The Ceilometer Redis Plugin

Release 1.0.3

Mirantis Inc.

July 29, 2016

CONTENTS

CHAPTER

ONE

OVERVIEW

The *Ceilometer Redis Plugin* installs Redis and the Tooz library, in a Mirantis OpenStack (MOS) environment deployed by Fuel. Both Redis and the Tooz library should be installed on all the controller nodes of the environment. Starting from MOS 9.0, Ceilometer alarming service was moved to the project called Aodh.

The *Ceilometer Redis Plugin* is used to provide coordination mechanisms to enable the horizontal scaling of the Ceilometer/Aodh services. Using the plugin, the Ceilometer/Aodh services are joined into a so-called **coordination group**, which allows for resources and alarms sharding. There is one coordination group per service type.

Please refer to the Telemetry architecture documentation for more information about the Ceilometer services.

In MOS 9.0, the *Ceilometer Redis Plugin* enables coordination for both:

- The **ceilometer-agent-central service**.

  The ceilometer-agent-central service is responsible for polling all the OpenStack resources, excepted those of Nova, like the VM instances, that are polled by the **ceilometer-agent-compute**. Without coordination, there can be only one ceilometer-agent-central running at a time. This is because, by default, the ceilometer-agent-central works with an entire set of resources. As such, running multiple ceilometer-agent-central without coordination would poll the entire set of resources as many times as the number of agents running on the controller nodes every polling interval. This is obviously not a proper way to scale out the ceilometer-agent-central. To cope with this problem, the coordination mechanism provided by the *Ceilometer Redis Plugin* allows distributing the polling workload across multiple instances of the ceilometer-agent-central using disjoint sets of resources.

- The **aodh-evaluator service**.

  The **aodh-evaluator** service is responsible for evaluating the Ceilometer alarms. By default, there is only one aodh-evaluator running per environment. Without coordination, there can be only one aodh-evaluator running at a time. This is because, as for the ceilometer-agent-central, the aodh-evaluator works with an entire set of alarms. Running multiple aodh-evaluator without coordination would evaluate all the alarms as many times as the number of evaluators running on the controller nodes every evaluation interval. To cope with this problem, the coordination mechanism provided by the *Ceilometer Redis Plugin* allows distributing the alarms evaluation workload across multiple instances of the aodh-evaluator using disjoint sets of alarms.

Please note that starting from MOS 8.0, the *Ceilometer Redis Plugin* doesn't provide support (out-of-the-box) for the coordination of the **ceilometer-agent-notification** service because it is not needed for the most common samples transformations.

Note: Before Liberty, the transformation of the samples was handled by the **ceilometer-agent-compute** and the **ceilometer-agent-central** services. In Liberty, the transformation of the samples was moved to the **ceilometer-agent-notification** service, but after thorough performance analysis of Ceilometer at scale, we discovered that this change has a bad impact on performance. Starting from MOS 8.0, the transformations for the following list of measurements were moved back to the ceilometer-agent-compute service.

- cpu_util
- disk.read.requests.rate

- disk.write.requests.rate
- disk.read.bytes.rate
- disk.write.bytes.rate
- disk.device.read.requests.rate
- disk.device.read.bytes.rate
- disk.device.write.bytes.rate
- network.incoming.bytes.rate
- network.outgoing.bytes.rate
- network.incoming.packets.rate
- network.outgoing.packets.rate

As a result, starting from MOS 8.0, there is no need to run the ceilometer-agent-notification in coordination mode unless you need to maintain the transformation of custom samples that are not listed above. In this case, it is possible to enable coordination for the ceilometer-agent-notification service manually event though, it is not recommended for performance reasons.

In addition to the above, the *Ceilometer Redis Plugin* configures *Pacemaker* and redis-sentinel to enable **high availability** of the Redis cluster. Redis clustering includes:

- Monitoring the state of the **redis-server** processes
- Elect a new redis-server master during a failover
- Connect Ceilometer to the elected redis-server
- Organize the synchronization between Redis nodes

1.1 Requirements

Requirements	Version/Comment
MOS	9.0
Tooz	>=1.28.0

1.2 Limitations

- The *Ceilometer Redis Plugin* requires to install on an odd number of controller nodes to work properly. This is because, Redis clustering requires an odd number of nodes to avoid the split brain effect when electing a master.
- If you have any custom transformers you need to ensure that they are cache-less. If transformation is cache-less, then there is no need to enable the coordination. That is, based only on the `unit_conversion` transformer or the `arithmetic` transformers. Otherwise, you will have to consider running only one instance of the ceilometer-agent-notification service in your MOS environment or install the *Ceilometer Redis Plugin* and do all the configuration manually.

CHAPTER

TWO

INSTALLATION GUIDE

2.1 Install the Plugin

To install the *Ceilometer Redis Plugin*, you need to follow these steps.

1. Please refer to the *Limitations* section before you proceed.
2. Download the plugin from the Fuel Plugins Catalog.
3. Copy the plugin's RPM file to the Fuel Master node with secure copy (scp):

```
# scp fuel-plugin-ceilometer-redis/ceilometer-redis-1.0-1.0.3-1.noarch.rpm /
root@:<the_Fuel_Master_node_IP address>:/tmp
```

4. Log into the Fuel Master node and install the plugin:

```
# ssh root@:<the_Fuel_Master_node_IP address>
[root@fuel-master ~]# cd /tmp
[root@fuel-master ~]# fuel plugins --install ceilometer-redis-1.0-1.0.3-1.noarch.rpm
```

5. Verify that the plugin is installed correctly:

```
[root@fuel-master ~]# fuel plugins list
id | name             | version       | package_version
---|------------------|---------------|----------------
4  | ceilometer-redis | 1.0.3         | 3.0.0
```

CHAPTER

THREE

USER GUIDE

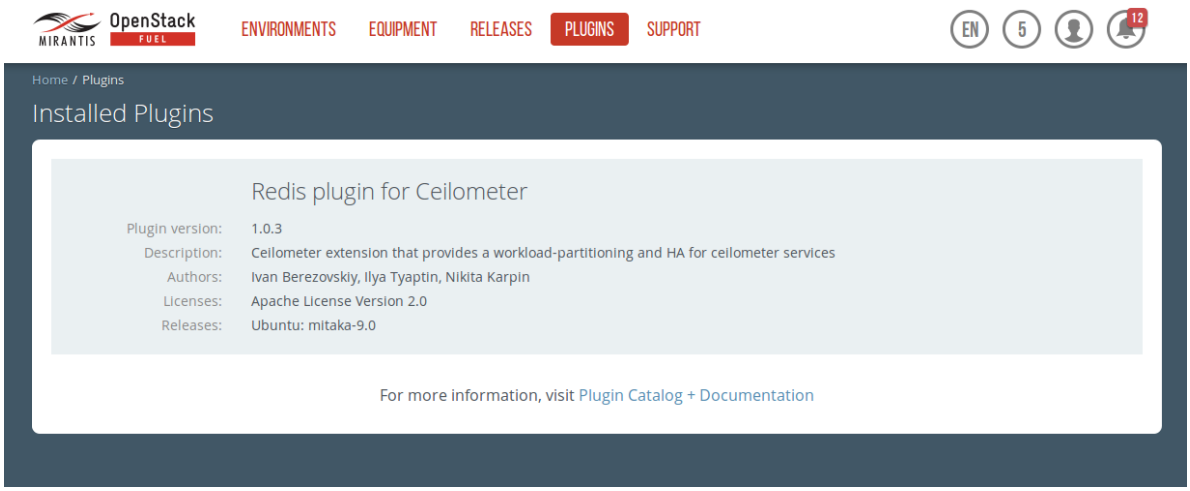
Once the *Ceilometer Redis Plugin* is installed following the instructions of the *Installation Guide*, you can create a Mirantis OpenStack (MOS) environment with Ceilometer and Aodh whose **ceilometer-agent-central** and **aodh-evaluator** services will work in **workload partitioned** mode. This plugin was created to enable the scale-out of these Ceilometer/Aodh services. It is useless and **shouldn't be used if Ceilometer and Aodh are not installed**.

3.1 Plugin Configuration

To use the *Ceilometer Redis Plugin*, you need to create a new MOS environment with the Telemetry service (a.k.a Ceilometer) enabled and follow these steps using the *Fuel UI Wizard*.

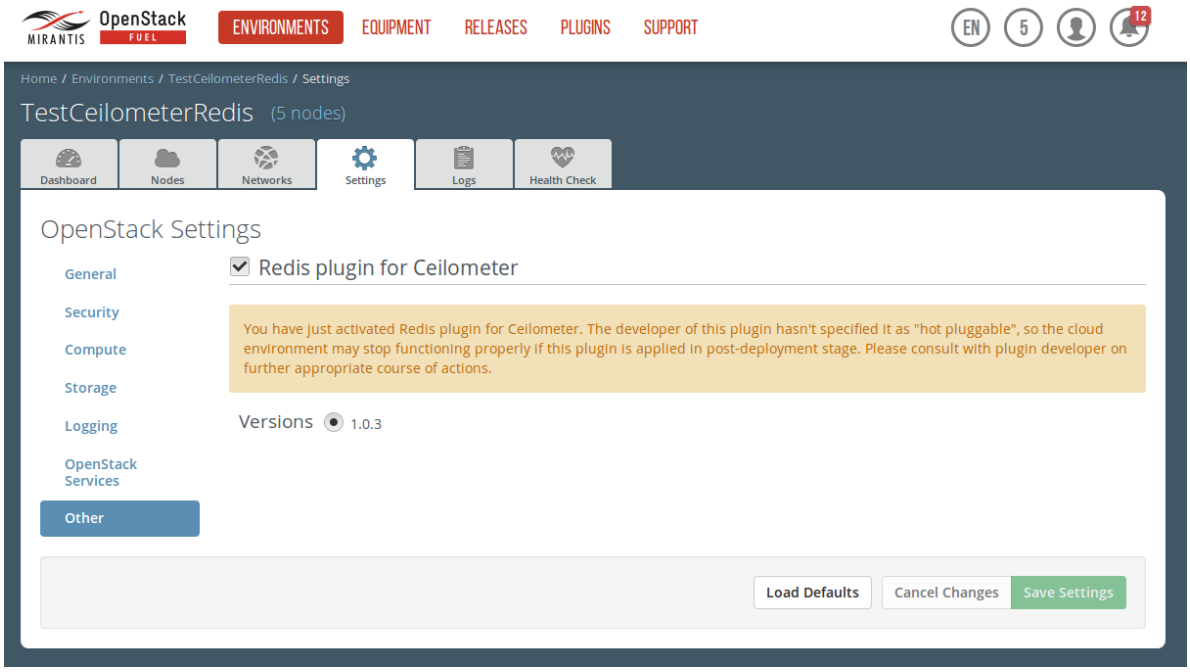
1. Make sure that the plugin is properly installed on the Fuel Master node.

   Go to the *Plugins* tab. You should see the following:



2. Enable the plugin.

   Go to the *Environments* tab and select the *Redis plugin for Ceilometer* checkbox:



3. Add nodes to your environment to which you will assign the **controller role**.

Note: When adding nodes to the environment and assign or change a role, do not forget to use an odd number of controllers as mentioned in *Limitations* section.

---

4. Verify your network configuration.
5. Deploy your changes once you are done with the configuration of your environment.

3.2 Plugin Verification

1. Check that the ceilometer-agent-central and aodh-evaluator services are running on each controller.

   Run "http://docs.openstack.org/developer/fuel-docs/userdocs/fuel-user-guide/deploy-environment.html". You should see the following in the output:

   ```
   Clone Set: clone_p_ceilometer-agent-central [p_ceilometer-agent-central]
     Started: [ node-21.domain.tld node-27.domain.tld node-33.domain.tld ]

   Clone Set: clone_p_aodh-evaluator [p_aodh-evaluator]
     Started: [ node-21.domain.tld node-27.domain.tld node-33.domain.tld ]
   ```

   The *Started* list should contain all controllers.

2. For the ceilometer-agent-central, check that the samples are not duplicated. For this check you may choose any metric collected by the ceilometer-agent-central. All the Ceilometer metrics can be found in Measurements . You may choose any section excepted *OpenStack Compute* and then select a metric with *Pollster Origin*. For example, let's choose *storage.objects*.

   The plugin **works correctly** if you see one sample for each resource type every *polling interval* (1 minute in this example):

```
root@node-2:~# ceilometer sample-list -m storage.objects  -l 10| grep storage.objects
| 65e486c7... | storage.objects | gauge | 0.0    | object | 2015-11-05T10:32:27 |
| 65e486c7... | storage.objects | gauge | 0.0    | object | 2015-11-05T10:31:29 |
```

The plugin **works incorrectly** if there are duplicates. In this example, the plugin works incorectly because there are three samples for the same resource type every *polling interval*:

```
root@node-2:~# ceilometer sample-list -m storage.objects  -l 20| grep storage.objects
| 65e486c7... | storage.objects | gauge | 0.0    | object | 2015-11-05T10:27:37 |
| 65e486c7... | storage.objects | gauge | 0.0    | object | 2015-11-05T10:27:26 |
| 65e486c7... | storage.objects | gauge | 0.0    | object | 2015-11-05T10:27:17 |
| 65e486c7... | storage.objects | gauge | 0.0    | object | 2015-11-05T10:26:38 |
| 65e486c7... | storage.objects | gauge | 0.0    | object | 2015-11-05T10:26:26 |
| 65e486c7... | storage.objects | gauge | 0.0    | object | 2015-11-05T10:26:17 |
```

3. For the alarm evaluator, it is possible to see that everything works as expected only from the logs:

   ```
   # grep extract_my_subset /var/log/aodh/aodh-evaluator.log
   ```

   There should be different *My subset: [* results for the aodh-evaluator instances.

CHAPTER

FOUR

INDICES AND TABLES

- *search*