Red Hat OpenStack Platform 11
Monitoring Tools Configuration Guide

A guide to OpenStack logging and monitoring tools

OpenStack Team
A guide to OpenStack logging and monitoring tools

OpenStack Team
rhos-docs@redhat.com
Abstract

This guide provides information on configuring logging and monitoring for a Red Hat OpenStack Platform environment.
Table of Contents

CHAPTER 1. INTRODUCTION ................................................................. 3

CHAPTER 2. ARCHITECTURE .............................................................. 4
  2.1. CENTRALIZED LOGGING ......................................................... 4
  2.2. AVAILABILITY MONITORING ............................................... 6

CHAPTER 3. INSTALL THE CLIENT-SIDE TOOLS ..................................... 10
  3.1. SET CENTRALIZED LOGGING CLIENT PARAMETERS .................... 10
  3.2. SET AVAILABILITY MONITORING CLIENT PARAMETERS ............... 11
  3.3. INSTALL OPERATIONAL TOOLS ON OVERCLOUD NODES ............ 11

CHAPTER 4. INSTALL THE SERVER-SIDE COMPONENTS .............................. 12

CHAPTER 5. MONITOR THE OPENSTACK PLATFORM .................................. 13

CHAPTER 6. VALIDATE THE SENSU CLIENT INSTALLATION ......................... 14

CHAPTER 7. REVIEW THE STATE OF A NODE ......................................... 15

CHAPTER 8. REVIEW THE STATE OF AN OPENSTACK SERVICE .................... 16
CHAPTER 1. INTRODUCTION

Monitoring tools are an optional suite of tools designed to help operators maintain an OpenStack environment. The tools perform the following functions:

- Centralized logging: Allows you gather logs from all components in the OpenStack environment in one central location. You can identify problems across all nodes and services, and optionally, export the log data to Red Hat for assistance in diagnosing problems.

- Availability monitoring: Allows you to monitor all components in the OpenStack environment and determine if any components are currently experiencing outages or are otherwise not functional. You can also configure the system to alert you when problems are identified.
CHAPTER 2. ARCHITECTURE

Monitoring tools use a client-server model with the client deployed onto the Red Hat OpenStack Platform overcloud nodes. The Fluentd service provides client-side centralized logging (CL) and the Sensu client service provides client-side availability monitoring (AM).

2.1. CENTRALIZED LOGGING

Centralized logging allows you to have one central place to view logs across your entire OpenStack environment. These logs come from the operating system, such as syslog and audit log files, infrastructure components such as RabbitMQ and MariaDB, and OpenStack services such as Identity, Compute, and others.

The centralized logging toolchain consists of a number of components, including:

- Log Collection Agent (Fluentd)
- Log Relay/Transformer (Fluentd)
- Data Store (Elasticsearch)
- API/Presentation Layer (Kibana)

Note

The director does not deploy the server-side components for centralized logging. Red Hat does not support the server-side components, including the Elasticsearch database, Kibana, and Fluentd with plugins running as a log aggregator.

The centralized logging components and their interactions are laid out in the following diagrams:

Note

Items shown in blue denote Red Hat-supported components.

Figure 2.1. Centralized logging architecture at a high level
Figure 2.2. Single-node deployment for Red Hat OpenStack Platform
2.2. AVAILABILITY MONITORING

Availability monitoring allows you to have one central place to monitor the high-level functionality of all components across your entire OpenStack environment.

The availability monitoring toolchain consists of a number of components, including:

- Monitoring Agent (Sensu client)
- Monitoring Relay/Proxy (RabbitMQ)
- Monitoring Controller/Server (Sensu server)
- API/Presentation Layer (Uchiwa)

Note

The director does not deploy the server-side components for availability monitoring. Red Hat does not support the server-side components, including Uchiwa, Sensu Server, the Sensu API plus RabbitMQ, and a Redis instance running on a monitoring node.

The availability monitoring components and their interactions are laid out in the following diagrams:
Note

Items shown in blue denote Red Hat-supported components.

Figure 2.4. Availability monitoring architecture at a high level

Figure 2.5. Single-node deployment for Red Hat OpenStack Platform
Figure 2.6. HA deployment for Red Hat OpenStack Platform
CHAPTER 3. INSTALL THE CLIENT-SIDE TOOLS

Before you deploy the overcloud, you need to determine the configuration settings to apply to each client. Copy the example environment files from the director’s Heat template collection and modify them to suit your environment.

3.1. SET CENTRALIZED LOGGING CLIENT PARAMETERS

For Fluentd configuration settings, copy `/usr/share/openstack-tripleo-heat-templates/environments/logging-environment.yaml` and modify the file to suit your environment. For example:

Simple configuration

```yaml
resource_registry:
  OS::TripleO::Services::FluentdClient:
  ../puppet/services/logging/fluentd-client.yaml

parameter_defaults:
  LoggingServers:
  - host: log0.example.com
    port: 24224
  - host: log1.example.com
    port: 24224
```

Example SSL configuration

```yaml
## (note the use of port 24284 for ssl connections)

resource_registry:
  OS::TripleO::Services::FluentdClient:
  ../puppet/services/logging/fluentd-client.yaml

parameter_defaults:
  LoggingServers:
    - host: 192.0.2.11
      port: 24284
  LoggingUsesSSL: true
  LoggingSharedKey: secret
  LoggingSSLCertificate: |
      -----BEGIN CERTIFICATE-----
      ...certificate data here...
      -----END CERTIFICATE-----
```

- **LoggingServers** - The destination system that will receive Fluentd log messages.
- **LoggingUsesSSL** - Setting that determines whether `secure_forward` is used when forwarding log messages.
- **LoggingSharedKey** - The shared secret used by `secure_forward`.
- **LoggingSSLCertificate** - The PEM-encoded contents of the SSL CA certificate.
3.2. SET AVAILABILITY MONITORING CLIENT PARAMETERS

For the Sensu client configuration settings, copy /usr/share/openstack-tripleo-heat-templates/environments/monitoring-environment.yaml and modify the file to suit your environment. For example:

```
resource_registry:
  OS::TripleO::Services::SensuClient:
    ../puppet/services/monitoring/sensu-client.yaml

parameter_defaults:
  MonitoringRabbitHost: 10.10.10.10
  MonitoringRabbitPort: 5672
  MonitoringRabbitUserName: sensu
  MonitoringRabbitPassword: sensu
  MonitoringRabbitUseSSL: false
  MonitoringRabbitVhost: "/sensu"
  SensuClientCustomConfig:
    api:
      warning: 10
      critical: 20
```

- **MonitoringRabbit** - These parameters connect the Sensu client services to the RabbitMQ instance that runs on the monitoring server.
- **MonitoringRabbitUseSSL** - This parameter is not currently available for availability monitoring.
- **SensuClientCustomConfig** - Specify additional Sensu client configuration. Defines the OpenStack credentials to be used, including username/password, auth_url, tenant, and region.

3.3. INSTALL OPERATIONAL TOOLS ON OVERCLOUD NODES

Include the modified YAML files with your `openstack overcloud deploy` command to install the Sensu client and Fluentd tools on all overcloud nodes. For example:

```
```
CHAPTER 4. INSTALL THE SERVER-SIDE COMPONENTS

Note

Red Hat does not support the server-side components and their deployment process.

You can use the `opstools-ansible` playbook to install the server-side components onto Red Hat Enterprise Linux 7. These server-side components include availability monitoring and centralized logging services that complement the Red Hat-supported client-side components. The most tested `opstools-ansible` scenario is the deployment of server-side components onto CentOS 7. Detailed instructions can be found in the README.md: https://github.com/centos-opstools/opstools-ansible
CHAPTER 5. MONITOR THE OPENSTACK PLATFORM

See the Sensu documentation for further details about the Sensu stack infrastructure: https://sensuapp.org/docs/latest/overview/architecture.html

Red Hat supplies a set of check scripts in the `osops-tools-monitoring-oschecks` package. The majority of the check scripts only check the API connection to the OpenStack component. However, certain scripts also perform additional OpenStack resource tests for OpenStack Compute (nova), OpenStack Block Storage (cinder), OpenStack Image (glance), and OpenStack Networking (neutron). For example, the OpenStack Identity (keystone) API check gives the following result when `keystone` is running:

```
OK: Got a token, Keystone API is working.
```
CHAPTER 6. VALIDATE THE SENSU CLIENT INSTALLATION

1. Check the status of the `sensu-client` on each overcloud node:
   # systemctl status sensu-client

2. Review the error log for any issues: `/var/log/sensu/sensu-client.log`

3. Verify that each overcloud node has the `/etc/sensu/conf.d/rabbitmq.json` file that sets the IP address of the monitoring server.
CHAPTER 7. REVIEW THE STATE OF A NODE

If you have a deployment of the Uchiwa dashboard, you can use it with the Sensu server to review the state of your nodes:

1. Login to the Uchiwa dashboard and click the **Data Center** tab to confirm that the Data Center is operational.

   ```plaintext
   http://<SERVER_IP_ADDRESS>:3000
   ```

2. Check that all overcloud nodes are in a **Connected** state.

3. At a suitable time, reboot one of the overcloud nodes and review the rebooted node’s status in the Uchiwa dashboard. After the reboot completes, verify that the node successfully reconnects to the Sensu server and starts executing checks.
CHAPTER 8. REVIEW THE STATE OF AN OPENSTACK SERVICE

This example tests the monitoring of the `openstack-ceilometer-central` service.

1. Confirm that the `openstack-ceilometer-central` service is running:

   ```bash
   systemctl status openstack-ceilometer-central.service
   ```

2. Connect to the Uchiwa dashboard and confirm that a successful `ceilometer` check is present and running as defined in the `ceilometer` JSON file.

3. Stop the `openstack-ceilometer-central` service.

   ```bash
   systemctl stop openstack-ceilometer-central.service
   ```

   **Note**
   
   This may disrupt services, so run this test at an appropriate time.

4. Log in to the Uchiwa dashboard and confirm that the failed `ceilometer` check has been reported.

5. Start the `openstack-ceilometer-central` service:

   ```bash
   systemctl start openstack-ceilometer-central.service
   ```

6. Log in to the Uchiwa dashboard and view the time interval between the `ceilometer` check reports to confirm that the check runs in the time interval defined in the `ceilometer` JSON file.