FUJITSU

FUJITSU Software
Cloud Monitoring Manager V2.0.14

System Operator's Guide

July 2021

Contents

About this Manual

This manual describes how system operators can install, operate, maintain, and monitor FUJITSU Software Cloud Monitoring Manager - hereafter referred to as Cloud Monitoring Manager (CMM).

The manual is structured as follows:

Chapter	Description
Introduction to CMM	Introduces CMM, its architecture and users.
Installation	Describes how to install and configure CMM.
Preparations for Application Operators	Describes how to prepare the monitoring environment for application operators.
Operation and Maintenance	Describes the main operation and maintenance tasks for CMM.
Monitoring	Describes the basic tasks involved in monitoring services and servers.
Log Management	Describes the basic tasks involved in managing the log data from the services and servers.
Glossary	Defines the central terms relevant for CMM.

Readers of this Manual

This manual is written for operators who install, operate, and maintain CMM. It also describes how the operators use CMM for monitoring and log management. The manual assumes that you have profound knowledge of OpenStack and CMM, especially the individual services CMM is composed of. For installing the CMM components, you must be familiar with the administration and operation of LINUX systems.

Notational Conventions

This manual uses the following notational conventions:

Notation	Description
`Add`	Names of graphical user interface elements.
`init`	System names, for example command names and text that is entered from the keyboard.
`<variable>`	Variables for which values must be entered.
`[option]`	Optional items, for example optional command parameters.
`one | two`	Alternative entries.
`{one | two}`	Mandatory entries with alternatives.

Abbreviations

This manual uses the following abbreviations:

Abbreviation	Description
CMM	Cloud Monitoring Manager
IaaS	Infrastructure as a Service
ICMP	Internet Control Message Protocol
OS	Operating System
OSS	Open Source Software
PaaS	Platform as a Service
SaaS	Software as a Service

Available Documentation

The following documentation on CMM is available:

- *Overview:* A manual introducing CMM. It is written for everybody interested in CMM.
- *System Operator's Guide:* A manual for system operators describing how to install, operate, and maintain CMM. The manual also describes how to prepare the OpenStack platform for CMM and how to use the CMM monitoring functions.
- *Application Operator's Guide:* A manual for application operators describing how CMM supports them in monitoring their services and virtual machines in OpenStack.

Related Information

The following links provide information on open-source offerings integrated with CMM:

- *OpenStack* : Documentation on OpenStack, the underlying platform technology.
- *OpenStack Horizon* : Documentation on the OpenStack Horizon dashboard.
- *Monasca* : Information on Monasca, the core of CMM.
- *Grafana* : Documentation on Grafana, the open-source application used for visualizing metrics data.
- *Kibana* : Documentation on Kibana, the open-source application used for visualizing log data.

Links to more detailed information provided in this manual are subject to change without notice.

Trademarks

Python and PyCon are trademarks or registered trademarks of the Python Software Foundation.

Other company names and product names are trademarks or registered trademarks of their respective owners.

High Risk Activity

The Customer acknowledges and agrees that the Product is designed, developed and manufactured as contemplated for general use, including without limitation, general office use, personal use, household use, and ordinary industrial use, but is not designed, developed and manufactured as contemplated for use accompanying fatal risks or dangers that, unless extremely high safety is secured, could lead directly to death, personal injury, severe physical damage or other loss (hereinafter “High Safety Required Use”), including without limitation, nuclear reaction control in nuclear facility, aircraft flight control, air traffic control, mass transport control, medical life support system, missile launch control in weapon system. The Customer shall not use the Product without securing the sufficient safety required for the High Safety Required Use. In addition, FUJITSU (or other affiliate’s name) shall not be liable against the Customer and/or any third party for any claims or damages arising in connection with the High Safety Required Use of the Product.

Export Restrictions

Exportation/release of this document may require necessary procedures in accordance with the regulations of your resident country and/or US export control laws.

1 Introduction to CMM

As more and more applications are deployed on cloud systems and cloud systems are growing in complexity, managing the cloud infrastructure is becoming increasingly difficult. Cloud Monitoring Manager (CMM) helps mastering this challenge by providing a sophisticated Monitoring as a Service solution that is operated on top of OpenStack-based cloud computing platforms.

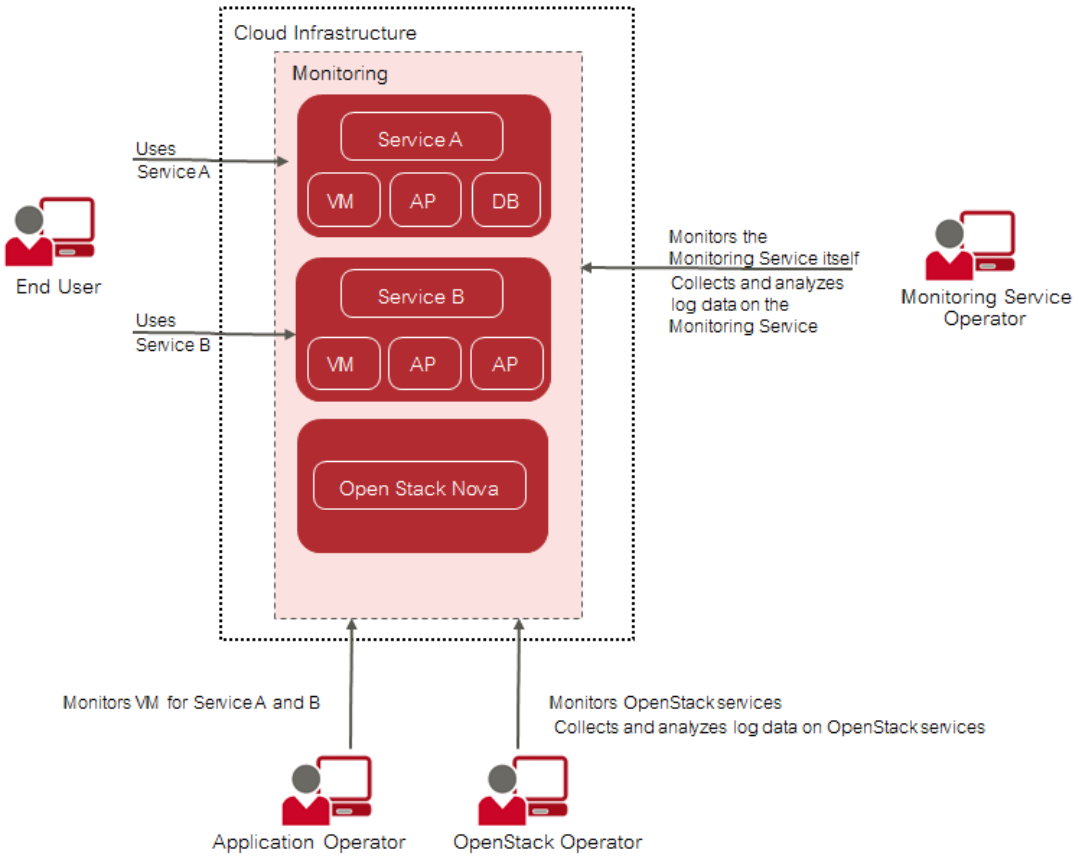
The component architecture of OpenStack provides for high flexibility, yet it increases the burden of system operation because multiple services must be handled. CMM offers an integrated view of all services and assembles and presents related metrics and log data in one convenient access point. While being flexible and scalable to instantly reflect changes in the OpenStack platform, CMM provides the ways and means required to ensure multi-tenancy, high availability, and data security.

CMM covers all aspects of a Monitoring as a Service solution:

- Central management of monitoring and log data from medium and large-size OpenStack deployments.
- Storage of metrics and log data in a resilient way.
- Multi-tenancy architecture to ensure the secure isolation of metrics and log data.
- Horizontal and vertical scalability to support constantly evolving cloud infrastructures. When physical and virtual servers are scaled up or down to varying loads, the monitoring and log management solution can be adapted accordingly.

1.1 Basic Usage Scenario

The basic usage scenario of setting up and using the monitoring features of CMM looks as follows:



Basic Usage Scenario

An **application operator** acts as a service provider in the OpenStack environment. This person books virtual machines to provide services to **end users** or to host services that are needed for his or her own development activities. CMM helps application operators ensure that their services and the servers on which they are provided are configured and working as required.

The **OpenStack operator** is responsible for administrating and maintaining the underlying OpenStack platform. The monitoring and log management services of CMM enable him or her to ensure the availability and quality of the platform. This person uses CMM for:

- Monitoring physical and virtual servers, hypervisors, and OpenStack services.
- Monitoring middleware components, for example, database services.
- Retrieving and analyzing the log data of the OpenStack services and servers, the middleware components, and the operating system.

The **Monitoring Service operator** is responsible for providing the monitoring and log management features to the application operators and the OpenStack operator. This enables them to focus on operation and the quality of their services and servers without having to carry

out the tedious tasks implied by setting up and administrating their own monitoring software. The Monitoring Service operator uses the monitoring features for ensuring the quality of CMM.

Tasks Overview

Depending on the distribution of tasks in your environment, the tasks of the Monitoring Service operator and the OpenStack Operator are performed by a single person or shared by different system operators.

When taking on the role of the Monitoring Service operator, you have the following responsibilities:

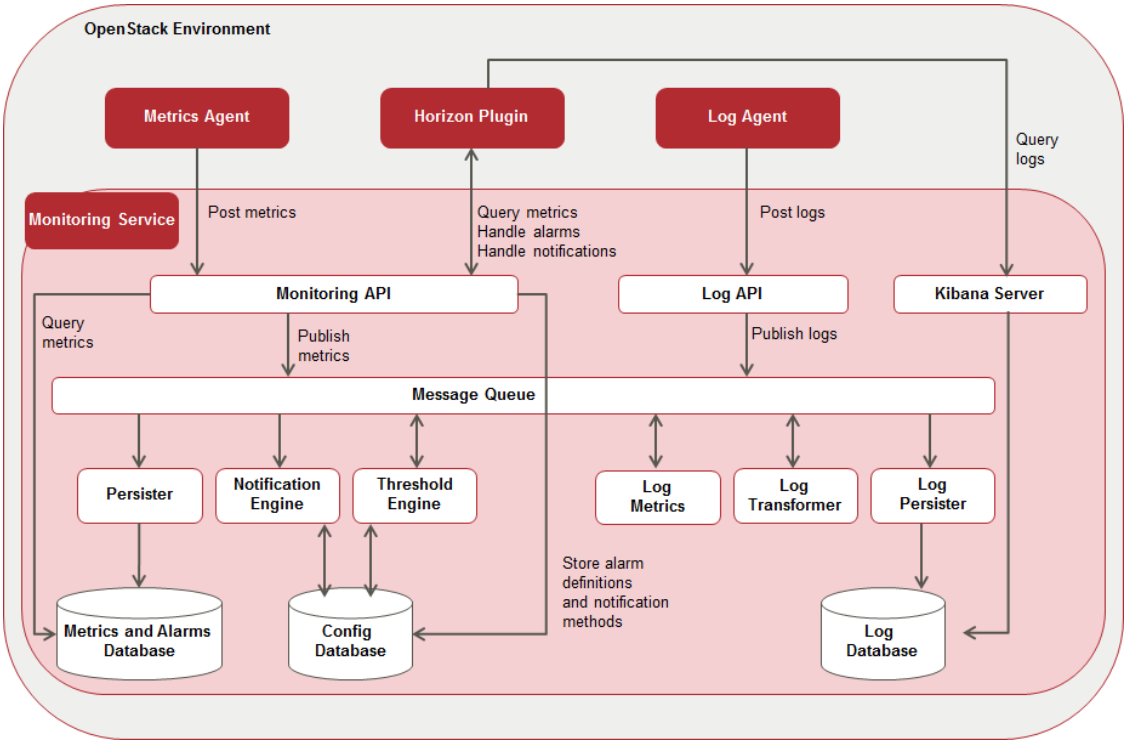
- Installation and setup of the Monitoring Service, thus providing the monitoring and log management features to the other CMM user groups.
- Regular maintenance of the components and services CMM consists of.
- Backup of the CMM databases, configuration files, and customized dashboards.
- Monitoring of CMM.

When taking on the role of the OpenStack operator, you have the following responsibilities:

- Installation and configuration of the OpenStack extention points required for CMM.
- Preparing the monitoring environment for the application operators.
- Monitoring the OpenStack services and servers.

1.2 Architecture and Components

The following illustration provides an overview of the main components of CMM and their interaction:



Architecture and Components

OpenStack

CMM relies on OpenStack, a technology for building cloud computing platforms for public and private clouds. OpenStack consists of a series of interrelated projects delivering various components for a cloud infrastructure solution and allowing for the deployment and management of Infrastructure as a Service (IaaS) platforms.

For details on OpenStack, refer to the *OpenStack documentation*.

Monitoring Service

The Monitoring Service is the central CMM component. It is responsible for receiving, persisting, and processing metrics and log data, as well as providing the data to the users.

The Monitoring Service relies on Monasca. It uses Monasca for high-speed metrics querying and log management, and integrates the streaming alarm engine as well as the notification engine of Monasca.

The Monitoring Service consists of the following components:

- **Monitoring API** A RESTful API for monitoring. It is primarily focused on the following areas:
  - Metrics: Store and query massive amounts of metrics in real- time.
  - Statistics: Provide statistics for metrics.
  - Alarm Definitions: Create, update, query, and delete alarm definitions.
  - Alarms: Query and delete the alarm history.
  - Notification Methods: Create and delete notification methods and associate them with alarms. Users can be notified directly when alarms are triggered, for example via email.
- **Message Queue** A component that primarily receives published metrics from the Monitoring API, alarm state transition messages from the Threshold Engine, and log data from the Log API. The data is consumed by other components, such as the Persister, the Notification Engine, and the Log Persister. The Message Queue is also used to publish and consume other events in the system. It is based on Kafka, a high-performance, distributed, fault-tolerant, and scalable message queue with durability built-in. For administrating the Message Queue, CMM uses Zookeeper, a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services.
- **Persister** A Monasca component that consumes metrics and alarm state transitions from the Message Queue and stores them in the Metrics and Alarms Database (InfluxDB).
- **Notification Engine** A Monasca component that consumes alarm state transition messages from the Message Queue and sends notifications for alarms, such as emails.
- **Threshold Engine** A Monasca component that computes thresholds on metrics and publishes alarms to the Message Queue when they are triggered. The Threshold Engine is based on Apache Storm, a free and open distributed real-time computation system.
- **Metrics and Alarms Database** An InfluxDB database used for storing metrics and the alarm history.
- **Config Database** A MySQL database used for storing configuration information, alarm definitions, and notification methods.
- **Log API** A RESTful API for log management. It gathers log data from the Log Agents and forwards it to the Message Queue. The CMM log management is based on Logstash, a tool for receiving, processing, and publishing all kinds of logs. It provides a powerful pipeline for querying and analyzing logs. Elasticsearch is used as the back-end datastore, and Kibana as the front-end tool for retrieving and visualizing the log data.
- **Log Transformer** A Logstash component that consumes the log data from the Message Queue, performs transformation and aggregation operations on the data, and publishes the data that it creates back to the Message Queue.
- **Log Metrics** A Monasca component that consumes log data from the Message Queue, filters the data according to severity, and generates metrics for specific severities, for example for errors or warnings. The generated metrics are published to the Message Queue and can be further processed by the Threshold Engine like any other metrics.
- **Log Persister** A Logstash component that consumes the transformed and aggregated log data from the Message Queue and stores it in the Log Database.
- **Kibana Server** A Web browser-based analytics and search interface to the Log Database.

- **Log Database** An Elasticsearch database for storing the log data.

Note: The installation of the Monitoring Service includes the installation of all CMM third-party components that are required. From time to time, it may be necessary to install bug fixes or security patches for these third-party components. In order to guarantee for the interoperability and integrity of your CMM installation, you should obtain such fixes and patches solely from your CMM support organization.

Horizon Plugin

CMM comes with a plugin for the OpenStack Horizon dashboard. The plugin extends the main dashboard in OpenStack with a view for monitoring and log management. This enables CMM users to access the CMM functionality from a central Web-based graphical user interface. For details, refer to the *OpenStack Horizon documentation*.

Based on the Monitoring Service, metrics and log data are visualized on convenient and user-friendly dashboards which fully integrate with the following applications:

- Grafana (for metrics data). An open-source application for visualizing large-scale measurement data.
- Kibana (for log data). An open-source analytics and visualization platform designed to work with Elasticsearch.

Metrics Agent

A Metrics Agent is required for retrieving metrics data from the host on which it runs and sending the data to the Monitoring Service. The push-based agent supports metrics from a variety of sources as well as a number of built-in system and service checks.

A Metrics Agent can be installed on each virtual or physical server to be monitored. The agent functionality is fully integrated into the source code base of the Monasca project. For details, refer to Monasca.

Log Agent

A Log Agent is needed for collecting log data from the host on which it runs and forwarding the data to the Monitoring Service for further processing. It can be installed on each virtual or physical server from which log data is to be retrieved.

The agent functionality is fully integrated into the source code base of the Monasca project. For details, refer to Monasca.

1.3 User Management

CMM is fully integrated with Keystone, the identity service which serves as the common authentication and authorization system in OpenStack.

The integration with Keystone requires any CMM user to be registered as an OpenStack user. All authentication and authorization in CMM is done through Keystone. If a user requests monitoring

data, for example, CMM verifies that the user is a valid user in OpenStack and allowed to access the requested metrics.

CMM users are created and administrated in OpenStack:

- Each user assumes a role in OpenStack to perform a specific set of operations. The OpenStack role specifies a set of rights and privileges.
- Each user is assigned to at least one project in OpenStack. A project is an organizational unit that defines a set of resources which can be accessed by the assigned users. Application operators in CMM can monitor the set of resources that is defined for the projects to which they are assigned.

For details on user management, refer to the *OpenStack documentation*.

1.4 Distribution Media

CMM is distributed in the CMM installation package. It contains archive files with the CMM software and documentation for installing and configuring CMM.

The `CMM_server_2.0.14-x.tar.gz` file includes (replace x with the current version number of CMM):

- `CMM_server_2.0.14-x.images.tar` Tarred archive file with the Docker images required for installing and configuring CMM.
- `docker-compose-Linux-x86_64_1.27.4` Docker Compose binary required to install CMM.
- `containerd.io-1.4.4-3.1.el7.x86_64.rpm` Docker-CE dependency rpm.
- `docker-ce-19.03.15-3.el7.x86_64.rpm` Docker-CE rpm.
- `docker-ce-cli-19.03.15-3.el7.x86_64.rpm` Docker-CLI rpm.
- `docker-compose-metric.yml` and `docker-compose-log.yml` Docker Compose YAML files required for installing and configuring CMM.
- `.env` Docker Compose environment file defining the default environment variables used by Docker Compose.

The `CMM_client_2.0.14-x.tar.gz` file includes (replace x with the current version number of CMM):

- `monasca-agent-CMM_2.0.14-x.run` Executable for installing a Metrics Agent on an OpenStack node.
- `log-agent-CMM_2.0.14-x.run` Executable for installing a Log Agent on an OpenStack node.
- `monasca-ui-CMM_2.0.14-x.run` Executable for installing the Horizon Plugin on the node where the OpenStack Horizon Service is deployed.

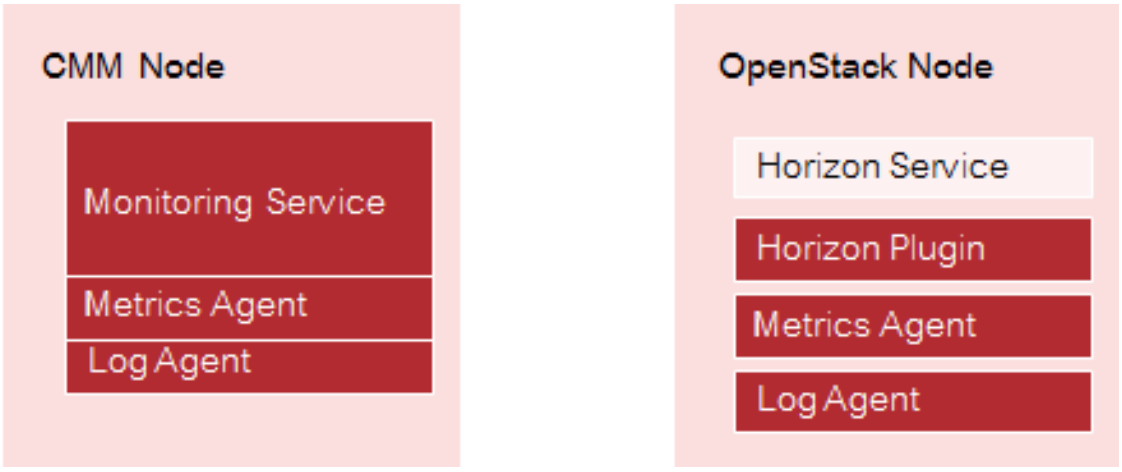
The `CMM_documentation_2.0.14-x.tar.gz` file includes PDF manuals providing an overview of CMM as well as information directed to system operators and application operators (replace `x` with the current version number of CMM).

2 Installation

The installation of CMM comprises the following steps:

1. Preparing the installation environment.
2. Preparing the integration of CMM with your OpenStack platform.
3. Installing the Monitoring Service. The installation of the Monitoring Service is based on Docker. Production-ready Docker images are provided for the services that make up the Monitoring Service. Instances of these images run in separate Docker containers as soon as the installation is successful.
4. Installing a Metrics Agent. A Metrics Agent is installed as an extension to your OpenStack platform for monitoring an OpenStack service and the server on which it is deployed.
5. Installing a Log Agent. A Log Agent is installed as an extension to your OpenStack platform for managing the logs of an OpenStack service and the server on which it is deployed.
6. Installing the Horizon Plugin. The Horizon Plugin is installed as an extension to your OpenStack platform. It is required for accessing the monitoring and log management functions of CMM from OpenStack Horizon.

The following picture depicts a simplified installation scenario with a CMM node and one OpenStack node only:



Installation

Observe the following when installing CMM:

- It is recommended that you use a dedicated physical machine as a CMM node which meets the requirements outlined in the next section. It is not recommended that there are any other services running on the machine.

Note:	The Monitoring Service can be installed on a virtual machine. For CMM in productive operation, however, keep in mind that CMM has demanding hardware requirements that are not met by a virtual machine. In addition, neighboring virtual machines may cause trouble.

- The installation of the Monitoring Service automatically includes the installation of a Metrics Agent and a Log Agent for monitoring CMM.
- The Horizon Plugin must be installed on the node where the OpenStack Horizon Service is deployed.
- A productive OpenStack environment consists of multiple OpenStack nodes on which multiple services are running. You can install the agents on any OpenStack node where a service to be monitored is deployed.

2.1 Prerequisites and Preparation

The following sections describe the prerequisites that must be fulfilled and the preparations you need to take before installing CMM.

2.1.1 Hardware and Operating Systems

CMM can be installed on a host machine with the following operating systems:

- Red Hat Enterprise Linux 7.7 for Intel64.

Note: Make sure that you use a clean operating system. This avoids compatibility-related issues during the installation.

As underlying platform technology, the following OpenStack platforms are supported:

- Red Hat Enterprise Linux OpenStack Platform 16.1 installed on Red Hat Enterprise Linux 8.2

The following hardware resources are recommended:

Requirement	Description
Architecture	x86_64
RAM	At least 32 GB, 64 GB or more recommended.
CPU	At least 8 cores, 16 cores or more recommended.
Hard disk	HDD or SSD, SSD strongly recommended for productive operation and high performance.
Disk space	See below.

CMM without any data requires about 2 GB of disk space. The disk capacity required for log data and metrics data varies considerably depending on the number of services and servers to be monitored. For information on how to calculate the required disk space, refer to the section below.

The installation mounts a volume for the data directories of Elasticsearch, InfluxDB, MySQL, Kafka, and Grafana. The default installation uses the `/opt/monasca-containers/` directory.

Calculating the Required Disk Space

As a basic rule of thumb, the following formula can be used for calculating the required disk space:

```
200 GB + [ <number of nodes> * <retention period> * (<space for log data/day> + <space for metrics data/day>) ]
```

Replace the variables as follows:

- The number of nodes corresponds to the number of nodes to be monitored.
- The retention period corresponds to 60 days for InfluxDB and Elasticsearch.
- The space for log data/day corresponds to approximately 2 GB.
- The space for metrics data/day corresponds to approximately 50 MB.

The formula is based on the following assumptions for log data:

- Approximately 50 log files per node
- Approximately 1 log entry/file/sec
- 200 bytes in size

The formula is based on the following assumptions for metrics data:

- 400 metrics per node
- Time interval of 30 secs
- 20 bytes in size

Note: Depending on the OpenStack nodes to be monitored there might be additional factors that have an impact on the required disk space, for example enabling debug logging for the OpenStack services.

2.1.2 Web Browsers

CMM has been tested with the following Web browsers:

- Google Chrome 90.
- Microsoft Edge 91.
- Mozilla Firefox 58.

Notes:

- Mozilla Firefox 58:
  When accessing Grafana, a message is displayed: '`Your browser is not fully supported. A newer Browser version is recommended`'. This browser version has been extensively tested. There is no known restriction when using Firefox 58 to access CMM metric information via Grafana.
- All Browsers:
  When accessing Kibana, a message is displayed: '`Your browser doesn't meet the security requirements for Kibana`'. However, this message is not related to

browser versions. Security in CMM is ensured: Only users with access to Horizon can successfully use Kibana to access CMM log information. Thus, this message can be safely ignored.

2.1.3 Security

In a default Red Hat Enterprise Linux installation, the following security precautions are taken:

- A firewall exists to prevent unauthorized user access.
- Security-Enhanced Linux (SELinux), a security module that adds mandatory access control mechanisms to the Linux kernel, is enabled.

Before installing CMM, make sure that the firewall and SELinux match your security requirements.

Access to the following ports must be enabled, before installing CMM:

- Port `5607` for the Log API.
- Port `8070` for the Monitoring API.
- Port `5601` for the Kibana Server.
- Port `3000` for Grafana.

In addition, access to port 8081 is required internally by the Monitoring API, for example for healthchecks or threads.

To integrate with the required OpenStack services, CMM requires access to the following ports:

- Port `80` for the OpenStack Horizon service.
- Port `5000` and `35357` for the OpenStack Keystone service.

2.2 Preparing the OpenStack Integration

All authentication and authorization in CMM is done through OpenStack Keystone. The integration of CMM with OpenStack must therefore be prepared before you install the Monitoring Service on the CMM node and the OpenStack extensions on an OpenStack node.

You need to take the following preparations on an OpenStack node:

- Setting the OpenStack administator credentials.
- Creating a project, a user, and the required roles.
- Defining services and endpoints.
- Configuring the HTTP proxy.

2.2.1 Setting the Administrator Credentials

To perform any action in OpenStack Keystone, it is required to provide the administrator credentials.

Example:

```
export OS_USERNAME=admin
export OS_PROJECT_NAME=admin
export OS_PASSWORD=<admin_password>
export OS_AUTH_URL=<http://<os_auth_ip>:5000
export OS_REGION_NAME=<region>
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USER_DOMAIN_NAME=Default
```

Please replace: * `<admin_password>` by password of OpenStack admin user * `<os_auth_ip>` by IP of the server where OpenStack keystone service is running * `<region>` by the name of your OpenStack region, e.g. `regionOne`.

You can verify the provided credentials with the following command:

```
openstack service list
```

2.2.2 Creating Projects, Users, and Roles

To integrate CMM with the OpenStack Keystone service, specific preparations must manually be taken in OpenStack Keystone.

You have to create a project, a user, and the required roles. The project, the user and the roles must exist before you proceed with installing the Monitoring Service.

Creating a Project

A dedicated OpenStack project must be prepared for CMM. The monitoring data retrieved for CMM and your OpenStack services and servers is accessible to users assigned to this project, provided they have the required role assigned.

Example:

```
openstack project create monasca
```

Creating Roles

The following OpenStack roles must be prepared:

- `monasca-user`. This role must be assigned to any user who wants to perform monitoring and log management tasks.
- `monasca-agent`. This role must be assigned to the user used for authenticating the Metrics Agent and the Log Agent against OpenStack Keystone.

Example:

```
openstack role create monasca-user
openstack role create monasca-agent
```

Creating a User

An OpenStack user must be prepared that is used for authenticating an agent against OpenStackKeystone. The user must be specified in the agent configuration. The user must have the `monasca-agent` role in OpenStack and be assigned to the OpenStack project that is to be monitored by the agent.

This user is used by the Metrics Agent and the Log Agent to submit data to the Monitoring Service. It is recommended that this user is used only for configuration purposes and not for actually monitoring services and servers.

Example:

```
openstack user create --project monasca --password password monasca-agent
```

Assigning Roles

The following assigments must be prepared:

- The new OpenStack user (`monasca-agent` in the example) must be assigned to the project prepared for CMM and must have the `monasca-agent` role in this project.
- The OpenStack `admin` user must be assigned to the project prepared for CMM and must have the `monasca-user` and `admin` roles in this project.

Example:

```
openstack role add --project monasca --user monasca-agent monasca-agent
openstack role add --project monasca --user admin monasca-user
openstack role add --project monasca --user admin admin
```

2.2.3 Defining Services and Endpoints

The Horizon Plugin requires a set of services and endpoints that must be defined in OpenStack Keystone.

For this purpose, check your OpenStack Keystone setup, and create the required services and endpoints. You can create them as follows:

```
openstack service create --name monasca monitoring
openstack service create --name logs logs

openstack endpoint create monasca public http://<cmm_ip>:8070/v2.0 --region
<region>
openstack endpoint create logs public http://<cmm_ip>:5607/v2.0 --region <region>
```

Replace <cmm_ip> by the IP address of the CMM node for example, http://192.168.10.6:8070/v2.0, and <region> by the name of your OpenStack region, for example, regionOne.

2.2.4 Configuring the HTTP Proxy

Grafana requires the configuration of an HTTP proxy in Apache:

1. Add Proxy Modules at the end of file in httpd.conf. Location of file:
   /var/lib/config-data/puppet-generated/horizon/etc/httpd/conf/httpd.conf

   Add these lines at the end:

   ```
   LoadModule proxy_module modules/mod_proxy.so
   LoadModule proxy_http_module modules/mod_proxy_http.so
   ```

2. Add the Proxy configuration inside VirtualHost section in 10-horizon_vhost.conf. Location of file:
   /var/lib/config-data/puppet-generated/horizon/etc/httpd/conf.d/10-horizon_vhost.conf

   Add lines:

   ```
   ProxyPass        "/grafana" "http://<CMM-SERVER-IP>:3000"
   ProxyPassReverse "/grafana" "http://<CMM-SERVER-IP>:3000"
   ```

   before line:
   </VirtualHost>

   Replace by CMM Server IP Address, e.g.:

   ```
   ProxyPass        "/grafana" "http://10.140.99.76:3000"
   ProxyPassReverse "/grafana" "http://10.140.99.76:3000"
   ```

3. Restart Horizon container:
   ```
   # systemctl restart tripleo_horizon
   ```

2.3 Installing the Monitoring Service

Before installing the Monitoring Service make sure that you have made the preparations required for your OpenStack platform.

2.3.1 Prerequisites

For installing the Monitoring Service, a server with RHEL7.7 is required with:

- Docker-CE 19.03.15 For details, refer to the *Docker documentation*. RPMs are included in the CMM_server_2.0.14-x.tar.gz file.
- Docker Compose binary as included in the CMM_server_2.0.14-x.tar.gz file.

Installation of docker and docker-compose is described in the following chapter.

Depending on the Elasticsearch requirements resulting from your production environment, it is recommended that you customize the default Elasticsearch configuration provided by the Monitoring Service installation. For details on preparations related to Elasticsearch in productive use, refer to the *Monasca Docker documentation*.

2.3.2 Installation

To install the Monitoring Service, proceed as follows:

1. Log in to the machine on which to install the Monitoring Service (CMM node) as a user with root privileges.
2. Prepare an installation directory.
3. Extract the `CMM_server_2.0.14-x.tar.gz` archive file from the CMM installation package to the installation directory. The archive provides the following files:
   - `CMM_server_2.0.14-x.images.tar`
   - `containerd.io-1.4.4-3.1.el7.x86_64.rpm`
   - `docker-ce-19.03.15-3.el7.x86_64.rpm`
   - `docker-ce-cli-19.03.15-3.el7.x86_64.rpm`
   - `docker-compose-Linux-x86_64_1.27.4`
   - `docker-compose-metric.yml`
   - `docker-compose-log.yml`
   - `.env`
   - `purge-zookeeper-txnlogs.sh`
4. go to the installation directory: `cd <install_dir>`
5. Install Docker-CE 19.03.15

   Note: Docker-CE requires the package `container-selinux` which is available in the repository `rhel-7-server-extras-rpms` please enable it.

   ```
   # yum install docker-ce-19.03.15-3.el7.x86_64.rpm docker-ce-cli-19.03.15-
   3.el7.x86_64.rpm containerd.io-1.4.4-3.1.el7.x86_64.rpm
   ```

   Note: It is recommended to configure data retention for Docker container logs. Refer to Log File Handling for details.

6. Copy the `docker-compose-Linux-x86_64_1.27.4` file to the `/usr/local/bin/` directory and rename it to `docker-compose`.
7. Open the `.env` file in the installation directory to make the adaptions required for your environment.

   Note: Restrict the access permissions of the .env file. It specifies passwords that must be protected from unauthorized access!

8. For integrating the Monitoring Service with OpenStack Keystone, you have to specify the following parameters:

   ```
   # Set the IPv4 address of the OpenStack Keystone host
   MON_KEYSTONE_URL=http://<ipv4_address>:5000
   ```

```
# Specify the URL of the OpenStack Horizon host
# The URL is needed for setting the Monasca data source in Grafana
HORIZON_URL=http://<ip_address:port>
HORIZON_PATH=/dashboard

# Enable Kibana authorization via OpenStack Horizon
MON_MONASCA_PLUGIN_ENABLED=True

# Set the path to mount Kibana to the OpenStack Horizon proxy
MON_BASE_PATH=/dashboard/monitoring/logs_proxy

# Define Grafana administrator settings
MON_GRAFANA_ADMIN_USER=<grafana_admin_user_name>
MON_GRAFANA_ADMIN_PASSWORD=<grafana_admin_password>
```

- Replace `<ipv4_address>` by the IP address of the node on which the OpenStack Keystone service is deployed. Example URL: `http://172.31.0.216:5000`.
- Replace `<ip_address:port>` by the IP address and port of the node on which the OpenStack Horizon service is deployed. This is required for configuring the Monasca data source in Grafana. Example URL: `http://172.31.0.216:80`.
- Set the `MON_MONASCA_PLUGIN_ENABLED` parameter to `True`.
- Set the `MON_BASE_PATH` parameter to `/dashboard/monitoring/logs_proxy`.
- Replace `<grafana_admin_user_name>` and `<grafana_admin_password>` by the credentials of a Grafana administrator. This user is automatically created with the installation of the Monitoring Service, and is assigned the `admin` flag in Grafana.

Note: CMM ships with preconfigured metrics dashboards. The Grafana administrator created with MON_GRAFANA_ADMIN_USER is authorized to create additional dashboards for the CMM users, or to update and delete the preconfigured ones, if required.

9. For enabling access to OpenStack Keystone, you have to specify credentials for the Metrics Agent and Log Agent, as well as for the Monitoring API and the Log API in the "Set the OpenStack Keystone credentials" section:

```
# Credentials of the user used for authenticating the agents
# against Keystone
MON_AGENT_USERNAME=<user_name>
MON_AGENT_PASSWORD=<password>
MON_AGENT_PROJECT_NAME=<project_name>

# Credentials of the OpenStack admin
MON_KEYSTONE_ADMIN_USER=<OpenStack_admin_user_name>
MON_KEYSTONE_ADMIN_PASSWORD=<OpenStack_admin_password>
```

- Replace `<user_name>` and `<password>` by the credentials of the user used for authenticating the agents against OpenStack Keystone, and replace `<project_name>` by the name of the OpenStack project for which data is to be retrieved by the agents. The names and passwords that you enter must correspond to the project and agent user you

have already prepared for the integration of CMM with OpenStack. For details, refer to *Creating Projects, Users, and Roles*.

Example:

```
MON_AGENT_USERNAME=monasca-agent
MON_AGENT_PASSWORD=password
MON_AGENT_PROJECT_NAME=monasca
```

- Replace <OpenStack_admin_user_name> and <OpenStack_admin_password> by the credentials of the OpenStack admin user. By default, the admin user is used for authenticating the Monitoring API and the Log API against OpenStack Keystone.

10. The installation of the Monitoring Service mounts /opt/monasca-containers/ as default volume for the data directories of Elasticsearch, InfluxDB, MySQL, Kafka, and Grafana.

    If required, you can update the MON_DOCKER_VOL_ROOT parameter, and specify a different volume.

    ```
    # Set the path for the data directories of Elasticsearch,
    # InfluxDB, MySQL, Kafka, and Grafana
    MON_DOCKER_VOL_ROOT=<path_to_data_directories>
    ```

11. The installation of the Monitoring Service mounts /mount/backup/ as default volume for backing up the databases.

    If required, you can update the MON_BACKUP_DIR parameter, and specify a different volume.

    ```
    # Set the path for the backup directories of Elasticsearch,
    # InfluxDb, and MySQL
    MON_BACKUP_DIR=<path_to_backup_directories>
    ```

12. By default, CMM retains the data stored in the Elasticsearch and InfluxDB database for 31 days. Older data is automatically deleted.

    If required, you can change the data retention parameters in the Configure data retention section.

    ```
    # Retention period for Elasticsearch database
    # Delete job is executed every day at 12 a.m. UTC
    MON_ELASTICSEARCH_DATA_RETENTION_DAYS=<number_of_days>

    # Retention period for InfluxDB database
    MON_INFLUXDB_RETENTION=<number_of_days>d
    ```

    Replace <number_of_days> by the number of days after which data is to be deleted.

    Example:

    ```
    MON_ELASTICSEARCH_DATA_RETENTION_DAYS=30
    MON_INFLUXDB_RETENTION=30d
    ```

13. Enable the notification methods to be used to inform CMM users when a threshold value for an alarm is reached or exceeded. Email, Slack, and Webhook are methods supported by CMM. If you want to use HipChat, PagerDuty, or Jira, or need an extension to the Notification Engine for exchanging information with additional external systems, contact your FUJITSU support organization. The notification methods to be enabled must be specified as comma-separated values for the NF_PLUGINS parameter in the Enable the Notification Engine plugins section. Use lower-case characters only. Example for enabling Email and WebHook:
    ```
    NF_PLUGINS=webhook,email
    ```

    The `webhook` plugin is enabled by default. For the other plugins, you have to additionally define the corresponding configuration parameters in the `.env` file. For details on the parameters, refer to the information in the file.

14. Load the tarred repository from the `CMM_server_2.0.14-x.images.tar` file. This restores both the images and the tags from the archive to your installation directory.
    ```
    docker load -i CMM_server_2.0.14-x.images.tar
    ```

15. Check that the images and tags required for the installation have been loaded.
    ```
    docker images
    ```

16. Run docker-compose up.
    ```
    docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml up -d
    ```

    Note: Make sure that the OpenStack Keystone service is up when running docker-compose up. Use -d to start the containers in the background and leave them running when the command exits. Refer to the Docker Compose documentation for additional details on the docker-compose up command.

    After a successful deployment, the monitoring pipeline starts within approximately one minute.

17. To restrict access to the backup and data directories, you must change the access permissions. Example:
    ```
    chmod -R 700 /opt/monasca-containers
    chmod -R 700 /mount/backup/
    ```

CMM Services

By default, Docker Compose starts containers for the following services. They correspond to the main components of CMM:

Service Name	Description
agent-collector	Collector service of the Metrics Agent. Based on a configurable interval, the collector component collects the metrics data from the monitored services and servers.
agent-forwarder	Forwarder service of the Metrics Agent. The forwarder component takes the data from the collector and sends them to the Monitoring API.

elasticsearch	Elasticsearch database service. CMM stores the log data in this database.
grafana	Grafana service. CMM uses it to enable the integration with the OpenStack Keystone service and to include Monasca as Grafana data source.
influxdb	InfluxDB database service. CMM stores metrics and alarms in this database.
kafka	Message queue service.
kibana	Kibana server.
log-agent	Log Agent.
log-api	Log API.
log-metrics	Log Metrics.
log-persister	Log Persister.
log-transformer	Log Transformer.
memcached	Memcached service. CMM uses it for caching authentication and authorization information required for the communication between the Log API and OpenStack Keystone.
monasca	Monitoring API.
monasca-notification	Notification Engine.
monasca-persister	Persister.
monasca-statsd	Service for sending statsd messages via the Monasca Agent.
mysql	MySQL database service. CMM stores configuration information in this database.
thresh	Threshold Engine.
zookeeper	Centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services.

In addition, Docker Compose starts a number of init containers that are used to bootstrap the Monitoring Service installation as well as a number of containers with direct dependencies to the services listed above.

2.4 Metrics Agent on the OpenStack Platform

For monitoring OpenStack services in your environment, you need to install a Metrics Agent on the OpenStack node on which the services are running. The agent installer performs the following tasks:

- It automatically configures the agent to retrieve metrics data from the server and send the data to the Monitoring Service for further processing.
- It automatically activates system metrics for monitoring the OpenStack services and the server on which they are running. The metrics include system checks, for example on CPU usage, disk space, or the average system load. No manual configuration is required for these checks.

- As enhancement to the system metrics, the installer auto-detects applications and OpenStack processes that are running on the server. The corresponding metrics are automatically configured. No manual configuration is required.

The agent installer creates all configuration files required for monitoring. They include the following:

- `agent.yaml` located in the `/etc/monasca/agent/` directory. It defines the agent configuration.
- `*.yaml` files located in the `/etc/monasca/agent/conf.d/` directory. They provide the metrics the agent uses for monitoring. A `*.yaml` file is provided for each set of system metrics and for the additional metrics used to monitor the specific applications and OpenStack processes that are auto-detected. The `*.yaml` files store the configuration information that is auto-detected during the installation process. For details on the system metrics and the additional metrics, refer to Appendix A *Supported Metrics* on Page 84.
- `*.yaml` files located in the `</<installation_dir>/share/monasca/agent/conf.d/` directory. They provide metrics that you can activate as enhancement to the metrics that are automatically provided with the agent installation. These files provide template configurations that you can adapt to your environment.

The installation of an agent includes its initial configuration. This ensures that comprehensive metrics data for monitoring services and servers is retrieved.

Options are also provided for reconfiguring an agent following a successful installation. If required, you can update the `agent.yaml` file, and you can activate metrics in addition to the ones that are automatically provided with the installation.

2.4.1 Metric Agent Prerequisites

Please check the following items for servers where metric agent (bare metal) shall be installed:

- OS: `RHEL 8.2`
- OpenStack: `RHOSP16.1`

2.4.2 Metric Agent Installation

To install a Metrics Agent, proceed as follows:

1. Log in to the OpenStack node on which to install the Metrics Agent.
2. Prepare an installation directory.
3. Extract the `CMM_client_2.0.14-x.tar.gz` archive file from the CMM installation package to the installation directory. The archive provides the following files:
   - `log-agent-7.3.0_2.0.x-CMM2.0.14-x.run`
   - `monasca-agent-3.0.x-CMM2.0.14-x.run`
   - `monasca-ui-1.17.x-CMM2.0.14-x.tar.gz`
4. Change the access permission of the `monasca-agent-3.0.x-CMM2.0.14-x.run` file to Execute.
5. Make sure that your `/root/.my.cnf` file does not define any passwords in single quotes.

6. Run the agent installer:

```
./monasca-agent-3.0.x-CMM2.0.14-x.run \
--target /opt/monasca-agent -- \
--username <user_name> \
--password <password> \
--project_name <project_name> \
--user_domain_name default \
--project_domain_name default \
--service_type monitoring \
--keystone_url <openstack_url> \
--monasca_statsd_port <port_no> \
--skip_detection_plugins OVS Libvirt
```

The following parameters must be configured for running the installer:

- `--target`. The directory in which the agent is installed. `--target` must be set to `/opt/monasca-agent`.
- `--username`. The user to be used for authenticating the agent against OpenStack Keystone, for example `monasca-agent`. The user specified here must have the `monasca-agent` role in OpenStack and be assigned to the OpenStack project that is to be monitored by the agent. The project is specified in project_name, for example `monasca`. It is recommended that this user is used only for configuration purposes and not for actually monitoring services and servers.
- `--password`. The password of the user specified in `--username`.
- `--project_name`. The name of the OpenStack project for which metrics data is to be retrieved by the agent. If `--project_name` is not specified, the data is retrieved from the default project assigned to the user specified in `--username`.
- `--user_domain_name`. The user domain name to be used for user name scoping.
- `--project_domain_name`. The project domain name used for authenticating the agent against OpenStack Keystone.
- `--service_type`. The identifier used for the Monasca API project in the OpenStack service catalog. `--service_type` must be set to `monitoring`.
- `--keystone_url`. URL used to access the server where the OpenStack Keystone service is installed. It must be a V3 endpoint. Example: `http://192.168.1.5:35357/v3`
- `--monasca_statsd_port`. The port number for the StatsD daemon. Check whether the default port number must be changed from `8125` to `8126`. Port `8125` is used by Gnocchi in Red Hat Enterprise Linux OpenStack Platform 10.
- `--skip_detection_plugins`. When running the installer, it is required to disable auto-detection for two agent plugins. `--skip_detection_plugins` must be set to `OVS Libvirt`.

For additional details, you can refer to the Monasca documentation. Help is also available on the configuration settings. For a description of the settings, execute the following command:

```
./monasca-agent-3.0.x-CMM2.0.14-x.run --help
```

In case the installation fails, check your configuration settings and passwords. To collect debugging information, you can retry the installation in verbose mode:

```
./monasca-agent-3.0.x-CMM2.0.14-x.run \
  --target /opt/monasca-agent -- \
  --username <user_name> \
  --password <password> \
  --project_name <project_name> \
  --user_domain_name default \
  --project_domain_name default \
  --service_type monitoring \
  --keystone_url <openstack_url> \
  --monasca_statsd_port <port_no> \
  --skip_detection_plugins OVS Libvirt \
  --verbose
```

Note: When running the installer, you can ignore error messages related to components that do not exist in your OpenStack environment, e.g. ERROR: Kibana process has not been found. Plugin for Kibana will not be configured.
The agent tries to auto-detect a specific set of processes and outputs an error message if a process is not found. Irrespective of these messages, the agent is successfully installed.

The installer creates a `monasca-agent.target` file in the `/etc/systemd/system/` directory, and automatically runs the service file to start the agent after a successful installation.

The agent is provided as a LINUX service. A startup script is created that automatically starts the agent each time the machine is booted.

The agent is installed in a virtualenv environment. By default, the virtualenv enviroment is located in the `/opt/monasca-agent/` directory.

Note: If you require to monitor disk mount points which are only accessible to root user, then `monasca-collector.service` should run as root:

1. Comment the `User` and `Group` parameters in the file `/etc/systemd/system/monasca-collector.service` as follows:

   ```
   [Service]
   Type=simple
   #User=mon-agent
   #Group=mon-agent
   Restart=always
   ExecStart=/opt/monasca-agent/bin/monasca-collector foreground
   ```

2. Reload the systemd units

   ```
   # systemctl daemon-reload
   ```

3. Restart monasca-collector service

   ```
   # systemctl restart monasca-collector.service
   ```

2.4.3 Metric Agent Updating the Configuration File

CMM allows you to change the configuration of an agent that is up and running. This might be required, for example, due to changes in your environment. To change the agent configuration, you have to edit the `agent.yaml` configuration file.

Proceed as follows:

1. Log in to the OpenStack node on which the agent is installed.
2. To stop the agent, execute the following command:
   ```
   systemctl stop monasca-agent.target
   ```
3. Open the file with your favorite editor. Example:
   ```
   vim /etc/monasca/agent/agent.yaml
   ```
4. Adapt the configuration settings as required.
5. To start the agent again, execute the following command:
   ```
   systemctl start monasca-agent.target
   ```

The agent is instantly available with the updated configuration settings.

2.4.4 Activating Additional Metrics

The agent installation automatically configures and activates a comprehensive set of metrics for monitoring your services and servers. The agent ships with additional metrics templates that you can manually adapt to your environment and activate for monitoring, if required.

For a list of the metrics that are supported by CMM, refer to Appendix A *Supported Metrics* on page 84.

For information on the complete set of metrics that is provided by the Monasca project, refer to the *Monasca documentation*. If you want to extend your monitoring environment to perform additional checks, contact your FUJITSU support organization.

To activate additional metrics, proceed as follows:

1. Log in to the OpenStack node on which the agent is installed.
2. To stop the agent, execute the following command:
   ```
   systemctl stop monasca-collector
   ```
3. Copy the required template file. Example:
   ```
   cp /opt/monasca-agent/share/monasca/agent/conf.d/<plugin>.yaml.example \
     /etc/monasca/agent/conf.d/<plugin>.yaml
   ```
4. Open the template file with your favorite editor. Example:
   ```
   vim /etc/monasca/agent/conf.d/rabbitmq.yaml
   ```
5. Adapt the configuration to your environment. For configuration examples, refer to *Additional Metrics*.

6. To start the agent again, execute the following command:

```
systemctl start monasca-collector
```

The activated metrics can instantly be used by the agent for retrieving monitoring data.

2.5 Log Agent on the OpenStack Platform

For monitoring OpenStack services in your environment, you need to install a Log Agent on the OpenStack node on which the services are running. The agent installer configures the agent so that it can automatically be started as soon as the installation is successful. You can enhance the agent configuration before running the installer or update the initial configuration later, if required.

The installer stores all configuration settings of the Log Agent in the following file:
`/<installation_dir>/conf/agent.conf`

The file is composed of an input and an output section:

- The input section specifies which log data is to be retrieved. The Log Agent is based on the so-called ELK stack, a solution for searching and analyzing log data that combines the open-source projects Elasticsearch, Logstash, and Kibana. For details on the ELK stack, refer to the documentation on *Elasticsearch, Logstash, and Kibana*. CMM supports the file plugin of Logstash as input mechanism. The file plugin enables Logstash to read log data from any log file on your file system. Logstash supports additional plugins. For details, refer to *Logstash Input Plugins*. Contact your CMM support if you want to integrate a different plugin.
- The output section specifies all parameters required for retrieving the log data and sending it to the Monitoring Service for further processing.

2.5.1 Log Agent Prerequisites

The Log Agent installer requires a Java Runtime Environment on the host machine where the agent is installed.

Before running the installer, you have to install the following OpenJDK package:

- `java-1.8.0-openjdk`

2.5.2 Log Agent Installation

To install a Log Agent, proceed as follows:

1. Log in to the OpenStack node on which to install the Log Agent.
2. Prepare an installation directory.
3. Extract the CMM_client_2.0.14-x.tar.gz archive file from the CMM installation package to the installation directory. The archive provides the following files:
   - `log-agent-7.3.0_2.0.x-CMM2.0.14-x.run`
   - `monasca-agent-3.0.x-CMM2.0.14-x.run`

- monasca-ui-1.17.x-CMM2.0.14-x.tar.gz

4. Change the access permission of the log-agent-7.3.0_2.0.x-CMM2.0.14-x.run file to Execute.
5. Run the agent installer:

```
./log-agent-7.3.0_2.0.x-CMM2.0.14-x.run \
  --target "/opt/monasca-log-agent" -- \
  --monasca_log_api_url "http://<cmm-server-ip>:5607/v2.0" \
  --keystone_auth_url "http://<keystone-ip>:35357/v3" \
  --project_name "<project>" \
  --username "<user-name>" \
  --password "<password>" \
  --user_domain_name "<user-domain-name>" \
  --project_domain_name "<user-domain-name>" \
  --hostname "<hostname>"
```

The following parameters must be configured for running the installer. Each value must be enclosed in double quotes (").

- --target. The directory in which the agent is installed. The agent must not be installed in the root user's home directory.
- --monasca_log_api_url. The URL used to access the server where the Monitoring Service is installed. Example: http://192.168.1.6:5607/v2.0

  Note: In CMM2.0.13 the monasca_log_api_url was http://<cmm-server-ip>:5607/v3.0, in CMM2.0.14 it uses the unified API, then it is http://<cmm-server-ip>:5607/v2.0.

- --keystone_auth_url. The URL used to access the server where the OpenStack Keystone service is installed. The service is used for authenticating the user specified in username. It must be a V3 endpoint. Example: http://192.168.1.5:35357/v3
- --project_name. The name of the OpenStack project for which log data is to be retrieved by the agent. Example: monasca.
- --username. The user to be used for authenticating the agent against Keystone, for example monasca-agent. The user specified here must have the monasca-agent role in OpenStack and be assigned to the OpenStack project that is to be monitored by the agent. The project is specified in project_name. It is recommended that this user is used only for configuration purposes and not for actually monitoring services and servers.
- --password. The password of the user specified in username.
- --user_domain_name. The user domain name to be used for user name scoping. For example Default.
- --project_domain_name. The project domain name used for authenticating the agent against OpenStack Keystone. For example Default.
- --hostname. Meta information to be collected with the log data that is retrieved by the agent, for example node1. If --hostname is not specified when running the installer, the host name defined in /etc/hostname is configured as dimension. The meta information defined by a dimension is attached to each log entry. It is represented as one or more

fields in the log management window. For the user who is working with the log data, dimensions provide additional filtering options.

Note: In order to use the default list of OSP16.1 log-paths avoid to pass any log-path in the configuration. The updated list of OSP16.1 log-paths has been taken from: RHOSP16.1 Location of log files for OpenStack services

- If you decided not to use th default list of OSP16.1 log-paths, add the parameter `"<path_to_log_file1>" "<path_to_log_file n >"`. You have to specify the log-paths as absolute paths as in the following example:

```
./log-agent-7.3.0_2.0.x-CMM2.0.14-x.run \
  --target "/opt/monasca-log-agent" -- \
  --monasca_log_api_url "http://<cmm-server-ip>:5607/v2.0" \
  --keystone_auth_url "http://<keystone-ip>:35357/v3" \
  --project_name "<project>" \
  --username "<user-name>" \
  --password "<password>" \
  --user_domain_name "<user-domain-name>" \
  --project_domain_name "<user-domain-name>" \
  --hostname "<hostname>" \
  "/var/log/keystone/keystone.log" "/var/log/neutron/firewall.log"
```

Any number of input file paths can be specified.

The installer creates a `monasca-log-agent.service` file in the `/etc/systemd/system/` directory, and automatically runs the service file to start the agent.

The agent is provided as a LINUX service. A startup script is created that automatically starts the agent each time the machine is booted.

2.5.3 Log Agent Updating the Configuration File

CMM allows you to change the configuration of an agent that is up and running. It might be required, for example, that you want to monitor additional logs due to changes in your environment. To change the agent configuration, you have to edit the agent.conf configuration file.

Proceed as follows:

1. Log in to the OpenStack node on which the agent is installed.
2. To stop the agent, execute the following command:
   `systemctl stop monasca-log-agent`
3. Open the file with your favorite editor. Example:
   `vim /opt/monasca/monasca-log-agent/conf/agent.conf`
4. Adapt the input section, if required. If you want to add files to be monitored, add a corresponding file block. If you want to define dimensions for the log files of a file block, define them with `add_field`. Dimensions allow you to collect meta information with the log

data that is retrieved by the agent. The meta information is attached to each log entry. It is represented as a field in the log management window. For the user who is working with the log data, dimensions provide additional filtering options.

Example:

```
input {
  file {
    path => "/var/log/monasca-agent/*.log"
  }
  file {
    path => "/var/log/monasca-log-agent/*.log"
  }
  file {
    add_field => { "dimensions" => { "service" => "keystone" }}
    path => "/var/log/keystone/*.log"
  }
}
```

5. Adapt the output section, if required. Update the corresponding parameter values. Each value must be enclosed in double quotes (").

   Example:

```
output {
  monasca_log_api {
    monasca_log_api_url => "http://192.168.1.6:5607/v2.0"
    keystone_api_url => "http://192.168.1.5:35357/v3"
    project_name => "monasca"
    username => "monasca-agent"
    password => "password"
    user_domain_name => "default"
    project_domain_name => "default"
    dimensions => ["hostname:myhostname"]
    num_of_logs => 100
    delay => 1
    elapsed_time_sec => 600
    max_data_size_kb => 5120
  }
}
```

6. To start the agent again, execute the following command:
   `systemctl start monasca-log-agent`

The agent is instantly available with the updated configuration settings.

2.5.4 Configure Log Rotation for Log Agent

Introduction

Monasca-log-agent is based on logstash-7.3. Logstash 7.3 is using log4j2. With log4j, log rotation can be configured.
Configuration file for logstash in CMM installation is located in / logstash-7.3.0/config/log4j2.properties.
E.g.: /opt/monasca-log-agent/logstash-7.3.0/config/log4j2.properties

As default, the following configuration is used:

- Rotation: if the file size exceeds 100 MB
- Rotated files will be compressed (gz file is stored)
- A max. of 30 files/day is kept
- Log files are not deleted automatically

How to configure different values

Let's assume the following requirements:

- A max. of 500 MB/day of log information shall be stored
- A new log file shall be created when log file has reached 250 MB
  -> A max. of 2 log files shall be archived every day
- Archived log files older than 30 days shall be deleted

Pls. proceed as follows:

1. Open /opt/monasca-log-agent/logstash-7.3.0/config/log4j2.properties with editor

2. Change the following values:

   appender.rolling.strategy.max = 30 -> appender.rolling.strategy.max = 2
   appender.rolling.policies.size.size = 100MB -> appender.rolling.policies.size.size = 250MB

3. Add the following lines:

   appender.rolling.strategy.action.type = Delete
   appender.rolling.strategy.action.basepath = ${sys:ls.logs}
   appender.rolling.strategy.action.ifLastModified.type = IfLastModified
   appender.rolling.strategy.action.ifLastModified.age = 30d

4. Save your changes

5. Stop monasca-log-agent: systemctl stop monasca-log-agent.service

6. Start monasca-log-agent systemctl start monasca-log-agent.service

Further information

Link to official documentation:
https://logging.apache.org/log4j/2.x/manual/appenders.html#RollingFileAppender

The following document provides an introduction to handling rolling files in log4j2:
https://howtodoinjava.com/log4j2/log4j2-rollingfileappender-example/

2.6 Horizon Plugin (Monasca-UI)

To make the monitoring functionality accessible in OpenStack Horizon, you have to install the Horizon Plugin on the node where the OpenStack Horizon service is deployed. After a successful installation, additional manual configuration steps must be performed before CMM users can access the monitoring and log management functionality in OpenStack Horizon.

2.6.1 Horizon Plugin (Monasca-UI) Installation and Configuration

Before you run the Horizon Plugin installer, make sure that you have made the required OpenStack Keystone preparations.

To install the Horizon Plugin, proceed as follows:

1. Log in as root to the OpenStack node on which the OpenStack Horizon service is installed.
2. Create the directory /opt/monasca-ui inside Horizon container:
   ```
   # podman exec horizon mkdir -p /opt/monasca-ui/
   ```
3. Copy the file monasca-ui-1.17.x-CMM2.0.14-x.tar.gz into Horizon container:
   ```
   # podman cp monasca-ui-1.17.x-CMM2.0.14-x.tar.gz horizon:/opt/monasca-ui/
   ```
4. Enter into Horizon container:
   ```
   # podman exec -it horizon /bin/sh
   ```
5. Extract the archive:
   ```
   $ tar -xzvf /opt/monasca-ui/monasca-ui-1.17.x-CMM2.0.14-x.tar.gz -C
   /opt/monasca-ui/
   ```
6. Install the libraries:
   ```
   $ python3.6 -m pip install --no-index --find-links="/opt/monasca-ui" monasca-
   ui==1.17.x
   ```

   Determine x to be used in monasca-ui version the following way:

   ```
   $ ls /opt/monasca-ui/monasca_ui-1.17.*.whl
   ```

- *Example file:* /opt/monasca-ui/monasca_ui-1.17.2-py2.py3-none-any.whl
- *Version* 1.17.2

Note: If there isn't any .whl file try the following way:

```
$ ls /opt/monasca-ui/monasca-ui-1.17.*.zip
```

- *Example file:* /opt/monasca-ui/monasca-ui-1.17.2.dev4.zip
- *Version:* 1.17.2-dev4

Note: Expected message:

```
WARNING: Running pip install with root privileges is generally not a good
idea.
```

7. Create symbolic links:

```
$ ln -sf /usr/local/lib/python3.6/site-
packages/monitoring/conf/monitoring_policy.json \
/etc/openstack-dashboard/monitoring_policy.json

$ ln -sf /usr/local/lib/python3.6/site-
packages/monitoring/enabled/_50_admin_add_monitoring_panel.py \
/usr/share/openstack-
dashboard/openstack_dashboard/local/enabled/_50_admin_add_monitoring_panel.py

$ ln -sf /usr/local/lib/python3.6/site-
packages/monitoring/config/local_settings.py \
/usr/share/openstack-dashboard \
/openstack_dashboard/local/local_settings.d/_50_monasca_ui_settings.py
```

8. Update configuration. Change the following entries in file:
   /usr/local/lib/python3.6/site-packages/monitoring/config/local_settings.py

```
MONITORING_SERVICES_GROUPS = [
    {'name': _('OpenStack Services'), 'groupBy': 'service'},
    {'name': _('Servers'), 'groupBy': 'hostname'},
    {'name': _('Log Paths'), 'groupBy': 'path'}
]

GRAFANA_URL = getattr(settings, 'GRAFANA_URL', {'<REGION>': '/grafana', })

KIBANA_HOST = getattr(settings, 'KIBANA_HOST', 'http://<CMM-SERVER-IP>:5601/')
```

   Please replace:

   - <REGION> by OpenStack region, e.g.:
     `GRAFANA_URL = getattr(settings, 'GRAFANA_URL', {'regionOne': '/grafana', })`
   - <CMM-SERVER-IP> by IP address of CMM-server, e.g.:
     `KIBANA_HOST = getattr(settings, KIBANA_HOST', 'http://10.140.99.78:5601/')`

9. Update Django settings:

```
$ python3 /usr/share/openstack-dashboard/manage.py collectstatic --noinput
$ python3 /usr/share/openstack-dashboard/manage.py compress --force
```

Note: Expected messages:

```
WARNING:root:"dashboards" and "default_dashboard" in (local_)settings is
DEPRECATED
ERROR:scss.ast:Function not found: twbs-font-path:1
ERROR:scss.compiler:Mixin not found: dropdown-arrow:0
ERROR:scss.compiler:Maximum number of supported selectors in Internet
Explorer (4095) exceeded!
```

Note: If japanese translations are required, pls. execute:

```
$ cd /usr/local/lib/python3.6/site-packages/monitoring
$ /usr/lib/python3.6/site-packages/django/bin/django-admin.py compilemessages
-l ja
```

10. Exit from Horizon container:
    ```
    $ exit
    ```

11. Restart Horizon container:
    ```
    # systemctl restart tripleo_horizon
    ```

12. Go to Horizon in the browser and login as admin, then choose the project monasca: The Monitoring tab is now present.

3 Preparations for Application Operators

Application operators who have booked a virtual machine in OpenStack can monitor their machine with libvirt. Libvirt provides a comprehensive toolkit for managing virtual machines. The toolkit includes checks for virtual machines that run on a hypervisor. As an OpenStack operator, you have to prepare the monitoring environment for your application operators. A Metrics Agent is required for monitoring the hypervisor on the Nova compute node on which the virtual machines are provisioned. On behalf of the application operator, the agent monitors the hypervisor on which it is installed as well as the provisioned virtual machines. The following steps are required:

1. Create an OpenStack role for libvirt monitoring.
2. Install a Metrics Agent on the hypervisor where virtual machines are to be monitored for the application operators.
3. Configure the agent.

For additional information on libvirt monitoring, you can also refer to the *Monasca documentation*.

3.1 Creating an OpenStack Role

As a prerequisite for installing a Metrics Agent, you need to take the following preparations.

- Create the `monitoring-delegate` role in OpenStack Keystone. This role is required for cross-tenant metrics submission. An application operator who has booked a virtual machine must receive only the monitoring data related to his virtual machine. This role enables the agent to submit metrics on behalf of an individual application operator.

  Example command for creating the `monitoring-delegate` role:

  ```
  openstack role create monitoring-delegate
  ```

- Assign the monitoring-delegate role to the OpenStack user you have created for authenticating an agent against OpenStack Keystone, for example monasca-agent. The user name and password of this user must be specified in the agent configuration when the agent is installed. For details, refer to *Creating Projects, Users, and Roles*.

  Example command for assigning the `monitoring-delegate` role to the `monasca-agent` user:

  ```
  openstack role add \
  --project monasca \
  --user monasca-agent monitoring-delegate
  ```

3.2 Installing the Metrics Agent

To enable monitoring for application operators, a Metrics Agent must be installed on the hypervisor that hosts the virtual machines of the application operator.

Enter the credentials of the OpenStack user you want to use for libvirt monitoring in the agent configuration. This user is used for the communication between the Monitoring Service and the agent.

For details on installing a Metrics Agent, refer to *Installing a Metrics Agent on the OpenStack Platform*.

3.3 Configuring the Metrics Agent

The installation of the Metrics Agent includes its initial configuration. To prepare the environment for application operators, you have to reconfigure the agent. It is necessary to explicitly activate the libvirt metrics that the application operators use for monitoring. To reconfigure the agent, proceed as follows:

1. Activate the libvirt metrics. For details, refer to *Activating Additional Metrics*. For an example configuration, refer to the information on `libvirt.yml` in *Additional Metrics*.
2. Make sure that the host name specified for the `nova-compute` service is specified as value for the hostname parameter in the `agent.yaml` configuration file. For this purpose, check the host name for the `nova-compute` service in your environment first. You can execute the following command:
   ```
   nova service-list
   ```

   The `Host` column in the nova-compute line shows the host name for the `nova-compute` service in your environment. 3. To check the `hostname` parameter in the `agent.yaml` configuration file, open the file with your favorite editor. Example:

   ```
   vim /etc/monasca/agent/agent.yaml
   ```

4. If the `hostname` parameter does not correspond to the host name for the `nova-compute` service in your environment (see Step 2), update it.
5. To start the agent again, execute the following command:
   ```
   systemctl start monasca-agent.target
   ```

As soon as the agent is started again, it is available with the updated configuration settings. The application operators can instantly use the libvirt metrics for monitoring.

4 Operation and Maintenance

Regular operation and maintenance includes:

- Managing projects, users, and roles.
- Starting and stopping agents and services.
- Disabling the monitoring of specific metrics in the configuration of a Metrics Agent.
- Disabling the collection of specific log data in the configuration of a Log Agent.
- Configuring data retention for the InfluxDB database.
- Configuring data retention for the Elasticsearch database.
- Removing metrics data from the InfluxDB database.
- Removing log data from the Elasticsearch database.
- Handling log files of agents and services.
- Backup and recovery of databases and dashboards.

4.1 Managing Projects, Users, and Roles

As soon as you have successfully installed CMM, access to the monitoring and log management functionality is enabled for CMM users from OpenStack Horizon. They must fulfill the prerequisites described in the following sections.

Monitoring Service Operator and OpenStack Operator

A system operator who either takes on the role of the Monitoring Service operator or the role of the OpenStack operator must have access to the OpenStack platform as a user with the `monasca-user` role or any other role that is authorized to use the CMM functions. Additional roles are optional.

The system operator must be assigned to the OpenStack project that was prepared for CMM, for example `monasca`.

For details on the project, users, and roles prepared for CMM, refer to *Creating Projects, Users, and Roles*.

Application Operator

An application operator who is responsible for monitoring a VM must have access to the OpenStack platform as a user with the `monasca-user` role or any other role that is authorized to use the CMM functions. Additional roles are optional.

The application operator must be assigned to the project that owns the VM to be monitored.

4.2 Starting and Stopping Agents and Services

System operation and maintenance may require that agents and services are manually stopped and restarted.

The following sections describe:

- Starting and stopping agents and services running in Docker containers on the CMM node.
- Starting and stopping agents installed on an OpenStack node.

Starting and Stopping the CMM Services and Agents

You can start and stop all containerized services and agents with one command. You can also start and stop the agents and services individually.

To start all agents and services, proceed as follows:

1. Log in to the CMM node as a user with root privileges.
2. Go to the installation directory.
3. Run `docker-compose` up as follows:
   ```
   docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml up -d
   ```

> **Note:** Make sure that the OpenStack Keystone service is up when executing the command. Use `-d` to start the containers in the background and leave them running when the command exits. Refer to the *Docker Compose documentation* for additional details on the `docker-compose up` command.

To start individual CMM agents and services, run `docker-compose` up as follows:

```
docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml up -d
<services>
```

Replace `<services>` by the names of the services you want to start. For a list of the service names, refer to *Installing the Monitoring Service*.

Example for starting Kafka and the Notification Engine:

```
docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml up -d kafka
monasca-notification
```

When starting a service that depends on other services, the other services are automatically started. If you start Kafka, for example, Zookeeper is automatically started when executing the example command.

> **Note:** For starting the Metrics Agent, it is required to start the collector service (`agent-collector`) and the forwarder service (`agent-forwarder`).

To stop all agents and services, proceed as follows:

1. Log in to the CMM node as a user with root privileges.
2. Go to the installation directory.
3. Run docker-compose stop as follows:

```
docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml stop
```

To stop individual CMM agents and services, run docker-compose stop as follows:

```
docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml stop
<services>
```

Replace `<services>` by the names of the services you want to stop. For a list of the service names, refer to *Installing the Monitoring Service*.

Example for stopping Kafka and the Notification Engine:

```
docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml stop
kafka monasca-notification
```

Note: For stopping the Metrics Agent, it is required to stop the collector service (`agent-collector`) and the forwarder service (`agent-forwarder`).

To check the status of all agents and services, run docker-compose ps as follows:

```
docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml ps
```

To view the status of an individual container, run docker-compose ps as follows:

```
docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml ps
<services>
```

Replace `<services>` by the names of the services you want to stop. For a list of the service names, refer to *Installing the Monitoring Service*.

Example:

```
docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml ps kafka
monasca-notification
```

Note: Run `docker-compose ps` in a separate shell unless you used `docker-compose up -d` for installing the Monitoring Service.

Starting and Stopping Agents on an OpenStack Node

System operation of your OpenStack platform may require that an agent used for monitoring an OpenStack service is manually stopped and restarted.

To start an agent, proceed as follows:

1. Log in to the OpenStack node on which the agent is installed.
2. To start a Metrics Agent, execute the following command:

```
systemctl start monasca-agent.target
```

To start a Log Agent, execute the following command:

```
systemctl start monasca-log-agent
```

To stop an agent, proceed as follows:

1. Log in to the OpenStack node on which the agent is installed.
2. To stop a Metrics Agent, execute the following command:

```
systemctl stop monasca-agent.target
```

To stop a Log Agent, execute the following command:

```
systemctl stop monasca-log-agent
```

4.3 Data Retention and Cleanup

By default, CMM retains the data stored in the Elasticsearch database and the InfluxDB database for 31 days. Older data is automatically deleted. You can change the default data retention configuration for the databases, if required. In addition to implementing your data retention requirements, you should check for the amount and size of the collected metrics and log data at regular intervals, and delete any unnecessary data to free disk space.

This section describes how to:

- Disable the monitoring of specific metrics in the configuration of a Metrics Agent installed on an OpenStack node.
- Disable the collection of specific log data in the configuration of a Log Agent installed on an OpenStack node.
- Configure data retention for the InfluxDB database.
- Configure data retention for the Elasticsearch database.
- Remove metrics data from the InfluxDB database.
- Remove log data from the Elasticsearch database.

4.3.1 Disabling Metrics for a Metrics Agent on the OpenStack Platform

To disable the monitoring of a specific metrics that is no longer needed in the configuration of a Metrics Agent, you have to delete the corresponding `.yaml` file.

Note: The following step-by-step description refers to a Metrics Agent that you have installed on an OpenStack node. It is assumed that there is no need to disable metrics for the Metrics Agent used for monitoring CMM itself.

Proceed as follows:

1. Log in to the OpenStack node on which the agent is installed.
2. To stop the agent, execute the following command:

   ```
   systemctl stop monasca-collector
   ```

3. Change to the directory that stores the metrics. Example:

   ```
   cd /etc/monasca/agent/conf.d
   ```

4. Delete the `.yaml` file. Example:

   ```
   rm -i process.yaml
   ```

5. To start the agent again, execute the following command:

   ```
   systemctl start monasca-collector
   ```

4.3.2 Disabling Log Data for a Log Agent on the OpenStack Platform

To disable the collection of specific log data that is no longer needed in the configuration of a Log Agent, you have to delete the corresponding entries in the `agent.conf` configuration file.

Note: The following step-by-step description refers to a Log Agent that you have installed on an OpenStack node. It is assumed that there is no need to disable log data for the Log Agent used for monitoring CMM itself.

Proceed as follows:

1. Log in to the OpenStack node on which the agent is installed.
2. To stop the agent, execute the following command:

   ```
   systemctl stop monasca-log-agent
   ```

3. Open the agent configuration file with your favorite editor. Example:

   ```
   vim /opt/monasca-log-agent/conf/agent.conf
   ```

4. In the `input` section, delete the file block for the log data you no longer want to monitor. If you do not want to monitor log data on Keystone any longer, for example, delete the following file block:

   ```
   file {
       path => "/var/log/containers/keystone/keystone/*.log"
   }
   ```

5. To start the agent again, execute the following command:

   ```
   systemctl start monasca-log-agent
   ```

4.3.3 Configuring Metrics Data Retention

Metrics and alarm history data is stored in the InfluxDB database. InfluxDB features data retention mechanisms that allow you to define your data retention policies as required by your monitoring environment. By default, CMM automatically deletes metrics and alarm history data from the database if it is older than 31 days.

Proceed as follows to change the data retention period which was defined when installing the Monitoring Service:

Retention period can be changed by using InfluxDB command line interface, the interactive shell that is provided for the database. Proceed as follows:

1. Log in to the CMM node as a user with root privileges.
2. For connecting to InfluxDB, you need to know the ID or name of the container in which the database is running:

   ```
   docker ps | grep influxd
   ```

   Example output with `c6be4ebebefe` as container ID and `monascadocker_influxdb_1` as container name:

   ```
   c6be4ebebefe   influxdb:1.8-alpine   "/entrypoint.sh infl…"
   19 hours ago  Up 6 seconds   8086/tcp   monascadocker_influxdb_1
   ```

3. Connect to InfluxDB as follows:

   ```
   docker exec -it <container_id> /bin/sh
   influx
   ```

   The output of this command is, for example, as follows:

   ```
   Connected to http://localhost:8086 version 1.8.5
   InfluxDB shell version: 1.8.5
   ```

4. Connect to the InfluxDB database of CMM (`mon`):

   ```
   show databases
   name: databases
   name
   ----
   mon
   _internal

   use mon
   Using database mon
   ```

5. To update your data retention policy, use the `ALTER RETENTION POLICY` command:

   ```
   ALTER RETENTION POLICY <policy_name> ON <db_name> \
   DURATION <duration> [SHARD DURATION <shard_duration>] [DEFAULT]
   ```

Replace <policy_name> by the name of the retention policy you define. Multiple policies can be defined, one of them is set as default. default_mon is the name of the policy that is initially installed by CMM.

Replace <db_name> by the name of your database.

Replace <duration> by the time period after which the data is to be deleted. The minimum retention period is one hour. You can use the following options for specifying the time period:

- m for minutes
- h for hours
- d for days
- w for weeks
- INF for infinite retention

Only single units are supported. This means, for example, that you cannot define a time span of 7230m as 120h 30m.

Use [SHARD DURATION] to define the time span covered by a shard group. When a retention policy is enforced for the database, entire shard groups are removed, and not individual data points. This means that specifying a shorter time range for shard groups than the database retention period allows you to remove data more efficiently. By default, the shard group duration is determined by the database retention period as follows:

- A database retention period < 2 days corresponds to a shard group duration of 1 hour.
- A database retention period >= 2 days and <= 6 months corresponds to a shard group duration of 1 day.
- A database retention period > 6 months corresponds to a shard group duration of 7 days

To deviate from this default, you can replace <shard_duration> by a different time span for shard groups. You can use the same options as for the database retention period, except for INF.

Use [DEFAULT] to set the new retention policy as the default retention policy for your database. Example for deleting data if it is older than 30 days:

```
ALTER RETENTION POLICY default_mon ON mon DURATION 30d DEFAULT
```

For an introduction to the InfluxDB command line interface, refer to the *InfluxDB CLI/Shell documentation*. For details on data retention, refer to *Retention Policy Management*.

4.3.4 Configuring Log Data Retention

Log data is stored in the Elasticsearch database. Elasticsearch stores the data in indices. CMM uses Elasticsearch Curator for managing data retention of these indices. Elasticsearch Curator jobs are automatically run by a Cron daemon that executes scheduled commands. By default, an Elasticsearch index is automatically deleted in CMM if it is older than 31 days. The delete job is executed every day at midnight (UTC). Proceed as follows to change the data retention settings that were defined when installing the Monitoring Service:

1. Log in to the CMM node as a user with root privileges.
2. Go to the installation directory.
3. Stop and remove the `elasticsearch-curator` service. For this purpose, run `docker-compose stop` as follows:

   ```
   docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml stop
   elasticsearch-curator
   ```

4. Remove `elasticsearch-curator` service. For this purpose, run `docker-compose rm` as follows:

   ```
   docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml rm
   elasticsearch-curator
   ```

5. Open the `.env` file, and update the data retention parameter as required. Example for deleting the data after a period of 30 days:

   ```
   # Data retention for Elasticsearch database
   # Delete job is executed every day at 12 a.m. UTC
   MON_ELASTICSEARCH_DATA_RETENTION_DAYS=30
   ```

6. Start the elasticsearch-curator service. For this purpose, run docker-compose up as follows:

   ```
   docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml up -d
   elasticsearch-curator
   ```

As soon as the elasticsearch-curator service is up and running, your new data retention policy takes effect.

For details on Elasticsearch Curator, refer to the *Elasticsearch Curator documentation*. Find details on configuring the Cron daemon, for example, in the *Ubuntu Wiki*.

4.3.5 Removing Metrics Data

Metrics data is stored in the Metrics and Alarms InfluxDB Database. InfluxDB features an SQL-like query language for querying data and performing aggregations on that data.

The Metrics Agent configuration defines the metrics and types of measurement for which data is stored. For each measurement, a so-called series is written to the InfluxDB database. A series consists of a timestamp, the metrics, and the value measured.

Example:

Measurement: cpu.user_perc

Timestamp	Metric	Value	
2015-08-11T11:55:03Z	cpu.user_perc	80	Series 1
2015-08-11T12:55:03Z	cpu.user_perc	70	Series 2
2015-08-11T13:55:03Z	cpu.user_perc	100	Series 3

Removing Metrics Data

Every series can be assigned key tags. In the case of CMM, this is the `_tenant_id tag`. This tag identifies the OpenStack project for which the metrics data has been collected.

From time to time, you may want to delete outdated or unnecessary metrics data from the Metrics and Alarms Database, for example to save space or remove data for metrics you are no longer interested in. To delete data, you use the InfluxDB command line interface, the interactive shell that is provided for the InfluxDB database.

Proceed as follows to delete metrics data from the database:

1. Create a backup of the database. For details, refer to *Backup and Recovery*.
2. Log in to the CMM node as a user with root privileges.
3. Retrieve the name of influxdb container with the following command:

```
# docker ps | grep influxdb
```

The name is the last parameter returned, ending with influxdb_1.
E.g.: monascadocker_influxdb_1

4. Connect to InfluxDB as follows:

```
# docker exec -it <container_id> /bin/sh
# influx
```

The output of this command is, for example, as follows:

```
Connected to http://localhost:8086 version 1.8.5
InfluxDB shell version: 1.8.5
```

5. Connect to the InfluxDB database of CMM (mon):

```
show databases
name: databases
name
----
mon
_ internal

use mon
Using database mon
```

6. Check the outdated or unnecessary data to be deleted.
   - You can view all measurements for a specific project as follows:

     ```
     SHOW MEASUREMENTS WHERE _tenant_id = '<project ID>'
     ```

   - You can view the series for a specific metrics and project, for example, as follows:

     ```
     SHOW SERIES FROM "cpu.user_perc" WHERE _tenant_id = '<project ID>'
     ```

7. Delete the desired data.
   - When a project is no longer relevant or a specific tenant is no longer used, delete all series for the project as follows:

     ```
     DROP SERIES WHERE _tenant_id = '<project ID>'
     ```

     Example:

     ```
     DROP SERIES WHERE _tenant_id = '27620d7ee6e948e29172f1d0950bd6f4'
     ```

   - When a metrics is no longer relevant for a project, delete all series for the specific project and metrics as follows:

     ```
     DROP SERIES FROM "<metrics>" WHERE _tenant_id = '<project ID>'
     ```

     Example:

     ```
     DROP SERIES FROM "cpu.user_perc" WHERE _tenant_id = '27620d7e'
     ```

4.3.6 Removing Log Data

Log data is stored in the Elasticsearch database. Elasticsearch stores the data in indices. One index per day is created for every OpenStack project.

By default, the indices are stored in the following directory on the CMM node:

```
/opt/monasca-containers/elasticsearch/data/nodes/<node-name>/indices
```

Example:

```
/opt/monasca-containers/elasticsearch/data/nodes/0/indices
```

If you want to delete outdated or unnecessary log data from the Elasticsearch database, proceed as follows:

1. Create a backup of the Elasticsearch database. For details, refer to *Backup and Recovery* on.
2. Log in to the CMM node as a user with root privileges.
3. To execute curl commands in the elasticsearch container, run docker-compose exec as follows:
   ```
   docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml exec elasticsearch /bin/bash
   ```
4. Make sure that the data you want to delete exists by executing the following command:
   ```
   curl -XHEAD -i 'http://localhost:<port>/logs-<projectID-date>'
   ```

   For example, if Elasticsearch is listening at port 9200 (default), the ID of the OpenStack project is abc123, and you want to check the index of 2015, July 1st, the command is as follows:

   ```
   curl -XHEAD -i 'http://localhost:9200/logs-abc123-2015-07-01'
   ```

If the HTTP response is `200` , the index exists; if the response is `404` , it does not exist. For listing all indices stored in your Elasticsearch database, you can execute the following command:

```
curl http://localhost:9200/_cat/indices?v
```

5. Delete an index as follows:
   `curl -XDELETE -i 'http://localhost:<port>/logs-<projectID-date>'`

   Example:

   ```
   curl -XDELETE -i 'http://localhost:9200/logs-abc123-2015-07-01'
   ```

   This command either returns an error, such as `IndexMissingException`, or acknowledges the successful deletion of the index.

   Note: Be aware that the `-XDELETE` command immediately deletes the index file!

   Both, for `-XHEAD` and `-XDELETE`, you can use wildcards for processing several indices. For example, you can delete all indices of a specific project for the whole month of July, 2015:

   ```
   curl -XDELETE -i 'http://localhost:9200/logs-abc123-2015-07-*'
   ```

   Note: Take extreme care when using wildcards for the deletion of indices. You could delete all existing indices with one single command!

4.4 Log File Handling

In case of trouble with the CMM services and the docker containers in which they are deployed, you can study the log files to find the reason. The log files are also useful if you need to contact your support organization.

For managing the logs of CMM, the installation automatically deploys a Log Agent on the CMM node. This allows you to instantly use the log management functions for self-monitoring. You can use Kibana as a front-end application to the log data collected from the CMM services and containers. For details, refer to *Log Management*.

You can also use `docker-compose logs` to manually check the log files of the containers deployed on your CMM node.

To check the log files of individual containers, run `docker-compose logs` as follows:

```
docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml logs
<services>
```

Replace `<services>` by the names of the services for which you want to check the log files. For a list of the service names, refer to *Installing the Monitoring Service*.

Example for checking the Kafka and Kibana log files:

```
docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml logs kafka
kibana
```

For more information on `docker-compose logs`, you can execute the following command:

```
docker-compose logs --help
```

Data Retention for Docker Log Files

By default, Docker log files are not rotated or removed once the containers are started. It is recommended that you configure data retention for the log files so that they do not become too big.

You have the following options:

- You configure data retention for the entire Docker engine, thus for all containers. For details, refer to the *Docker documentation*.
- You configure data retention individually for each Docker container by adding a `max-size` entry to the Docker Compose YAML files. For details, refer to the *Docker Compose documentation*.

4.5 Backup and Recovery

Typical tasks of the Monitoring Service operator are to make regular backups, particularly of the data created during operation.

At regular intervals, you should make a backup of all:

- Databases.
- Monitoring and log dashboards you have created and saved.

CMM does not offer integrated backup and recovery mechanisms. Instead, you use the mechanisms and procedures of the individual components.

4.5.1 Databases

You need to create regular backups of the following databases on the host where the Monitoring Service is installed:

- Elasticsearch database
- InfluxDB database
- MySQL database

Note: It is recommended that backup and restore operations for databases are carried out by experienced operators only.

Elasticsearch Database

Backup

For backing up and restoring your Elasticsearch database, you can use the Snapshot and Restore module of Elasticsearch. You have to create a snapshot repository, before creating a regular backup of your database.

To create a snapshot repository, proceed as follows:

1. Log in to the CMM node as a user with root privileges.
2. To execute `curl` commands in the elasticsearch container, run `docker-compose exec` as follows:

   ```
   docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml exec
   elasticsearch /bin/bash
   ```

3. Create the snapshot repository. Example:

   ```
   curl -XPUT http://localhost:9200/_snapshot/my_backup -H'Content-Type:
   application/json' -d '{
    "type": "fs",
    "settings": {
    "location": "/usr/share/elasticsearch/backup/my_backup",
    "compress": true
    }
   }'
   ```

   The example registers a shared file system repository (with `"type": "fs"`) that uses the `/elasticsearch_backup` directory for storing snapshots. The parent directory for storing snapshots was configured during the installation of the Monitoring Service (`MON_BACKUP_DIR` parameter in `.env` file). It must have been manually mounted before creating the snapshot. `compress` is turned on to compress the metadata files.

4. Check whether the repository was created successfully. Example:

   ```
   curl -XGET http://localhost:9200/_snapshot/my_backup
   ```

   Example response for a successfully created repository:

   ```
   {
     "my_backup": {
       "type": "fs",
       "settings": {
         "compress": "true",
         "location": "/usr/share/elasticsearch/backup/my_backup"
       }
     }
   }
   ```

5. Exit the elasticsearch container:

   ```
   exit
   ```

As soon as you have successfully created the repository, you can create regular backups of the database. To create a backup, proceed as follows:

1. Log in to the CMM node as a user with root privileges.
2. Stop all CMM agents and services:

   ```
   docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml stop
   ```

3. Start the elasticsearch service:

   ```
   docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml up -d
   elasticsearch
   ```

4. To execute curl commands in the elasticsearch container, run docker-compose exec as follows:

   ```
   docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml exec
   elasticsearch /bin/bash
   ```

5. Create a snapshot of your database that contains all indices. A repository can contain multiple snapshots of the same database. The name of a snapshot must be unique within the snapshots created for your database. Example:

   ```
   curl -XPUT
   http://localhost:9200/_snapshot/my_backup/snapshot_1?wait_for_completion=true
   ```

   The example creates a snapshot named `snapshot_1` for all indices in the `my_backup` repository.

6. Exit the `elasticsearch` container:

   ```
   exit
   ```

7. For easier portability and storage, you can create an archive file for the snapshot you have created. To configure the security context, the `--selinux` parameter must be specified.

   ```
   tar --selinux -zcvf es_snapshot_1.tar.gz -C \
     <MON_BACKUP_DIR>/elasticsearch_backup/ my_backup
   ```

   Replace `<MON_BACKUP_DIR>` with its value in `.env`. Example:

   ```
   tar --selinux -zcvf es_snapshot_1.tar.gz -C \
     /mount/backup/elasticsearch_backup/ my_backup
   ```

8. Stop the `elasticsearch` service:

   ```
   docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml stop
   elasticsearch
   ```

9. Start all CMM agents and services:

   ```
   docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml up -d
   ```

Restore

To restore a database instance, proceed as follows:

1. Log in to the CMM node as a user with root privileges.
2. Make sure that the backup of your database is available in the directory that was mounted before creating the snapshot.
3. Stop all CMM agents and services:

   ```
   docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml stop
   ```

4. Start the `elasticsearch` service:

   ```
   docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml up -d
   elasticsearch
   ```

5. To execute `curl` commands in the `elasticsearch` container, run `docker-compose` exec as follows:

   ```
   docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml exec
   elasticsearch /bin/bash
   ```

6. Before restoring the database instance, check whether the snapshot repository already exists: Example:

   ```
   curl -XGET http://localhost:9200/_snapshot/my_backup
   ```

   Example response for an existing repository:

   ```
   {
     "my_backup": {
       "type": "fs",
       "settings": {
         "compress": "true",
         "location": "/usr/share/elasticsearch/backup/my_backup"
       }
     }
   }
   ```

7. If there is no snapshot repository, you have to create it. Example:

   ```
   curl -XPUT http://localhost:9200/_snapshot/my_backup -H'Content-Type:
   application/json' -d '{
    "type": "fs",
    "settings": {
    "location": "/usr/share/elasticsearch/backup/my_backup",
    "compress": true
    }
   }'
   ```

   If the snapshot repository is created successfully, Elasticsearch returns `{"acknowledged":true}`

8. Close all indices of your database. Example:

   ```
   curl -XPOST http://localhost:9200/_all/_close
   ```

   If the call is successful, Elasticsearch returns `{"acknowledged":true}`.

9. Restore all indices from the snapshot you have created. Example:
   ```
   curl -XPOST http://localhost:9200/_snapshot/my_backup/snapshot_1/_restore
   ```
   The example restores all indices from `snapshot_1` that is stored in the `my_backup` repository. If the call is successful, Elasticsearch returns `{"accepted":true}`.

10. Now, only the data restored from is visible in your repository.
    If you want to see all data created after the backup has been taken, one additional step is required:
    Restore all indices from the snapshot you have created. Example:
    ```
    curl -XPOST http://localhost:9200/_all/_open
    ```

11. Exit the elasticsearch container:
    ```
    exit
    ```

12. Stop the `elasticsearch` service:
    ```
    docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml stop
    elasticsearch
    ```

13. Start all CMM agents and services:
    ```
    docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml up -d
    ```

For additional information on backing up and restoring an Elasticsearch database, refer to the *Elasticsearch documentation*.

InfluxDB Database

To create a backup of the database, proceed as follows:

1. Log in to the CMM node as a user with root privileges.
2. Stop all CMM agents and services:
   ```
   docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml stop
   ```

3. Start the `influxdb` service:
   ```
   docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml up -d
   influxdb
   ```

4. For backing up the InfluxDB database, you need to know the ID of the container in which the database is running:
   ```
   docker ps | grep influxd
   ```
   Example output with `c6be4ebebefe` as container ID:
   ```
   c6be4ebebefe influxdb:1.8-alpine "/entrypoint.sh infl..."
   19 hours ago Up 6 seconds 8086/tcp monascadocker_influxdb_1
   ```

5. Back up the database:

```
docker exec <container_id> influxd backup -database <db_name>
"/influxdb_backup/<name>.backup"
```

Replace <container_id> by the ID of the container in which the database is running, <db_name> by the name of the database and <name> by the name of the backup file to be created.

The following example creates a backup of the mon database, running in container 07e9007e0be8, in mon.backup:

```
docker exec c6be4ebebefe influxd backup -database mon
"/influxdb_backup/mon.backup"
```

6. Stop the influxdb service:

```
docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml stop
influxdb
```

7. Start all CMM agents and services:

```
docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml up -d
```

To restore the database, proceed as follows:

1. Log in to the CMM node as a user with root privileges.
2. Stop all CMM agents and services:

```
docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml stop
```

3. Make sure that the backup file is available in the <path_to_backup_dir>/influxdb_backup directory. <path_to_backup_dir> is the default volume you configured for backing up the databases during the installation of the Monitoring Service (MON_BACKUP_DIR parameter in .env file).

4. With SELinux enabled, make sure that svirt_sandbox_file_t or container_file_t is set as security context type for the backup file.

Example with MON_BACKUP_DIR set to /mount/backup and svirt_sandbox_file_t set as security context type:

```
ls -lZ /mount/backup/influxdb_backup/
drwx------. root root
system_u:object_r:svirt_sandbox_file_t:s0:c335,c530 mon.backup
```

5. Restore the database to an ephemeral container:

```
docker run --rm --entrypoint /bin/ash \
 -v <path_to_data_dir>/influxdb:/var/lib/influxdb:Z \
 -v <path_to_backup_dir>/influxdb_backup:/influxdb_backup:Z \
 influxdb:1.8-alpine \
 -c "influxd restore -metadir /var/lib/influxdb/meta \
 -datadir /var/lib/influxdb/data \
 -database <db_name> /influxdb_backup/<name>.backup"
```

Replace the variables as follows:

- <path_to_data_dir> by the default volume you configured for the data directories during the installation of the Monitoring Service (MON_DOCKER_VOL_ROOT parameter in .env file).
- <path_to_backup_dir> by the default volume you configured for backing up the databases.
- <db_name> by the name of the database.
- <name> by the name of the backup file to be restored.

With SELinux enabled, :Z must be specified for the -v option.

6. Start all CMM agents and services:

```
docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml up -d
```

MySQL Database

For backing up and restoring your MySQL database, you can use the mysqldump utility program. mysqldump performs a logical backup that produces a set of SQL statements. These statements can later be executed to restore the database.

To back up your MySQL database, proceed as follows:

1. Log in to the CMM node as a user with root privileges.
2. Stop all CMM agents and services:

```
docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml stop
```

3. Start the mysql service:

```
docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml up -d
mysql
```

4. Backup the database. To create the backup from your local machine, execute the following command:

```
docker exec <container_id> mysqldump -u root --password=secretmysql <db_name>
> <backup_dir>/<name> && chmod 700 <backup_dir>/<name>
```

Replace <container_id> by the ID of the container in which the database is running, <db_name> by the name of the database, <backup_dir> by the directory you have mounted for storing the backup file, and <name> by the name of the backup file to be created. For the --password parameter, check the MYSQL_ROOT_PASSWORD specified in the docker-compose-metric.yml file.

The following example creates a backup of the mon database, running in container 07e9007e0be8, in /backup/mon.sql:

```
docker exec 07e9007e0be8 mysqldump -u root --password=secretmysql mon >
/backup/mon.sql &&
chmod 700 /backup/mon.sql
```

5. Check whether the backup was created successfully.
6. Stop the `mysql` service:

   ```
   docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml stop
   mysql
   ```

7. Start all CMM agents and services:

   ```
   docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml up -d
   ```

To restore your MySQL database, proceed as follows:

1. Log in to the CMM node as a user with root privileges.
2. Stop all CMM agents and services:

   ```
   docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml stop
   ```

3. Start the mysql service:

   ```
   docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml up -d
   mysql
   ```

4. Restore the database:

   ```
   cat <backup_dir>/<name> | docker exec -i <container_id> mysql -u root --
   password=secretmysql <db_name>
   ```

   Replace `<backup_dir>` by the directory where the backup file is stored, `<name>` by the name of the file, `<container_id>` by the ID of the container in which the database is running, and `<db_name>` by the name of the database.

   For the `--password` parameter, check the `MYSQL_ROOT_PASSWORD` specified in the `docker-compose-metric.yml` file.

   The following example restores the mon database, running in container `07e9007e0be8`, from `/ backup/mon.sql`:

   ```
   cat /backup/mon.sql | docker exec -i 07e9007e0be8 mysql -u root --
   password=secretmysql mon
   ```

5. Stop the `mysql` service:

   ```
   docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml stop
   mysql
   ```

6. Start all CMM agents and services:

   ```
   docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml up -d
   ```

For additional information on backing up and restoring a MySQL database with `mysqldump`, refer to the *MySQL documentation*.

4.5.2 Dashboards

Kibana can persist customized dashboard designs to the Elasticsearch database, and allows you to recall them. For details on saving, loading, and sharing dashboards for log management, refer to the *Kibana documentation*.

As a Grafana administrator, you are allowed to export dashboards to a JSON file, and to re-import them when necessary.

Export

Use the administrator credentials defined in the `.env` file for the installation of the Monitoring Service to directly access Grafana and export dashboards. For backing up and restoring the exported dashboards, you can then use the standard mechanisms of your file system. For details on exporting monitoring dashboards, refer to the *Grafana documentation*.

Import

Note: Due to an existing bug in Kibana, the exported JSON file can only be imported with API. Otherwise, when attempting to import from UI, an error message is shown, including information of "Unsupported Media Type".

Use following `curl` command to import the JSON data from API:

```
curl –XPOST
"<ip-address of kibana server>:5601/api/saved_objects/_import?overwrite=true" -H
"kbn-xsrf: true" --form file=@<file to be imported>
```

Replace `<ip-address of kibana server>` with the IP address of the server where Kibana is running and `<file to be imported>` with the local path of the file to be imported.

5 Monitoring

CMM offers various monitoring features which support you in proactively managing your cloud resources. A large number of metrics in combination with early warnings about problems and outages assist you in analyzing and troubleshooting any issue you encounter in your environment. The monitoring features of CMM include:

- A monitoring overview which allows you to access all monitoring information.
- Metrics dashboards for visualizing your monitoring data.
- Alerting features for monitoring.

In the following sections, you will find information on the monitoring overview and the metrics dashboards as well as details on how to define and handle alarms and notifications.

Accessing CMM

For accessing CMM and performing monitoring tasks, the following prerequisites must be fulfilled:

- You must have access to the OpenStack platform as a user with the `monasca-user` role or any other role that is authorized to use the CMM monitoring functions. Additional roles are optional.
- You must be assigned to the OpenStack project you want to monitor.

Log in to OpenStack Horizon with your user name and password. The functions you can use in OpenStack Horizon depend on your access permissions.

The CMM functionality is available on the **Monitoring** tab. It provides access to the monitoring data of all projects to which you are assigned. Before you start, select the project you want to work on.

5.1 Overview

CMM provides one convenient access point to your monitoring data. Use **Monitoring > Overview** to keep track of your services and servers and quickly check their status. The overview also indicates any irregularities in the log data of the system components you are monitoring.

On the **Overview** page, you can:

- View the status of your services, servers, and log data at a glance. As soon as you have defined an alarm for a service, a server, or log data and metrics data has been received, there is status information displayed on the **Overview** page. For details on the status information, refer to *Status of Services, Servers, and Log Data*. For details on defining alarms, refer to *Defining Alarms*.
- Access preconfigured dashboards that visualize your metrics data. For details, refer to *Viewing Metrics Data*.
- Access the CMM log management functionality. For details, refer to *Log Management*.

5.2 Viewing Metrics Data

The user interface for monitoring your environment integrates with Grafana, an open-source application for visualizing large-scale metrics data. Use the **Grafana Home** option located along the top of the **Overview** page to access Grafana.

CMM ships with preconfigured metrics dashboards that allow you to instantly monitor your enviroment:

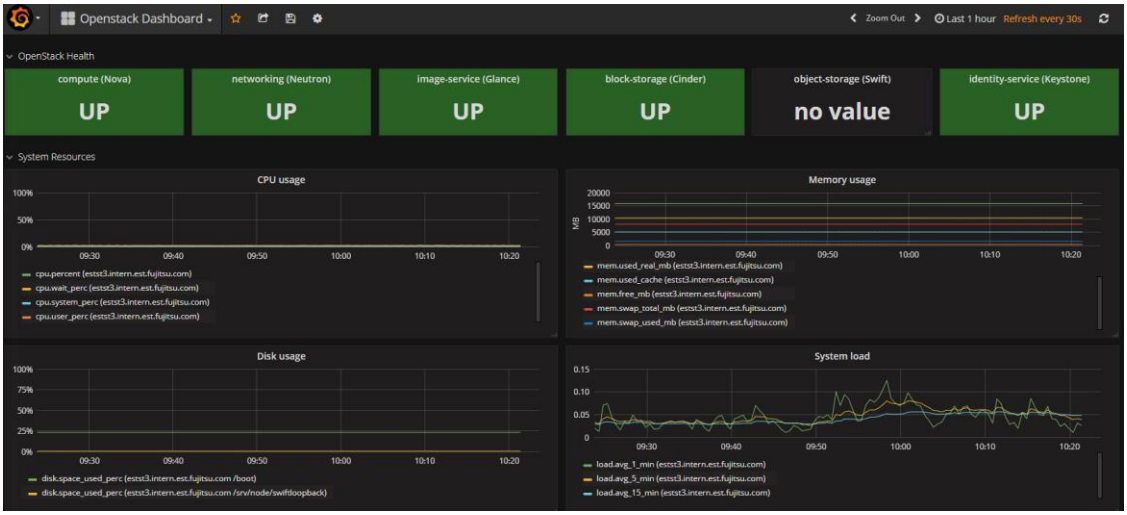
- The **OpenStack Services** dashboard visualizes the metrics data from your OpenStack node.
- The **OpenStack Hypervisor** and **OpenStack VMs** dashboards visualize the metrics data from your OpenStack hypervisor (aggregated metrics) and the metrics data from the VMs (per tenant metrics) respectively. To display the correct data on the OpenStack VMs dashboard you need to be logged in OSP with the tenant of the VMs.
- The **CMM Stats** dashboard visualizes the metrics data from the node on which the Monitoring Service is installed.

You are authorized to view the metrics data that is displayed.

The Grafana administrator created during the installation of the Monitoring Service is allowed to export dashboards to a JSON file, and to re-import them when necessary. The Grafana administrator credentials defined in the `.env` file must be used for accessing Grafana. Contact your FUJITSU support organization for information on additional preparations that must be taken before exporting and importing JSON files in Grafana.

OpenStack Services

To view the metrics data on the OpenStack services and the node on which they are deployed, select the **OpenStack Services** dashboard from the **Home** menu located at the top part of the Grafana window.



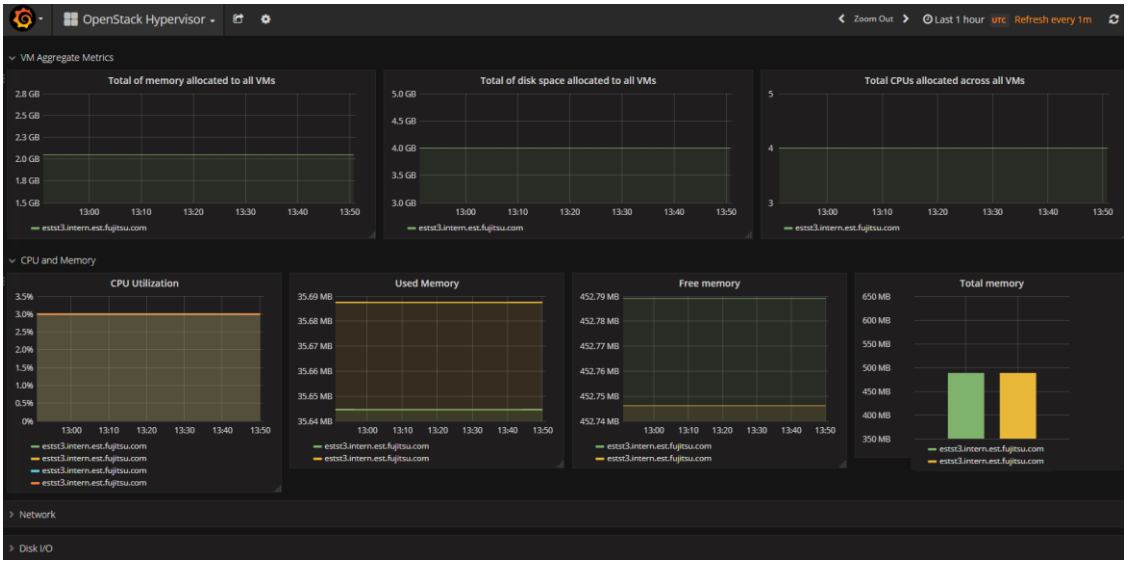
OpenStack

The preconfigured dashboard shows the following:

- Status of the main OpenStack services (`UP` or `DOWN`). Information on Nova, Neutron, Glance, Cinder, Swift, and Keystone is displayed.
- Information on the system resources. CPU usage: The percentage of time the CPU is used in total (`cpu.percent`), at user level (`cpu.user_perc`), and at system level (`cpu.system_perc`), as well as the percentage of time the CPU is idle when no I/O requests are in progress (`cpu.wait_perc`).
- Memory usage: The number of megabytes of total memory (`mem.total_mb`), free memory (`mem.free_mb`), total swap memory (`mem.swap_total_mb`), and used swap memory (`mem.swap_used_mb`). In addition, it shows the number of megabytes used for the page cache (`mem.used_cache`), and the number of megabytes of used memory minus the memory used for buffers and the page cache (`mem.used_real_mb`).
- The percentage of disk space that is being used on a device (`disk.space_used_perc`).
- The system load over different periods (`load.avg_1_min`, `load.avg_5_min`, and `load.avg_15_min`).
- Network usage: The number of network bytes received and sent per second (`net.in_bytes_sec` and `net.out_bytes_sec`).

OpenStack Hypervisor

To view the metrics data on your OpenStack hypervisor and the aggregated metrics data from the VMs running on the hypervisor, select the `OpenStack Hypervisor` dashboard from the `Home` menu located at the top part of the Grafana window.



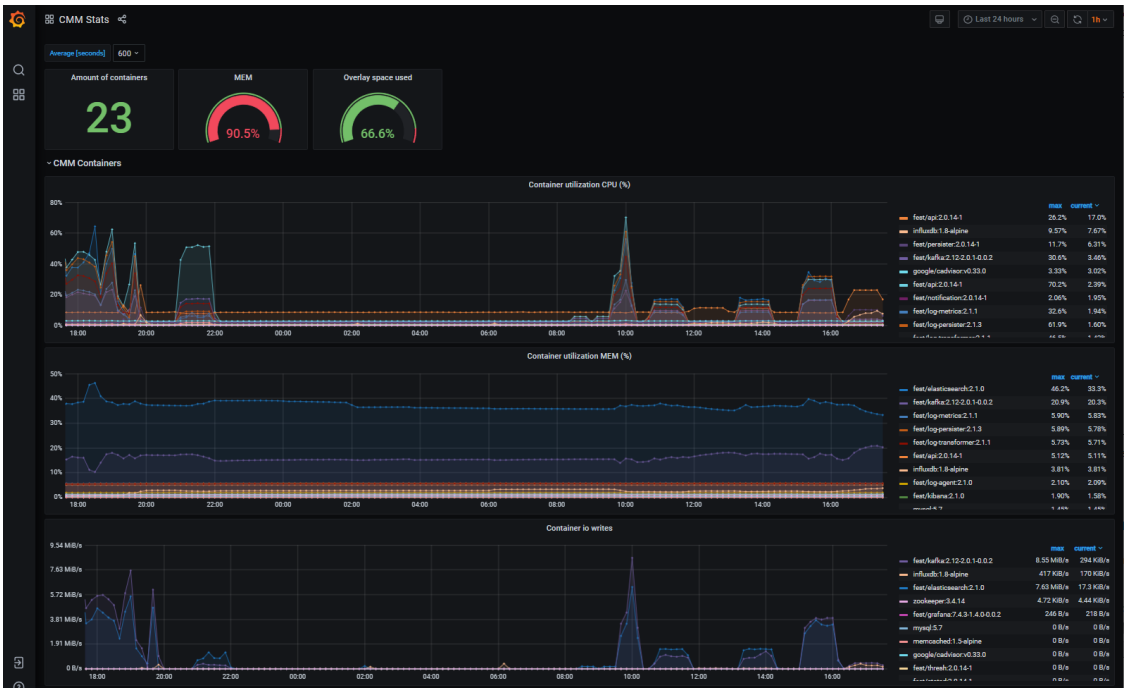
OpenStack

The preconfigured dashboard shows the following:

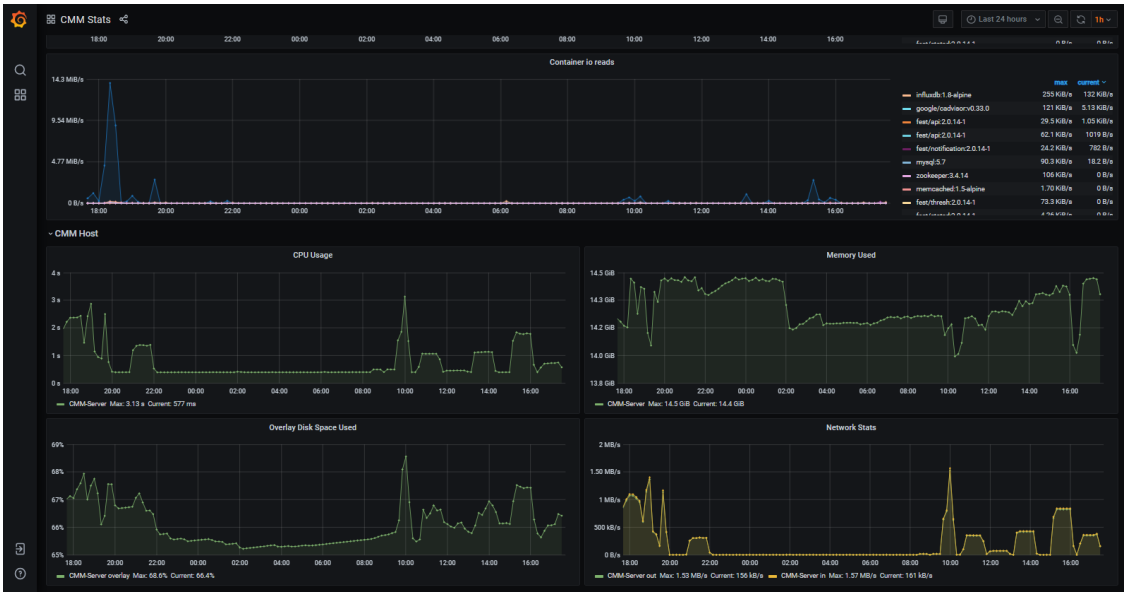
- Aggregated metrics: The number of megabytes of total memory allocated to the VMs (`nova.vm.mem.total_allocated_mb`), the number of gigabytes of total disk space allocated to the VMs (`nova.vm.disk.total_allocated_gb`), and the total number of CPUs allocated to the VMs (`nova.vm.cpu.total_allocated`).
- CPU and memory usage: The percentage of time the CPU is used by the VMs (`vm.cpu.utilization_perc`), the number of megabytes of memory used by the VMs (`vm.mem.used_mb`), the number of megabytes of free memory (`vm.mem.free_mb`), and the number of megabytes of total memory allocated to the VMs (`vm.mem.total_mb`).
- Network usage: The number of network bytes received and sent per second (`vm.net.in_bytes_sec` and `vm.net.out_bytes_sec`), as well as the total number of network bytes received and sent (`vm.net.in_bytes` and `vm.net.out_bytes`).
- Disk I/O: The number of bytes written per second (`vm.io.write_bytes_sec`), and the number of bytes read per second (`vm.io.read_bytes_sec`).

CMM Stats

To view the metrics data on CMM itself, select the **CMM Stats** Dashboard from the **Home** menu located at the top part of the Grafana window.



CMM



CMM

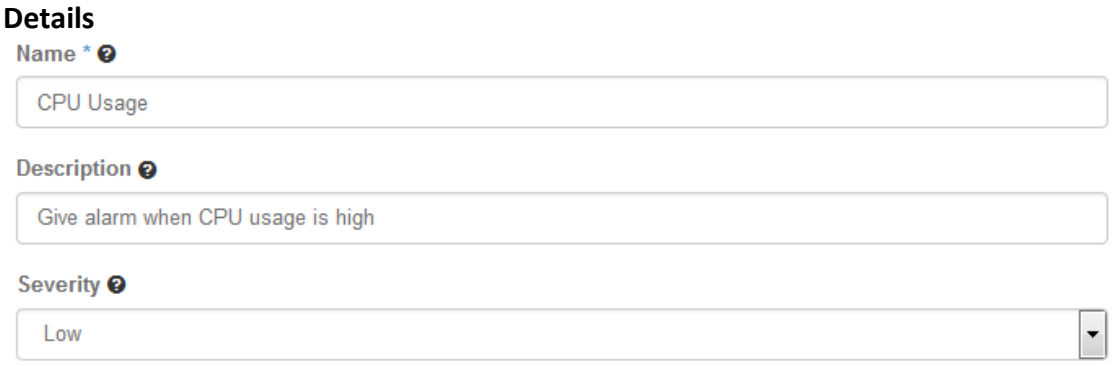
The preconfigured dashboard shows the following information on the CMM system resources:

- First section: Quick view on current status:
  - Number of containers running (`container.running_count`)
  - Total Memory usage (`mem.used_perc`)
  - Total Usage of Docker overlay space (`fs.usage_perc`)
- **CMM Containers**: Utilization by individual containers:
  - Container utilization CPU: CPU usage (`container.cpu.utilization_perc`).
  - Container utilization MEM (%): Memory usage (`container.mem.used_perc`).
  - Container io writes: write operations (`container.io.write_bytes_sec`)
  - Container io reads: read operations (`container.io.read_bytes_sec`)
- **CMM Host**: Utilization of CMM server:
  - CPU usage: The time the CPU is used in total (`cpu.total_time_sec`).
  - Memory Used: Memory used (`mem.used_bytes`)
  - Overlay Disk Space Used: Available overlay disk space used (`fs.usage_perc`)
  - Network stats: The number of network bytes received and sent per second (`net.in_bytes_sec` and `net.out_bytes_sec`).

5.3 Defining Alarms

You have to define alarms to monitor your resources. An alarm definition specifies the metrics to be collected and the threshold at which an alarm is to be triggered for a resource. By default, an alarm definition is evaluated over a succession of one-minute periods. If the specified threshold is reached or exceeded within one such period, the alarm is triggered and notifications can be sent to inform users. The alarm definition is re-evaluated in each subsequent time period. To create, edit, and delete alarms, use **Monitoring > Alarm Definitions**. The elements that define an alarm

are grouped into **Details, Expression**, and **Notifications**. They are described in the following sections.



Details

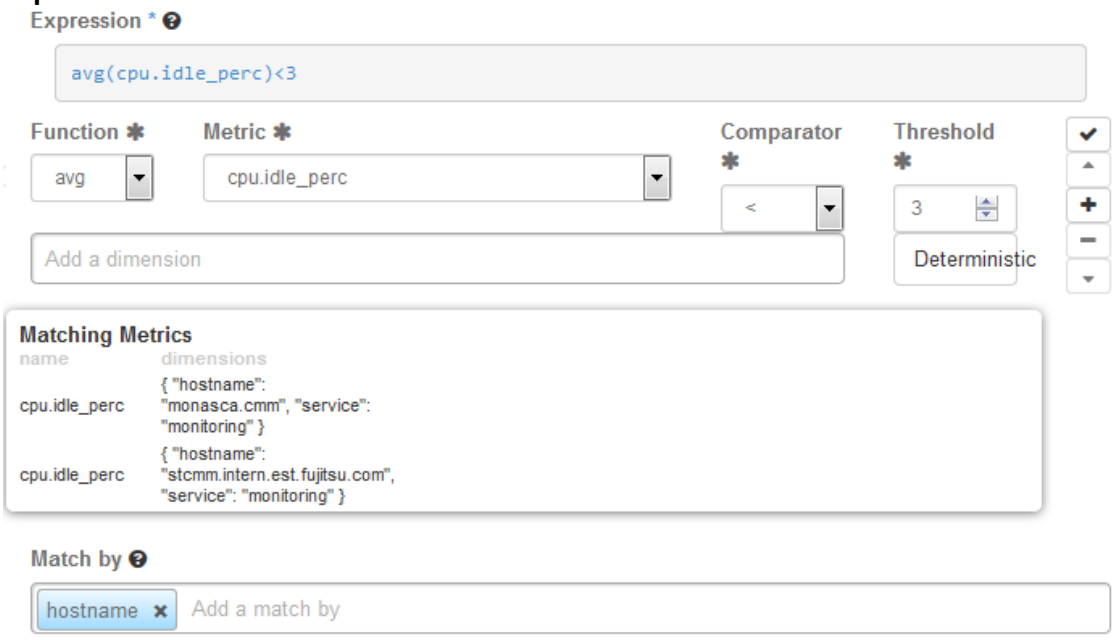
For an alarm definition, you specify the following details:

- **Name**. Mandatory identifier of the alarm. The name must be unique within the project for which you define the alarm.
- **Description**. Optional. A short description that outlines the purpose of the alarm.
- **Severity**. The following severities for an alarm are supported: **Low** (default), **Medium** , **High** , or **Critical**.

  The severity affects the status information on the **Overview** page. If an alarm that is defined as **Critical** is triggered, the corresponding resource is displayed in a red box. If an alarm that is defined as **Low** , **Medium** , or **High** is triggered, the corresponding resource is displayed in a yellow box.

  The severity level is subjective. Choose a level that is appropriate for prioritizing the alarms in your environment.

Expression



Expression

The expression defines how to evaluate the metrics. The expression syntax is based on a simple expressive grammar. For details, refer to the *Monasca API documentation*.

To handle a large variety of monitoring requirements, you can create either simple alarm definitions that refer to one metrics only, or compound alarm definitions that combine multiple metrics in one expression.

Example for a simple alarm definition that checks whether the system-level load of the CPU exceeds a threshold of 90 percent:

```
cpu.system_perc{hostname=monasca} > 90
```

To define an alarm expression, proceed as follows:

1. Select the metrics to be evaluated.
2. Select a statistical function for the metrics: `min` to monitor the minimum values, `max` to monitor the maximum values, `sum` to monitor the sum of the values, `count` for the monitored number, `avg` for the arithmetic average, or `last` for the most recent value.
3. Enter one or multiple dimensions in the **Add a dimension** field to further qualify the metrics.

   Dimensions filter the data to be monitored. They narrow down the evaluation to specific entities. Each dimension consists of a key/value pair that allows for a flexible and concise description of the data to be monitored, for example region, availability zone, service tier, or resource ID.

The dimensions available for the selected metrics are displayed in the **Matching Metrics** section. Type the name of the key you want to associate with the metrics in the **Add a dimension** field. You are offered a selection for adding the required key/value pair.

4. Enter the threshold value at which an alarm is to be triggered, and combine it with a relational operator <, >, <=, or >=.

   The unit of the threshold value is related to the metrics for which you define the threshold, for example the unit is percentage for `cpu.idle_perc` or MB for `disk.total_used_space_mb`.

5. Switch on the **Deterministic** option if you evaluate a metrics for which data is received only sporadically. The option should be switched on, for example, for all log metrics. This ensures that the alarm status is `OK` and displayed as a green box on the **Overview** page although metrics data has not yet been received.

   Do not switch on the option if you evaluate a metrics for which data is received regularly. This ensures that you instantly notice, for example, that a host machine is offline and that there is no metrics data for the agent to collect. On the **Overview** page, the alarm status therefore changes from OK to `UNDETERMINED` and is displayed as a white box.

6. Enter one or multiple dimensions in the **Match by** field if you want these dimensions to be taken into account for triggering alarms.

   Example: If you enter `hostname` as a dimension, individual alarms will be created for each host machine on which metrics data is collected. The expression you have defined is not evaluated as a whole but individually for each host machine in your environment.

   If **Match by** is set to a dimension, the number of alarms depends on the number of dimension values on which metrics data is received. An empty **Match by** field results in exactly one alarm.

   To enter a dimension, you can simply type the name of the dimension in the **Match by** field.

   The dimensions you enter cannot be changed once the alarm definition is saved.

7. Build a compound alarm definition to combine multiple metrics in one expression. Using the logical operators AND or OR, any number of expressions can be combined.

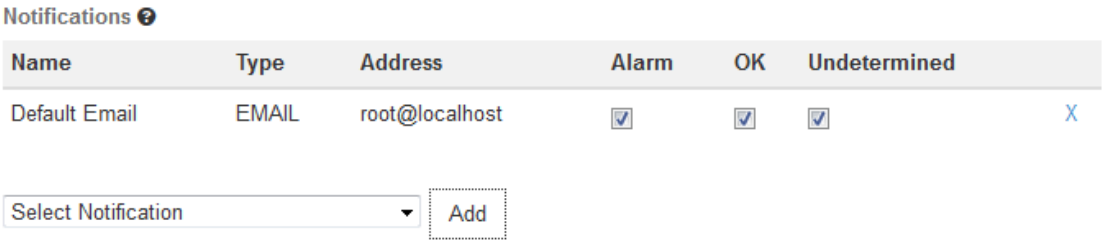
   Use the **Add** button to append an expression, and choose either AND or OR as the **Operator** to connect it to the one you have already defined. Proceed with the second expression as described in Step 1 to Step 6 above.

   The following options are provided for creating and organizing compound alarm definitions:

   - Append an additional expression using the **Add** button.
   - Finish editing an appended expression using the **Submit** button.
   - Delete an appended expression using the **Remove** button.
   - Change the position of an appended expression using the **Up** or **Down** button.

Notes: * `Function`, `Metric`, `Comparator` and `Threshold` are required fields and must not be empty. * You can also edit the expression syntax directly. For this purpose, save your alarm definition and update it using the **Edit Alarm Definition option**. When updating the alarm definition, you can also change the default time period for each alarm definition evaluation. For syntax details, refer to the Monasca API documentation on *Alarm Definition Expressions*. * Restrictions apply to usage of time/times and compound alarm definitions. Please refer to *Release Notes* for details.

Notifications



Notification

You can enable notifications for an alarm definition. As soon as an alarm is triggered, the enabled notifications will be sent.

The **Notifications** tab allows you to select the notifications from the ones that are defined in your environment. For a selected notification, you specify whether you want to send it for a status transition to **Alarm** , **OK** , and/or **Undetermined**.

For details on defining notifications, refer to *Defining Notifications*. For details on alarm statuses, refer to *Status of Services, Servers, and Log Data*.

5.4 Defining Notifications

Notifications define how users are informed when a threshold value defined for an alarm is reached or exceeded. In an alarm definition, you can assign one or multiple notifications.

For a notification, you specify the following elements:

- **Name**. A unique identifier of the notification. The name is offered for selection when defining an alarm.
- **Type**. The notification method to be used. `Email`, `Pagerduty` or `WebHook` can be selected, provided that the methods were enabled when installing the Monitoring Service.
- **Address**.
  For `Email`, the email address to be notified when an alarm is triggered.

Note: Generic top-level domains such as business domain names are not supported in email addresses (for example `user@xyz.company`).

For `Pagerduty`, the Integration Key of Pagerduty `Events API v1`. Example:

`2e8b50c787ab4405d0a0d6ad89740d75`
See Monasca-Notification Pagerduty Plugin for more info.

For `WebHook`, the WebHook URL to be loaded when an alarm is triggered.

- **Period**. For `WebHook` only. The integer value indicates how often a notification is to be resent.

To create, edit, and delete notifications, use **Monitoring > Notifications**.

5.5 Status of Services, Servers, and Log Data

The following alarm statuses are distinguished when an alarm definition has been evaluated:

- `Alarm`. The alarm expression has evaluated to `true`. An alarm has been triggered for the cloud resource.
- `OK`. The alarm expression has evaluated to false. There is no need to trigger an alarm.
- `Undetermined`. No metrics data has been received within the defined time period.

As soon as you have defined an alarm for a resource, status information is displayed for it on the **Overview** page:

The color of the boxes in the three sections indicates the status:

- A green box for a service or server indicates that it is up and running. A green box for a log path indicates that a defined threshold for errors or warnings, for example, has not yet been reached or exceeded. There are alarms defined for the services, servers, or log paths, but no alarms have been triggered.
- A red box for a service, server, or log path indicates that there is a severe problem that needs to be checked. One or multiple alarms defined for a service, a server, or log data have been triggered.
- A yellow box indicates a problem. One or multiple alarms have already been triggered, however, the severity of these alarms is low.
- A white box indicates that though alarms have indeed been defined, metrics data has not yet been received.

The status information on the **Overview** page results from one or multiple alarms that have been defined for the corresponding resource. If multiple alarms are defined, the severity of the individual alarms controls the status color.

You can click a resource on the **Overview** page to display details on the related alarms. The details include the status of each alarm and the expression that is evaluated. For each alarm, you can drill down into the alarm history. To narrow down the problem, the history presents detailed information on the status transitions.

6 Log Management

For managing the log data of your services and the virtual and physical servers on which they are provisioned, CMM integrates with Kibana, an open-source analytics and visualization platform. CMM uses Kibana as a front-end application to the log data held in the Elasticsearch database.

Kibana allows you to easily understand large data volumes. Based on the data that is stored in Elasticsearch indices, you can perform advanced data analysis and visualize your log data in a variety of charts, tables, or maps. Changes to the Elasticsearch indices are displayed in CMM in real-time.

The log management features of CMM include:

- Features for searching, visualizing, and analyzing the log data.
- Alerting features for monitoring.

In the following sections, you will find information on the log management window where you search, visualize, and analyze your log data, as well as details on how to use the alerting features.

Accessing CMM

For accessing CMM and performing log management tasks, the following prerequisites must be fulfilled:

- You must have access to the OpenStack platform as a user with the `monasca-user` role, or any other role that is authorized to use the CMM monitoring functions. Additional roles are optional.
- You must be assigned to the OpenStack project you want to monitor.

Log in to OpenStack Horizon with your user name and password. The functions you can use in OpenStack Horizon depend on your access permissions.

The CMM functionality is available on the **Monitoring** tab. It provides access to the log data of all projects to which you are assigned. Before you start, select the project you want to work on. The **Log Management** option located at the top part of the **Overview** page displays the log management window where you can work on the log data of the selected project.

6.1 Working with the Log Management Window

Index patterns determine which data from the underlying Elasticsearch database can be viewed and analyzed in CMM's log management window. Index patterns are used to identify the Elasticsearch indices to run search and analytics against.

CMM ships with a preconfigured index pattern which allows you to instantly view and analyze your log data when accessing the log management window for the first time. You can configure additional index patterns to view and analyze different data from different indices. For details, refer to *Configuring Index Patterns*.

Search queries allow you to search the Elasticsearch indices for data that match your information requirements. The query results can be graphically represented in visualizations, and visualizations can be organized in dashboards.

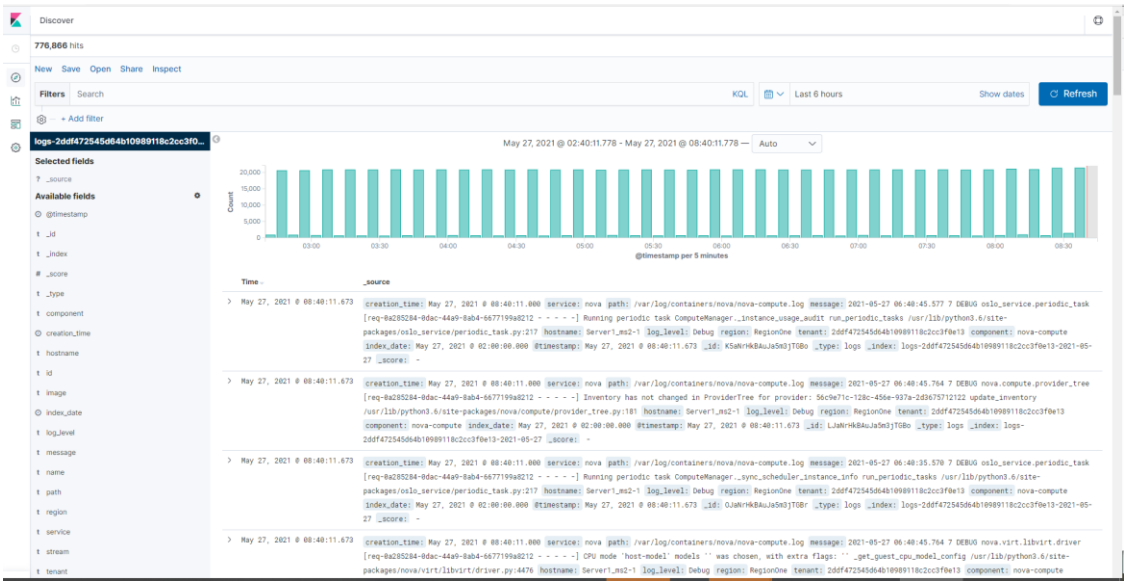
The log management window provides features for:

- Querying log data.
- Visualizing query results.
- Combining visualizations in dashboards.
- Filtering query results.
- Sharing dashboards.

The following sections provide an introduction to queries, visualizations, and dashboards. For additional details, refer to the *Kibana documentation*.

Querying Log Data

For querying log data, you use the **Discover** page in the log management window. It is instantly displayed when you access the window. It shows the most recently collected log data:



Querying Log Data

The **Discover** page allows you to access the log data in every index that matches the current index pattern. In addition to submitting queries, you can view, filter, and analyze the log data that is returned by your queries.

On the **Discover** page the following elements assist you in analyzing your log data:

- In the header section on top of the window, there is a **search box** for querying the log data. By submitting a query, you search all indices that match the current index pattern. The name of the current index pattern is displayed below the search box. You can select a

different index pattern, if required. For details on configuring and selecting index patterns, refer to *Configuring Index Patterns*.

For entering strings in the search box, you can use Kibana Query Language (KQL) or Lucene query syntax. KQL can be switched on or off with a button located besides the search box. For details, refer to the *Elasticsearch Reference documentation*.

- Use the **calender icon** located besides the search box to define a time range for filtering the log data. By default, CMM displays the log data collected during the last 15 minutes. You can deviate from this default. Multiple options are provided for defining relative or absolute time ranges. The time range you define is instantly applied to all log data.
- In the bottom part of the **Discover** page, you can view the **log data** returned by your search queries. Depending on whether you have filtered the data by index fields, the log data is either restricted to these fields or entire records are displayed.
- Above the histogram of log count, you see the **Selected fields** from the indices that match the current index pattern. You can select individual fields to modify which log data is displayed below.

  Select a field from the **Available Fields** section for this purpose and use **add**. To remove a field, select it in the **Selected Fields** section and use **remove**.

  When selecting a field from the field list, the most common values for the field are shown. You can also set field values as filter, or you can exclude log data with specific field values.
- If a time field is configured for the current index pattern, the distribution of log entries over time is displayed in a **histogram** above the display of log data.

  By default, the histogram shows the number of logs entries versus time, matched by the underlying query and time filter. You can click the bars in the histogram to narrow down the time filter.

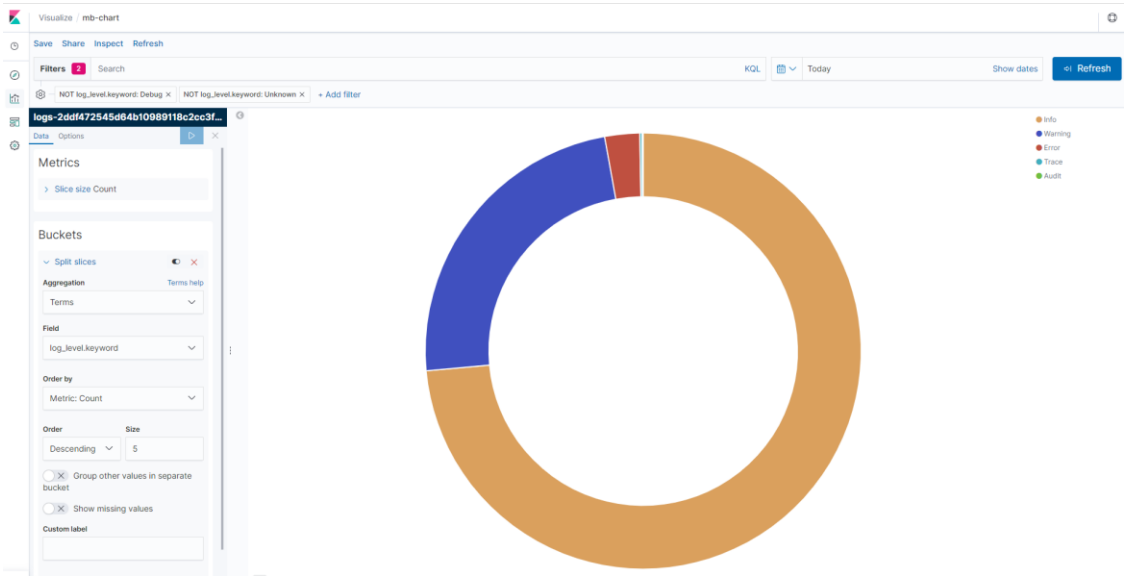
Queries can be saved and re-used. They can also be shared with other users. For this purpose, use the options provided above the search box:

- To save a query, use **Save**. Saving a query means saving both the query syntax and the current index pattern.
- To load a query, use **Open**. A saved query can be loaded and used by any OpenStack or Monitoring Service operator.
- To share a query with other users, use **Share**. The option displays a direct link to the query that you can forward. As a prerequisite for using a direct link, a user must have CMM access.

Visualizing Query Results

CMM supports you in building graphical representations of your query results. You can choose from different visualization types, for example pie charts, data tables, line charts, or vertical bar charts.

In order to create a new visualization of your results, please proceed the following way: - Click on **Visualize** symbol located on the right side - Confirm “Create New Visualization”



Visualizing Query Results

You have to select a visualization type and the query to be used. You can either create a new query or load a query you have already saved.

Based on the visualization type and the query, you can proceed with designing the graphical representation in a visualization editor. Multiple design options and a preview function are provided for creating, modifying, and viewing the graphical representation.

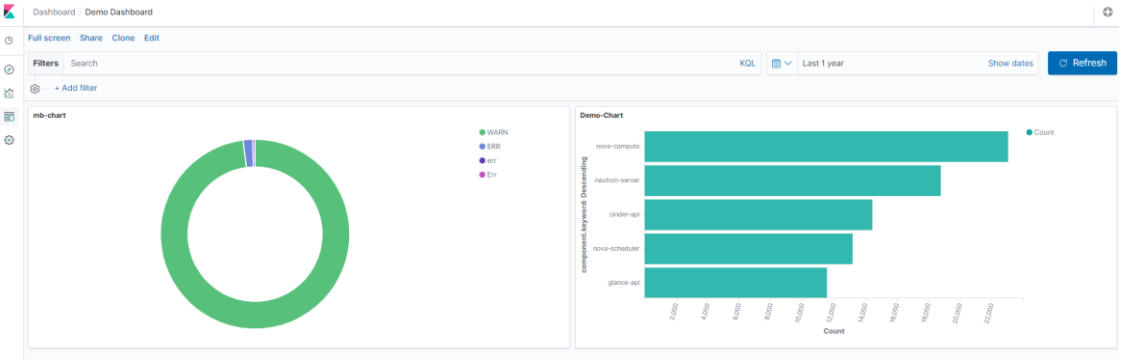
You can save and re-use visualizations. You can also share them with other users. For this purpose, use the options provided above the search box:

- To save a visualization, use **Save**.
- To load a visualization, use **Open**. A saved visualization can be loaded and used by any OpenStack or Monitoring Service operator.
- To share a visualization with other users, use **Share**. The option displays an HTML snippet that can be used to embed the visualization in a Web page. It also displays a direct link to the visualization that you can forward. As a prerequisite for using an embedded visualization or a direct link, a user must have CMM access.

Combining Visualizations in Dashboards

For correlating related information or providing an overview, you can combine visualizations in dashboards. In order to create a new dashboard, please proceed the following way:

- Click on Dashboard symbol located on the right side
- If you want to create a new dashboard: Confirm “Create new dashboard”
- If you want to work with an existing dashboard: Select the dashboard from the list of existing dashboards



Combining Visualizations in Dashboards

Possible operations have been provided above the search box:

- To save a dashboard, use **Save**.
- To add a visualization from a list of existing visualizations to the dashboard, use **Add**. Saved visualizations and saved queries can be added. You need at least one saved visualization or query to create a dashboard.
- To share a dashboard with other users, use **Share**. The option displays an HTML snippet that can be used to embed the visualization in a Web page. It also displays a direct link to the visualization that you can forward. Prerequisites for sharing dashboards:
  - User must have CMM access.
  - User must be logged in to OpenStack before accessing the dashboard.

A visualization or query result is displayed in a container on your dashboard. Various options are provided for arranging containers:

- Move a container by clicking and dragging its title bar.
- Resize a container by dragging its bottom right corner.
- Remove a container by clicking the wheel symbol using in the top right corner of the container and selecting **Delete from dashboard** from **OPTIONS** displayed.

OPTIONS sub-menu provides other funtionality, like **Edit visualization/saved search**. This allows you to design the graphical representation or edit the query.

For each dashboard, you can configure a refresh interval to automatically refresh its content with the latest data. The current interval is displayed in a box besides the calender button. If you want to change it, click the calendar button and select the appropriate time interval. In the dialog box shown, a refresh interval for automated refresh can be defined to instantly submit the underlying queries and refresh the dashboard content.

Filtering Query Results

By submitting a query on the data displayed in a dashboard, you can filter out specific sets of data that you want to aggregate while not changing the logic of the individual visualizations.

Use the search box above the Dashboard in the header section of the window for entering a query on the whole dashboard. If a visualization is already based on a saved query, both queries apply.

6.2 Configuring Index Patterns

CMM ships with a preconfigured index pattern that allows you to instantly explore your Elasticsearch indices when accessing the dashboard for the first time. Thus, configuring additional index patterns is not required.

6.3 Monitoring Log Data

CMM provides alerting features for monitoring your log data. Specific log metrics support you in checking the severity of the entries in your log files. Log metrics are handled like any other metrics in CMM. They complete the log management features and support you in analyzing and troubleshooting any issue that you encounter in your log data.

By default, CMM supports the following log metrics:

- `log.warning` to count warnings in your log data.
- `log.error` to count errors in your log data.
- `log.fatal` to count fatal errors in your log data.
- `log.critical` to count critical errors in your log data.

Note: Your installation might deviate from this default. The log levels that can be evaluated have been specified during the installation of CMM.

Using the log metrics for monitoring corresponds to using any other metrics:

- Use **Monitoring > Alarm Definitions** to create, edit, and delete alarms for log data.
- Use **Monitoring > Notifications** to create, edit, and delete notifications for alarms.
- Use **Monitoring > Overview** to check whether there are any irregularities in your log data. As soon as you have defined an alarm for your log data and metrics data has been received, there is status information displayed on the **Overview** page.

For details on using the log metrics, refer to *Monitoring*.

7 Uninstallation

The uninstallation of CMM and its OpenStack extensions comprises the following steps:

1. Uninstalling a Metrics Agent from an OpenStack node.
2. Uninstalling a Log Agent from an OpenStack node.
3. Uninstalling the Horizon Plugin from the node where the OpenStack Horizon service is deployed.
4. Uninstalling the Monitoring Service.
5. Removing the projects, users, and roles prepared for the OpenStack integration.
6. Removing the services and endpoints prepared for the Horizon Plugin in OpenStack Keystone.

7.1 Uninstalling a Metrics Agent

Before uninstalling a Metrics Agent from an OpenStack node, it is recommended that you make a backup of the configuration files created for operation. The configuration files for an agent are located in the `/etc/monasca/agent/` directory.

CMM does not offer integrated backup and recovery mechanisms. Use the standard file system mechanisms instead.

To uninstall a Metrics Agent, proceed as follows:

1. Log in to the OpenStack node on which the agent is installed.
2. Stop the `monasca-agent` service and delete all related files:

```
systemctl stop monasca-agent.target
systemctl disable monasca-agent.target
rm -f /etc/systemd/system/monasca-agent.target
rm -f /etc/systemd/system/monasca-collector.service
rm -f /etc/systemd/system/monasca-forwarder.service
rm -f /etc/systemd/system/monasca-statsd.service
systemctl daemon-reload
systemctl reset-failed monasca-agent.target
systemctl reset-failed monasca-collector.service
systemctl reset-failed monasca-forwarder.service
systemctl reset-failed monasca-statsd.service
```

3. Remove all directories and files created by the agent installer:

```
rm -rf /opt/monasca-agent/
rm -rf /etc/monasca/
rm -f /etc/sudoers.d/mon-agent
```

4. Remove the agent's log files and the log directory:

```
rm -rf /var/log/monasca-agent/
```

5. Remove the `mon-agent` user. Use -r to also remove the `mon-agent` user's home directory.

```
userdel -r mon-agent
```

7.2 Uninstalling a Log Agent

Before uninstalling a Log Agent from an OpenStack node, it is recommended that you make a backup of the agent's configuration settings. They are stored in the `/<installation_dir>/conf/agent.conf` file.

CMM does not offer integrated backup and recovery mechanisms. Use the standard file system mechanisms instead.

To uninstall a Log Agent, proceed as follows:

1. Log in to the OpenStack node on which the agent is installed.
2. Stop the monasca-log-agent service and delete all related files:

   ```
   systemctl stop monasca-log-agent
   systemctl disable monasca-log-agent
   rm -f /etc/systemd/system/monasca-log-agent.service
   systemctl daemon-reload
   systemctl reset-failed monasca-log-agent
   ```

3. Remove all directories and files created by the agent installer:

   ```
   rm -rf <target_dir>
   ```

   Replace `<target_dir>` by the directory in which the agent is installed, (e.g. `/opt/monasca-log-agent/`).

4. Remove the agent's log files and the log directory:

   ```
   rm -rf /var/log/monasca-log-agent/
   ```

5. Remove the Logstash script used for defining the position of monitored log files:

   ```
   rm -f /etc/profile.d/logstash_sincedb_dir.sh
   ```

7.3 Uninstalling the Horizon Plugin (Monasca-UI)

To uninstall the Horizon Plugin (Monasca-UI), proceed as follows:

1. Log in as root to the OpenStack node on which the OpenStack Horizon service is installed.
2. Enter into Horizon container:

   ```
   # podman exec -it horizon /bin/sh
   ```

3. Remove monasca-ui directories:

   ```
   $ rm -rf /opt/monasca-ui/
   $ rm -rf /usr/share/openstack-dashboard/static/monitoring
   ```

4. Uninstall monasca-ui and python-monascaclient:

   ```
   $ python3.6 -m pip uninstall monasca-ui python-monascaclient
   ```

Note: If the japanese translations were installed, remove this directory:
`rm -rf /usr/local/lib/python3.6/site-packages/monitoring`

5. Remove the symbolic links:

```
$ rm /etc/openstack-dashboard/monitoring_policy.json
$ rm /usr/share/openstack-
dashboard/openstack_dashboard/local/enabled/_50_admin_add_monitoring_panel.py
$ rm /usr/share/openstack-
dashboard/openstack_dashboard/local/local_settings.d/_50_monasca_ui_settings.p
y
```

6. Update Django settings:

```
$ python3 /usr/share/openstack-dashboard/manage.py collectstatic --noinput
$ python3 /usr/share/openstack-dashboard/manage.py compress --force
```

Note: Expected messages:

```
WARNING:root:"dashboards" and "default_dashboard" in (local_)settings is
DEPRECATED
ERROR:scss.ast:Function not found: twbs-font-path:1
ERROR:scss.compiler:Mixin not found: dropdown-arrow:0
ERROR:scss.compiler:Maximum number of supported selectors in Internet
Explorer (4095) exceeded!
```

7. Exit from Horizon container:

```
$ exit
```

8. Remove Proxy Modules from file httpd.conf. Location of file:

```
/var/lib/config-data/puppet-generated/horizon/etc/httpd/conf/httpd.conf
```

Remove these lines:

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
```

9. Remove the Proxy configuration inside VirtualHost section in 10-horizon_vhost.conf. Location of file:

```
/var/lib/config-data/puppet-generated/horizon/etc/httpd/conf.d/10-
horizon_vhost.conf
```

Remove these lines:

```
ProxyPass        "/grafana" "http://<CMM-SERVER-IP>:3000"
ProxyPassReverse "/grafana" "http://<CMM-SERVER-IP>:3000"
```

10. Restart Horizon container:

```
# systemctl restart tripleo_horizon
```

7.4 Uninstalling the Monitoring Service

To uninstall the Monitoring Service, proceed as follows:

1. Log in to the CMM node as a user with root privileges.
2. Go to the installation directory.
3. To stop all agents and services and remove the containers, the instances of the docker images, and any volumes you mounted for CMM, `run docker-compose down` as follows:

   ```
   docker-compose -f docker-compose-metric.yml \
   -f docker-compose-log.yml down --rmi all --volumes
   ```

   Note: When running the command, you can ignore warnings about images to be removed not being found. The messages are written because some images are used twice, for setting up Kafka for the metrics pipeline as well as for the log pipeline. Irrespective of these messages, all docker images are successfully removed.

4. Delete the `docker-compose` file in the `/usr/local/bin/` directory:

   ```
   rm -f /usr/local/bin/docker-compose
   ```

5. Delete the installation directory of the Monitoring Service to which you extracted the `CMM_server_2.0.14-x.tar.gz` archive file from the CMM installation package:

   ```
   rm -rf <path_to_installation_directory>
   ```

6. Remove the volume you mounted for the data directories of Elasticsearch, InfluxDB, MySQL, Kafka, and Grafana. Example with the default volume:

   ```
   rm -rf /opt/monasca-containers/
   ```

7. Remove the volume you mounted for backing up the databases. Example with the default volume:

   ```
   rm -rf /mount/backup/
   ```

7.5 Removing OpenStack Projects, Users, and Roles

To integrate CMM with the OpenStack Keystone service, you created a project, a user, and two roles in OpenStack.

To remove them from your OpenStack environment, proceed as follows:

1. Provide the administrator credentials that are required to perform any action in OpenStack Keystone. For details, refer to *Setting the Administrator Credentials*.
2. Remove the OpenStack project that was used for the monitoring data retrieved for CMM and your OpenStack services and servers. Example with `monasca` as project name:

   ```
   openstack project delete monasca
   ```

3. Remove the OpenStack user that was used for authenticating an agent against OpenStack Keystone. Example with `monasca-agent` as user name:

```
openstack user delete monasca-agent
```

4. Remove the OpenStack roles that were prepared. Example with `monasca-user` and `monasca-agent` as role names:

```
openstack role delete monasca-user
openstack role delete monasca-agent
```

7.6 Removing Services and Endpoints

For the Horizon Plugin, you defined a set of services and endpoints in OpenStack Keystone. To remove the services and endpoints from your OpenStack environment, proceed as follows:

1. Provide the administrator credentials that are required to perform any action in OpenStack Keystone. For details, refer to Setting the *Administrator Credentials*.

2. For removing the endpoints, you need to know the endpoint IDs:

```
openstack endpoint list | grep monasca
openstack endpoint list | grep logs
```

3. Remove the endpoints as follows:

```
openstack endpoint delete <endpoint-id1> <endpoint-id2>
```

Replace `<endpoint-id1>` and `<endpoint-id2>` by the endpoints of the two services.

4. Remove the services as follows:

```
openstack service delete monasca logs
```

8 Migration

This chapter describes the process of migration from CMM v2.0.13 to CMM v2.0.14

8.1 MySQL Database Migration

Note 1: The migration process of MySQL Database from CMM v2.0.13 to CMM v2.0.14 is based on the MySQL Database Backup and Recovery steps on page 58.

Note 2: It is assumed that the tenant and its tenant-id used for monitoring have not changed.

Back up CMM v2.0.13 MySQL database:

1. Log in to the CMM node with v2.0.13 as a user with root privileges.
2. Stop all CMM agents and services:

```
docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml stop
```

3. Start the `mysql` service:

```
docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml up -d
mysql
```

4. Backup the database. To create the backup from your local machine, execute the following command:

   ```
   docker exec <container_id> mysqldump -u root --password=secretmysql mon > <backup_dir>/<name>
   ```

   Replace: - <container_id> by the ID of the container in which the database is running. - <db_name> by the name of the database. - <backup_dir> by the directory you have mounted for storing the backup file. - <name> by the name of the backup file to be created.

   For the --password parameter, check the MYSQL_ROOT_PASSWORD specified in the docker-compose-metric.yml file.

   The following example creates a backup of the mon database, running in container 07e9007e0be8, in /backup/mon.sql:

   ```
   docker exec 07e9007e0be8 mysqldump -u root --password=secretmysql mon > /backup/mon.sql
   ```

5. Check whether the backup was created successfully. In order to see its content you can use vim:

   ```
   vim /backup/mon.sql
   ```

Restore MySQL database in CMM v2.0.14:

Note: In this step it is assumed that the CMM node with v 2.0.14 was successfully installed and all containers are running as described in chapter 2.3 Installing the Monitoring Service

1. Log in to the new CMM node with v2.0.14 as a user with root privileges and transfer the file mon.sql.
2. Stop all CMM agents and services:

   ```
   docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml stop
   ```

3. Start the mysql service:

   ```
   docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml up -d mysql
   ```

4. Restore the database:

   ```
   cat <backup_dir>/<name> | docker exec -i <container_id> mysql -u root --password=secretmysql mon
   ```

   Replace: - <backup_dir> by the directory where the backup file is stored. - <name> by the name of the file. - <container_id> by the ID of the container in which the database is running.

   For the --password parameter, check the MYSQL_ROOT_PASSWORD specified in the docker-compose-metric.yml file.

The following example restores the mon database, running in container `07e9007e0be8`, from `/ backup/mon.sql`:

```
cat /backup/mon.sql | docker exec -i 07e9007e0be8 mysql -u root --
password=secretmysql mon
```

5. Stop the `mysql` service:

   ```
   docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml stop
   mysql
   ```

6. Start all CMM agents and services:

   ```
   docker-compose -f docker-compose-metric.yml -f docker-compose-log.yml up -d
   ```

For additional information on backing up and restoring a MySQL database with `mysqldump`, refer to the *MySQL documentation*.

Appendix A: Supported Metrics

The sections below describe the metrics supported by CMM:

- Standard metrics for general monitoring of servers and networks.
- Additional metrics for monitoring specific servers and services.

Note: Adding dimensions for metrics is not supported by CMM. The installer auto-detects applications and processes that are running on your machine and saves the corresponding settings to the agent's configuration file. Additional dimensions cannot be specified.

A.1 Standard Metrics

CMM supports the following standard metrics for monitoring servers and networks. These metrics usually do not require specific settings. The metrics are grouped by metrics types. Each metrics type references a set of related metrics.

cpu.yaml

Metrics on CPU usage, e.g. the percentage of time the CPU is idle when no I/O requests are in progress, or the percentage of time the CPU is used at system level or user level.

disk.yaml

Metrics on disk space, e.g. the percentage of disk space that is used on a device, or the total amount of disk space aggregated across all the disks on a particular node.

load.yaml

Metrics on the average system load over different periods (e.g. 1 minute, 5 minutes, or 15 minutes).

memory.yaml

Metrics on memory usage, e.g. the number of megabytes of total memory or free memory, or the percentage of free swap memory.

network.yaml

Metrics on the network, e.g. the number of network bytes received or sent per second, or the number of network errors on incoming or outgoing network traffic per second.

A.2 Additional Metrics

CMM supports the additional metrics described below for monitoring specific servers and services. The metrics are grouped by metrics types. Each metrics type references a set of related metrics. Depending on the services running on the host where you install a Metrics Agent, some or all of these metrics are automatically added to the agent configuration. You should check the

individual yaml files and change or correct the settings as required, or remove individual `yaml` files from the agent configuration if you do not want to monitor the metrics they include.

Note: In addition to the metrics below, many more metrics are provided by the Monasca project. These are not automatically installed by CMM. For details on the complete set of metrics provided by the Monasca project, refer to the *Monasca documentation*. If you want to extend your monitoring environment to perform additional checks, contact your FUJITSU support organization.

apache.yaml

Apache Web Server checks collect metrics from an Apache Web Server. If you want the installer to automatically configure the checks, update the `apache.cnf` file in the `root` directory before installing the agent.

Example configuration for `apache.cnf` with the URL of the server, the user name, and the password:

```
[client]
url=http://localhost/server-status?auto
user=root
password=password
#dimensions:
    #dim1: value1
```

Specify the configuration information in the `apache.yaml` file after the installation.

Example configuration:

```
init_config: null
instances:
- apache_status_url: [http://localhost/server-
status?auto](http://localhost/server-status?auto)
  apache_user: root
  apache_password: password
```

ceph.yaml

Ceph checks provide information on one or more Ceph clusters. The configuration file must specify the name of the cluster to be monitored. You can configure the agent so that only a specific set of metrics data is collected.

As a prerequisite, the `ceph-common` package must be installed. In addition, the user running the agent must have the permission to execute Ceph commands. For this purpose, assign the `monasca-agent` user to the ceph group, and give read permission to the group in the `ceph.client.admin.keyring` file:

```
usermod -a -G ceph monasca-agent chmod 0640 /etc/ceph/ceph.client.admin.keyring
```

Alternatively, you can configure `sudo` access for the `monasca-agent` user.

Example configuration with monasca-user assigned to the ceph group:

```
init_config: null
instances:
- cluster_name: ceph
  use_sudo: False
  collect_usage_metrics: True
  collect_stats_metrics: True
  collect_mon_metrics: True
  collect_osd_metrics: False
  collect_pool_metrics: True
```

crash.yaml

Crash checks provide information on system crash dumps. The default directory where the crash dumps are provided is `/var/crash`.

The agent installer automatically checks whether a crash kernel is loaded on the machine where the agent is installed. If so, the detected settings are saved to the `crash.yaml` configuration file, and the configuration is automatically provided in the `/etc/monasca/agent/conf.d/` directory.

Example configuration:

```
init_config: null
instances:
- built_by: Crash
  name: crash_stats
```

elastic.yaml

Elastic checks gather metrics for Elasticsearch databases, such as the Log Database of CMM. The configuration file must specify the URL for HTTP requests. If basic authentication is used, for example `elasticsearch-http-basic`, the configuration file must also specify the user name and password for every instance that requires authentication.

The agent installer automatically creates the `elastic.yaml` configuration file in the `/etc/monasca/agent/conf.d/` directory. If there is an Elasticsearch database instance installed on the machine where the agent is installed, you have to specify the configuration information in the `elastic.yaml` file after the installation.

Example configuration:

```
init_config: null
instances:

- dimensions:
  component: elasticsearch
  service: monitoring
  url: [http://localhost:9200](http://localhost:9200)
  username: username
  password: password
```

host_alive.yaml

Host alive checks perform checks on a remote host to determine whether it is alive. The checks use either ping (ICMP) or SSH.

SSH checks provide extensive tests on the availability of remote host machines. They check the banner that is returned. A remote host machine may still respond to a ping request but may not return an SSH banner. Therefore it is recommended that you use SSH checks instead of ping checks.

The agent installer automatically creates the `host_alive.yaml` configuration file in the `/etc/monasca/agent/conf.d/` directory. If there are applications and processes running on the machine where the agent is installed, you have to specify the configuration information in the `host_alive.yaml` file after the installation.

Example configuration:

```
init_config:
  ssh_port:     22
  ssh_timeout:  0.5
  ping_timeout: 1
instances:
# - name: ssh to host1
#   host_name:     host1.domain.net
#   alive_test:    ssh
#   dimensions:
#     dim1: value1

# - name: ping host1
#   host_name:     10.140.16.153
#   alive_test:    ping

- name: ssh to 192.168.0.221
  host_name:     192.168.0.221
  alive_test:    ssh
```

http_check.yaml

HTTP endpoint checks perform up/down checks on HTTP endpoints. Based on a list of URLs, the agent sends an HTTP request and reports success or failure to the Monitoring Service.

The agent installer auto-detects HTTP endpoints on the machine where the agent is installed. It saves the detected settings to the `http_check.yaml` configuration file, and the configuration is automatically provided in the `/etc/monasca/agent/conf.d/` directory.

Example configuration:

```
init_config: null
instances:
- built_by: Cinder
  dimensions:
```

```
    service: block-storage
  collect_response_time: true
  match_pattern: .*version=[1-3].*
  name: block-storage-api
  timeout: 10
  url: http://localhost:8776/v2
  use_keystone: true
- built_by: Glance
  dimensions:
    service: image-service
  match_pattern: .*v2.0.*
  name: image-service-api
  timeout: 10
  url: http://localhost:9292
  use_keystone: true
```

http_metrics.yaml

HTTP metrics checks retrieve metrics from any URL that returns a JSON response. Based on a list of URLs, the agent can dispatch an HTTP request and parse the desired metrics from the JSON response.

The agent installer auto-detects applications and processes on the machine where the agent is installed. It saves the detected settings to the `http_metrics.yaml` configuration file, and the configuration is automatically provided in the `/etc/monasca/agent/conf.d/` directory.

Example configuration:

```
init_config: null
instances:
- built_by: MonAPI
  dimensions:
  component: monasca-api
  service: monitoring
  name: monitoring-monasca-api metrics
  timeout: 5
  url: http://localhost:8081/metrics
  whitelist:
  - name: jvm.memory.total.max
    path: gauges/jvm.memory.total.max/value
    type: gauge
  - name: jvm.memory.total.used
    path: gauges/jvm.memory.total.used/value
    type: gauge
    - name: metrics.published
    path: meters/monasca.api.app.MetricService.metrics.published/count
    type: rate
```

kafka_consumer.yaml

Kafka consumer checks gather metrics related to services consuming Kafka topics, such as the Persister or Notification Engine of CMM.

For Kafka consumer checks, the Kafka consumer module (`kafka-python`) must be installed in the virtualenv environment of the Metrics Agent. To install it in the default directory, execute the following command:

```
# source /opt/monasca-agent/bin/activate
# pip install kafka-python
# deactivate
```

The agent installer automatically detects services that are consuming Kafka topics on the machine where the agent is installed. If such services are detected, the corresponding settings are saved to the `kafka_consumer.yaml` configuration file, and the configuration is automatically provided in the `/etc/monasca/agent/conf.d/` directory.

Example configuration:

```
init_config: null
instances:
- built_by: Kafka
  consumer_groups:
    1_alarm-state-transitions:
      alarm-state-transitions: []
    1_metrics:
      metrics: []
    log-metric:
      transformed-log: []
    logstash-persister:
      transformed-log: []
    thresh-event:
      events: []
    thresh-metric:
      metrics: []
    transformer-logstash-consumer:
      log: []
  kafka_connect_str: 10.140.18.49:9092
  name: 10.140.18.49:9092
  per_partition: false
```

kibana.yaml

Kibana checks gather metrics from a Kibana server. The checks access the status endpoint of Kibana (`curl -XGET http://localhost:5601/api/status`). The configuration information in the `kibana.yaml` file corresponds to the Kibana configuration.

The agent installer automatically checks whether a Kibana server instance is installed on the machine where the agent is installed. If a Kibana server instance is detected, the corresponding

settings are saved to the `kibana.yaml` configuration file, and the configuration is automatically provided in the `/etc/monasca/agent/conf.d/` directory.

Example configuration:

```
init_config:
  url: http://10.140.18.49:5601/api/status
instances:
- built_by: Kibana
  metrics:
  - load
  - heap_size
  - heap_used
  - req_sec
  - resp_time_max
  - resp_time_avg
```

libvirt.yaml

Libvirt checks provide metrics for virtual machines that run on a hypervisor. The checks provide a set of metrics for the owner of the virtual machine as well as for the owner of the hypervisor. For details on configuring the metrics, refer to the *Monasca documentation*.

Check the `keystone_authtoken` section in the `nova.conf` file for the `password`, `username`, and `auth_url` configuration.

The user specified in the configuration file must be assigned the `admin` role.

Replace `<keystone_ip>` in the `libvirt.yaml.example` file by the IP address of the node on which the OpenStack Keystone service is deployed. Example URL: `http://192.168.1.5:35357`.

With Red Hat Enterprise Linux OpenStack Platform as underlying platform technology, the `project_name` must be set to `services`.

Example configuration:

```
init_config:
    password: pass
    project_name: services
    username: nova
    auth_url: 'http://<keystone_ip>:35357'
    endpoint_type: 'publicURL'
    cache_dir: /dev/shm
    nova_refresh: 14400
    vm_probation: 300
    ping_check: sudo -n /sbin/ip exec NAMESPACE /usr/bin/fping -n -c1 -t250 -q
    alive_only: false
    metadata:
    - scale_group
    customer_metadata:
    - scale_group
```

```
    network_use_bits: false
    vm_cpu_check_enable: True
    vm_disks_check_enable: True
    vm_extended_disks_check_enable: True
    vm_network_check_enable: True
    vm_ping_check_enable: True
    vm_extended_disks_check_enable: False
    host_aggregate_re: None
    disk_collection_period: 0
    vnic_collection_period: 0
instances:
- {}
```

mysql.yaml

MySQL checks gather metrics from a MySQL database server. The metrics are related to the server status variables of MySQL.

As a prerequisite, you need to find out the MySQL password credentials stored in `/root/.my.cnf` file inside the galera-bundle-podman container.

Run following commands:

```
podman exec -it galera-bundle-podman-0 /bin/sh
cat /root/.my.cnf
```

Example configuration for `/root/.my.cnf`:

```
[client]
user=root
password="FRjSTiKbzaq"

[mysql]
user=root
password="FRjSTiKbzaq"
```

Edit the file `mysql.yaml` in `/etc/monasca/agent/conf.d/` and insert the password as obtained before.

Note: Make sure that the password is set correctly. The agent installer fails if you specify the password enclosed in quotation marks.

In addition, the MySQL module (`PyMySQL`) must be installed in the virtualenv environment of the Metrics Agent. To install it in the default directory, execute the following command:

```
# source /opt/monasca-agent/bin/activate
# pip install PyMySQL
# deactivate
```

Example configuration for `mysql.yaml`:

```
init_config: null
instances:
- built_by: MySQL
  name: localhost
  pass: FRjSTiKbzaq
  port: 3306
  server: localhost
  sock: /var/lib/mysql/mysql.sock
  ssl_ca: null
  ssl_cert: null
  ssl_key: null
  user: root
```

ntp.yaml

Network Time Protocol checks monitor the time offset between the NTP server and the host machine. The configuration file must specify the parameters that you want to monitor.

Example configuration:

```
init_config: null
instances:
- host: pool.ntp.org
  port: ntp
  version: 3
  timeout: 5
# dimensions:
#     dim1: value1
```

Note: To collect the complete set of metrics from the NTP server and the host machine, it might additionally be required to update your NTP server configuration. Check whether changes resulting from an update are also required in the `ntp.yaml` file.

ovs.yaml

Open vSwitch Neutron Router Monitoring checks monitor neutron virtual routers implemented with OpenVSwitch. The checks include rate metrics as well as health-related metrics.

The agent installer automatically checks whether a neutron virtual router is installed on an OpenStack host. If so, the detected settings are saved to the `ovs.yaml` configuration file, and the configuration is automatically provided in the `/etc/monasca/agent/conf.d/` directory.

Example configuration:

```
init_config:
  admin_password: password
  admin_tenant_name: services
  admin_user: neutron
  cache_dir: /dev/shm
  identity_uri: http://10.140.18.53:35357/v2.0
  included_interface_re: qg.*|vhu.*|sg.*
```

```
  network_use_bits: false
  neutron_refresh: 14400
  ovs_cmd: sudo /usr/bin/ovs-vsctl
  region_name: RegionOne
  use_absolute_metrics: true
  use_health_metrics: true
  use_rate_metrics: true
instances: {}
```

postfix.yaml

Postfix checks monitor a Postfix mail server. The agent installer automatically checks whether a Postfix mail server instance is installed on the machine where the agent is installed. If so, the detected settings are saved to the `postfix.yaml` configuration file, and the configuration is automatically provided in the `/etc/monasca/agent/conf.d/` directory.

Example configuration:

```
init_config: null
instances:
- built_by: Postfix
  directory: /var/spool/postfix
  name: /var/spool/postfix
  queues:
  - incoming
  - active
  - deferred
```

postgres.yaml

Postgres checks gather metrics from a PostgreSQL database.

For PostgreSQL checks, the `postgresql-devel` RPM package and the `psycopg2` Python package are required. The Python package must be installed in the virtualenv environment of the Metrics Agent. To install the packages in the default directory, execute the following command:

```
# yum install postgresql-devel
# source /opt/monasca-agent/bin/activate
# pip install psycopg2
# deactivate
```

Example configuration:

```
init_config: null
instances:
  - host: localhost
    port: 5432
    username: my_username
    password: my_password
    dbname: db_name
    relations:
```

```
  - my_table
  - my_other_table
```

process.yaml

Process checks verify that a defined set of processes is up and running. The processes can be identified by specifying the process name or a pattern match.

The agent installer auto-detects processes that are running on the machine where the agent is installed. It saves the detected settings to the `process.yaml` configuration file, and the configuration is automatically provided in the `/etc/monasca/agent/conf.d/` directory.

Example configuration:

```
init_config: null
instances:
- built_by: MySQL
  detailed: true
  dimensions:
    service: mysql
  exact_match: true
  name: mysqld
  search_string:
  - mysqld
- built_by: Nova
  detailed: true
  dimensions:
  component: nova-compute
  service: compute
  exact_match: false
  name: nova-compute
  search_string:
  - nova-compute
```

rabbitmq.yaml

RabbitMQ checks gather metrics on nodes, exchanges, and queues from a RabbitMQ server.

As a prerequisite, you need to find out the RabbitMQ password credentials stored in `/etc/rabbitmq/rabbitmq.config` file inside the rabbitmq-bundle-podman-0 container.

Run following commands:

```
podman exec -it rabbitmq-bundle-podman-0 /bin/sh
cat /etc/rabbitmq/rabbitmq.config
```

For RabbitMQ checks, the RabbitMQ Management plugin must be installed. It is included in the RabbitMQ distribution. To enable the plugin, execute the following command:

```
rabbitmq-plugins enable rabbitmq_management
```

Specify the configuration information in the `rabbitmq.yaml` file after the installation. Replace `<rabbitmq-user>` and `<rabbitmq-password>` with the current RabbitMQ user and password. It must specify the names of the exchanges and queues to be monitored.

Example configuration:

```
init_config: null
instances:
-  rabbitmq_api_url: http://localhost:15672/api/
   rabbitmq_user: <rabbitmq-user>
   rabbitmq_pass: <rabbitmq-password>
   nodes:
    - rabbit@localhost
    - rabbit2@domain
   queues:
    - queue1
    - queue2
   whitelist:
     queue:
      - message_stats/deliver_details/rate
      - message_stats/publish_details/rate
      - message_stats/redeliver_details/rate
     exchange:
      - message_stats/publish_out
      - message_stats/publish_out_details/rate
      - message_stats/publish_in
      - message_stats/publish_in_details/rate
     node:
      - fd_used
      - mem_used
      - run_queue
      - sockets_used
```

Note: To collect the complete set of metrics from the RabbitMQ server, it might additionally be required to update your RabbitMQ server configuration. Check whether changes resulting from an update are also required in the `rabbitmq.yaml` file.

Glossary

Application Operator

A person responsible for providing services to end users or hosting services for development activities. An application operator has limited access to cloud resources in OpenStack.

Dimension

A key/value pair that allows for a flexible and concise description of the data to be monitored, for example region, availability zone, service tier, or resource ID. Each dimension describes a specific characteristic of the metrics to be monitored.

In CMM, metrics are uniquely identified by a name and a set of dimensions. Dimensions can serve as a filter for the monitoring data.

Elasticsearch

An open-source application that provides a highly scalable full-text search and analytics engine. CMM uses Elasticsearch as the underlying technology for storing, searching, and analyzing large volumes of log data.

Grafana

An open-source application for visualizing large-scale measurement data. CMM integrates with Grafana for visualizing the monitoring data.

InfluxDB

An open-source time-series database that supports high write loads and large data set storage. CMM uses InfluxDB as the underlying technology for storing metrics and the alarm history.

Infrastructure as a Service (IaaS)

The delivery of computer infrastructure (typically a platform virtualization environment) as a service.

Kibana

An open-source analytics and visualization platform designed to work with Elasticsearch. CMM integrates with Kibana for visualizing the log data.

Logstash

An open-source application that provides a data collection engine with pipelining capabilities. CMM integrates with Logstash for collecting, processing, and outputting logs.

MySQL

An open-source relational database that provides an SQL-compliant interface for accessing data. CMM uses MySQL as the underlying technology for storing configuration information, alarm definitions, and notification methods.

Metrics

Self-describing data structures that allow for a flexible and concise description of the data to be monitored. Metrics values represent the actual monitoring data that is collected and presented in CMM.

Monasca

An open-source Monitoring as a Service solution that integrates with OpenStack. It forms the core of CMM.

Monitoring Service Operator

A person responsible for maintaining and administrating CMM.

OpenStack Operator

A person responsible for maintaining and administrating OpenStack, the underlying platform technology of CMM.

Platform as a Service (PaaS)

The delivery of a computing platform and solution stack as a service.

Software as a Service (SaaS)

A model of software deployment where a provider licenses an application to customers for use as a service on demand.