



Red Hat OpenStack Platform 12

Logging, Monitoring, and Troubleshooting Guide

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

Red Hat OpenStack Platform 12 Logging, Monitoring, and Troubleshooting Guide

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

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Copyright © 2018 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution-Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

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	4
CHAPTER 2. LOGGING	5
2.1. LOG FILES FOR OPENSTACK SERVICES	5
2.1.1. Bare Metal Provisioning (ironic) Log Files	5
2.1.2. Block Storage (cinder) Log Files	5
2.1.3. Compute (nova) Log Files	5
2.1.4. Dashboard (horizon) Log Files	6
2.1.5. Data Processing (sahara) Log Files	7
2.1.6. Database as a Service (trove) Log Files	7
2.1.7. Identity Service (keystone) Log Files	7
2.1.8. Image Service (glance) Log Files	8
2.1.9. Networking (neutron) Log Files	8
2.1.10. Object Storage (swift) Log Files	8
2.1.11. Orchestration (heat) Log Files	9
2.1.12. Shared Filesystem Service (manila) Log Files	9
2.1.13. Telemetry (ceilometer) Log Files	10
2.1.14. Log Files for Supporting Services	10
2.2. CONFIGURE LOGGING OPTIONS	11
2.3. REMOTE LOGGING INSTALLATION AND CONFIGURATION	11
CHAPTER 3. CONFIGURING THE TIME SERIES DATABASE (GNOCCHI) FOR TELEMETRY	12
3.1. UNDERSTANDING THE TIME SERIES DATABASE	12
3.2. METRICS	12
3.3. TIME SERIES DATABASE COMPONENTS	13
3.4. RUNNING THE TIME SERIES DATABASE	14
3.5. RUNNING AS A WSGI APPLICATION	14
3.6. METRICD WORKERS	14
3.7. MONITORING THE TIME SERIES DATABASE	14
3.8. BACKING UP AND RESTORING THE TIME SERIES DATABASE	14
CHAPTER 4. CAPACITY METERING USING THE TELEMETRY SERVICE	15
4.1. VIEW MEASURES	15
4.2. CREATE NEW MEASURES	15
4.3. EXAMPLE: VIEW CLOUD USAGE MEASURES	15
4.4. EXAMPLE: VIEW L3 CACHE USAGE	15
4.5. VIEW EXISTING ALARMS	17
4.6. CREATE AN ALARM	17
4.7. DISABLE OR DELETE AN ALARM	19
4.8. EXAMPLE: MONITOR THE DISK ACTIVITY OF INSTANCES	19
4.9. EXAMPLE: MONITOR CPU USAGE	20
4.10. MANAGE RESOURCE TYPES	26
CHAPTER 5. TROUBLESHOOTING	28
5.1. SUPPORT	28
5.2. TROUBLESHOOT IDENTITY CLIENT (KEYSTONE) CONNECTIVITY PROBLEMS	28
5.3. TROUBLESHOOT OPENSTACK NETWORKING ISSUES	29
5.4. TROUBLESHOOT NETWORKS AND ROUTES TAB DISPLAY ISSUES IN THE DASHBOARD	30
5.5. TROUBLESHOOT INSTANCE LAUNCHING ERRORS IN THE DASHBOARD	30
5.6. TROUBLESHOOT KEYSTONE V3 DASHBOARD AUTHENTICATION	31
5.6.1. Search	34
5.6.2. Logs	35

CHAPTER 1. ABOUT THIS GUIDE



WARNING

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

This document is based on the Red Hat OpenStack Platform 11 document, available at https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/?version=11.

If you require assistance for the current Red Hat OpenStack Platform release, 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 5, Troubleshooting](#).



NOTE

Since the release of Red Hat OpenStack Platform 12, most of the services are containerized. The only exceptions are neutron and cinder. The new log path for these services is `/var/log/containers`.

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

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

2.1.2. 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
Block Storage Volume	openstack-cinder-volume.service	/var/log/cinder/volume.log

2.1.3. Compute (nova) Log Files

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

2.1.4. Dashboard (horizon) Log Files

Service	Service Name	Log Path
Log of certain user interactions	Dashboard interface	/var/log/containers/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/containers/httpd/horizon_access.log
HTTP errors	/var/log/containers/httpd/horizon_error.log
Admin-role API requests	/var/log/containers/httpd/keystone_wsgi_admin_access.log

Purpose	Log Path
Admin-role API errors	<code>/var/log/containers/httpd/keystone_wsgi_admin_error.log</code>
Member-role API requests	<code>/var/log/containers/httpd/keystone_wsgi_main_access.log</code>
Member-role API errors	<code>/var/log/containers/httpd/keystone_wsgi_main_error.log</code>

**NOTE**

There is also `/var/log/containers/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

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

2.1.6. Database as a Service (trove) Log Files

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

2.1.7. Identity Service (keystone) Log Files

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

2.1.8. Image Service (glance) Log Files

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

2.1.9. Networking (neutron) Log Files

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

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/containers/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 rhel-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 rhel-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 rhel-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 rhel-a24c-02 account-replicator: Beginning replication run
Apr 27 17:08:16 rhel-a24c-02 account-replicator: Replication run OVER
Apr 27 17:08:16 rhel-a24c-02 account-replicator: Attempted to replicate 5
dbs in 0.12589 seconds (39.71876/s)
Apr 27 17:08:16 rhel-a24c-02 account-replicator: Removed 0 dbs
Apr 27 17:08:16 rhel-a24c-02 account-replicator: 10 successes, 0 failures
```

2.1.11. Orchestration (heat) Log Files

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

2.1.12. Shared Filesystem Service (manila) Log Files

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

Service	Service Name	Log Path
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`.

2.1.13. Telemetry (ceilometer) Log Files

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

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.

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

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
```

- 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. CONFIGURING THE TIME SERIES DATABASE (GNOCCHI) FOR TELEMETRY

Time series database (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.

3.1. UNDERSTANDING THE TIME SERIES DATABASE

This section defines the commonly used terms for the Time series database (Gnocchi) features.

Aggregation method

A function used to aggregate multiple measures into an aggregate. For example, the `min` aggregation method aggregates the values of different measures to the minimum value of all the measures in the time range.

Aggregate

A data point tuple generated from several measures according to the archive policy. An aggregate is composed of a time stamp and a value.

Archive policy

An aggregate storage policy attached to a metric. An archive policy determines how long aggregates are kept in a metric and how aggregates are aggregated (the aggregation method).

Granularity

The time between two aggregates in an aggregated time series of a metric.

Measure

An incoming data point tuple sent to the Time series database by the API. A measure is composed of a time stamp and a value.

Metric

An entity storing aggregates identified by an UUID. A metric can be attached to a resource using a name. How a metric stores its aggregates is defined by the archive policy that the metric is associated to.

Resource

An entity representing anything in your infrastructure that you associate a metric with. A resource is identified by a unique ID and can contain attributes.

Time series

A list of aggregates ordered by time.

Timespan

The time period for which a metric keeps its aggregates. It is used in the context of archive policy.

3.2. METRICS

The Time series database (Gnocchi) stores *metrics* from Telemetry that designate anything that can be measured, for example, the CPU usage of a server, the temperature of a room or the number of bytes sent by a network interface.

A metric has the following properties:

- UUID to identify the metric

- Metric name
- Archive policy used to store and aggregate the measures

The Time series database stores the following metrics by default, as defined in the `etc/ceilometer/polling.yaml` file:

```
[root@controller-0 ~]# docker exec -ti ceilometer_agent_central cat
/etc/ceilometer/polling.yaml
---
sources:
  - name: some_pollsters
    interval: 300
    meters:
      - cpu
      - memory.usage
      - network.incoming.bytes
      - network.incoming.packets
      - network.outgoing.bytes
      - network.outgoing.packets
      - disk.read.bytes
      - disk.read.requests
      - disk.write.bytes
      - disk.write.requests
      - hardware.cpu.util
      - hardware.memory.used
      - hardware.memory.total
      - hardware.memory.buffer
      - hardware.memory.cached
      - hardware.memory.swap.avail
      - hardware.memory.swap.total
      - hardware.system_stats.io.outgoing.blocks
      - hardware.system_stats.io.incoming.blocks
      - hardware.network.ip.incoming.datagrams
      - hardware.network.ip.outgoing.datagrams
```

The `polling.yaml` file also specifies the default polling interval of 300 seconds (5 minutes).

3.3. TIME SERIES DATABASE COMPONENTS

Currently, Gnocchi uses the Identity service for authentication and Redis for incoming measure storage. To store the aggregated measures, Gnocchi relies on either Swift or Ceph (Object Storage). Gnocchi also leverages MySQL to store the index of resources and metrics.

The Time series database provides the `statsd` daemon (`gnocchi-statsd`) that is compatible with the `statsd` protocol and can listen to the metrics sent over the network. In order to enable `statsd` support in Gnocchi, 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 are created dynamically as the metrics are sent to `gnocchi-statsd`, and attached with the provided name to the resource ID you configured.

3.4. RUNNING THE TIME SERIES DATABASE

Run the Time series database by running the HTTP server and metric daemon:

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

3.5. RUNNING AS A WSGI APPLICATION

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

The Gnocchi 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.

3.6. METRICD WORKERS

By default, the `gnocchi-metricd` daemon spans all your CPU power in order to maximize CPU utilization 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 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.

For director-based deployments, you can adjust certain metric processing parameters in your environment file:

- `MetricProcessingDelay` - Adjusts the delay period between iterations of metric processing.
- `GnocchiMetricdWorkers` - Configure the number of `metricd` workers.

3.7. MONITORING THE TIME SERIES DATABASE

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.

3.8. BACKING UP AND RESTORING THE TIME SERIES DATABASE

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, re-install Gnocchi if necessary, and restart it.

CHAPTER 4. 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 Instances](#).

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](#).

4.1. 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*.

4.2. 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
```

4.3. EXAMPLE: 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
```

4.4. EXAMPLE: VIEW L3 CACHE USAGE

If your Intel hardware and libvirt version supports *Cache Monitoring Technology* (CMT), you can use the `cpu_l3_cache` meter to monitor the amount of L3 cache used by an instance.

Monitoring the L3 cache requires the following:

- `cmt` in the `LibvirtEnabledPerfEvents` parameter.
- `cpu_l3_cache` in the `gnocchi_resources.yaml` file.

- `cpu_l3_cache` in the Ceilometer `polling.yaml` file.

Enable L3 Cache Monitoring

To enable L3 cache monitoring:

1. Create a YAML file for telemetry (for example, `ceilometer-environment.yaml`) and add `cmt` to the `LibvirtEnabledPerfEvents` parameter.

```
parameter_defaults:
  LibvirtEnabledPerfEvents: cmt
```

2. Launch the overcloud with this YAML file.

```
#!/bin/bash

openstack overcloud deploy \
  --templates \
  <additional templates> \
  -e /home/stack/ceilometer-environment.yaml
```

3. Verify that `cpu_l3_cache` is enabled in `gnocchi` on the Compute node.

```
$ sudo -i
# docker exec -ti ceilometer_agent_compute cat
/etc/ceilometer/gnocchi_resources.yaml | grep cpu_l3_cache
```

4. Verify that `cpu_l3_cache` is enabled for Telemetry polling.

```
# docker exec -ti ceilometer_agent_compute cat
/etc/ceilometer/polling.yaml | grep cpu_l3_cache
```

5. If `cpu_l3_cache` is not enabled for Telemetry, enable it and restart the service.

```
# docker exec -ti ceilometer_agent_compute echo "
cpu_l3_cache" >> /etc/ceilometer/polling.yaml

# docker exec -ti ceilometer_agent_compute pkill -HUP -f
"ceilometer.*master process"
```



NOTE

This docker change will not persist over a reboot.

After you have launched a guest instance on this compute node, you can use the `gnocchi measures show` command to monitor the CMT metrics.

```
(overcloud) [stack@undercloud-0 ~]$ gnocchi measures show --resource-id
a6491d92-b2c8-4f6d-94ba-edc9dfde23ac cpu_l3_cache
+-----+-----+-----+
| timestamp | granularity | value |
+-----+-----+-----+
| 2017-10-25T09:40:00+00:00 | 300.0 | 1966080.0 |
```

```

| 2017-10-25T09:45:00+00:00 |          300.0 | 1933312.0 |
| 2017-10-25T09:50:00+00:00 |          300.0 | 2129920.0 |
| 2017-10-25T09:55:00+00:00 |          300.0 | 1966080.0 |
| 2017-10-25T10:00:00+00:00 |          300.0 | 1933312.0 |
| 2017-10-25T10:05:00+00:00 |          300.0 | 2195456.0 |
| 2017-10-25T10:10:00+00:00 |          300.0 | 1933312.0 |
| 2017-10-25T10:15:00+00:00 |          300.0 | 1998848.0 |
| 2017-10-25T10:20:00+00:00 |          300.0 | 2097152.0 |
| 2017-10-25T10:25:00+00:00 |          300.0 | 1933312.0 |
| 2017-10-25T10:30:00+00:00 |          300.0 | 1966080.0 |
| 2017-10-25T10:35:00+00:00 |          300.0 | 1933312.0 |
| 2017-10-25T10:40:00+00:00 |          300.0 | 1933312.0 |
| 2017-10-25T10:45:00+00:00 |          300.0 | 1933312.0 |
| 2017-10-25T10:50:00+00:00 |          300.0 | 2850816.0 |
| 2017-10-25T10:55:00+00:00 |          300.0 | 2359296.0 |
| 2017-10-25T11:00:00+00:00 |          300.0 | 2293760.0 |
+-----+-----+-----+

```

4.5. VIEW EXISTING ALARMS

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

```

# aodh alarm list
+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+
----+-----+
| alarm_id          | type          |
| name              | 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
```

4.6. 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 |
+-----+-----+
| aggregation_method | sum |
| alarm_actions | [u'log:/' |
| alarm_id | b794adc7-ed4f-4edb-ace4-88cbe4674a94 |
| comparison_operator | eq |
| description | gnocchi_aggregation_by_resources_threshold |
alarm rule |
| enabled | True |
| evaluation_periods | 1 |
| granularity | 60 |
| insufficient_data_actions | [] |
| metric | cpu_util |
| name | cpu_usage_high |
| ok_actions | [] |
| project_id | 13c52c41e0e543d9841a3e761f981c20 |
| query | {"=": {"id": "94619081-abf5-4f1f-81c7- |
9cedaa872403"}} |
| repeat_actions | False |
| resource_type | instance |
| severity | low |
| state | insufficient data |
| state_timestamp | 2016-12-09T05:18:53.326000 |
| threshold | 80.0 |
| time_constraints | [] |
| timestamp | 2016-12-09T05:18:53.326000 |
| type | gnocchi_aggregation_by_resources_threshold |
| user_id | 32d3f2c9a234423cb52fb69d3741dbbc |

```

```
|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+
```

To edit an existing threshold alarm, use the `aodh alarm update` command. For example, to increase the alarm threshold to 75%:

```
# aodh alarm update --name cpu_usage_high --threshold 75
```

4.7. 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
```

4.8. EXAMPLE: 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
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ID                                     | Name      |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 745d33000ac74d30a77539f8920555e7     | admin     |
| 983739bb834a42ddb48124a38def8538     | services  |
| be9e767afd4c4b7ead1417c6dfedde2b    | demo      |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

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
| resource_type | instance
| severity | low
| state | insufficient data
| state_timestamp | 2016-11-08T23:41:22.919000
| threshold | 42000.0
| time_constraints | []
| timestamp | 2016-11-08T23:41:22.919000
| type | gnocchi_aggregation_by_resources_threshold
| user_id | 8c4aea738d774967b4ef388eb41fef5e
+-----+-----+
-----+

```

4.9. EXAMPLE: MONITOR CPU USAGE

If you want to monitor an instance's performance, you would start by examining the gnocchi database to identify which metrics you can monitor, such as memory or CPU usage. For example, run `gnocchi resource show` against an instance to identify which metrics can be monitored:

1. Query the available metrics for a particular instance UUID:


```

$ gnocchi resource show --type instance d71cdf9a-51dc-4bba-8170-
9cd95edd3f66
+-----+-----+
| Field                | Value
|
+-----+-----+
| created_by_project_id | 44adccdc32614688ae765ed4e484f389
|
| created_by_user_id    | c24fa60e46d14f8d847fca90531b43db
|
| creator               |
c24fa60e46d14f8d847fca90531b43db:44adccdc32614688ae765ed4e484f389
|
| display_name          | test-instance
|
| ended_at              | None
|
| flavor_id             | 14c7c918-df24-481c-b498-0d3ec57d2e51
|
| flavor_name           | m1.tiny
|
| host                  | overcloud-compute-0
|
| id                    | d71cdf9a-51dc-4bba-8170-9cd95edd3f66
|
| image_ref             | e75dff7b-3408-45c2-9a02-61fbfbf054d7
|
| metrics               | compute.instance.booting.time: c739a70d-
2d1e-45c1-8c1b-4d28ff2403ac |
|                        | cpu.delta: 700ceb7c-4cff-4d92-be2f-
6526321548d6 |
|                        | cpu: 716d6128-1ea6-430d-aa9c-ceaff2a6bf32
|
|                        | cpu_l3_cache: 3410955e-c724-48a5-ab77-
c3050b8cbe6e |
|                        | cpu_util: b148c392-37d6-4c8f-8609-
e15fc15a4728 |
|                        | disk.allocation: 9dd464a3-acf8-40fe-bd7e-
3cb5fb12d7cc |
|                        | disk.capacity: c183d0da-e5eb-4223-a42e-
855675dd1ec6 |
|                        | disk.ephemeral.size: 15d1d828-fbb4-4448-
b0f2-2392dcfed5b6 |
|                        | disk.iops: b8009e70-dae-403f-94ed-
73853359a087 |
|                        | disk.latency: 1c648176-18a6-4198-ac7f-
33ee628b82a9 |
|                        | disk.read.bytes.rate: eb35828f-312f-41ce-
b0bc-cb6505e14ab7 |
|                        | disk.read.bytes: de463be7-769b-433d-9f22-
f3265e146ec8 |
|                        | disk.read.requests.rate: 588ca440-bd73-
4fa9-a00c-8af67262f4fd |
|                        | disk.read.requests: 53e5d599-6cad-47de-

```

```

b814-5cb23e8aaf24      |
|                       | disk.root.size: cee9d8b1-181e-4974-9427-
aa7adb3b96d9          |
|                       | disk.usage: 4d724c99-7947-4c6d-9816-
abbbc166f6f3          |
|                       | disk.write.bytes.rate: 45b8da6e-0c89-
4a6c-9cce-c95d49d9cc8b |
|                       | disk.write.bytes: c7734f1b-b43a-48ee-
8fe4-8a31b641b565     |
|                       | disk.write.requests.rate: 96ba2f22-8dd6-
4b89-b313-1e0882c4d0d6 |
|                       | disk.write.requests: 553b7254-be2d-481b-
9d31-b04c93dbb168     |
|                       | memory.bandwidth.local: 187f29d4-7c70-
4ae2-86d1-191d11490aad |
|                       | memory.bandwidth.total: eb09a4fc-c202-
4bc3-8c94-aa2076df7e39 |
|                       | memory.resident: 97cfb849-2316-45a6-9545-
21b1d48b0052          |
|                       | memory.swap.in: f0378d8f-6927-4b76-8d34-
a5931799a301          |
|                       | memory.swap.out: c5fba193-1a1b-44c8-82e3-
9fdc9ef21f69          |
|                       | memory.usage: 7958d06d-7894-4ca1-8c7e-
72ba572c1260          |
|                       | memory: a35c7eab-f714-4582-aa6f-
48c92d4b79cd          |
|                       | perf.cache.misses: da69636d-d210-4b7b-
bea5-18d4959e95c1     |
|                       | perf.cache.references: e1955a37-d7e4-
4b12-8a2a-51de4ec59efd |
|                       | perf.cpu.cycles: 5d325d44-b297-407a-b7db-
cc9105549193          |
|                       | perf.instructions: 973d6c6b-bbeb-4a13-
96c2-390a63596bfc     |
|                       | vcpus: 646b53d0-0168-4851-b297-
05d96cc03ab2          |
| original_resource_id | d71cdf9a-51dc-4bba-8170-9cd95edd3f66
| project_id           | 3cee262b907b4040b26b678d7180566b
| revision_end         | None
| revision_start       | 2017-11-16T04:00:27.081865+00:00
| server_group         | None
| started_at           | 2017-11-16T01:09:20.668344+00:00
| type                 | instance
| user_id              | 1dbf5787b2ee46cf9fa6a1dfea9c9996
+-----+-----+
-----+

```

In this result, the `metrics` value lists the components you can monitor using Aodh alarms, for example `cpu_util`.

- To monitor CPU usage, you will need the `cpu_util` metric. To see more information on this metric:

```
$ gnocchi metric show --resource d71cdf9a-51dc-4bba-8170-9cd95edd3f66 cpu_util
+-----+-----+
+-----+-----+
| Field                                     | Value
+-----+-----+
+-----+-----+
| archive_policy/aggregation_methods      | std, count, min, max, sum,
mean                                     |
| archive_policy/back_window              | 0
|
| archive_policy/definition                | - points: 8640, granularity:
0:05:00, timespan: 30 days, 0:00:00    |
| archive_policy/name                     | low
|
| created_by_project_id                   |
44adccdc32614688ae765ed4e484f389
|
| created_by_user_id                       |
c24fa60e46d14f8d847fca90531b43db
|
| creator                                  |
c24fa60e46d14f8d847fca90531b43db:44adccdc32614688ae765ed4e484f389 |
| id                                       | b148c392-37d6-4c8f-8609-
e15fc15a4728                             |
| name                                     | cpu_util
|
| resource/created_by_project_id           |
44adccdc32614688ae765ed4e484f389
|
| resource/created_by_user_id             |
c24fa60e46d14f8d847fca90531b43db
|
| resource/creator                         |
c24fa60e46d14f8d847fca90531b43db:44adccdc32614688ae765ed4e484f389 |
| resource/ended_at                       | None
|
| resource/id                              | d71cdf9a-51dc-4bba-8170-
9cd95edd3f66                             |
| resource/original_resource_id            | d71cdf9a-51dc-4bba-8170-
9cd95edd3f66                             |
| resource/project_id                     |
3cee262b907b4040b26b678d7180566b
|
| resource/revision_end                   | None
|
| resource/revision_start                  | 2017-11-
17T00:05:27.516421+00:00
| resource/started_at                     | 2017-11-
```

```

16T01:09:20.668344+00:00 |
| resource/type | instance |
| resource/user_id | |
1dbf5787b2ee46cf9fa6a1dfea9c9996 |
| unit | None |
+-----+-----+
-----+

```

- **archive_policy** - Defines the aggregation interval for calculating the **std**, **count**, **min**, **max**, **sum**, **mean** values.

3. Use Aodh to create a monitoring task that queries `cpu_util`. This task will trigger events based on the settings you specify. For example, to raise a log entry when an instance's CPU spikes over 80% for an extended duration:

```

aodh alarm create \
  --project-id 3cee262b907b4040b26b678d7180566b \
  --name high-cpu \
  --type gnocchi_resources_threshold \
  --description 'High CPU usage' \
  --metric cpu_util \
  --threshold 80.0 \
  --comparison-operator ge \
  --aggregation-method mean \
  --granularity 300 \
  --evaluation-periods 1 \
  --alarm-action 'log://' \
  --ok-action 'log://' \
  --resource-type instance \
  --resource-id d71cdf9a-51dc-4bba-8170-9cd95edd3f66
+-----+-----+
--+
| Field | Value |
| | |
+-----+-----+
--+
| aggregation_method | mean |
| | |
| alarm_actions | [u'log://'] |
| | |
| alarm_id | 1625015c-49b8-4e3f-9427-3c312a8615dd |
| | |
| comparison_operator | ge |
| | |
| description | High CPU usage |
| | |
| enabled | True |
| | |
| evaluation_periods | 1 |
| | |
| granularity | 300 |
| | |
| insufficient_data_actions | [] |

```

```

| metric                | cpu_util
| name                  | high-cpu
| ok_actions            | [u'log://']
| project_id            | 3cee262b907b4040b26b678d7180566b
| repeat_actions        | False
| resource_id           | d71cdf9a-51dc-4bba-8170-9cd95edd3f66
| resource_type         | instance
| severity              | low
| state                 | insufficient data
| state_reason          | Not evaluated yet
| state_timestamp       | 2017-11-16T05:20:48.891365
| threshold              | 80.0
| time_constraints      | []
| timestamp             | 2017-11-16T05:20:48.891365
| type                  | gnocchi_resources_threshold
| user_id               | 1dbf5787b2ee46cf9fa6a1dfea9c9996
+-----+-----+
--+
```

- **comparison-operator** - The `ge` operator defines that the alarm will trigger if the CPU usage is greater than (or equal to) 80%.
- **granularity** - Metrics have an archive policy associated with them; the policy can have various granularities (for example, 5 minutes aggregation for 1 hour + 1 hour aggregation over a month). The **granularity** value must match the duration described in the archive policy.
- **evaluation-periods** - Number of **granularity** periods that need to pass before the alarm will trigger. For example, setting this value to 2 will mean that the CPU usage will need to be over 80% for two polling periods before the alarm will trigger.
- `[u'log://']` - This value will log events to your Aodh log file.



NOTE

You can define different actions to run when an alarm is triggered (**alarm_actions**), and when it returns to a normal state (**ok_actions**), such as a webhook URL.

4. To check if your alarm has been triggered, query the alarm's history:

```
aodh alarm-history show 1625015c-49b8-4e3f-9427-3c312a8615dd --fit-
width
+-----+-----+-----+
-----
-----
-----
---+-----+
| timestamp          | type          | detail
| event_id          |               |
+-----+-----+-----+
-----
-----
-----
---+-----+
| 2017-11-16T05:21:47.850094 | state transition |
{"transition_reason": "Transition to ok due to 1 samples inside
threshold, most recent: 0.0366665763", "state": "ok"}
| 3b51f09d-ded1-4807-b6bb-65fdc87669e4 |
+-----+-----+-----+
-----
-----
-----
---+-----+
```

4.10. MANAGE RESOURCE TYPES

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
|
```

```

+-----+-----+
-----+
| attributes/bla | max_length=255, min_length=123, required=True,
type=string |
| name          | testResource01
|
| state         | active
|
+-----+-----+
-----+

```

3. Delete the *resource-type*:

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



NOTE

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

CHAPTER 5. TROUBLESHOOTING

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

5.1. 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.

5.2. 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 *Managing the OpenStack Firewall* in the *Advanced OpenStack Customization* guide for instructions on checking your firewall settings and defining custom rules.

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, add the correct endpoint and remove any incorrect endpoints using the **endpoint delete** action of the **openstack** command. For example:

```
# openstack endpoint delete 2d32fa6feecc49aab5de538bdf7aa018 --
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.

5.3. 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.

5.4. 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.

5.5. 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.

5.6. 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 horizonsdb;
```

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.

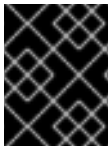
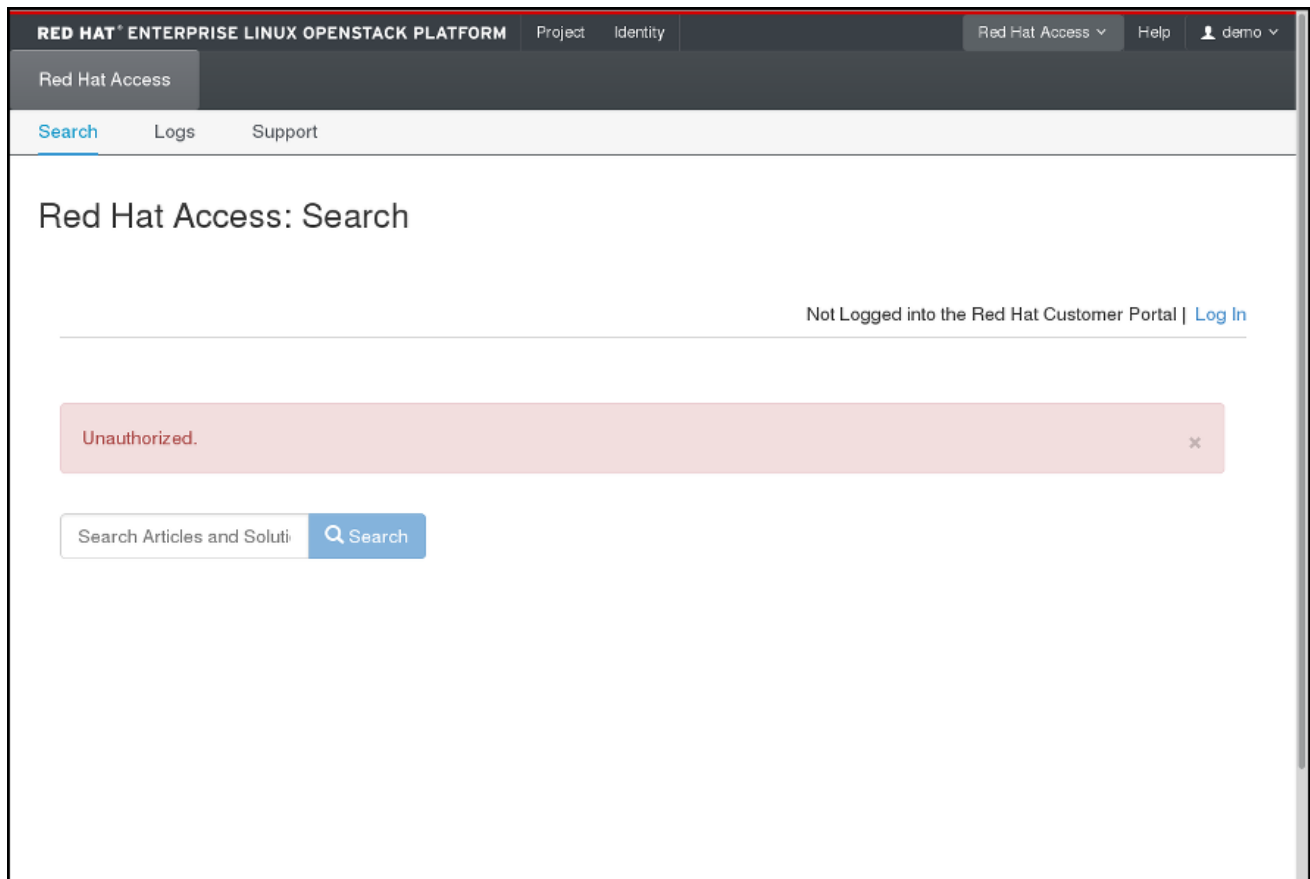
7. Restart Apache http server. For Red Hat Enterprise Linux:

```
#service httpd restart
```

=== OpenStack Dashboard - Red Hat Access Tab

The *Red Hat Accesstab*, 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 5.1. Red Hat Access Tab.



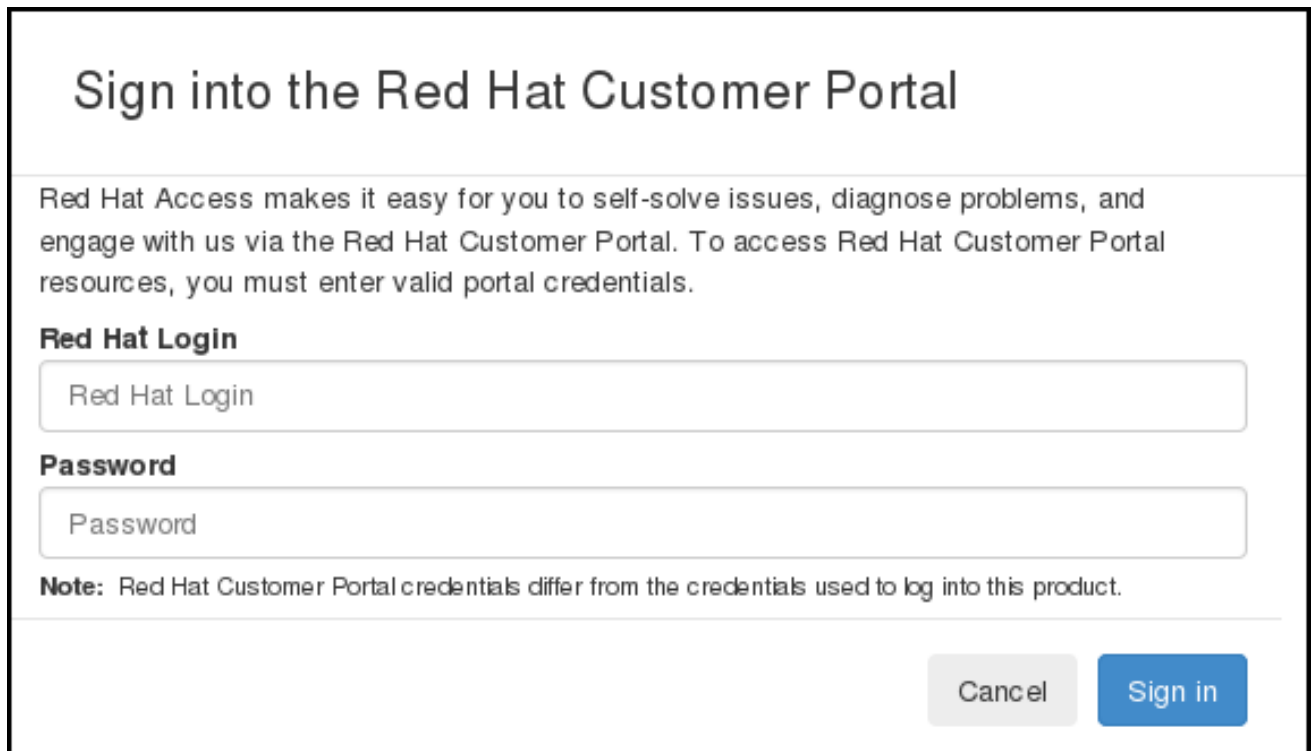
IMPORTANT

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 5.2. Logging in to the Red Hat Customer Portal.

The screenshot shows a login form titled "Sign into the Red Hat Customer Portal". Below the title is a paragraph of text: "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." Below this text are two input fields: "Red Hat Login" and "Password". Below the input fields is a note: "Note: Red Hat Customer Portal credentials differ from the credentials used to log into this product." At the bottom right of the form are two buttons: "Cancel" and "Sign in".

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

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.

5.6.1. 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:

Figure 5.3. Example of Search Results on the Red Hat Access Tab.

The screenshot shows the Red Hat Access Search interface. The search term 'POODLE' is entered in the search bar. The results are categorized into Recommendations, Environment, and Issue.

Recommendations:

- Poodle TLS vulnerability CVE-2014-8730
- EAP 6.2.1 JBossWeb native and POODLE
- Disabling SSLv3 For POODLE vulnerability produces errors
- Resolution for POODLE SSLv3.0 vulnerability (CVE-2014-3566) in

Environment:

- Red Hat Enterprise Linux (RHEL) 7
- Red Hat Enterprise Linux (RHEL) 6
- Red Hat Enterprise Linux (RHEL) 5
- Red Hat Enterprise Linux (RHEL) 4

Issue:

Recent media publications are publishing articles indicating that in some cases, TLS is now also impacted by the POODLE flaw and has been tracked by Red Hat as [CVE-2014-8730](#) at [Bugzilla-CVE-2014-8730 TLS: incorrect check of padding bytes when using CBC cipher suites](#).

5.6.2. Logs

Here you can read logs from your OpenStack instances:

Figure 5.4. Instance Logs on the Red Hat Access Tab.

The screenshot shows the Red Hat Access Logs interface. The 'Logs' tab is selected. Below the header, there is a table of instances with columns for Instance Name, Image Name, IP Address, Size, Key Pair, Status, Availability Zone, Task, Power State, Time since created, and Actions.

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
testinstance2	cirros	192.168.0.7	m1.small	OS-Key	Error	nova	None	No State	1 week, 6 days	View Log
testinstance	cirros	192.168.0.2	m1.tiny	-	Shutoff	nova	None	Shut Down	3 weeks, 2 days	View Log

Displaying 2 items

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 5.5. Instance Logs on the Red Hat Access Tab.

The screenshot shows the Red Hat Access interface. At the top, there's a navigation bar with 'Red Hat Access', 'Project', and 'Identity'. Below that, there's a search bar and tabs for 'Search', 'Logs', and 'Support'. The main content area is titled 'Instance Log' and displays a list of failed requests with timestamps and error messages. To the right of the logs, there's a 'Log Out' link and a 'Red Hat Diagnose' button. Below the logs, there's a 'Recommendations' section with a list of suggested actions, including 'taskomatic does not start, times out waiting on the JVM 5 times before stopping on Red Hat Satellite', 'System panic with message : VXFEN WARNING V-11-1-20 Could not eject node 0 from disk', and 'Getting error: SCSI error: return code = 0x00010000'. At the bottom right, there's an 'Open a New Support Case' button.

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.

5.6.3. 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 5.6. Search Keys for Support Cases.

The screenshot shows the Red Hat Access interface with the 'Support' tab selected. The main content area is titled 'Red Hat Access: Support'. Below the title, there's a search bar with a 'Search' button, a dropdown menu for 'All Groups', and a dropdown menu for 'Open'. At the bottom left, there's a message that says 'No cases found with given filters.' At the bottom right, there's an 'Open a New Support Case' button.

You can also open a new support case by clicking the appropriate button and filling out the form on the following page:

Figure 5.7. Open a New Support Case.

The screenshot displays the Red Hat Access Support portal interface. At the top, the navigation bar includes "RED HAT ENTERPRISE LINUX OPENSTACK PLATFORM", "Project", "Identity", "Red Hat Access", "Help", and "demo". Below the navigation bar, there are links for "Search", "Logs", and "Support". The main heading is "Red Hat Access: Support". On the right side, it indicates the user is logged in as "demo" with a "Log Out" link.

The form is divided into two main sections:

- Form Fields:**
 - Account:** A text input field with a "My Account" button next to it.
 - Owner:** A dropdown menu showing "No match found".
 - Product:** A dropdown menu showing "Red Hat OpenStack".
 - Product Version:** A dropdown menu showing "6.0".
 - Summary:** A text input field.
 - Description:** A large text area for entering details.
 - Next:** A blue button at the bottom right of the form.
- Recommendations:** A section on the right with three suggested articles:
 - ▶ The Production Support Scope of Coverage and Production Support Service Level Agreement
 - ▶ What Is The Red Hat Satellite 6 Managed Design Program (MDP) and will there be a Beta?
 - ▶ Error message from subscription-manager when attempting to auto-attach shows No installed products on system. No need to attach subscriptions.