Red Hat OpenStack Platform 8
Logging, Monitoring, and Troubleshooting Guide

An In-Depth Guide to OpenStack Logging, Monitoring, and Troubleshooting

OpenStack Team

Red Hat OpenStack Platform 8 Logging, Monitoring, and Troubleshooting Guide

---

An In-Depth Guide to OpenStack Logging, Monitoring, and Troubleshooting

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

This guide provides a detailed overview on logging and monitoring a Red Hat OpenStack Platform environment, and how to solve problems.

Table of Contents

PREFACE

This document provides an overview of the logging and monitoring capabilities that are availbale in a Red Hat OpenStack Platform environment, and how to troubleshoot possible issues.

CHAPTER 1. LOGGING

Red Hat OpenStack Platform writes informational messages to specific log files; you can use these messages for troubleshooting and monitoring system events.

Note

You need not attach the individual log files to your support cases manually. All the required information will be gathered automatically by the **`sosreport`** utility, which is described in Chapter 3, *Troubleshooting*.

1.1. LOG FILES FOR OPENSTACK SERVICES

Each OpenStack component has a separate logging directory containing files specific to a running service.

1.1.1. Bare Metal Provisioning (ironic) Log Files

Service	Service Name	Log Path
OpenStack Ironic API	openstack-ironic-api.service	/var/log/ironic/ironic-api.log
OpenStack Ironic Conductor	openstack-ironic-conductor.service	/var/log/ironic/ironic-conductor.log

1.1.2. Block Storage (cinder) Log Files

Service	Service Name	Log Path
Block Storage API	openstack-cinder-api.service	/var/log/cinder/api.log
Block Storage Backup	openstack-cinder-backup.service	/var/log/cinder/backup.log
Informational messages	The cinder-manage command	/var/log/cinder/cinder-manage.log
Block Storage Scheduler	openstack-cinder-scheduler.service	/var/log/cinder/scheduler.log

Service	Service Name	Log Path
Block Storage Volume	openstack-cinder-volume.service	/var/log/cinder/volume.log

1.1.3. Compute (nova) Log Files

Service	Service Name	Log Path
OpenStack Compute API service	openstack-nova-api.service	/var/log/nova/nova-api.log
OpenStack Compute certificate server	openstack-nova-cert.service	/var/log/nova/nova-cert.log
OpenStack Compute service	openstack-nova-compute.service	/var/log/nova/nova-compute.log
OpenStack Compute Conductor service	openstack-nova-conductor.service	/var/log/nova/nova-conductor.log
OpenStack Compute VNC console authentication server	openstack-nova-consoleauth.service	/var/log/nova/nova-consoleauth.log
Informational messages	nova-manage command	/var/log/nova/nova-manage.log
OpenStack Compute NoVNC Proxy service	openstack-nova-novncproxy.service	/var/log/nova/nova-novncproxy.log
OpenStack Compute Scheduler service	openstack-nova-scheduler.service	/var/log/nova/nova-scheduler.log

1.1.4. Dashboard (horizon) Log Files

Service	Service Name	Log Path
Log of certain user interactions	Dashboard interface	/var/log/horizon/horizon.log

The Apache HTTP server uses several additional log files for the Dashboard web interface, which can be accessed using a web browser or command-line clients (keystone, nova). The following log files can be helpful in tracking the usage of the Dashboard and diagnosing faults:

Purpose	Log Path
All processed HTTP requests	/var/log/httpd/horizon_access.log
HTTP errors	/var/log/httpd/horizon_error.log
Admin-role API requests	/var/log/httpd/keystone_wsgi_admin_access.log
Admin-role API errors	/var/log/httpd/keystone_wsgi_admin_error.log
Member-role API requests	/var/log/httpd/keystone_wsgi_main_access.log
Member-role API errors	/var/log/httpd/keystone_wsgi_main_error.log

Note

There is also **`/var/log/httpd/default_error.log`**, which stores errors reported by other web services running on the same host.

1.1.5. Data Processing (sahara) Log Files

Service	Service Name	Log Path
Sahara API Server	openstack-sahara-all.service openstack-sahara-api.service	/var/log/sahara/sahara-all.log /var/log/messages
Sahara Engine Server	openstack-sahara-engine.service	/var/log/messages

1.1.6. Database as a Service (trove) Log Files

Service	Service Name	Log Path
OpenStack Trove API Service	openstack-trove-api.service	/var/log/trove/trove-api.log
OpenStack Trove Conductor Service	openstack-trove-conductor.service	/var/log/trove/trove-conductor.log
OpenStack Trove guestagent Service	openstack-trove-guestagent.service	/var/log/trove/logfile.txt
OpenStack Trove taskmanager Service	openstack-trove-taskmanager.service	/var/log/trove/trove-taskmanager.log

1.1.7. Identity Service (keystone) Log Files

Service	Service Name	Log Path
OpenStack Identity Service	openstack-keystone.service	/var/log/keystone/keystone.log

1.1.8. Image Service (glance) Log Files

Service	Service Name	Log Path
OpenStack Image Service API server	openstack-glance-api.service	/var/log/glance/api.log
OpenStack Image Service Registry server	openstack-glance-registry.service	/var/log/glance/registry.log

1.1.9. Networking (neutron) Log Files

Service	Service Name	Log Path
OpenStack Neutron DHCP Agent	neutron-dhcp-agent.service	/var/log/neutron/dhcp-agent.log

Service	Service Name	Log Path
OpenStack Networking Layer 3 Agent	neutron-l3-agent.service	/var/log/neutron/l3-agent.log
Metadata agent service	neutron-metadata-agent.service	/var/log/neutron/metadata-agent.log
Metadata namespace proxy	n/a	/var/log/neutron/neutron-ns-metadata-proxy-*UUID*.log
Open vSwitch agent	neutron-openvswitch-agent.service	/var/log/neutron/openvswitch-agent.log
OpenStack Networking service	neutron-server.service	/var/log/neutron/server.log

1.1.10. Object Storage (swift) Log Files

OpenStack Object Storage sends logs to the system logging facility only.

Note

By default, all Object Storage log files to /var/log/swift/swift.log, using the local0, local1, and local2 syslog facilities.

The log messages of Object Storage are classified into two broad categories: those by REST API services and those by background daemons. The API service messages contain one line per API request, in a manner similar to popular HTTP servers; both the frontend (Proxy) and backend (Account, Container, Object) services post such messages. The daemon messages are less structured and typically contain human-readable information about daemons performing their periodic tasks. However, regardless of which part of Object Storage produces the message, the source identity is always at the beginning of the line.

An example of a proxy message:

```
Apr 20 15:20:34 rhev-a24c-01 proxy-server: 127.0.0.1 127.0.0.1
20/Apr/2015/19/20/34 GET
/v1/AUTH_zaitcev%3Fformat%3Djson%26marker%3Dtestcont HTTP/1.0 200 -
python-swiftclient-2.1.0 AUTH_tk737d6... - 2 - txc454fa8ea4844d909820a-
0055355182 - 0.0162 - - 1429557634.806570053 1429557634.822791100
```

An example of ad-hoc messages from background daemons:

```
Apr 27 17:08:15 rhev-a24c-02 object-auditor: Object audit (ZBF). Since
Mon Apr 27 21:08:15 2015: Locally: 1 passed, 0 quarantined, 0 errors
files/sec: 4.34 , bytes/sec: 0.00, Total time: 0.23, Auditing time:
```

```
0.00, Rate: 0.00
Apr 27 17:08:16 rhev-a24c-02 object-auditor: Object audit (ZBF)
"forever" mode completed: 0.56s. Total quarantined: 0, Total errors: 0,
Total files/sec: 14.31, Total bytes/sec: 0.00, Auditing time: 0.02,
Rate: 0.04
Apr 27 17:08:16 rhev-a24c-02 account-replicator: Beginning replication
run
Apr 27 17:08:16 rhev-a24c-02 account-replicator: Replication run OVER
Apr 27 17:08:16 rhev-a24c-02 account-replicator: Attempted to replicate
5 dbs in 0.12589 seconds (39.71876/s)
Apr 27 17:08:16 rhev-a24c-02 account-replicator: Removed 0 dbs
Apr 27 17:08:16 rhev-a24c-02 account-replicator: 10 successes, 0
failures
```

1.1.11. Orchestration (heat) Log Files

Service	Service Name	Log Path
OpenStack Heat API Service	openstack-heat-api.service	/var/log/heat/heat-api.log
Openstack Heat Engine Service	openstack-heat-engine.service	/var/log/heat/heat-engine.log
Orchestration service events	n/a	/var/log/heat/heat-manage.log

1.1.12. Shared Filesystem Service (manila) Log Files

Service	Service Name	Log Path
OpenStack Manila API Server	openstack-manila-api.service	/var/log/manila/api.log
OpenStack Manila Scheduler	openstack-manila-scheduler.service	/var/log/manila/scheduler.log
OpenStack Manila Share Service	openstack-manila-share.service	/var/log/manila/share.log

Note

Some information from the Manila Python library can also be logged in **/var/log/manila/manila-manage.log**.

1.1.13. Telemetry (ceilometer) Log Files

Service	Service Name	Log Path
OpenStack ceilometer notification agent	openstack-ceilometer-notification.service	/var/log/ceilometer/agent-notification.log
OpenStack ceilometer alarm evaluation	openstack-ceilometer-alarm-evaluator.service	/var/log/ceilometer/alarm-evaluator.log
OpenStack ceilometer alarm notification	openstack-ceilometer-alarm-notifier.service	/var/log/ceilometer/alarm-notifier.log
OpenStack ceilometer API	openstack-ceilometer-api.service	/var/log/ceilometer/api.log
Informational messages	MongoDB integration	/var/log/ceilometer/ceilometer-dbsync.log
OpenStack ceilometer central agent	openstack-ceilometer-central.service	/var/log/ceilometer/central.log
OpenStack ceilometer collection	openstack-ceilometer-collector.service	/var/log/ceilometer/collector.log
OpenStack ceilometer compute agent	openstack-ceilometer-compute.service	/var/log/ceilometer/compute.log

1.1.14. Log Files for Supporting Services

The following services are used by the core OpenStack components and have their own log directories and files.

Service	Service Name	Log Path
Message broker (RabbitMQ)	rabbitmq-server.service	/var/log/rabbitmq/rabbit@*short_hostname*.log /var/log/rabbitmq/rabbit@*short_hostname*-sasl.log (for Simple Authentication and Security Layer related log messages)
Database server (MariaDB)	mariadb.service	/var/log/mariadb/mariadb.log
Document-oriented database (MongoDB)	mongod.service	/var/log/mongodb/mongodb.log
Virtual network switch (Open vSwitch)	openvswitch-nonetwork.service	/var/log/openvswitch/ovsdb-server.log /var/log/openvswitch/ovs-vswitchd.log

1.2. CONFIGURE LOGGING OPTIONS

Each component maintains its own separate logging configuration in its respective configuration file. For example, in Compute, these options are set in **`/etc/nova/nova.conf`**:

- Increase the level of informational logging by enabling debugging. This option greatly increases the amount of information captured, so you may want to consider using it only temporarily, or first reviewing your log rotation settings.

```
debug=True
```

- Enable verbose logging:

```
verbose=True
```

- Change the log file path:

```
log_dir=/var/log/nova
```

- Send your logs to a central syslog server:

```
use_syslog=True
syslog_log_facility=LOG_USER
```

Note

Options are also available for timestamp configuration and log formatting, among others. Review the component’s configuration file for additional logging options.

1.3. REMOTE LOGGING INSTALLATION AND CONFIGURATION

1.3.1. Introduction to Remote Logging

All systems generate and update log files recording their actions and any problems they encounter. In a distributed or cloud computing environment that contains many systems, collecting these log files in a central location simplifies debugging.

The **rsyslog** service provides facilities both for running a centralized logging server and for configuring individual systems to send their log files to the centralized logging server. This is referred to as configuring the systems for *remote logging*.

1.3.2. Install rsyslog Server

The **rsyslog** package must be installed on the system that you intend to use as a centralized logging server and all systems that will be configured to send logs to it. To do so, log in as the *root* user and install the **rsyslog** package:

```
# yum install rsyslog
```

The **rsyslog** package is installed and ready to be configured.

1.3.3. Configure rsyslog on the Centralized Logging Server

The steps in this procedure must be followed on the system that you intend to use as your centralized logging sever. All steps in this procedure must be run while logged in as the *root* user.

1. Configure SELinux to allow **rsyslog** traffic.

   ```
   # semanage port -a -t syslogd_port_t -p udp 514
   ```

2. Open the **/etc/rsyslog.conf** file in a text editor.

   a. Add the following lines to the file, defining the location logs will be saved to:

   ```
   $template TmplMsg, "/var/log/%HOSTNAME%/%PROGRAMNAME%.log"
   $template TmplAuth, "/var/log/%HOSTNAME%/%PROGRAMNAME%.log"

   authpriv.*   ?TmplAuth
   *.info,mail.none,authpriv.none,cron.none   ?TmplMsg
   ```

   b. Remove the comment character (**#**) from the beginning of these lines in the file:

   ```
   #$ModLoad imudp
   #$UDPServerRun 514
   ```

   c. Save the changes to the **/etc/rsyslog.conf** file.

Your centralized log server is now configured to receive and store log files from the other systems in your environment.

1.3.4. Configure rsyslog on Individual Nodes

Apply the steps listed in this procedure to each of your systems to configure them to send logs to a centralized log server. All steps listed in this procedure must be performed while logged in as the *root* user.

1. Edit the **/etc/rsyslog.conf**, and specify the address of your centralized log server by adding the following:

   ```
   *.*    @YOURSERVERADDRESS:YOURSERVERPORT
   ```

   Replace *YOURSERVERADDRESS* with the address of the centralized logging server. Replace *YOURSERVERPORT* with the port on which the rsyslog service is listening. For example:

   ```
   *.*   @192.168.20.254:514
   ```

   Or:

   ```
   *.*   @@log-server.example.com:514
   ```

   The single @ sign specifies the UDP protocol for transmission. Use @@ to specify the TCP protocol for transmission.

   > **Important**
   >
   > The use of the wildcard (*****) character in these example configurations indicates to rsyslog that log entries from all log facilities and of all log priorities must be sent to the remote rsyslog server.
   >
   > For information on applying more precise filtering of log files refer to the manual page for the rsyslog configuration file, **rsyslog.conf**. Access the manual page by running **man rsyslog.conf**.

2. Once the *rsyslog* service is started or restarted the system will send all log messages to the centralized logging server.

1.3.5. Start the rsyslog Server

The **rsyslog** service must be running on both the centralized logging server and the systems attempting to log to it.

The steps in this procedure must be performed while logged in as the *root* user.

1. Start the rsyslog service:

   ```
   # service rsyslog start
   ```

2. Ensure the rsyslog service starts automatically in the future:

   ```
   # chkconfig rsyslog on
   ```

The **rsyslog** service has been started. The service will start sending or receiving log messages based on its local configuration.

CHAPTER 2. MONITORING USING THE TELEMETRY SERVICE

For help with the ceilometer command, use:

```
# ceilometer help
```

For help with the subcommands, use:

```
# ceilometer help subcommand
```

2.1. VIEW EXISTING ALARMS

To list configured Telemetry alarms, use:

```
# ceilometer alarm-list
```

To list configured meters for a resource, use:

```
# ceilometer meter-list --query resource=UUID
+---------------------------+------------+-----------+-----------+---
-------+----------+
| Name                      | Type       | Unit      | Resource  |
User ID  | Project  |
+---------------------------+------------+-----------+-----------+---
-------+----------+
| cpu                       | cumulative | ns        | 5056eda...|
b0e500...| f23524...|
| cpu_util                  | gauge      | %         | 5056eda...|
b0e500...| f23524...|
| disk.ephemeral.size       | gauge      | GB        | 5056eda...|
b0e500...| f23524...|
| disk.read.bytes           | cumulative | B         | 5056eda...|
b0e500...| f23524...|

| instance                  | gauge      | instance  | 5056eda...|
b0e500...| f23524...|
| instance:m1.tiny          | gauge      | instance  | 5056eda...|
b0e500...| f23524...|
| memory                    | gauge      | MB        | 5056eda...|
b0e500...| f23524...|
| vcpus                     | gauge      | vcpu      | 5056eda...|
b0e500...| f23524...|
+---------------------------+------------+-----------+--------------
-------------------+
```

Where *UUID* is the resource ID for an existing resource (for example, an instance, image, or volume).

2.2. CONFIGURE AN ALARM

To configure an alarm to activate when a threshold value is crossed, use the **`ceilometer alarm-`**

threshold-create command with the following syntax:

```
# ceilometer alarm-threshold-create --name alarm-name [--description
alarm-text] --meter-name meter-name --threshold value
```

Example

To configure an alarm that activates when the average CPU utilization for an individual instance exceeds 50% for three consecutive 600s (10 minute) periods, use:

```
# ceilometer alarm-threshold-create --name cpu_high --description 'CPU
usage high' --meter-name cpu_usage_high --threshold 50 --comparison-
operator gt --statistic avg --period 600 --evaluation-periods 3 --
alarm-action 'log://' --query resource_id=5056eda6-8a24-4f52-9cc4-
c3ddb6fb4a69
```

In this example, the notification action is a log message.

To edit an existing threshold alarm, use the **ceilometer alarm-threshold-update** command together with the alarm ID, followed by one or more options to be updated.

Example

To increase the alarm threshold to 75%, use:

```
# ceilometer alarm-threshold-update 35addb25-d488-4a74-a038-
076aad3a3dc3 --threshold=75
```

2.3. DISABLE OR DELETE AN ALARM

To disable an alarm, use:

```
# ceilometer alarm-threshold-update --enabled False ALARM_ID
```

To delete an alarm, use:

```
# ceilometer alarm-delete ALARM_ID
```

2.4. VIEW SAMPLES

To list all the samples for a particular meter name, use:

```
# ceilometer sample-list --meter METER_NAME
```

To list samples only for a particular resource within a range of time stamps, use:

```
# ceilometer sample-list --meter METER_NAME --query
'resource_id=INSTANCE_ID;timestamp>_START_TIME_;timestamp>=END_TIME'
```

Where *START_TIME* and *END_TIME* are in the form *iso-dateThh:mm:ss*.

Example

To query an instance for samples taken between **13:10:00** and **14:25:00**, use:

```
# ceilometer sample-list --meter cpu --query 'resource_id=5056eda6-
8a24-4f52-9cc4-c3ddb6fb4a69;timestamp>2015-01-
12T13:10:00;timestamp>=2015-01-12T14:25:00'
+-------------------+------+------------+--------------+------+----
-----------------+
| Resource ID       | Name | Type       | Volume       | Unit |
Timestamp           |
+-------------------+------+------------+--------------+------+----
-----------------+
| 5056eda6-8a24-... | cpu  | cumulative | 3.5569e+11   | ns   | 2015-
01-12T14:21:44 |
| 5056eda6-8a24-... | cpu  | cumulative | 3.0041e+11   | ns   | 2015-
01-12T14:11:45 |
| 5056eda6-8a24-... | cpu  | cumulative | 2.4811e+11   | ns   | 2015-
01-12T14:01:54 |
| 5056eda6-8a24-... | cpu  | cumulative | 1.3743e+11   | ns   | 2015-
01-12T13:30:54 |
| 5056eda6-8a24-... | cpu  | cumulative | 84710000000.0 | ns   | 2015-
01-12T13:20:54 |
| 5056eda6-8a24-... | cpu  | cumulative | 31170000000.0 | ns   | 2015-
01-12T13:10:54 |
+-------------------+------+------------+--------------+------+----
-----------------+
```

2.5. CREATE A SAMPLE

Samples can be created for sending to the Telemetry service and they need not correspond to a previously defined meter. Use the following syntax:

```
# ceilometer sample-create --resource_id RESOURCE_ID --meter-name
METER_NAME --meter-type METER_TYPE --meter-unit METER_UNIT --sample-
volume SAMPLE_VOLUME
```

Where *METER_TYPE* can be one of:

- Cumulative — a running total
- Delta — a change or difference over time
- Gauge — a discrete value

Example

```
# ceilometer sample-create -r 5056eda6-8a24-4f52-9cc4-c3ddb6fb4a69 -m
On_Time_Mins --meter-type cumulative --meter-unit mins --sample-volume
0
+-------------------+--------------------------------------------+
| Property          | Value                                      |
+-------------------+--------------------------------------------+
| message_id        | 521f138a-9a84-11e4-8058-525400ee874f       |
| name              | On_Time_Mins                               |
```

```
| project_id        | f2352499957d4760a00cebd26c910c0f           |
| resource_id       | 5056eda6-8a24-4f52-9cc4-c3ddb6fb4a69       |
| resource_metadata | {}                                         |
| source            | f2352499957d4760a00cebd26c910c0f:openstack |
| timestamp         | 2015-01-12T17:56:23.179729                 |
| type              | cumulative                                 |
| unit              | mins                                       |
| user_id           | b0e5000684a142bd89c4af54381d3722           |
| volume            | 0.0                                        |
+-------------------+--------------------------------------------+
```

Where **volume**, normally the value obtained as a result of the sampling action, is in this case the value being created by the command.

Note

Samples are not updated because the moment a sample is created, it is sent to the Telemetry service. Samples are essentially messages, which is why they have a message ID. To create new samples, repeat the **sample-create** command and update the **--sample-volume** value.

2.6. VIEW CLOUD USAGE STATISTICS

OpenStack administrators can use the dashboard to view cloud statistics.

1. As an admin user in the dashboard, select **Admin** > **System** > **Resource Usage**.
2. Click one of the following:
   - Daily Report — View a report of daily usage per project. Select the date range and a limit for the number of projects, and click **Generate Report**; the daily usage report is displayed.
   - Stats — View a graph of metrics grouped by project. Select the values and time period using the drop-down menus; the displayed graph is automatically updated.

The **ceilometer** command line client can also be used for viewing cloud usage statics.

Example

To view all the statistics for the **cpu_util** meter, use:

```
# ceilometer statistics --meter cpu_util
+--------+----------------+---------------+-----+-----+------+------
-+------+--------
| Period | Period Start   |Period End     | Max | Min | Avg  | Sum   |
Count| Dura...
+--------+----------------+---------------+-----+-----+------+------
-+------+--------
| 0      | 2015-01-09T14: |2015-01-09T14:2| 9.44| 0.0 | 6.75 | 337.94|
50   | 2792...
+--------+----------------+---------------+-----+-----+------+------
-+------+--------
```

Example

Statistics can be restricted to a specific resource by means of the **--query** option, and restricted to a specific range by means of the **timestamp** option.

```
# ceilometer statistics --meter cpu_util --query 'resource_id=5056eda6-
8a24-4f52-9cc4-c3ddb6fb4a69;timestamp>2015-01-
12T13:00:00;timestamp<=2015-01-13T14:00:00'
+--------+---------------+--------------+-----+-----+-----+------
-+------+--------
| Period | Period Start   |Period End     | Max | Min | Avg  | Sum   |
Count| Dura...
+--------+---------------+--------------+-----+-----+-----+------
-+------+--------
| 0      | 2015-01-12T20:1|2015-01-12T20:1| 9.44| 5.95| 8.90 | 347.10|
39   | 2465...
+--------+---------------+--------------+-----+-----+-----+------
-+------+--------
```

2.7. USING THE TIME-SERIES-DATABASE-AS-A-SERVICE

Time-Series-Database-as-a-Service (gnocchi) is a multi-tenant, metrics and resource database. It is designed to store metrics at a very large scale while providing access to metrics and resources information to operators and users.

Currently, the TSDaaS uses the Identity service for authentication, and Ceph, Object Storage to store data.

TDSaaS provides the **statsd** deamon that is compatible with the **statsd** protocol and can listen to the metrics sent over the network, named **gnocchi-statsd**. In order to enable **statsd** support in TDSaaS, you need to configure the **[statsd]** option in the configuration file. The resource ID parameter is used as the main generic resource where all the metrics are attached, a user and project ID that are associated with the resource and metrics, and an archive policy name thatis used to create the metrics.

All the metrics will be created dynamically as the metrics are sent to **gnocchi-statsd**, and attached with the provided name to the resource ID you configured. For more information on installing and configuring TSDaaS, see the **Install Time-Series-Database-as-a-Service** chapter in the **Installation Reference Guide** available at: https://access.redhat.com/documentation/en/red-hat-enterprise-linux-openstack-platform/

Note

Time-Series-Database-as-a-Service (gnocchi) is marked as Technology Preview for the Red Hat OpenStack Platform 8.

For more information on the support scope for features marked as technology previews, see https://access.redhat.com/support/offerings/techpreview/

2.7.1. Running Time-Series-Database-as-a-Service

Run Time-Series-Database-as-a-Service (TSDaaS) by running the HTTP server and metric daemon:

```
# gnocchi-api
# gnocchi-metricd
```

2.7.2. Running As A WSGI Application

You can run the TSDaaS through a WSGI service such as **mod_wsgi** or any other WSGI application. The file **gnocchi/rest/app.wsgi** provided with TSDaaS allows you to enable Gnocchi as a WSGI application.

The TSDaaS API tier runs using WSGI. This means it can be run using Apache **httpd** and **mod_wsgi**, or another HTTP daemon such as **uwsgi**. You should configure the number of processes and threads according to the number of CPUs you have, usually around **1.5 × number of CPUs**. If one server is not enough, you can spawn any number of new API servers to scale Gnocchi out, even on different machines.

2.7.3. `metricd` Workers

By default, the **gnocchi-metricd** daemon spans all your CPU power in order to maximize CPU utilisation when computing metric aggregation. You can use the **gnocchi status** command to query the HTTP API and get the cluster status for metric processing. This command displays the number of metrics to process, known as the processing backlog for the **gnocchi-metricd**. As long as this backlog is not continuously increasing, that means that **gnocchi-metricd** is able to cope with the amount of metric that are being sent. If the number of measure to process is continuously increasing, you will need to (maybe temporarily) increase the number of the **gnocchi-metricd** daemons. You can run any number of metricd daemons on any number of servers.

2.7.4. Monitoring the Time-Series-Database-as-a-Service

The **/v1/status** endpoint of the HTTP API returns various information, such as the number of measures to process (measures backlog), which you can easily monitor. Making sure that the HTTP server and the **gnocchi-metricd** daemon are running and are not writing anything alarming in their logs is a sign of good health of the overall system.

2.7.5. Backing up and Restoring Time-Series-Database-as-a-Service

In order to be able to recover from an unfortunate event, you need to backup both the index and the storage. That means creating a database dump (PostgreSQL or MySQL) and doing snapshots or copies of your data storage (Ceph, Swift or your file system). The procedure to restore is: restore your index and storage backups, reinstall TSDaaS if necessary, and restart it.

CHAPTER 3. TROUBLESHOOTING

This chapter contains logging and support information to assist with troubleshooting your Red Hat OpenStack Platform deployment.

3.1. SUPPORT

If client commands fail or you run into other issues, contact Red Hat Technical Support with a description of what happened, the full console output, all log files referenced in the console output, and an **sosreport** from the node that is (or might be) in trouble. For example, if you encounter a problem on the compute level, run **sosreport** on the Nova node, or if it is a networking issue, run the utility on the Neutron node. For general deployment issues, it is best to run **sosreport** on the cloud controller.

For information about the **sosreport** command (**sos** package), refer to What is a sosreport and how to create one in Red Hat Enterprise Linux 4.6 and later.

Check also the **/var/log/messages** file for any hints.

3.2. TROUBLESHOOT IDENTITY CLIENT (KEYSTONE) CONNECTIVITY PROBLEMS

When the Identity client (**keystone**) is unable to contact the Identity service it returns an error:

```
Unable to communicate with identity service: [Errno 113] No route to
host. (HTTP 400)
```

To debug the issue check for these common causes:

Identity service is down

On the system hosting the Identity service check the service status:

```
# openstack-status | grep keystone
openstack-keystone:                          active
```

If the service is not running then log in as the root user and start it.

```
# service openstack-keystone start
```

Firewall is not configured properly

The firewall might not be configured to allow TCP traffic on ports **5000** and **35357**. If so, see *Configure the Firewall to Allow Identity Service Traffic* in the Installation Reference for instructions on how to correct this.

Service Endpoints not defined correctly

On the system hosting the Identity service check that the endpoints are defined correctly.

1. Obtain the administration token:

```
# grep admin_token /etc/keystone/keystone.conf
admin_token = 0292d404a88c4f269383ff28a3839ab4
```

2. Determine the correct administration endpoint for the Identity service:

```
http://IP:35357/VERSION
```

Replace *IP* with the IP address or host name of the system hosting the Identity service. Replace *VERSION* with the API version (**v2.0**, or **v3**) that is in use.

3. Unset any pre-defined Identity service related environment variables:

```
# unset OS_USERNAME OS_TENANT_NAME OS_PASSWORD OS_AUTH_URL
```

4. Use the administration token and endpoint to authenticate with the Identity service. Confirm that the Identity service endpoint is correct:

```
# keystone --os-token=TOKEN \
            --os-endpoint=ENDPOINT \
            endpoint-list
```

Verify that the listed **publicurl**, **internalurl**, and **adminurl** for the Identity service are correct. In particular ensure that the IP addresses and port numbers listed within each endpoint are correct and reachable over the network.

If these values are incorrect then see *Create the Identity Service Endpoint* in the Installation Reference for information on adding the correct endpoint. Once the correct endpoints have been added, remove any incorrect endpoints using the **endpoint-delete** action of the **keystone** command:

```
# keystone --os-token=TOKEN \
            --os-endpoint=ENDPOINT \
            endpoint-delete ID
```

Replace *TOKEN* and *ENDPOINT* with the values identified previously. Replace *ID* with the identity of the endpoint to remove as listed by the **endpoint-list** action.

3.3. TROUBLESHOOT OPENSTACK NETWORKING ISSUES

This section discusses the different commands you can use and procedures you can follow to troubleshoot the OpenStack Networking service issues.

Debugging Networking Device

- Use the **ip a** command to display all the physical and virtual devices.
- Use the **ovs-vsctl show** command to display the interfaces and bridges in a virtual switch.
- Use the **ovs-dpctl show** command to show datapaths on the switch.

Tracking Networking Packets

- Use the **tcpdump** command to see where packets are not getting through.

```
# tcpdump -n -i INTERFACE -e -w FILENAME
```

Replace *INTERFACE* with the name of the network interface to see where the packets are not getting through. The interface name can be the name of the bridge or host Ethernet device.

The **-e** flag ensures that the link-level header is dumped (in which the **vlan** tag will appear).

The **-w** flag is optional. You can use it only if you want to write the output to a file. If not, the output is written to the standard output (**stdout**).

For more information about **tcpdump**, refer to its manual page by running **man tcpdump**.

Debugging Network Namespaces

- Use the **ip netns list** command to list all known network namespaces.
- Use the **ip netns exec** command to show routing tables inside specific namespaces.

```
# ip netns exec NAMESPACE_ID bash
# route -n
```

Start the **ip netns exec** command in a bash shell so that subsequent commands can be invoked without the **ip netns exec** command.

3.4. TROUBLESHOOT NETWORKS AND ROUTES TAB DISPLAY ISSUES IN THE DASHBOARD

The *Networks* and *Routers* tabs only appear in the dashboard when the environment is configured to use OpenStack Networking. In particular note that by default the PackStack utility currently deploys Nova Networking and as such in environments deployed in this manner the tab will not be visible.

If OpenStack Networking is deployed in the environment but the tabs still do not appear ensure that the service endpoints are defined correctly in the Identity service, that the firewall is allowing access to the endpoints, and that the services are running.

3.5. TROUBLESHOOT INSTANCE LAUNCHING ERRORS IN THE DASHBOARD

When using the dashboard to launch instances if the operation fails, a generic **ERROR** message is displayed. Determining the actual cause of the failure requires the use of the command line tools.

Use the **nova list** command to locate the unique identifier of the instance. Then use this identifier as an argument to the **nova show** command. One of the items returned will be the error condition. The most common value is **NoValidHost**.

This error indicates that no valid host was found with enough available resources to host the instance. To work around this issue, consider choosing a smaller instance size or increasing the overcommit allowances for your environment.

Note

To host a given instance, the compute node must have not only available CPU and RAM resources but also enough disk space for the ephemeral storage associated with the instance.

3.6. TROUBLESHOOT KEYSTONE V3 DASHBOARD AUTHENTICATION

django_openstack_auth is a pluggable Django authentication back end, that works with Django's contrib.auth framework, to authenticate a user against the OpenStack Identity service API. Django_openstack_auth uses the token object to encapsulate user and Keystone related information. The dashboard uses the token object to rebuild the Django user object.

The token object currently stores:

- Keystone token
- User information
- Scope
- Roles
- Service catalog

The dashboard uses Django's sessions framework for handling user session data. The following is a list of numerous session back ends available, which are controlled through the SESSION_ENGINE setting in your local_settings.py file:

- Local Memory Cache
- Memcached
- Database
- Cached Database
- Cookies

In some cases, particularly when a signed cookie session back end is used and, when having many or all services enabled all at once, the size of cookies can reach its limit and the dashboard can fail to log in. One of the reasons for the growth of cookie size is the service catalog. As more services are registered, the bigger the size of the service catalog would be.

In such scenarios, to improve the session token management, include the following configuration settings for logging in to the dashboard, especially when using Keystone v3 authentication.

1. In /usr/share/openstack-dashboard/openstack_dashboard/settings.py add the following configuration:

```
DATABASES =
{
  'default':
  {
    'ENGINE': 'django.db.backends.mysql',
```

```
        'NAME': 'horizondb',
        'USER': 'User Name',
        'PASSWORD': 'Password',
        'HOST': 'localhost',
       }
    }
```

2. In the same file, change SESSION_ENGINE to:

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cached_db'
```

3. Connect to the database service using the mysql command, replacing USER with the user name by which to connect. The USER must be a root user (or at least as a user with the correct permission: create db).

```
# mysql -u USER -p
```

4. Create the Horizon database.

```
mysql > create database horizondb;
```

5. Exit the mysql client.

```
mysql > exit
```

6. Change to the openstack_dashboard directory and sync the database using:

```
# cd /usr/share/openstack-dashboard/openstack_dashboard
$ ./manage.py syncdb
```

You do not need to create a superuser, so answer 'n' to the question.

7. Restart Apache http server. For Red Hat Enterprise Linux:

```
#service httpd restart
```