Red Hat OpenStack Platform 13

運用データの計測

物理リソースおよび仮想リソースのトラッキングおよびメトリックの収集

Last Updated: 2023-04-10

Red Hat OpenStack Platform 13 運用データの計測

物理リソースおよび仮想リソースのトラッキングおよびメトリックの収集

OpenStack Team
rhos-docs@redhat.com

法律上の通知

Copyright © 2023 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, the Red Hat 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 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

運用ツールを使用して、Red Hat OpenStack Platform 環境の計測と維持に役立てます。

目次

第1章 運用データ計測の概要

Red Hat OpenStack Platform 環境の Telemetry サービスのコンポーネントにより、物理リソースおよび仮想リソースをトラッキングし、デプロイメントにおける CPU の使用状況やリソースの可用性などのメトリックを収集することができます。これには、Gnocchi バックエンドに集約値を保管するデータ収集デーモンが使用されます。

1.1. 運用データの計測について

運用ツールを使用して、Red Hat OpenStack Platform 環境の計測と維持に役立てます。これらの計測ツールは、以下の機能を果たします。

- **可用性のモニターリング**: Red Hat OpenStack Platform (RHOSP) 環境内の全コンポーネントを監視して、いずれかのコンポーネントが現在使用できない、または機能していない状態かどうかを判断します。また、問題が確認された時にシステムがアラートを送信するように設定することも可能です。

- **パフォーマンスのモニターリング**: データ収集デーモンを使用してシステム情報を定期的に収集し、その値をさまざまな方法で保管および監視することができます。このデーモンにより、収集したオペレーティングシステムやログファイル等のデータを保管することや、ネットワークを通じてデータを利用できるようにすることが可能です。データから取得した統計値を使用して、システムの監視、パフォーマンスのボトルネックの特定、および将来的なシステム負荷の予測を行うことができます。

1.2. TELEMETRY のアーキテクチャー

Red Hat OpenStack Platform (RHOSP) Telemetry は、OpenStack をベースとするクラウドのユーザーレベルの使用状況データを提供します。このデータを、顧客への請求、システムのモニターリング、またはアラートに使用することができます。Telemetry のコンポーネントを設定し、既存の OpenStack コンポーネントにより送信される通知から (例: Compute の使用状況イベント)、または RHOSP インフラストラクチャーリソースへのポーリングにより (例: libvirt)、データを収集することができます。Telemetry は、収集したデータをデータストアやメッセージキュー等のさまざまなターゲットに公開します。

Telemetry は以下のコンポーネントで設定されます。

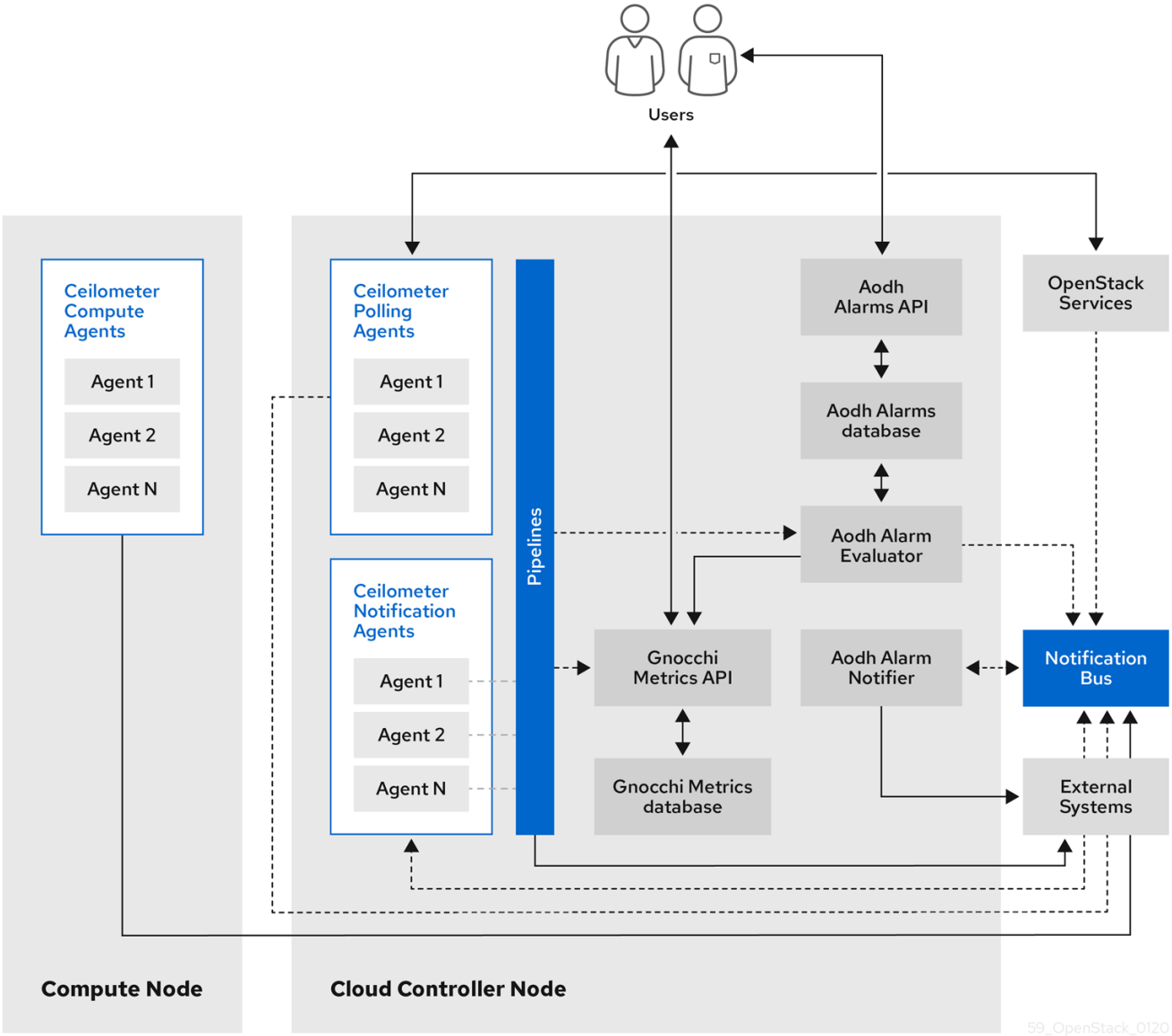
- **データ収集**: Telemetry は Ceilometer を使用してメトリックデータおよびイベントデータを収集します。詳細は、「Ceilometer」を参照してください。

- **ストレージ**: Telemetry は、メトリックデータを Gnocchi に、イベントデータを Panko に保存します。詳細は、「Gnocchi を使用したストレージ」を参照してください。

- **アラームサービス**: Telemetry は、Aodh を使用してアクションをトリガーします。アクションのトリガーは、Ceilometer の収集するメトリックデータまたはイベントデータに対して定義されたルールに基づきます。

データを収集したら、Red Hat Cloudforms 等のサードパーティーツールを使用してメトリックデータを表示および解析し、アラームサービス Aodh を使用してイベントに対するアラームを設定することができます。

図1.1 Telemetry のアーキテクチャー



1.3. データ収集

Red Hat OpenStack Platform (RHOSP) は、2 種類のデータ収集をサポートします。

- インフラストラクチャーモニターリング用の collectd。詳細は、「collectd」を参照してください。
- OpenStack コンポーネントレベルのモニターリング用の Ceilometer。詳細は、「Ceilometer」を参照してください。

1.3.1. Ceilometer

Ceilometer は OpenStack Telemetry サービスのデフォルトのデータ収集コンポーネントで、現在の OpenStack コアコンポーネント全体にわたってデータを正規化し変換することができます。Ceilometer は、OpenStack サービスに関連する計測データおよびイベントデータを収集します。デプロイメントの設定に応じて、ユーザーは収集したデータにアクセスすることができます。

Ceilometer サービスは、3 つのエージェントを使用して Red Hat OpenStack Platform (RHOSP) コンポーネントからデータを収集します。

- **コンピュートエージェント (ceilometer-agent-compute)**: 各コンピュートノードで実行され、リソースの使用状況の統計値をポーリングします。このエージェントは、パラメーター **--polling namespace-compute** を使用して実行しているポーリングエージェント **ceilometer-polling** と同じです。

- **中央エージェント (ceilometer-agent-central)**: 中央の管理サーバーで実行され、インスタンスまたはコンピュートノードに関連付けられないリソースの使用状況の統計値をポーリングします。複数のエージェントを起動して、サービスをスケーリングすることができます。これは、パラメーター **--polling namespace-central** で実行されているポーリングエージェント **ceilometer-polling** と同じです。

- **通知エージェント (ceilometer-agent-notification)**: 中央の管理サーバーで実行され、メッセージキューからのメッセージを処理してイベントデータおよび計測データをビルドします。その後、データは定義されたターゲットに公開されます。デフォルトでは、データは Gnocchi にプッシュされます。これらのサービスは、RHOSP の通知バスを使用して通信します。

Ceilometer エージェントは、パブリッシャーを使用してデータを該当するエンドポイント (Gnocchi 等) に送信します。この情報は、**pipeline.yaml** ファイルで設定することができます。

関連情報

- パブリッシャーについての詳細は、「パブリッシャー」を参照してください。

1.3.1.1. パブリッシャー

Telemetry サービスでは、さまざまな転送方法により収集したデータを外部システムに送付することができます。転送方法の要件はこのデータを使用するシステムにより異なり、たとえばモニターリングシステムではデータ損失が許容され、請求システムでは信頼性の高いデータ転送が必要とされます。Telemetry は、両方のシステム種別の要件を満たす方法を提供します。サービスのパブリッシャーコンポーネントを使用して、メッセージバスを介してデータを永続ストレージに保存することや、1 つまたは複数の外部コンシューマーに送信することができます。1 つのチェーンに複数のパブリッシャーを含めることができます。

以下のパブリッシャー種別がサポートされます。

- **Gnocchi (デフォルト)**: Gnocchi パブリッシャーが有効な場合、計測およびリソース情報が時系列データに最適化されたストレージ用の Gnocchi にプッシュされます。Ceilometer は Identity サービスを通じて正確なパスを検出するので、Gnocchi を Identity サービスに登録するようにします。

- **panko**: Ceilometer からのイベントデータを panko に保管することができます。panko は、Red Hat OpenStack Platform のシステムイベントのクエリーを行うための HTTP REST インターフェイスを提供します。データを panko にプッシュするには、パブリッシャーを **direct://?dispatcher=panko** に設定します。

1.3.1.1.1. パブリッシャーパラメーターの設定

Telemetry サービス内の各データポイントに、複数のパブリッシャーを設定することができます。これにより、同じ技術メーターまたはイベントを複数のターゲットに複数回公開することができます。この場合、それぞれ異なる転送方法を使用することが可能です。

手順

1. パブリッシャーパラメーターおよびデフォルト値を記述するための YAML ファイルを作成します (例: **ceilometer-publisher.yaml**)。以下のパラメーターを **parameter_defaults** に追加します。

```
parameter_defaults:

ManagePipeline: true
ManageEventPipeline: true
EventPipelinePublishers:
- gnocchi://?archive_policy=high
PipelinePublishers:
- gnocchi://?archive_policy=high
```

2. カスタマイズしたオーバークラウドをデプロイします。オーバークラウドをデプロイするには、2 とおりの方法があります。

   - 変更した YAML ファイルを **openstack overcloud deploy** コマンドに含めて、パブリッシャーを定義します。次の例では、**<environment-files>** をデプロイに含める他の YAML ファイルに置き換えます。

     ```
     $ openstack overcloud  deploy
     --templates \
     -e /home/custom/ceilometer-publisher.yaml
     -e <environment-files>
     ```

   - **local_modifications.yaml** など、すべてのローカル変更を含める YAML ファイルを作成します。以下の例に示すように、スクリプトを使用してデプロイメントを実行することができます。

     ```
     $ sh deploy.sh:

     #!/bin/bash
     openstack overcloud deploy
     -e <environment-files> \
     -e local_modifications.yaml \
     ....
     ```

関連情報

- パラメーターについての詳細は、**オーバークラウドのパラメーター**の Telemetry パラメーターおよび**オーバークラウドの高度なカスタマイズ**の パラメーター を参照してください。

1.3.2. collectd

パフォーマンスのモニターリングでは、データ収集エージェントを使用してシステム情報を定期的に収集し、その値をさまざまな方法で保管および監視することができます。Red Hat は、データ収集エージェントとして collectd デーモンをサポートします。このデーモンは、データを時系列データベースに保管します。Red Hat がサポートするデータベースの 1 つは Gnocchi と呼ばれます。保管されたこのデータを使用して、システムの監視、パフォーマンスのボトルネックの特定、および将来的なシステム負荷の予測を行うことができます。

関連情報

- Gnocchi についての詳細は、「Gnocchi を使用したストレージ」を参照してください。
- collectd についての詳細は、「collectd のインストール」を参照してください。

1.4. GNOCCHI を使用したストレージ

Gnocchi はオープンソースの時系列データベースです。大量のメトリックを保管し、運用者およびユーザーにメトリックおよびリソースへのアクセスを提供します。Gnocchi は、アーカイブポリシーを使用して処理する集約および保持する集約値の数を定義します。インデクサードライバーは、すべてのリソース、アーカイブポリシー、およびメトリックのインデックスを保管します。

1.4.1. アーカイブポリシー: 時系列データベースへの短期および長期両データの保管

アーカイブポリシーにより、処理する集約および保持する集約値の数を定義します。Gnocchi は、最小値、最大値、平均値、N 番目パーセンタイル、標準偏差などのさまざまな集約メソッドをサポートします。これらの集約は粒度と呼ばれる期間にわたって処理され、特定のタイムスパンの間保持されます。

アーカイブポリシーは、メトリックの集約方法および保管期間を定義します。それぞれのアーカイブポリシーは、タイムスパンにおけるポイント数として定義されます。

たとえば、アーカイブポリシーで 1 秒の粒度および 10 ポイントのポリシーを定義すると、時系列アーカイブは最大 10 秒間保持し、それぞれが 1 秒間の集約を表します。つまり、時系列は最大で、より新しいポイントと古いポイント間の 10 秒間のデータを保持します。

アーカイブポリシーは、使用する集約メソッドも定義します。デフォルトのメソッドはパラメーター **default_aggregation_methods** で設定し、そのデフォルト値は mean、min、max、sum、std、count に設定されています。したがって、ユースケースによってアーカイブポリシーおよび粒度は異なります。

関連情報

- アーカイブポリシーについての詳細は、アーカイブポリシーのプランニングおよび管理を参照してください。

1.4.2. インデクサードライバー

インデクサーは、すべてのリソース、アーカイブポリシー、およびメトリックのインデックス、ならびにそれらの定義、種別、および属性を保管するロールを担います。また、リソースとメトリックをリンクさせる機能も果たします。Red Hat OpenStack Platform director は、デフォルトでインデクサードライバーをインストールします。Gnocchi が処理するすべてのリソースおよびメトリックをインデックス化するデータベースが必要です。サポートされるドライバーは MySQL です。

1.4.3. Gnocchi Metric-as-a-Service に関する用語

Metric-as-a-Service 機能で一般的に使用される用語の定義を以下の表にまとめます。

表1.1 Metric-as-a-Service に関する用語

用語	定義
集約メソッド	複数の計測値から 1 つの集約値を生成するのに使用される関数。min 集約メソッドであれば、さまざまな計測値を、特定期間内の全計測値の最小値に集約します。
集約値 (Aggregate)	アーカイブポリシーに従って複数の計測値から生成されたデータポイントタプル。集約値はタイムスタンプおよび値で設定されます。

用語	定義
アーカイブポリシー	メトリックに割り当てられた集約値の保管ポリシー。アーカイブポリシーにより、集約値がメトリックに保持される期間および集約値の生成方法 (集約メソッド) が決定されます。
粒度 (Granularity)	メトリックの集約時系列における 2 つの集約値の時間間隔
計測値 (Measure)	API によって時系列データベースに送信される受信データポイントタプル。計測値はタイムスタンプおよび値で設定されます。
メトリック	UUID で識別される集約値を保管するエンティティー。名前を使用して、メトリックをリソースに割り当てることができます。メトリックがその集約値をどのように保管するかは、メトリックが関連付けられたアーカイブポリシーで定義されます。
リソース	メトリックを関連付ける、インフラストラクチャー内の任意の項目を表すエンティティー。リソースは一意の ID で識別され、属性を含めることができます。
時系列 (Time series)	集約値を時刻順に並べた一覧
タイムスパン	メトリックがその集約値を保持する期間。アーカイブポリシーを定義する際に使用されます。

1.5. メトリックデータの表示

以下のツールを使用して、メトリックデータを表示および解析することができます。

- **Grafana**: オープンソースのメトリック分析および可視化スイート。Grafana は、インフラストラクチャーおよびアプリケーションを解析する際、時系列データの可視化用に最も一般的に使用されているツールです。
- **Red Hat CloudForms**: インフラストラクチャーの管理プラットフォーム。IT 部門はこれを使用して、プロビジョニングおよび管理に関するユーザーのセルフサービス機能をコントロールし、仮想マシンおよびプライベートクラウド全体にわたるコンプライアンスを確保します。

関連情報

- Grafana についての詳細は、「データ表示のための Grafana の使用および接続」を参照してください。
- Red Hat Cloudforms についての詳細は、製品ドキュメント を参照してください。

1.5.1. データ表示のための Grafana の使用および接続

Grafana 等のサードパーティーソフトウェアを使用して、収集および保管したメトリックをグラフィカルに表示することができます。

Grafana は、オープンソースのメトリック解析、モニターリング、および可視化スイートです。Grafana をインストールおよび設定するには、公式の Grafana documentation を参照してください。

第2章 運用データ計測のプランニング

監視するリソースは、ビジネスの要件によって異なります。Ceilometer または collectd を使用して、リソースを監視することができます。

- collectd による計測の詳細は、「collectd による計測」を参照してください。
- Ceilometer による計測の詳細は、「Ceilometer による計測」を参照してください。

2.1. CEILOMETER による計測

Ceilometer の計測値の完全なリストは、Measurements を参照してください。

2.2. COLLECTD による計測

以下の計測が、最も一般的に使用される collectd メトリックです。

- disk
- interface
- load
- memory
- processes
- tcpconns

計測の完全なリストは、Collectd Metrics and Events を参照してください。

2.3. GNOCCHI および CEILOMETER パフォーマンスの監視

openstack metric コマンドを使用して、デプロイメントのアーカイブポリシー、ベンチマーク、メジャー、メトリック、およびリソースを管理できます。

手順

- コマンドラインで **openstack metric status** と入力して、デプロイでの Gnocchi のインストールを監視し、測定のステータスを確認します。

```
(overcloud) [stack@undercloud-0 ~]$ openstack metric status
+-----------------------------------------------------+-------+
| Field                                               | Value |
+-----------------------------------------------------+-------+
| storage/number of metric having measures to process | 0     |
| storage/total number of measures to process         | 0     |
+-----------------------------------------------------+-------+
```

2.4. データストレージのプランニング

Gnocchi は、データポイントのコレクションを保管します。この場合、それぞれのデータポイントが集約値です。ストレージの形式は、異なる技術を使用して圧縮されます。したがって、時系列データベースのサイズを計算する場合、ワーストケースのシナリオに基づいてサイズを見積もります。

手順

1. データポイントの数を計算します。
   ポイント数 = タイムスパン / 粒度

   たとえば、1 分間の解像度で 1 年分のデータを保持する場合は、以下の式を使用します。

   データポイント数 = (365 日 X 24 時間 X 60 分) / 1 分 = 525600

2. 時系列データベースのサイズを計算します。
   サイズ (バイト単位) = データポイント数 X 8 バイト

   この式を例に当てはめると、結果は 4.1 MB になります。

   サイズ (バイト単位) = 525600 ポイント X 8 バイト = 4204800 バイト = 4.1 MB

   この値は、単一の集約時系列データベースの推定ストレージ要件です。アーカイブポリシーで複数の集約メソッド (min、max、mean、sum、std、および count) が使用される場合は、使用する集約メソッドの数をこの値に掛けます。

関連資料

- 詳細は、「アーカイブポリシー: 時系列データベースへの短期および長期両データの保管」を参照してください。

2.5. アーカイブポリシーのプランニングおよび管理

アーカイブポリシーは、メトリックの集約方法および時系列データベースへの保管期間を定義します。アーカイブポリシーは、タイムスパンにおけるポイント数として定義されます。

アーカイブポリシーで 1 秒の粒度および 10 ポイントのポリシーを定義すると、時系列アーカイブは最大 10 秒間保持し、それぞれが 1 秒間の集約を表します。つまり、時系列は最大で、より新しいポイントと古いポイント間の 10 秒間のデータを保持します。アーカイブポリシーは、使用する集約メソッドも定義します。デフォルトはパラメーター **default_aggregation_methods** に設定され、デフォルト値は **mean**、**min**、**max**、**sum**、**std**、**count** に設定されます。したがって、ユースケースによってアーカイブポリシーおよび粒度は異なる場合があります。

アーカイブポリシーをプランニングするには、以下の概念に精通している必要があります。

- メトリック:詳細は、「メトリック」を参照してください。
- 計測値:詳細は、「カスタム計測値の作成」を参照してください。
- 集約:詳細は、「時系列集約値のサイズの計算」を参照してください。
- metricd ワーカー:詳細は、「metricd ワーカー」を参照してください。

アーカイブポリシーを作成および管理するには、以下のタスクを実行します。

1. アーカイブポリシーを作成する。詳細は、「アーカイブポリシーの作成」を参照してください。

2. アーカイブポリシーを管理する。詳細は、「アーカイブポリシーの管理」を参照してください。

3. アーカイブポリシールールを作成する。詳細は、「アーカイブポリシールールの作成」を参照してください。

2.5.1. メトリック

Gnocchi は、**メトリック** と呼ばれるオブジェクトタイプを提供します。メトリックとは、サーバーの CPU 使用状況、部屋の温度、ネットワークインターフェイスによって送信されるバイト数など、計測することのできる任意の項目を指します。メトリックには以下の属性が含まれます。

- 識別用の UUID
- 名前
- 計測値を保管および集約するのに使用されるアーカイブポリシー

関連情報

- 用語の定義については、Gnocchi Metric-as-a-Service に関する用語 を参照してください。

2.5.1.1. メトリックの作成

手順

1. リソースを作成します。<resource_name> をリソースの名前に置き換えます。

```
$ openstack metric resource create <resource_name>
```

2. メトリックを作成します。<resource_name> をリソースの名前に、<metric_name> をメトリックの名前に、それぞれ置き換えます。

```
$ openstack metric metric create -r <resource_name> <metric_name>
```

メトリックを作成する場合、アーカイブポリシーの属性は固定され、変更することはできません。**archive_policy** エンドポイントを使用して、アーカイブポリシーの定義を変更することができます。

2.5.2. カスタム計測値の作成

計測値とは、API が Gnocchi に送信する受信データポイントタプルを指します。計測値はタイムスタンプおよび値で設定されます。独自のカスタム計測値を作成することができます。

手順

- カスタム計測値を作成します。

```
$ openstack metric measures add -m <MEASURE1> -m <MEASURE2> .. -r
<RESOURCE_NAME> <METRIC_NAME>
```

2.5.3. デフォルトのアーカイブポリシー

デフォルトでは、Gnocchi には以下のアーカイブポリシーがあります。

- low
  - 5 分間の粒度で 30 日間のタイムスパン
  - 使用される集計方法: **default_aggregation_methods**
  - メトリック 1 つあたりの最大推定サイズ: 406 KiB
- medium
  - 1 分間の粒度で 7 日間のタイムスパン
  - 1 時間の粒度で 365 日間のタイムスパン
  - 使用される集計方法: **default_aggregation_methods**
  - メトリック 1 つあたりの最大推定サイズ: 887 KiB
- high
  - 1 秒間の粒度で 1 時間のタイムスパン
  - 1 分間の粒度で 1 週間のタイムスパン
  - 1 時間の粒度で 1 年間のタイムスパン
  - 使用される集計方法: **default_aggregation_methods**
  - メトリック 1 つあたりの最大推定サイズ: 1 057 KiB
- ブール
  - 1 秒間の粒度で 1 年間のタイムスパン
  - 使用される集約メソッド: last
  - メトリック 1 つあたりの最大サイズ (楽観的): 1 539 KiB
  - メトリック 1 つあたりの最大サイズ (ワーストケース): 277 172 KiB

2.5.4. 時系列集約値のサイズの計算

Gnocchi は、データポイントのコレクションを保管します。この場合、それぞれのポイントが集約値です。ストレージの形式は、異なる技術を使用して圧縮されます。したがって、時系列のサイズの計算は、以下の例に示すようにワーストケースのシナリオに基づいて見積もられます。

手順

1. 以下の式を使用して、ポイント数を計算します。
   ポイント数 = タイムスパン / 粒度

   たとえば、1 分間の解像度で 1 年分のデータを保持する場合の計算は、以下のようになります。

   ポイント数 = (365 日 X 24 時間 X 60 分) / 1 分

   ポイント数 = 525600

2. ポイントサイズをバイト単位で計算するには、以下の式を使用します。
サイズ (バイト単位) = ポイント数 X 8 バイト

サイズ (バイト単位) = 525600 ポイント X 8 バイト = 4204800 バイト = 4.1 MB

この値は、単一の集約時系列の推定ストレージ要件です。アーカイブポリシーで複数の集約メソッド (min、max、mean、sum、std、および count) が使用される場合は、使用する集約メソッドの数をこの値に掛けます。

2.5.5. metricd ワーカー

metricd デーモンを使用して、計測値の処理、集約値の作成、集約値ストレージへの計測値の保管、およびメトリックの削除を行うことができます。metricd デーモンは、Gnocchi のほとんどの CPU 使用および I/O ジョブを管理します。各メトリックのアーカイブポリシーは、metricd デーモンが実行する速度を決定します。metricd は、受信ストレージに新しい計測値がないか定期的に確認します。各チェック間の遅延を設定するには、**[metricd]metric_processing_delay configuration** オプションを使用できます。

2.5.6. アーカイブポリシーの作成

手順

- アーカイブポリシーを作成します。<archive-policy-name> をポリシーの名前に、<aggregation-method> を集約メソッドに、それぞれ置き換えます。

```
# openstack metric archive policy create <archive-policy-name> --definition <definition> \
--aggregation-method <aggregation-method>
```

注記

<definition> はポリシー定義です。コンマ (,) を使用して、複数の属性を区切ります。コロン (:) を使用して、アーカイブポリシー定義の名前と値を区切ります。

2.5.7. アーカイブポリシーの管理

- アーカイブポリシーを削除するには、以下のコマンドを実行します。

```
openstack metric archive policy delete <archive-policy-name>
```

- すべてのアーカイブポリシーを表示するには、以下のコマンドを実行します。

```
# openstack metric archive policy list
```

- アーカイブポリシーの詳細を表示するには、以下のコマンドを実行します。

```
# openstack metric archive-policy show <archive-policy-name>
```

2.5.8. アーカイブポリシールールの作成

アーカイブポリシールールにより、メトリックとアーカイブポリシー間のマッピングを定義します。これにより、ユーザーはルールを事前に定義し、一致するパターンに基づいてアーカイブポリシーをメトリックに割り当てることができます。

手順

- アーカイブポリシールールを作成します。<rule-name> をルールの名前に、<archive-policy-name> をアーカイブポリシーの名前に、それぞれ置き換えます。

```
# openstack metric archive-policy-rule create <rule-name> /
--archive-policy-name  <archive-policy-name>
```

第3章 運用データ計測ツールのインストールおよび設定

データ収集エージェント collectd および時系列データベース Gnocchi をインストールする必要があります。

3.1. COLLECTD のインストール

collectd をインストール場合、お使いの環境に合わせて複数の collectd プラグインを設定することができます。

手順

1. ファイル **/usr/share/openstack-tripleo-heat-templates/environments/collectd-environment.yaml** をローカルディレクトリーにコピーします。
2. **collectd-environment.yaml** を開き、**CollectdExtraPlugins** の下に必要なプラグインをリストします。**ExtraConfig** セクションにパラメーターを指定することもできます。

```
parameter_defaults:
  CollectdExtraPlugins:
    - disk
    - df
    - virt

  ExtraConfig:
    collectd::plugin::virt::connection: "qemu:///system"
    collectd::plugin::virt::hostname_format: "hostname uuid"
```

   デフォルトでは、collectd には **disk**、**interface**、**load**、**memory**、**processes**、**tcpconns** のプラグインが設定されています。追加のプラグインは、**CollectdExtraPlugins** パラメーターを使用して追加できます。示されているように、**ExtraConfig** オプションを使用して、CollectdExtraPlugins の追加の設定情報を提供することもできます。この例では、**virt** プラグインを追加し、接続文字列とホスト名の形式を設定します。

3. 変更した YAML ファイルを **openstack overcloud deploy** コマンドに含めて、collectd デーモンをすべてのオーバークラウドノードにインストールします。

```
$ openstack overcloud deploy
--templates \/home/templates/environments/collectd.yaml \
-e /path-to-copied/collectd-environment.yaml
```

関連情報

- collectd についての詳細は、「collectd」を参照してください。
- collectd プラグインおよび設定を確認するには、**Service Telemetry Framework 1.0**の collectd plug-ins を参照してください。

3.2. GNOCCHI のインストール

デフォルトでは、アンダークラウドで Gnocchi は有効になっていません。Red Hat では、アンダークラウド上で Telemetry を有効にすることは推奨していません。多くのデータ (リソースの制限によりアンダークラウドが処理できない) および単一障害点が生成されるためです。

デフォルトでは、Telemetry および Gnocchi はコントローラーノードおよびコンピュートノードにインストールされます。Gnocchi のデフォルトのストレージバックエンドはファイルです。

以下の 2 つの方法のどちらかで、オーバークラウドに Gnocchi をデプロイすることができます。

- 内部的: 詳細は、「Gnocchi の内部デプロイ」を参照してください。
- 外部的: 詳細は、「Gnocchi の外部デプロイ」を参照してください。

3.2.1. Gnocchi の内部デプロイ

デフォルトのデプロイメントは内部です。

手順

- 内部 Gnocchi にメトリックデータを送信するために collectd をデプロイするには、**overcloud deploy** コマンドに **/usr/share/openstack-tripleo-heat-templates/environments/services/collectd.yaml** を追加します。

関連情報

- 詳細は、「collectd のインストール」を参照してください。

3.2.2. Gnocchi の外部デプロイ

手順

1. ローカルディレクトリーにカスタム YAML ファイル (例: **ExternalGnocchi.yaml**) を作成し、次の詳細が含まれていることを確認します。

```
CollectdGnocchiServer: <IPofExternalServer>
CollectdGnocchiUser: admin
CollectdGnocchiAuth: basic
```

2. Gnocchi をデプロイするには、カスタム YAML ファイルを **overcloud deploy** コマンドに追加します。**<existing_overcloud_environment_files>** を既存のデプロイメントに含まれる環境ファイルの一覧に置き換えます。

```
openstack overcloud deploy \
-e <existing_overcloud_environment_files> \
-e /usr/share/openstack-tripleo-heat-templates/environments/services/collectd.yaml \
-e /home/templates/environments/ExternalGnocchi.yaml \
...
```

注記

すべての Gnocchi パラメーターは、YAML ファイル **/usr/share/openstack-tripleo-heat-templates/puppet/services/metrics/collectd.yaml** にあります。

3.2.3. Gnocchi デプロイメントの確認

手順

- 新規リソースおよびメトリックを一覧表示します。

```
$ (overcloud) [stack@undercloud-0 ~]$ openstack metric  metric list |more
+--------------------------------------+---------------------+-----------------------------------------------
-+------+--------------------------------------+
| id                                   | archive_policy/name | name                                          | unit | resource_id
|
+--------------------------------------+---------------------+-----------------------------------------------
-+------+--------------------------------------+
| 001715fe-8ad1-4f21-b0fa-1dee2b1c8201 | low    | interface-eth4@if_packets-rx  | None |
e1357192-e563-5524-b0f0-b30fd11ea8df |
| 003057df-0271-4c59-b793-f885fe09668a | low    | pconns-6000-local@tcp_connections-
LISTEN   | None | 33ac72d9-ab4d-54f1-9fcf-56e02f770b70 |
| 0030a5ba-0e4a-4cce-a48b-4122bfbc4e7a | low  | tcpconns-8004-local@tcp_connections-
CLOSING   | None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 0030e0a7-2cea-42db-860c-fa9ab175c8e1 | low  | tcpconns-25-local@tcp_connections-
CLOSING     | None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 007ec848-f16a-48b8-8706-742a28e2efcf | low    | memcached-
local@memcached_command-get          | None | 8900c37d-501a-565d-b38c-85e5a07a0463 |
| 009d08c7-7483-4e0a-b1bd-0b1451a3b2ab | low | tcpconns-8041-local@tcp_connections-
TIME_WAIT | None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 00a8183c-910e-4849-8d08-9401fbe82029 | low  | tcpconns-11211-local@tcp_connections-
CLOSING  | None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 010af93c-62be-442a-94da-5cb426053fe8 | low    | tcpconns-4567-local@tcp_connections-
FIN_WAIT2 | None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 0113b4f3-786d-4a0a-bb4b-59417d63b60f | low    | tcpconns-38857-local@tcp_connections-
LAST_ACK | None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 0128dac7-b237-4e4c-8d5f-83f6e53771b4 | low    | tcpconns-37871-local@tcp_connections-
SYN_SENT | None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 014ca1f7-565d-4ce7-b35f-52fd916a8b06 | low    | tcpconns-43752-local@tcp_connections-
TIME_WAIT   | None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 0154f901-18c2-4dd9-a593-db26d356611a | low  | tcpconns-1993-local@tcp_connections-
CLOSED    | None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 0158d618-d3ba-45e8-bce8-33411d6f35e3 | low    | tcpconns-111-local@tcp_connections-
CLOSED     | None | e1357192-e563-5524-b0f0-b30fd11ea8df |
| 016fe93f-2794-490e-8057-fd5ab75eb4ec | low      | tcpconns-6001-local@tcp_connections-
SYN_RECV  | None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 01ce3b82-98ad-4f61-88d0-ba46b9862688 | low    | interface-br-tenant@if_dropped-rx
| None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 01d4d186-cf26-4264-87a0-9f5deb8872b5 | low    | tcpconns-8774-local@tcp_connections-
ESTABLISHED  | None | 33ac72d9-ab4d-54f1-9fcf-56e02f770b70 |
| 01f19617-3ef6-43e8-9ad8-8c84b923f13f | low      | tcpconns-43394-
local@tcp_connections-FIN_WAIT2   | None | 33ac72d9-ab4d-54f1-9fcf-56e02f770b70 |
| 020646e9-2d50-4126-9a63-a5f36180cc0b | low    | tcpconns-6000-local@tcp_connections-
CLOSE_WAIT   | None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 02155fd3-0d68-4eec-8cd5-8b158f5f03a4 | low      | tcpconns-6633-
local@tcp_connections-LISTEN    | None | 33ac72d9-ab4d-54f1-9fcf-56e02f770b70 |
| 0236355e-3415-4a72-99f2-38ada7b3db68 | low      | tcpconns-43806-
local@tcp_connections-FIN_WAIT2   | None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 024e89d2-aa77-49c0-ae6e-65a665db01b3 | low      | tcpconns-35357-
local@tcp_connections-FIN_WAIT2   | None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 027adb62-272f-4331-8167-28c3489d3b44 | low      | tcpconns-9292-
local@tcp_connections-LISTEN    | None | 33ac72d9-ab4d-54f1-9fcf-56e02f770b70 |
| 0290d15e-7687-4683-bd25-4c030cad12cf | low      | tcpconns-37378-
local@tcp_connections-CLOSE_WAIT  | None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 02a5383f-b061-4422-9570-bfd3b2532832 | low      | processes@ps_state-zombies
```

```
| None | 33ac72d9-ab4d-54f1-9fcf-56e02f770b70 |
| 02c959d5-8ae2-4d14-a140-189530b4e8f6 | low        | disk-vda2@disk_merged-write
| None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 02d174b5-6783-4db5-bfe7-8d45931aa0b0 | low        | interface-br-tun@if_octets-rx
| None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 02d8c001-90da-4997-bfa5-e5d3afe599fa | low        | tcpconns-25672-
local@tcp_connections-CLOSING  | None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 02d932f0-5745-4694-86d9-84e365789a7d | low        | tcpconns-9292-
local@tcp_connections-CLOSING   | None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 02e1a5e2-194d-4e49-8b36-a843b5dbdc3d | low        | tcpconns-45228-
local@tcp_connections-CLOSE_WAIT  | None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 02e5dcec-f5b7-41e9-9f3c-714ade502836 | low        | load@load-5min                    |
None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 02fe05ed-e4a5-4a18-ad6c-64f8787225c9 | low        | tcpconns-8774-
local@tcp_connections-SYN_SENT  | None | 33ac72d9-ab4d-54f1-9fcf-56e02f770b70 |
| 03129089-7bbd-47f3-ab14-9cd583d775ae | low        | tcpconns-6379-
local@tcp_connections-LAST_ACK  | None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 032b1771-93c4-4051-bbec-9115c9d329c4 | low        | tcpconns-8042-
local@tcp_connections-TIME_WAIT | None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 033516d5-ac15-4767-b005-cff52276badd | low        | tcpconns-22-local@tcp_connections-
CLOSED      | None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 03589183-efa7-43ed-aea6-9d3c8accf97a | low        | interface-br-tun@if_errors-tx
| None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 0360077f-729d-4591-a9fc-d96fbb7e0326 | low        | tcpconns-6080-
local@tcp_connections-CLOSING   | None | 33ac72d9-ab4d-54f1-9fcf-56e02f770b70 |
| 0365f5a1-4a98-435e-a809-c8706b0671bd | low        | tcpconns-34296-
local@tcp_connections-LAST_ACK | None | 33ac72d9-ab4d-54f1-9fcf-56e02f770b70 |
| 039c57b7-ae5b-4f13-846e-8e5f184cdc4d | low        | tcpconns-111-
local@tcp_connections-CLOSE_WAIT | None | 926d87fe-b388-501d-afa6-dc6af55ce4ad |
| 04169046-80c4-478e-b263-b9529b5ca739 | low        | interface-br-tenant@if_packets-tx
| None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 0432314d-eb8b-4bf9-ab8f-5ecc5b30592d | low        | tcpconns-37415-
local@tcp_connections-CLOSE_WAIT  | None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 043d9d6e-5db0-40e4-83b1-b8120506e9ec | low        | tcpconns-6379-
local@tcp_connections-ESTABLISHED  | None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 04639fe9-6dc1-46eb-8b86-9b39e8fd782d | low        | tcpconns-35357-
local@tcp_connections-SYN_RECV | None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 04871c86-b176-45ed-9f37-bb6fae27e930 | low        | tcpconns-8042-
local@tcp_connections-CLOSE_WAIT   | None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 049c50c4-bf5e-4098-988a-fb867649ee17 | low        | tcpconns-11211-
local@tcp_connections-SYN_SENT | None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 04e37efb-fc67-418e-b53b-6b8055967a2a | low        | tcpconns-8004-
local@tcp_connections-CLOSING   | None | 33ac72d9-ab4d-54f1-9fcf-56e02f770b70 |
| 04ef0d0f-db22-4501-ae4a-4bc9ee719075 | low        | tcpconns-9200-
local@tcp_connections-SYN_RECV  | None | 33ac72d9-ab4d-54f1-9fcf-56e02f770b70 |
```

第4章 運用データ計測の管理

4.1. デプロイメントに基づく環境変数の変更

手順

1. **/usr/share/openstack-tripleo-heat-templates/environments/gnocchi-environment.yaml** ファイルをホームディレクトリーにコピーします。

2. 実際の環境に合わせてパラメーターを変更します。YAML ファイルの以下のプライマリーパラメーターを変更することができます。

   - GnocchiIndexerBackend: **mysql** など、使用するデータベースインデクサーバックエンド。https://github.com/openstack/tripleo-heat-templates/blob/stable/queens/puppet/services/gnocchi-base.yaml#L33 を参照してください。

   - GnocchiBackend: 一時ストレージの種別。値は **rbd**、**swift**、または **file** (ceph) です。詳細は、https://github.com/openstack/tripleo-heat-templates/blob/stable/queens/environments/storage-environment.yaml#L29-L30 を参照してください。

   - NumberOfStorageSacks: ストレージサックの数。詳細は、「サックの数」 を参照してください。

3. **gnocchi-environment.yaml** を **overcloud deploy** コマンドに追加し、環境に関連するその他の環境ファイルをすべて追加してデプロイします。**<existing_overcloud_environment_files>** を既存のデプロイメントの一部である環境ファイルのリストに置き換えます。

```
$ openstack overcloud deploy \
<existing_overcloud_environment_files> \
-e ~gnocchi-environment.yaml \
...
```

4.1.1. metricd ワーカーの実行

デフォルトでは、**gnocchi-metricd** デーモンは、メトリック集計を計算するときに CPU 使用率を最大化するために CPU パワーを拡張します。

手順

- **openstack metric status** コマンドを使用して、HTTP API にクエリーを実行し、メトリック処理のステータスを取得します。

```
# openstack metric status
```

  コマンド出力には、**gnocchi-metrid** デーモンの処理バックログが表示されます。このバックログが継続的に増加しないかぎり、**gnocchi-metricd** は収集されるメトリックの量に対処できることを意味します。処理するメジャーの数が継続的に増加している場合は、**gnocchi-metricd** デーモンの数を増やします。任意の数のサーバー上で、metricd デーモンをいくつでも実行することができます。

4.1.2. サックの数

Gnocchi の受信メトリックデータは異なる Sack にプッシュされ、各 Sack は処理のために 1 つ以上の **gnocchi-metricd** デーモンに割り当てられます。サックの数は、システムがキャプチャーするアクティブなメトリックの数により異なります。

Red Hat は、Sack の数がアクティブな **gnocchi-metricd** ワーカーの総数よりも多いことを推奨しています。

4.1.3. サックサイズの変更

当初の予測よりも多くのメトリックを収集する場合は、サックのサイズを変更することができます。

Gnocchi にプッシュされる計測データは、分散性を向上させるためにサックに分割されます。受信メトリックは特定の Sack にプッシュされ、各 Sack は処理のために 1 つ以上の **gnocchi-metricd** デーモンに割り当てられます。サックの数を設定するには、システムがキャプチャーするアクティブなメトリックの数を使用します。Sack の数は、アクティブな **gnocchi-metricd** ワーカーの総数よりも多くする必要があります。

手順

- 設定する適切なサック数を決定するには、以下の式を使用します。
  サック数 = **アクティブな** メトリックの数 / 300

注記

推定されるメトリック数が絶対最大値である場合、値を 500 で割ります。アクティブなメトリックの数の推定が控えめで増加することが予想される場合は、増加に対応できるように値を 100 で割ります。

4.2. 時系列データベースサービスの監視

HTTP API の **/v1/status** エンドポイントは、処理するメジャーの数 (メジャーバックログ) などの監視可能な情報を返します。正常なシステムの条件を以下に示します。

- HTTP サーバーと **gnocchi-metricd** が実行されています。
- HTTP サーバーと **gnocchi-metricd** がログファイルにエラーメッセージを書き込んでいません。

手順

- 時系列データベースのステータスを表示します。

```
# openstack metric status
```

この出力に、時系列データベースのステータスと共に処理するメトリックの数が表示されます。この値は小さいほど望ましく、0 に近いことが理想です。

4.3. 時系列データベースのバックアップおよびリストア

問題のある状況から回復できるようにするには、インデックスおよびストレージをバックアップします。PostgreSQL または MySQL を使用してデータベースのダンプを作成し、Ceph、Swift、またはファイルシステムを使用してデータストレージのスナップショットまたはコピーを作成する必要があります。

手順

1. インデックスおよびストレージのバックアップを復元します。
2. 必要に応じて Gnocchi を再インストールします。
3. Gnocchi を再起動します。

4.4. 計測値の表示

特定リソースの計測値の一覧を表示することができます。

手順

1. **metric measures** コマンドを使用します。

```
# openstack metric measures show --resource-id UUID <METER_NAME>
```

2. タイムスタンプの範囲内の特定リソースの計測値を一覧表示します。

```
# openstack metric measures show --aggregation mean --start <START_TIME> --stop
<STOP_TIME> --resource-id UUID <METER_NAME>
```

タイムスタンプの変数 <START_TIME> および <END_TIME> の形式は、iso-dateThh:mm:ss です。

4.5. リソース種別の管理

リソース種別を作成、表示、および削除することができます。デフォルトのリソース種別は汎用的なものですが、属性を自由に追加して独自のリソース種別を作成することができます。

手順

1. 新しいリソース種別を作成します。

```
$ openstack metric 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. リソース種別の設定を確認します。

```
$ openstack metric resource-typegnocchi resource-type show testResource01
+--------------------------------------------------------------------------+
| Field          | Value                                                   |
+--------------------------------------------------------------------------+
| attributes/bla | max_length=255, min_length=123, required=True, type=string |
| name           | testResource01                                          |
| state          | active                                                  |
+--------------------------------------------------------------------------+
```

3. リソース種別を削除します。

```
$ openstack metric resource-type delete testResource01
```

注記

リソースが使用中のリソース種別を削除することはできません。

4.6. クラウドの使用状況に関する計測値の表示

手順

- 各プロジェクトの全インスタンスの平均メモリー使用量を表示します。

```
openstack metrics measures aggregation --resource-type instance --groupby project_id -m
“memoryView L3” --resource-id UUID
```

4.6.1. L3 キャッシュモニターリングの有効化

Intel ハードウェアと **libvirt** のバージョンが Cache Monitoring Technology (CMT) をサポートしている場合、**cpu_l3_cache** メーターを使用して、インスタンスが使用する L3 キャッシュの量を監視できます。

L3 キャッシュを監視するには、以下のパラメーターおよびファイルが必要です。

- **LibvirtEnabledPerfEvents** パラメーターの **cmt**。
- **gnocchi_resources.yaml** ファイルの **cpu_l3_cache**。
- Ceilometer **polling.yaml** ファイルの **cpu_l3_cache**。

手順

1. Telemetry 用の YAML ファイル (**ceilometer-environment.yaml** など) を作成します。
2. **ceilometer-environment.yaml** ファイルで、**cmt** を **LibvirtEnabledPerfEvents** パラメーターに追加します。詳細については、**/usr/share/openstack-triple-heat-templates/puppet/services/nova_libvirt.yaml** を参照してください。
3. この YAML ファイルを使用してオーバークラウドをデプロイします。**<existing_overcloud_environment_files>** を既存のデプロイメントの一部である環境ファイルのリストに置き換えます。

```
#!/bin/bash

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

4. コンピュートノードの Gnocchi で **cpu_l3_cache** が有効になっていることを確認します。

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

注記

コンテナーイメージの設定を変更しても、リブート後は維持されません。

5. このコンピュートノードでゲストインスタンスを起動したら、**CMT** メトリックを監視します。

```
(overcloud) [stack@undercloud-0 ~]$ openstack metric 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 |
+---------------------------+-------------+-----------+
```

4.7. GNOCCHI のアップグレード

デフォルトでは、Red Hat OpenStack Platform director を使用してデプロイメントをアップグレードすると、Gnocchi がアップグレードされます。デプロイメントのアップグレードに関する情報は、Red Hat OpenStack Platform のアップグレード を参照してください。Red Hat OpenStack Platform 10 を使用していて、Red Hat OpenStack Platform 13 にアップグレードする場合は、Fast Forward Upgrade を参照してください。

第5章 アラームの管理

aodh と呼ばれるアラームサービスを使用して、アクションをトリガーすることができます。アクションのトリガーは、Ceilometer または Gnocchi の収集するメトリックデータまたはイベントデータに対して定義されたルールに基づきます。

5.1. 既存のアラームの表示

手順

1. 既存の Telemetry アラームを一覧表示します。

```
# openstack 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    |
+--------------------------------------+----------------------------------------+-------------------------
--+------------------+----------+---------+
```

2. リソースに割り当てられたメーターを一覧表示するには、リソースの UUID を指定します。以下に例を示します。

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

5.2. アラームの作成

aodh を使用して、しきい値に達した時にアクティブになるアラームを作成することができます。以下の例では、個々のインスタンスの平均 CPU 使用率が 80% を超えると、アラームがアクティブになりログエントリーが追加されます。

手順

1. アラームを作成し、クエリーを使用してモニターリングするインスタンスの特定の ID (94619081-abf5-4f1f-81c7-9cedaa872403) を分離します。

```
# openstack 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                      |
+--------------------------+-------------------------------------------------------+
```

2. 既存のアラームのしきい値を編集するには、aodh の alarm update コマンドを使用します。たとえば、アラームのしきい値を 75% に増やすには、以下のコマンドを使用します。

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

5.3. アラームの無効化

手順

- アラームを無効にするには、以下のコマンドを入力します。

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

5.4. アラームの削除

手順

- アラームを削除するには、以下のコマンドを入力します。

```
# openstack alarm delete --name cpu_usage_high
```

5.5. 例: インスタンスのディスク動作の監視

aodh アラームを使用して、特定のプロジェクトに含まれるすべてのインスタンスの漸増するディスク動作を監視する方法を、以下の例で説明します。

手順

1. 既存のプロジェクトを確認し、監視するプロジェクトの適切な UUID を選択します。以下の例では、**admin** テナントを使用します。

```
$ openstack project list
```

```
+----------------------------------+----------+
| ID                               | Name     |
+----------------------------------+----------+
| 745d33000ac74d30a77539f8920555e7 | admin    |
| 983739bb834a42ddb48124a38def8538 | services |
| be9e767afd4c4b7ead1417c6dfedde2b | demo     |
+----------------------------------+----------+
```

2. プロジェクトの UUID を使用して、**admin** テナント内のインスタンスが生成するすべての読み取りリクエストの **sum()** を解析するアラームを作成します。**--query** パラメーターを使用して、クエリーをさらに絞り込むことができます。

```
# openstack 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                            |
+---------------------------+-------------------------------------------------------------+
```

5.6. 例: CPU 使用率の監視

インスタンスのパフォーマンスを監視するには、Gnocchi データベースを調べ、メモリーや CPU の使用状況などの監視可能なメトリックを特定します。

手順

1. インスタンス UUID を指定して **openstack metric resource show** コマンドを入力し、監視できるメトリックを特定します。

```
$ openstack metric 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-daee-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                                          |
+-----------------------+---------------------------------------------------------------------------+
```

この結果のメトリック値には、aodh アラームを使用して監視できるコンポーネント (**cpu_util** など) がリストされています。

2. CPU 使用率を監視するには、**cpu_util** メトリックを使用します。

```
$ openstack metric show --resource-id 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: std、count、min、max、sum、average の値を計算する際の集約間隔を定義します。

3. aodh を使用して、**cpu_util** をクエリーする監視タスクを作成します。このタスクは、指定した設定に基づいてイベントをトリガーします。たとえば、インスタンスの CPU 使用率が上昇し一定期間 80% を超える場合にログエントリーを生成するには、以下のコマンドを使用します。

```
$ openstack alarm create \
  --project-id 3cee262b907b4040b26b678d7180566b \
  --name high-cpu \
  --type gnocchi_resources_threshold \
```

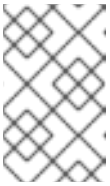
```
  --description 'High CPU usage' \
  --metric cpu_util \
  --threshold 80.0 \
  --comparison-operator ge \
  --aggregation-method mean \
  --granularity 300 \
  --evaluation-periods 1 \
  --alarm-action 'log://' \
  --ok-action 'log://' \
  --resource-type instance \
  --resource-id d71cdf9a-51dc-4bba-8170-9cd95edd3f66
+---------------------------+--------------------------------------+
| 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: ge 演算子は、CPU 使用率が 80% またはそれを超えた場合にアラームがトリガーされることを定義します。

- granularity: メトリックにはアーカイブポリシーが関連付けられます。ポリシーには、さまざまな粒度を設定することができます。たとえば、5 分間の粒度で 1 時間、および 1 時間の粒度で 1 カ月粒度の値は、アーカイブポリシーで指定された期間と一致する必要があります。

- evaluation-periods: アラームがトリガーされる前に満たさなければならない粒度期間の数。たとえば、この値を 2 に設定すると、アラームがトリガーされる前に、2 つのポーリング期間において CPU の使用率が 80% を超える必要があります。

- [u'log://']: **alarm_actions** または **ok_actions** を **[u'log://']** に設定した場合、イベント (アラームのトリガーまたは通常状態への復帰) が aodh のログファイルに記録されます。



注記

アラームがトリガーされた時 (alarm_actions) や通常の状態に復帰した時 (ok_actions) に実行するアクションを、自由に定義することができます (例: Webhook URL)。

5.7. アラーム履歴の表示

アラーム履歴にクエリーを行い、アラームがトリガーされているかどうかを確認することができます。

手順

- **openstack alarm-history show** コマンドを使用します。

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

第6章 ログ

Red Hat OpenStack Platform (RHOSP) は、情報メッセージを特定のログファイルに書き込みます。これらのメッセージを、トラブルシューティングおよびシステムイベントの監視に使用することができます。

注記

個々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。**sosreport** ユーティリティーは、必要なログを自動的に収集します。

6.1. OPENSTACK サービスのログファイルの場所

それぞれの OpenStack コンポーネントには、実行中のサービス固有のファイルが含まれる個別のログディレクトリーがあります。

6.1.1. Bare Metal Provisioning (ironic) のログファイル

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

6.1.2. Block Storage (cinder) のログファイル

サービス	サービス名	ログのパス
Block Storage API	openstack-cinder-api.service	/var/log/containers/cinder-api.log
Block Storage Backup	openstack-cinder-backup.service	/var/log/containers/cinder/backup.log
情報メッセージ	cinder-manage コマンド	/var/log/containers/cinder/cinder-manage.log
Block Storage Scheduler	openstack-cinder-scheduler.service	/var/log/containers/cinder/scheduler.log
Block Storage Volume	openstack-cinder-volume.service	/var/log/containers/cinder/volume.log

6.1.3. Compute (nova) のログファイル

サービス	サービス名	ログのパス
OpenStack Compute API サービス	openstack-nova-api.service	/var/log/containers/nova/nova-api.log
OpenStack Compute 証明書サーバー	openstack-nova-cert.service	/var/log/containers/nova/nova-cert.log
OpenStack Compute サービス	openstack-nova-compute.service	/var/log/containers/nova/nova-compute.log
OpenStack Compute Conductor サービス	openstack-nova-conductor.service	/var/log/containers/nova/nova-conductor.log
OpenStack Compute VNC コンソール認証サーバー	openstack-nova-consoleauth.service	/var/log/containers/nova/nova-consoleauth.log
情報メッセージ	nova-manage コマンド	/var/log/containers/nova/nova-manage.log
OpenStack Compute NoVNC Proxy サービス	openstack-nova-novncproxy.service	/var/log/containers/nova/nova-novncproxy.log
OpenStack Compute Scheduler サービス	openstack-nova-scheduler.service	/var/log/containers/nova/nova-scheduler.log

6.1.4. Dashboard (horizon) のログファイル

サービス	サービス名	ログのパス
特定ユーザーの対話のログ	Dashboard インターフェイス	/var/log/containers/horizon/horizon.log

Apache HTTP サーバーは、Dashboard Web インターフェイス用にさまざまな追加ログファイルを使用します。このログファイルには、Web ブラウザーまたは keystone および nova 等のコマンドラインクライアントを使用してアクセスすることができます。以下のログファイルは、Dashboard の使用状況のトラッキングおよび障害の診断に役立ちます。

目的	ログのパス
すべての処理された HTTP リクエスト	/var/log/containers/httpd/horizon_access.log
HTTP エラー	/var/log/containers/httpd/horizon_error.log
管理者ロールの API リクエスト	/var/log/containers/httpd/keystone_wsgi_admin_access.log

目的	ログのパス
管理者ロールの API エラー	/var/log/containers/httpd/keystone_wsgi_admin_error.log
メンバーロールの API リクエスト	/var/log/containers/httpd/keystone_wsgi_main_access.log
メンバーロールの API エラー	/var/log/containers/httpd/keystone_wsgi_main_error.log

注記

/var/log/containers/httpd/default_error.log もあり、同じホストで実行されている他の Web サービスによって報告されたエラーが保存されます。

6.1.5. Data Processing (sahara) のログファイル

サービス	サービス名	ログのパス
Sahara API サーバー	openstack-sahara-all.service openstack-sahara-api.service	/var/log/containers/sahara/sahara-all.log /var/log/containers/messages
Sahara Engine サーバー	openstack-sahara-engine.service	/var/log/containers/messages

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

サービス	サービス名	ログのパス
OpenStack Trove API サービス	openstack-trove-api.service	/var/log/containers/trove/trove-api.log
OpenStack Trove Conductor サービス	openstack-trove-conductor.service	/var/log/containers/trove/trove-conductor.log
OpenStack Trove ゲストエージェントサービス	openstack-trove-guestagent.service	/var/log/containers/trove/logfile.txt
OpenStack Trove タスクマネージャーサービス	openstack-trove-taskmanager.service	/var/log/containers/trove/trove-taskmanager.log

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

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

6.1.8. Image サービス (glance) のログファイル

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

6.1.9. Networking (neutron) のログファイル

サービス	サービス名	ログのパス
OpenStack Neutron DHCP エージェント	neutron-dhcp-agent.service	/var/log/containers/neutron/dhcp-agent.log
OpenStack Networking レイヤー 3 エージェント	neutron-l3-agent.service	/var/log/containers/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/containers/neutron/metadata-agent.log
メタデータ名前空間プロキシー	該当せず	/var/log/containers/neutron/neutron-ns-metadata-proxy-**UUID**.log
Open vSwitch エージェント	neutron-openvswitch-agent.service	/var/log/containers/neutron/openvswitch-agent.log
OpenStack Networking サービス	neutron-server.service	/var/log/containers/neutron/server.log

6.1.10. Object Storage (swift) のログファイル

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、すべての Object Storage ログファイルは、local0、local1、および local2 syslog ファシリティーを使用して **/var/log/containers/swift/swift.log** に保存されます。

Object Storage のログメッセージは、REST API サービスによるものとバックグラウンドデーモンによるものの 2 つのカテゴリーに大別されます。API サービスのメッセージには、一般的な HTTP サーバーと同様に、API リクエストごとに 1 つの行が含まれます。フロントエンド (プロキシー) とバックエンド (アカウント、コンテナー、オブジェクト) サービスの両方がこのメッセージの POST を行います。デーモンメッセージは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人間が判読できる情報が含まれます。ただし、メッセージを生成する Object Storage の部分に関係なく、ソースの ID は常に行の先頭に置かれます。

プロキシーメッセージの例を以下に示します。

```
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の例を以下に示します。

```
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
```

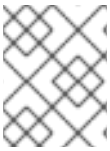
6.1.11. Orchestration (heat) のログファイル

サービス	サービス名	ログのパス
OpenStack Heat API サービス	openstack-heat-api.service	/var/log/containers/heat/heat-api.log
OpenStack Heat エンジンサービス	openstack-heat-engine.service	/var/log/containers/heat/heat-engine.log
Orchestration サービスイベント	該当せず	/var/log/containers/heat/heat-manage.log

6.1.12. Shared File Systems サービス (manila) のログファイル

サービス	サービス名	ログのパス
OpenStack Manila API サーバー	openstack-manila-api.service	/var/log/containers/manila/api.log

サービス	サービス名	ログのパス
OpenStack Manila Scheduler	openstack-manila-scheduler.service	/var/log/containers/manila/scheduler.log
OpenStack Manila ファイル共有サービス	openstack-manila-share.service	/var/log/containers/manila/share.log



注記

Manila Python ライブラリーの一部の情報は、**/var/log/containers/manila/manila-manage.log** に記録することもできます。

6.1.13. Telemetry (ceilometer) のログファイル

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

6.1.14. サポートサービスのログファイル

以下のサービスは OpenStack のコアコンポーネントにより使用され、独自のログディレクトリーおよびファイルを持ちます。

サービス	サービス名	ログのパス
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 (簡易認証およびセキュリティーレイヤーに関するログメッセージ用)
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-vswitchd.log

6.2. 集中ログシステムのアーキテクチャーおよびコンポーネント

モニターリングツールは、クライアントが Red Hat OpenStack Platform (RHOSP) オーバークラウドノードにデプロイされる、クライアント/サーバーモデルを使用します。Fluentd サービスは、クライアント側の集中ロギング (CL) を提供します。すべての RHOSP サービスはログファイルを生成および更新します。これらのログファイルは、アクション、エラー、アラート、およびその他のイベントを記録します。OpenStack のような分散環境では、これらのログを一元的な場所に収集することで、デバッグおよび管理が簡素化されます。集中ロギングにより、OpenStack 環境全体にわたるログを 1 カ所で確認することができます。これらのログは、syslog や監査ログファイル等のオペレーティングシステム、RabbitMQ や MariaDB 等のインフラストラクチャーコンポーネント、および Identity や Compute 等の OpenStack サービスから収集されます。集中ロギングのツールチェーンは、以下のコンポーネントで設定されます。

- ログ収集エージェント (Fluentd)
- ログリレー/トランスフォーマー (Fluentd)
- データストア (ElasticSearch)
- API/プレゼンテーション層 (Kibana)

注記

Red Hat OpenStack Platform director は、集中ロギング向けのサーバー側のコンポーネントをデプロイしません。Red Hat では、ログアグリゲーターとして実行するプラグインを使用する ElasticSearch データベース、Kibana、Fluentd 等のサーバー側のコンポーネントをサポートしていません。集中ロギングのコンポーネントおよびコンポーネント間の対話を以下の図に示します。

注記

青で示した項目は Red Hat のサポート対象コンポーネントです。

図6.1 Red Hat OpenStack Platform の単一 HA デプロイメント

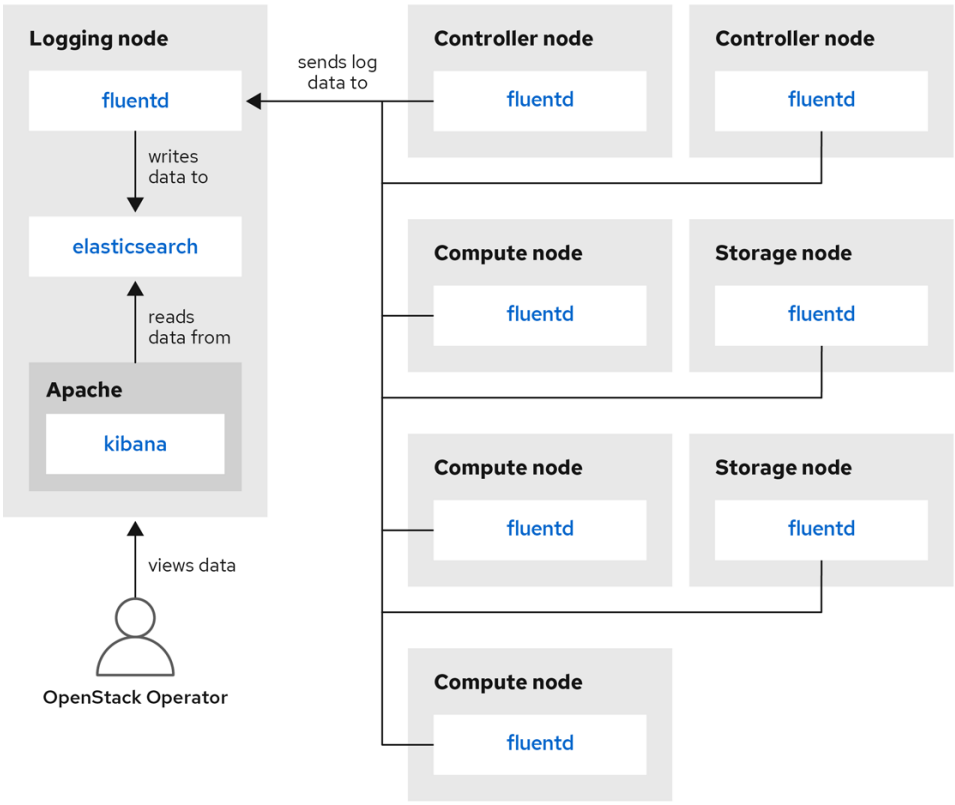
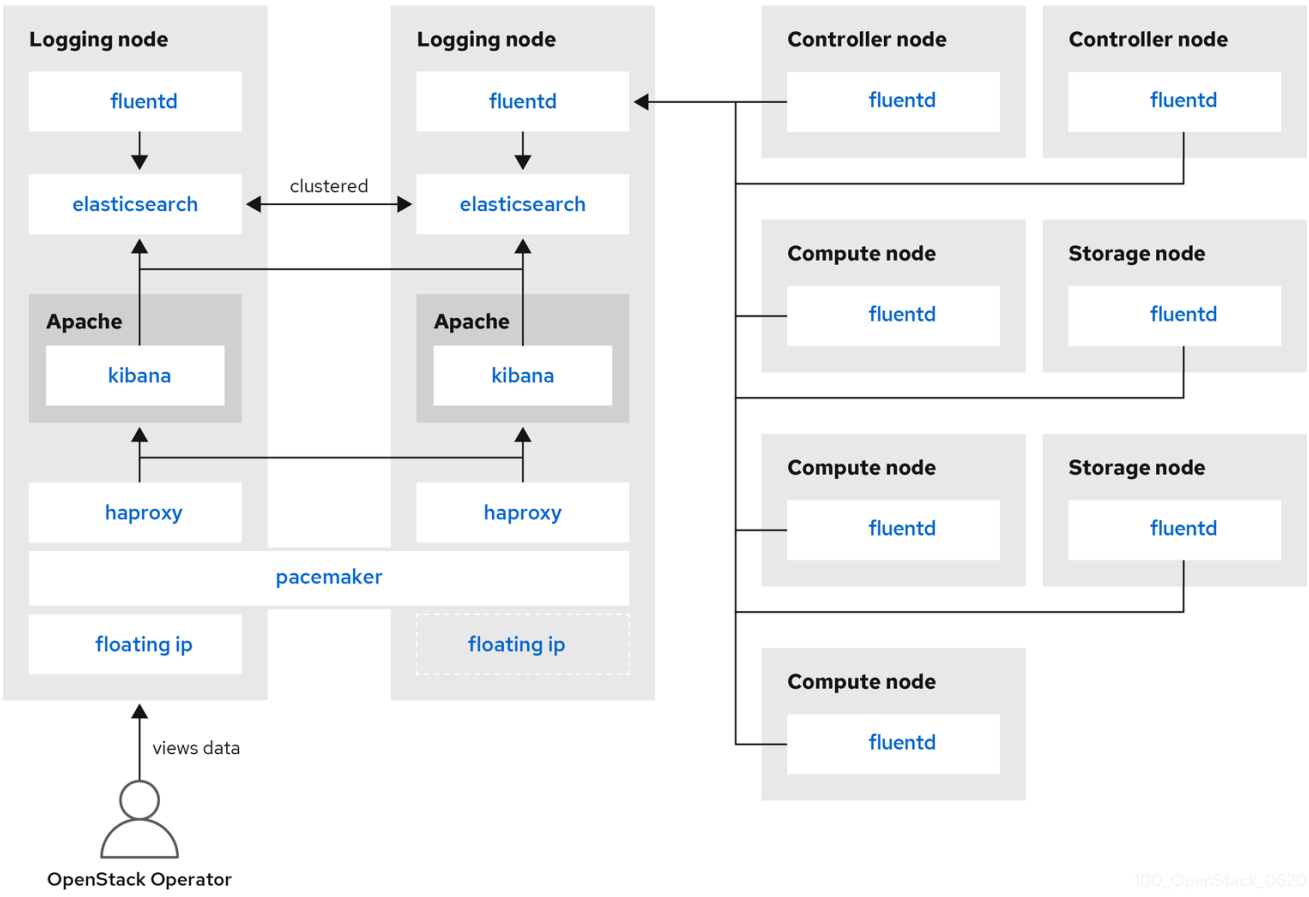


図6.2 Red Hat OpenStack Platform の HA デプロイメント



6.3. ログサービスのインストールの概要

ログ収集エージェント Fluentd はクライアント側のログを収集し、それらのログをサーバー側で実行されている Fluentd インスタンスに送信します。この Fluentd インスタンスは、保管のためにログの記録を Elasticsearch にリダイレクトします。

6.4. すべてのマシンへの FLUENTD のデプロイ

Fluentd はログ収集エージェントで、集中ロギングツールチェーンに含まれます。すべてのマシンに Fluentd をデプロイするには、**logging-environment.yaml** ファイルの **LoggingServers** パラメーターを変更する必要があります。

前提条件

- Elasticsearch および Fluentd リレーがサーバー側にインストールされるようにします。詳細は、クライアント側のインテグレーションと互換性のある opstools-ansible プロジェクト のデプロイメント例を参照してください。

手順

1. **tripleo-heat-templates/environments/logging-environment.yaml** ファイルをホームディレクトリーにコピーします。
2. コピーしたファイルで、環境に合わせて **LoggingServers** パラメーターにエントリーを作成します。次のスニペットは、**LoggingServers** パラメーターの設定例です。

```
parameter_defaults:

Simple configuration

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

3. 変更した環境ファイルを **openstack overcloud deploy** コマンドに含め、環境およびデプロイに関連するその他の環境ファイルを含めます。**<existing_overcloud_environment_files>** を既存のデプロイメントの一部である環境ファイルのリストに置き換えます。

```
$ openstack overcloud deploy \
<existing_overcloud_environment_files> \
-e /home/templates/environments/logging-environment.yaml \
...
```

関連資料

- 詳細は、「設定可能なロギングパラメーター」を参照してください。

6.5. 設定可能なロギングパラメーター

設定可能なロギングパラメーターの説明を以下の表にまとめます。これらのパラメーターは **tripleo-heat-templates/puppet/services/logging/fluentd-config.yaml** ファイルにあります。

パラメーター	説明
LoggingDefaultFormat	ログファイルからのメッセージを解析するのに使用されるデフォルトの形式
LoggingPosFilePath	Fluentd **pos_file** ファイルを配置するディレクトリー。**tail** 入力タイプのファイル位置を追跡するために使用されます。
LoggingDefaultGroups	Fluentd ユーザーを追加するグループを指定します。デフォルトのグループ一覧を変更する場合は、このパラメーターをオーバーライドします。**LoggingExtraGroups** パラメーターを使用して、Fluentd ユーザーを追加のグループに追加します。
LoggingExtraGroups	**LoggingDefaultGroups** パラメーター、および個々のコンポーザブルサービスによって提供されるグループに加えて、Fluentd ユーザーをこれらのグループに追加します。
LoggingDefaultFilters	Fluentd のデフォルトフィルターの一覧。このリストは、**fluentd::config** リソースの **filter** キーにそのまま渡されます。デフォルトのフィルターセットを使用しない場合は、この設定をオーバーライドします。サーバーを追加する場合は、**LoggingExtraFilters** パラメーターを使用します。
LoggingExtraFilters	追加の Fluentd フィルターの一覧。このリストは、**fluentd::config** リソースの **filter** キーにそのまま渡されます。
LoggingUsesSSL	**secure_forward** プラグインを使用してログメッセージを転送するかどうかを示すブール値。
LoggingSSLKey	PEM 形式でエンコードされた Fluentd CA 証明書の鍵。**in_secure_forward** パラメーターは、**LoggingSSLKey** パラメーターの値を使用します。
LoggingSSLCertificate	PEM 形式でエンコードされた Fluentd の SSL CA 証明書
LoggingSSLKeyPassphrase	**LoggingSSLKey** パラメーターのパスフレーズ。**in_secure_forward** パラメーターは、**LoggingSSLKeyPassphrase** パラメーターの値を使用します。
LoggingSharedKey	Fluentd **secure-forward** プラグインの共有シークレット。
LoggingDefaultSources	Fluentd のデフォルトロギングソースの一覧。デフォルトのロギングソースを無効にするには、このパラメーターをオーバーライドします。**LoggingExtraSources** パラメーターを使用して、追加のソース設定を定義します。
LoggingExtraSources	このリストは、**LoggingDefaultSources** パラメーター、およびコンポーザブルサービスによって定義されたログソースと組み合わされます。

6.6. デフォルトのログファイルパスのオーバーライド

デフォルトのコンテナーを変更し、変更箇所にサービスログファイルへのパスが含まれる場合は、デフォルトのログファイルパスも変更する必要があります。すべてのコンポーザブルサービスには **<service_name>LoggingSource** パラメーターがあります。たとえば、nova-compute サービスの場合、パラメーターは **NovaComputeLoggingSource** です。

手順

1. nova-compute サービスのデフォルトパスをオーバーライドするには、設定ファイルの **NovaComputeLoggingSource** パラメーターにパスを追加します。

```
NovaComputeLoggingSource:
    tag: openstack.nova.compute
    path: /some/other/path/nova-compute.log
```

tag および path 属性は、**<service_name>LoggingSource** パラメーターの必須要素です。それぞれのサービスで tag および path が定義され、残りの値はデフォルトで派生されます。

2. 特定のサービスの形式を変更することができます。これは Fluentd 設定に直接渡します。**LoggingDefaultFormat** パラメーターのデフォルト形式は、/(?<time>\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}.\d+) (?<pid>\d+) (?<priority>\S+) (?<message>.*)$/ です。以下の構文を使用します。

```
<service_name>LoggingSource:
   tag: <service_name>.tag
   path: <service_name>.path
   format: <service_name>.format
```

より複雑な変換の例を以下のスニペットに示します。

```
ServiceLoggingSource:
   tag: openstack.Service
   path: /var/log/containers/service/service.log
   format: multiline
   format_firstline: '/^\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}.\d{3} \d+ \S+ \S+ \[(req-\S+ \S+ \S+ \S+ \S+ \S+|-)\]/'
   format1: '/^(?<Timestamp>\S+ \S+) (?<Pid>\d+) (?<log_level>\S+) (?<python_module>\S+) (\[(req-(?<request_id>\S+) (?<user_id>\S+) (?<tenant_id>\S+) (?<domain_id>\S+) (?<user_domain>\S+) (?<project_domain>\S+)|-)\])? (?<Payload>.*)?$/'
```

6.7. 正常なデプロイメントの確認

集中ロギングが正常にデプロイされたことを確認するには、ログを表示して、出力が予想と一致するかどうかを確認します。Kibana 等のサードパーティーの可視化ソフトウェアを使用することができます。