Red Hat OpenStack Platform 16.1

監視ツール設定ガイド

OpenStack のロギングおよび監視ツールについてのガイド

Last Updated: 2022-03-26

Red Hat OpenStack Platform 16.1 監視ツール設定ガイド

OpenStack のロギングおよび監視ツールについてのガイド

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

本ガイドでは、Red Hat OpenStack Platform 環境でのロギングと監視の設定方法について説明します。

目次

前書き

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、弊社 の CTO、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)

弊社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連絡を取り問題についてお伺いできるように、ご自分のメールアドレスを追加します。
7. **Submit** をクリックします。

第1章 RED HAT OPENSTACK PLATFORM の監視ツールの概要

監視ツールは、オペレーターが OpenStack 環境を維持管理するのに役立つオプションのツールセットです。これらのツールは、以下の機能を果たします。

- 集中ロギング: OpenStack 環境の全コンポーネントからのログを 1 つの場所に収集します。すべてのノードとサービスにわたって問題を特定することができます。また、オプションで Red Hat にログデータをエクスポートして、問題を診断するサポートを受けることもできます。
- 可用性監視: OpenStack 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停止中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。

1.1. 監視用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	完全サポートが開始されたリリース	非推奨になったリリース	廃止されたリリース	注記
Aodh	OSP 9	OSP 15		16.1 での自動スケーリングに使用します。
Ceilometer	OSP 4			
Collectd	OSP 11			
Gnocchi	OSP 9	OSP 15		
Panko	OSP 11	OSP 12 (OSP 14 以降、デフォルトではインストールされない)	OSP 16.1	16.1 まで、Cloudforms で必要
osops-tools-monitoring-oscheck		OSP 14		

第2章 監視アーキテクチャー

監視ツールは、クライアントが Red Hat OpenStack Platform 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、有効な collectd-sensubility プラグインは、クライアント側の可用性監視 (AM) を提供します。

2.1. 集中ロギング

Red Hat OpenStack 環境では、一元的な場所に全サービスからログを収集すると、デバッグと管理が容易になります。これらのログは、syslog や audit ログファイルなどのオペレーティングシステム、RabbitMQ や MariaDB などのインフラストラクチャーコンポーネント、Identity や Compute などの 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構成されます。

- 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 (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はデプロイしません。Red Hat は、Elasticsearch データベースおよび Kibana 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。

2.2. 可用性監視

可用性監視により、OpenStack 環境全体にわたる全コンポーネントのハイレベルな機能性を一元的に監視することができます。

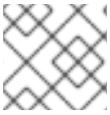
可用性監視のツールチェーンは、複数のコンポーネントで構成されます。

- 監視エージェント (有効な collectd-sensubility プラグイン)
- 監視リレー/プロキシー (RabbitMQ)
- 監視コントローラー/サーバー (Sensu サーバー)
- API/プレゼンテーション層 (Uchiwa)

注記

Red Hat OpenStack Platform director は、サーバー側の可用性監視のコンポーネントはデプロイしません。Red Hat では、Uchiwa、Sensu Server、Sensu API plus、RabbitMQ、監視ノードで実行する Redis インスタンスなどのサーバー側のコンポーネントはサポートしていません。

以下の図は、可用性監視のコンポーネントとそれらの対話を示しています。



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2.1 ハイレベルでの可用性監視のアーキテクチャー

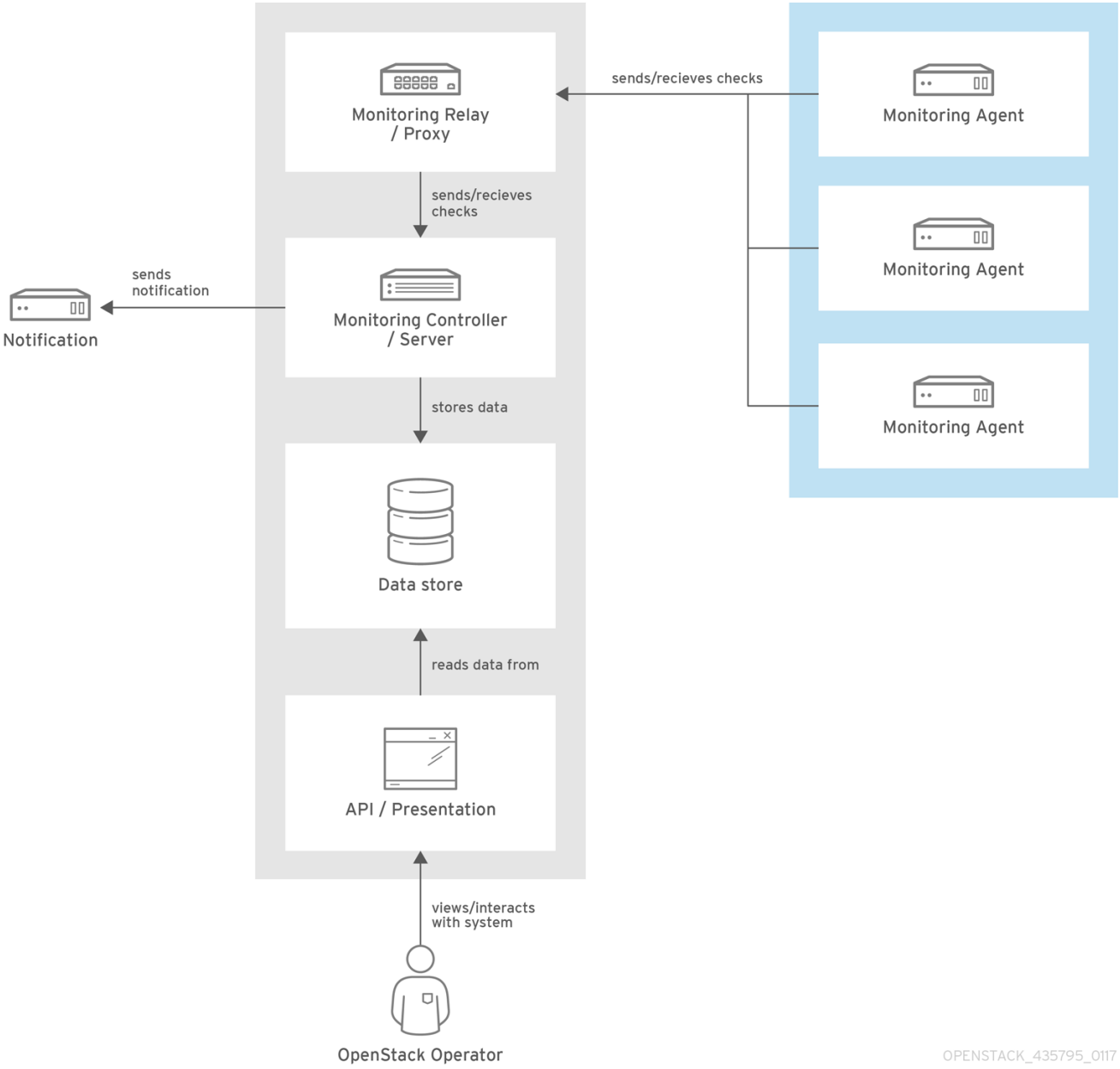


図2.2 Red Hat OpenStack Platform の単一ノードデプロイメント

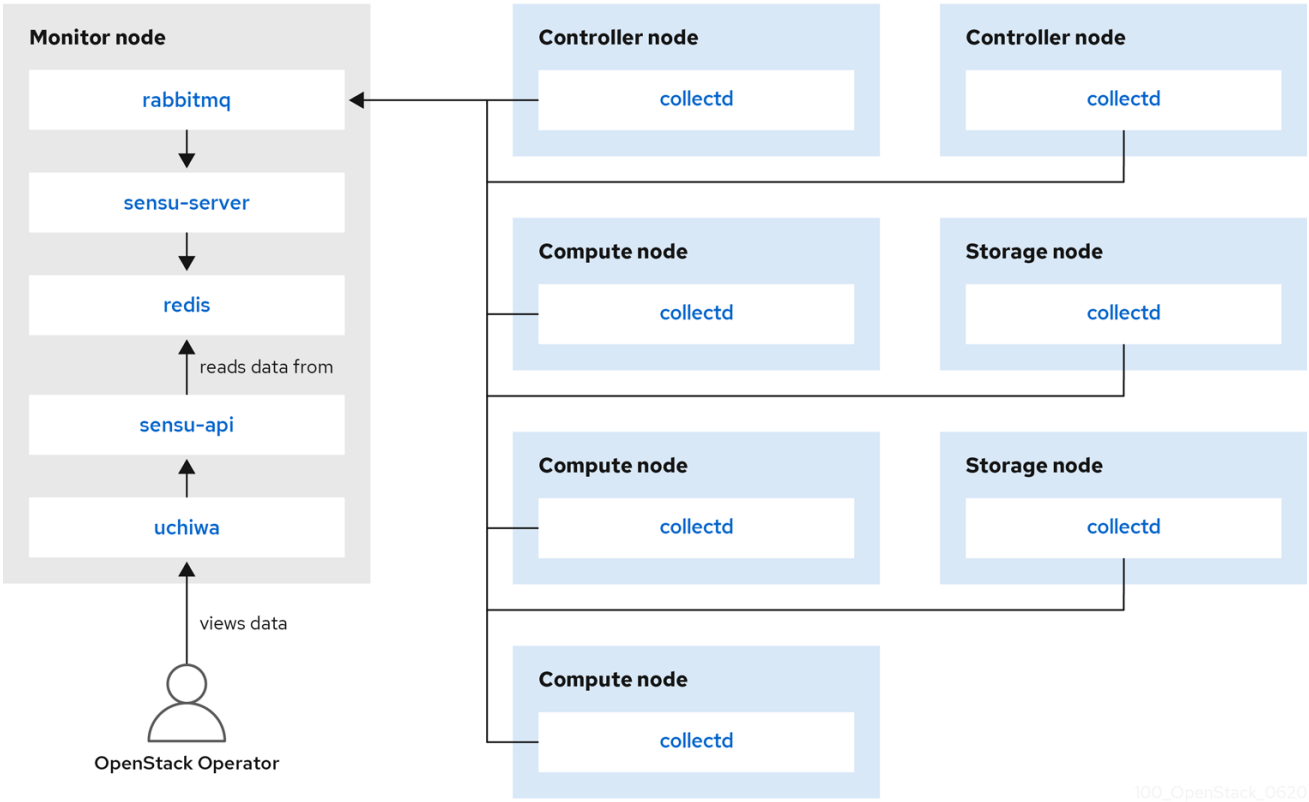
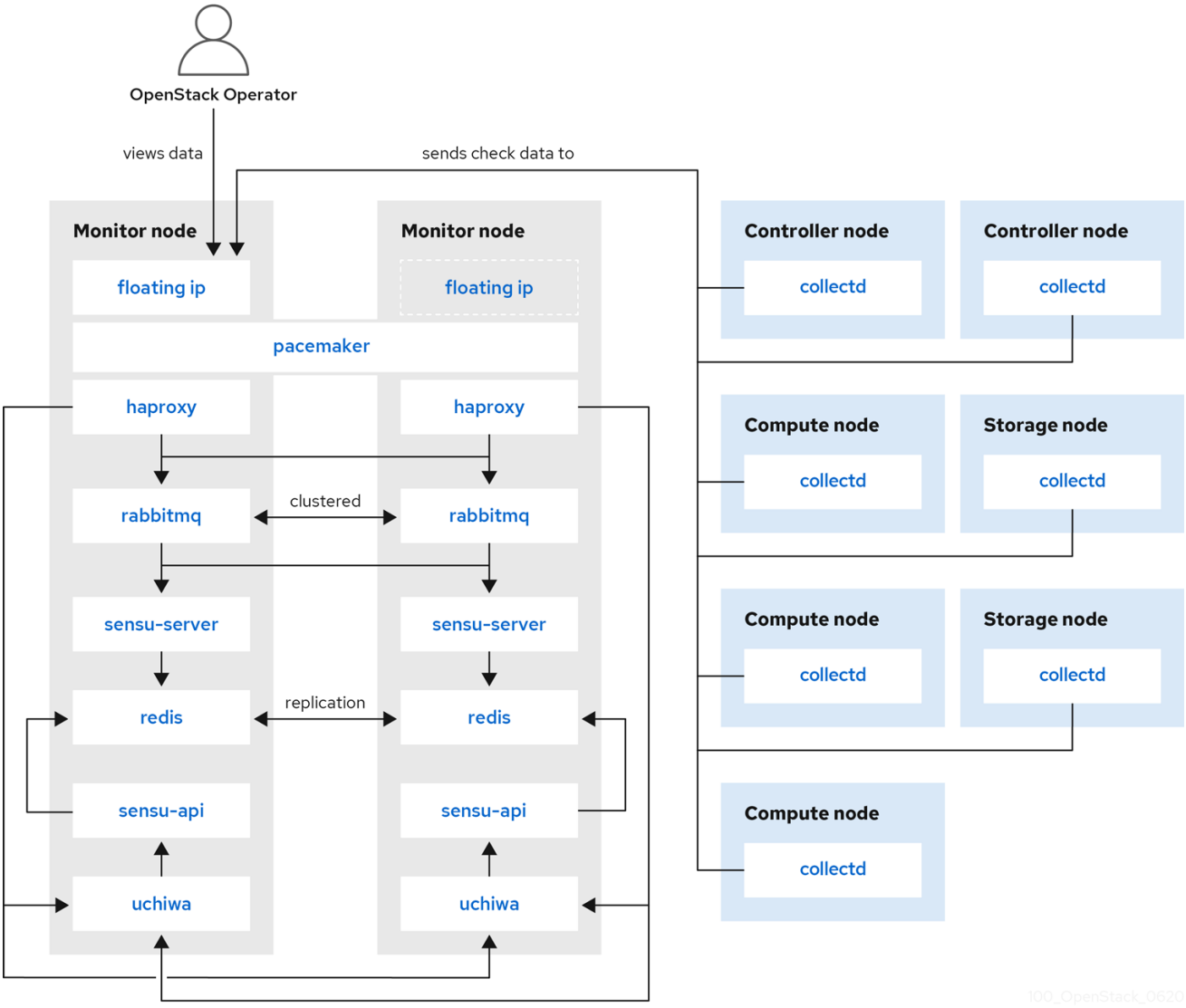


図2.3 Red Hat OpenStack Platform の HA デプロイメント



第3章 クライアント側のツールのインストール

オーバークラウドをデプロイする前には、各クライアントに適用する構成の設定を決定する必要があります。Heat テンプレートコレクションからサンプルの環境ファイルをコピーし、ご使用の環境に応じてファイルを変更します。

3.1. 集中ロギングのクライアントパラメーターの設定

詳細は、『 ロギング、モニタリング 、**およびトラブルシューティング** 』ガイドの「Elasticsearch を使用した集中ロギングの有効化 」を参照してください。

3.2. 監視クライアントパラメーターの設定

監視ソリューションは、定期的にシステム情報を収集し、データ収集エージェントを使用してさまざまな方法で値を保管し、監視するメカニズムを提供します。Red Hat は、collectd をコレクションエージェントとしてサポートします。collectd-sensubility は collectd の機能拡張であり、RabbitMQ を介して Sensu サーバー側と通信します。Service Telemetry Framework (STF) を使用してデータを保存し、続いてシステムの監視、パフォーマンスのボトルネックの特定、将来のシステム負荷の予測を行うことができます。Service Telemetry Framework についての詳細は、『Service Telemetry Framework 1.3 』ガイドを参照してください。

collectd および collectd-sensubility を設定するには、以下の手順を完了します。

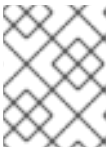
1. ホームディレクトリーに **/home/templates/custom** などの **config.yaml** を作成し、**MetricsQdrConnectors** パラメーター が STF サーバー側をポイントするように設定します。

```
MetricsQdrConnectors:
  - host: qdr-normal-sa-telemetry.apps.remote.tld
    port: 443
    role: inter-router
    sslProfile: sslProfile
    verifyHostname: false
MetricsQdrSSLProfiles:
  - name: sslProfile
```

2. **config.yaml** ファイルで、**CollectdExtraPlugins** の下に使用するプラグインを一覧表示します。**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。デフォルトでは、collectd のプラグインには **cpu**、**df**、**disk**、**hugepages**、**interface**、**load**、**memory**、**processes**、**tcpconns**、**unixsock**、および **uptime** があります。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。また、**ExtraConfig** オプションを使用して、**CollectdExtraPlugins** の設定情報を追加することもできます。たとえば、**virt** プラグインを有効にし、接続文字列とホスト名の形式を設定するには、以下の構文を使用します。

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

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



注記

unixsock プラグインは削除しないでください。削除すると、collectd コンテナーは正常ではないと永続的に指定されます。

3. オプション: AMQ Interconnect でメトリクスおよびイベントデータを収集するには、**MetricsQdrExternalEndpoint: true** 行を **config.yaml** ファイルに追加します。

```
parameter_defaults:
  MetricsQdrExternalEndpoint: true
```

4. collectd-sensubility を有効にするには、以下の環境設定を **config.yaml** ファイルに追加します。

```
parameter_defaults:
  CollectdEnableSensubility: true

  # Use this if there is restricted access for your checks by using the sudo command.
  # The rule will be created in /etc/sudoers.d for sensubility to enable it calling restricted commands via sensubility executor.
  CollectdSensubilityExecSudoRule: "collectd ALL = NOPASSWD: <some command or ALL for all commands>"

  # Connection URL to Sensu server side for reporting check results.
  CollectdSensubilityConnection: "amqp://sensu:sensu@<sensu server side IP>:5672//sensu"

  # Interval in seconds for sending keepalive messages to Sensu server side.
  CollectdSensubilityKeepaliveInterval: 20

  # Path to temporary directory where the check scripts are created.
  CollectdSensubilityTmpDir: /var/tmp/collectd-sensubility-checks

  # Path to shell used for executing check scripts.
  CollectdSensubilityShellPath: /usr/bin/sh

  # To improve check execution rate use this parameter and value to change the number of goroutines spawned for executing check  scripts.
  CollectdSensubilityWorkerCount: 2

  # JSON-formatted definition of standalone checks to be scheduled on client side. If you need to schedule checks
  # on overcloud nodes instead of Sensu server, use this parameter. Configuration is compatible with Sensu check definition.
  # For more information, see https://docs.sensu.io/sensu-core/1.7/reference/checks/#check-definition-specification
  # There are some configuration options which sensubility ignores such as: extension, publish, cron, stdin, hooks.
  CollectdSensubilityChecks:
    example:
      command: "ping -c1 -W1 8.8.8.8"
      interval: 30

  # The following parameters are used to modify standard, standalone checks for monitoring container health on overcloud nodes.
  # Do not modify these parameters.
```

```
  # CollectdEnableContainerHealthCheck: true
  # CollectdContainerHealthCheckCommand: <snip>
  # CollectdContainerHealthCheckInterval: 10
  # The Sensu server side event handler to use for events created by the container health
check.
  # CollectdContainerHealthCheckHandlers:
  #   - handle-container-health-check
  # CollectdContainerHealthCheckOccurrences: 3
  # CollectdContainerHealthCheckRefresh: 90
```

5. オーバークラウドをデプロイします。overcloud deploy コマンドに、**config.yaml** と **collectd-write-qdr.yaml** のほか、**qdr-*.yaml** ファイルのいずれかを含めます。

```
$ openstack overcloud deploy
-e  /home/templates/custom/config.yaml
-e tripleo-heat-templates/environments/metrics/collectd-write-qdr.yaml
-e tripleo-heat-templates/environments/metrics/qdr-form-controller-mesh.yaml
```

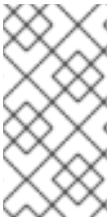
6. オプション: オーバークラウドの RabbitMQ 監視を有効にするには、**overcloud deploy** コマンドに **collectd-read-rabbitmq.yaml** ファイルを追加します。

関連情報

- YAML ファイルの詳細は、「YAML ファイル」を参照してください。
- collectd プラグインの詳細は、「collectd プラグインの設定」を参照してください。
- Service Telemetry Framework についての詳細は、『Service Telemetry Framework 1.3 』ガイドを参照してください。

3.3. AMQ INTERCONNECT を使用したデータの収集

メトリクスおよびイベントデータ消費のために利用可能な AMQ Interconnect アドレスにサブスクライブするには、クライアント接続用に AMQ Interconnect を公開する環境ファイルを作成し、オーバークラウドをデプロイします。



注記

Service Telemetry Operator は、単一クラウドデプロイメント用の全データ取得およびデータストレージコンポーネントのデプロイメントを単純化します。データストレージドメインを複数のクラウドと共有するには、『 **Service Telemetry Framework 1.3**』 の「Configuring multiple clouds 」を参照してください。

警告

Service Telemetry Framework (STF) で使用されるように QDR メッシュモードと QDR エッジモードを切り替えることはできません。さらに、STF のデータ収集を有効にする場合は、QDR メッシュモードを使用することはできません。

手順

1. Red Hat OpenStack Platform アンダークラウドに **stack** ユーザーとしてログオンします。
2. **/home/stack** ディレクトリーに **data-collection.yaml** という名前の設定ファイルを作成します。
3. 外部エンドポイントを有効にするには、**MetricsQdrExternalEndpoint: true** パラメーターを **data-collection.yaml** ファイルに追加します。

```
parameter_defaults:
  MetricsQdrExternalEndpoint: true
```

4. collectd および AMQ Interconnect を有効にするには、以下のファイルを Red Hat OpenStack Platform director デプロイメントに追加します。
   - **data-collection.yaml** 環境ファイル
   - クライアント側の AMQ Interconnect が外部エンドポイントに接続できるようにする **qdr-form-controller-mesh.yaml** ファイル

```
openstack overcloud deploy <other arguments>
  --templates /usr/share/openstack-tripleo-heat-templates \
  --environment-file <...other-environment-files...> \
  --environment-file /usr/share/openstack-tripleo-heat-
templates/environments/metrics/qdr-form-controller-mesh.yaml \
  --environment-file /home/stack/data-collection.yaml
```

5. オプション: Ceilometer および collectd イベントを収集するには、**overcloud deploy** コマンドに **ceilometer-write-qdr.yaml** および **collectd-write-qdr.yaml** ファイルを含めます。
6. オーバークラウドをデプロイします。

関連情報

- YAML ファイルの詳細は、「YAML ファイル」を参照してください。

3.4. COLLECTD プラグインの設定

Red Hat OpenStack Platform director には、多くの設定の可能性があります。お使いの環境に応じて複数の collectd プラグインを設定できます。文書化された各プラグインには、説明と設定例があります。一部のプラグインには、Grafana または Prometheus からクエリーできるメトリクスのテーブルと、設定可能なオプションの一覧 (利用可能な場合) があります。

関連情報

- collectd プラグインオプションの完全な一覧を表示するには、『**Service Telemetry Framework**』の「collectd plugins」を参照してください。

3.5. YAML ファイル

collectd を設定する際に、**overcloud deploy** コマンドに以下の YAML ファイルを追加することができます。

- **collectd-read-rabbitmq.yaml**: **python-collect-rabbitmq** を有効にし、オーバークラウドの RabbitMQ インスタンスを監視するように設定します。
- **collectd-write-qdr.yaml**: collectd が AMQ Interconnect を使用して Telemetry および通知データを送信できるようにします。
- **qdr-edge-only.yaml**: AMQ Interconnect のデプロイメントを有効にします。各オーバークラウドノードでは、エッジモードで 1 つのローカルの qdrouterd サービスが実行および操作されます。たとえば、受信したデータを直接定義された **MetricsQdrConnectors** に送信し ます。
- **qdr-form-controller-mesh.yaml**: AMQ Interconnect のデプロイメントを有効にします。各オーバークラウドノードでは、1 つのローカルの qdrouterd サービスが実行され、メッシュトポロジーを形成します。たとえば、コントローラー上の AMQ Interconnect ルーターは、定義された **MetricsQdrConnectors** に接続して内部ルーターモードで動作し、他のノード種別上の AMQ Interconnect ルーターは、エッジモードでコントローラー上で実行される内部ルーターに接続されます。

関連情報

collectd の設定に関する詳細は、「監視クライアントパラメーターの設定」を参照してください。