



Red Hat OpenStack Platform

8

ロギング、モニタリング、トラブル シューティングガイド

OpenStack のロギング、モニタリング、トラブルシューティングの詳細ガイド

OpenStack Team

Red Hat OpenStack Platform 8 ロギング、モニタリング、トラブルシューティングガイド

OpenStack のロギング、モニタリング、トラブルシューティングの詳細ガイド

OpenStack Team
rhos-docs@redhat.com

法律上の通知

Copyright © 2017 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.

概要

本ガイドは、Red Hat OpenStack Platform 環境のロギングおよびモニタリング、さらには問題の解決方法について詳しく説明します。

目次

前書き	3
第1章 ログイン	4
1.1. OPENSTACK サービスのログファイル	4
1.2. ログインオプションの設定	11
1.3. リモートログインのインストールおよび設定	12
第2章 TELEMETRY サービスを使用したモニタリング	15
2.1. 既存のアラームの表示	15
2.2. アラームの設定	15
2.3. アラームの無効化または削除	16
2.4. サンプルの表示	16
2.5. サンプルの作成	17
2.6. クラウドの使用状況の統計の表示	18
2.7. TIME-SERIES-DATABASE-AS-A-SERVICE の使用	19
第3章 NAGIOS を使用したモニタリング	22
3.1. NAGIOS サービスのインストール	22
3.2. NAGIOS の設定	23
第4章 トラブルシューティング	31
4.1. サポート	31
4.2. IDENTITY クライアント (KEYSTONE) の接続性における問題のトラブルシューティング	31
4.3. OPENSTACK NETWORKING に関する問題のトラブルシューティング	32
4.4. ダッシュボードでのネットワークまたはルータータブの表示に関するトラブルシューティング	33
4.5. DASHBOARD でのインスタンス起動エラーに関するトラブルシューティング	33
4.6. DASHBOARD の KEYSTONE V3 認証のトラブルシューティング	34

前書き

本ガイドは、Red Hat OpenStack Platform 環境で利用可能なロギングおよびモニタリング機能の概要と発生する可能性のある問題のトラブルシューティングの方法を説明します。

第1章 ロギング

Red Hat OpenStack Platform は、特定のログファイルに情報メッセージを書き込みます。このメッセージを使用することで、システムイベントのモニタリングやトラブルシューティングが可能です。



注記

個別のログファイルをサポートケースに手動で添付する必要はありません。必要な情報はすべて **sosreport** で自動的に収集されます。この件に関しては、「[4章 トラブルシューティング](#)」で説明しています。

1.1. OPENSTACK サービスのログファイル

OpenStack のコンポーネントごとに、個別のロギングディレクトリーがあり、その中に各実行サービスに固有のファイルが格納されます。

1.1.1. Bare Metal Provisioning (Ironic) のログファイル

サービス	サービス名	ログへのパス
OpenStack Ironic API	openstack-ironic-api.service	/var/log/ironic/ironic-api.log
OpenStack Ironic Conductor	openstack-ironic-conductor.service	/var/log/ironic/ironic-conductor.log

1.1.2. Block Storage (cinder) ログファイル

サービス	サービス名	ログへのパス
Block Storage API	openstack-cinder-api.service	/var/log/cinder/api.log
Block Storage バックアップ	openstack-cinder-backup.service	/var/log/cinder/backup.log
情報メッセージ	cinder-manage コマンド	/var/log/cinder/cinder-manage.log
Block Storage スケジューラー	openstack-cinder-scheduler.service	/var/log/cinder/scheduler.log

サービス	サービス名	ログへのパス
Block Storage ボリューム	openstack-cinder-volume.service	/var/log/cinder/volume.log

1.1.3. Compute (nova) ログファイル

サービス	サービス名	ログへのパス
OpenStack Compute API サービス	openstack-nova-api.service	/var/log/nova/nova-api.log
OpenStack Compute 証明書サーバー	openstack-nova-cert.service	/var/log/nova/nova-cert.log
OpenStack Compute サービス	openstack-nova-compute.service	/var/log/nova/nova-compute.log
OpenStack Compute コンダクターサービス	openstack-nova-conductor.service	/var/log/nova/nova-conductor.log
OpenStack Compute VNC コンソールの認証サーバー	openstack-nova-consoleauth.service	/var/log/nova/nova-consoleauth.log
情報メッセージ	nova-manage コマンド	/var/log/nova/nova-manage.log
OpenStack Compute NoVNC プロキシサービス	openstack-nova-novncproxy.service	/var/log/nova/nova-novncproxy.log
OpenStack Compute スケジューラーサービス	openstack-nova-scheduler.service	/var/log/nova/nova-scheduler.log

1.1.4. Dashboard (horizon) ログファイル

サービス	サービス名	ログへのパス
------	-------	--------

サービス	サービス名	ログへのパス
特定のユーザーとの対話ログ	Dashboard インターフェース	/var/log/horizon/horizon.log

Apache HTTP サーバーは、Dashboard Web インターフェース用に追加のログファイルを複数使用します。これらのファイルは、Web ブラウザーまたはコマンドラインクライアント (keystone、nova) を使用してアクセスできます。以下のログファイルは、Dashboard の使用のトラッキングや、問題診断に役立ちます。

目的	ログへのパス
処理済みの HTTP 要求すべて	/var/log/httpd/horizon_access.log
HTTP エラー	/var/log/httpd/horizon_error.log
管理者ロールの API 要求	/var/log/httpd/keystone_wsgi_admin_access.log
管理者ロールの API エラー	/var/log/httpd/keystone_wsgi_admin_error.log
メンバーロールの API 要求	/var/log/httpd/keystone_wsgi_main_access.log
メンバーロールの API エラー	/var/log/httpd/keystone_wsgi_main_error.log



注記

nagios など、同じホスト上で実行する他の Web サービスから報告されたエラーを格納する **/var/log/httpd/default_error.log** もあります。

1.1.5. Data Processing (sahara) ログファイル

サービス	サービス名	ログへのパス
Sahara API サーバー	openstack-sahara-all.service openstack-sahara-api.service	/var/log/sahara/sahara-all.log /var/log/messages

サービス	サービス名	ログへのパス
Sahara Engine サーバー	openstack-sahara-engine.service	/var/log/messages

1.1.6. Database as a Service (trove) ログファイル

サービス	サービス名	ログへのパス
OpenStack Trove API サービス	openstack-trove-api.service	/var/log/trove/trove-api.log
OpenStack Trove Conductor サービス	openstack-trove-conductor.service	/var/log/trove/trove-conductor.log
OpenStack Trove guestagent サービス	openstack-trove-guestagent.service	/var/log/trove/logfile.txt
OpenStack Trove taskmanager サービス	openstack-trove-taskmanager.service	/var/log/trove/trove-taskmanager.log

1.1.7. Identity サービス (keystone) ログファイル

サービス	サービス名	ログへのパス
OpenStack Identity サービス	openstack-keystone.service	/var/log/keystone/keystone.log

1.1.8. Image Service (glance) ログファイル

サービス	サービス名	ログへのパス
OpenStack Image Service API サーバー	openstack-glance-api.service	/var/log/glance/api.log
OpenStack Image Service レジストリーサーバー	openstack-glance-registry.service	/var/log/glance/registry.log

1.1.9. Networking (neutron) ログファイル

サービス	サービス名	ログへのパス
OpenStack Neutron DHCP エージェント	neutron-dhcp-agent.service	/var/log/neutron/dhcp-agent.log
OpenStack Networking レイヤー 3 エージェント	neutron-l3-agent.service	/var/log/neutron/l3-agent.log
メタデータエージェントサービス	neutron-metadata-agent.service	/var/log/neutron/metadata-agent.log
メタデータの名前空間プロキシ	なし	/var/log/neutron/neutron-ns-metadata-proxy- UUID .log
Open vSwitch エージェント	neutron-openvswitch-agent.service	/var/log/neutron/openvswitch-agent.log
OpenStack Networking サービス	neutron-server.service	/var/log/neutron/server.log

1.1.10. Object Storage (swift) ログファイル

OpenStack Object Storage は、システムのロギング機能にのみ、ログを送信します。



注記

デフォルトでは、Object Storage ログファイルはすべて local0、local1、local2 syslog 機能を使用して /var/log/swift/swift.log に送られます。

Object Storage のログメッセージは主に、REST API サービスのログメッセージと、バックグラウンドデーモンのログメッセージの 2 つのカテゴリに分類されます。API サービスのメッセージには、API 要求ごとに 1 行含まれています。これは、一般的に使用されている HTTP サーバーによく似た形式となっており、フロントエンド (プロキシ) およびバックエンド (アカウント、コンテナー、オブジェクト) の両サービスがこのようなメッセージをポストします。デーモンメッセージは、(API サービスのものと比べ) 構造化されておらず、通常、定期的なタスクを実行するデーモンに関する情報が人間が判読できる形で含まれています。ただし、Object Storage のどの部分がメッセージを生成するかに関わらず、ソースのアイデンティティは必ず行頭に記載されます。

プロキシメッセージ例

```
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
```

バックグラウンドデーモンからのアドホックメッセージ例

```
Apr 27 17:08:15 rhev-a24c-02 object-auditor: Object audit (ZBF). Since
Mon Apr 27 21:08:15 2015: Locally: 1 passed, 0 quarantined, 0 errors
files/sec: 4.34 , bytes/sec: 0.00, Total time: 0.23, Auditing time:
0.00, Rate: 0.00
Apr 27 17:08:16 rhev-a24c-02 object-auditor: Object audit (ZBF)
"forever" mode completed: 0.56s. Total quarantined: 0, Total errors: 0,
Total files/sec: 14.31, Total bytes/sec: 0.00, Auditing time: 0.02,
Rate: 0.04
Apr 27 17:08:16 rhev-a24c-02 account-replicator: Beginning replication
run
Apr 27 17:08:16 rhev-a24c-02 account-replicator: Replication run OVER
Apr 27 17:08:16 rhev-a24c-02 account-replicator: Attempted to replicate
5 dbs in 0.12589 seconds (39.71876/s)
Apr 27 17:08:16 rhev-a24c-02 account-replicator: Removed 0 dbs
Apr 27 17:08:16 rhev-a24c-02 account-replicator: 10 successes, 0
failures
```

1.1.11. Orchestration (heat) ログファイル

サービス	サービス名	ログへのパス
OpenStack Heat API サービス	openstack-heat-api.service	/var/log/heat/heat-api.log
Openstack Heat エンジンサービス	openstack-heat-engine.service	/var/log/heat/heat-engine.log
Orchestration サービスのイベント	なし	/var/log/heat/heat-manage.log

1.1.12. Shared Filesystem サービス (manila) ログファイル

サービス	サービス名	ログへのパス
OpenStack Manila API サーバー	openstack-manila-api.service	/var/log/manila/api.log
OpenStack Manila スケジューラー	openstack-manila-scheduler.service	/var/log/manila/scheduler.log

サービス	サービス名	ログへのパス
OpenStack Manila 共有サービス	openstack-manila-share.service	/var/log/manila/share.log



注記

Manila Python ライブラリーからの情報の一部は **/var/log/manila/manila-manage.log** にロギングすることもできます。

1.1.13. Telemetry (ceilometer) ログファイル

サービス	サービス名	ログへのパス
OpenStack ceilometer 通知エージェント	openstack-ceilometer-notification.service	/var/log/ceilometer/agent-notification.log
OpenStack ceilometer アラーム評価	openstack-ceilometer-alarm-evaluator.service	/var/log/ceilometer/alarm-evaluator.log
OpenStack ceilometer アラーム通知	openstack-ceilometer-alarm-notifier.service	/var/log/ceilometer/alarm-notifier.log
OpenStack ceilometer API	openstack-ceilometer-api.service	/var/log/ceilometer/api.log
情報メッセージ	MongoDB integration	/var/log/ceilometer/ceilometer-dbsync.log
OpenStack ceilometer 中央エージェント	openstack-ceilometer-central.service	/var/log/ceilometer/central.log
OpenStack ceilometer コレクション	openstack-ceilometer-collector.service	/var/log/ceilometer/collector.log
OpenStack ceilometer コンピュートエージェント	openstack-ceilometer-compute.service	/var/log/ceilometer/compute.log

1.1.14. 補足サービスのログファイル

以下のサービスは、中核となる OpenStack コンポーネントにより使用されており、サービスごとに独自のログのディレクトリーとファイルが存在します。

サービス	サービス名	ログへのパス
メッセージブローカー (RabbitMQ)	rabbitmq-server.service	/var/log/rabbitmq/rabbit@short_hostname.log /var/log/rabbitmq/rabbit@short_hostname-sasl.log (Simple Authentication and Security Layer 関連のログメッセージ)
データベースサーバー (MariaDB)	mariadb.service	/var/log/mariadb/mariadb.log
ドキュメント指向データベース (MongoDB)	mongod.service	/var/log/mongodb/mongodb.log
仮想ネットワークス イッチ (Open vSwitch)	openvswitch- nonetwork.service	/var/log/openvswitch/ovsdb-server.log /var/log/openvswitch/ovs-vsitchd.log

1.2. ロギングオプションの設定

コンポーネントごとに、別のロギング設定が適切な設定ファイルに保管されています。たとえば、Compute ではこれらのオプションは **/etc/nova/nova.conf** に設定されます。

- ※ デバッグを有効化することで、情報ロギングのレベルを上げます。このオプションにより、取得する情報量が大幅に増加するため、この機能を一時的にだけ使用するか、またはログの回転設定を先に確認するようにしてください。

```
debug=True
```

- ※ 詳細ロギングを有効化します。

```
verbose=True
```

- ※ ログファイルのパスを変更します。

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

- ※ ログを中央の syslog サーバーに送信します。

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




注記

タイムスタンプの設定やログのフォーマットなどのオプションも利用できます。追加のロギングオプションについてはコンポーネントの設定ファイルを確認してください。

1.3. リモートロギングのインストールおよび設定

1.3.1. リモートロギングの概要

すべてのシステムは、直面した問題やアクションを記録するログファイルを作成して更新します。多くのシステムが含まれる分散またはクラウドコンピューティング環境では、中央ロケーションでこれらのログファイルをまとめることで、デバッグを簡素化します。

rsyslog サービスにより、集中ロギングサーバーを実行したり、個別システムがログファイルを集中ロギングサーバーに送信するように設定したりする機能が提供されます。これは、システムの **リモートロギング** 設定と呼ばれます。

1.3.2. rsyslog サーバーのインストール

rsyslog パッケージは、集中ロギングサーバーとして使用予定のシステムおよびログの送信元として設定する全システムにインストールする必要があります。これには、**root** ユーザーとしてログインして **rsyslog** パッケージをインストールします。

```
# yum install rsyslog
```

rsyslog パッケージがインストールされ、設定の準備ができました。

1.3.3. 集中ロギングサーバーでの rsyslog の設定

集中ロギングサーバーとして使用予定のシステム上で、以下の手順に記載するステップを行うようにしてください。また、すべてのステップは **root** ユーザーとしてログインして実行する必要があります。

1. SELinux が **rsyslog** トラフィックを許可するように設定します。

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

2. テキストエディターで **/etc/rsyslog.conf** ファイルを開きます。

- a. ファイルに以下の行を追加して、ログの保存先となる場所を定義します。

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

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

- b. ファイル内の以下の行から、行頭のコメント文字 (#) を削除します。


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

- c. **/etc/rsyslog.conf** ファイルへの変更を保存します。

環境内の他のシステムからのログファイルを受信し、保管するための集中ログサーバーの設定が完了しました。

1.3.4. 個々のノードでの **rsyslog** 設定

以下の手順に記載するステップを各システムに適用して、集中ログサーバーにログを送信するように設定します。これらのステップはすべて **root** ユーザーとしてログインして実行する必要があります。

1. **/etc/rsyslog.conf** を編集して以下を追加し、集中ログサーバーのアドレスを指定します。

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

YOURSERVERADDRESS は集中ログサーバーのアドレスに、**YOURSERVERPORT** は、rsyslog サービスをリッスンするポートに置き換えます。以下はその例です。

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

または

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

@ が 1 つの場合は、転送プロトコルに UDP が指定されます。TCP 転送プロトコルを指定するには、@@ を使用してください。

重要

上記の例で、ワイルドカード文字 (*) を使用している箇所は、全ログファシリティからの全ログプライオリティのログエントリをリモートの rsyslog サーバーに送信する必要があることを rsyslog に対して示しています。

より厳密なフィルターをログファイルに適用する方法についての説明は、rsyslog 設定ファイル (**rsyslog.conf**) の man ページを参照してください。このページには、**man rsyslog.conf** のコマンドを実行するとアクセスすることができます。

2. **rsyslog** サービスが起動または再起動されると、システムは全ログメッセージを集中ログサーバーに送信します。

1.3.5. **rsyslog** サーバーの起動

rsyslog サービスは、集中ログサーバーと、そのサーバーにログ記録を試みるシステムの両方で実行する必要があります。

以下の手順に記載するステップは **root** ユーザーとしてログインして実行する必要があります。

1. rsyslog サービスを起動します。

```
# service rsyslog start
```

2. 今後 rsyslog サービスが自動的に起動するように設定します。

```
# chkconfig rsyslog on
```

rsyslog サービスが起動されました。サービスは、ローカル設定に基づいて、ログの送受信を開始します。

第2章 TELEMETRY サービスを使用したモニタリング

ceilometer コマンドのヘルプを表示するには、以下のコマンドを使用します。

```
# ceilometer help
```

サブコマンドのヘルプを表示するには、以下のコマンドを使用します。

```
# ceilometer help subcommand
```

2.1. 既存のアラームの表示

設定済みの Telemetry アラームを一覧表示するには、以下のコマンドを使用します。

```
# ceilometer alarm-list
```

リソースの設定済みメーターを一覧表示するには、以下のコマンドを実行します。

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

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

UUID は、既存のリソース (例: インスタンス、イメージ、ボリュームなど) のリソース ID に置き換えます。

2.2. アラームの設定

閾値を超えるとアラームがアクティブ化されるように設定するには、**ceilometer alarm-threshold-create** コマンドを以下の構文で使します。


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

例

個別のインスタンスの平均 CPU 使用率が、3 回連続で 600 秒間 (10 分間)、50% を超過した場合にアクティベートされるようにアラームを設定するには、以下のコマンドを使用します。

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

この例では、通知アクションはログメッセージです。

既存の閾値アラームを編集するには、**ceilometer alarm-threshold-update** コマンドでアラームの ID を指定し、1 つまたは複数のオプションを追加して実行します。

例

アラームの閾値を 75% に増やすには、以下のコマンドを使用します。

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

2.3. アラームの無効化または削除

アラームを無効にするには、以下のコマンドを使用します。

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

アラームを削除するには、以下のコマンドを使用します。

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

2.4. サンプルの表示

特定のメーター名のサンプルをすべて一覧表示するには、以下のコマンドを使用します。

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

タイムスタンプの範囲内で特定のリソースを対象としたサンプルのみを一覧表示するには、以下のコマンドを使用します。

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

START_TIME および **END_TIME** は **iso-dateThh:mm:ss** の形式で指定してください。

例

13:10:00 から 14:25:00 までの範囲で取得したサンプルをインスタンスに照会するには、以下のコマンドを使用します。

```
# ceilometer sample-list --meter cpu --query 'resource_id=5056eda6-8a24-4f52-9cc4-c3ddb6fb4a69;timestamp>2015-01-12T13:10:00;timestamp>=2015-01-12T14:25:00'
```

Resource ID Timestamp	Name	Type	Volume	Unit
5056eda6-8a24-... 2015-01-12T14:21:44	cpu	cumulative	3.5569e+11	ns
5056eda6-8a24-... 2015-01-12T14:11:45	cpu	cumulative	3.0041e+11	ns
5056eda6-8a24-... 2015-01-12T14:01:54	cpu	cumulative	2.4811e+11	ns
5056eda6-8a24-... 2015-01-12T13:30:54	cpu	cumulative	1.3743e+11	ns
5056eda6-8a24-... 2015-01-12T13:20:54	cpu	cumulative	84710000000.0	ns
5056eda6-8a24-... 2015-01-12T13:10:54	cpu	cumulative	31170000000.0	ns

2.5. サンプルの作成

Telemetry サービスに送信するためのサンプルを作成することができます。これらは、以前に定義されていたメーターに対応する必要はありません。以下の構文を使用します。

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

METER_TYPE は以下のいずれかに置き換えます。

- ※ Cumulative: 累計
- ※ Delta: 経時的な変化または差異
- ※ Gauge: 離散値

例

```
# ceilometer sample-create -r 5056eda6-8a24-4f52-9cc4-c3ddb6fb4a69 -m On_Time_Mins --meter-type cumulative --meter-unit mins --sample-volume 0
```

Property	Value
message_id	521f138a-9a84-11e4-8058-525400ee874f


```

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

```

volume の箇所は通常、サンプリングのアクションの結果として取得した値ですが、上記の例の場合は、コマンドによって生成された値です。



注記

サンプルは、作成された時点に Telemetry サービスに送信されるため、更新されません。サンプルは、実質的にはメッセージであるため、メッセージ ID があります。新規サンプルを作成するには、**sample-create** コマンドを再度実行して **--sample-volume** の値を更新します。

2.6. クラウドの使用状況の統計の表示

OpenStack の管理者は、ダッシュボードを使用してクラウドの統計情報を確認することができます。

1. Dashboard に管理ユーザーとしてログインして **管理 > システム > リソース使用状況** を選択します。
2. 以下のいずれかを選択します。
 - ※ 日次レポート: プロジェクト別の日次使用状況のレポートを表示します。期間とプロジェクト数の上限を選択して、**レポートの作成** をクリックすると日次使用状況のレポートが表示されます。
 - ※ 統計情報: プロジェクト別にグループ化されたメトリックのグラフを表示します。ドロップダウンメニューで値と期間を選択すると、表示されるグラフが自動的に更新されます。

クラウドの使用状況の統計情報は、**ceilometer** コマンドラインクライアントを使用して表示することもできます。

例

cpu_util メーターの全統計を確認するには、以下のコマンドを入力します。

```

# ceilometer statistics --meter cpu_util
+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+
| Period | Period Start | Period End | Max | Min | Avg | Sum |
Count| Dura...
+-----+-----+-----+-----+-----+-----+-----+

```



```

+-----+-----+
| 0          | 2015-01-09T14: | 2015-01-09T14:2 | 9.44 | 0.0 | 6.75 | 337.94 |
50    | 2792...
+-----+-----+-----+-----+-----+-----+-----+
-+-----+-----+

```

例

統計は、**--query** オプションで特定のリソースに限定したり、**timestamp** オプションで特定の期間に限定したりすることが可能です。

```

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

```

2.7. TIME-SERIES-DATABASE-AS-A-SERVICE の使用

Time-Series-Database-as-a-Service (gnocchi) はマルチテナントのメトリックおよびリソースのデータベースです。大規模なメトリックを格納する一方でオペレーターやユーザーにメトリックおよびリソースの情報へのアクセスを提供します。

現在、TSDaaS は認証に Identity サービスを、データの保存に Ceph と Object Storage を使用します。

TDSaaS は **statsd** プロトコルと互換性のある **statsd** デーモンを提供し、**gnocchi-statsd** と呼ばれるネットワークで送信されるメトリックをリッスンすることができます。TDSaaS での **statsd** サポートを有効化するには、設定ファイルで **[statsd]** オプションを設定する必要があります。リソース ID パラメーターは、全メトリックがアタッチされる主要な一般リソース、リソースとメトリックに関連付けられるユーザーとプロジェクト ID、メトリックの作成に使用するアーカイブポリシー名として使用されます。

メトリックは **gnocchi-statsd** に送信されるため、すべてのメトリックは動的に作成され、指定の名前を設定したリソース ID にアタッチします。TSDaaS のインストールおよび設定に関する詳しい情報は、<https://access.redhat.com/documentation/ja/red-hat-enterprise-linux-openstack-platform/> から『インストールリファレンスガイド』の「Time-Series-Database-as-a-サービスのインストール」の章を参照してください。

注記

Time-Series-Database-as-a-Service (gnocchi) は、Red Hat OpenStack Platform 8 ではテクノロジープレビューとして提供しています。

テクノロジープレビューとして記された機能のサポート範囲についての詳しい情報は、<https://access.redhat.com/support/offerings/techpreview/> を参照してください。

2.7.1. Time-Series-Database-as-a-Service の実行

HTTP サーバーとメトリックデーモンを実行して、Time-Series-Database-as-a-Service (TSDaaS) を実行します。

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

2.7.2. WSGI アプリケーションとしての実行

mod_wsgi または他の WSGI アプリケーションなど WSGI サービスで TSDaaS を実行できます。TSDaaS で提供される **gnocchi/rest/app.wsgi** により、WSGI アプリケーションとして Gnocchi を有効化できます。

TSDaaS API 層は、WSGI を使用して実行します。つまり、Apache **httpd** および **mod_wsgi** または **uwsgi** などの別の HTTP デーモンを使用して実行できるということです。CPU の数に合わせてプロセスやスレッド数を設定してください。通常は **1.5 × CPU の数** です。サーバーが 1 台では十分でない場合には、新たに API サーバーを起動して異なるマシン上にでも Gnocchi をスケールアウトできます。

2.7.3. metricd ワーカー

デフォルトでは **gnocchi-metricd** デーモンは、すべての CPU 電源機能をチェックしてメトリックの集計を算出する時の CPU の使用率を最大化します。**gnocchi status** コマンドを使用して HTTP API を照会し、メトリック処理のクラスターのステータスを取得します。このコマンドにより、処理するメトリック数が表示されます。これは、**gnocchi-metricd** の処理バックログとして知られています。このバックログが増え続けている限り、**gnocchi-metricd** は受信するメトリックの量を処理できるということです。処理する測定値の数が継続的に増えている場合には、**gnocchi-metricd** デーモンの数を (一時的に) 増やす必要があります。実行できる metricd デーモンやサーバー数に制約はありません。

2.7.4. Time-Series-Database-as-a-Service の監視

HTTP API の **/v1/status** エンドポイントは、処理する測定値の数 (測定値のバックログ) などさまざまな情報を返し、簡単に監視することができます。HTTP サーバーと **gnocchi-metricd** デーモンは実行中であり、ログに警告内容が書きこまれてないことを確認できると、全体的なシステムのヘルスが良好であることが分かります。

2.7.5. Time-Series-Database-as-a-Service のバックアップと復元

障害から回復できるように、インデックスとストレージの両方をバックアップする必要があります。つまり、データベースダンプ (PostgreSQL または MySQL) を作成して、データストレージ (Ceph、Swift、またはファイルシステム) のスナップショットかコピーを作成してください。復元

の手順は、インデックスとストレージのバックアップを復元して、必要に応じて TSDaaS を再インストールしてから、再起動します。

第3章 NAGIOS を使用したモニタリング

3.1. NAGIOS サービスのインストール

Nagios モニタリングシステムは、OpenStack ネットワークとインフラストラクチャーのモニタリングおよび警告の提供に使用することができます。本インストール手順では、以下のパッケージをインストールします。

nagios

ネットワーク上のホストとサービスをモニタリングして、問題が発生した場合や解決した場合にメールの送信や警告の通知ができる Nagios プログラム

nagios-devel

Nagios 関連のアプリケーションで使用可能なファイルが含まれます。

nagios-plugins*

(ping および nrpe など) Nagios 関連のアプリケーションの Nagios プラグイン

gd

動的にイメージを作成するためのグラフィックライブラリー

gd-devel

グラフィックライブラリーの開発ライブラリー (gd)

php

Web インターフェース向けに Nagios が使用する HTML 埋め込みスクリプト言語

gcc、glibc、glibc-common

GNU コンパイラコレクション、標準プログラミングライブラリー、およびバイナリー (ロケールサポートを含む)

openssl

マシン間のセキュアな通信のサポートを提供する OpenSSL ツールキット

yum コマンドで **root** ユーザーとして必要なパッケージをインストールします。

```
# yum install nagios nagios-devel nagios-plugins\* gd gd-devel php gcc  
glibc glibc-common openssl
```

注記

これらのいずれかのパッケージが即時に利用できない場合には (例: gd-devel や gcc など)、**subscription-manager** を使用してオプションの Red Hat チャンネルを有効にする必要がある可能性があります。

```
# subscription-manager repos --enable rhel-7-server-optional-  
rpms
```


3.1.1. Nagios サービスの配置

OpenStack 環境の外部にあるサーバーへ Nagios をデプロイするように検討してみてください。システムに問題が発生した場合に診断情報を受信することができます。さらに、Nagios を最適な状態に設定するには、複数の点を検討する必要があります。

1. SSH を使用する場合には、Nagios サービスの CPU オーバーヘッドは高くなる可能性があります。
2. Nagios は、セキュリティイベントを監視する場合は特に、セキュアにロックダウンされたサーバーでホストする必要があります。Nagios サーバーは、幅広い範囲のシステムからトラフィックを受信します。セキュリティの分離が必要な場合には、特権システムと見なされ、OpenStack ノードに適用するルール以外に、ファイアウォールルールを追加する必要があります。
3. Nagios サーバーは、大量のネットワークトラフィックを受信するため、リソースの競合が発生する可能性があります。

3.1.2. NRPE アドオンのインストール

NRPE (Nagios Remote Plugin Executor) プラグインは、ホストのサービスのステータスをチェックして、Nagios サービスにレポートを送り返すのに使用する、コンパイル済みの実行可能ファイルまたはスクリプトです。OpenStack クラウドが複数のマシン間で分散されている場合には、NRPE アドオンを使用して、それらのリモートマシン上のプラグイン情報にアクセスすることができます。

NRPE と Nagios プラグインは、モニタリングの対象となる各リモートマシンにインストールしておく必要があります。各リモートマシンで、**root** ユーザーとして次のコマンドを実行します。

```
# yum install -y nrpe nagios-plugins\* openssl
```

インストールの終了後には、**/usr/lib64/nagios/plugins/** ディレクトリーで使用可能なすべてのプラグインを確認することができます。



注記

リモートの Nagios プラグインへのアクセスには、SSH を使用することも可能ですが、この操作を行うと、Nagios ホストとリモートマシンの両方で、CPU 負荷が過度に高くなるため、お勧めできません。

3.2. NAGIOS の設定

Nagios はサーバー、ローカルおよびリモートマシンからサーバーにオブジェクト/ホストの情報のレポートを送り返すプラグイン、Web インターフェース、およびそれらすべてを結びつける設定によって構成されます。

少なくとも以下の操作を完了しておく必要があります。

1. Web インターフェースのユーザー名とパスワードを確認し、基本設定をチェックします。
2. ローカルサーバーに OpenStack モニタリングを追加します。
3. OpenStack クラウドに分散ホストが含まれている場合には、次の作業を行います。

- a. 各リモートマシン (監視対象のサービスが実行される) に NRPE をインストールし、設定します。
- b. 監視対象のホストを Nagios に指示します。
- c. 各ホストの監視対象のサービスを Nagios に指示します。

表3.1 Nagios の設定ファイル

ファイル名	説明
/etc/nagios/nagios.cfg	Nagios のメインの設定ファイル
/etc/nagios/cgi.cfg	CGI 設定ファイル
/etc/httpd/conf.d/nagios.conf	Nagios の httpd 用設定
/etc/nagios/passwd	Nagios ユーザーのパスワードファイル
/usr/local/nagios/etc/ResourceName.cfg	ユーザー固有情報を格納
/etc/nagios/objects/ObjectsDir/ObjectsFile.cfg	サービスやコンタクトグループなどの項目についての情報を保管するために使用されるオブジェクト定義ファイル
/etc/nagios/nrpe.cfg	NRPE 設定ファイル

3.2.1. Nagios 用の HTTPD の設定

デフォルトで Nagios のインストール時に設定されるデフォルトの httpd ユーザーとパスワードは **nagiosadmin / nagiosadmin** です。この値は、**/etc/nagios/cgi.cfg** ファイルで確認することができます。

nagios の HTTPD を設定するには、以下のステップに従います。

1. **root** ユーザーとしてログインします。
2. ユーザー **nagiosadmin** のデフォルトパスワードを変更するには、以下のコマンドを実行します。

```
# htpasswd -c /etc/nagios/passwd nagiosadmin
```




注記

新規ユーザーを作成するには、新規ユーザー名で以下のコマンドを使用します。

```
# httpasswd /etc/nagios/passwd newUserName
```

3. **/etc/nagios/objects/contacts.cfg** で **nagiosadmin** のメールアドレスを更新します。

```
define contact{
    contact_name    nagiosadmin                ; Short name of user
    [...snip...]
    email           yourName@example.com    ; << CHANGE THIS
}
```

4. 基本設定が機能しているかどうかを確認します。

```
# nagios -v /etc/nagios/nagios.cfg
```

エラーが発生した場合には、**/etc/nagios/nagios.cfg** に設定されているパラメーターをチェックします。

5. システムのブート時に Nagios が自動的に起動するようにします。

```
# chkconfig --add nagios
# chkconfig nagios on
```

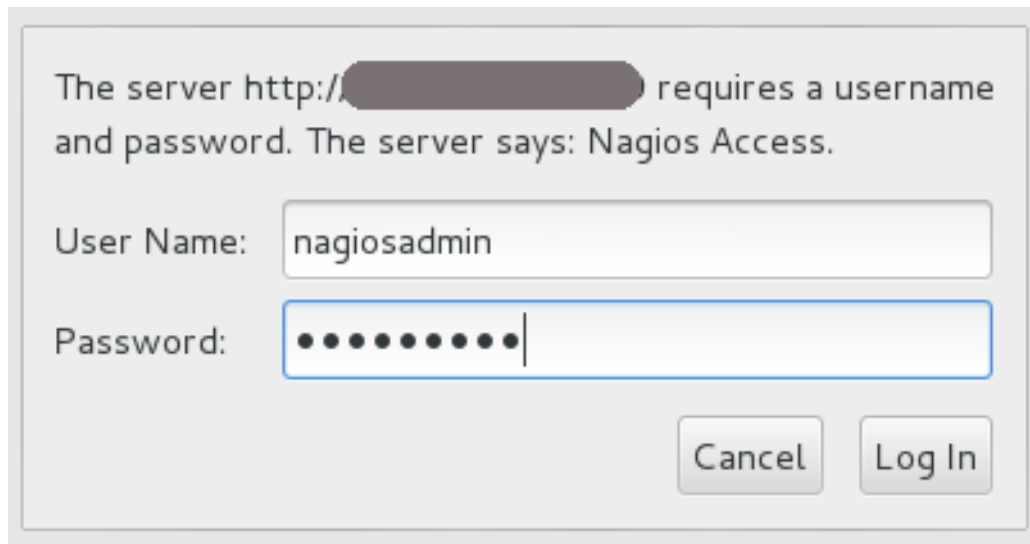
6. Nagios を起動し、httpd を再起動します。

```
# service httpd restart
# service nagios start
```

7. ブラウザーで以下の URL を使用して Nagios のアクセスをチェックします。nagiosadmin ユーザーとステップ 2 で設定したパスワードを使用してください。

```
http://nagiosHostURL/nagios
```

図3.1 Nagios のログイン



The server http://[redacted] requires a username and password. The server says: Nagios Access.

User Name:

Password:

注記

Nagios URL にアクセスできない場合には、ファイアウォールルールが正しく設定されているかどうか確認してください。

3.2.2. Nagios が OpenStack サービスを監視するようにするための設定

デフォルトでは、Nagios サーバー上の `/etc/nagios/objects/localhost.cfg` ファイルを使用して、基本ローカル統計 (例: スワップの使用率や現在のユーザー数など) のためのサービスを定義します。これらのサービスが必要なくなった場合には、各行頭に「#」の文字を入れることによって、随時コメントアウトすることが可能です。またこれと同じファイルを使用して、新規 OpenStack モニタリングサービスを追加することができます。

注記

追加のサービスファイルを使用することが可能ですが、それらのファイルは、`/etc/nagios/nagios.cfg` ファイルで `cfg_file` パラメーターとして指定する必要があります。

1. root ユーザーとしてログインします。
2. 監視対象の項目用に短いスクリプトを記述して (例: サービスが実行中か否かなど)、`/usr/lib64/nagios/plugins` ディレクトリーに配置します。

たとえば、以下のスクリプトは、コンピュートインスタンスの数をチェックし、`nova-list` という名前のファイルに保管されます。

```
#!/bin/env bash
export OS_USERNAME=username
export OS_TENANT_NAME=tenantName
export OS_PASSWORD=password
export OS_AUTH_URL=http://identityURL:35357/v2.0/

data=$(nova list 2>&1)
rv=$?
```



```

if [ "$rv" != "0" ] ; then
    echo $data
    exit $rv
fi

echo "$data" | grep -v -e '-----' -e '| Status |' -e '^$' |
wc -l

```

3. スクリプトを実行可能にします。

```
# chmod a+x nova-list
```

4. `/etc/nagios/objects/commands.cfg` ファイルで、各新規スクリプトにコマンドセクションを指定します。

```

define command {
    command_line
    /usr/lib64/nagios/plugins/nova-list
    command_name          nova-list
}

```

5. `/etc/nagios/objects/localhost.cfg` ファイルで、以下の例のように、定義済みのコマンドを使用して、各新規項目に対してサービスを定義します。

```

define service {
    check_command    nova-list
    host_name        localURL
    name             nova-list
    normal_check_interval 5
    service_description    Number of nova vm instances
    use              generic-service
}

```

6. 以下のコマンドで nagios を再起動します。

```
# service nagios restart
```

3.2.3. NRPE の設定

各リモートマシンで監視の設定をするには、以下の手順を **root** ユーザーとして実行します。

1. `/etc/nagios/nrpe.cfg` ファイルで、**allowed_hosts** の行に中央 Nagios サーバーの IP アドレスを追加します。

```
allowed_hosts=127.0.0.1, NagiosServerIP
```

2. `/etc/nagios/nrpe.cfg` ファイルで、OpenStack サービスの監視に使用するコマンドを以下の例のように追加します。

```
command[keystone]=/usr/lib64/nagios/plugins/check_procs -c 1: -w
3: -C keystone-all
```


次に、Nagios 監視サーバーの **services.cfg** ファイルに、定義済みの各コマンドを指定することができます。



注記

複雑な監視はスクリプトに配置して、コマンドの定義で参照することができます。

3. 次に、ファイアウォールを設定して **nrpe** トラフィックを許可します。
4. NRPE サービスを起動します。

```
# service nrpe start
```

3.2.4. ホストの定義の作成

Nagios がインストールされているホストに加えて、クラウドで追加のマシンを使用する場合には、Nagios が認識するようにオブジェクトファイルで設定する必要があります。

1. **root** ユーザーとしてログインします。
2. **/etc/nagios/objects/** ディレクトリーに **hosts.cfg** ファイルを作成します。
3. このファイルで、OpenStack サービスを実行している、モニタリングする必要のある各マシンの **host** セクションを指定します。

```
define host{
    use linux-server
    host_name remoteHostName
    alias remoteHostAlias
    address remoteAddress
}
```

ここで、

- ✳ **host_name** には、モニタリングするリモートマシンを指定します (通常は、ローカルの **/etc/hosts** ファイルに記載されています)。この名前は、サービスおよびホストグループの定義でホストを参照するのに使用されます。
- ✳ **alias** には、ホストを容易に特定するために使用する名前を指定します (通常は、**host_name** と同じです)。
- ✳ **address** には、ホストアドレスを指定します (通常は IP アドレスですが、FQDN を使用することも可能です。DNS サービスが使用可能であることを確認してください)。

例:

```
define host{
    host_name      Server-ABC
    alias          OS-ImageServices
    address        192.168.1.254
}
```


4. `/etc/nagios/nagios.cfg` ファイルの **OBJECT CONFIGURATION FILES** セクションの下に以下の行を記載します。

```
cfg_file=/etc/nagios/objects/hosts.cfg
```

3.2.5. リモートサービスのサービス定義の作成

リモートサービスをモニタリングするには、新規ファイル (本手順では `/etc/nagios/objects/services.cfg`) でそれらのサービスを定義する必要があります。

1. **root** ユーザーとしてログインします。
2. `/etc/nagios/objects/commands.cfg` ファイルに以下の内容を記載して、リモートのスクリプトまたはプラグインで **check_nrpe** プラグインの使用を処理するように指定します。

```
define command{
    command_name    check_nrpe
    command_line    $USER1$/check_nrpe -H $HOSTADDRESS$ -c
$ARG1$
}
```

3. `/etc/nagios/objects/` ディレクトリーに **services.cfg** ファイルを作成します。
4. 作成したファイルで、モニタリングするリモートの OpenStack ホストごとに以下の **services.cfg** セクションを記載します。

```
##Basic remote checks#####
##Remember that remoteHostName is defined in the hosts.cfg file.

define service{
    use generic-service
    host_name remoteHostName
    service_description PING
    check_command check_ping!100.0,20%!500.0,60%
}

define service{
    use generic-service
    host_name remoteHostName
    service_description Load Average
    check_command check_nrpe!check_load
}

##OpenStack Service Checks#####
define service{
    use generic-service
    host_name remoteHostName
    service_description Identity Service
    check_command check_nrpe!keystone
}
```


上記のセクションにより、サーバーのハートビート、負荷チェック、OpenStack Identity サービスのステータスレポートが、Nagios サーバーに返送されます。すべての OpenStack サービスがレポート可能です。リモートサーバーの **nrpe.cfg** ファイルに、一致するコマンドが記載されていることを確認してください。

5. **/etc/nagios/nagios.cfg** ファイルの **OBJECT CONFIGURATION FILES** セクションの下に以下の行を記載します。

```
cfg_file=/etc/nagios/objects/services.cfg
```

3.2.6. Nagios 設定の検証

1. **root** ユーザーとしてログインします。
2. 更新した設定ファイルが機能しているかどうかを確認します。

```
# nagios -v /etc/nagios/nagios.cfg
```

エラーが発生した場合に

は、**/etc/nagios/nagios.cfg**、**/etc/nagios/services.cfg**、**/etc/nagios/hosts.cfg** で指定されているパラメーターを確認してください。

3. Nagios を再起動します。

```
# service nagios restart
```

4. ブラウザーで以下の URL にアクセスして、Nagios ダッシュボードに再度ログインします。**nagiosadmin** ユーザーとステップ 1 で設定したパスワードを使用します。

```
http://nagiosHostURL/nagios
```


第4章 トラブルシューティング

本章では、Red Hat OpenStack Platform のトラブルシューティングに役立つログ記録およびサポート情報について記載します。

4.1. サポート

クライアントコマンドが失敗した場合には、Red Hat テクニカルサポートまでご連絡ください。その際には、発生した問題についての状況説明、コンソールの全出力、およびコンソールの出力で参照されているすべてのログファイル、問題のある (可能性のある) ノードからの **sosreport** を提供してください。たとえば、コンピュートレベルで問題が発生した場合には Nova ノードで **sosreport** を実行します。ネットワークの問題の場合は、Neutron ノードでユーティリティーを実行します。一般的なデプロイメントの問題の場合は、クラウドコントローラー上で **sosreport** を実行するのがベストです。

sosreport コマンド (**sos** パッケージ) についての情報は、[「What is a sosreport and how to create one in Red Hat Enterprise Linux 4.6 and later」](#) の記事を参照してください。

ヒントについては **/var/log/messages** ファイルも確認します。

4.2. IDENTITY クライアント (KEYSTONE) の接続性における問題のトラブルシューティング

Identity クライアント (**keystone**) が Identity サービスにコンタクトできない場合には、次のようなエラーが返されます。

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

この問題をデバッグするには、以下にあげる一般的な原因を確認してください。

Identity サービスが稼働していない場合

Identity サービスをホストするシステムで、サービスのステータスを確認します。

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

サービスが実行されていない場合には、root ユーザーとしてログインして起動します。

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

ファイアウォールが適切に設定されていない場合

ファイアウォールがポート **5000** と **35357** で TCP トラフィックを許可するように設定されていない可能性があります。そのような場合の修正方法については『インストールリファレンス』の「Identity サービスのトラフィックを許可するためのファイアウォール設定」を参照してください。

サービスエンドポイントが正しく定義されていない場合

Identity サービスをホストするシステムで、エンドポイントが正しく定義されているかどうかを確認します。

1. 管理トークンを取得します。

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

2. Identity サービスの正しい管理エンドポイントを決定します。

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

IP は Identity サービスをホストするシステムの IP アドレスまたはホスト名に置き換えます。**VERSION** は、使用中の API バージョンに置き換えます (**v2.0** または **v3**)。

3. 事前に定義されている Identity サービス関連の環境変数の設定を解除します。

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

4. 管理トークンとエンドポイントを使用して、Identity サービスとの認証を行います。Identity サービスのエンドポイントが正しいことを確認してください。

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

一覧表示された Identity サービスの **publicurl**、**internalurl**、および **adminurl** が正しいことを確認してください。特に、各エンドポイント内にリストされている IP アドレスとポート番号が正しく、ネットワーク上で到達可能であるようにしてください。

これらの値が正しくない場合には、正しいエンドポイントの追加方法について記載した『インストールリファレンス』の **Identity サービスのエンドポイントの作成**の説明を参照してください。正しいエンドポイントが追加されたら、誤ったエンドポイントは **keystone** コマンドの **endpoint-delete** アクションを使用して削除します。

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

TOKEN および **ENDPOINT** は、上記のステップで特定した値に置き換えます。**ID** は **endpoint-list** アクションにより一覧表示される、削除対象のエンドポイントに置き換えます。

4.3. OPENSTACK NETWORKING に関する問題のトラブルシューティング

本項では、OpenStack Networking サービスに関する問題のトラブルシューティングに使用することができるさまざまなコマンドと手順について説明します。

ネットワークデバイスのデバッグ

※ **ip a** コマンドで、全物理/仮想デバイスを表示します。

- ※ **ovs-vsctl show** コマンドで、仮想スイッチ内のインターフェースとブリッジを表示します。
- ※ **ovs-dpctl show** コマンドで、スイッチ上のデータパスを表示します。

ネットワークパケットの追跡

- ※ **tcpdump** コマンドで、パケットが通過しない場所を確認します。

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

INTERFACE は、パケットが通過できない箇所を確認するためのネットワークインターフェース名に置き換えます。このインターフェース名には、ブリッジまたはイーサネットデバイスの名前を使用することができます。

-e フラグで、リンクレベルヘッダーがダンプされるようにします (その場合には、**vlan** タグが表示されます)。

-w フラグはオプションです。出力をファイルに書き込みたい場合にのみ使用することができます。使用しない場合には、その出力は標準出力 (**stdout**) に書き込まれます。

tcpdump についての詳細は、**man tcpdump** のコマンドで **man** ページを開いて参照してください。

ネットワーク名前空間のデバッグ

- ※ **ip netns list** コマンドで、既知のネットワーク名前空間をすべて一覧表示します。
- ※ **ip netns exec** コマンドで、特定の名前空間内のルーティングテーブルを表示します。

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

bash シェルで **ip netns exec** コマンドを起動し、それ以降に実行するコマンドが **ip netns exec** コマンドを実行しなくても呼び出されるようにします。

4.4. ダッシュボードでのネットワークまたはルータータブの表示に関するトラブルシューティング

Networks および **Routers** のタブは、OpenStack Networking を使用するように環境が設定されている場合にのみ表示されます。現在、デフォルトでは、Packstack ユーティリティーによって Nova ネットワークがデプロイされるため、この方法でデプロイされた環境には、これらのタブは表示されない点に特に注意してください。

OpenStack Networking が環境にデプロイされているにもかかわらずタブが表示されない場合には、Identity サービスでサービスエンドポイントが正しく定義されて、ファイアウォールがそのエンドポイントへのアクセスを許可し、サービスが稼働していることを確認してください。

4.5. DASHBOARD でのインスタンス起動エラーに関するトラブルシューティング

ダッシュボードを使用したインスタンス起動時に操作が失敗した場合には、汎用の **ERROR** メッセージが表示されます。実際の原因を究明するには、コマンドラインツールを使用する必要があります。

ます。

nova list でインスタンスの一意識別子を確認します。次にその識別子を **nova show** コマンドの引数として使用します。返される項目の 1 つがエラー条件となります。最も一般的な値は **NoValidHost** です。

このエラーは、インスタンスをホストするのに十分なリソースが利用できる有効なホストがないことを示しています。この問題を回避するには、より小さなインスタンスサイズを選択するか、その環境のオーバーコミットの上限を高くする方法を検討してください。



注記

インスタンスをホストするには、コンピュータノードで CPU および RAM リソースが使用可能なだけでなく、インスタンスに関連付けられる一時ストレージ用に十分なディスク領域がある必要もあります。

4.6. DASHBOARD の KEYSTONE V3 認証のトラブルシューティング

django_openstack_auth は、Django の contrib.auth フレームワークと連携する、プラグ可能な Django 認証バックエンドで、OpenStack Identity サービス API に対してユーザー認証を行います。Django_openstack_auth は、トークンオブジェクトを使用して、ユーザーおよび Keystone 関連の情報をカプセル化し、Dashboard は、トークンオブジェクトを使用して Django ユーザーオブジェクトを再構築します。

現在、トークンオブジェクトは以下を格納します。

- ※ keystone トークン
- ※ ユーザー情報
- ※ 範囲
- ※ ロール
- ※ サービスカタログ

Dashboard は、ユーザーセッションデータの処理に Django のセッションフレームワークを使用します。以下は、利用可能な各種セッションバックエンド一覧です。これらは、local_settings.py ファイルの SESSION_ENGINE 設定で制御されます。

- ※ ローカルメモリーキャッシュ
- ※ Memcached
- ※ データベース
- ※ キャッシュされたデータベース
- ※ クッキー

特に署名付きクッキーのセッションバックエンドが使用されている場合、多数または多くのサービスが一度に有効化された場合など、クッキーのサイズが制限に到達して、Dashboard へのログインに失敗する可能性があります。クッキーサイズが増加する理由の 1 つとして、サービスカタログが挙げられます。多くのサービスが登録されるにつれ、サービスカタログのサイズも増加します。

このようなシナリオでは (特に keystone v3 認証を使用している場合)、セッショントークン管理を向上するため、Dashboard へログインするための以下の設定を含めてください。

1. /usr/share/openstack-dashboard/openstack_dashboard/settings.py では、以下の設定を追加します。

```
DATABASES =
{
    'default':
    {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'horizondb',
        'USER': 'User Name',
        'PASSWORD': 'Password',
        'HOST': 'localhost',
    }
}
```

2. 同じファイルで、SESSION_ENGINE を以下に変更します。

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

3. mysql コマンドを使用してデータベースサービスに接続します。USER は、接続に使用するユーザー名に置き換えます。また、USER は root ユーザー (または正しいパーミッション「create db」を持つユーザー) でなければなりません。

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

4. Horizon データベースを作成します。

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

5. mysql クライアントを終了します。

```
mysql > exit
```

6. 以下のコマンドで、openstack_dashboard ディレクトリーに移動して、データベースを同期します。

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

スーパーユーザーを作成する必要はないため、質問には「n」と回答します。

7. Apache http サーバーを再起動します。Red Hat Enterprise Linux の場合は以下を実行します。

```
#service httpd restart
```