<table>
<thead>
<tr>
<th>Time</th>
<th>Source</th>
</tr>
</thead>
</table>
| January 11th, 2016, 16:09:31.000 | level: INFO message: START recordStats(overrideName) from:/tmp/1635162053735F, flow_id:1056331600, task_id:7c2e7774-ec0f-4005-9d25-7b377b7a788d
| January 11th, 2016, 16:09:31.000 | level: INFO message: FINISH recordStats return:1j55b000-34-143-5396-1056331600

3.2. USING THE DISCOVER PAGE

From the Discover page you can:

- Set the time filter
- Submit search queries
- Filter the search results
- View the results in the Visualization page
- Customize the Documents table
3.2.1. Setting the Time Filter

By default, data from the last 15 minutes is displayed. There are several ways to change the time filter:

- Click the time filter and either select a predefined time filter or define a time range from the Relative or Absolute menus.
- Define a filter directly from the histogram by clicking a bar or click and drag over several bars. For more information, see Setting the Time Filter in the Kibana documentation.

3.2.2. Searching Your Data

Use the search field at the top of the page to filter the results according to a specific value. For example, to display results containing the word "login", type *login* in the search field. For more information about searches, see Searching Your Data in the Kibana documentation.

3.2.3. Filtering By Field

Filtering log data by field enables you to focus on the specific error that interests you.

To filter the log data by field:

1. Click the name of the field you want to filter on from the Available Fields pane. This displays the top five values for that field. To the right of each value, there are two magnifying glass buttons, one for adding a regular (positive) filter, and one for adding a negative filter.

### Table 3.1. Available Fields

<table>
<thead>
<tr>
<th>Available Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_id</td>
<td>The unique ID of the document.</td>
</tr>
<tr>
<td>_index</td>
<td>The ID of the index to which the document belongs. The index with the project.ovirt-logs prefix is the only relevant index in the Discover page.</td>
</tr>
<tr>
<td>hostname</td>
<td>For the engine.log this is the hostname of the Manager. For the vdsm.log this is hostname of the host.</td>
</tr>
<tr>
<td>level</td>
<td>The log record’s severity: TRACE, DEBUG, INFO, WARN, ERROR, FATAL.</td>
</tr>
<tr>
<td>message</td>
<td>The body of the document’s message.</td>
</tr>
<tr>
<td>ovirt.class</td>
<td>The name of a Java class that produced this log.</td>
</tr>
<tr>
<td>ovirt.correlationid</td>
<td>For the engine.log only. This ID is used to correlate the multiple parts of a single task performed by the Manager.</td>
</tr>
<tr>
<td>Available Field</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ovirt.thread</td>
<td>The name of a Java thread inside which the log record was produced.</td>
</tr>
<tr>
<td>tag</td>
<td>Predefined sets of metadata that can be used to filter the data.</td>
</tr>
<tr>
<td>@timestamp</td>
<td>The time that the record was issued.</td>
</tr>
<tr>
<td>_score</td>
<td>N/A</td>
</tr>
<tr>
<td>_type</td>
<td>N/A</td>
</tr>
<tr>
<td>ipaddr4</td>
<td>The machine’s IP address.</td>
</tr>
<tr>
<td>ovirt.cluster_name</td>
<td>For the vdsm.log only. The name of the cluster to which the host belongs.</td>
</tr>
<tr>
<td>ovirt.engine_fqdn</td>
<td>The Manager’s FQDN.</td>
</tr>
<tr>
<td>ovirt.module_lineno</td>
<td>The file and line number within the file that ran the command defined in ovirt.class.</td>
</tr>
<tr>
<td>pipeline_metadata.collector.inputname</td>
<td>N/A</td>
</tr>
<tr>
<td>pipeline_metadata.collector.ipaddr4</td>
<td>N/A</td>
</tr>
<tr>
<td>pipeline_metadata.collector.ipaddr6</td>
<td>N/A</td>
</tr>
<tr>
<td>pipeline_metadata.collector.name</td>
<td>N/A</td>
</tr>
<tr>
<td>pipeline_metadata.collector.received_at</td>
<td>N/A</td>
</tr>
<tr>
<td>pipeline_metadata.collector.version</td>
<td>N/A</td>
</tr>
<tr>
<td>service</td>
<td>The log file from which the document was extracted.</td>
</tr>
</tbody>
</table>

2. To add a positive filter, click the **Positive Filter** button 🔄. This filters out results that do not contain that value in the field.

3. To add a negative filter, click the **Negative Filter** button 🔄. This excludes results that contain that value in the field.

For more information about working with filters, see Working with Filters in the Kibana documentation.
3.2.4. Visualizing Log Data

You can visualize and aggregate log data in the Visualization page by selecting a specific field from within the Discover page.

To visualize log data:

1. Click the name of the field you want to visualize from the Available Fields pane (see Section 3.2.3, "Filtering By Field").
2. Click the Visualize button that appears beneath the top five values. You are transferred to the Visualize page where you can view the filtered value in a graphical format.

3.2.5. Customizing the Documents Table

You can customize the way that the data is displayed in the Documents table by adding fields to the table as columns and changing the display order.

To add fields to the table as columns:

1. Hover over the name of the field you want to add to the documents table from the Available Fields pane (see Section 3.2.3, "Filtering By Field").
2. Click add. The field is added to the table.
3. Optionally click the Sort by arrow that appears next to the column title to sort the results by that column.
CHAPTER 4. TROUBLESHOOTING

The following sections explain how to resolve issues that may occur in Metrics Store.

4.1. INFORMATION IS MISSING FROM KIBANA

If Kibana is not displaying metric or log information as expected, use `journalctl` to investigate the collectd and rsyslog log files as follows:

- If only metrics information is missing, check the collectd log files.
- If only log information is missing, check the rsyslog log files.
- If both metrics and logs information are missing, check both log files.
  1. To investigate collectd log files, log in as `root` and run the following command:
     ```bash
     # journalctl -u collectd
     ```
  2. To investigate rsyslog log files, run the following command:
     ```bash
     # journalctl -u rsyslog
     ```

To learn about additional `journalctl` options, see `journalctl` in Linux man pages.

4.2. EXTRACTING ELASTICSEARCH LOGS

To extract Elasticsearch logs

1. Log in to the Metrics Store virtual machine as `root`, and run the following command. Where `openshift-logging` is the namespace for the Elasticsearch pod:
   ```bash
   for espod in $(oc -n openshift-logging get pods -l component=es -o jsonpath='{.items[*].metadata.name}') ; do
     oc -n openshift-logging exec -c elasticsearch $espod -- logs > $espod.log 2>&1
   done
   ```
   Optionally, you can use the the OpenShift logging-dump tool located at logging dump tool script

4.3. SEARCHING ELASTICSEARCH LOGS

1. To search for all RHV/oVirt logs, ordered by timestamp from newest to oldest. Log in to the Metrics Store virtual machine as `root`, and run the following command, where `openshift-logging` is the namespace for the Elasticsearch pod:
   ```bash
   oc -n openshift-logging exec -c elasticsearch $(oc -n openshift-logging get pods -l component=es -o jsonpath='{.items[0].metadata.name}') -- es_util --query=project.ovirt-logs/*/_search?sort=@timestamp:desc&pretty | more
   ```

2. The output is presented in the human readable JSON format.
4.4. SEARCHING LOG RECORD RESULTS

To search the log report results, you can use the `get last rec from host` search tool.

The search tool shows how long it has been since the Metrics Store virtual machine received a record from each host that it knows about during a given time interval (last 3 hours by default).

For each host that the Metrics Store virtual machine receives logs from over the defined duration (default 3h) duration, it prints out "green", "yellow", or "red", depending on whether the Metrics Store virtual machine received a record from that host recently or not.

This is followed by the number of seconds it has been since the last record was received from that host, and the number of records received.

1. Log in to the Metrics Store virtual machine as `root`, and clone the script repository:

   ```bash
   # git clone https://github.com/jcantrill/cluster-logging-tools -b release-3.11
   ```

2. Run the script:

   ```bash
   # cd cluster-logging-tools/scripts
   [OLDEST=3h] ./get-last-rec-from-host
   ```

3. To change the interval, change the `OLDEST` value to a longer or shorter interval in hours. For example, to go back 1 day (24 hours), use:

   ```bash
   OLDEST=24h ./get-last-rec-from-host
   ```

**NOTE**

Some hosts may not be listed, if no records were received from that host during the given time interval.