Red Hat OpenStack Platform 16.1

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: 2023-11-07

Red Hat OpenStack Platform 16.1 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

多様性を受け入れるオープンソースの強化 4

RED HAT ドキュメントへのフィードバック (英語のみ) 5

第1章 運用データ計測の概要 6
1.1. TELEMETRY のアーキテクチャー 6
1.2. RED HAT OPENSTACK PLATFORM におけるデータ収集 8
1.3. GNOCCHI を使用したストレージ 10
1.4. メトリックデータの表示 12

第2章 運用データ計測のプランニング 13
2.1. CEILOMETER による計測 13
2.2. COLLECTD による計測 13
2.3. データストレージのプランニング 13
2.4. アーカイブポリシーのプランニングおよび管理 14
2.5. RED HAT OPENSTACK PLATFORM デプロイメントの検証 17

第3章 アラームの管理 19
3.1. 既存のアラームの表示 19
3.2. アラームの作成 20
3.3. アラームの編集 21
3.4. アラームの無効化 21
3.5. アラームの削除 22
3.6. 例: インスタンスのディスク動作の監視 22
3.7. 例: CPU 使用率の監視 23
3.8. アラーム履歴の表示 26

第4章 ログサービスのインストールおよび設定 27
4.1. 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 27
4.2. ELASTICSEARCH を使用した集中ロギングの有効化 27
4.3. ロギング機能の設定 28
4.4. ログの管理 29
4.5. デフォルトのログファイルパスのオーバーライド 30
4.6. ログレコードの形式の変更 31
4.7. RSYSLOG と ELASTICSEARCH 間の接続のテスト 32
4.8. サーバー側のロギング 32
4.9. トレースバック 32
4.10. OPENSTACK サービスのログファイルの場所 32

第5章 COLLECTD プラグイン 39
collectd::plugin::aggregation 39
collectd::plugin::amqp1 40
collectd::plugin::apache 41
collectd::plugin::battery 43
collectd::plugin::bind 43
collectd::plugin::ceph 44
collectd::plugins::cgroups 45
collectd::plugin::connectivity 45
collectd::plugin::conntrack 46
collectd::plugin::contextswitch 46
collectd::plugin::cpu 46
collectd::plugin::cpufreq 48
collectd::plugin::csv 48

collectd::plugin::df 48
collectd::plugin::disk 49
collectd::plugin::hugepages 50
collectd::plugin::interface 51
collectd::plugin::load 52
collectd::plugin::mcelog 52
collectd::plugin::memcached 52
collectd::plugin::memory 53
collectd::plugin::ntpd 54
collectd::plugin::ovs_stats 54
collectd::plugin::processes 55
collectd::plugin::smart 56
collectd::plugin::swap 56
collectd::plugin::tcpconns 57
collectd::plugin::thermal 57
collectd::plugin::uptime 58
collectd::plugin::virt 58
collectd::plugin::vmem 59
collectd::plugin::write_http 59
collectd::plugin::write_kafka 60

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、Red Hat CTO である Chris Wright のメッセージ を参照してください。

RED HAT ドキュメントへのフィードバック (英語のみ)

Red Hat ドキュメントに対するご意見をお聞かせください。ドキュメントの改善点があればお知らせください。

ドキュメントへのダイレクトフィードバック (DDF) 機能の使用 (英語版のみ)

特定の文章、段落、またはコードブロックに対して直接コメントを送付するには、DDF の **Add Feedback** 機能を使用してください。なお、この機能は英語版のドキュメントでのみご利用いただけます。

1. **Multi-page HTML** 形式でドキュメントを表示します。
2. ドキュメントの右上隅に **Feedback** ボタンが表示されていることを確認してください。
3. コメントするテキスト部分をハイライト表示します。
4. **Add Feedback** をクリックします。
5. **Add Feedback** フィールドにコメントを入力します。
6. オプション: ドキュメントチームが問題の詳細を確認する際に使用できるメールアドレスを記入してください。
7. **Submit** をクリックします。

第1章 運用データ計測の概要

Red Hat OpenStack Platform (RHOSP) 環境の Telemetry サービスのコンポーネントを使用して、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。

可用性およびパフォーマンス監視ツールを使用して、RHOSP 環境の計測および維持を行うことができます。これらのツールは、以下の機能を果たします。

可用性監視

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。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。

パフォーマンスの監視

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します。また、ネットワーク経由でデータを利用できるようになります。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。

1.1. TELEMETRY のアーキテクチャー

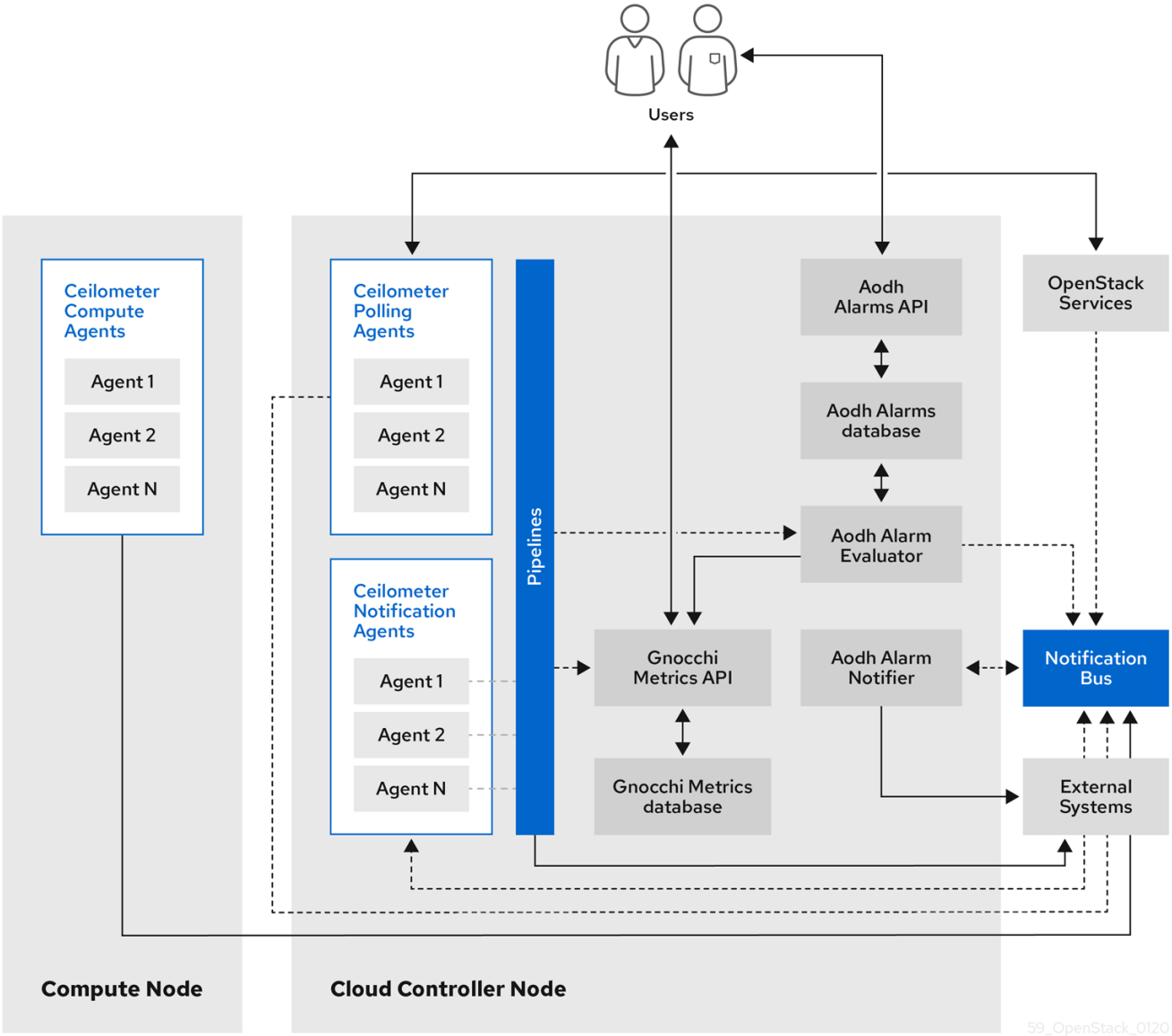
Red Hat OpenStack Platform (RHOSP) Telemetry は、OpenStack をベースとするクラウドのユーザーレベルの使用状況データを提供します。このデータを、顧客への請求、システムのモニタリング、またはアラートに使用することができます。Telemetry のコンポーネントを設定し、既存の OpenStack コンポーネントにより送信される通知から (例: Compute の使用状況イベント)、または RHOSP インフラストラクチャーリソースへのポーリングにより (例: libvirt)、データを収集することができます。Telemetry は、収集したデー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

- **Data collection**: 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。
- **Alarm service**: Telemetry は、Alarming サービス 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、サードパーティー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し、Alarming サービスを使用してイベントのアラームを設定できます。

図1.1 Telemetry のアーキテクチャー



1.1.1. 監視コンポーネントのサポート状況

以下の表に、Red Hat OpenStack Platform の監視用コンポーネントのサポート状況を示します。

表1.1 サポート状況

コンポーネント	以降でフルサポート	非推奨化	以降で削除	注記
Aodh	RHOSP 9	RHOSP 15		自動スケーリングのユースケースでサポートされています。

コンポーネント	以降でフルサポート	非推奨化	以降で削除	注記
Ceilometer	RHOSP 4			自動スケーリングおよびサービステレメトリーフレームワーク (STF) のユースケースで RHOSP のメトリクスとイベントの収集がサポートされています。
Collectd	RHOSP 11	RHOSP 17.1		STF のインフラストラクチャーメトリクスの収集がサポートされています。
Gnocchi	RHOSP 9	RHOSP 15		自動スケーリングのユースケースのメトリックのストレージがサポートされています。
Panko	RHOSP 11	RHOSP 12、RHOSP 14 以降、デフォルトではインストールされていません	RHOSP 17.0	
QDR	RHOSP 13	RHOSP 17.1		RHOSP から STF へのメトリクスおよびイベントデータの送信がサポートされています。

1.2. RED HAT OPENSTACK PLATFORM におけるデータ収集

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

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

1.2.1. 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 Compute 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** 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。

- **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。定義されたターゲットにデータを公開します。デフォルトのターゲットは gnocchi です。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

- パブリッシャーの詳細は、「パブリッシャー」を参照してください。

1.2.1.1. 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。Panko は非推奨となりました。

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. オーバークラウドをデプロイする。**openstack overcloud deploy** コマンドに修正した YAML ファイルを追加します。以下の例の **<environment_files>** を、デプロイメントに含めるその他の 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>
```

関連情報

- パラメーターの詳細は、**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。

1.2.2. collectd

パフォーマンスのモニタリングでは、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。

1.3. 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、operator 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。

1.3.1. 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。

1.3.2. 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。

1.3.3. 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1.2 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。たとえば、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。
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。

用語	定義
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リスト
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。

1.4. 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。

1.4.1. 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. 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る必要があります。

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。

関連情報

- 「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のプランニングおよび管理」

2.4. 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。

2.4.1. 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。

2.4.1.1. 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。

2.4.2. 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>
```

2.4.3. 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

- bool
  - 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

2.4.4. 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。

2.4.5. metricd ワーカー

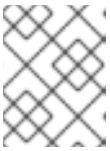
metricd デーモンを使用して、計測値の処理、集約値の作成、集約値ストレージへの計測値の保管、およびメトリックの削除を行うことができます。metricd デーモンは、Gnocchi のほとんどの CPU 使用および I/O ジョブを管理します。各メトリックのアーカイブポリシーは、metricd デーモンが実行する速度を決定します。metricd は、受信ストレージに新しい計測値がないか定期的に確認します。各チェック間の遅延を設定するには、**[metricd]metric_processing_delay configuration** オプションを使用できます。

2.4.6. 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。

2.4.7. 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>
```

2.4.8. 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>
```

2.5. RED HAT OPENSTACK PLATFORM デプロイメントの検証

openstack metric コマンドを使用して、デプロイメントが成功したことを確認できます。

手順

- デプロイメントを確認します。

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

第3章 アラームの管理

Telemetry Alarming サービス (aodh) 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。

アラームは以下の状態のいずれかになります。

ok

メトリックまたはイベントは受け入れ可能な状態になります。

firing

メトリックまたはイベントは、定義された ok 状態外です。

insufficient data

アラームの状態は不明です。これにはいくつかの理由があります。たとえば、要求される粒度のデータがなく、チェックはまだ実行されていないため、それなどがこれに該当します。

3.1. 既存のアラームの表示

既存の Telemetry アラーム情報を表示し、リソースに割り当てられたメーターを表示して、メトリックの現在の状態を確認することができます。

手順

1. 既存の Telemetry アラームをリスト表示します。

```
# 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リスト表示するには、リソースの UUID を指定します。以下に例を示します。

```
# openstack metric resource show 22592ae1-922a-4f51-b935-20c938f48753

| Field                | Value                                                                    |
+----------------------+--------------------------------------------------------------------------+
| created_by_project_id | 1adaed3aaa7f454c83307688c0825978                                         |
| created_by_user_id    | d8429405a2764c3bb5184d29bd32c46a                                         |
| creator               |
d8429405a2764c3bb5184d29bd32c46a:1adaed3aaa7f454c83307688c0825978 |
| ended_at              | None                                                                     |
| id                    | 22592ae1-922a-4f51-b935-20c938f48753                                     |
| metrics               | cpu: a0375b0e-f799-47ea-b4ba-f494cf562ad8                                |
|                       | disk.ephemeral.size: cd082824-dfd6-49c3-afdf-6bfc8c12bd2a                |
|                       | disk.root.size: cd88dc61-ba85-45eb-a7b9-4686a6a0787b                     |
```

```
|                     | memory.usage: 7a1e787c-5fa7-4ac3-a2c6-4c3821eaf80a              |
|                     | memory: ebd38ef7-cdc1-49f1-87c1-0b627d7c189e                    |
|                     | vcpus: cc9353f1-bb24-4d37-ab8f-d3e887ca8856                     |
| original_resource_id  | 22592ae1-922a-4f51-b935-20c938f48753                          |
| project_id          | cdda46e0b5be4782bc0480dac280832a                                |
| revision_end        | None                                                            |
| revision_start      | 2021-09-16T17:00:41.227453+00:00                                |
| started_at          | 2021-09-16T16:17:08.444032+00:00                                |
| type                | instance                                                        |
| user_id             | f00de1d74408428cadf483ea7dbb2a83                                |
+---------------------+-----------------------------------------------------------------+
```

3.2. アラームの作成

Telemetry Alarming サービス (aodh) を使用して、特定の条件が満たされる際にトリガーされるアラームを作成します (例: しきい値に到達する場合)。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. アーカイブポリシーはデプロイメントプロセス時に事前に設定されるため、新しいアーカイブポリシーを作成する必要はほとんどありません。ただし、設定したアーカイブポリシーがない場合は、アーカイブポリシーを作成する必要があります。5s * 86400 ポイント (5 日) のメトリックを作成するアーカイブポリシーを作成するには、以下のコマンドを使用します。

```
# openstack archive-policy create <name> \
    -d granularity:5s,points:86400 \
    -b 3 -m mean -m rate:mean
```

+ **<name>** をアーカイブポリシーの名前に置き換えます。

注記

Telemetry Alarming サービスの評価期間の値を 60 より大きい整数に設定するようにしてください。Ceilometer のポーリング間隔は、評価期間にリンクされます。Ceilometer のポーリング間隔の値を 60 から 600 までの数字に設定し、値が Telemetry Alarming サービスの評価期間の値よりも大きくされていることを確認します。Ceilometer のポーリング間隔が低すぎる場合には、システム負荷が大幅に影響を受ける可能性があります。

2. アラームを作成し、クエリーを使用してモニタリングするインスタンスの特定の ID を分離します。以下の例のインスタンスの ID は 94619081-abf5-4f1f-81c7-9cedaa872403 です。

注記

しきい値を計算するには、数式 1,000,000,000 x {granularity} x {percentage_in_decimal} を使用します。

```
# openstack alarm create \
--type gnocchi_aggregation_by_resources_threshold \
--name cpu_usage_high \
--granularity 5
--metric cpu \
```

```
--threshold 48000000000 \
--aggregation-method rate:mean \
--resource-type instance \
--query '{"=": {"id": "94619081-abf5-4f1f-81c7-9cedaa872403"}}' --alarm-action 'log://'
+---------------------------+---------------------------------------------------------+
| Field                     | Value                                                   |
+---------------------------+---------------------------------------------------------+
| aggregation_method        | rate:mean                                               |
| 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               | 5                                                       |
| insufficient_data_actions | []                                                      |
| metric                    | cpu                                                     |
| 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                 | 48000000000.0                                           |
| time_constraints          | []                                                      |
| timestamp                 | 2016-12-09T05:18:53.326000                              |
| type                      | gnocchi_aggregation_by_resources_threshold              |
| user_id                   | 32d3f2c9a234423cb52fb69d3741dbbc                        |
+---------------------------+---------------------------------------------------------+
```

3.3. アラームの編集

アラームを編集すると、アラームの値のしきい値を増減します。

手順

- しきい値を更新するには、**openstack 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
```

3.4. アラームの無効化

アラームを無効にして有効化することができます。

手順

- アラームを無効にします。

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

3.5. アラームの削除

openstack alarm delete コマンドを使用してアラームを削除します。

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
```

3.6. 例: インスタンスのディスク動作の監視

この例では、Telemetry Alarming サービスの一部である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累積ディスクアクティビティーを監視する方法を説明します。

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

3.7. 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. 監視可能なメトリックを特定するには、インスタンスの UUID を指定して **openstack metric resource show** コマンドを入力します。

```
$ openstack metric resource show --type instance 22592ae1-922a-4f51-b935-20c938f48753

+----------------------+--------------------------------------------------------------------+
| Field                | Value                                                              |
+----------------------+--------------------------------------------------------------------+
| availability_zone    | nova                                                               |
| created_at           | 2021-09-16T16:16:24+00:00                                          |
| created_by_project_id | 1adaed3aaa7f454c83307688c0825978                                  |
| created_by_user_id   | d8429405a2764c3bb5184d29bd32c46a                                   |
| creator              |
d8429405a2764c3bb5184d29bd32c46a:1adaed3aaa7f454c83307688c0825978 |
| deleted_at           | None                                                               |
| display_name         | foo-2                                                              |
| ended_at             | None                                                               |
| flavor_id            | 0e5bae38-a949-4509-9868-82b353ef7ffb                               |
| flavor_name          | workload_flavor_0                                                  |
| host                 | compute-0.redhat.local                                             |
| id                   | 22592ae1-922a-4f51-b935-20c938f48753                               |
| image_ref            | 3cde20b4-7620-49f3-8622-eeacbdc43d49                               |
| launched_at          | 2021-09-16T16:17:03+00:00                                          |
| metrics              | cpu: a0375b0e-f799-47ea-b4ba-f494cf562ad8                          |
|                      | disk.ephemeral.size: cd082824-dfd6-49c3-afdf-6bfc8c12bd2a          |
|                      | disk.root.size: cd88dc61-ba85-45eb-a7b9-4686a6a0787b               |
|                      | memory.usage: 7a1e787c-5fa7-4ac3-a2c6-4c3821eaf80a                 |
|                      | memory: ebd38ef7-cdc1-49f1-87c1-0b627d7c189e                       |
|                      | vcpus: cc9353f1-bb24-4d37-ab8f-d3e887ca8856                        |
| original_resource_id | 22592ae1-922a-4f51-b935-20c938f48753                               |
| project_id           | cdda46e0b5be4782bc0480dac280832a                                   |
```

```
| revision_end          | None                                                                 |
| revision_start        | 2021-09-16T17:00:41.227453+00:00                                     |
| server_group          | None                                                                 |
| started_at            | 2021-09-16T16:17:08.444032+00:00                                     |
| type                  | instance                                                             |
| user_id               | f00de1d74408428cadf483ea7dbb2a83                                     |
+-----------------------+----------------------------------------------------------------------+
```

この出力結果の metrics の値に、Alarming サービスを使用して監視可能なコンポーネントがリスト表示されます (例: **cpu**)。

2. CPU の使用状況を監視するには、**cpu** メトリックを使用します。

```
$ openstack metric show --resource-id 22592ae1-922a-4f51-b935-20c938f48753 cpu
+-------------------------------+----------------------------------------------------------------------+
| Field                         | Value                                                                |
+-------------------------------+----------------------------------------------------------------------+
| archive_policy/name           | ceilometer-high-rate                                                 |
| creator                       |
d8429405a2764c3bb5184d29bd32c46a:1adaed3aaa7f454c83307688c0825978 |
| id                            | a0375b0e-f799-47ea-b4ba-f494cf562ad8                                 |
| name                          | cpu                                                                  |
| resource/created_by_project_id | 1adaed3aaa7f454c83307688c0825978
|
| resource/created_by_user_id   | d8429405a2764c3bb5184d29bd32c46a
|
| resource/creator              |
d8429405a2764c3bb5184d29bd32c46a:1adaed3aaa7f454c83307688c0825978 |
| resource/ended_at             | None                                                                 |
| resource/id                   | 22592ae1-922a-4f51-b935-20c938f48753                                 |
| resource/original_resource_id | 22592ae1-922a-4f51-b935-20c938f48753
|
| resource/project_id           | cdda46e0b5be4782bc0480dac280832a                                     |
| resource/revision_end         | None                                                                 |
| resource/revision_start       | 2021-09-16T17:00:41.227453+00:00                                     |
| resource/started_at           | 2021-09-16T16:17:08.444032+00:00                                     |
| resource/type                 | instance                                                             |
| resource/user_id              | f00de1d74408428cadf483ea7dbb2a83                                     |
| unit                          | ns                                                                   |
+-------------------------------+----------------------------------------------------------------------+
```

archive_policy は、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. 現在選択されている **cpu** メトリックのアーカイブポリシーを検査します。

```
$ openstack metric archive-policy show ceilometer-high-rate

+--------------------+-----------------------------------------------------------------+
| Field              | Value                                                           |
+--------------------+-----------------------------------------------------------------+
| aggregation_methods | rate:mean, mean                                                |
| back_window        | 0                                                               |
| definition         | - timespan: 1:00:00, granularity: 0:00:01, points: 3600         |
|                    | - timespan: 1 day, 0:00:00, granularity: 0:01:00, points: 1440  |
```

```
|                     | - timespan: 365 days, 0:00:00, granularity: 1:00:00, points: 8760 |
| name                | ceilometer-high-rate                                              |
+---------------------+-------------------------------------------------------------------+
```

4. アラームサービスを使用して、**cpu** にクエリーを行うモニタリング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 \
  --threshold 800,000,000.0 \
  --comparison-operator ge \
  --aggregation-method mean \
  --granularity 300 \
  --evaluation-periods 1 \
  --alarm-action 'log://' \
  --ok-action 'log://' \
  --resource-type instance \
  --resource-id 22592ae1-922a-4f51-b935-20c938f48753

  +---------------------------+--------------------------------------+
  | Field                     | Value                                |
  +---------------------------+--------------------------------------+
  | aggregation_method        | rate:mean                            |
  | alarm_actions             | ['log:']                             |
  | alarm_id                  | c7b326bd-a68c-4247-9d2b-56d9fb18bf38 |
  | comparison_operator       | ge                                   |
  | description               | High CPU usage                       |
  | enabled                   | True                                 |
  | evaluation_periods        | 1                                    |
  | granularity               | 300                                  |
  | insufficient_data_actions | []                                   |
  | metric                    | cpu                                  |
  | name                      | high-cpu                             |
  | ok_actions                | ['log:']                             |
  | project_id                | cdda46e0b5be4782bc0480dac280832a     |
  | repeat_actions            | False                                |
  | resource_id               | 22592ae1-922a-4f51-b935-20c938f48753 |
  | resource_type             | instance                             |
  | severity                  | low                                  |
  | state                     | insufficient data                    |
  | state_reason              | Not evaluated yet                    |
  | state_timestamp           | 2021-09-21T08:02:57.090592           |
  | threshold                 | 800000000.0                          |
  | time_constraints          | []                                   |
  | timestamp                 | 2021-09-21T08:02:57.090592           |
  | type                      | gnocchi_resources_threshold          |
  | user_id                   | f00de1d74408428cadf483ea7dbb2a83     |
  +---------------------------+--------------------------------------+
```

- 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)。

3.8. アラーム履歴の表示

特定のアラームがトリガーされているかどうかを確認するには、アラーム履歴にクエリーを行い、イベント情報を表示します。

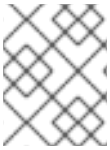
手順

- **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4章 ログサービスのインストールおよび設定

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モニタリングに使用することができます。ログ収集エージェント Rsyslog 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 Rsyslog インスタンスに送信します。サーバー側の Rsyslog 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。



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。

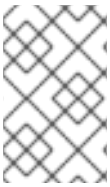
4.1. 集中ログシステムのアーキテクチャーおよびコンポーネント

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。Rsyslog 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。

集中ロギングの場合、RHOSP 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 (Rsyslog)
- データストア (ElasticSearch)
- API/プレゼンテーション層 (Grafana)

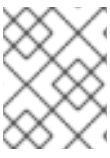


注記

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 は、Elasticsearch データベースおよび Grafana を含むサーバー側のコンポーネントはサポートしません。

4.2. ELASTICSEARCH を使用した集中ロギングの有効化

集中ロギングを有効にするには、**OS::TripleO::Services::Rsyslog** コンポーザブルサービスの実装を指定する必要があります。



注記

Rsyslog サービスは、集中ロギングのデータストアとして Elasticsearch だけを使用します。

前提条件

- Elasticsearch がサーバー側にインストールされている。

手順

- 以下の例に示すように、ご自分の環境およびデプロイに該当するその他の環境ファイルと共に、ロギング環境ファイルのファイルパスを **overcloud deployment** コマンドに追加します。

```
openstack overcloud deploy \
<existing_overcloud_environment_files> \
-e /usr/share/openstack-tripleo-heat-templates/environments/logging-environment-rsyslog.yaml
```

<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リストに置き換えます。

4.3. ロギング機能の設定

ロギング機能を設定するには、**logging-environment-rsyslog.yaml** ファイルの **RsyslogElasticsearchSetting** パラメーターを変更します。

手順

1. **tripleo-heat-templates/environments/logging-environment-rsyslog.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 **RsyslogElasticsearchSetting** パラメーターのエントリーを作成します。**RsyslogElasticsearchSetting** パラメーターの設定例を以下のスニペットに示します。

```
parameter_defaults:
    RsyslogElasticsearchSetting:
        uid: "elastic"
        pwd: "yourownpassword"
        skipverifyhost: "on"
        allowunsignedcerts: "on"
        server: "https://log-store-service-telemetry.apps.stfcloudops1.lab.upshift.rdu2.redhat.com"
        serverport: 443
```

関連情報

- 設定可能なパラメーターについての詳細は、「設定可能なロギングパラメーター」 を参照してください。

4.3.1. 設定可能なロギングパラメーター

以下の表で、Red Hat OpenStack Platform (RHOSP) のロギング機能の設定に使用するロギングパラメーターについて説明します。これらのパラメーターは **tripleo-heat-templates/deployment/logging/rsyslog-container-puppet.yaml** ファイルにあります。

表4.1 設定可能なロギングパラメーター

パラメーター	説明

パラメーター	説明
RsyslogElasticsearchSetting	**rsyslog-elasticsearch** プラグインの設定。詳細は、https://www.rsyslog.com/doc/v8-stable/configuration/modules/omelasticsearch.html を参照してください。
RsyslogElasticsearchTlsCACert	Elasticsearch サーバーの証明書を発行した CA の CA 証明書の内容が含まれます。
RsyslogElasticsearchTlsClientCert	Elasticsearch に対してクライアント証明書の認可を行うためのクライアント証明書の内容が含まれます。
RsyslogElasticsearchTlsClientKey	証明書 **RsyslogElasticsearchTlsClientCert** に対応する秘密鍵の内容が含まれます。

4.4. ログの管理

コンテナー化されたサービスのログファイルは **/var/log/containers/<service>** に保存されます (例: **/var/log/containers/cinder**)。コンテナー内で実行されているサービスからのログファイルはローカルに保存されます。利用可能なログは、有効/無効のサービスによって異なる場合があります。

以下の例では、**10** メガバイトに達すると **logrotate** のタスクを強制的に新しいログファイルを作成し、ログファイルを **14** 日間保持します。

```
parameter_defaults
  LogrotateRotate: '14'
  LogrotatePurgeAfterDays: '14'
  LogrotateMaxsize: '10M'
```

ログローテーションのパラメーターをカスタマイズするには、これらの **parameter_defaults** を環境テンプレートに追加してから、オーバークラウドをデプロイします。

```
openstack overcloud deploy \
--timeout 100 \
--templates /usr/share/openstack-tripleo-heat-templates \
... \
-e /home/stack/templates/rotate.yaml \
--log-file overcloud_deployment_90.log
```

検証: オーバークラウドノードで **logrotate_crond** が更新されていることを確認します。

```
[root@compute0 ~]# podman exec -it logrotate_crond cat /etc/logrotate-crond.conf
/var/log/containers/*/*log /var/log/containers/*/*/*log /var/log/containers/*/*err {
  daily
  rotate 14
  maxage 14
  # minsize 1 is required for GDPR compliance, all files in
  # /var/log/containers not managed with logrotate will be purged!
  minsize 1
  # Do not use size as it's not compatible with time-based rotation rules
  # required for GDPR compliance.
  maxsize 10M
```

```
    missingok
    notifempty

    copytruncate


    delaycompress


    compress


  }
```

以下の例では、**nova-compute.log** ファイルが 1 度ローテーションされました。

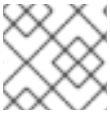
```
[root@compute0 ~]# ls -lah /var/log/containers/nova/
total 48K
drwxr-x---. 2 42436 42436  79 May 12 09:01 .
drwxr-x---. 7 root  root   82 Jan 21  2021 ..
-rw-r--r--. 1 42436 42436 12K May 12 14:00 nova-compute.log
-rw-r--r--. 1 42436 42436 33K May 12 09:01 nova-compute.log.1
-rw-r--r--. 1 42436 42436   0 Jan 21  2021 nova-manage.log
```

ログファイルのローテーションプロセスは **logrotate_crond** コンテナーで実行されます。**/var/spool/cron/root** 設定ファイルは読み取り、プロセスに送信された設定です。

検証: 設定が任意のコントローラーノードに存在することを確認します。

```
[root@controller0 ~]# podman exec -it logrotate_crond /bin/sh
()[root@9410925fded9 /]$ cat /var/spool/cron/root
# HEADER: This file was autogenerated at 2021-01-21 16:47:27 +0000 by puppet.
# HEADER: While it can still be managed manually, it is definitely not recommended.
# HEADER: Note particularly that the comments starting with 'Puppet Name' should
# HEADER: not be deleted, as doing so could cause duplicate cron jobs.
# Puppet Name: logrotate-crond
PATH=/bin:/usr/bin:/usr/sbin SHELL=/bin/sh
0 * * * * sleep `expr ${RANDOM} \% 90`; /usr/sbin/logrotate -s /var/lib/logrotate/logrotate-crond.status
/etc/logrotate-crond.conf 2>&1|logger -t logrotate-crond
```

/var/lib/config-data/puppet-generated/crond/etc/logrotate-crond.conf ファイルは **logrotate_crond** コンテナー内の **/etc/logrotate-crond.conf** にバインドされます。



注記

以前の設定ファイルは過去の理由で残されていますが、使用されません。

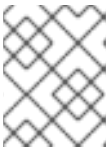
4.5. デフォルトのログファイルパスのオーバーライド

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、このパラメーターは **NovaComputeLoggingSource** です。

手順

- 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
    file: /some/other/path/nova-compute.log
```



注記

サービスごとに、**tag** および **file** を定義します。他の値にはデフォルト値が反映されます。

1. 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 Rsyslog 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>.*)?$/'
```

4.6. ログレコードの形式の変更

特定のサービスについて、ログレコードの開始の形式を変更することができます。これは Rsyslog 設定に直接渡します。

Red Hat OpenStack Platform (RHOSP) のログレコードのデフォルト形式は ('^[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}:[0-9]{2}:[0-9]{2}(.[0-9]+ [0-9]+)?(DEBUG|INFO|WARNING|ERROR) ') です。

手順

- ログレコード開始の解析に異なる正規表現を追加するには、設定に **startmsg.regex** を追加します。

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

```
    startmsg.regex: "^[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}:[0-9]{2}:[0-9]{2}(.[0-9]+ \\+[0-9]+)? [A-Z]+ \\([a-z]+\\)
```

4.7. RSYSLOG と ELASTICSEARCH 間の接続のテスト

クライアント側で、Rsyslog と Elasticsearch 間の通信を確認することができます。

手順

- Elasticsearch 接続ログファイル (Rsyslog コンテナーの **/var/log/rsyslog/omelasticsearch.log**) またはホスト上の **/var/log/containers/rsyslog/omelasticsearch.log** に移動します。このログファイルが存在しない場合や、ログファイルは存在するがログが含まれていない場合、接続の問題はありません。ログファイルが存在しログが含まれている場合は、Rsyslog は正常に接続されていません。

注記

サーバー側から接続をテストするには、Elasticsearch ログを表示して接続に問題を確認します。

4.8. サーバー側のロギング

Elasticsearch クラスターが動作中の場合は、**logging-environment-rsyslog.yaml** ファイルの **RsyslogElasticsearchSetting** パラメーターを設定して、オーバークラウドノードで実行している Rsyslog を接続する必要があります。**RsyslogElasticsearchSetting** パラメーターを設定するには、https://www.rsyslog.com/doc/v8-stable/configuration/modules/omelasticsearch.html を参照してください。

4.9. トレースバック

問題が発生してトラブルシューティングを開始する場合、トレースバックログを使用して問題の診断を行うことができます。ログファイル中、通常トレースバックとしてすべて同じ問題に関連する複数の情報行が表示されます。

Rsyslog は、ログレコードの開始を定義する正規表現を提供します。通常、各ログレコードはタイムスタンプで始まります。トレースバックの最初の行は、この情報だけが含まれる行です。Rsyslog は最初の行と共に該当するレコードをバンドルし、1 つのログレコードとして送信します。

この動作設定オプションには、<Service>LoggingSource の **startmsg.regex** が使用されます。以下の正規表現が、director のすべての <service>LoggingSource パラメーターのデフォルト値です。

```
startmsg.regex='^[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}:[0-9]{2}:[0-9]{2}(.[0-9]+ [0-9]+)? (DEBUG|INFO|WARNING|ERROR) '
```

このデフォルトが追加または変更した **LoggingSource** のログレコードと一致しない場合は、それに応じて **startmsg.regex** を変更する必要があります。

4.10. 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。

4.10.1. 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

4.10.2. 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

4.10.3. 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

サービス	サービス名	ログのパス
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

4.10.4. 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。以下の表のログファイルは、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
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

同じホストで実行中の他の Web サービスが報告するエラーを保管するログファイル **/var/log/containers/httpd/default_error.log** もあります。

4.10.5. 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

4.10.6. 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

4.10.7. 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	該当なし	/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

4.10.8. 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
```

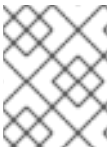
4.10.9. Orchestration (heat) のログファイル

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	該当なし	/var/log/containers/heat/heat-manage.log

4.10.10. 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。

4.10.11. 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	ceilometer_agent_notification	/var/log/containers/ceilometer/agent-notification.log
OpenStack ceilometer 中央エージェント	ceilometer_agent_central	/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	ceilometer_agent_compute	/var/log/containers/ceilometer/compute.log

4.10.12. 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
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

4.10.13. aodh (アラームサービス) のログファイル

サービス	コンテナー名	ログのパス
アラーム用 API	aodh_api	/var/log/containers/httpd/aodh-api/aodh_wsgi_access.log
アラームエバリュエーターログ	aodh_evaluator	/var/log/containers/aodh/aodh-evaluator.log
アラームリスナー	aodh_listener	/var/log/containers/aodh/aodh-listener.log
アラーム通知	aodh_notifier	/var/log/containers/aodh/aodh-notifier.log

4.10.14. gnocchi (メトリックストレージ) のログファイル

サービス	コンテナー名	ログのパス
gnocchi API	gnocchi_api	/var/log/containers/httpd/gnocchi-api/gnocchi_wsgi_access.log
gnocchi metricd	gnocchi_metricd	/var/log/containers/gnocchi/gnocchi-metricd.log
gnocchi statsd	gnocchi_statsd	/var/log/containers/gnocchi/gnocchi-statsd.log

第5章 COLLECTD プラグイン

Red Hat OpenStack Platform (RHOSP) 16.1 環境に応じて、複数の collectd プラグインを設定できます。

次のプラグインのリストは、デフォルトをオーバーライドするのに設定できる使用可能なヒートテンプレート **ExtraConfig** パラメーターを示しています。各セクションには、**ExtraConfig** オプションの一般的な設定名が記載されています。たとえば、**example_plugin** という collectd プラグインがある場合、プラグインタイトルの形式は **collectd::plugin::example_plugin** です。

以下の例のように、特定のプラグインで利用可能なパラメーターの表を参照してください。

```
ExtraConfig:
  collectd::plugin::example_plugin::<parameter>: <value>
```

Prometheus または Grafana クエリーの特定プラグインのメトリックテーブルを参照します。

collectd::plugin::aggregation

複数の値を **aggregation** プラグインで集約できます。メトリックを算出するには、**sum**、**average**、**min**、**max** などの集約関数を使用します (例: 平均および合計の CPU 統計)。

表5.1 集約パラメーター

パラメーター	型
ホスト	文字列
プラグイン	文字列
plugininstance	整数
agg_type	文字列
typeinstance	文字列
sethost	文字列
setplugin	文字列
setplugininstance	整数
settypeinstance	文字列
groupBy	文字列の配列
calculatesum	Boolean
calculatenum	Boolean
calculateaverage	Boolean

パラメーター	型
calculateminimum	Boolean
calculatemaximum	Boolean
calculatestddev	Boolean

設定例:

以下のファイルを作成するために、3 つのアグリゲート設定をデプロイします。

1. **aggregator-calcCpuLoadAvg.conf**: ホストおよび状態に分類されるすべての CPU コアの平均 CPU 負荷
2. **aggregator-calcCpuLoadMinMax.conf**: ホストおよび状態による CPU ロードグループの最小および最大数
3. **aggregator-calcMemoryTotalMaxAvg.conf**: タイプ別にグループ化されたメモリーの最大値、平均および合計

集約設定は、デフォルトの CPU およびメモリープラグイン設定を使用します。

```
parameter_defaults:
  CollectdExtraPlugins:
    - aggregation

  ExtraConfig:
    collectd::plugin::aggregation::aggregators:
      calcCpuLoadAvg:
        plugin: "cpu"
        agg_type: "cpu"
        groupby:
          - "Host"
          - "TypeInstance"
        calculateaverage: True
      calcCpuLoadMinMax:
        plugin: "cpu"
        agg_type: "cpu"
        groupby:
          - "Host"
          - "TypeInstance"
        calculatemaximum: True
        calculateminimum: True
      calcMemoryTotalMaxAvg:
        plugin: "memory"
        agg_type: "memory"
        groupby:
          - "TypeInstance"
        calculatemaximum: True
        calculateaverage: True
        calculatesum: True
```

collectd::plugin::amqp1

amqp1 プラグインを使用して、AMQ Interconnect などの amqp1 メッセージバスに値を書き込みます。

表5.2 amqp1 パラメーター

パラメーター	タイプ
manage_package	ブール値
transport	文字列
ホスト	文字列
port	整数
user	文字列
password	文字列
address	文字列
instances	ハッシュ
retry_delay	整数
send_queue_limit	整数
interval	整数

send_queue_limit パラメーターを使用して、送信メトリックキューの長さを制限します。

注記

AMQP1 接続がない場合、プラグインは送信するメッセージをキューに入れ続けます。これにより、バインドされていないメモリー消費が生じる可能性があります。デフォルト値は 0 で、発信メトリックキューを無効にします。

メトリックが見つからない場合は、**send_queue_limit** パラメーターの値を増やします。

設定例:

```
parameter_defaults:
  CollectdExtraPlugins:
    - amqp1

  ExtraConfig:
    collectd::plugin::amqp1::send_queue_limit: 5000
```

collectd::plugin::apache

apache プラグインを使用して、Apache Web サーバーによって提供される **mod_status** プラグインから Apache データを収集します。提供される各インスタンスには **interval** ごとの値 (秒単位) を指定します。インスタンスの **timeout** interval パラメーターを指定すると、値はミリ秒単位です。

表5.3 Apache パラメーター

パラメーター	型
instances	Hash
interval	整数
manage-package	ブール値
package_install_options	List

表5.4 Apache インスタンスパラメーター

パラメーター	型
url	HTTP URL
user	文字列
password	文字列
verifypeer	Boolean
verifyhost	Boolean
cacert	AbsolutePath
sslciphers	文字列
timeout	整数

設定例:

この例では、インスタンス名は **localhost** で、Apache Web サーバー (http://10.0.0.111/mod_status?auto) に接続します。プラグインと互換性のないタイプとしてステータスページが返すのを防ぐために、URL の末尾に **?auto** を追加する必要があります。

```
parameter_defaults:
  CollectdExtraPlugins:
  - apache

  ExtraConfig:
    collectd::plugin::apache::instances:
      localhost:
        url: "http://10.0.0.111/mod_status?auto"
```

関連情報

apache プラグインの設定の詳細は、apache を参照してください。

collectd::plugin::battery

battery プラグインを使用して、ラップトップのバッテリーの残量、電源、または電圧を報告します。

表5.5 バッテリーパラメーター

パラメーター	型
values_percentage	ブール値
report_degraded	ブール値
query_state_fs	ブール値
interval	整数

関連情報

battery プラグインの設定の詳細は、バッテリー を参照してください。

collectd::plugin::bind

bind プラグインを使用して、DNS サーバーからクエリーおよび応答に関するエンコードされた統計を取得します。プラグインは、値を collectd に送信します。

表5.6 バインドパラメーター

パラメーター	型
url	HTTP URL
memorystats	Boolean
opcodes	Boolean
parsetime	Boolean
qtypes	Boolean
resolverstats	Boolean
serverstats	Boolean
zonemaintstats	Boolean
views	Array
interval	整数

表5.7 バインドビューパラメーター

パラメーター	型
name	String
qtypes	Boolean
resolverstats	Boolean
cacherrsets	Boolean
zones	文字列のリスト

設定例:

```
parameter_defaults:
  CollectdExtraPlugins:
  - bind

  ExtraConfig:
    collectd::plugins::bind:
      url: http://localhost:8053/
      memorystats: true
      opcodes: true
      parsetime: false
      qtypes: true
      resolverstats: true
      serverstats: true
      zonemaintstats: true
      views:
      - name: internal
        qtypes: true
        resolverstats: true
        cacherrsets: true
      - name: external
        qtypes: true
        resolverstats: true
        cacherrsets: true
        zones:
        - "example.com/IN"
```

collectd::plugin::ceph

ceph プラグインを使用して、ceph デーモンからデータを収集します。

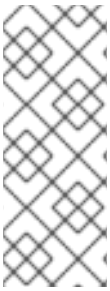
表5.8 Ceph パラメーター

パラメーター	型
daemons	Array

パラメーター	型
longrunavglatency	Boolean
convertspecialmetrictypes	Boolean
package_name	String

設定例:

```
parameter_defaults:
    ExtraConfig:
        collectd::plugin::ceph::daemons:
          - ceph-osd.0
          - ceph-osd.1
          - ceph-osd.2
          - ceph-osd.3
          - ceph-osd.4
```



注記

Object Storage Daemon (OSD) がすべてのノードにない場合には、OSD をリスト表示する必要があります。

collectd をデプロイする時に、**ceph** プラグインを Ceph ノードに追加します。デプロイメントが失敗するので、Ceph ノードの **ceph** プラグインを **CollectdExtraPlugins** に追加しないでください。

関連情報

ceph プラグインの設定の詳細は、ceph を参照してください。

collectd::plugins::cgroups

cgroups プラグインを使用して、cgroup 内のプロセスの情報を収集します。

表5.9 cgroups パラメーター

パラメーター	型
ignore_selected	Boolean
interval	整数
cgroups	List

関連情報

cgroups プラグインの設定の詳細は、cgroups を参照してください。

collectd::plugin::connectivity

connectivity プラグインを使用して、ネットワークインターフェイスの状態を監視します。

注記

インターフェイスがリストにない場合は、すべてのインターフェイスがデフォルトで監視されます。

表5.10 接続性のパラメーター

パラメーター	型
interfaces	Array

設定例:

```
parameter_defaults:
  ExtraConfig:
    collectd::plugin::connectivity::interfaces:
    - eth0
    - eth1
```

関連情報

connectivity プラグインの設定の詳細は、接続性 を参照してください。

collectd::plugin::conntrack

conntrack プラグインを使用して、Linux 接続追跡テーブルのエントリー数を追跡します。このプラグインのパラメーターはありません。

collectd::plugin::contextswitch

ContextSwitch プラグインを使用して、システムが処理するコンテキストスイッチの数を収集します。使用できるパラメーターは **interval** (秒単位) のみです。

関連情報

contextswitch プラグインの設定の詳細は、contextswitch を参照してください。

collectd::plugin::cpu

cpu プラグインを使用して、CPU がさまざまな状態に費やした時間 (例: idle、ユーザーコードの実行中、システムコードの実行中、IO 操作の待機中、その他の状態など) を監視します。

cpu プラグインは、パーセンテージの値ではなく、**jiffies** を収集します。jiffy の値は、ハードウェアプラットフォームのクロック周波数により異なるため、絶対的な間隔単位ではありません。

パーセンテージの値を報告するには、ブール値パラメーター **reportbycpu** および **reportbystate** を **true** に設定し、ブール値のパラメーター値 **percentage** を true に設定します。

ヒント

このプラグインはデフォルトで有効です。

表5.11 CPU メトリック

名前	説明	クエリー
idle	アイドル時間	**collectd_cpu_total{...,type_instance='idle'}**
interrupt	割り込みでブロックされる CPU	**collectd_cpu_total{...,type_instance='interrupt'}**
nice	優先度の低いプロセスを実行する時間	**collectd_cpu_total{...,type_instance='nice'}**
softirq	割り込み要求の処理に費やされたサイクル数	**collectd_cpu_total{...,type_instance='waitirq'}**
steal	ハイパーバイザーが別の仮想プロセッサーに対応している間、仮想 CPU が実際の CPU を待機する時間の割合	**collectd_cpu_total{...,type_instance='steal'}**
システム	システムレベル (カーネル) で費やした時間	**collectd_cpu_total{...,type_instance='system'}**
user	ユーザープロセスが使用する Jiffies	**collectd_cpu_total{...,type_instance='user'}**
wait	未処理の I/O 要求で待機中の CPU	**collectd_cpu_total{...,type_instance='wait'}**

表5.12 CPU パラメーター

パラメーター	型	Defaults
reportbystate	Boolean	true
valuespercentage	Boolean	true
reportbycpu	Boolean	true
reportnumcpu	Boolean	false
reportgueststate	Boolean	false
subtractgueststate	Boolean	true
interval	整数	120

設定例:

```
parameter_defaults:
```

```
CollectdExtraPlugins:
  - cpu
ExtraConfig:
    collectd::plugin::cpu::reportbystate: true
```

関連情報

cpu プラグインの設定の詳細は、cpu を参照してください。

collectd::plugin::cpufreq

cpufreq プラグインを使用して現在の CPU 周波数を収集します。このプラグインのパラメーターはありません。

collectd::plugin::csv

csv プラグインを使用して、CSV 形式のローカルファイルに値を書き込みます。

表5.13 csv parameters

パラメーター	型
datadir	String
storerates	Boolean
interval	整数

collectd::plugin::df

df プラグインを使用して、ファイルシステムのディスク領域の使用状況に関する情報を収集します。

ヒント

このプラグインはデフォルトで有効です。

表5.14 df メトリック

名前	説明	Query
free	空きディスク容量	**collectd_df_df_complex{...,type_instance="free"}**
reserved	予約済みディスク容量	**collectd_df_df_complex{...,type_instance="reserved"}**
used	使用済みディスク容量	**collectd_df_df_complex{...,type_instance="used"}**

表5.15 df パラメーター

パラメーター	型	Defaults
devices	Array	**[]**
fstypes	Array	**['xfs']**
ignoreselected	Boolean	true
mountpoints	Array	**[]**
reportbydevice	Boolean	true
reportinodes	Boolean	true
reportreserved	Boolean	true
valuesabsolute	Boolean	true
valuespercentage	Boolean	false

設定例:

```
parameter_defaults:
  ExtraConfig:
    collectd::plugin::df::fstypes: ['tmpfs','xfs']
```

関連情報

df プラグインの設定の詳細は、df を参照してください。

collectd::plugin::disk

disk プラグインを使用してハードディスクのパフォーマンス統計と (サポートされている場合には) パーティションの情報を収集します。

注記

すべてのディスクはデフォルトでモニターされます。**ignoreselected** パラメーターを使用して、ディスクのリストを無視できます。この設定例では、**sda** ディスク、**sdb** ディスク、および **sdc** ディスクを無視し、リストに含まれていないディスクをすべてモニターします。

ヒント

このプラグインはデフォルトで有効です。

表5.16 ディスクパラメーター

パラメーター	型	Defaults
disks	Array	[]
ignoreselected	Boolean	false
udevnameattr	String	<undefined>

表5.17 ディスクメトリック

名前	説明
merged	結合可能なキューに置かれた操作の数。たとえば、1 つの物理ディスクアクセスで 2 つ以上の論理操作が提供されます。
time	I/O 操作が完了するまでの平均時間。値は正確ではない場合があります。
io_time	I/O (ms) の処理に費やした時間。このメトリックは、デバイスの負荷率として使用できます。1 秒の値は、負荷の 100% に一致します。
weighted_io_time	I/O の完了時間と、累積する可能性のあるバックログを測定します。
pending_operations	保留中の I/O 操作のキューサイズを表示します。

設定例:

```
parameter_defaults:
  ExtraConfig:
    collectd::plugin::disk::disks: ['sda', 'sdb', 'sdc']
    collectd::plugin::disk::ignoreselected: true
```

関連情報

disk プラグインの設定の詳細は、disk を参照してください。

collectd::plugin::hugepages

hugepages プラグインを使用して hugepages 情報を収集します。

ヒント

このプラグインはデフォルトで有効です。

表5.18 hugepages パラメーター

パラメーター	型	Defaults
report_per_node_hp	Boolean	true

パラメーター	型	Defaults
report_root_hp	Boolean	true
values_pages	Boolean	true
values_bytes	Boolean	false
values_percentage	Boolean	false

設定例:

```
parameter_defaults:
  ExtraConfig:
    collectd::plugin::hugepages::values_percentage: true
```

関連情報

- **hugepages** プラグインの設定の詳細は、ヒュージページ を参照してください。

collectd::plugin::interface

interface プラグインを使用して、オクテットごとのパケット数、秒ごとのパケットレート、およびエラーレートでインターフェイストラフィックを測定します。

ヒント

このプラグインはデフォルトで有効です。

表5.19 インターフェイスパラメーター

パラメーター	型	デフォルト
interfaces	Array	[]
ignoreselected	Boolean	false
reportinactive	Boolean	true

設定例:

```
parameter_defaults:
  ExtraConfig:
    collectd::plugin::interface::interfaces:
      - lo
    collectd::plugin::interface::ignoreselected: true
```

関連情報

- **interfaces** プラグインの設定の詳細は、インターフェイス を参照してください。

collectd::plugin::load

load プラグインを使用して、システムロードとシステム使用の概要を収集します。

ヒント

このプラグインはデフォルトで有効です。

表5.20 プラグインパラメーター

パラメーター	型	デフォルト
report_relative	Boolean	true

設定例:

```
parameter_defaults:
  ExtraConfig:
    collectd::plugin::load::report_relative: false
```

関連情報

- **load** プラグインの設定の詳細は、ロード を参照してください。

collectd::plugin::mcelog

mcelog プラグインを使用して、マシンチェック例外 (MCE) の発生時に関連する通知および統計を送信します。デーモンモードで実行するように **mcelog** を設定し、ログ機能を有効にします。

表5.21 mcelog パラメーター

パラメーター	型
Mcelogfile	String
Memory	Hash **{ mcelogclientsocket[string], persistentnotification[boolean] }**

設定例:

```
parameter_defaults:
    CollectdExtraPlugins: mcelog
    CollectdEnableMcelog: true
```

関連情報

- **mcelog** プラグインの設定の詳細は、celog を参照してください。

collectd::plugin::memcached

memcached プラグインを使用して、memcached キャッシュの使用状況、メモリー、およびその他の関連情報に関する情報を取得します。

表5.22 Memcached パラメーター

パラメーター	型
instances	Hash
interval	整数

設定例:

```
parameter_defaults:
  CollectdExtraPlugins:
  - memcached

  ExtraConfig:
    collectd::plugin::memcached::instances:
      local:
        host: "%{hiera('fqdn_canonical')}"
        port: 11211
```

関連情報

- **memcached** プラグインの設定に関する詳細は、memcached を参照してください。

collectd::plugin::memory

memory プラグインを使用して、システムのメモリーに関する情報を取得します。

ヒント

このプラグインはデフォルトで有効です。

表5.23 メモリーパラメーター

パラメーター	型
Defaults	valuesabsolute
Boolean	true
valuespercentage	Boolean

設定例:

```
parameter_defaults:
  ExtraConfig:
    collectd::plugin::memory::valuesabsolute: true
    collectd::plugin::memory::valuespercentage: false
```

関連情報

- **memory** プラグインの設定の詳細は、メモリー を参照してください。

collectd::plugin::ntpd

ntpd プラグインを使用して、統計情報にアクセスできるように設定されたローカルの NTP サーバーにクエリーし、設定されたパラメーターおよび時刻同期ステータスに関する情報を取得します。

表5.24 ntpd パラメーター

パラメーター	型
host	Hostname
port	Port number (Integer)
reverselookups	Boolean
includeunitid	Boolean
interval	整数

設定例:

```
parameter_defaults:
  CollectdExtraPlugins:
  - ntpd

  ExtraConfig:
    collectd::plugin::ntpd::host: localhost
    collectd::plugin::ntpd::port: 123
    collectd::plugin::ntpd::reverselookups: false
    collectd::plugin::ntpd::includeunitid: false
```

関連情報

- **ntpd** プラグインの設定に関する詳細は、ntpd を参照してください。

collectd::plugin::ovs_stats

OVS に接続されたインターフェイスの統計値を収集するには、**ovs_stats** プラグインを使用します。**ovs_stats** プラグインは、OVSDB 管理プロトコル (RFC7047) モニターメカニズムを使用して OVSDB から統計値を取得します。

表5.25 ovs_stats パラメーター

パラメーター	型
address	String
bridges	List

パラメーター	型
port	整数
socket	String

設定例:

以下の例は、**ovs_stats** プラグインを有効にする方法を示しています。オーバークラウドを OVS でデプロイする場合には、**ovs_stats** プラグインを有効にする必要はありません。

```
parameter_defaults:
  CollectdExtraPlugins:
    - ovs_stats
  ExtraConfig:
    collectd::plugin::ovs_stats::socket: '/run/openvswitch/db.sock'
```

関連情報

- **ovs_stats** プラグインの設定の詳細は、ovs_stats を参照してください。

collectd::plugin::processes

processes プラグインは、システムプロセスに関する情報を提供します。プロセスが一致すると、プラグインは状態とプロセスのフォークレートでのみプロセス数を収集します。

特定のプロセスの詳細を収集するには、**process** パラメーターを使用してプロセス名または **process_match** オプションを使用して、正規表現に一致するプロセス名を指定できます。**process_match** 出力の統計情報はプロセス名でグループ化されます。

ヒント

このプラグインはデフォルトで有効です。

表5.26 プラグインパラメーター

パラメーター	型	Defaults
processes	Array	**<undefined>**
process_matches	Array	**<undefined>**
collect_context_switch	Boolean	**<undefined>**
collect_file_descriptor	Boolean	**<undefined>**
collect_memory_maps	Boolean	**<undefined>**

関連情報

- **processes** プラグインの設定の詳細は プロセス を参照してください。

collectd::plugin::smart

smart プラグインを使用して、ノード上の物理ディスクから SMART 情報を収集します。また、**CollectdContainerAdditionalCapAdd** パラメーターを **CAP_SYS_RAWIO** に設定して、**smart** プラグインが SMART Telemetry を読み取ることができるようにする必要があります。**CollectdContainerAdditionalCapAdd** パラメーターを設定しないと、以下のメッセージが collectd エラーログに書き込まれます。

Smart プラグイン: collectd を root として実行しますが、CAP_SYS_RAWIO 機能がありません。プラグインの読み取り機能は失敗する可能性があります。init システムで機能がドロップされましたか？

表5.27 スマートパラメーター

パラメーター	型
disks	Array
ignoreselected	Boolean
interval	整数

設定例:

```
parameter_defaults:
  CollectdExtraPlugins:
  - smart

  CollectdContainerAdditionalCapAdd: "CAP_SYS_RAWIO"
```

関連情報

- **smart** プラグインの設定の詳細については、smart を参照してください。

collectd::plugin::swap

swap プラグインを使用して、利用可能なスワップ領域および使用されているスワップ領域に関する情報を収集します。

表5.28 スワップパラメーター

パラメーター	型
reportbydevice	Boolean
reportbytes	Boolean
valuesabsolute	Boolean
valuespercentage	Boolean

パラメーター	型
reportio	Boolean

設定例:

```
parameter_defaults:
  CollectdExtraPlugins:
  - swap

  ExtraConfig:
    collectd::plugin::swap::reportbydevice: false
    collectd::plugin::swap::reportbytes: true
    collectd::plugin::swap::valuesabsolute: true
    collectd::plugin::swap::valuespercentage: false
    collectd::plugin::swap::reportio: true
```

collectd::plugin::tcpconns

tcpconns プラグインを使用して、設定されたポートから TCP 接続のインバウンドまたは送信の数に関する情報を収集します。ローカルポート設定は、ポートへの受信接続を表します。リモートポート設定は、ポートからの送信接続を表します。

表5.29 tcpconns パラメーター

パラメーター	型
localports	Port (Array)
remoteports	Port (Array)
listening	Boolean
allportssummary	Boolean

設定例:

```
parameter_defaults:
  CollectdExtraPlugins:
  - tcpconns

  ExtraConfig:
    collectd::plugin::tcpconns::listening: false
    collectd::plugin::tcpconns::localports:
    - 22
    collectd::plugin::tcpconns::remoteports:
    - 22
```

collectd::plugin::thermal

thermal プラグインを使用して、ACPI の通常のゾーン情報を取得します。

表5.30 thermal パラメーター

パラメーター	型
devices	Array
ignoreselected	Boolean
interval	整数

設定例:

```
parameter_defaults:
  CollectdExtraPlugins:
  - thermal
```

collectd::plugin::uptime

uptime プラグインを使用して、システムのアップタイムに関する情報を収集します。

ヒント

このプラグインはデフォルトで有効です。

表5.31 アップタイムのパラメーター

パラメーター	型
interval	整数

collectd::plugin::virt

virt プラグインを使用して、ホスト上の仮想マシンの **libvirt** API で CPU、ディスク、ネットワーク負荷、およびその他のメトリックを収集します。

ヒント

このプラグインはコンピュートホストでデフォルトで有効になっています。

表5.32 virt パラメーター

パラメーター	型
connection	String
refresh_interval	Hash
domain	String
block_device	String

パラメーター	型
interface_device	String
ignore_selected	Boolean
plugin_instance_format	String
hostname_format	String
interface_format	String
extra_stats	String

設定例:

```
ExtraConfig:
  collectd::plugin::virt::hostname_format: "name uuid hostname"
  collectd::plugin::virt::plugin_instance_format: metadata
```

関連情報

virt プラグインの設定の詳細は、virt を参照してください。

collectd::plugin::vmem

vmem プラグインを使用して、カーネルサブシステムから仮想メモリーに関する情報を収集します。

表5.33 vmem パラメーター

パラメーター	型
verbose	Boolean
interval	整数

設定例:

```
parameter_defaults:
  CollectdExtraPlugins:
  - vmem

  ExtraConfig:
    collectd::plugin::vmem::verbose: true
```

collectd::plugin::write_http

write_http 出力プラグインを使用して、POST リクエストを使用し JSON でメトリックをエンコードして、または **PUTVAL** コマンドを使用して、HTTP サーバーに値を送信します。

表5.34 write_http パラメーター

パラメーター	型
ensure	Enum[**present**, **absent**]
nodes	Hash[String, Hash[String, Scalar]]
urls	Hash[String, Hash[String, Scalar]]
manage_package	Boolean

設定例:

```
parameter_defaults:
  CollectdExtraPlugins:
    - write_http
  ExtraConfig:
    collectd::plugin::write_http::nodes:
      collectd:
        url: “http://collectd.tld.org/collectd”
        metrics: true
        header: “X-Custom-Header: custom_value"
```

関連情報

- **write_http** プラグインの設定に関する詳細は、write_http を参照してください。

collectd::plugin::write_kafka

write_kafka プラグインを使用して、値を Kafka トピックに送信します。**write_kafka** プラグインを 1 つ以上のトピックブロックで設定します。トピックブロックごとに、一意の名前と 1 つの Kafka プロデューサーを指定する必要があります。topic ブロックでは、以下の per-topic パラメーターを使用できます。

表5.35 write_kafka パラメーター

パラメーター	型
kafka_hosts	Array[String]
topics	Hash
properties	Hash
meta	Hash

設定例:

```
parameter_defaults:
  CollectdExtraPlugins:
    - write_kafka
```

```
ExtraConfig:
  collectd::plugin::write_kafka::kafka_hosts:
    - remote.tld:9092
  collectd::plugin::write_kafka::topics:
    mytopic:
      format: JSON
```

関連情報:

write_kafka プラグインの設定方法は write_kafka を参照してください。