Red Hat OpenStack Platform 11

Logging, Monitoring, and Troubleshooting Guide

An In-Depth Guide to OpenStack Logging, Monitoring, and Troubleshooting
Red Hat OpenStack Platform 11 Logging, Monitoring, and Troubleshooting Guide

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

OpenStack Team
rhos-docs@redhat.com
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**

**CHAPTER 1. ABOUT THIS GUIDE** .................................................................................................................. 3

**CHAPTER 2. LOGGING** ................................................................................................................................. 4
  2.1. LOG FILES FOR OPENSTACK SERVICES ......................................................................................... 4
    2.1.1. Bare Metal Provisioning (ironic) Log Files ............................................................................. 4
    2.1.2. Block Storage (cinder) Log Files ......................................................................................... 4
    2.1.3. Compute (nova) Log Files ................................................................................................. 4
    2.1.4. Dashboard (horizon) Log Files .......................................................................................... 5
    2.1.5. Data Processing (sahara) Log Files ................................................................................... 6
    2.1.6. Database as a Service (trove) Log Files ............................................................................ 6
    2.1.7. Identity Service (keystone) Log Files .................................................................................. 6
    2.1.8. Image Service (glance) Log Files ....................................................................................... 6
    2.1.9. Networking (neutron) Log Files .......................................................................................... 7
    2.1.10. Object Storage (swift) Log Files ......................................................................................... 7
    2.1.11. Orchestration (heat) Log Files ............................................................................................ 8
    2.1.12. Shared Filesystem Service (manila) Log Files .................................................................. 8
    2.1.13. Telemetry (ceilometer) Log Files ....................................................................................... 8
    2.1.14. Log Files for Supporting Services ...................................................................................... 9
  2.2. CONFIGURE LOGGING OPTIONS .................................................................................................. 9
  2.3. REMOTE LOGGING INSTALLATION AND CONFIGURATION ............................................................ 10

**CHAPTER 3. CAPACITY METERING USING THE TELEMETRY SERVICE** ....................................................... 11
  3.1. VIEW EXISTING ALARMs .................................................................................................................. 11
  3.2. CREATE AN ALARM ......................................................................................................................... 11
  3.3. DISABLE OR DELETE AN ALARM .................................................................................................. 13
  3.4. VIEW MEASURES ............................................................................................................................ 13
  3.5. CREATE NEW MEASURES ............................................................................................................. 13
  3.6. VIEW CLOUD USAGE MEASURES ............................................................................................... 13
  3.7. VIEW L3 CACHE USAGE ................................................................................................................. 13
  3.8. MONITOR THE DISK ACTIVITY OF INSTANCES ........................................................................ 13
  3.9. MANAGE RESOURCE TYPES ...................................................................................................... 15
  3.10. USING THE TIME-SERIES-DATABASE-AS-A-SERVICE ................................................................. 16
  3.10.1. Running Time-Series-Database-as-a-Service ...................................................................... 16
  3.10.2. Running As A WSGI Application ......................................................................................... 16
  3.10.3. metricd Workers .................................................................................................................. 17
  3.10.4. Monitoring the Time-Series-Database-as-a-Service ............................................................... 17
  3.10.5. Backing up and Restoring Time-Series-Database-as-a-Service ........................................... 17

**CHAPTER 4. TROUBLESHOOTING** ............................................................................................................. 18
  4.1. SUPPORT ........................................................................................................................................ 18
  4.2. TROUBLESHOOT IDENTITY CLIENT (KEYSTONE) CONNECTIVITY PROBLEMS ....................... 18
  4.3. TROUBLESHOOT OPENSTACK NETWORKING ISSUES ................................................................ 19
  4.4. TROUBLESHOOT NETWORKS AND ROUTES TAB DISPLAY ISSUES IN THE DASHBOARD ........ 20
  4.5. TROUBLESHOOT INSTANCE LAUNCHING ERRORS IN THE DASHBOARD ................................. 20
  4.6. TROUBLESHOOT KEYSTONE V3 DASHBOARD AUTHENTICATION ........................................... 21
  4.7. OPENSTACK DASHBOARD - RED HAT ACCESS TAB .................................................................. 22
  4.7.1. Search ....................................................................................................................................... 24
  4.7.2. Logs ......................................................................................................................................... 25
  4.7.3. Support ..................................................................................................................................... 26
CHAPTER 1. ABOUT THIS GUIDE

WARNING

Red Hat is currently reviewing the information and procedures provided in this guide for this release.


If you require assistance for Red Hat OpenStack Platform 11, please contact Red Hat support.

This document provides an overview of the logging and monitoring capabilities that are available in a Red Hat OpenStack Platform environment, and how to troubleshoot possible issues.
CHAPTER 2. 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 4, Troubleshooting.

2.1. LOG FILES FOR OPENSTACK SERVICES

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

2.1.1. Bare Metal Provisioning (ironic) Log Files

<table>
<thead>
<tr>
<th>Service</th>
<th>Service Name</th>
<th>Log Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenStack Ironic API</td>
<td>openstack-ironic-api.service</td>
<td>/var/log/ironic/ironic-api.log</td>
</tr>
<tr>
<td>OpenStack Ironic Conductor</td>
<td>openstack-ironic-conductor.service</td>
<td>/var/log/ironic/ironic-conductor.log</td>
</tr>
</tbody>
</table>

2.1.2. Block Storage (cinder) Log Files

<table>
<thead>
<tr>
<th>Service</th>
<th>Service Name</th>
<th>Log Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block Storage API</td>
<td>openstack-cinder-api.service</td>
<td>/var/log/cinder/api.log</td>
</tr>
<tr>
<td>Block Storage Backup</td>
<td>openstack-cinder-backup.service</td>
<td>/var/log/cinder/backup.log</td>
</tr>
<tr>
<td>Informational messages</td>
<td>The cinder-manage command</td>
<td>/var/log/cinder/cinder-manage.log</td>
</tr>
<tr>
<td>Block Storage Scheduler</td>
<td>openstack-cinder-scheduler.service</td>
<td>/var/log/cinder/scheduler.log</td>
</tr>
<tr>
<td>Block Storage Volume</td>
<td>openstack-cinder-volume.service</td>
<td>/var/log/cinder/volume.log</td>
</tr>
</tbody>
</table>

2.1.3. Compute (nova) Log Files

<table>
<thead>
<tr>
<th>Service</th>
<th>Service Name</th>
<th>Log Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenStack Compute API service</td>
<td>openstack-nova-api.service</td>
<td>/var/log/nova/nova-api.log</td>
</tr>
</tbody>
</table>
### Service Log Paths

<table>
<thead>
<tr>
<th>Service</th>
<th>Service Name</th>
<th>Log Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenStack Compute certificate server</td>
<td>openstack-nova-cert.service</td>
<td>/var/log/nova/nova-cert.log</td>
</tr>
<tr>
<td>OpenStack Compute service</td>
<td>openstack-nova-compute.service</td>
<td>/var/log/nova/nova-compute.log</td>
</tr>
<tr>
<td>OpenStack Compute Conductor service</td>
<td>openstack-nova-conductor.service</td>
<td>/var/log/nova/nova-conductor.log</td>
</tr>
<tr>
<td>OpenStack Compute VNC console authentication server</td>
<td>openstack-nova-consoleauth.service</td>
<td>/var/log/nova/nova-consoleauth.log</td>
</tr>
<tr>
<td>Informational messages</td>
<td>nova-manage command</td>
<td>/var/log/nova/nova-manage.log</td>
</tr>
<tr>
<td>OpenStack Compute NoVNC Proxy service</td>
<td>openstack-nova-novncproxy.service</td>
<td>/var/log/nova/nova-novncproxy.log</td>
</tr>
<tr>
<td>OpenStack Compute Scheduler service</td>
<td>openstack-nova-scheduler.service</td>
<td>/var/log/nova/nova-scheduler.log</td>
</tr>
</tbody>
</table>

### 2.1.4. Dashboard (horizon) Log Files

<table>
<thead>
<tr>
<th>Service</th>
<th>Service Name</th>
<th>Log Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log of certain user interactions</td>
<td>Dashboard interface</td>
<td>/var/log/horizon/horizon.log</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Log Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>All processed HTTP requests</td>
<td>/var/log/httpd/horizon_access.log</td>
</tr>
<tr>
<td>HTTP errors</td>
<td>/var/log/httpd/horizon_error.log</td>
</tr>
<tr>
<td>Admin-role API requests</td>
<td>/var/log/httpd/keystone_wsgi_admin_access.log</td>
</tr>
<tr>
<td>Admin-role API errors</td>
<td>/var/log/httpd/keystone_wsgi_admin_error.log</td>
</tr>
<tr>
<td>Member-role API requests</td>
<td>/var/log/httpd/keystone_wsgi_main_access.log</td>
</tr>
<tr>
<td>Member-role API errors</td>
<td>/var/log/httpd/keystone_wsgi_main_error.log</td>
</tr>
</tbody>
</table>
NOTE
There is also `/var/log/httpd/default_error.log`, which stores errors reported by other web services running on the same host.

### 2.1.5. Data Processing (sahara) Log Files

<table>
<thead>
<tr>
<th>Service</th>
<th>Service Name</th>
<th>Log Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sahara API Server</td>
<td>openstack-sahara-all.service</td>
<td><code>/var/log/sahara/sahara-all.log</code></td>
</tr>
<tr>
<td></td>
<td>openstack-sahara-api.service</td>
<td><code>/var/log/messages</code></td>
</tr>
<tr>
<td>Sahara Engine Server</td>
<td>openstack-sahara-engine.service</td>
<td><code>/var/log/messages</code></td>
</tr>
</tbody>
</table>

### 2.1.6. Database as a Service (trove) Log Files

<table>
<thead>
<tr>
<th>Service</th>
<th>Service Name</th>
<th>Log Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenStack Trove API Service</td>
<td>openstack-trove-api.service</td>
<td><code>/var/log/trove/trove-api.log</code></td>
</tr>
<tr>
<td>OpenStack Trove Conductor Service</td>
<td>openstack-trove-conductor.service</td>
<td><code>/var/log/trove/trove-conductor.log</code></td>
</tr>
<tr>
<td>OpenStack Trove guestagent Service</td>
<td>openstack-trove-guestagent.service</td>
<td><code>/var/log/trove/logfile.txt</code></td>
</tr>
<tr>
<td>OpenStack Trove taskmanager Service</td>
<td>openstack-trove-taskmanager.service</td>
<td><code>/var/log/trove/trove-taskmanager.log</code></td>
</tr>
</tbody>
</table>

### 2.1.7. Identity Service (keystone) Log Files

<table>
<thead>
<tr>
<th>Service</th>
<th>Service Name</th>
<th>Log Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenStack Identity Service</td>
<td>openstack-keystone.service</td>
<td><code>/var/log/keystone/keystone.log</code></td>
</tr>
</tbody>
</table>

### 2.1.8. Image Service (glance) Log Files

<table>
<thead>
<tr>
<th>Service</th>
<th>Service Name</th>
<th>Log Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenStack Image Service API server</td>
<td>openstack-glance-api.service</td>
<td><code>/var/log/glance/api.log</code></td>
</tr>
<tr>
<td>OpenStack Image Service Registry server</td>
<td>openstack-glance-registry.service</td>
<td><code>/var/log/glance/registry.log</code></td>
</tr>
</tbody>
</table>
### 2.1.9. Networking (neutron) Log Files

<table>
<thead>
<tr>
<th>Service</th>
<th>Service Name</th>
<th>Log Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenStack Neutron DHCP Agent</td>
<td>neutron-dhcp-agent.service</td>
<td>/var/log/neutron/dhcp-agent.log</td>
</tr>
<tr>
<td>OpenStack Networking Layer 3 Agent</td>
<td>neutron-l3-agent.service</td>
<td>/var/log/neutron/l3-agent.log</td>
</tr>
<tr>
<td>Metadata agent service</td>
<td>neutron-metadata-agent.service</td>
<td>/var/log/neutron/metadata-agent.log</td>
</tr>
<tr>
<td>Metadata namespace proxy</td>
<td>n/a</td>
<td>/var/log/neutron/neutron-ns-metadata-proxy-UUID.log</td>
</tr>
<tr>
<td>Open vSwitch agent</td>
<td>neutron-openvswitch-agent.service</td>
<td>/var/log/neutron/openvswitch-agent.log</td>
</tr>
<tr>
<td>OpenStack Networking service</td>
<td>neutron-server.service</td>
<td>/var/log/neutron/server.log</td>
</tr>
</tbody>
</table>

### 2.1.10. 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

2.1.11. Orchestration (heat) Log Files

<table>
<thead>
<tr>
<th>Service</th>
<th>Service Name</th>
<th>Log Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenStack Heat API Service</td>
<td>openstack-heat-api.service</td>
<td>/var/log/heat/heat-api.log</td>
</tr>
<tr>
<td>Orchestration service events</td>
<td>n/a</td>
<td>/var/log/heat/heat-manage.log</td>
</tr>
</tbody>
</table>

2.1.12. Shared Filesystem Service (manila) Log Files

<table>
<thead>
<tr>
<th>Service</th>
<th>Service Name</th>
<th>Log Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenStack Manila API Server</td>
<td>openstack-manila-api.service</td>
<td>/var/log/manila/api.log</td>
</tr>
<tr>
<td>OpenStack Manila Scheduler</td>
<td>openstack-manila-scheduler.service</td>
<td>/var/log/manila/scheduler.log</td>
</tr>
<tr>
<td>OpenStack Manila Share Service</td>
<td>openstack-manila-share.service</td>
<td>/var/log/manila/share.log</td>
</tr>
</tbody>
</table>

**NOTE**

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

2.1.13. Telemetry (ceilometer) Log Files

<table>
<thead>
<tr>
<th>Service</th>
<th>Service Name</th>
<th>Log Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenStack ceilometer notification agent</td>
<td>openstack-ceilometer-notification.service</td>
<td>/var/log/ceilometer/agent-notification.log</td>
</tr>
<tr>
<td>OpenStack ceilometer alarm evaluation</td>
<td>openstack-ceilometer-alarm-evaluator.service</td>
<td>/var/log/ceilometer/alarm-evaluator.log</td>
</tr>
</tbody>
</table>
## 2.1.14. Log Files for Supporting Services

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

<table>
<thead>
<tr>
<th>Service</th>
<th>Service Name</th>
<th>Log Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenStack ceilometer alarm notification</td>
<td>openstack-ceilometer-alarm-notifier.service</td>
<td>/var/log/ceilometer/alarm-notifier.log</td>
</tr>
<tr>
<td>OpenStack ceilometer API</td>
<td>httpd.service</td>
<td>/var/log/ceilometer/api.log</td>
</tr>
<tr>
<td>Informational messages</td>
<td>MongoDB integration</td>
<td>/var/log/ceilometer/ceilometer-dbsync.log</td>
</tr>
<tr>
<td>OpenStack ceilometer central agent</td>
<td>openstack-ceilometer-central.service</td>
<td>/var/log/ceilometer/central.log</td>
</tr>
<tr>
<td>OpenStack ceilometer collection</td>
<td>openstack-ceilometer-collector.service</td>
<td>/var/log/ceilometer/collector.log</td>
</tr>
<tr>
<td>OpenStack ceilometer compute agent</td>
<td>openstack-ceilometer-compute.service</td>
<td>/var/log/ceilometer/compute.log</td>
</tr>
</tbody>
</table>

## 2.2. 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.

### 2.3. REMOTE LOGGING INSTALLATION AND CONFIGURATION

All OpenStack services generate and update log files. These log files record actions, errors, warnings, and other events. In a distributed environment like OpenStack, collecting these logs in a central location simplifies debugging and administration.

For more information about centralized logging, see the Monitoring Tools Configuration guide.
CHAPTER 3. CAPACITY METERING USING THE TELEMETRY SERVICE

The Openstack Telemetry service provides usage metrics that can be leveraged for billing, charge-back, and show-back purposes. Such metrics data can also be used by third-party applications to plan for capacity on the cluster and can also be leveraged for auto-scaling virtual instances using Openstack Heat. For more information, see Auto Scaling for Compute.

The combination of ceilometer and gnocchi can be used for monitoring and alarms. This is supported on small-size clusters and with known limitations. For real-time monitoring, Red Hat Openstack Platform ships with agents that provide metrics data, and can be consumed by separate monitoring infrastructure and applications. For more information, see Monitoring Tools Configuration.

3.1. VIEW EXISTING ALARMS

To list the existing Telemetry alarms, use the `aodh` command. For example:

```
# aodh alarm list
```

```
+--------------------------------------+----------------------------+-------------------+------+-
| alarm_id                             | type                       | state             | severity | enabled |
+--------------------------------------+-----------------------------+-------------------+------+-
| 922f899c-27c8-4c7d-a2cf-107be51ca90a | gnocchi_aggregation_by_resources_threshold | iops-monitor-read-requests | insufficient data | low      | True    |
+--------------------------------------+-----------------------------+-------------------+------+-
```

To list the meters assigned to a resource, specify the `UUID` of the resource (an instance, image, or volume, among others). For example:

```
# gnocchi resource show 5e3fcbe2-7aab-475d-b42c-a440aa42e5ad
```

3.2. CREATE AN ALARM

You can use `aodh` to create an alarm that activates when a threshold value is reached. In this example, the alarm activates and adds a log entry when the average CPU utilization for an individual instance exceeds 80%. A query is used to isolate the specific instance's id (94619081-abf5-4f1f-81c7-9cedaa872403) for monitoring purposes:

```
# aodh alarm create --type gnocchi_aggregation_by_resources_threshold --name cpu_usage_high --metric cpu_util --threshold 80 --aggregation-method sum --resource-type instance --query "{""id"": "94619081-abf5-4f1f-81c7-9cedaa872403"}" --alarm-action 'log://'
```

```
+---------------------------+-----------------------------------------+----------------------------------|
| Field                     | Value                                  |
```

11
To edit an existing threshold alarm, use the `aodh alarm update` command. For example, to increase the alarm threshold to 75%:
3.3. DISABLE OR DELETE AN ALARM

To disable an alarm:

```
# aodh alarm update --name cpu_usage_high --enabled=false
```

To delete an alarm:

```
# aodh alarm delete --name cpu_usage_high
```

3.4. VIEW MEASURES

To list all the measures for a particular resource:

```
# gnocchi measures show --resource-id UUID METER_NAME
```

To list only measures for a particular resource, within a range of timestamps:

```
# gnocchi measures show --aggregation mean --start START_TIME --end STOP_TIME --resource-id UUID METER_NAME
```

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

3.5. CREATE NEW MEASURES

You can use measures to send data to the Telemetry service, and they do not need to correspond to a previously-defined meter. For example:

```
# gnocchi measures add -m 2015-01-12T17:56:23@42 --resource-id UUID METER_NAME
```

3.6. VIEW CLOUD USAGE MEASURES

This example shows the average memory usage of all instances for each project.

```
gnocchi measures aggregation --resource-type instance --groupby project_id -m memory
```

3.7. VIEW L3 CACHE USAGE

If your Intel hardware and libvirt version supports Cache Monitoring Technology (CMT), you can use the cpu_13_cache meter to monitor the amount of L3 cache used by an instance.

3.8. MONITOR THE DISK ACTIVITY OF INSTANCES

The following example demonstrates how to use an aodh alarm to monitor the cumulative disk activity for all the instances contained within a particular project.
1. Review the existing projects, and select the appropriate UUID of the project you need to monitor. This example uses the **admin** tenant:

```
$ openstack project list
+----------------------------------+----------+
<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>745d33000ac74d30a77539f8920555e7</td>
<td>admin</td>
</tr>
<tr>
<td>983739bb834a42ddb48124a38def8538</td>
<td>services</td>
</tr>
<tr>
<td>be9e767afd4c4b7ead1417c6dfedde2b</td>
<td>demo</td>
</tr>
</tbody>
</table>
+----------------------------------+----------+
```

2. Use the project's UUID to create an alarm that analyses the **sum()** of all read requests generated by the instances in the **admin** tenant (the query can be further restrained with the **--query** parameter).

```
# aodh alarm create --type gnocchi_aggregation_by_resources_threshold --name iops-monitor-read-requests --metric disk.read.requests.rate --threshold 42000 --aggregation-method sum --resource-type instance --query '{"=": {"project_id": "745d33000ac74d30a77539f8920555e7"}}'
```

| Field                     | Value |
|---------------------------+-------|
| aggregation_method        | sum   |
| alarm_actions             | []    |
| alarm_id                  | 192aba27-d823-4ede-a404-7f6b3cc12469 |
| comparison_operator       | eq    |
| description               | gnocchi_aggregation_by_resources_threshold alarm rule |
| enabled                   | True  |
| evaluation_periods        | 1     |
| granularity               | 60    |
| insufficient_data_actions | []    |
| metric                    | disk.read.requests.rate |
| name                      | iops-monitor-read-requests |
| ok_actions                | []    |
| project_id                | 745d33000ac74d30a77539f8920555e7 |
| query                     | '{"=": {"project_id": "745d33000ac74d30a77539f8920555e7"}}' |
| repeat_actions            | False |
Telemetry resource types that were previously hardcoded can now be managed by the `gnocchi` client. You can use the gnocchi client to create, view, and delete resource types, and you can use the gnocchi API to update or delete attributes.

1. Create a new resource-type:

   ```
   $ gnocchi resource-type create testResource01 -a bla:string:True:min_length=123
   +----------------+----------------------------------------------------
   | Field          | Value                                               |
   +----------------+----------------------------------------------------
   | attributes/bla | max_length=255, min_length=123, required=True,     |
   | type=string    |                                                    |
   | name           | testResource01                                       |
   | state          | active                                              |
   +----------------+----------------------------------------------------
   ``

2. Review the configuration of the resource-type:

   ```
   $ gnocchi resource-type show testResource01
   +----------------+----------------------------------------------------
   | Field          | Value                                               |
   +----------------+----------------------------------------------------
   | name           | testResource01                                       |
   | state          | active                                              |
   +----------------+----------------------------------------------------
   ```
3. Delete the resource-type:

```bash
$ gnocchi resource-type delete testResource01
```

**NOTE**

You cannot delete a resource type if a resource is using it.

### 3.10. 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 that is 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 Manual Installation Procedures available at: https://access.redhat.com/documentation/en/red-hat-enterprise-linux-openstack-platform/

#### 3.10.1. Running Time-Series-Database-as-a-Service

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

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

#### 3.10.2. 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 \(1.5 \times \text{number of CPUs}\). If one server is not enough, you can spawn any number of new API servers to scale Gnocchi out, even on different machines.

### 3.10.3. 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.

### 3.10.4. 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.

### 3.10.5. 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 4. TROUBLESHOOTING

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

4.1. 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.

4.2. 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**

Identity Service now runs within httpd.service. On the system hosting the Identity service, check the service status:

```
# systemctl status httpd.service
```

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

```
# systemctl start httpd.service
```

**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 Manual Installation Procedures 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 = 91f0866234a64fc299db8f26f8729488
   ```

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. For example:

```
# openstack endpoint list --os-token=91f0556234a64fc299db8f26f8729488 --os-url=https://osp.lab.local:35357/v3/ --os-identity-api-version 3
```

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 an Administrator Account and the Identity Service Endpoint in the Manual Installation Procedures 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 openstack command. For example:

```
# openstack endpoint delete 2d32fa6f6eecc49a5d5e538bdf7aa018 --os-token=91f0866234a64fc299db8f26f8729488 --os-url=https://osp.lab.local:35357/v3/ --os-identity-api-version 3
```

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.

### 4.3. 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.

4.4. **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.

4.5. **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.

4.6. 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:

```python
DATABASES =
{
    'default':
    {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'horizondb',
        'USER': 'User Name',
```
2. In the same file, change SESSION_ENGINE to:

```python
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).

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

4. Create the Horizon database.

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

5. Exit the mysql client.

```bash
mysql > exit
```

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

```bash
# 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:

```bash
#service httpd restart
```

### 4.7. OPENSTACK DASHBOARD - RED HAT ACCESS TAB

The *Red Hat Access* tab, which is part of the OpenStack dashboard, allows you to search for and read articles or solutions from the Red Hat Customer Portal, view logs from your instances and diagnose them, and work with your customer support cases.
Figure 4.1. Red Hat Access Tab.

Important

You must be logged in to the Red Hat Customer Portal in the browser in order to be able to use the functions provided by the Red Hat Access tab.

If you are not logged in, you can do so now:

1. Click Log In.
2. Enter your Red Hat login.
3. Enter your Red Hat password.
4. Click Sign in.

This is how the form looks:
Figure 4.2. Logging in to the Red Hat Customer Portal.

**Sign into the Red Hat Customer Portal**

Red Hat Access makes it easy for you to self-solve issues, diagnose problems, and engage with us via the Red Hat Customer Portal. To access Red Hat Customer Portal resources, you must enter valid portal credentials.

**Red Hat Login**

Red Hat Login

**Password**

Password

**Note:** Red Hat Customer Portal credentials differ from the credentials used to log into this product.

If you do not log in now, you will be prompted for your Red Hat login and password when you use one of the functions that require authentication.

**4.7.1. Search**

You can search for articles and solutions from Red Hat Customer Portal by entering one or more search keywords. The titles of the relevant articles and solutions will then be displayed. Click on a title to view the given article or solution:
4.7.2. Logs

Here you can read logs from your OpenStack instances:

Find the instance of your choice in the table. If you have many instances, you can filter them by name, status, image ID, or flavor ID. Click View Log in the Actions column for the instance to check.
When an instance log is displayed, you can click *Red Hat Diagnose* to get recommendations regarding its contents:

**Figure 4.5. Instance Logs on the Red Hat Access Tab.**

If none of the recommendations are useful or a genuine problem has been logged, click *Open a New Support Case* to report the problem to Red Hat Support.

### 4.7.3. Support

The last option in the Red Hat Access Tab allows you to search for your support cases at the Red Hat Customer Portal:

**Figure 4.6. Search for Support Cases.**

You can also open a new support case by clicking the appropriate button and filling out the form on the following page:
Figure 4.7. Open a New Support Case.