Blueprint: Provide support for auditing events in standardized formats

From the onset, Ceilometer's goal was:

> *"to become the infrastructure to collect measurements within OpenStack" with its "primary targets [being] are monitoring and metering". However it was noted that its framework "should be easily expandable to collect for other needs. To that effect, Ceilometer should be able to share collected data with a variety of consumers".*

It is clear that core project developers appreciate the strengths of Ceilometer in having a reliable, core centralized service with the ability to track usage information towards statistical usage analysis and billing. It seems that many of these same projects are seeing similar "auditing" requirements but now with the emphasis on tracking access to services (by users and other services) for the purposes of security auditing. Leveraging Ceilometer's design to enable different types of event auditing standards (and to provide for different purposes and views on events) seems to make good sense.

Proposal Background

In the auditing and compliance space, standards are being developed for "Cloud Audit" that are designed to align with international and industry requirements yet address "clouds" unique deployment and service models. These standards recognize the challenge of auditing workloads in all types of cloud deployments while providing the data required for regulatory compliance (e.g. PCI-DSS, SoX, ISO 27017, etc.), as well as for proving adherence to localized customer security policies. One of these standards is the Distributed Mgmt. Task Force's (DMTF) "Cloud Audit" standard which provides a standardized format (in JSON) and a RESTful query syntax

DMTF Cloud Audit Standard Resources

The last public draft can be found here: http://dmtf.org/sites/default/files/standards/documents/DSP0262_1.0.0a_0.pdf. A "final" v1.0 working draft is due out May 2013 with hopes of being confirmed as committee spec. by end of June.

The specification is designed to address a wide range of auditing use cases as published here: http://www.dmtf.org/sites/default/files/standards/documents/DSP2028_1.0.0a.pdf

Proposal

This proposal seeks to leverage the design (agents/meters, etc.) and logging facilities already in Ceilometer and enable users to configure additional meters to allow event data to be produced in one or more (standard) formats. The configuration may also provide the location as to where the logs are created for each meter type (e.g. DMTF format at …/audit/dmtf/xxx).

We would like to add support for auditing APIs access using the DMTF format as a reference implementation as it provides normative data for the key "auditor questions" needed for any

compliance regime known in compliance circles as the “7 Ws” and also provides guidance on classifying data by resource, correlation of events (across transactions and service layers) and guidance for federation/merging with other logs (e.g. use UUIDs, normalized timestamps). The format also supports optionally providing geolocation information (per ISO 27017 standards) and Metric/Measurement information that aligns with the developing NIST cloud model/standard.

Below is a sample of a representative DMTF format mapping to show it provides data similar to that in Ceilometer today, but also know that it provides for extended fields/structured data for many of the target regulatory standards mentioned standards above.

In this (Cinder, volume) example, since a “metric” and “measurement” are carried these fields appear in the DMTF format.

Sample CADF Mapping of Ceilometer: Volume (Cinder) Data

Ceilometer

```
{
  'counter_name': 'volume',
  'counter_type': 'gauge',
  'counter_unit': 'volume',
  'counter_volume': 1,
  'message_id': '9ae6a15a-9c6b-11e2-9478-080027317d13',
  'project_id': '6c97f1ecf17047eab696786d56a0bff5',
  'resource_id': '84c363b9-9854-48dc-b949-fe04263f4cf0',
  'resource_metadata': {
    'display_name': 'volume1',
    'event_type': 'volume.exists',
    'host': 'volume.ubuntu-VirtualBox',
    'size': 2,
    'status': 'available',
    'volume_type': None
  },
  'source': 'openstack', # currently hardcoded
  'timestamp': '2012-09-21 09:29:10.620731',
  'user_id': '4d2fa4b76a4a4ecab8c468c8dea42f89'
}
```

Color Code / Meaning
Direct Mapping to CADF
Mapping using CADF “Attachment” Extension
Mapping using CADF “Tag” Extension
CADF Specific value (needed)

CADF

```
{
 'typeURI': 'http://schemas.dmtf.org/cloud/audit/1.0/event',
 'id': '<message_id>',
 'eventType': 'monitor,
 'eventTime': '<timestamp in UTC>',
 'action': 'monitor.guage',
 'outcome': 'success',
 'initiator': {  'id': '<agent_id>',
                 'typeURI': 'service.bss.metering'
                 'credential': {'user_id': '<user_id>' }
              },
 'target': { 'id': '84c363b9-9854-48dc-b949-fe04263f4cf0'
             'typeURI': 'storage.<counter_name>'
             'attachments': [ { 'name': 'resource-metadata',
                                'contentType': '<ceilometer volume format>',
                                'content': { 'display_name': 'volume1',
                                             'event_type': 'volume.exists',
                                             'host': 'volume.ubuntu-VirtualBox',  ..., }}
             ],
},
'tags': { "source?<source>", 'project_id?<project_id>' },
'reporterchain': [ { 'role': 'observer',
                     'reporterId': 'initiator' }
                   ],
'measurements': {
 'measurement': {
  'result': '<counter_volume>',
  'metric': {
    'metricId': '<metric_id>'
    'unit': '<counter_unit>'
    'name': '<counter_name>'
   }
 }
}
```

For auditing security and access control (e.g. for APIs calls) the CADF format may not contain any measurements (optional), but would include all the information needed for auditing (i.e. see the 7 “Ws” listed in the table below) such as information about the account (user). The reference/use of the DMTF standard format, which is designed for all types of cloud auditing, brings other forward- thinking benefits. For example, if OpenStack services scaled across regional boundaries, geolocation and other

detailed resource information could optionally be tracked using the CADF standard. Note that this is just an example benefit the standard could offer for the future.

CADF Model is designed to answer all Audit and Compliance Questions

How CADF standard expresses the 7 "W"s of Audit and Compliance

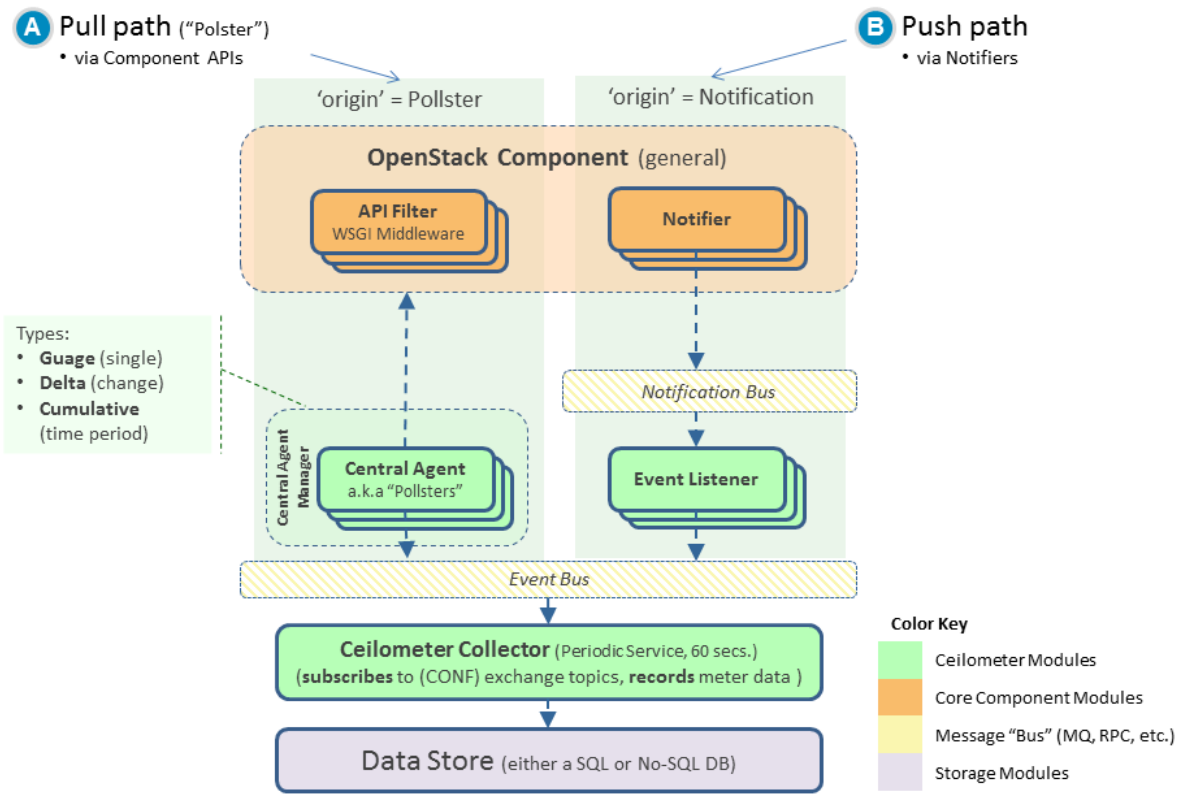
"W" Component	CADF Mandatory Component	CADF Optional Components (where applicable)	Comment
What	event.action event.outcome	event.reason (e.g. severity, reason code, policy id)	"what" activity occurred; "what" was the result
When	event.timestamp	reporter.timestamp (for each reporter that modifies the record)	"when" did it happen
Who	initiator.id initiator.type	Initiator.user (basic user) initiator.account (detailed, incl. "sudo" tracking) initiator.credentials (e.g. tokens)	"who" (person or service) initiated the action
OnWhat	target.id target.type		"onWhat" resource did the activity target
Where	reporter.id reporter.type		"where" did the activity get (observed) logged
FromWhere		initiator.endpoint (basic) initiator.connection (detailed) Initiator.geolocation (precise)	
ToWhere		target.endpoint (basic) target.connection (detailed) target.geolocation (precise)	

CADF provides methods to "Extend" the format to carry domain-specific information

CADF Extension Type	CADF Optional Component	Purpose
Attachment	event.attachments	For adding domain specific structured or unstructured data. If structured, the type can be supplied and referenced by the CADF Query API.
Tags	event.tags	For adding domain-specific identifiers and classifications that enable domain specific identification and can be used with the CADF Query API to construct custom reports.

Currently, there are two established methods for automatically sending data to Ceilometer (as shown below).

Mechanisms for getting Component Data into Ceilometer



This blueprint proposes leveraging the existing Ceilometer design to establish a configurable audit filter/notification path to the Ceilometer collector that would use the CADF format (initially carried within the payload). The following diagram shows that each component that would require configurable API audit would derive a filter from a common audit filter based upon CADF, validated by an audit notifier which would use the existing common notification method and register a new

“audit.api” event type which would be handled by a matching event filter for that type.

Changes needed to enable Ceilometer to collect “API Audit” CADF Data

Audit Data needs to be “pushed” through notification path (no delay)