# The OpenStack Telemetry plugin for Fuel Documentation

*Release 1.0.1*

Mirantis Inc.

Jan 16, 2017

# CONTENTS

CHAPTER

ONE

# OVERVIEW

## 1.1 Introduction

The OpenStack Telemetry plugin collects metrics about OpenStack resources and provides this data through the Ceilometer API. By default, the plugin supports only sample and statistics API. However, you can enable full Ceilometer API support. The OpenStack Telemetry plugin implements all the Ceilometer functionality except complex queries with InfluxDB and Elasticsearch as back ends for samples and events.

The OpenStack Telemetry plugin uses the following Ceilometer components:

- Polling agents (both central and computes)
- Notification agent
- Ceilometer API agent

Ceilometer collector is not used. Instead, the Telemetry plugin uses its own tools to collect telemetry data from the Ceilometer agents.

The Telemetry plugin provides a better functionality if deployed together with the Kafka plugin. In this case, the Telemetry plugin configures Kafka as a message bus for the Ceilometer agents and OpenStack services still send notifications to RabbitMQ. To process this correctly, the Ceilometer notification agent listens to Kafka and RabbitMQ simultaneously. However, the Telemetry plugin works without Kafka as well, but there are some scalability limitations. For more information, see *Limitations*.

Depending on the message broker installed, Hindsight or Heka are used as collectors:

- Hindsight – fetches Ceilometer samples from Kafka and is installed on the same nodes as Kafka.
- Heka – works with RabbitMQ and is installed on controller nodes under Pacemaker.

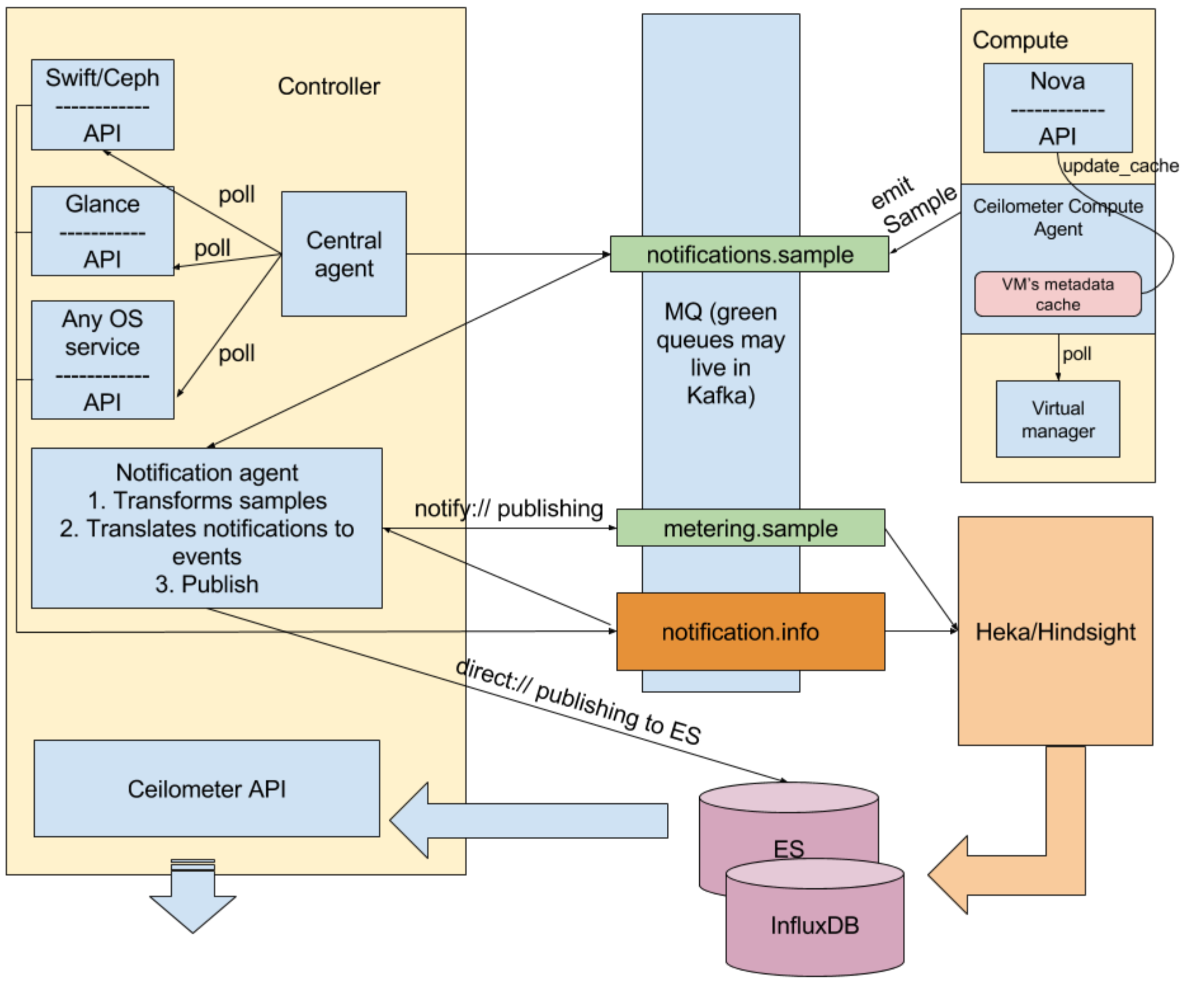
We recommend installing the Telemetry plugin along with the StackLight plugins. In this case, the Telemetry plugin will use the same databases as StackLight: InfluxDB and Elasticsearch. Otherwise, you can configure external storages.

**See also:**

- *Architecture overview*

## 1.2 Architecture overview

The Telemetry plugin uses Ceilometer agents to collect data and its own processing mechanism to put the data into storages. Ceilometer API is used to retrieve the data and present it to the end user. The following diagram shows the OpenStack Telemetry plugin architecture:



Ceilometer agents are deployed as follows:

- The central agents service is placed on controllers. This service polls metrics about OpenStack services. A central agent obtains the measurements and sends them to the `notifications.sample` queue.

  **Note:** If Kafka is not deployed, only one central agent will be running in the environment under Pacemaker. If Kafka is deployed, the coordination mechanism with Zookeeper will be automatically enabled.

- Compute agents work on compute nodes and use the same code base as the central agents. The main difference is the configuration and the fact that compute agents use metadata cache that is enabled by the Telemetry plugin. The compute agents request instance metadata from Nova API every 10 minutes, but not each polling interval. For more information, see the corresponding specification. A compute agent obtains the measurements and sends them to the `notifications.sample` queue.
- Notification agents are placed on controllers. Each notification agent performs the following:
  - Obtains data from polling agents and OpenStack services. In other words, it listens to the `notifications.sample` and `notifications.info` queues. The Telemetry plugin may be customized at this point. By default, Ceilometer notification agents do not convert OpenStack notifications to Ceilometer Events. If you enable Event API, notification agents will write Events directly to Elasticsearch with the `direct://` publisher.
  - Performs transformations and sends the data further to the `metering.sample` queue.

**Note:** In Mirantis OpenStack, Ceilometer notification agents do not require coordination. For more details, see Custom transformed metrics.

A notification agent is the last Ceilometer-related processor. As a result, all the data collected is placed in the `metering.sample` queue and Ceilometer Events are written into Elasticsearch (if Event API is enabled). Ceilometer agents work with the message brokers through `oslo.messaging` and do not depend on the message broker we use.

To continue data processing, Hindsight or Heka are used. The diagram above shows Heka/Hindsight separately because their placement depends on what is actually chosen. For information about Heka, see Heka documentation. For proper work with Kafka, we use a new generation of Heka called Hindsight. Hindsight supports all the required Kafka functionality but cannot be used to work with RabbitMQ. Therefore, these instruments are used depending on the message broker type:

- If Kafka is deployed, Hindsight is deployed on the same nodes where Kafka is running. Hindsight is started with four input plugins to make data consumption fast enough. Analysis plugins are not used. The output plugins have a batching mechanism to deliver data into the storages in an optimal manner. Hindsight services are not running under Pacemaker but will be restarted automatically in case of any failures. Heka is not used in this scenario.
- If Kafka is not deployed, RabbitMQ is used as a transport system and Heka is running on each controller under Pacemaker. Hindsight is not used in this scenario.

Once Heka or Hindsight receives a data sample, it is processed through a chain of plugins and finally sent to InfluxDB or Elasticsearch.

## 1.3 Requirements

The OpenStack Telemetry plugin has the following requirements:

| Requirement | Version/Comment |
|---|---|
| Fuel | 9.0 on Mitaka |

If you use external back ends:

| Requirement | Version/Comment |
|---|---|
| An Elasticsearch server (for Ceilometer Resources and Events) | 2.0.0 or higher, the RESTful API must be enabled over port 9200 |
| A running InfluxDB server (for Ceilometer Samples) | 0.10.0 or higher, the RESTful API must be enabled over port 8086 |

## 1.4 Compatibilities

The OpenStack Telemetry plugin is compatible with the following plugins:

- To install the back-end services automatically, use the following StackLight plugins:

| Plugin | Version/Comment |
|---|---|
| StackLight InfluxDB-Grafana | 0.10.0 or newer |
| StackLight Elasticsearch-Kibana | 0.10.2 or newer. If the Resource API is disabled, the version may be 0.10.0 |

- To use Kafka as a message queue, install:

| Plugin | Version/Comment |
|---|---|
| Kafka | 1.0.0 |

## 1.5 Prerequisites

Prior to installing the OpenStack Telemetry plugin, you may want to install the back-end services the plugin uses to store the data. These back-end services include the following:

- Elasticsearch
- InfluxDB

To install the back-end services, use one of the options:

- Automatic installation within a Fuel environment using the following Fuel plugins:
  - StackLight Elasticsearch-Kibana plugin
  - StackLight InfluxDB-Grafana plugin
- Manual installation outside of a Fuel environment. The installation must comply with the *Requirements* of the OpenStack Telemetry plugin.

## 1.6 Limitations

The OpenStack Telemetry plugin for Fuel has the following limitations:

- Ceilometer API is not fully supported by default. The following Ceilometer commands are supported:
  - By default:
    - `ceilometer sample-list`
    - `ceilometer statistics`
  - If the *Resource API* is enabled:
    - `ceilometer resource-list`
    - `ceilometer meter-list`
  - If the *Event API* is enabled:
    - `ceilometer event-list`

  Ceilometer complex queries are not supported.
- The Telemetry plugin does not store all the OpenStack resources metadata along with the Ceilometer Samples. The default list is as follows:

```
status
deleted
container_format
min_ram
updated_at
min_disk
is_public size
checksum
```

```
created_at disk_format
protected
instance_host
host
display_name
instance_id
instance_type
status
state
user_metadata.stack
```

To use the Ceilometer API requests based on metadata, add the required metadata as described in *Configure the plugin*.

- The coordination for Ceilometer central agent and Aodh alarm evaluator services is switched off if RabbitMQ is used. The Telemetry plugin is based on the Ceilometer used in Mirantis OpenStack. Therefore, the notification agents do not require coordination. The coordination through tooz with Zookeeper back end is supported if the Kafka plugin is installed.
- The OpenStack Telemetry plugin cannot be used if the Redis plugin is already enabled in the environment.

## 1.7 Licenses

### 1.7.1 Third-party components

| Name | Project website | License |
|---|---|---|
| Heka | https://github.com/mozilla-services/heka | Mozilla Public License |
| Hindsight | https://github.com/mozilla-services/hindsight | Mozilla Public License |

## 1.8 References

For more information about the software discussed in this document, see the following links:

- The StackLight Collector plugin project at GitHub
- The StackLight Elasticsearch-Kibana plugin project at GitHub
- The StackLight InfluxDB-Grafana plugin project at GitHub
- The official Kibana documentation
- The official Elasticsearch documentation
- The official InfluxDB documentation
- The official Grafana documentation
- The official Heka documentation
- The official Hindsight documentation

CHAPTER

TWO

# INSTALLING THE OPENSTACK TELEMETRY PLUGIN FOR FUEL

## 2.1 Introduction

Before you install the OpenStack Telemetry plugin, verify that your environment meets the requirements described in *Requirements*. You must have the Fuel Master node installed and configured before you can install the plugin.

You can install the OpenStack Telemetry plugin using one of the following options:

- Install using the RPM file
- Install from source

## 2.2 Install using the RPM file

**To install the OpenStack Telemetry plugin using the RPM file of the Fuel plugins catalog:**

1. Download the OpenStack Telemetry plugin from the Fuel plugins catalog.
2. Copy the plugin `.rpm` file to the Fuel Master node:

   **Example:**

   ```
   # scp telemetry-1.0-1.0.1-1.noarch.rpm root@fuel-master:/tmp
   ```

3. Log in to the Fuel Master node CLI as root.
4. Install the plugin using the Fuel Plugins CLI:

   ```
   # fuel plugins --install telemetry-1.0-1.0.0-1.noarch.rpm
   ```

5. Verify that the plugin is installed correctly:

   ```
   # fuel plugins
   id | name        | version | package_version | releases
   ---|------------|---------|-----------------|--------------------
   1  | telemetry  |  1.0.1  |  4.0.0          | ubuntu (mitaka-9.0)
   ```

6. Proceed to *Configure the plugin*.

## 2.3 Install from source

Alternatively, you may want to build the plugin RPM file from source if, for example, you want to test the latest features of the master branch or customize the plugin.

**Note:** Running a Fuel plugin that you built yourself is at your own risk and will not be supported.

To install the OpenStack Telemetry plugin from source, first prepare an environment to build the RPM file. The recommended approach is to build the RPM file directly onto the Fuel Master node so that you will not have to copy that file later on.

**To prepare an environment and build the plugin:**

1. Install the standard Linux development tools:

```
[root@home ~] yum install createrepo rpm rpm-build dpkg-devel
```

2. Install the Fuel Plugin Builder. To do that, you should first get pip:

```
[root@home ~] easy_install pip
```

3. Then install the Fuel Plugin Builder (the `fpb` command line) with `pip`:

```
[root@home ~] pip install fuel-plugin-builder
```

   **Note:** You may also need to build the Fuel Plugin Builder if the package version of the plugin is higher than the package version supported by the Fuel Plugin Builder you get from `pypi`. For instructions on how to build the Fuel Plugin Builder, see the *Install Fuel Plugin Builder* section of the Fuel Plugin SDK Guide.

4. Clone the plugin repository:

```
[root@home ~] git clone https://github.com/openstack/fuel-plugin-openstack-
↪telemetry
```

5. Verify that the plugin is valid:

```
[root@home ~] fpb --check ./fuel-plugin-openstack-telemetry
```

6. Build the plugin:

```
[root@home ~] fpb --build ./fuel-plugin-openstack-telemetry
```

**To install the plugin:**

1. Once you create the RPM file, install the plugin:

```
[root@fuel ~] fuel plugins --install ./fuel-plugin-openstack-telemetry/*.noarch.
↪rpm
```

2. Verify that the plugin is installed correctly:

```
# fuel plugins
id | name         | version | package_version | releases
---|-------------|---------|------------------|--------------------
1  | telemetry   |  1.0.1  |  4.0.0           | ubuntu (mitaka-9.0)
```

3. Proceed to *Configure the plugin*.

CHAPTER

THREE

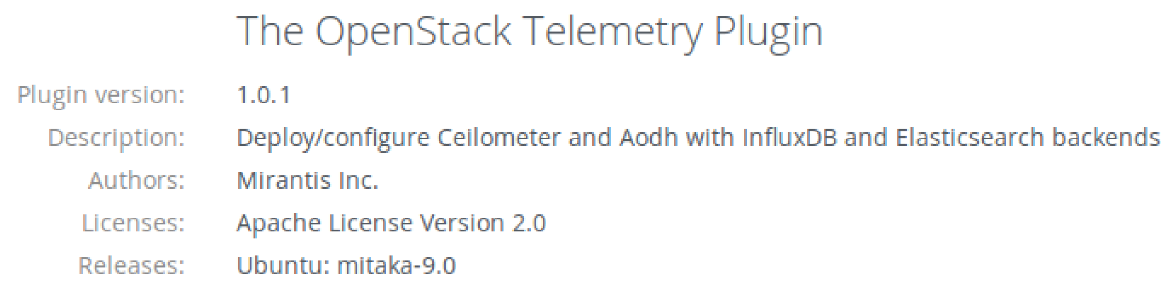
# CONFIGURING THE OPENSTACK TELEMETRY PLUGIN FOR FUEL

## 3.1 Configure the plugin

Once installed, configure the OpenStack Telemetry plugin.

**To configure the OpenStack Telemetry plugin:**

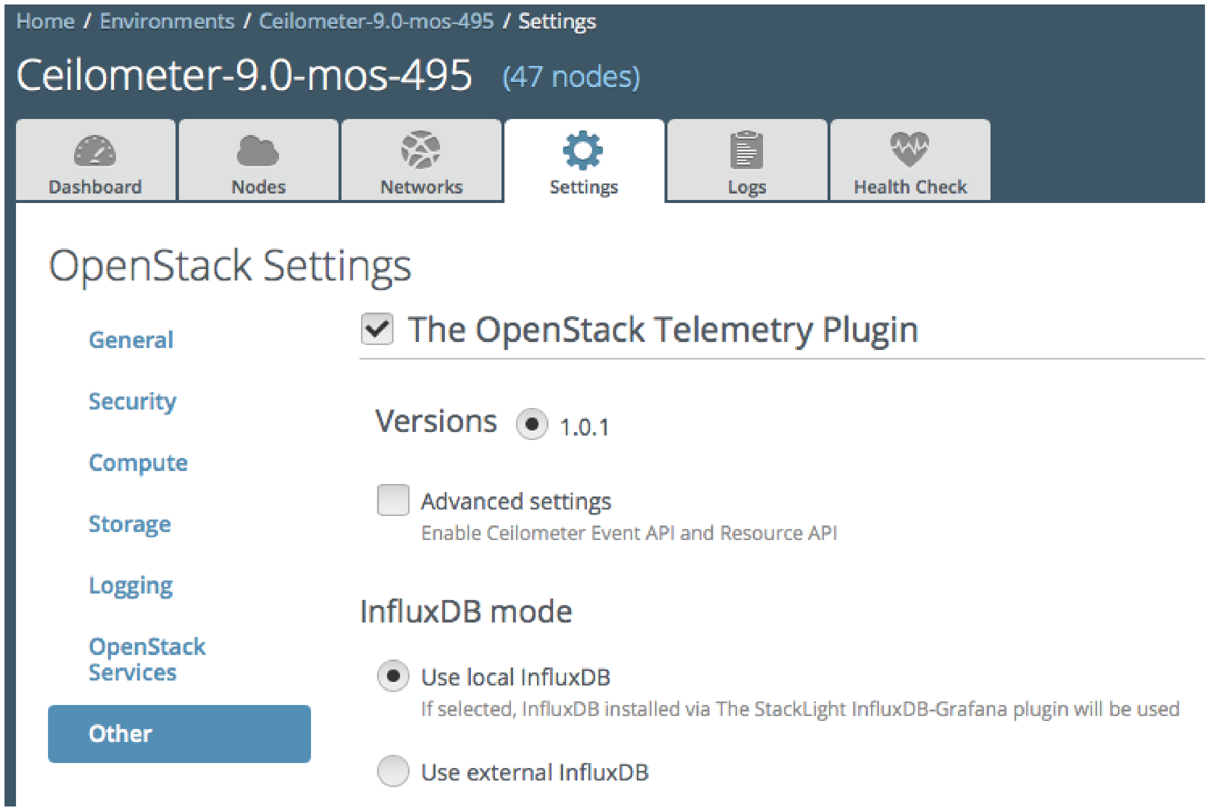
1. Log in to the Fuel web UI.
2. Verify that the Telemetry plugin is listed in the *Plugins* tab:



The OpenStack Telemetry Plugin

| | |
|---|---|
| Plugin version: | 1.0.1 |
| Description: | Deploy/configure Ceilometer and Aodh with InfluxDB and Elasticsearch backends |
| Authors: | Mirantis Inc. |
| Licenses: | Apache License Version 2.0 |
| Releases: | Ubuntu: mitaka-9.0 |

3. Create an OpenStack environment as described in the Fuel User Guide or use an existing one.
4. To enable the plugin, navigate to the *Environments* tab and select *The OpenStack Telemetry Plugin*:



☑ The OpenStack Telemetry Plugin

Versions ● 1.0.1

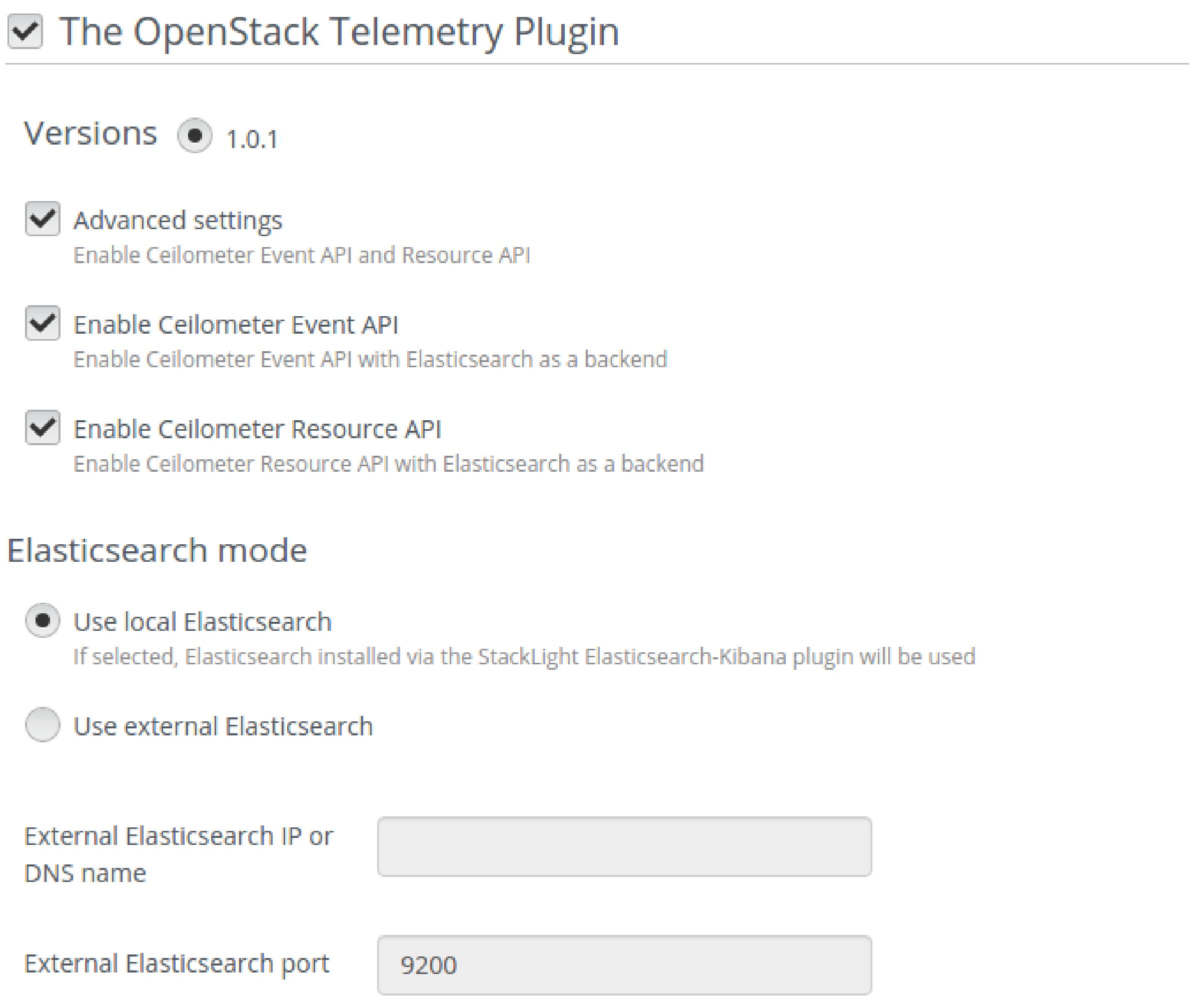
☐ Advanced settings
Enable Ceilometer Event API and Resource API

InfluxDB mode

● Use local InfluxDB
If selected, InfluxDB installed via The StackLight InfluxDB-Grafana plugin will be used

○ Use external InfluxDB

5. Optional. To enable Event API and Resource API, select *Advanced Settings*:



[✓] The OpenStack Telemetry Plugin

Versions [● 1.0.1]

[✓] Advanced settings
Enable Ceilometer Event API and Resource API

[✓] Enable Ceilometer Event API
Enable Ceilometer Event API with Elasticsearch as a backend

[✓] Enable Ceilometer Resource API
Enable Ceilometer Resource API with Elasticsearch as a backend

Elasticsearch mode

[✓ Use local Elasticsearch]
If selected, Elasticsearch installed via the StackLight Elasticsearch-Kibana plugin will be used

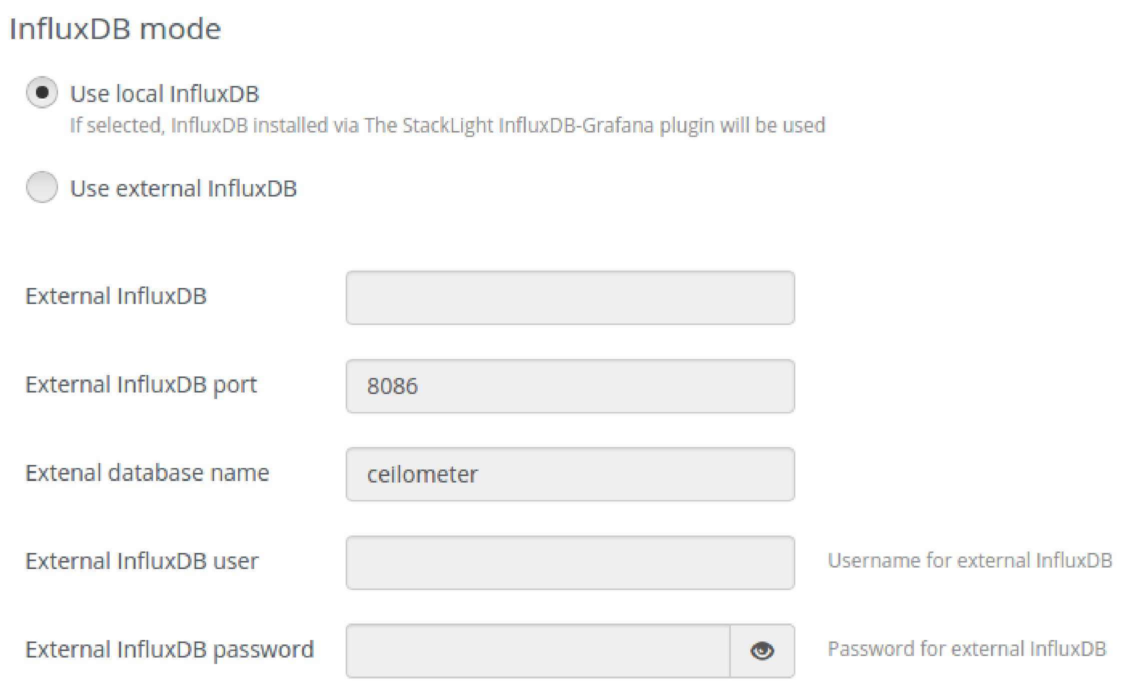
Use external Elasticsearch

| | |
|---|---|
| External Elasticsearch IP or DNS name | |
| External Elasticsearch port | 9200 |

Once selected, configure the Elasticsearch cluster that stores Ceilometer events and resources:

- Select *Use local Elasticsearch* if you have deployed the Elasticsearch-Kibana plugin.
- Otherwise, select *Use External Elasticsearch* and define the IP or DNS name and port for the Elasticsearch cluster you want to use.

6. Configure InfluxDB:



InfluxDB mode

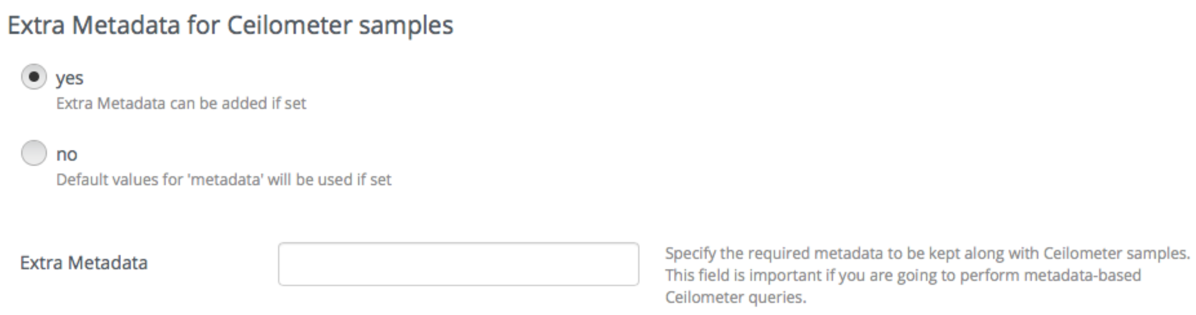
● Use local InfluxDB
If selected, InfluxDB installed via The StackLight InfluxDB-Grafana plugin will be used

○ Use external InfluxDB

| | | |
|---|---|---|
| External InfluxDB | | |
| External InfluxDB port | 8086 | |
| Extenal database name | ceilometer | |
| External InfluxDB user | | Username for external InfluxDB |
| External InfluxDB password | | Password for external InfluxDB |

- Select *Use local InfluxDB* if you have deployed the InfluxDB-Grafana plugin.
- Otherwise, select *Use External InfluxDB* and define the IP or DNS name, port, database name, username, and password for the InfluxDB server you want to use to store the Ceilometer-related data.

7. Configure additional metadata to be stored along with Ceilometer samples in InfluxDB:



Extra Metadata for Ceilometer samples

● yes
Extra Metadata can be added if set

○ no
Default values for 'metadata' will be used if set

| | | |
|---|---|---|
| Extra Metadata | | Specify the required metadata to be kept along with Ceilometer samples. This field is important if you are going to perform metadata-based Ceilometer queries. |

By default, the Telemetry plugin keeps the list of metadata fields described in the *Limitations* section. If this list is not sufficient, add the names of metadata fields.