cloudkitty Documentation

Release 0.1

Objectif Libre

March 01, 2017

Contents

CHAPTER 1

Introduction

CloudKitty is a Rating As A Service project aimed at translating metrics to prices.

CHAPTER 2

Installation

DevStack installation

Add the following lines in your `local.conf` file to enable CloudKitty with the ceilometer collector:

```
[[local|localrc]]
# ceilometer
enable_plugin ceilometer https://git.openstack.org/openstack/ceilometer.git master

# horizon
enable_service horizon

# cloudkitty
enable_plugin cloudkitty https://git.openstack.org/openstack/cloudkitty.git master
enable_service ck-api ck-proc
CLOUDKITTY_COLLECTOR=ceilometer
```

To enable the gnocchi collector, use the following instead:

```
[[local|localrc]]
# gnocchi
enable_plugin gnocchi https://github.com/openstack/gnocchi master
enable_service gnocchi-api,gnocchi-metricd

# ceilometer
enable_plugin ceilometer https://git.openstack.org/openstack/ceilometer.git master

# horizon
enable_service horizon

# cloudkitty
enable_plugin cloudkitty https://git.openstack.org/openstack/cloudkitty.git master
enable_service ck-api ck-proc
CLOUDKITTY_COLLECTOR=gnocchi
```

Then start devstack:

```
./stack.sh
```

Cloudkitty installation and configuration

Many method can be followed to install cloudkitty.

Install from source

Install the services

Retrieve and install cloudkitty:

```
git clone https://git.openstack.org/openstack/cloudkitty.git
cd cloudkitty
python setup.py install
```

This procedure installs the `cloudkitty` python library and the following executables:

- `cloudkitty-api`: API service
- `cloudkitty-processor`: Processing service (collecting and rating)
- `cloudkitty-dbsync`: Tool to create and upgrade the database schema
- `cloudkitty-storage-init`: Tool to initiate the storage backend
- `cloudkitty-writer`: Reporting tool

Install sample configuration files:

```
mkdir /etc/cloudkitty
tox -e genconfig
cp etc/cloudkitty/cloudkitty.conf.sample /etc/cloudkitty/cloudkitty.conf
cp etc/cloudkitty/policy.json /etc/cloudkitty
cp etc/cloudkitty/api_paste.ini /etc/cloudkitty
```

Create the log directory:

```
mkdir /var/log/cloudkitty/
```

Install the client

Retrieve and install cloudkitty client:

```
git clone https://git.openstack.org/openstack/python-cloudkittyclient.git
cd python-cloudkittyclient
python setup.py install
```

Install the dashboard module

1. Retrieve and install cloudkitty's dashboard:

   ```
   git clone https://git.openstack.org/openstack/cloudkitty-dashboard.git
   cd cloudkitty-dashboard
   python setup.py install
   ```

2. Find where the python packages are installed:

```
PY_PACKAGES_PATH=`pip --version | cut -d' ' -f4`
```

3. Add the enabled file to the horizon settings or installation. Depending on your setup, you might need to add it to `/usr/share` or directly in the horizon python package:

```
# If horizon is installed by packages:
ln -sf $PY_PACKAGES_PATH/cloudkittydashboard/enabled/_[0-9]*.py \
/usr/share/openstack-dashboard/openstack_dashboard/enabled/

# Directly from sources:
ln -sf $PY_PACKAGES_PATH/cloudkittydashboard/enabled/_[0-9]*.py \
$PY_PACKAGES_PATH/openstack_dashboard/enabled/
```

4. Restart the web server hosting Horizon.

Install from packages

Packages for RHEL/CentOS 7 and Ubuntu 16.04 are available for the Newton release.

For RHEL/CentOS 7

1. Install the RDO repositories for Newton:

```
yum install centos-release-openstack-newton
```

2. Install the packages:

```
yum install openstack-cloudkitty-api openstack-cloudkitty-processor openstack-cloudkitty-ui
```

For Ubuntu 16.04

1. Enable the OpenStack repository for the Newton release:

```
apt install software-properties-common
add-apt-repository ppa:objectif-libre/cloudkitty
```

2. Upgrade the packages on your host:

```
apt update && apt dist-upgrade
```

3. Install the packages:

```
apt-get install cloudkitty-api cloudkitty-processor cloudkitty-dashboard
```

Configure cloudkitty

Edit `/etc/cloudkitty/cloudkitty.conf` to configure cloudkitty.

Then you need to know which keystone API version you use (which can be determined using *openstack endpoint list*)

For keystone (identity) API v2 (deprecated)

```
[DEFAULT]
verbose = True
log_dir = /var/log/cloudkitty

[oslo_messaging_rabbit]
rabbit_userid = openstack
rabbit_password = RABBIT_PASSWORD
rabbit_host = RABBIT_HOST
rabbit_port = 5672

[auth]
username = cloudkitty
password = CK_PASSWORD
tenant = service
region = RegionOne
url = http://localhost:5000/v2.0

[keystone_authtoken]
username = cloudkitty
password = CK_PASSWORD
project_name = service
region = RegionOne
auth_url = http://localhost:5000/v2.0
auth_plugin = password

[database]
connection = mysql://cloudkitty:CK_DBPASSWORD@localhost/cloudkitty

[keystone_fetcher]
username = cloudkitty
password = CK_PASSWORD
tenant = service
region = RegionOne
url = http://localhost:5000/v2.0

[collect]
collector = ceilometer
period = 3600
services = compute, volume, network.bw.in, network.bw.out, network.floating, image

[ceilometer_collector]
username = cloudkitty
password = CK_PASSWORD
tenant = service
region = RegionOne
url = http://localhost:5000/v2.0
```

Please note that:

- *http://localhost:5000/v2.0* and *http://localhost:35357/v2.0* are your identity endpoints.
- the tenant named *service* is also commonly called *services*

For keystone (identity) API v3

The following shows the basic configuration items:

```
[DEFAULT]
verbose = True
log_dir = /var/log/cloudkitty

[oslo_messaging_rabbit]
rabbit_userid = openstack
rabbit_password = RABBIT_PASSWORD
rabbit_host = RABBIT_HOST
rabbit_port = 5672

[ks_auth]
auth_type = v3password
auth_protocol = http
auth_url = http://localhost:5000/v3
identity_uri = http://localhost:35357/v3
username = cloudkitty
password = CK_PASSWORD
project_name = service
user_domain_name = default
project_domain_name = default
debug = True

[keystone_authtoken]
auth_section = ks_auth

[database]
connection = mysql://cloudkitty:CK_DBPASSWORD@localhost/cloudkitty

[keystone_fetcher]
auth_section = ks_auth
keystone_version = 3

[tenant_fetcher]
backend = keystone

[collect]
collector = ceilometer
period = 3600
services = compute, volume, network.bw.in, network.bw.out, network.floating, image

[ceilometer_collector]
auth_section = ks_auth
```

Please note that:

- *http://localhost:5000/v3* and *http://localhost:35357/v3* are your identity endpoints.
- the tenant named *service* is also commonly called *services*

Setup the database and storage backend

MySQL/MariaDB is the recommended database engine. To setup the database, use the `mysql` client:

```
mysql -uroot -p << EOF
CREATE DATABASE cloudkitty;
GRANT ALL PRIVILEGES ON cloudkitty.* TO 'cloudkitty'@'localhost' IDENTIFIED BY 'CK_DBPASSWORD';
EOF
```

If you need to authorize the cloudkitty mysql user from another host you have to change the line accordingly.

Run the database synchronisation scripts:

```
cloudkitty-dbsync upgrade
```

Init the storage backend:

```
cloudkitty-storage-init
```

Setup Keystone

cloudkitty uses Keystone for authentication, and provides a `rating` service.

To integrate cloudkitty to Keystone, run the following commands (as OpenStack administrator):

```
openstack user create cloudkitty --password CK_PASSWORD --email cloudkitty@localhost
openstack role add --project service --user cloudkitty admin
```

Give the `rating` role to `cloudkitty` for each project that should be handled by cloudkitty:

```
openstack role create rating
openstack role add --project XXX --user cloudkitty rating
```

Create the `rating` service and its endpoints:

```
openstack service create rating --name cloudkitty \
    --description "OpenStack Rating Service"
openstack endpoint create rating --region RegionOne \
    public http://localhost:8889
openstack endpoint create rating --region RegionOne \
    admin http://localhost:8889
openstack endpoint create rating --region RegionOne \
    internal http://localhost:8889
```

Note: The default port for the API service changed from 8888 to 8889 in the Newton release. If you installed Cloudkitty in an earlier version, make sure to either explicitly define the `[api]/port` setting to 8888 in `cloudkitty.conf`, or update your keystone endpoints to use the 8889 port.

Start cloudkitty

If you installed cloudkitty from packages

Start the API and processing services:

```
systemctl start cloudkitty-api.service
systemctl start cloudkitty-processor.service
```

If you installed cloudkitty from sources

Start the API and processing services:

```
cloudkitty-api --config-file /etc/cloudkitty/cloudkitty.conf
cloudkitty-processor --config-file /etc/cloudkitty/cloudkitty.conf
```

CHAPTER 3

Architecture

CloudKitty's Architecture

CloudKitty can be cut in five big parts:

- API
- Data collection (collector)
- Rating processing
- Storage
- Report writer

Module loading and extensions

Nearly every part of CloudKitty makes use of stevedore to load extensions dynamically.

Every rating module is loaded at runtime and can be enabled/disabled directly via CloudKitty's API. The module is responsible of its own API to ease the management of its configuration.

Collectors and storage backends are loaded with stevedore but configured in CloudKitty's configuration file.

Collector

Loaded with stevedore

The name of the collector to use is specified in the configuration, only one collector can be loaded at once. This part is responsible of information gathering. It consists of a python class that loads data from a backend and return it in a format that CloudKitty can handle.

The data format of CloudKitty is the following:

```
{
    "myservice": [
        {
            "rating": {
                "price": 0.1
            },
            "desc": {
                "sugar": "25",
                "fiber": "10",
```

```
                "name": "apples",
            },
            "vol": {
                "qty": 1,
                "unit": "banana"
            }
        }
    ]
}
```

Example code of a basic collector:

```
class MyCollector(BaseCollector):
    def __init__(self, **kwargs):
        super(MyCollector, self).__init__(**kwargs)

    def get_mydata(self, start, end=None, project_id=None, q_filter=None):
        # Do stuff
        return ck_data
```

You'll now be able to add the gathering of mydata in CloudKitty by modifying the configuration and specifying the new service in collect/services.

If you need to load multiple collectors, you can use the `meta` collector and use its API to enable/disable collector loading, and set priority.

Rating

Loaded with stevedore

This is where every rating calculations is done. The data gathered by the collector is pushed in a pipeline of rating processors. Every processor does its calculations and updates the data.

Example of minimal rating module (taken from the Noop module):

```
class Noop(rating.RatingProcessorBase):

    controller = NoopController
    description = 'Dummy test module'

    @property
    def enabled(self):
        """Check if the module is enabled

        :returns: bool if module is enabled
        """
        return True

    @property
    def priority(self):
        return 1

    def reload_config(self):
        pass

    def process(self, data):
        for cur_data in data:
            cur_usage = cur_data['usage']
```

```
        for service in cur_usage:
            for entry in cur_usage[service]:
                if 'rating' not in entry:
                    entry['rating'] = {'price': decimal.Decimal(0)}
    return data
```

Storage

Loaded with stevedore

The storage module is responsible of storing the data in a backend. It implements an API on top of the storage to be able to query the data without the need of knowing the type of backend used.

You can use the API to create reports on the fly for example.

Writer

Loaded with stevedore

In the same way as the rating pipeline, the writing is handled with a pipeline. The data is pushed to write orchestrator that will store the data in a transient DB (in case of output file invalidation). And then to every writer in the pipeline which is responsible of the writing.

CHAPTER 4

API References

CloudKitty REST API (root)

type `APILink`
API link description.

Data samples:

Json

```
{
    "href": "http://127.0.0.1:8889/v1",
    "rel": "self",
    "type": "text/html"
}
```

XML

```
<value>
  <type>text/html</type>
  <rel>self</rel>
  <href>http://127.0.0.1:8889/v1</href>
</value>
```

`href`

Type unicode

URL of the link.

`rel`

Type unicode

Relationship with this link.

`type`

Type unicode

Type of link.

type `APIMediaType`
Media type description.

Data samples:

Json

```
{
    "base": "application/json",
    "type": "application/vnd.openstack.cloudkitty-v1+json"
}
```

XML

```
<value>
  <base>application/json</base>
  <type>application/vnd.openstack.cloudkitty-v1+json</type>
</value>
```

base

Type unicode

Base type of this media type.

type

Type unicode

Type of this media type.

type APIVersion

API Version description.

Data samples:

Json

```
{
    "id": "v1",
    "links": [
        {
            "href": "http://127.0.0.1:8889/v1",
            "rel": "self",
            "type": "text/html"
        }
    ],
    "media_types": [
        {
            "base": "application/json",
            "type": "application/vnd.openstack.cloudkitty-v1+json"
        }
    ],
    "status": "STABLE",
    "updated": "2014-08-11T16:00:00Z"
}
```

XML

```
<value>
  <id>v1</id>
  <status>STABLE</status>
  <updated>2014-08-11T16:00:00Z</updated>
  <links>
    <item>
      <type>text/html</type>
      <rel>self</rel>
      <href>http://127.0.0.1:8889/v1</href>
    </item>
```

```
      </links>
      <media_types>
        <item>
          <base>application/json</base>
          <type>application/vnd.openstack.cloudkitty-v1+json</type>
        </item>
      </media_types>
    </value>
```

id

Type unicode

ID of the version.

links

Type list(APILink)

List of links to API resources.

media_types

Type list(APIMediaType)

Types accepted by this API.

status

Type Enum(EXPERIMENTAL, STABLE)

Status of the version.

updated

Type unicode

Last update in iso8601 format.

CloudKitty REST API (v1)

Collector

GET /v1/collector
Unused function, hack to let pecan route requests to subcontrollers.

GET /v1/collector/mappings
Return the list of every services mapped to a collector.

Parameters

- **collector** (unicode) – Filter on the collector name.

Return Service to collector mappings collection.

Return type ServiceToCollectorMappingCollection

GET /v1/collector/mappings/ (*service*)
Return a service to collector mapping.

Parameters

- **service** (unicode) – Name of the service to filter on.

Return type ServiceToCollectorMapping

POST /v1/collector/mappings
Create a service to collector mapping.

Parameters

- **collector** (unicode) – Name of the collector to apply mapping on.
- **service** (unicode) – Name of the service to apply mapping on.

Return type ServiceToCollectorMapping

DELETE /v1/collector/mappings
Delete a service to collector mapping.

Parameters

- **service** (unicode) – Name of the service to filter on.

GET /v1/collector/states
Query the enable state of a collector.

Parameters

- **name** (unicode) – Name of the collector.

Return State of the collector.

Return type CollectorInfos

PUT /v1/collector/states
Set the enable state of a collector.

Parameters

- **name** (unicode) – Name of the collector.
- **infos** (CollectorInfos) – New state informations of the collector.

Return State of the collector.

Return type CollectorInfos

type CollectorInfos
Type describing a collector module.

Data samples:

Json

```
{
    "enabled": true,
    "name": "ceilometer"
}
```

XML

```
<value>
  <name>ceilometer</name>
  <enabled>true</enabled>
</value>
```

enabled

Type bool

State of the collector.

name

Type unicode

Name of the collector.

type ServiceToCollectorMapping

Type describing a service to collector mapping.

Data samples:

Json

```
{
    "collector": "ceilometer",
    "service": "compute"
}
```

XML

```
<value>
  <service>compute</service>
  <collector>ceilometer</collector>
</value>
```

collector

Type unicode

Name of the collector.

service

Type unicode

Name of the service.

type ServiceToCollectorMappingCollection

Type describing a service to collector mapping collection.

Data samples:

Json

```
{
    "mappings": [
        {
            "collector": "ceilometer",
            "service": "compute"
        }
    ]
}
```

XML

```
<value>
  <mappings>
    <item>
      <service>compute</service>
      <collector>ceilometer</collector>
    </item>
  </mappings>
</value>
```

mappings

Type list(ServiceToCollectorMapping)

List of service to collector mappings.

Info

GET /v1/info/config
Return current configuration.

Return type dict(`str`: `None`)

GET /v1/info/services
Get the service list.

Return List of every services.

Return type `CloudkittyServiceInfoCollection`

GET /v1/info/services/ (*service_name*)
Return a service.

Parameters

- **service_name** (`unicode`) – name of the service.

Return type `CloudkittyServiceInfo`

type CloudkittyServiceInfo
Type describing a service info in CloudKitty.

Data samples:

Json

```
{
    "metadata": [
        "resource_id",
        "flavor",
        "availability_zone"
    ],
    "service_id": "compute",
    "unit": "instance"
}
```

XML

```
<value>
  <service_id>compute</service_id>
  <metadata>
    <item>resource_id</item>
    <item>flavor</item>
    <item>availability_zone</item>
  </metadata>
  <unit>instance</unit>
</value>
```

metadata

Type list(unicode)

List of service metadata

service_id

Type Enum(compute, image, volume, network.bw.in, network.bw.out, network.floating)

Name of the service.

unit

Type unicode

service unit

type CloudkittyServiceInfoCollection

A list of CloudKittyServiceInfo.

Data samples:

Json

```
{
    "services": [
        {
            "metadata": [
                "resource_id",
                "flavor",
                "availability_zone"
            ],
            "service_id": "compute",
            "unit": "instance"
        }
    ]
}
```

XML

```
<value>
  <services>
    <item>
      <service_id>compute</service_id>
      <metadata>
        <item>resource_id</item>
        <item>flavor</item>
        <item>availability_zone</item>
      </metadata>
      <unit>instance</unit>
    </item>
  </services>
</value>
```

Rating

GET /v1/rating/modules

return the list of loaded modules.

Return name of every loaded modules.

Return type CloudkittyModuleCollection

GET /v1/rating/modules/(*module_id*)

return a module

Return CloudKittyModule

Return type CloudkittyModule

PUT /v1/rating/modules
Change the state and priority of a module.

Parameters

- **module_id** (unicode) – name of the module to modify
- **module** (CloudkittyModule) – CloudKittyModule object describing the new desired state

Return type CloudkittyModule

POST /v1/rating/quote
Get an instant quote based on multiple resource descriptions.

Parameters

- **res_data** (CloudkittyResourceCollection) – List of resource descriptions.

Return Total price for these descriptions.

Return type float

GET /v1/rating/reload_modules
Trigger a rating module list reload.

type CloudkittyModule
A rating extension summary

Data samples:

Json

```
{
    "description": "Sample extension.",
    "enabled": true,
    "hot-config": false,
    "priority": 2
}
```

XML

```
<value>
  <description>Sample extension.</description>
  <enabled>true</enabled>
  <hot-config>false</hot-config>
  <priority>2</priority>
</value>
```

description

Type unicode

Short description of the extension.

enabled

Type bool

Extension status.

hot_config

Type bool

On-the-fly configuration support.

module_id

Type unicode

Name of the extension.

priority

Type int

Priority of the extension.

type CloudkittyModuleCollection

A list of rating extensions.

Data samples:

Json

```
{}
```

XML

```
<value />
```

type CloudkittyResource

Type describing a resource in CloudKitty.

Data samples:

Json

```
{
    "desc": {
        "image_id": "a41fba37-2429-4f15-aa00-b5bc4bf557bf"
    },
    "service": "compute",
    "volume": "1"
}
```

XML

```
<value>
  <service>compute</service>
  <desc>
    <item>
      <key>image_id</key>
      <value>a41fba37-2429-4f15-aa00-b5bc4bf557bf</value>
    </item>
  </desc>
  <volume>1</volume>
</value>
```

desc

Type dict(unicode: None)

Description of the resources parameters.

service

Type Enum(compute, image, volume, network.bw.in, network.bw.out, network.floating)

Name of the service.

volume

Type Decimal

Volume of resources.

type `CloudkittyResourceCollection`
A list of CloudKittyResources.

Data samples:

Json

```
{}
```

XML

```
<value />
```

Report

`GET /v1/report/summary`
Return the summary to pay for a given period.

Return type `SummaryCollectionModel`

`GET /v1/report/tenants`
Return the list of rated tenants.

Return type list(`unicode`)

`GET /v1/report/total`
Return the amount to pay for a given period.

Return type `Decimal`

Storage

`GET /v1/storage/dataframes`
Return a list of rated resources for a time period and a tenant.

Parameters

- **begin** (`datetime`) – Start of the period
- **end** (`datetime`) – End of the period
- **tenant_id** (`unicode`) – UUID of the tenant to filter on.
- **resource_type** (`unicode`) – Type of the resource to filter on.

Return Collection of DataFrame objects.

Return type `DataFrameCollection`

type `RatedResource`
Represents a rated CloudKitty resource.

Data samples:

Json

```
{
    "desc": {
        "flavor": "m1.tiny",
        "vcpus": "1"
    },
    "rating": "1.0",
    "service": "compute",
    "volume": "1.0"
}
```

XML

```
<value>
  <rating>1.0</rating>
  <service>compute</service>
  <desc>
    <item>
      <key>flavor</key>
      <value>m1.tiny</value>
    </item>
    <item>
      <key>vcpus</key>
      <value>1</value>
    </item>
  </desc>
  <volume>1.0</volume>
</value>
```

type `DataFrame`

Type describing a stored data frame.

Data samples:

Json

```
{
    "begin": "2015-04-22T07:00:00",
    "end": "2015-04-22T08:00:00",
    "resources": [
        {
            "desc": {
                "flavor": "m1.tiny",
                "vcpus": "1"
            },
            "rating": "1.0",
            "service": "compute",
            "volume": "1.0"
        }
    ],
    "tenant_id": "69d12143688f413cbf5c3cfe03ed0a12"
}
```

XML

```
<value>
  <begin>2015-04-22T07:00:00</begin>
  <end>2015-04-22T08:00:00</end>
  <tenant_id>69d12143688f413cbf5c3cfe03ed0a12</tenant_id>
  <resources>
    <item>
```

```
            <rating>1.0</rating>
            <service>compute</service>
            <desc>
              <item>
                <key>flavor</key>
                <value>m1.tiny</value>
              </item>
              <item>
                <key>vcpus</key>
                <value>1</value>
              </item>
            </desc>
            <volume>1.0</volume>
          </item>
        </resources>
      </value>
```

begin

Type datetime

Begin date for the sample.

end

Type datetime

End date for the sample.

resources

Type list(RatedResource)

A resource list.

tenant_id

Type unicode

Tenant owner of the sample.

type DataFrameCollection

A list of stored data frames.

Data samples:

Json

```
{
    "dataframes": [
        {
            "begin": "2015-04-22T07:00:00",
            "end": "2015-04-22T08:00:00",
            "resources": [
                {
                    "desc": {
                        "flavor": "m1.tiny",
                        "vcpus": "1"
                    },
                    "rating": "1.0",
                    "service": "compute",
                    "volume": "1.0"
                }
```

```
                ],
                "tenant_id": "69d12143688f413cbf5c3cfe03ed0a12"
            }
        ]
    }
```

XML

```
<value>
  <dataframes>
    <item>
      <begin>2015-04-22T07:00:00</begin>
      <end>2015-04-22T08:00:00</end>
      <tenant_id>69d12143688f413cbf5c3cfe03ed0a12</tenant_id>
      <resources>
        <item>
          <rating>1.0</rating>
          <service>compute</service>
          <desc>
            <item>
              <key>flavor</key>
              <value>m1.tiny</value>
            </item>
            <item>
              <key>vcpus</key>
              <value>1</value>
            </item>
          </desc>
          <volume>1.0</volume>
        </item>
      </resources>
    </item>
  </dataframes>
</value>
```

CHAPTER 5

Modules API

HashMap Module REST API

GET /v1/rating/module_config/hashmap/types
Return the list of every mapping type available.

Return type list(unicode)

GET /v1/rating/module_config/hashmap/services
Get the service list

Return List of every services.

Return type ServiceCollection

GET /v1/rating/module_config/hashmap/services/(*service_id*)
Return a service.

Parameters

- **service_id** (uuid) – UUID of the service to filter on.

Return type Service

POST /v1/rating/module_config/hashmap/services
Create hashmap service.

Parameters

- **service_data** (Service) – Informations about the service to create.

Return type Service

DELETE /v1/rating/module_config/hashmap/services
Delete the service and all the sub keys recursively.

Parameters

- **service_id** (uuid) – UUID of the service to delete.

type Service
Type describing a service.

A service is directly mapped to the usage key, the collected service.

Data samples:

Json

```
{
    "name": "compute",
    "service_id": "a733d0e1-1ec9-4800-8df8-671e4affd017"
}
```

XML

```
<value>
  <service_id>a733d0e1-1ec9-4800-8df8-671e4affd017</service_id>
  <name>compute</name>
</value>
```

name

Type unicode

Name of the service.

service_id

Type uuid

UUID of the service.

type ServiceCollection

Type describing a list of services.

Data samples:

Json

```
{
    "services": [
        {
            "name": "compute",
            "service_id": "a733d0e1-1ec9-4800-8df8-671e4affd017"
        }
    ]
}
```

XML

```
<value>
  <services>
    <item>
      <service_id>a733d0e1-1ec9-4800-8df8-671e4affd017</service_id>
      <name>compute</name>
    </item>
  </services>
</value>
```

services

Type list(Service)

List of services.

GET /v1/rating/module_config/hashmap/fields

Get the field list.

Parameters

- **service_id** (uuid) – Service's UUID to filter on.

Return List of every fields.

Return type FieldCollection

GET /v1/rating/module_config/hashmap/fields/ (*field_id*)
Return a field.

Parameters

- **field_id** (uuid) – UUID of the field to filter on.

Return type Field

POST /v1/rating/module_config/hashmap/fields
Create a field.

Parameters

- **field_data** (Field) – Informations about the field to create.

Return type Field

DELETE /v1/rating/module_config/hashmap/fields
Delete the field and all the sub keys recursively.

Parameters

- **field_id** (uuid) – UUID of the field to delete.

type Field
Type describing a field.

A field is mapping a value of the 'desc' dict of the CloudKitty data. It's used to map the name of a metadata.

Data samples:

Json

```
{
    "field_id": "ac55b000-a05b-4832-b2ff-265a034886ab",
    "name": "image_id",
    "service_id": "a733d0e1-1ec9-4800-8df8-671e4affd017"
}
```

XML

```
<value>
  <field_id>ac55b000-a05b-4832-b2ff-265a034886ab</field_id>
  <name>image_id</name>
  <service_id>a733d0e1-1ec9-4800-8df8-671e4affd017</service_id>
</value>
```

field_id

Type uuid

UUID of the field.

name

Type unicode

Name of the field.

service_id

Type uuid

UUID of the parent service.

type `FieldCollection`
Type describing a list of fields.

Data samples:

Json

```
{
    "fields": [
        {
            "field_id": "ac55b000-a05b-4832-b2ff-265a034886ab",
            "name": "image_id",
            "service_id": "a733d0e1-1ec9-4800-8df8-671e4affd017"
        }
    ]
}
```

XML

```
<value>
  <fields>
    <item>
      <field_id>ac55b000-a05b-4832-b2ff-265a034886ab</field_id>
      <name>image_id</name>
      <service_id>a733d0e1-1ec9-4800-8df8-671e4affd017</service_id>
    </item>
  </fields>
</value>
```

`fields`

Type list(Field)

List of fields.

`GET /v1/rating/module_config/hashmap/mappings`
Get the mapping list

Parameters

- **service_id** – Service UUID to filter on.
- **field_id** – Field UUID to filter on.
- **group_id** – Group UUID to filter on.
- **no_group** – Filter on orphaned mappings.
- **tenant_id** – Tenant UUID to filter on.
- **filter_tenant** – Explicitly filter on tenant (default is to not

filter on tenant). Useful if you want to filter on tenant being None. :return: List of every mappings. :type service_id: `uuid` :type field_id: `uuid` :type group_id: `uuid` :type no_group: `bool` :type tenant_id: `uuid` :type filter_tenant: `bool`

Return type `MappingCollection`

`GET /v1/rating/module_config/hashmap/mappings/` (*mapping_id*)
Return a mapping.

Parameters

- **mapping_id** (`uuid`) – UUID of the mapping to filter on.

Return type Mapping

POST /v1/rating/module_config/hashmap/mappings
Create a mapping.

Parameters

- **mapping_data** (Mapping) – Informations about the mapping to create.

Return type Mapping

PUT /v1/rating/module_config/hashmap/mappings
Update a mapping.

Parameters

- **mapping_id** (uuid) – UUID of the mapping to update.
- **mapping** (Mapping) – Mapping data to insert.

DELETE /v1/rating/module_config/hashmap/mappings
Delete a mapping.

Parameters

- **mapping_id** (uuid) – UUID of the mapping to delete.

GET /v1/rating/module_config/hashmap/mappings/group
Get the group attached to the mapping.

Parameters

- **mapping_id** (uuid) – UUID of the mapping to filter on.

Return type Group

type Mapping
Type describing a Mapping.

A mapping is used to apply rating rules based on a value, if the parent is a field then it's check the value of a metadata. If it's a service then it directly apply the rate to the volume.

Data samples:

Json

```
{
    "cost": "4.2",
    "field_id": "ac55b000-a05b-4832-b2ff-265a034886ab",
    "mapping_id": "39dbd39d-f663-4444-a795-fb19d81af136",
    "tenant_id": "7977999e-2e25-11e6-a8b2-df30b233ffcb",
    "type": "flat",
    "value": "m1.micro"
}
```

XML

```
<value>
  <mapping_id>39dbd39d-f663-4444-a795-fb19d81af136</mapping_id>
  <value>m1.micro</value>
  <type>flat</type>
  <cost>4.2</cost>
  <field_id>ac55b000-a05b-4832-b2ff-265a034886ab</field_id>
  <tenant_id>7977999e-2e25-11e6-a8b2-df30b233ffcb</tenant_id>
</value>
```

cost

Type Decimal

Value of the mapping.

field_id

Type uuid

UUID of the field.

group_id

Type uuid

UUID of the hashmap group.

map_type

Type Enum(flat, rate)

Type of the mapping.

mapping_id

Type uuid

UUID of the mapping.

service_id

Type uuid

UUID of the service.

tenant_id

Type uuid

UUID of the hashmap tenant.

value

Type unicode

Key of the mapping.

type MappingCollection

Type describing a list of mappings.

Data samples:

Json

```
{
    "mappings": [
        {
            "cost": "4.2",
            "field_id": "ac55b000-a05b-4832-b2ff-265a034886ab",
            "mapping_id": "39dbd39d-f663-4444-a795-fb19d81af136",
            "tenant_id": "7977999e-2e25-11e6-a8b2-df30b233ffcb",
            "type": "flat",
            "value": "m1.micro"
        }
    ]
}
```

XML

```
<value>
  <mappings>
    <item>
      <mapping_id>39dbd39d-f663-4444-a795-fb19d81af136</mapping_id>
      <value>m1.micro</value>
      <type>flat</type>
      <cost>4.2</cost>
      <field_id>ac55b000-a05b-4832-b2ff-265a034886ab</field_id>
      <tenant_id>7977999e-2e25-11e6-a8b2-df30b233ffcb</tenant_id>
    </item>
  </mappings>
</value>
```

mappings

Type list(Mapping)

List of mappings.

type Threshold

Type describing a Threshold.

A threshold is used to apply rating rules based on a level, if the parent is a field then the level is checked against a metadata. If it's a service then it's the quantity of the resource that is checked.

Data samples:

Json

```
{
    "cost": "4.2",
    "field_id": "ac55b000-a05b-4832-b2ff-265a034886ab",
    "level": "1024",
    "tenant_id": "7977999e-2e25-11e6-a8b2-df30b233ffcb",
    "threshold_id": "39dbd39d-f663-4444-a795-fb19d81af136",
    "type": "flat"
}
```

XML

```
<value>
  <threshold_id>39dbd39d-f663-4444-a795-fb19d81af136</threshold_id>
  <level>1024</level>
  <type>flat</type>
  <cost>4.2</cost>
  <field_id>ac55b000-a05b-4832-b2ff-265a034886ab</field_id>
  <tenant_id>7977999e-2e25-11e6-a8b2-df30b233ffcb</tenant_id>
</value>
```

cost

Type Decimal

Value of the threshold.

field_id

Type uuid

UUID of the field.

group_id

Type uuid

UUID of the hashmap group.

level

Type Decimal

Level of the threshold.

map_type

Type Enum(flat, rate)

Type of the threshold.

service_id

Type uuid

UUID of the service.

tenant_id

Type uuid

UUID of the hashmap tenant.

threshold_id

Type uuid

UUID of the threshold.

type ThresholdCollection

Type describing a list of mappings.

Data samples:

Json

```
{
    "thresholds": [
        {
            "cost": "4.2",
            "field_id": "ac55b000-a05b-4832-b2ff-265a034886ab",
            "level": "1024",
            "tenant_id": "7977999e-2e25-11e6-a8b2-df30b233ffcb",
            "threshold_id": "39dbd39d-f663-4444-a795-fb19d81af136",
            "type": "flat"
        }
    ]
}
```

XML

```
<value>
  <thresholds>
    <item>
      <threshold_id>39dbd39d-f663-4444-a795-fb19d81af136</threshold_id>
      <level>1024</level>
      <type>flat</type>
      <cost>4.2</cost>
      <field_id>ac55b000-a05b-4832-b2ff-265a034886ab</field_id>
      <tenant_id>7977999e-2e25-11e6-a8b2-df30b233ffcb</tenant_id>
    </item>
```

```
      </thresholds>
    </value>
```

thresholds

Type list(Threshold)

List of thresholds.

GET /v1/rating/module_config/hashmap/groups
Get the group list

Return List of every group.

Return type GroupCollection

GET /v1/rating/module_config/hashmap/groups/ (*group_id*)
Return a group.

Parameters

- **group_id** (uuid) – UUID of the group to filter on.

Return type Group

POST /v1/rating/module_config/hashmap/groups
Create a group.

Parameters

- **group_data** (Group) – Informations about the group to create.

Return type Group

DELETE /v1/rating/module_config/hashmap/groups
Delete a group.

Parameters

- **group_id** (uuid) – UUID of the group to delete.
- **recursive** (bool) – Delete mappings recursively.

GET /v1/rating/module_config/hashmap/groups/mappings
Get the mappings attached to the group.

Parameters

- **group_id** (uuid) – UUID of the group to filter on.

Return type MappingCollection

GET /v1/rating/module_config/hashmap/groups/thresholds
Get the thresholds attached to the group.

Parameters

- **group_id** (uuid) – UUID of the group to filter on.

Return type ThresholdCollection

type Group
Type describing a group.

A group is used to divide calculations. It can be used to create a group for the instance rating (flavor) and one if we have premium images (image_id). So you can take into account multiple parameters during the rating.

Data samples:

Json

```
{
    "group_id": "afe898cb-86d8-4557-ad67-f4f01891bbee",
    "name": "instance_rating"
}
```

XML

```
<value>
  <group_id>afe898cb-86d8-4557-ad67-f4f01891bbee</group_id>
  <name>instance_rating</name>
</value>
```

group_id

Type uuid

UUID of the group.

name

Type unicode

Name of the group.

type GroupCollection

Type describing a list of groups.

Data samples:

Json

```
{
    "groups": [
        {
            "group_id": "afe898cb-86d8-4557-ad67-f4f01891bbee",
            "name": "instance_rating"
        }
    ]
}
```

XML

```
<value>
  <groups>
    <item>
      <group_id>afe898cb-86d8-4557-ad67-f4f01891bbee</group_id>
      <name>instance_rating</name>
    </item>
  </groups>
</value>
```

groups

Type list(Group)

List of groups.

PyScripts Module REST API

GET /v1/rating/module_config/pyscripts/scripts
Get the script list

Parameters

- **no_data** (bool) – Set to True to remove script data from output.

Return List of every scripts.

Return type ScriptCollection

GET /v1/rating/module_config/pyscripts/scripts/ (*script_id*)
Return a script.

Parameters

- **script_id** (uuid) – UUID of the script to filter on.

Return type Script

POST /v1/rating/module_config/pyscripts/scripts
Create pyscripts script.

Parameters

- **script_data** (Script) – Informations about the script to create.

Return type Script

PUT /v1/rating/module_config/pyscripts/scripts
Update pyscripts script.

Parameters

- **script_id** (uuid) – UUID of the script to update.
- **script_data** (Script) – Script data to update.

Return type Script

DELETE /v1/rating/module_config/pyscripts/scripts
Delete the script.

Parameters

- **script_id** (uuid) – UUID of the script to delete.

type Script
Type describing a script.

Data samples:

Json

```
{
    "checksum": "da39a3ee5e6b4b0d3255bfef95601890afd80709",
    "data": "return 0",
    "name": "policy1",
    "script_id": "bc05108d-f515-4984-8077-de319cbf35aa"
}
```

XML

```
<value>
  <script_id>bc05108d-f515-4984-8077-de319cbf35aa</script_id>
  <name>policy1</name>
  <data>return 0</data>
  <checksum>da39a3ee5e6b4b0d3255bfef95601890afd80709</checksum>
</value>
```

checksum

Type unicode

Checksum of the script data.

data

Type unicode

Data of the script.

name

Type unicode

Name of the script.

script_id

Type uuid

UUID of the script.

type ScriptCollection

Type describing a list of scripts.

Data samples:

Json

```
{
    "scripts": [
        {
            "checksum": "da39a3ee5e6b4b0d3255bfef95601890afd80709",
            "data": "return 0",
            "name": "policy1",
            "script_id": "bc05108d-f515-4984-8077-de319cbf35aa"
        }
    ]
}
```

XML

```
<value>
  <scripts>
    <item>
      <script_id>bc05108d-f515-4984-8077-de319cbf35aa</script_id>
      <name>policy1</name>
      <data>return 0</data>
      <checksum>da39a3ee5e6b4b0d3255bfef95601890afd80709</checksum>
    </item>
  </scripts>
</value>
```

scripts

Type list(Script)

List of scripts.

CHAPTER 6

Rating Module Documentation

Rating module introduction

There are three rating modules in Cloudkitty now, including the `noop`, `hashmap` and `pyscripts`. Only the noop rating module is just for testing. All modules can be enabled and disabled dynamically. Cloudkitty allows to run several rating modules simultaneously, and the user or operator can set the priority for a module. The order in which the modules process the data depends on their priority. The module with the highest priority comes first.

List current modules

List current rating modules:

```
$ cloudkitty module-list
+-----------+---------+----------+
| Module    | Enabled | Priority |
+-----------+---------+----------+
| hashmap   | False   | 1        |
| noop      | True    | 1        |
| pyscripts | True    | 1        |
+-----------+---------+----------+
```

Enable or disable module

Enable the hashmap rating module:

```
$ cloudkitty module-enable -n hashmap
+---------+---------+----------+
| Module  | Enabled | Priority |
+---------+---------+----------+
| hashmap | True    | 1        |
+---------+---------+----------+
```

Disable the pyscripts rating module:

```
$ cloudkitty module-disable -n pyscripts
+-----------+---------+----------+
| Module    | Enabled | Priority |
+-----------+---------+----------+
| pyscripts | False   | 1        |
+-----------+---------+----------+
```

Set priority

Set the hashmap rating module priority to 100:

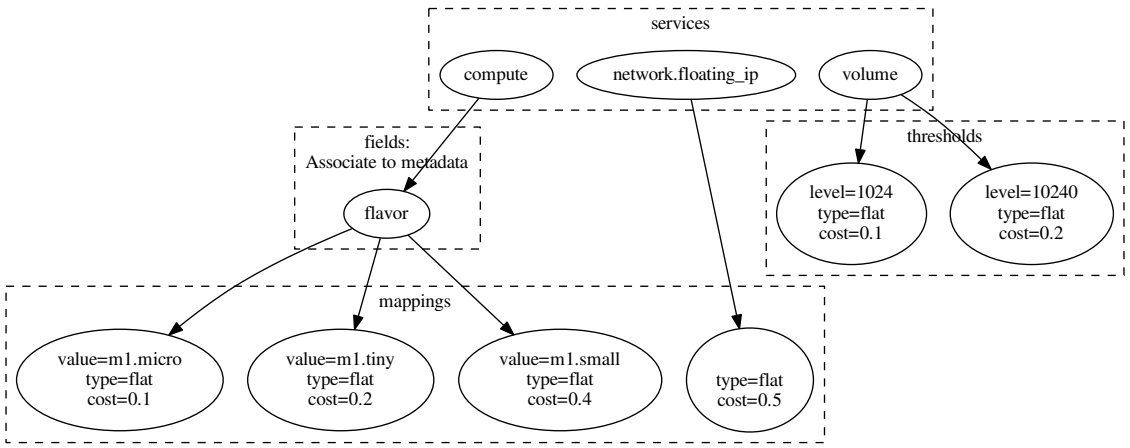
```
$ cloudkitty module-set-priority -n hashmap -p 100
+---------+---------+----------+
| Module  | Enabled | Priority |
+---------+---------+----------+
| hashmap | True    | 100      |
+---------+---------+----------+
```

Hashmap rating module

CloudKitty is shipped with core rating modules.

Hashmap composition

You can see hashmap as a simple tree:



HashMap module data structure

HashMap is composed of different resources and groups.

Group

A group is a way to group calculations of mappings. For example you might want to apply a set of rules to charge instance_uptime and another set to block storage volume. You don't want the two to be linked so you'll create one group for each calculation.

Service

A service is a way to map the rule to the type of data collected. Currently, the following services are available:

- compute

- image
- volume
- network.bw.in
- network.bw.out
- network.floating

Enabled services are defined in the configuration file. By default, only the compute service is enabled.

Field

A field is referring to a metadata field of a resource. For example on an instance object (**compute**), you can use the flavor to define specific rules.

With Ceilometer as the collector, the following fields are available for each service:

- Compute: flavor, vcpus, memory (MB), image_id, availability_zone
- Volume: name, volume_type, availability_zone
- Image: disk_format, container_format, is_public, availability_zone

With Gnocchi as collector, the following fields are available for each service:

- Compute: flavor_id, vcpus, image_id, memory (MB)
- Image: container_format, disk_format

Mapping

A mapping is the final object, it's what triggers calculation, for example a specific value of flavor on an instance. It maps cost to a value of metadata in case of field mapping. And directly a cost in case of service mapping.

A mapping can be project specific by providing a project id at creation and supports overloading, i.e. you can specify multiple mappings for the same value with different project ids and costs.

Threshold

A threshold entry is used to apply rating rules base on level. Its behaviour is similar to a mapping except that it applies the cost base on the level.

As for mapping, a threshold can be project specific by providing a project id at creation.

HashMap formula

Based on all the previous objects here's the calculation formula : $\sum_{n=1}^{N} G_n(qty.(T_{rate} \prod(M_{rate})(T_{flat} + M_{flat})))$

G Group

qty Quantity of resource

T Threshold

M Mapping

For an active resource on a collection period, quantity is defined as follow:

- compute: 1 (unit: instance)

- image: upload image size (unit: MB)
- volume: volume size (unit: GB)
- network.bw.in: ingoing network usage (unit: MB)
- network.bw.out: outgoing network usage (unit: MB)
- network.floating: 1 (unit: ip)

Example

Compute uptime

Apply rating rule on the compute service to charge the instance based on it's flavor and uptime:

Create a group *instance_uptime_flavor*:

```
$ cloudkitty hashmap-group-create -n instance_uptime_flavor
+----------+--------------------------------------+
| Property | Value                                |
+----------+--------------------------------------+
| group_id | 26d2d69a-4c42-47f1-9d44-2cdfad167f7d |
| name     | instance_uptime_flavor               |
+----------+--------------------------------------+

$ cloudkitty hashmap-group-list
+------------------------+--------------------------------------+
| Name                   | Group id                             |
+------------------------+--------------------------------------+
| instance_uptime_flavor | 26d2d69a-4c42-47f1-9d44-2cdfad167f7d |
+------------------------+--------------------------------------+
```

Create the service matching rule:

```
$ cloudkitty hashmap-service-create -n compute
+------------+--------------------------------------+
| Property   | Value                                |
+------------+--------------------------------------+
| name       | compute                              |
| service_id | 08ab2d27-fe95-400c-9602-e5ad5efdda8b |
+------------+--------------------------------------+
```

Create a field matching rule:

```
$ cloudkitty hashmap-field-create \
 -s 08ab2d27-fe95-400c-9602-e5ad5efdda8b -n flavor
+------------+--------------------------------------+
| Property   | Value                                |
+------------+--------------------------------------+
| field_id   | f37364af-6525-40fc-ae08-6d4087429862 |
| name       | flavor                               |
| service_id | 08ab2d27-fe95-400c-9602-e5ad5efdda8b |
+------------+--------------------------------------+
```

Create a mapping in the group *instance_uptime_flavor* that will map m1.tiny instance to a cost of 0.01:

```
$ cloudkitty hashmap-mapping-create \
 -f f37364af-6525-40fc-ae08-6d4087429862 \
 -v m1.tiny -t flat -c 0.01 -g 26d2d69a-4c42-47f1-9d44-2cdfad167f7d
```

```
+------------+--------------------------------------+
| Property   | Value                                |
+------------+--------------------------------------+
| cost       | 0.01                                 |
| field_id   | f37364af-6525-40fc-ae08-6d4087429862 |
| group_id   | 26d2d69a-4c42-47f1-9d44-2cdfad167f7d |
| mapping_id | df592a91-a6a5-41fa-ba2e-2f763eaa36e5 |
| service_id | None                                 |
| tenant_id  | None                                 |
| type       | flat                                 |
| value      | m1.tiny                              |
+------------+--------------------------------------+
```

In this example every machine in any project with the flavor m1.tiny will be charged 0.01 per collection period.

Volume per gb with discount

Now let's do some threshold based rating.

Create a group *volume_thresholds*:

```
$ cloudkitty hashmap-group-create -n volume_thresholds
+----------+--------------------------------------+
| Property | Value                                |
+----------+--------------------------------------+
| group_id | dd3dc30e-0e63-11e6-9f83-ab4208c1fe2d |
| name     | volume_thresholds                    |
+----------+--------------------------------------+

$ cloudkitty hashmap-group-list
+-------------------+--------------------------------------+
| Name              | Group id                             |
+-------------------+--------------------------------------+
| volume_thresholds | dd3dc30e-0e63-11e6-9f83-ab4208c1fe2d |
+-------------------+--------------------------------------+
```

Create the service matching rule:

```
$ cloudkitty hashmap-service-create -n volume
+------------+--------------------------------------+
| Property   | Value                                |
+------------+--------------------------------------+
| name       | volume                               |
| service_id | 16a48060-0e64-11e6-8e4e-1b285514a36e |
+------------+--------------------------------------+
```

Now let's setup the price per gigabyte:

```
$ cloudkitty hashmap-mapping-create \
 -s 16a48060-0e64-11e6-8e4e-1b285514a36e \
 -t flat -c 0.001 -g dd3dc30e-0e63-11e6-9f83-ab4208c1fe2d
+------------+--------------------------------------+
| Property   | Value                                |
+------------+--------------------------------------+
| cost       | 0.001                                |
| field_id   | None                                 |
| group_id   | dd3dc30e-0e63-11e6-9f83-ab4208c1fe2d |
| mapping_id | 41669786-240b-11e6-872c-af96ddb6619c |
| service_id | 16a48060-0e64-11e6-8e4e-1b285514a36e |
```

```
| tenant_id  | None                                 |
| type       | flat                                 |
| value      |                                      |
+------------+--------------------------------------+
```

We have the basic price per gigabyte be we now want to apply a discount on huge data volumes. Create the thresholds in the group *volume_thresholds* that will map different volume quantity to costs:

Here we set a threshold when going past 50GB, and apply a 2% discount (0.98):

```
$ cloudkitty hashmap-threshold-create \
 -s 16a48060-0e64-11e6-8e4e-1b285514a36e \
 -l 50 -t rate -c 0.98 -g dd3dc30e-0e63-11e6-9f83-ab4208c1fe2d
+--------------+--------------------------------------+
| Property     | Value                                |
+--------------+--------------------------------------+
| cost         | 0.98                                 |
| field_id     | None                                 |
| group_id     | dd3dc30e-0e63-11e6-9f83-ab4208c1fe2d |
| level        | 50                                   |
| threshold_id | 8eb45bfc-0e64-11e6-ad0e-07a62425f284 |
| service_id   | 16a48060-0e64-11e6-8e4e-1b285514a36e |
| tenant_id    | None                                 |
| type         | rate                                 |
+--------------+--------------------------------------+
```

Here we set the same threshold for project 8f1e8645a0e7496a95a4fdf4b2795b2c but with a 3% discount (0.97):

```
$ cloudkitty hashmap-threshold-create \
 -s 16a48060-0e64-11e6-8e4e-1b285514a36e \
 -l 50 -t rate -c 0.98 -g dd3dc30e-0e63-11e6-9f83-ab4208c1fe2d \
 -p 8f1e8645a0e7496a95a4fdf4b2795b2c
+--------------+--------------------------------------+
| Property     | Value                                |
+--------------+--------------------------------------+
| cost         | 0.97                                 |
| field_id     | None                                 |
| group_id     | dd3dc30e-0e63-11e6-9f83-ab4208c1fe2d |
| level        | 50                                   |
| threshold_id | 8eb45bfc-0e64-11e6-ad0e-07a62425f284 |
| service_id   | 16a48060-0e64-11e6-8e4e-1b285514a36e |
| tenant_id    | 8f1e8645a0e7496a95a4fdf4b2795b2c     |
| type         | rate                                 |
+--------------+--------------------------------------+
```

Here we set a threshold when going past 200GB, and apply a 5% discount (0.95):

```
$ cloudkitty hashmap-threshold-create \
 -s 16a48060-0e64-11e6-8e4e-1b285514a36e \
 -l 200 -t rate -c 0.95 -g dd3dc30e-0e63-11e6-9f83-ab4208c1fe2d
+--------------+--------------------------------------+
| Property     | Value                                |
+--------------+--------------------------------------+
| cost         | 0.95                                 |
| field_id     | None                                 |
| group_id     | dd3dc30e-0e63-11e6-9f83-ab4208c1fe2d |
| level        | 200                                  |
| threshold_id | baf180c8-0e64-11e6-abb3-cbae153a6d44 |
| service_id   | 16a48060-0e64-11e6-8e4e-1b285514a36e |
| tenant_id    | None                                 |
```

```
| type          | rate                                   |
+---------------+----------------------------------------+
```

In this example every volume is charged 0.01 per GB but if the size goes past 50GB you'll get a 2% discount, if you even go further you'll get 5% discount (only one level apply at a time).

For project 8f1e8645a0e7496a95a4fdf4b2795b2c only, you'll get a 3% discount instead of 2% when the size goes past 50GB and the same %5 discount it it goes further.

20GB 0.02 per collection period.

50GB 0.049 per collection period (0.0485 for project 8f1e8645a0e7496a95a4fdf4b2795b2c).

80GB 0.0784 per collection period (0.0776 for project 8f1e8645a0e7496a95a4fdf4b2795b2c).

250GB 0.2375 per collection period.

PyScripts rating module

The PyScripts module allows you to create your own rating module. A script is supposed to process the given data and to set the different prices.

CAUTION: If you add several PyScripts, the order in which they will be executed is not guaranteed.

Custom module example

Price definitions

```
import decimal


# Price for each flavor. These are equivalent to hashmap field mappings.
flavors = {
    'm1.micro': decimal.Decimal(0.65),
    'm1.nano': decimal.Decimal(0.35),
    'm1.large': decimal.Decimal(2.67)
}

# Price per MB / GB for images and volumes. These are equivalent to
# hashmap service mappings.
image_mb_price = decimal.Decimal(0.002)
volume_gb_price = decimal.Decimal(0.35)
```

Price calculation functions

```
# These functions return the price of a service usage on a collect period.
# The price is always equivalent to the price per unit multiplied by
# the quantity.
def get_compute_price(item):
    if not item['desc']['flavor'] in flavors:
        return 0
    else:
        return (decimal.Decimal(item['vol']['qty'])
                * flavors[item['desc']['flavor']])
```

```
def get_image_price(item):
    if not item['vol']['qty']:
        return 0
    else:
        return decimal.Decimal(item['vol']['qty']) * image_mb_price


def get_volume_price(item):
    if not item['vol']['qty']:
        return 0
    else:
        return decimal.Decimal(item['vol']['qty']) * volume_gb_price

# Mapping each service to its price calculation function
services = {
    'compute': get_compute_price,
    'volume': get_volume_price,
    'image': get_image_price
}
```

Processing the data

```
def process(data):
    # The 'data' parameter is a list of dictionaries containing a
    # "usage" and a "period" field
    for d in data:
        usage = d['usage']
        for service_name, service_data in usage.items():
            # Do not calculate the price if the service has no
            # price calculation function
            if service_name in services.keys():
                # A service can have several items. For example,
                # each running instance is an item of the compute service
                for item in service_data:
                    item['rating'] = {'price': services[service_name](item)}
    return data


# 'data' is passed as a global variable. The script is supposed to set the
# 'rating' element of each item in each service
data = process(data)
```

Using your Script for rating

Enabling the PyScripts module

To use you script for rating, you will need to enable the pyscripts module

```
$ cloudkitty module-enable -n pyscripts
+-----------+---------+
| Module    | Enabled |
+-----------+---------+
| pyscripts | True    |
+-----------+---------+
```

Adding the script to CloudKitty

Create the script and specify its name.

```
$ cloudkitty pyscripts-script-create -n my_awesome_script -f script.py
+-----------+-----------------------------------------------+
| Property  | Value                                         |
+-----------+-----------------------------------------------+
| checksum  | 7650349ba3a913cef526dfb953575042ec3332e1      |
| data      | from __future__ import print_function         |
|           | from cloudkitty import rating                 |
|           |                                               |
|           | import decimal                                |
|           |                                               |
|           |       {...}                                   |
|           |                                               |
|           | data = process(data)                          |
|           |                                               |
| name      | my_awesome_script                             |
| script_id | 93de054f-9d6f-40b0-8e4f-9b8fee8cad04          |
+-----------+-----------------------------------------------+
```

CHAPTER 7

Indices and tables

- *genindex*
- *modindex*
- *search*

/v1