HP Helion Openstack 2.0: Monasca Overview

Contents

HP Helion Openstack® 2.0: Monasca Overview

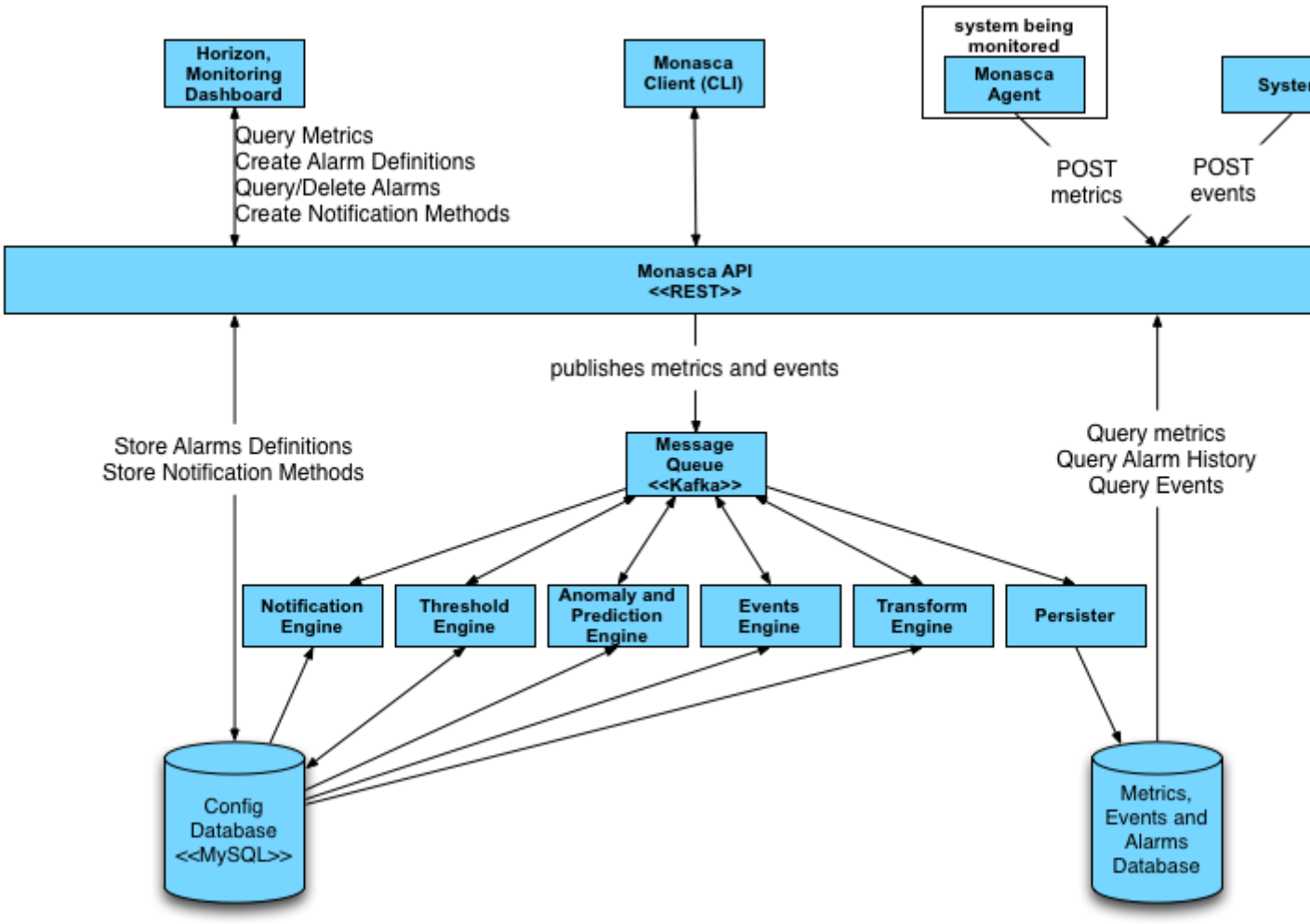
Monasca is a multi-tenant, scalable, fault-tolerant monitoring-as-a-service, HTTP-based solution. It is open-source and integrates with Openstack®. It uses a REST API for high-speed metrics processing and querying, and has a streaming alarm and notification engine. The name "Monasca" is short for "monitoring at scale".

Monasca components consist of those that are part of the monitor server cluster and those that interact only with the Monasca API.

The Monasca agent is highly configurable and can collect measurements from many sources (unlike monitors like Icinga, which only monitor the Swift cluster). Monasca can even monitor itself. This self monitoring is known as Monitoring of Monasca or MoM.

Metrics are submitted and authenticated using Keystone and are stored associated with a tenant ID.

Monasca checks for services such as MySQL, RabbitMQ, and many others, and also supports Nagios checks and StatsD.



Monasca components

Monasca consists of:

- Monitoring Agent (monasca-agent): A modern Python based monitoring agent that consists of several sub-components and supports system metrics, such as cpu utilization and available memory, Nagios plugins, StatsD and many built-in checks for services such as MySQL, RabbitMQ, and others.
- Monitoring API (monasca-api): A well-defined and documented RESTful API for monitoring primarily on the following concepts and areas:
  - Metrics: Store and query massive amounts of metrics in real-time.
  - Statistics: Query statistics for metrics.
  - Alarm Definitions: Create, update, query and delete alarm definitions.
  - Alarms: Query and delete the alarm history.
    - Simple expressive grammar for creating compound alarms composed of alarm subexpressions and logical operators.
    - Alarm severities can be associated with alarms.
    - The complete alarm state transition history is stored and queryable which allows for subsequent root cause analysis (RCA) or advanced analytics.
  - Notification Methods: Create and delete notification methods and associate them with alarms, such as email. Supports the ability to notify users directly via email when an alarm state transitions occur.
- Message Queue: A third-party component that primarily receives published metrics from the Monitoring API and alarm state transition messages from the Threshold Engine that are consumed by other components, such as the Persister and Notification Engine. The Message Queue is also used to publish and consume other events in the system. Currently, a Kafka based MessageQ is supported. Kafka is a high performance, distributed, fault-tolerant, and scalable message queue with durability built-in. We will look at other alternatives, such as RabbitMQ and in-fact in our previous implementation RabbitMQ was supported, but due to performance, scale, durability and high-availability limitiations with RabbitMQ we have moved to Kafka. See Monasca Messaging for Message Queue details.
- Notification Engine (monasca-notification): Consumes alarm state transition messages from the MessageQ and sends notifications, such as emails for alarms. The Notification Engine is Python based.
- Threshold Engine (monasca-thresh): Computes thresholds on metrics and publishes alarms to the MessageQ when exceeded. Based on Apache Storm a free and open distributed real-time computation system.
- Anomaly and Prediction Engine: Evaluates prediction and anomalies and generates predicted metrics as well as anomaly likelihood and anomaly scores.
- Transform and Aggregation Engine: Transform metric names and values, such as delta or time-based derivative calculations, and creates new metrics that are published to the Message Queue. The Transform Engine is not available yet.
- Events Engine [[need description]]
- Persister (monasca-persister): Consumes metrics and alarm state transitions from the MessageQ and stores them in the Metrics and Alarms database. We will look into converting the Persister to a Python component in the future.
- Metrics and Alarms Database: A third-party component that primarily stores metrics and the alarm state history. Currently, Vertica and InfluxDB are supported.
- Config Database: A third-party component that stores a lot of the configuration and other information in the system. Currently, MySQL is supported.
- Monitoring Client (python-monascaclient): A Python command line client and library that communicates and controls the Monitoring API. The Monitoring Client was written using the Openstack® Heat Python client as a framework. The Monitoring Client also has a Python library, "monascaclient" similar to the other Openstack® clients, that can be used to quickly build additional capabilities. The Monitoring Client library is used by the Monitoring UI, Ceilometer publisher, and other components.
- Alarm Configuration Manager: A Python process that will detect new metrics and configure alarms based on the configuration. It uses the monitoring client library that communicates with the Monitoring API. The Alarm Configuration Manager is a Python Daemon that runs on a configurable interval and detects new metrics that need to be alarmed and creates the alarms.
- Monitoring UI: A Horizon dashboard for visualizing the overall health and status of an Openstack® cloud.
- Ceilometer publisher: A multi-publisher plugin for Ceilometer, not shown, that converts and publishes samples to the Monitoring API.

Monasca messaging

The Monasca Message Queue (MessageQ) is a distributed, scalable, highly available message queue for distributing metrics, alarms and events in the monitoring system. The message queue is based on Kafka.

There are several messages that are published and consumed by various components in Monasca via the MessageQ. For details, see:

Message Schema https://wiki.openstack.org/wiki/Monasca/Message_Schema

Metrics and Alarms Database

A high-performance, analytics database that can store massive amounts of metrics and alarms in real-time and also support interactive queries. Currently Vertica and InfluxDB are supported.

Config Database

The config database is MySQL-based and stores all configuration information.

Events

Support for real-time event stream processing in Monasca is in progress. For more details see the link at, Monasca/Events.

Logging

Support for logging in Monasca is under discussion. For more details see the link at, Monasca > Logging.

Monitoring

Enablement and usage for monitoring the status of Monasca is under discussion. For more details see the link at, Monasca > Monitoring_Of_Monasca

Value Metadata

Adding Metadata to the value of a measurement in Monasca is under discussion. For more details see the link at, Monasca/Value_Metadata

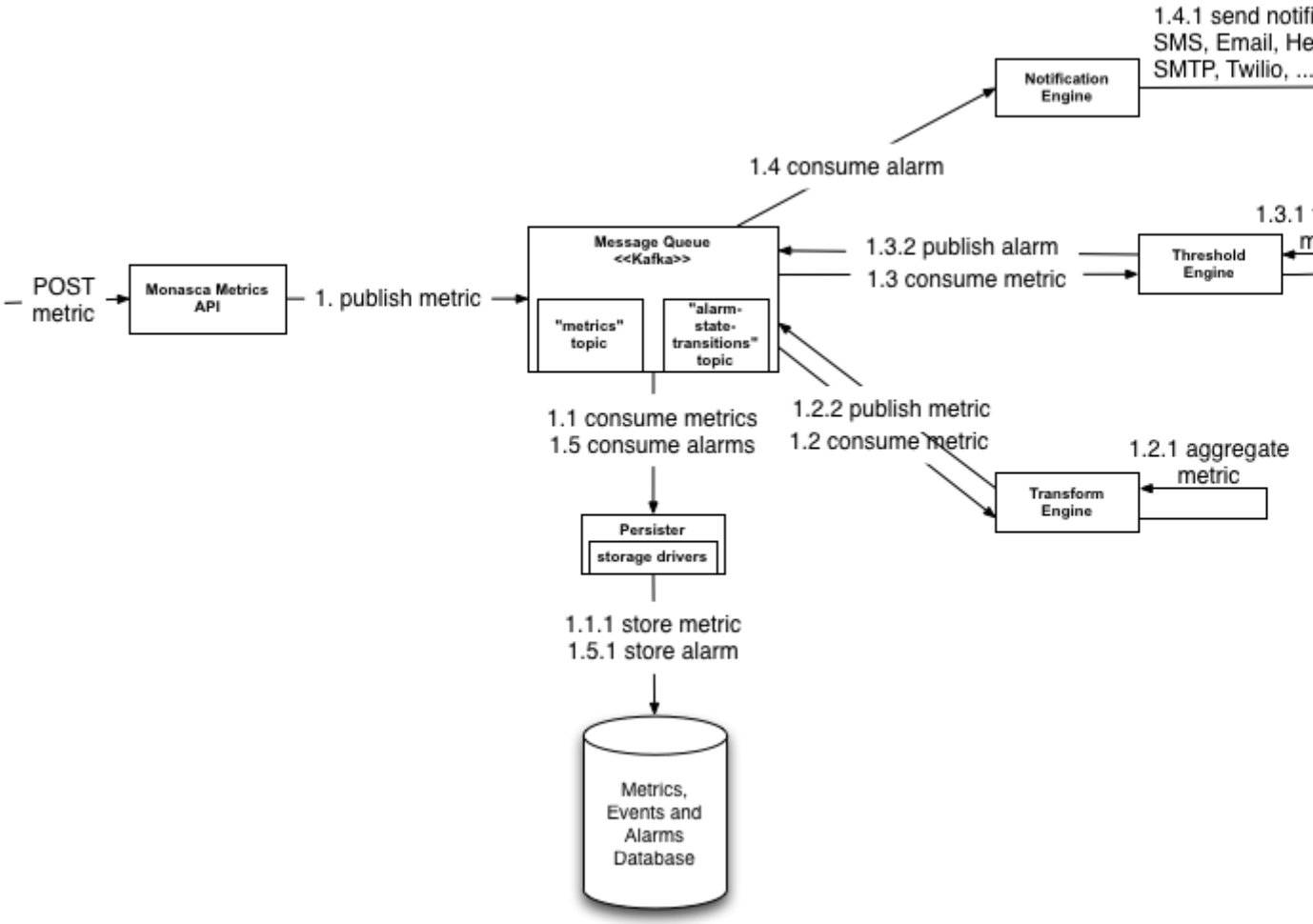
Keystone Requirements

Monasca relies on Keystone and there are requirements about which Keystone configuration must exist.

The endpoint for the API must be registered in Keystone as the 'monasca' service. The API must have an admin token to use in verifying the Keystone tokens it receives. For each project using Monasca, two users must exist: one will be in the 'monasca-agent' role and be used by the monasca-agent's running on machines. The other should not be in that role and can be used logging into the UI, using the CLI or for direct queries against the API.

Post Metric Sequence

This section describes the sequence of operations involved in posting a metric to the Monasca API.



The illustration above shows a metric being posted to the Monasca API. The Monasca API authenticates and validates the request and publishes the metric to the Message Queue. The Persister consumes the metric from the Message Queue and stores it in the Metrics Store. The Transform Engine consumes the metrics from the Message Queue, performs transform and aggregation operations on metrics, and publishes metrics that it creates back to Message Queue. The Threshold Engine consumes metrics from the Message Queue and evaluates alarms. If a state change occurs in an alarm, an "alarm-state-transitioned-event" is published to the Message Queue. The Notification Engine consumes "alarm-state-transitioned-events" from the Message Queue, evaluates whether they have a Notification Method associated with it, and sends the appropriate notification, such as email. The Persister consumes the "alarm-state-transitioned-event" from the Message Queue and stores it in the Alarm State History Store.

Technologies

Monasca relies on such underlying technologies as:

- Apache Kafka (http://kafka.apache.org): Apache Kafka is publish-subscribe messaging process delivered as a distributed commit log. Kafka is a highly performing, distributed, fault-tolerant, and scalable message queue.
- Apache Storm (http://storm.incubator.apache.org/): Apache Storm is a free and open source distributed real-time computation system. Storm makes it easy to reliably process unbounded streams of data, doing for real-time processing what Hadoop did for batch processing.
- ZooKeeper (http://zookeeper.apache.org/): Used by Kafka and Storm.
- MySQL:
- • Vagrant (http://www.vagrantup.com/): Vagrant provides easy to configure, reproducible, and portable work environments built on top of industry-standard technology and controlled by a single consistent work flow to help maximize the productivity and flexibility of you and your team.

- Dropwizard (https://dropwizard.github.io/dropwizard/): Dropwizard pulls together stable, mature libraries from the Java ecosystem into a simple, light-weight package that lets you focus on getting things done. Dropwizard has out-of-the-box support for sophisticated configuration, application metrics, logging, operational tools, and much more, allowing you and your team to ship a production-quality web service in the shortest time possible.
- InfluxDB (http://influxdb.com/): An open-source distributed time series database with no external dependencies.
- Vertica (http://www.vertica.com): A commercial Enterprise class SQL analytics database that is highly scalable. It offers built-in automatic high-availability and excels at in-database analytics and compressing and storing massive amounts of data. In the HP Public Cloud we use Vertica in a number of areas such as metrics and many other data streams. Currently, we process around 25 K metrics/sec and store them for > 13 month data retention periods. A free version of Vertica that can store up to 1 TB of data with no time-limit is available at, https://my.vertica.com/community/.

Return to Top