



Red Hat OpenStack Platform 13

Service Telemetry Framework 1.5

Service Telemetry Framework 1.5 のインストールおよびデプロイ

Red Hat OpenStack Platform 13 Service Telemetry Framework 1.5

Service Telemetry Framework 1.5 のインストールおよびデプロイ

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

コアコンポーネントをインストールし、Service Telemetry Framework 1.5 をデプロイします。

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 SERVICE TELEMETRY FRAMEWORK 1.5 の概要	6
1.1. SERVICE TELEMETRY FRAMEWORK のサポート	6
1.2. SERVICE TELEMETRY FRAMEWORK アーキテクチャー	7
1.3. RED HAT OPENSIFT CONTAINER PLATFORM のインストールサイズ	9
第2章 RED HAT OPEN SHIFT CONTAINER PLATFORM の環境を SERVICE TELEMETRY FRAMEWORK 用に準備	11
2.1. SERVICE TELEMETRY FRAMEWORK の可観測性ストラテジー	11
2.2. 永続ボリューム	11
2.3. リソースの割り当て	12
2.4. SERVICE TELEMETRY FRAMEWORK のネットワークに関する考慮事項	12
第3章 SERVICE TELEMETRY FRAMEWORK のコアコンポーネントのインストール	13
3.1. SERVICE TELEMETRY FRAMEWORK の RED HAT OPENSIFT CONTAINER PLATFORM 環境へのデプロイ	13
3.2. RED HAT OPENSIFT CONTAINER PLATFORM での SERVICE TELEMETRY オブジェクトの作成	17
3.3. STF コンポーネントのユーザーインターフェイスへのアクセス	25
3.4. 代替可観測性ストラテジーの設定	25
第4章 サービス TELEMETRY フレームワーク向けの RED HAT OPENSTACK PLATFORM の設定	27
4.1. SERVICE TELEMETRY FRAMEWORK 向けの RED HAT OPENSTACK PLATFORM オーバークラウドのデプロイ	27
4.2. SERVICE TELEMETRY FRAMEWORK で使用される RED HAT OPENSTACK PLATFORM サービスの無効化	35
4.3. 標準外のネットワークトポロジーへのデプロイメント	36
4.4. アンダークラウドへの RED HAT OPENSTACK PLATFORM コンテナイメージの追加	37
4.5. 複数のクラウドの設定	38
第5章 SERVICE TELEMETRY FRAMEWORK の運用機能の使用	47
5.1. SERVICE TELEMETRY FRAMEWORK でのダッシュボード	47
5.2. SERVICE TELEMETRY FRAMEWORK でのメトリクスの保持期間	49
5.3. SERVICE TELEMETRY FRAMEWORK でのアラート	50
5.4. SNMP トラップの設定	57
5.5. 高可用性	57
5.6. 一時ストレージ	58
5.7. SERVICE TELEMETRY FRAMEWORK の可観測性ストラテジー	60
第6章 AMQ INTERCONNECT 証明書の更新	62
6.1. 期限切れの AMQ INTERCONNECT CA 証明書の確認	62
6.2. AMQ INTERCONNECT CA 証明書の更新	63
第7章 RED HAT OPENSIFT CONTAINER PLATFORM 環境からの SERVICE TELEMETRY FRAMEWORK の削除	64
7.1. NAMESPACE の削除	64
7.2. CATALOGSOURCE の削除	64
7.3. CERT-MANAGER OPERATOR の削除	64
第8章 SERVICE TELEMETRY FRAMEWORK のバージョン 1.5 へのアップグレード	67
8.1. SERVICE TELEMETRY FRAMEWORK 1.4 オペレーターの削除	67
8.2. RED HAT OPENSIFT CONTAINER PLATFORM の 4.10 へのアップグレード	71

8.3. SERVICE TELEMETRY FRAMEWORK 1.5 オペレーターのインストール	71
8.4. RED HAT OPENSTACK PLATFORM での AMQ INTERCONNECT CA 証明書の更新	74

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](#) をご覧ください。

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 SERVICE TELEMETRY FRAMEWORK 1.5 の概要

Service Telemetry Framework (STF) は、Red Hat OpenStack Platform (RHOSP) またはサードパーティーのノードからモニターリングデータを収集します。STF を使用して、以下のタスクを実行できます。

- 履歴情報の監視データの格納またはアーカイブします。
- ダッシュボードで図表としてモニターリングデータを表示します。
- モニターリングデータを使用したアラートまたは警告をトリガーします。

モニターリングデータはメトリクスまたはイベントのいずれかです。

メトリクス

アプリケーションまたはシステムの数値測定。

イベント

システムで不規則な状態や目立った事態の発生。

STF のコンポーネントは、データトランスポートにメッセージバスを使用します。データを受信して保存する他のモジュラーコンポーネントは、Red Hat OpenShift Container Platform のコンテナとしてデプロイされます。



重要

STF は、Red Hat OpenShift Container Platform バージョン 4.10 から 4.12 と互換性があります。

関連情報

- Red Hat OpenShift Container Platform のデプロイ方法に関する詳細は、Red Hat OpenShift Container Platform の [製品ドキュメント](#) を参照してください。
- Red Hat Open Shift Container Platform は、クラウドプラットフォームまたはベアメタルにインストールすることができます。STF のパフォーマンスとスケーリングの詳細については、<https://access.redhat.com/articles/4907241> を参照してください。
- Red Hat OpenShift Container Platform は、ベアメタルまたはその他のサポートされているクラウドプラットフォームにインストールできます。Red Hat OpenShift Container Platform のインストールに関する詳細は、[OpenShift Container Platform 4.12 ドキュメント](#) を参照してください。

1.1. SERVICE TELEMETRY FRAMEWORK のサポート

Red Hat は Service Telemetry Framework (STF) の最新の 2 つのバージョンをサポートしています。それ以前のバージョンには対応していません。詳細は、[Service Telemetry Framework Supported Version Matrix](#) を参照してください。

Red Hat は、AMQ Interconnect、Service Telemetry Operator、Smart Gateway Operator などのコア Operator とワークロードをサポートしています。Red Hat は、ElasticSearch、Prometheus、Alertmanager、Grafana、およびそれらの Operator を含むコミュニティの Operator やワークロードコンポーネントをサポートしていません。

STF は、完全に接続されたネットワーク環境でのみデプロイできます。Red Hat OpenShift Container Platform に接続されていない環境またはネットワークプロキシ環境に STF をデプロイすることはできません。

1.2. SERVICE TELEMETRY FRAMEWORK アーキテクチャー

Service Telemetry Framework (STF) は、クライアント/サーバーアーキテクチャーを使用します。Red Hat OpenStack Platform (RHOSP) はクライアントであり、Red Hat OpenShift Container Platform はサーバーです。

STF は、以下のコンポーネントで設定されます。

- データ収集
 - collectd: インフラストラクチャーメトリクスおよびイベントを収集します。
 - Ceilometer: RHOSP のメトリクスやイベントを収集します。
- トランスポート
 - AMQ 相互接続: AMQP 1.x 互換のメッセージングバス。高速で信頼性の高いデータ転送を提供し、ストレージ用にメトリックを STF に転送します。
 - Smart Gateway: AMQP 1.x バスからメトリクスおよびイベントを取得し、ElasticSearch または Prometheus に配信する Golang アプリケーション。
- データストレージ
 - Prometheus: Smart Gateway から受信される STF メトリクスを保存する時系列データストレージ。
 - ElasticSearch: Smart Gateway から受信される STF イベントを保存するデータストレージ。
- 観察
 - Alertmanager: Prometheus アラートルールを使用してアラートを管理するアラートツール。
 - Grafana: データのクエリー、視覚化、および管理に使用できる可視化アプリケーションおよび解析アプリケーション。

以下の表は、クライアントおよびサーバーコンポーネントのアプリケーションについて説明しています。

表1.1 STF のクライアントおよびサーバーコンポーネント

Component (コンポーネント)	クライアント	Server
AMQP 1.x と互換性のあるメッセージングバス	はい	はい
Smart Gateway	いいえ	はい
Prometheus	いいえ	はい

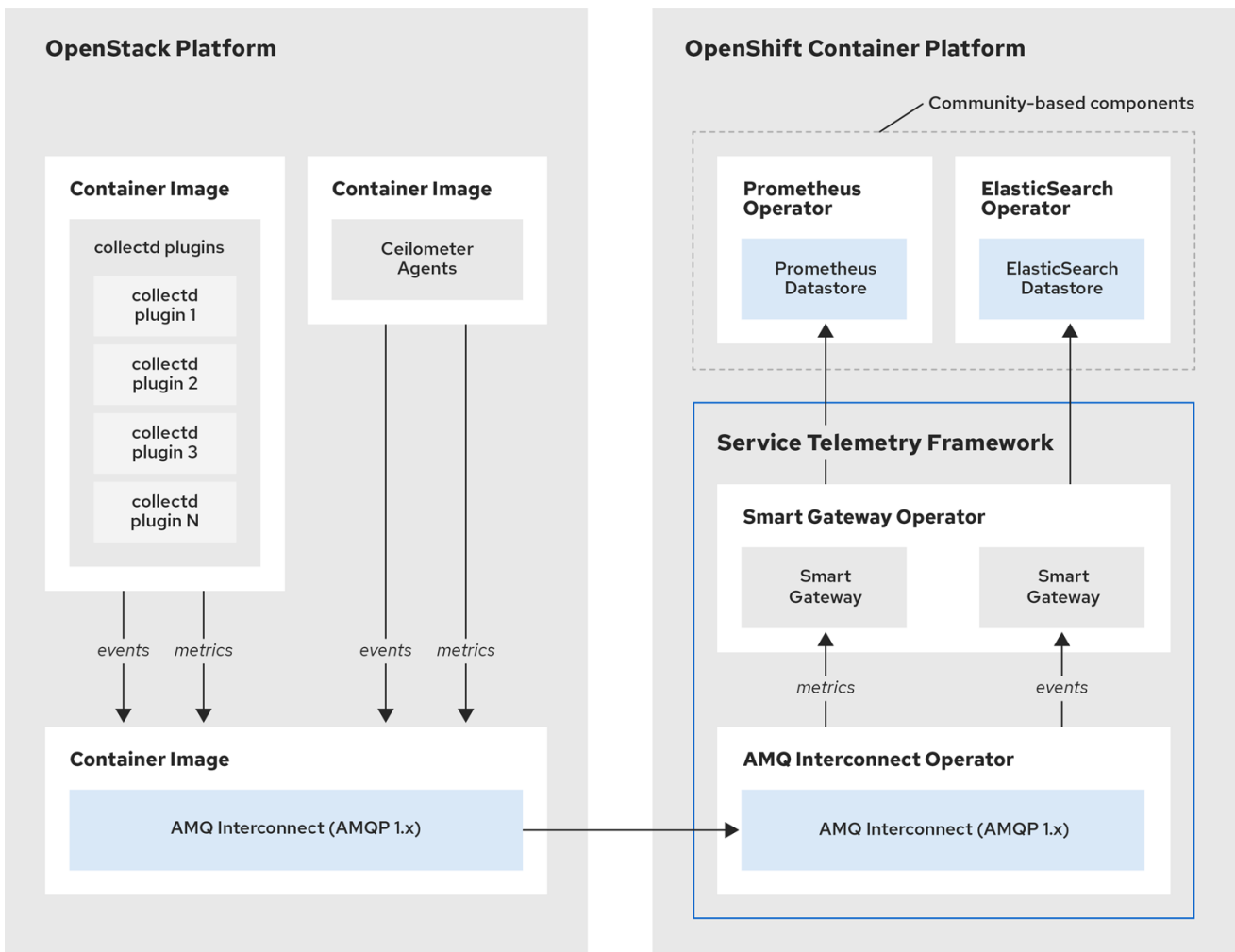
Component (コンポーネント)	クライアント	Server
Elasticsearch	いいえ	はい
collectd	はい	いいえ
Ceilometer	はい	いいえ



重要

モニターリングプラットフォームがクラウドで動作する問題を報告できるようにするには、監視しているものと同じインフラストラクチャーに STF をインストールしないでください。

図1.1 Service Telemetry Framework アーキテクチャー概要



65_OpenStack_0620

クライアント側のメトリクスの場合、collectd はプロジェクトデータなしでインフラストラクチャーメトリクスを提供し、Ceilometer はプロジェクトまたはユーザーのワークロードに基づいて RHOSP プラットフォームデータを提供します。Ceilometer も collectd も、AMQ Interconnect トランスポートを

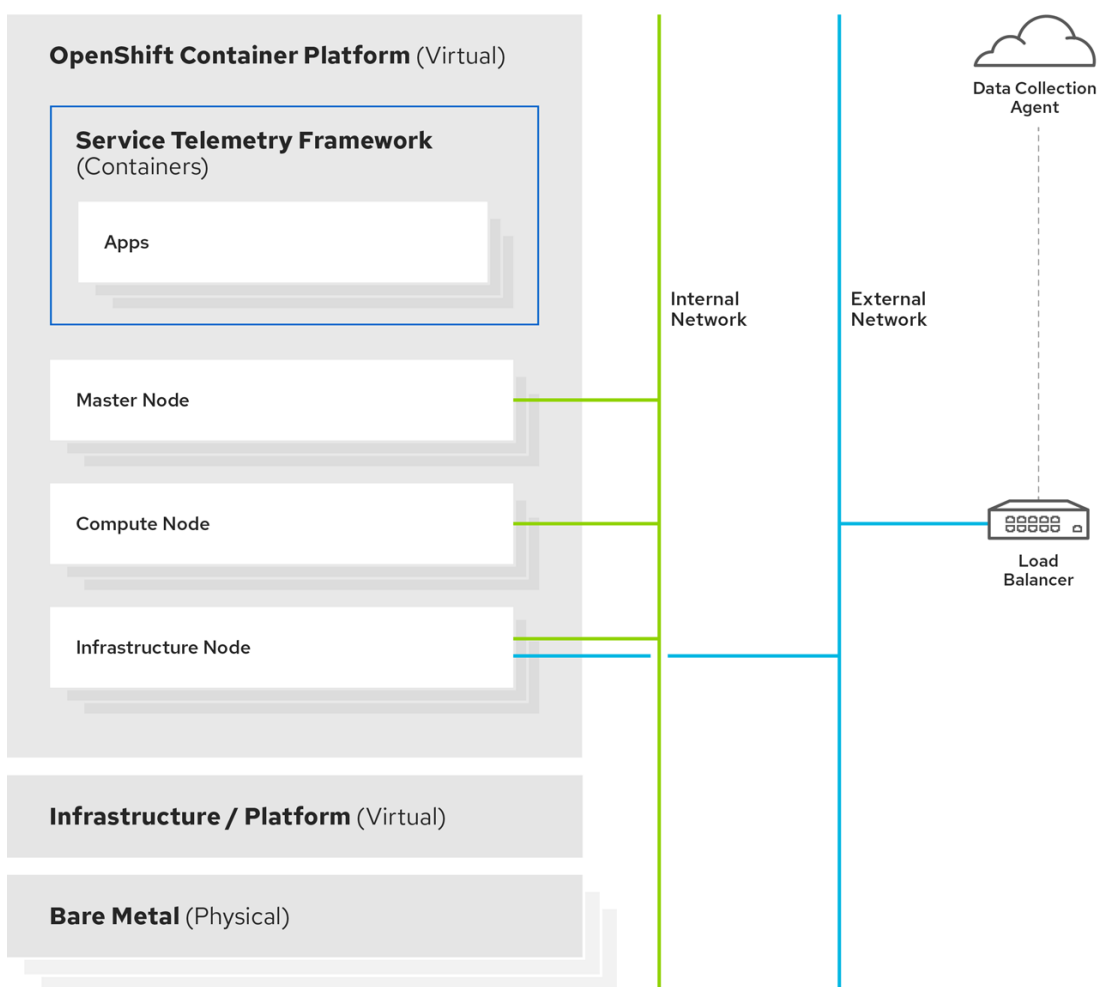
使用して、メッセージバスを介してデータを Prometheus に配信します。サーバー側では、Smart Gateway と呼ばれる Golang アプリケーションがバスからのデータストリームを受け取り、それを Prometheus のローカルスクレイプエンドポイントとして公開します。

イベントを収集して保存する予定の場合は、collectd および Ceilometer が AMQ Interconnect トランスポートを使用し、イベントデータをサーバー側に渡します。別の Smart Gateway が Elasticsearch データストアにデータを書き込みます。

サーバー側の STF 監視インフラストラクチャーは、以下のレイヤーで設定されています。

- Service Telemetry Framework 1.5
- Red Hat OpenShift Container Platform 4.10 から 4.12
- インフラストラクチャープラットフォーム

図1.2 サーバーサイドの STF 監視インフラストラクチャー



65_OpenStack_0120

1.3. RED HAT OPENSIFT CONTAINER PLATFORM のインストールサイズ

Red Hat OpenShift Container Platform のインストールサイズは、以下の要素によって異なります。

- 選択するインフラストラクチャー。
- 監視するノード数。

- 収集するメトリクスの数。
- メトリクスの解決。
- データを保存する期間です。

Service Telemetry Framework (STF) のインストールは、既存の Red Hat OpenShift Container Platform 環境によって異なります。

Red Hat OpenShift Container Platform をベアメタルにインストールする場合の最低リソース要件の詳細は、[ベアメタルへのインストールの最小リソース要件](#)を参照してください。インストールできる各種パブリックおよびプライベートのクラウドプラットフォームのインストール要件については、選択したクラウドプラットフォームに対応するインストールドキュメントを参照してください。

第2章 RED HAT OPEN SHIFT CONTAINER PLATFORM の環境を SERVICE TELEMETRY FRAMEWORK 用に準備

Service Telemetry Framework (STF) 用に Red Hat OpenShift Container Platform 環境を準備するには、永続ストレージ、適切なリソース、イベントストレージ、およびネットワークに関する考慮事項を計画する必要があります。

- 実稼働環境レベルのデプロイメントができるように、永続ストレージが Red Hat OpenShift Container Platform クラスタで利用できるようにしてください。詳細は、「[永続ボリューム](#)」を参照してください。
- Operators とアプリケーションコンテナを動かすのに十分なリソースが確保されていることを確認してください。詳細は、「[リソースの割り当て](#)」を参照してください。
- 完全に接続されたネットワーク環境があることを確認します。詳細は、「[Service Telemetry Framework のネットワークに関する考慮事項](#)」を参照してください。

2.1. SERVICE TELEMETRY FRAMEWORK の可観測性ストラテジー

Service Telemetry Framework (STF) には、ストレージバックエンドおよびアラートツールは含まれません。STF はコミュニティ Operator を使用して Prometheus、Alertmanager、Grafana、および Elasticsearch をデプロイします。STF は、これらのコミュニティ Operator にリクエストを行い、STF と連携するように設定された各アプリケーションのインスタンスを作成します。

Service Telemetry Operator がカスタムリソース要求を作成する代わりに、これらのアプリケーションまたは他の互換性のあるアプリケーションの独自のデプロイメントを使用し、Telemetry ストレージ用の独自の Prometheus 互換システムに配信するためにメトリクス Smart Gateways を収集できます。代わりに別のバックエンドを使用するように可観測性ストラテジーを設定する場合、STF には永続ストレージまたは一時ストレージは必要ありません。

2.2. 永続ボリューム

Service Telemetry Framework (STF) は、Red Hat OpenShift Container Platform で永続的なストレージを使用して永続的なボリュームを要求し、Prometheus と Elasticsearch がメトリクスとイベントを保存できるようにします。

永続ストレージが Service Telemetry Operator で有効な場合には、STF デプロイメントで要求される Persistent Volume Claim (永続ボリューム要求、PVC) のアクセスモードは RWO (ReadWriteOnce) になります。お使いの環境に事前にプロビジョニングされた永続ボリュームが含まれている場合は、Red Hat OpenShift Container Platform でデフォルト設定されている **storageClass** で RWO のボリュームが利用できるようにしてください。

関連情報

- Red Hat OpenShift Container Platform の永続ストレージの設定についての詳細は、「[永続ストレージについて](#)」を参照してください。
- Red Hat OpenShift Container Platform で設定可能な推奨のストレージ技術の詳細は、「[設定可能な推奨のストレージ技術](#)」を参照してください。
- STF における Prometheus の永続的なストレージの設定については、「[Prometheus に永続ストレージの設定](#)」を参照してください。

- STF における Elasticsearch の永続的ストレージの設定については、「[Elasticsearch のための永続的なストレージの設定](#)」を参照してください。

2.2.1. 一時ストレージ

一時ストレージを使用して、Red Hat OpenShift Container Platform クラスターにデータを永続的に保存せずに Service Telemetry Framework (STF) を実行できます。



警告

一時ストレージを使用している場合は、Pod が別のノードで再起動、更新、再スケジューリングされると、データが失われる可能性があります。一時ストレージは、本番環境ではなく、開発やテストにのみ使用してください。

2.3. リソースの割り当て

Red Hat OpenShift Container Platform インフラストラクチャー内での Pod のスケジューリングを有効にするには、実行中のコンポーネント向けにリソースが必要になります。十分なリソースが割り当てられていない場合には、Pod をスケジューリングできないため **Pending** 状態のままになります。

Service Telemetry Framework (STF) の実行に必要なリソースの量は、環境とモニターするノードおよびクラウドの数によって異なります。

関連情報

- メトリクス収集のためのサイジングに関する推奨事項については、[Service Telemetry Framework Performance and Scaling](#) を参照してください。
- Elasticsearch のサイジング要件については、<https://www.elastic.co/guide/en/cloud-on-k8s/current/k8s-managing-compute-resources.html> を参照してください。

2.4. SERVICE TELEMETRY FRAMEWORK のネットワークに関する考慮事項

Service Telemetry Framework (STF) は、完全に接続されたネットワーク環境にのみデプロイできます。Red Hat OpenShift Container Platform に接続されていない環境またはネットワークプロキシ環境に STF をデプロイすることはできません。

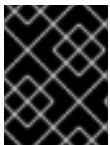
第3章 SERVICE TELEMETRY FRAMEWORK のコアコンポーネントのインストール

Operator を使用して Service Telemetry Framework (STF) コンポーネントおよびオブジェクトを読み込むことができます。Operator は以下の STF コアおよびコミュニティコンポーネントのそれぞれを管理します。

- cert-manager
- AMQ Interconnect
- Smart Gateway
- Prometheus と AlertManager
- Elasticsearch
- Grafana

前提条件

- 4.10 から 4.12 までの Red Hat OpenShift Container Platform バージョンが実行されています。
- Red Hat OpenShift Container Platform 環境を準備し、永続ストレージがあり、Red Hat OpenShift Container Platform 環境の上部で STF コンポーネントを実行するのに十分なリソースがあることを確認している。詳細は、[Service Telemetry Framework Performance and Scaling](#) を参照してください。
- 環境は完全に接続されています。STF は、Red Hat OpenShift Container Platform に接続されていない環境またはネットワークプロキシ環境では機能しません。



重要

STF は、Red Hat OpenShift Container Platform バージョン 4.10 から 4.12 と互換性があります。

関連情報

- Operator についての詳細は、[Operator についてガイドを参照して](#) ください。

3.1. SERVICE TELEMETRY FRAMEWORK の RED HAT OPENSIFT CONTAINER PLATFORM 環境へのデプロイ

Service Telemetry Framework (STF) をデプロイして、イベントを収集し、保存し、監視します。

手順

1. STF コンポーネントが含まれる namespace を作成します (例: **service-telemetry**)。

```
$ oc new-project service-telemetry
```

2. Operator Pod をスケジュールできるように、namespace に OperatorGroup を作成します。

```
$ oc create -f - <<EOF
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: service-telemetry-operator-group
  namespace: service-telemetry
spec:
  targetNamespaces:
  - service-telemetry
EOF
```

詳細は、[OperatorGroups](#) を参照してください。

- cert-manager Operator の namespace を作成します。

```
$ oc create -f - <<EOF
apiVersion: project.openshift.io/v1
kind: Project
metadata:
  name: openshift-cert-manager-operator
spec:
  finalizers:
  - kubernetes
EOF
```

- cert-manager Operator の OperatorGroup を作成します。

```
$ oc create -f - <<EOF
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: openshift-cert-manager-operator
  namespace: openshift-cert-manager-operator
spec: {}
EOF
```

- redhat-operators CatalogSource を使用して、cert-manager Operator にサブスクライブします。

```
$ oc create -f - <<EOF
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: openshift-cert-manager-operator
  namespace: openshift-cert-manager-operator
spec:
  channel: tech-preview
  installPlanApproval: Automatic
  name: openshift-cert-manager-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
EOF
```

6. ClusterServiceVersion を検証します。cert-manager Operator が **Succeeded** のフェーズを表示していることを確認します。

```
$ oc get csv --namespace openshift-cert-manager-operator --
selector=operators.coreos.com/openshift-cert-manager-operator.openshift-cert-manager-
operator
```

NAME	DISPLAY	VERSION	REPLACES	PHASE
openshift-cert-manager.v1.7.1	cert-manager Operator	for Red Hat OpenShift	1.7.1-1	Succeeded

7. redhat-operators CatalogSource を使用して AMQ Interconnect Operator にサブスクライブします。

```
$ oc create -f - <<EOF
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: amq7-interconnect-operator
  namespace: service-telemetry
spec:
  channel: 1.10.x
  installPlanApproval: Automatic
  name: amq7-interconnect-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
EOF
```

8. ClusterServiceVersion を検証します。amq7-interconnect-operator.v1.10.10 が **Succeeded** のフェーズを表示していることを確認します。

```
$ oc get csv --selector=operators.coreos.com/amq7-interconnect-operator.service-telemetry
```

NAME	DISPLAY	VERSION	REPLACES
amq7-interconnect-operator.v1.10.10	Red Hat Integration - AMQ Interconnect	1.10.10	
amq7-interconnect-operator.v1.10.4	Succeeded		

9. OperatorHub.io Community Catalog Source を有効にし、データストレージおよび可視化 Operator をインストールします。



警告

Red Hat は、AMQ Interconnect、Service Telemetry Operator、Smart Gateway Operator などのコア Operator とワークロードをサポートしています。Red Hat は、Elasticsearch、Prometheus、Alertmanager、Grafana、およびそれらの Operator を含むコミュニティの Operator やワークロードコンポーネントをサポートしていません。

```
$ oc create -f - <<EOF
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: operatorhubio-operators
  namespace: openshift-marketplace
spec:
  sourceType: grpc
  image: quay.io/operatorhubio/catalog:latest
  displayName: OperatorHub.io Operators
  publisher: OperatorHub.io
EOF
```

10. メトリクスを Prometheus に保存する計画がある場合、Prometheus Operator を有効にする必要があります。Prometheus Operator を有効にするには、Red Hat OpenShift Container Platform 環境で以下のマニフェストを作成します。

```
$ oc create -f - <<EOF
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: prometheus
  namespace: service-telemetry
spec:
  channel: beta
  installPlanApproval: Automatic
  name: prometheus
  source: operatorhubio-operators
  sourceNamespace: openshift-marketplace
EOF
```

11. Prometheus **Succeeded** の ClusterServiceVersion を確認します。

```
$ oc get csv --selector=operators.coreos.com/prometheus.service-telemetry
```

NAME	DISPLAY	VERSION	REPLACES	PHASE
prometheusoperator.0.47.0	Prometheus Operator	0.47.0	prometheusoperator.0.37.0	Succeeded

12. イベントを Elasticsearch に保存する予定の場合は、Elastic Cloud on Kubernetes (ECK) Operator を有効にする必要があります。ECK Operator を有効にするには、Red Hat OpenShift Container Platform 環境で以下のマニフェストを作成します。

```
$ oc create -f - <<EOF
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: elasticsearch-eck-operator-certified
  namespace: service-telemetry
spec:
  channel: stable
  installPlanApproval: Automatic
  name: elasticsearch-eck-operator-certified
```

```
source: certified-operators
sourceNamespace: openshift-marketplace
EOF
```

13. Kubernetes **Succeeded** で Elasticsearch Cloud の ClusterServiceVersion を確認します。

```
$ oc get csv --selector=operators.coreos.com/elasticsearch-eck-operator-certified.service-telemetry
```

NAME	DISPLAY	VERSION	REPLACES	PHASE
elasticsearch-eck-operator-certified.v2.4.0	Elasticsearch (ECK) Operator	2.4.0		
elasticsearch-eck-operator-certified.v2.3.0	Succeeded			

14. Service Telemetry Operator サブスクリプションを作成し、STF インスタンスを管理します。

```
$ oc create -f - <<EOF
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: service-telemetry-operator
  namespace: service-telemetry
spec:
  channel: stable-1.5
  installPlanApproval: Automatic
  name: service-telemetry-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
EOF
```

15. Service Telemetry Operator および依存する Operator を検証します。

```
$ oc get csv --namespace service-telemetry
```

NAME	DISPLAY	VERSION
amq7-interconnect-operator.v1.10.10	Red Hat Integration - AMQ Interconnect	
1.10.10	amq7-interconnect-operator.v1.10.4	Succeeded
elasticsearch-eck-operator-certified.v2.4.0	Elasticsearch (ECK) Operator	2.4.0
elasticsearch-eck-operator-certified.v2.3.0	Succeeded	
openshift-cert-manager.v1.7.1	cert-manager Operator for Red Hat OpenShift	
1.7.1-1	Succeeded	
prometheusoperator.0.47.0	Prometheus Operator	0.47.0
prometheusoperator.0.37.0	Succeeded	
service-telemetry-operator.v1.5.1664298822	Service Telemetry Operator	
1.5.1664298822	Succeeded	
smart-gateway-operator.v5.0.1664298817	Smart Gateway Operator	
5.0.1664298817	Succeeded	

3.2. RED HAT OPENSIFT CONTAINER PLATFORM での SERVICETELEMETRY オブジェクトの作成

Red Hat OpenShift Container Platform で **ServiceTelemetry** オブジェクトを作成します。これにより、Service Telemetry Operator が Service Telemetry Framework (STF) デプロイメントのサポートコンポーネントを作成します。詳細は、「[ServiceTelemetry オブジェクトのパラメーター](#)」を参照して

ください。

手順

1. デフォルト値を使用して STF をデプロイする **ServiceTelemetry** オブジェクトを作成するには、空の **spec** パラメーターで **ServiceTelemetry** オブジェクトを作成します。

```
$ oc apply -f - <<EOF
apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
metadata:
  name: default
  namespace: service-telemetry
spec: {}
EOF
```

デフォルト値を上書きするには、上書きするパラメーターを定義します。この例では、**enabled** を **true** に設定して Elasticsearch を有効にします。

```
$ oc apply -f - <<EOF
apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
metadata:
  name: default
  namespace: service-telemetry
spec:
  backends:
    events:
      elasticsearch:
        enabled: true
EOF
```

空の **spec** パラメーターを使用して **ServiceTelemetry** オブジェクトを作成すると、STF デプロイメントに以下のデフォルト値が設定されます。

```
apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
metadata:
  name: default
  namespace: service-telemetry
spec:
  alerting:
    alertmanager:
      receivers:
        snmpTraps:
          enabled: false
          target: 192.168.24.254
        storage:
          persistent:
            pvcStorageRequest: 20G
            strategy: persistent
          enabled: true
        backends:
          events:
            elasticsearch:
```

```
enabled: false
storage:
  persistent:
    pvcStorageRequest: 20Gi
    strategy: persistent
  version: 7.16.1
logs:
  loki:
    enabled: false
    flavor: 1x.extra-small
    replicationFactor: 1
    storage:
      objectStorageSecret: test
      storageClass: standard
metrics:
  prometheus:
    enabled: true
    scrapeInterval: 10s
    storage:
      persistent:
        pvcStorageRequest: 20G
        retention: 24h
        strategy: persistent
clouds:
- events:
  collectors:
  - collectorType: collectd
    debugEnabled: false
    subscriptionAddress: collectd/cloud1-notify
  - collectorType: ceilometer
    debugEnabled: false
    subscriptionAddress: anycast/ceilometer/cloud1-event.sample
metrics:
  collectors:
  - collectorType: collectd
    debugEnabled: false
    subscriptionAddress: collectd/cloud1-telemetry
  - collectorType: ceilometer
    debugEnabled: false
    subscriptionAddress: anycast/ceilometer/cloud1-metering.sample
  - collectorType: sensubility
    debugEnabled: false
    subscriptionAddress: sensubility/cloud1-telemetry
name: cloud1
graphing:
  enabled: false
grafana:
  adminPassword: secret
  adminUser: root
  baseImage: docker.io/grafana/grafana:latest
  disableSignoutMenu: false
  ingressEnabled: false
highAvailability:
  enabled: false
observabilityStrategy: use_community
transports:
```

```
qdr:
  enabled: true
web:
  enabled: false
```

これらのデフォルトを上書きするには、設定を **spec** パラメーターに追加します。

2. Service Telemetry Operator で STF デプロイメントログを表示します。

```
$ oc logs --selector name=service-telemetry-operator
...
----- Ansible Task Status Event StdOut -----

PLAY RECAP *****
localhost          :ok=90  changed=0  unreachable=0  failed=0  skipped=26
rescued=0  ignored=0
```

検証

- Pod および各 Pod のステータスを表示し、すべてのワークロードが正常に動作していることを確認するには、以下を実行します。



注記

backends.events.elasticsearch.enabled を **true** 設定した場合、通知スマートゲートウェイは Elasticsearch を開始するまでの時間のために **Error** と **CrashLoopBackOff** エラーメッセージを報告します。

```
$ oc get pods

NAME                                READY  STATUS   RESTARTS  AGE
alertmanager-default-0              2/2    Running  0          17m
default-cloud1-ceil-meter-smartgateway-6484b98b68-vd48z  2/2    Running  0          17m
default-cloud1-coll-meter-smartgateway-799f687658-4gxpn  2/2    Running  0          17m
default-cloud1-sens-meter-smartgateway-c7f4f7fc8-c57b4  2/2    Running  0          17m
default-interconnect-54658f5d4-pzrpt  1/1    Running  0          17m
elastic-operator-66b7bc49c4-sxkc2     1/1    Running  0          52m
interconnect-operator-69df6b9cb6-7hhp9  1/1    Running  0          50m
prometheus-default-0                2/2    Running  1          17m
prometheus-operator-6458b74d86-wbdqp   1/1    Running  0          51m
service-telemetry-operator-864646787c-hd9pm  1/1    Running  0          51m
smart-gateway-operator-79778cf548-mz5z7  1/1    Running  0          51m
```

3.2.1. ServiceTelemetry オブジェクトのパラメーター

ServiceTelemetry オブジェクトは、以下の主要な設定パラメーターで設定されます。

- **alerting**
- **バックエンド**
- **clouds**

- **graphing**
- **highAvailability**
- **transports**

これらの設定パラメーターをそれぞれ設定し、STF デプロイメントで異なる機能を提供できます。

バックエンドパラメーター

backends パラメーターを使用して、メトリクスおよびイベントの保存に使用できるストレージバックエンドを制御し、**clouds** パラメーターで定義されている Smart Gateway の有効化を制御します。詳細は、「[clouds パラメーター](#)」を参照してください。

現時点で、Prometheus をメトリクスストレージバックエンドとして、Elasticsearch をイベントストレージバックエンドとして使用できます。

メトリクスのストレージバックエンドとしての Prometheus の有効化

Prometheus をメトリクスのストレージバックエンドとして有効にするには、**ServiceTelemetry** オブジェクトを設定する必要があります。

手順

- **ServiceTelemetry** オブジェクトを設定します。

```
apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
metadata:
  name: default
  namespace: service-telemetry
spec:
  backends:
  metrics:
    prometheus:
      enabled: true
```

Prometheus に永続ストレージの設定

backends.metrics.prometheus.storage.persistent で定義されている追加のパラメーターを使用して、ストレージクラスやボリュームサイズなど、Prometheus の永続的なストレージオプションを設定します。

storageClass を使用して、バックエンドのストレージクラスを定義します。このパラメーターを設定しない場合、Service Telemetry Operator は Red Hat Open Shift Container Platform クラスターのデフォルトのストレージクラスを使用します。

pvcStorageRequest パラメーターを使用して、ストレージ要求を満たすために必要な最小のボリュームサイズを定義します。ボリュームが静的に定義されている場合は、要求されたよりも大きなボリュームサイズが使用される可能性があります。デフォルトでは、Service Telemetry Operator は **20G** (20 ギガバイト) のボリュームサイズを要求します。

手順

- 利用可能なストレージクラスを一覧表示します。

```
$ oc get storageclasses
NAME          PROVISIONER          RECLAIMPOLICY  VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION  AGE
```

csi-manila-ceph 20h	manila.csi.openstack.org	Delete	Immediate	false
standard (default) 20h	kubernetes.io/cinder	Delete	WaitForFirstConsumer	true
standard-csi 20h	cinder.csi.openstack.org	Delete	WaitForFirstConsumer	true

- **ServiceTelemetry** オブジェクトを設定します。

```

apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
metadata:
  name: default
  namespace: service-telemetry
spec:
  backends:
    metrics:
      prometheus:
        enabled: true
      storage:
        strategy: persistent
        persistent:
          storageClass: standard-csi
          pvcStorageRequest: 50G

```

Elasticsearch のイベントのストレージバックエンドとしての有効化

Elasticsearch をイベントのストレージバックエンドとして有効にするには、**ServiceTelemetry** オブジェクトを設定する必要があります。

手順

- **ServiceTelemetry** オブジェクトを設定します。

```

apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
metadata:
  name: default
  namespace: service-telemetry
spec:
  backends:
    events:
      elasticsearch:
        enabled: true

```

Elasticsearch のための永続的なストレージの設定

backends.events.elasticsearch.storage.persistent に定義されている追加のパラメーターを使用して、ストレージクラスやボリュームサイズなど、Elasticsearch の永続的なストレージオプションを設定します。

storageClass を使用して、バックエンドのストレージクラスを定義します。このパラメーターを設定しない場合、Service Telemetry Operator は Red Hat Open Shift Container Platform クラスターのデフォルトのストレージクラスを使用します。

pvcStorageRequest パラメーターを使用して、ストレージ要求を満たすために必要な最小のボリュームサイズを定義します。ボリュームが静的に定義されている場合は、要求されたよりも大きなボリューム

ムサイズが使用される可能性があります。デフォルトでは、Service Telemetry Operator は **20Gi** (20 ギビバイト) のボリュームサイズを要求します。

手順

- 利用可能なストレージクラスを一覧表示します。

```
$ oc get storageclasses
NAME                PROVISIONER                RECLAIMPOLICY  VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION  AGE
csi-manila-ceph     manila.csi.openstack.org  Delete         Immediate         false
20h
standard (default)  kubernetes.io/cinder      Delete         WaitForFirstConsumer  true
20h
standard-csi        cinder.csi.openstack.org  Delete         WaitForFirstConsumer  true
20h
```

- **ServiceTelemetry** オブジェクトを設定します。

```
apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
metadata:
  name: default
  namespace: service-telemetry
spec:
  backends:
  events:
    elasticsearch:
      enabled: true
      version: 7.16.1
    storage:
      strategy: persistent
      persistent:
        storageClass: standard-csi
        pvcStorageRequest: 50G
```

clouds パラメーター

clouds パラメーターを使用して、デプロイされる Smart Gateway オブジェクトを提議し、STF のインスタンスに接続する、複数の監視対象のクラウド環境にインターフェイスを提供されます。サポートするバックエンドが利用可能な場合に、デフォルトのクラウド設定のメトリクスおよびイベント Smart Gateway が作成されます。デフォルトで、Service Telemetry Operator は **cloud1** の Smart Gateway を作成します。

クラウドオブジェクトの一覧を作成して、定義されたクラウドに作成される Smart Gateway を制御できます。各クラウドはデータタイプとコレクターで設定されます。データタイプは **metrics** または **events** イベントです。各データタイプは、コレクターの一覧、メッセージバスサブスクリプションアドレス、およびデバッグを有効にするパラメーターで設定されます。メトリックに使用できるコレクターは、**collectd**、**ceilometer**、および **sensubility** です。イベントで利用可能なコレクターは **collectd** および **ceilometer** です。これらのコレクターのサブスクリプションアドレスは、クラウド、データタイプ、コレクターの組み合わせごとに一意であることを確認してください。

デフォルトの **cloud1** 設定は、特定のクラウドインスタンスの collectd、Ceilometer、および Sensubility データコレクターのメトリクスおよびイベントのサブスクリプションおよびデータストレージを提供する以下の **ServiceTelemetry** オブジェクトによって表されます。

```

apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
metadata:
  name: stf-default
  namespace: service-telemetry
spec:
  clouds:
    - name: cloud1
      metrics:
        collectors:
          - collectorType: collectd
            subscriptionAddress: collectd/telemetry
          - collectorType: ceilometer
            subscriptionAddress: anycast/ceilometer/metering.sample
          - collectorType: sensubility
            subscriptionAddress: sensubility/telemetry
            debugEnabled: false
      events:
        collectors:
          - collectorType: collectd
            subscriptionAddress: collectd/notify
          - collectorType: ceilometer
            subscriptionAddress: anycast/ceilometer/event.sample

```

clouds パラメーターの各項目はクラウドインスタンスを表します。クラウドインスタンスは、**name**、**metrics**、および **events** の3つの最上位のパラメーターで設定されます。**metrics** および **events** パラメーターは、対象のデータタイプのストレージに対応するバックエンドを表します。**collectors** パラメーターは、2つの必須パラメーター **collectorType** と **subscriptionAddress** で設定されるオブジェクトの一覧を指定し、これらは Smart Gateway のインスタンスを表します。**collectorType** パラメーターは、collectd、Ceilometer、または Sensubility のいずれかによって収集されるデータを指定します。**subscriptionAddress** パラメーターは、Smart Gateway がサブスクライブする AMQ Interconnect アドレスを提供します。

collectors パラメーター内でオプションのブール値パラメーター **debugEnabled** を使用して、実行中の Smart Gateway Pod で追加のコンソールのデバッグを有効にすることができます。

関連情報

- デフォルトの Smart Gateway の削除に関する詳細は、「[デフォルトの Smart Gateway を削除](#)」を参照してください。
- 複数のクラウドを設定する方法は、「[複数のクラウドの設定](#)」を参照してください。

alerting パラメーター

alerting パラメーターを使用して、Alertmanager インスタンスの作成とストレージバックエンドの設定を制御します。デフォルトでは **alerting** は有効になっています。詳細は、「[Service Telemetry Framework でのアラート](#)」を参照してください。

graphing パラメーター

graphing パラメーターを使用して Grafana インスタンスの作成を制御します。デフォルトでは、**graphing** は無効になっています。詳細は、「[Service Telemetry Framework でのダッシュボード](#)」を参照してください。

highAvailability パラメーター

highAvailability パラメーターを使用して複数の STF コンポーネントコピーのインスタンス化を制御し、失敗または再スケジュールされたコンポーネントの復旧時間を短縮します。デフォルトで、**highAvailability** は無効になっています。詳細は、「[高可用性](#)」を参照してください。

transports パラメーター

STF デプロイメントに対するメッセージバスの有効化を制御するには、**transports** パラメーターを使用します。現在サポートされているトランスポートは AMQ Interconnect のみです。デフォルトでは、**qdr** トランスポートが有効です。

3.3. STF コンポーネントのユーザーインターフェイスへのアクセス

Red Hat OpenShift Container Platform では、アプリケーションはルートを通じて外部ネットワークに公開されます。ルートについての詳細は、[Configuring ingress cluster traffic](#) を参照してください。

Service Telemetry Framework (STF) では、HTTPS ルートは Web ベースのインターフェイスを持つサービスごとに公開されます。これらのルートは、Red Hat OpenShift Container Platform RBAC によって保護されており、Red Hat OpenShift Container Platform 名前空間を表示できる

ClusterRoleBinding を持つすべてのユーザーがログインできます。RBAC の詳細は、[Using RBAC to define and apply permissions](#) を参照してください。

手順

1. Red Hat OpenShift Container Platform にログインします。
2. **service-telemetry** namespace に切り替えます。

```
$ oc project service-telemetry
```

3. **service-telemetry** プロジェクトで利用可能な Web UI ルートを一覧表示します。

```
$ oc get routes | grep web
default-alertmanager-proxy default-alertmanager-proxy-service-telemetry.apps.infra.watch
default-alertmanager-proxy web reencrypt/Redirect None
default-prometheus-proxy default-prometheus-proxy-service-telemetry.apps.infra.watch
default-prometheus-proxy web reencrypt/Redirect None
```

4. Web ブラウザーで https://<route_address> に移動し、対応するサービスの Web インターフェイスにアクセスします。

3.4. 代替可観測性ストラテジーの設定

ストレージ、可視化、およびアラートバックエンドのデプロイメントを省略するように STF を設定するには、ServiceTelemetry 仕様に **observabilityStrategy: none** を追加します。このモードでは、AMQ Interconnect ルーターおよびメトリクス Smart Gateway のみがデプロイされ、外部の Prometheus 互換システムを設定して、STF Smart Gateways からメトリクスを収集する必要があります。



注記

現在、**observabilityStrategy** を **none** に設定すると、メトリクスのみがサポートされません。Events Smart Gateways はデプロイされません。

手順

1. **spec** パラメーターに **observabilityStrategy: none** プロパティを指定して

ServiceTelemetry オブジェクトを作成します。マニフェストは、すべてのメトリクスコレクタータイプを持つ単一のクラウドから Telemetry を受け取るのに適した STF のデフォルトデプロイメントを示しています。

```
$ oc apply -f - <<EOF
apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
metadata:
  name: default
  namespace: service-telemetry
spec:
  observabilityStrategy: none
EOF
```

2. すべてのワークロードが適切に動作していることを確認するには、Pod および各 Pod のステータスを確認します。

```
$ oc get pods
NAME                                READY STATUS RESTARTS AGE
default-cloud1-ceil-meter-smartgateway-59c845d65b-gzhcs 3/3 Running 0      132m
default-cloud1-coll-meter-smartgateway-75bbd948b9-d5phm 3/3 Running 0      132m
default-cloud1-sens-meter-smartgateway-7fdbb57b6d-dh2g9 3/3 Running 0      132m
default-interconnect-668d5bbcd6-57b2l                1/1 Running 0      132m
interconnect-operator-b8f5bb647-tlp5t                1/1 Running 0      47h
service-telemetry-operator-566b9dd695-wkvjq          1/1 Running 0      156m
smart-gateway-operator-58d77dcf7-6xsq7               1/1 Running 0      47h
```

関連情報

追加のクラウドの設定、またはサポート対象のコレクターのセットの変更に関する詳細は、[「Smart Gateway の導入」](#) を参照してください。

第4章 サービス TELEMETRY フレームワーク向けの RED HAT OPENSTACK PLATFORM の設定

メトリクス、イベント、またはその両方のコレクション、および Service Telemetry Framework (STF) ストレージドメインに送信するには、Red Hat OpenStack Platform (RHOSP) オーバークラウドを設定して、データ収集とトランスポートを有効にする必要があります。

STF は単一クラウドと複数のクラウドの両方をサポートすることができます。RHOSP と STF のデフォルトの設定は、単一のクラウドのインストールのために設定されています。

- デフォルトの設定での単一の RHOSP オーバークラウドの展開については、「[Service Telemetry Framework 向けの Red Hat OpenStack Platform オーバークラウドのデプロイ](#)」を参照してください。
- RHOSP のインストールと設定 STF を複数のクラウドで計画するには、「[複数のクラウドの設定](#)」を参照してください。
- RHOSP のオーバークラウド導入の一環として、お客様の環境で追加の機能を設定する必要がある場合があります。
 - 分散コンピュートノード (DCN) やスパイン/リーフなど、ルーティング対応 L3 ドメインを使用する RHOSP クラウドノードで、データ収集と STF への転送をデプロイするには、「[標準外のネットワークポロジへのデプロイメント](#)」を参照してください。
 - データコレクターサービスを無効にするには、次を参照してください。「[Service Telemetry Framework で使用される Red Hat OpenStack Platform サービスの無効化](#)」
 - コンテナイメージをローカルレジストリーに同期した場合は、環境ファイルを作成し、コンテナイメージへのパスを追加する必要があります。詳細は、「[アンダークラウドへの Red Hat OpenStack Platform コンテナイメージの追加](#)」を参照してください。

4.1. SERVICE TELEMETRY FRAMEWORK 向けの RED HAT OPENSTACK PLATFORM オーバークラウドのデプロイ

Red Hat OpenStack Platform (RHOSP) オーバークラウドのデプロイメントの一環として、データコレクターおよびデータトランスポートを Service Telemetry Framework (STF) に設定する必要があります。

手順

1. 「[オーバークラウドの設定向け Service Telemetry Framework からの CA 証明書の取得](#)」
2. [AMQ Interconnect ルートアドレスの取得](#)
3. [STF の基本設定の作成](#)
4. [オーバークラウドの STF 接続の設定](#)
5. [オーバークラウドのデプロイ](#)
6. [クライアント側のインストールの検証](#)

4.1.1. オーバークラウドの設定向け Service Telemetry Framework からの CA 証明書の取得

Red Hat OpenStack Platform オーバークラウドを Service Telemetry Framework (STF) に接続するには、Service Telemetry Framework 内で実行される AMQ Interconnect の CA 証明書を取得し、Red Hat OpenStack Platform 設定で証明書を使用します。

手順

1. Service Telemetry Framework で利用可能な証明書の一覧を表示します。

```
$ oc get secrets
```

2. **default-interconnect-selfsigned** シークレットの内容を取得して書き留めます。

```
$ oc get secret/default-interconnect-selfsigned -o jsonpath='{.data.ca.crt}' | base64 -d
```

4.1.2. AMQ Interconnect ルートアドレスの取得

Service Telemetry Framework (STF) 向けに Red Hat OpenStack Platform (RHOSP) オーバークラウドを設定する場合には、STF 接続ファイルに AMQ Interconnect ルートアドレスを指定する必要があります。

手順

1. Red Hat OpenShift Container Platform 環境にログインします。
2. **service-telemetry** プロジェクトで、AMQ Interconnect ルートアドレスを取得します。

```
$ oc get routes -ogo-template='{{ range .items }}{{printf "%s\n" .spec.host }}{{ end }}' | grep "\-5671"
default-interconnect-5671-service-telemetry.apps.infra.watch
```

4.1.3. STF の基本設定の作成

Service Telemetry Framework (STF) と互換性があるデータ収集とトランスポートを提供するようにベースパラメーターを設定するには、デフォルトのデータ収集値を定義するファイルを作成する必要があります。

手順

1. Red Hat OpenStack Platform (RHOSP) 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 **enable-stf.yaml** という名前の設定ファイルを作成します。



重要

EventPipelinePublishers および **PipelinePublishers** を空のリストに設定すると、Gnocchi や Panko などの RHOSP Telemetry コンポーネントにイベントやメトリクスデータを渡すことはありません。追加のパイプラインにデータを送信する必要がある場合は、**ExtraConfig** で指定されるように、Ceilometer の 30 秒のポーリング間隔は、RHOSP のテレメトリーコンポーネントに圧倒され、**300** などの大きな値に間隔を引き上げる必要があります。ポーリング間隔の値を増やすと、STF の Telemetry 解決が少なくなります。

enable-stf.yaml

```
parameter_defaults:
  # only send to STF, not other publishers
  EventPipelinePublishers: []
  PipelinePublishers: []

  # manage the polling and pipeline configuration files for Ceilometer agents
  ManagePolling: true
  ManagePipeline: true

  # enable Ceilometer metrics and events
  CeilometerQdrPublishMetrics: true
  CeilometerQdrPublishEvents: true

  # set collectd overrides for higher telemetry resolution and extra plugins to load
  CollectdConnectionType: amqp1
  CollectdAmqpInterval: 5
  CollectdDefaultPollingInterval: 5
  CollectdExtraPlugins:
    - vmem

  # set standard prefixes for where metrics and events are published to QDR
  MetricsQdrAddresses:
    - prefix: 'collectd'
      distribution: multicast
    - prefix: 'anycast/ceilometer'
      distribution: multicast

  ExtraConfig:
    ceilometer::agent::polling::polling_interval: 30
    ceilometer::agent::polling::polling_meters:
      - cpu
      - disk.*
      - ip.*
      - image.*
      - memory
      - memory.*
      - network.*
      - perf.*
      - port
      - port.*
      - switch
      - switch.*
      - storage.*
      - volume.*

  # to avoid filling the memory buffers if disconnected from the message bus
  # note: Adjust the value of the `send_queue_limit` to handle your required volume of
  # metrics.
  collectd::plugin::amqp1::send_queue_limit: 5000

  # receive extra information about virtual memory
  collectd::plugin::vmem::verbose: true

  # set memcached collectd plugin to report its metrics by hostname
```

```
# rather than host IP, ensuring metrics in the dashboard remain uniform
collectd::plugin::memcached::instances:
  local:
    host: "%{hiera('fqdn_canonical')}}"
    port: 11211

# align defaults across OSP versions
collectd::plugin::cpu::reportbycpu: true
collectd::plugin::cpu::reportbystate: true
collectd::plugin::cpu::reportnumcpu: false
collectd::plugin::cpu::valuespercentage: true
collectd::plugin::df::ignoreselected: true
collectd::plugin::df::reportbydevice: true
collectd::plugin::df::fstypes: ['xfs']
collectd::plugin::load::reportrelative: true
collectd::plugin::virt::extra_stats: "pcpu cpu_util vcpupin vcpu memory disk disk_err
disk_allocation disk_capacity disk_physical domain_state job_stats_background perf"
```

4.1.4. オーバークラウドの STF 接続の設定

Service Telemetry Framework (STF) 接続を設定するには、オーバークラウド用の AMQ Interconnect の接続設定など、ファイルを STF デプロイメントに対して作成する必要があります。イベントの収集と STF への保存を有効にし、オーバークラウドを展開します。デフォルト設定は、デフォルトのメッセージバストピックを使用して単一のクラウドインスタンスに対して指定されます。複数のクラウドのデプロイメントの設定については、「[複数のクラウドの設定](#)」を参照してください。

前提条件

- STF によってデプロイされる AMQ Interconnect から CA 証明書を取得します。詳細は、「[オーバークラウドの設定向け Service Telemetry Framework からの CA 証明書の取得](#)」を参照してください。
- AMQ インターコネクトのルートアドレスを取得します。詳細は、「[AMQ Interconnect ルートアドレスの取得](#)」を参照してください。

手順

1. RHOSP のアンダークラウドに **stack** ユーザーとしてログインします。
2. **/home/stack** ディレクトリーに **stf-connectors.yaml** という設定ファイルを作成します。
3. **stf-connectors.yaml** ファイルで、オーバークラウド上の AMQ Interconnect を STF デプロイメントに接続するように **MetricsQdrConnectors** アドレスを設定します。このファイルの Sensubility、Ceilometer、および collectd のトピックアドレスを、STF のデフォルトに一致するように設定します。トピックおよびクラウド設定のカスタマイズに関する詳細は、「[複数のクラウドの設定](#)」を参照してください。
 - 複数のクラウドデプロイメント用に **collectd-write-qdr.yaml** 環境ファイルを含めないため、**resource_registry** 設定は collectd サービスを直接ロードします。
 - **host** パラメーターは、「[AMQ Interconnect ルートアドレスの取得](#)」で取得した **HOST/PORT** の値に置き換えます。
 - **caCertFileContent** パラメーターを、「[オーバークラウドの設定向け Service Telemetry Framework からの CA 証明書の取得](#)」で取得したコンテンツに置き換えます。

- **MetricsQdrConnectors** の **host** サブパラメーターを、取得した **HOST/PORT** の値に置き換えます。「[AMQ Interconnect ルートアドレスの取得](#)」。
- **CeilometerQdrEventsConfig.topic** を設定して、Ceilometer イベントのトピックを定義します。この値の形式は **anycast/ceilometer/cloud1-event.sample** です。
- **CeilometerQdrMetricsConfig.topic** を設定して、Ceilometer メトリックのトピックを定義します。この値のフォーマットは、**anycast/ceilometer/cloud1-metering.sample** です。
- **CollectdAmqpInstances** を設定して、collectd イベントのトピックを定義します。この値の形式は、**collectd/cloud1-notify** です。
- **CollectdAmqpInstances** を設定して、collectd メトリックのトピックを定義します。この値の形式は **collectd/cloud1-telemetry** です。

stf-connectors.yaml

```
resource_registry:
  OS::TripleO::Services::Collectd: /usr/share/openstack-tripleo-heat-
  templates/deployment/metrics/collectd-container-puppet.yaml

parameter_defaults:
  MetricsQdrConnectors:
    - host: stf-default-interconnect-5671-service-telemetry.apps.infra.watch
      port: 443
      role: edge
      verifyHostname: false
      sslProfile: sslProfile

  MetricsQdrSSLProfiles:
    - name: sslProfile
      caCertFileContent: |
        -----BEGIN CERTIFICATE-----
        <snip>
        -----END CERTIFICATE-----

  CeilometerQdrEventsConfig:
    driver: amqp
    topic: cloud1-event

  CeilometerQdrMetricsConfig:
    driver: amqp
    topic: cloud1-metering

  CollectdAmqpInstances:
    cloud1-notify:
      notify: true
      format: JSON
      presettle: false
    cloud1-telemetry:
      format: JSON
      presettle: false
```

4.1.5. オーバークラウドのデプロイ

必要な環境ファイルでオーバークラウドをデプロイまたは更新すると、データが収集されて、Service Telemetry Framework (STF) に送信されます。

手順

1. Red Hat OpenStack Platform (RHOSP) 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. 認証ファイルのソース

```
[stack@undercloud-0 ~]$ source stackrc
(undercloud) [stack@undercloud-0 ~]$
```

3. RHOSP director デプロイメントに以下のファイルを追加して、データ収集と AMQ Interconnect を設定します。
 - Ceilometer Telemetry およびイベントが STF に送信されることを確認する **ceilometer-write-qdr.yaml** ファイル
 - メッセージバスが有効で、STF メッセージバスマルターに接続されていることを確認する **qdr-edge-only.yaml** ファイル
 - デフォルト値が正しく設定されていることを確認する **enable-stf.yaml** 環境ファイル
 - STF への接続を定義する **stf-connectors.yaml** 環境ファイル
4. RHOSP オーバークラウドをデプロイします。

```
(undercloud) [stack@undercloud-0 ~]$ 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/ceilometer-write-qdr.yaml \
--environment-file /usr/share/openstack-tripleo-heat-templates/environments/metrics/qdr-edge-only.yaml \
--environment-file /home/stack/enable-stf.yaml \
--environment-file /home/stack/stf-connectors.yaml
```

4.1.6. クライアント側のインストールの検証

Service Telemetry Framework (STF) ストレージドメインからデータ収集を検証するには、配信されたデータに対してデータソースをクエリーします。Red Hat OpenStack Platform (RHOSP) デプロイメントの個別ノードを検証するには、SSH を使用してコンソールに接続します。

ヒント

一部のテレメトリデータは、RHOSP にアクティブなワークロードがある場合にのみ利用可能です。

手順

1. オーバークラウドノード (例: controller-0) にログインします。
2. **metrics_qdr** コンテナがノードで実行されていることを確認します。

```
$ sudo docker container inspect --format '{{.State.Status}}' metrics_qdr
running
```

3. AMQ Interconnect が実行されている内部ネットワークアドレスを返します (ポート **5666** でリスンする **172.17.1.44** など)。

```
$ sudo docker exec -it metrics_qdr cat /etc/qpid-dispatch/qdrouterd.conf

listener {
  host: 172.17.1.44
  port: 5666
  authenticatePeer: no
  saslMechanisms: ANONYMOUS
}
```

4. ローカルの AMQ インターコネクトへの接続のリストを返します。

```
$ sudo docker exec -it metrics_qdr qdstat --bus=172.17.1.44:5666 --connections

Connections
 id host                               container
 role dir security                     authentication tenant
=====
=====
=====
=====
 1 default-interconnect-5671-service-telemetry.apps.infra.watch:443 default-
interconnect-7458fd4d69-bgzfb                               edge out
  TLSv1.2(DHE-RSA-AES256-GCM-SHA384) anonymous-user
 12 172.17.1.44:60290
  openstack.org/om/container/controller-0/ceilometer-agent-
notification/25/5c02cee550f143ec9ea030db5cccba14 normal in no-security
no-auth
 16 172.17.1.44:36408                               metrics
normal in no-security                               anonymous-user
 899 172.17.1.44:39500                               10a2e99d-1b8a-4329-b48c-
4335e5f75c84 normal in no-security
no-auth
```

接続は 4 つあります。

- STF へのアウトバウンド接続
- ceilometer からのインバウンド接続
- collectd からのインバウンド接続
- **qdstat** クライアントからの受信接続
STF の送信接続は **MetricsQdrConnectors** ホストパラメーターに提供され、STF ストレージドメインのルートとなります。他のホストは、この AMQ インターコネクトへのクライアント接続の内部ネットワークアドレスです。

5. メッセージが配信されていることを確認するには、リンクを一覧表示してメッセージ配信の **deliv** 列に **_edge** アドレスを表示します。

```
$ sudo docker exec -it metrics_qdr qdstat --bus=172.17.1.44:5666 --links
Router Links
  type  dir conn id id  peer class  addr          phs cap pri undel unsett deliv
  presett psdrop acc rej rel  mod delay rate
=====
=====
=====
endpoint out 1 5 local _edge 250 0 0 0 2979926 0 0
0 0 2979926 0 0 0
endpoint in 1 6 250 0 0 0 0 0 0 0 0
0 0 0 0
endpoint in 1 7 250 0 0 0 0 0 0 0 0
0 0 0 0
endpoint out 1 8 250 0 0 0 0 0 0 0 0
0 0 0 0
endpoint in 1 9 250 0 0 0 0 0 0 0 0
0 0 0 0
endpoint out 1 10 250 0 0 0 911 911 0 0
0 0 0 911 0
endpoint in 1 11 250 0 0 0 0 911 0 0
0 0 0 0 0
endpoint out 12 32 local temp.ISY6Mccol4J2Kp 250 0 0 0 0 0
0 0 0 0 0 0 0
endpoint in 16 41 250 0 0 0 2979924 0 0
0 0 2979924 0 0 0
endpoint in 912 1834 mobile $management 0 250 0 0 0 1 0
0 1 0 0 0 0 0
endpoint out 912 1835 local temp.9Ok2resI9tmt+CT 250 0 0 0 0
0 0 0 0 0 0 0
```

6. RHOSP ノードから STF へのアドレスを一覧表示するには、Red Hat OpenShift Container Platform に接続して AMQ Interconnect Pod 名を取得し、接続を一覧表示します。利用可能な AMQ Interconnect Pod を一覧表示します。

```
$ oc get pods -l application=default-interconnect

NAME                                READY STATUS RESTARTS AGE
default-interconnect-7458fd4d69-bgzfb 1/1 Running 0 6d21h
```

7. Pod に接続し、既知の接続を一覧表示します。この例では、RHOSP ノードから接続 **id** 22、23、および 24 の 3 つの **edge** 接続があります。

```
$ oc exec -it default-interconnect-7458fd4d69-bgzfb -- qdstat --connections

2020-04-21 18:25:47.243852 UTC
default-interconnect-7458fd4d69-bgzfb

Connections
  id host          container          role  dir security
authentication tenant last dlv  uptime
```

```

=====
=====
=====
5 10.129.0.110:48498 bridge-3f5 edge in no-security
anonymous-user 000:00:00:02 000:17:36:29
6 10.129.0.111:43254 rcv[default-cloud1-ceil-meter-smartgateway-58f885c76d-xmwn]
edge in no-security anonymous-user 000:00:00:02 000:17:36:20
7 10.130.0.109:50518 rcv[default-cloud1-coll-event-smartgateway-58fbbd4485-rl9bd]
normal in no-security anonymous-user - 000:17:36:11
8 10.130.0.110:33802 rcv[default-cloud1-ceil-event-smartgateway-6cfb65478c-g5q82]
normal in no-security anonymous-user 000:01:26:18 000:17:36:05
22 10.128.0.1:51948 Router.ceph-0.redhat.local edge in
TLSv1/SSLv3(DHE-RSA-AES256-GCM-SHA384) anonymous-user 000:00:00:03
000:22:08:43
23 10.128.0.1:51950 Router.compute-0.redhat.local edge in
TLSv1/SSLv3(DHE-RSA-AES256-GCM-SHA384) anonymous-user 000:00:00:03
000:22:08:43
24 10.128.0.1:52082 Router.controller-0.redhat.local edge in
TLSv1/SSLv3(DHE-RSA-AES256-GCM-SHA384) anonymous-user 000:00:00:00
000:22:08:34
27 127.0.0.1:42202 c2f541c1-4c97-4b37-a189-a396c08fb079 normal in
no-security no-auth 000:00:00:00 000:00:00:00

```

8. ネットワークによって配信されるメッセージ数を表示するには、各アドレスを **oc exec** コマンドで使用します。

```

$ oc exec -it default-interconnect-7458fd4d69-bgzfb -- qdstat --address

2020-04-21 18:20:10.293258 UTC
default-interconnect-7458fd4d69-bgzfb

Router Addresses
class addr          phs distrib  pri local remote in      out      thru
fallback

=====
=====
mobile anycast/ceilometer/event.sample 0 balanced - 1 0 970 970
0 0
mobile anycast/ceilometer/metering.sample 0 balanced - 1 0 2,344,833
2,344,833 0 0
mobile collectd/notify 0 multicast - 1 0 70 70 0 0
mobile collectd/telemetry 0 multicast - 1 0 216,128,890 216,128,890
0 0

```

4.2. SERVICE TELEMETRY FRAMEWORK で使用される RED HAT OPENSTACK PLATFORM サービスの無効化

Red Hat OpenStack Platform (RHOSP) をデプロイして Service Telemetry Framework (STF) に接続するとき使用されるサービスを無効にします。サービスの無効化の一部として、ログまたは生成された設定ファイルが削除されることはありません。

手順

1. RHOSP のアンダークラウドに **stack** ユーザーとしてログインします。
2. 認証ファイルのソース

```
[stack@undercloud-0 ~]$ source stackrc
(undercloud) [stack@undercloud-0 ~]$
```

3. **disable-stf.yaml** 環境ファイルを作成します。

```
(undercloud) [stack@undercloud-0]$ cat > $HOME/disable-stf.yaml <<EOF
---
resource_registry:
  OS::TripleO::Services::CeilometerAgentCentral: OS::Heat::None
  OS::TripleO::Services::CeilometerAgentNotification: OS::Heat::None
  OS::TripleO::Services::CeilometerAgentIpmi: OS::Heat::None
  OS::TripleO::Services::ComputeCeilometerAgent: OS::Heat::None
  OS::TripleO::Services::Redis: OS::Heat::None
  OS::TripleO::Services::Collectd: OS::Heat::None
  OS::TripleO::Services::MetricsQdr: OS::Heat::None
EOF
```

4. RHOSP director デプロイメントから以下のファイルを削除します。

- **ceilometer-write-qdr.yaml**
- **qdr-edge-only.yaml**
- **enable-stf.yaml**
- **stf-connectors.yaml**

5. RHOSP オーバークラウドを更新します。環境ファイルのリストの早い段階で **disable-stf.yaml** ファイルを使用していることを確認してください。リストの早い段階で **disable-stf.yaml** を追加することにより、他の環境ファイルは、サービスを無効にする設定をオーバーライドできません。

```
(undercloud) [stack@undercloud-0 ~]$ openstack overcloud deploy <other_arguments>
--templates /usr/share/openstack-tripleo-heat-templates \
--environment-file /home/stack/disable-stf.yaml
--environment-file <other_environment_files> \
```

4.3. 標準外のネットワークトポロジーへのデプロイメント

ノードがデフォルトの **InternalApi** ネットワークとは別のネットワーク上にある場合は、AMQ Interconnect がデータを Service Telemetry Framework (STF) サーバーインスタンスに転送できるように、設定の調整を行う必要があります。このシナリオはスパイン/リーフや DCN トポロジーで典型的です。DCN の設定の詳細は、[スパイン/リーフ型ネットワーク](#) を参照してください。

Red Hat OpenStack Platform (RHOSP) 13.0 で STF を使用し、Ceph、ブロック、またはオブジェクトストレージノードを監視する予定の場合には、スパイン/リーフ型ネットワークおよび DCN ネットワーク設定の変更と同様の設定変更を行う必要があります。Ceph ノードを監視するには、**CephStorageExtraConfig** パラメーターを使用して、AMQ Interconnect および collectd 設定ファイルに読み込むネットワークインターフェイスを定義します。


```
CephStorageExtraConfig:
```

```
tripleo::profile::base::metrics::collectd::amqp_host: "%{hiera('storage')}}"
tripleo::profile::base::metrics::qdr::listener_addr: "%{hiera('storage')}}"
tripleo::profile::base::ceilometer::agent::notification::notifier_host_addr: "%{hiera('storage')}}"
```

同様に、環境が Block および Object Storage ロールを使用する場合は、**BlockStorageExtraConfig** および **ObjectStorageExtraConfig** パラメーターを指定する必要があります。

スパイン/リーフ型トポロジをデプロイするには、ロールとネットワークを作成し、それらのネットワークを利用可能なロールに割り当てる必要があります。RHOSP デプロイメントで STF のデータ収集およびトランスポートを設定する場合には、ロールのデフォルトのネットワークは **InternalApi** になります。Ceph、ブロックおよびオブジェクトストレージロールの場合には、デフォルトのネットワークは **Storage** です。スパイン/リーフ型設定により、異なるネットワークが異なるリーフグループに割り当てられ、その名前は通常一意となるので、RHOSP 環境ファイルの **parameter_defaults** セクションで追加の設定が必要になります。

手順

1. Leaf ロールごとにどのネットワークが利用できるかを記録します。ネットワーク名の定義例は、**Spine Leaf Networking**の [ネットワークデータファイルの作成](#) を参照してください。Leaf Group (ロール) の作成と、これらのグループへのネットワークの割り当ての詳細は、**Spine Leaf Networking**の [Creating a roles data file](#) を参照してください。
2. 各 Leaf ロールの **ExtraConfig** セクションに以下の設定例を追加します。この例では、**internal_api0** はネットワーク定義の **name_lower** パラメーターで定義された値で、**Compute0** leaf ロールが接続されているネットワークです。この場合、ネットワーク識別の 0 は、Leaf 0 の Compute ロールに対応し、デフォルトの内部 API ネットワーク名とは異なる値を表しています。

Compute0 leaf ロールには、hiera ルックアップを実行して collectd AMQP ホストパラメーターに割り当てるネットワークインターフェイスを決定するために、追加の設定を指定します。AMQ Interconnect のリスナーアドレスパラメーターについても同様の設定を行います。

```
Compute0ExtraConfig:
```

```
tripleo::profile::base::metrics::collectd::amqp_host: "%{hiera('internal_api0')}}"
tripleo::profile::base::metrics::qdr::listener_addr: "%{hiera('internal_api0')}}"
```

この設定例は、Ceph Storage の Leaf ロール上のものです。

```
CephStorage0ExtraConfig:
```

```
tripleo::profile::base::metrics::collectd::amqp_host: "%{hiera('storage0')}}"
tripleo::profile::base::metrics::qdr::listener_addr: "%{hiera('storage0')}}"
```

4.4. アンダークラウドへの RED HAT OPENSTACK PLATFORM コンテナイメージの追加

コンテナイメージをローカルレジストリーに同期した場合は、環境ファイルを作成し、コンテナイメージへのパスを追加する必要があります。

手順

1. 新しい環境ファイル (例: **container-images.yaml**) を作成し、以下のコンテナイメージとパスを挿入します。

```
parameter_defaults:
  DockerCollectdConfigImage: <image-registry-path>/rhosp{vernum}/openstack-
collectd:latest
  DockerCollectdImage: <image-registry-path>/rhosp{vernum}/openstack-collectd:latest
  DockerMetricsQdrConfigImage: <image-registry-path>/rhosp{vernum}/openstack-
qdrouterd:latest
  DockerMetricsQdrImage: <image-registry-path>/rhosp{vernum}/openstack-qdrouterd:latest
```

<image-registry-path> をイメージレジストリーへのパスに置き換えます。

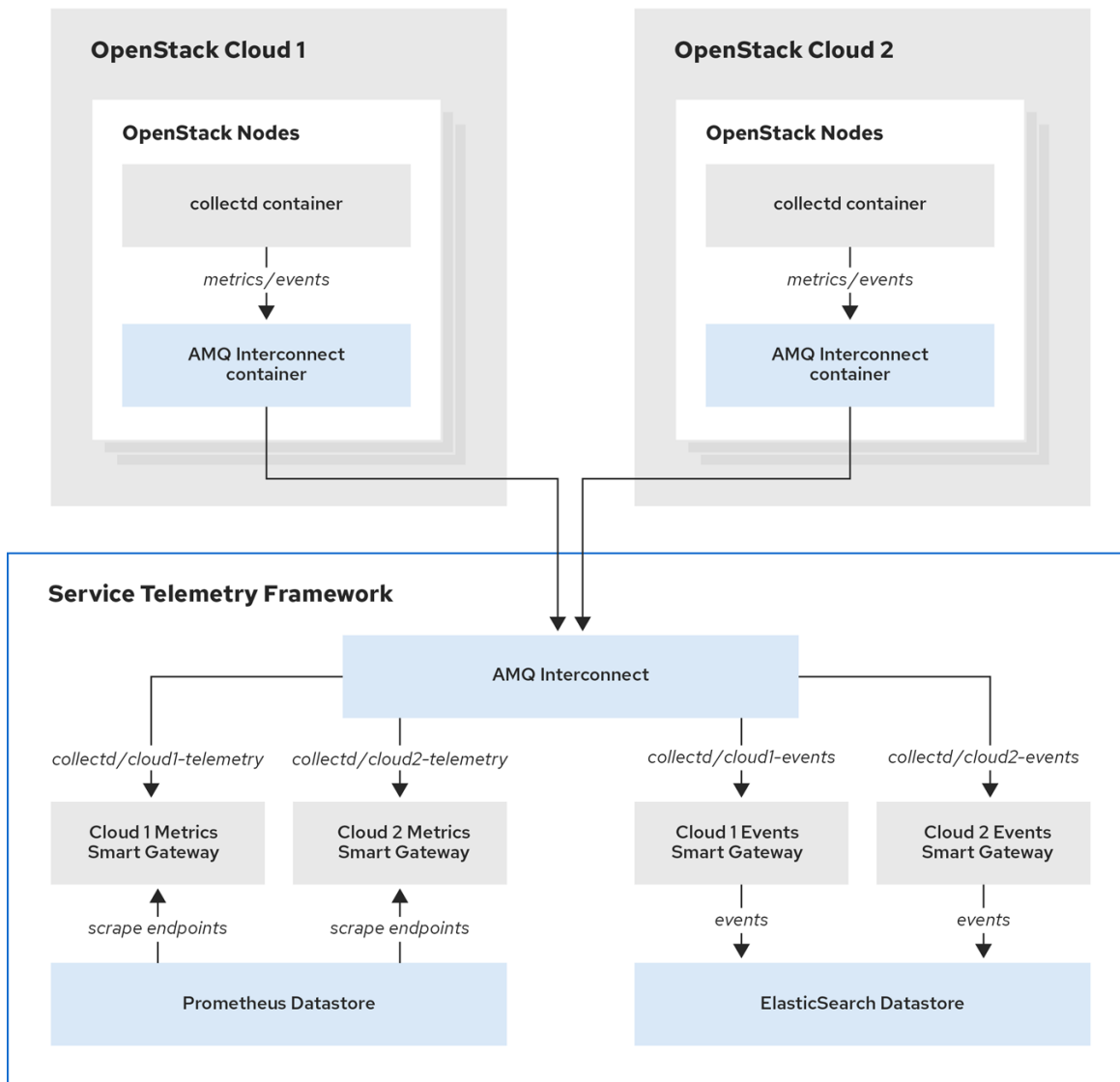
- Red Hat OpenStack Platform director はイメージを準備することができるため、collectd と AMQ Interconnect をオーバークラウドにデプロイするために、**/usr/share/local-container-images/container-images.yaml** 環境ファイルが含まれています。以下のスニペットは、この環境ファイルの追加方法の例を示しています。

```
$ openstack overcloud container image prepare \
...
-e /usr/share/local-container-images/container-images.yaml \
...
```

4.5. 複数のクラウドの設定

Service Telemetry Framework (STF) の単一インスタンスをターゲットにするように複数の Red Hat OpenStack Platform (RHOSP) クラウドを設定できます。複数のクラウドを設定する場合に、クラウドはすべて独自で一意的なメッセージバストピックでメトリクスおよびイベントを送信する必要があります。STF デプロイメントでは、Smart Gateway インスタンスは、これらのトピックをリッスンして、共通のデータストアに情報を保存します。データストレージドメインの Smart Gateway によって保存されるデータは、各 Smart Gateway が作成するメタデータを使用してフィルターされます。

図4.1 RHOSP の2つのクラウドが STF に接続



65_OpenStack_0120

複数のクラウドのシナリオで RHOSP オーバークラウドを設定するには、次のタスクを実行します。

1. 各クラウドで使用する AMQP アドレスの接頭辞を計画します。詳細は、「[AMQP アドレス接頭辞の計画](#)」を参照してください。
2. メトリクスとイベントのコンシューマーである Smart Gateway を各クラウドに配備し、対応するアドレス接頭辞をリッスンします。詳細は、「[Smart Gateway の導入](#)」を参照してください。
3. 各クラウドに固有のドメイン名を設定します。詳細は、「[独自のクラウドドメインの設定](#)」を参照してください。
4. STF の基本設定を作成します。詳細は、「[STF の基本設定の作成](#)」を参照してください。
5. 各クラウドがメトリクスやイベントを正しいアドレスで STF に送信するように設定します。詳細は、「[複数クラウドの Red Hat OpenStack Platform 環境ファイルの作成](#)」を参照してください。

4.5.1. AMQP アドレス接頭辞の計画

デフォルトでは、Red Hat OpenStack Platform ノードは、collectd および Ceilometer の 2 つのデータコレクターを使用してデータを受信します。これらのコンポーネントは、Telemetry データまたは通知をそれぞれの AMQP アドレス (例: collectd/telemetry) に送信します。STF Smart Gateway は、これらの AMQP アドレスをリッスンしてデータを監視します。複数のクラウドをサポートし、どのクラウドが監視データを生成したかを識別するために、各クラウドがデータを固有のアドレスに送信するように設定します。アドレスの 2 番目の部分に、クラウド識別子の接頭辞を追加します。以下のリストは、アドレスと識別子の例です。

- **collectd/cloud1-telemetry**
- **collectd/cloud1-notify**
- **anycast/ceilometer/cloud1-metering.sample**
- **anycast/ceilometer/cloud1-event.sample**
- **collectd/cloud2-telemetry**
- **collectd/cloud2-notify**
- **anycast/ceilometer/cloud2-metering.sample**
- **anycast/ceilometer/cloud2-event.sample**
- **collectd/us-east-1-telemetry**
- **collectd/us-west-3-telemetry**

4.5.2. Smart Gateway の導入

各クラウドのデータ収集タイプごとに (collectd メトリクスに 1 つ、collectd イベントに 1 つ、Ceilometer メトリクスに 1 つ、Ceilometer イベントに 1 つ) Smart Gateway をデプロイする必要があります。各 Smart Gateway は、対応するクラウドに定義された AMQP アドレスをリッスンするように設定します。Smart Gateway を定義するには、**ServiceTelemetry** マニフェストに **clouds** パラメーターを設定します。

STF を初めてデプロイする際は、1 つのクラウドに対する初期の Smart Gateway を定義する Smart Gateway マニフェストが作成されます。複数のクラウドサポートに Smart Gateway をデプロイする場合には、メトリクスおよび各クラウドのイベントデータを処理するデータ収集タイプごとに、複数の Smart Gateway をデプロイします。最初の Smart Gateway は、以下のサブスクリプションアドレスを使用して **cloud1** で定義されます。

collector	type	デフォルトサブスクリプションアドレス
collectd	metrics	collectd/telemetry
collectd	events	collectd/notify
Ceilometer	metrics	anycast/ceilometer/metering.sample
Ceilometer	events	anycast/ceilometer/event.sample

前提条件

- クラウドの命名方法を決めています。命名スキームの決定の詳細は、「[AMQP アドレス接頭辞の計画](#)」を参照してください。
- clouds オブジェクトのリストを作成しました。**clouds** パラメーターのコンテンツ作成の詳細は、「[clouds パラメーター](#)」を参照してください。

手順

1. Red Hat OpenShift Container Platform にログインします。
2. **service-telemetry** namespace に切り替えます。

```
$ oc project service-telemetry
```

3. **default** の ServiceTelemetry オブジェクトを編集し、設定で **clouds** パラメーターを追加します。



警告

クラウド名が長くなると、Pod 名の最大長 63 文字を超える可能性があります。**ServiceTelemetry** 名の **default** と **clouds.name** の組み合わせが 19 文字を超えないようにしてください。クラウド名には、- などの特殊文字を含めることはできません。クラウド名を英数字 (a-z、0-9) に制限します。

トピックアドレスには文字の制限がなく、**clouds.name** の値とは異なる場合があります。

```
$ oc edit stf default
```

```
apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
metadata:
  ...
spec:
  ...
  clouds:
  - name: cloud1
    events:
      collectors:
      - collectorType: collectd
        subscriptionAddress: collectd/cloud1-notify
      - collectorType: ceilometer
        subscriptionAddress: anycast/ceilometer/cloud1-event.sample
  metrics:
    collectors:
    - collectorType: collectd
      subscriptionAddress: collectd/cloud1-telemetry
    - collectorType: ceilometer
```

```
subscriptionAddress: anycast/ceilometer/cloud1-metering.sample
- name: cloud2
events:
...
```

- ServiceTelemetry オブジェクトを保存します。
- 各 Smart Gateway が起動していることを確認します。この作業は、Smart Gateway の台数によっては数分かかることがあります。

```
$ oc get po -l app=smart-gateway
NAME                                READY STATUS RESTARTS AGE
default-cloud1-ceil-event-smartgateway-6cfb65478c-g5q82 2/2 Running 0 13h
default-cloud1-ceil-meter-smartgateway-58f885c76d-xmxwn 2/2 Running 0 13h
default-cloud1-coll-event-smartgateway-58fbbd4485-r19bd 2/2 Running 0 13h
default-cloud1-coll-meter-smartgateway-7c6fc495c4-jn728 2/2 Running 0 13h
```

4.5.3. デフォルトの Smart Gateway を削除

複数のクラウドに Service Telemetry Framework (STF) を設定したら、デフォルトの Smart Gateway が使用されなくなった場合に、そのゲートウェイを削除できます。Service Telemetry Operator は、作成済みではあるが、オブジェクトの ServiceTelemetry **clouds** 一覧に表示されなくなった **SmartGateway** オブジェクトを削除できます。**clouds** パラメーターで定義されていない SmartGateway オブジェクトの削除を有効にするには、**ServiceTelemetry** マニフェストで **cloudsRemoveOnMissing** パラメーターを **true** に設定する必要があります。

ヒント

Smart Gateway をデプロイしない場合は、**clouds: []** パラメーターを使用して空のクラウド一覧を定義します。



警告

cloudsRemoveOnMissing パラメーターはデフォルトで無効にされています。**cloudsRemoveOnMissing** パラメーターが有効な場合には、現在の namespace で手動で作成された **SmartGateway** オブジェクトを削除するとそのオブジェクトは復元できません。

手順

- Service Telemetry Operator が管理するクラウドオブジェクトの一覧で **clouds** パラメーターを定義します。詳細は、「[clouds パラメーター](#)」を参照してください。
- ServiceTelemetry オブジェクトを編集し、**cloudsRemoveOnMissing** パラメーターを追加します。

```
apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
metadata:
...
```

```
spec:
  ...
  cloudsRemoveOnMissing: true
  clouds:
    ...
```

- 修正内容を保存します。
- オペレーターが Smart Gateway を削除したことを確認します。Operator が変更を調整する間、数分かかることがあります。

```
$ oc get smartgateways
```

4.5.4. 独自のクラウドドメインの設定

Red Hat OpenStack Platform (RHOSP) から Service Telemetry Framework (STF) への AMQ Interconnect ルーター接続が一意で、競合しないようにするには、**CloudDomain** パラメーターを設定します。

手順

- 新しい環境ファイルを作成します (例: **hostnames.yaml**)。
- 以下の例に示すように、環境ファイルで **CloudDomain** パラメーターを設定します。

hostnames.yaml

```
parameter_defaults:
  CloudDomain: newyork-west-04
  CephStorageHostnameFormat: 'ceph-%index%'
  ObjectStorageHostnameFormat: 'swift-%index%'
  ComputeHostnameFormat: 'compute-%index%'
```

- 新しい環境ファイルをデプロイメントに追加します。

関連情報

- 「[複数クラウドの Red Hat OpenStack Platform 環境ファイルの作成](#)」
- オーバークラウドパラメーター ガイドの [コアオーバークラウドパラメーター](#)

4.5.5. 複数クラウドの Red Hat OpenStack Platform 環境ファイルの作成

発信元のクラウドに応じてトラフィックをラベリングするには、クラウド固有のインスタンス名を持つ設定を作成する必要があります。**stf-connectors.yaml** ファイルを作成し、**CeilometerQdrEventsConfig**、**CeilometerQdrMetricsConfig**、および **CollectdAmqpInstances** の値を調整して AMQP アドレスの接頭辞スキームと一致するようにします。

前提条件

- STF によってデプロイされる AMQ Interconnect から CA 証明書を取得している。詳細は、「[オーバークラウドの設定向け Service Telemetry Framework からの CA 証明書の取得](#)」を参照してください。

- clouds オブジェクトのリストを作成しました。clouds パラメーターのコンテンツを作成する方法については、[clouds 設定パラメーター](#) を参照してください。
- AMQ Interconnect のルートアドレスを取得しました。詳細は、「[AMQ Interconnect ルートアドレスの取得](#)」を参照してください。
- これで STF の基本設定ができました。詳細は、「[STF の基本設定の作成](#)」を参照してください。
- 固有のドメイン名環境ファイルを作成しました。詳細は、「[独自のクラウドドメインの設定](#)」を参照してください。

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 **stf-connectors.yaml** という設定ファイルを作成します。
3. **stf-connectors.yaml** ファイルで、オーバークラウドデプロイメント上の AMQ Interconnect をに接続するように **MetricsQdrConnectors** アドレスを設定します。 **CeilometerQdrEventsConfig**、**CeilometerQdrMetricsConfig**、および **CollectdAmqpInstances** トピックの値を、このクラウドデプロイメントに必要な AMQP アドレスに一致するように設定します。
 - 複数のクラウドデプロイメント用に **collectd-write-qdr.yaml** 環境ファイルを含めないため、**resource_registry** 設定は collectd サービスを直接ロードします。
 - **MetricsQdrConnectors.host** パラメーターは、「[AMQ Interconnect ルートアドレスの取得](#)」で取得した **HOST/PORT** の値に置き換えます。
 - **caCertFileContent** パラメーターを、「[オーバークラウドの設定向け Service Telemetry Framework からの CA 証明書の取得](#)」で取得したコンテンツに置き換えます。
 - **CeilometerQdrEventsConfig.topic** を設定して、Ceilometer イベントのトピックを定義します。この値は、**anycast/ceilometer/cloud1-event.sample** のアドレス形式です。
 - **CeilometerQdrMetricsConfig.topic** を設定して、Ceilometer メトリックのトピックを定義します。この値は、**anycast/ceilometer/cloud1-metering.sample** のアドレス形式です。
 - **CollectdAmqpInstances** を設定して、collectd イベントのトピックを定義します。この値は、**collectd/cloud1-notify** の形式になります。
 - **CollectdAmqpInstances** を設定して、collectd メトリックのトピックを定義します。この値は、**collectd/cloud1-telemetry** の形式になります。

stf-connectors.yaml

```
resource_registry:
  OS::TripleO::Services::Collectd: /usr/share/openstack-tripleo-heat-
  templates/deployment/metrics/collectd-container-puppet.yaml

parameter_defaults:
  MetricsQdrConnectors:
    - host: stf-default-interconnect-5671-service-telemetry.apps.infra.watch
      port: 443
      role: edge
      verifyHostname: false
```



```

sslProfile: sslProfile

MetricsQdrSSLProfiles:
- name: sslProfile
  caCertFileContent: |
    -----BEGIN CERTIFICATE-----
    <snip>
    -----END CERTIFICATE-----

CeilometerQdrEventsConfig:
  driver: amqp
  topic: cloud1-event

CeilometerQdrMetricsConfig:
  driver: amqp
  topic: cloud1-metering

CollectdAmqpInstances:
  cloud1-notify:
    notify: true
    format: JSON
    presettle: false
  cloud1-telemetry:
    format: JSON
    presettle: false

```

4. **stf-connectors.yaml** ファイルの命名規則が、Smart Gateway 設定の **spec.bridge.amqpUrl** フィールドと一致していることを確認します。たとえば、**CeilometerQdrEventsConfig.topic** フィールドを **cloud1-event** の値に設定します。
5. 認証ファイルのソース

```

[stack@undercloud-0 ~]$ source stackrc

(undercloud) [stack@undercloud-0 ~]$

```

6. 実際の環境に該当するその他の環境ファイルと共に、**openstack overcloud deployment** コマンドに **stf-connectors.yaml** ファイルおよび一意のドメイン名環境ファイル **hostnames.yaml** を追加します。



警告

カスタム **CollectdAmqpInstances** パラメーターで **collectd-write-qdr.yaml** ファイルを使用する場合、データはカスタムおよびデフォルトのトピックに公開されます。複数のクラウド環境では、**stf-connectors.yaml** ファイルの **resource_registry** パラメーターの設定では collectd サービスを読み込みます。

```

(undercloud) [stack@undercloud-0 ~]$ 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/ceilometer-write-qdr.yaml \  
--environment-file /usr/share/openstack-tripleo-heat-templates/environments/metrics/qdr-  
edge-only.yaml \  
--environment-file /home/stack/hostnames.yaml \  
--environment-file /home/stack/enable-stf.yaml \  
--environment-file /home/stack/stf-connectors.yaml
```

7. Red Hat OpenStack Platform オーバークラウドのデプロイ

関連情報

- デプロイメントの検証方法は、「[クライアント側のインストールの検証](#)」を参照してください。

4.5.6. 複数のクラウドからメトリクスデータを照会

Prometheus に保存されるデータには、収集元の Smart Gateway に従って **service** ラベルが割り当てられます。このラベルを使用して、特定のクラウドのデータを照会することができます。

特定のクラウドからデータをクエリーするには、関連する **service** ラベルに一致する Prometheus `promql` クエリーを使用します (例: `collectd_uptime{service="default-cloud1-coll-meter"}`)。

第5章 SERVICE TELEMETRY FRAMEWORK の運用機能の使用

以下の操作機能を使用して、Service Telemetry Framework (STF) に追加機能を提供できます。

- [ダッシュボードの設定](#)
- [メトリクスの保持期間の設定](#)
- [アラートの設定](#)
- [SNMP トラップの設定](#)
- [高可用性の設定](#)
- [一時ストレージの設定](#)
- [代替可観測性ストラテジーの設定](#)

5.1. SERVICE TELEMETRY FRAMEWORK でのダッシュボード

サードパーティーアプリケーション Grafana を使用して、データコレクター collectd および Ceilometer が個々のホストノードごとに収集するシステムレベルのメトリックを視覚化します。

データコレクターの設定の詳細は、次を参照してください。 [「Service Telemetry Framework 向けの Red Hat OpenStack Platform オークラウドのデプロイ」](#)

5.1.1. ダッシュボードをホストするための Grafana の設定

Grafana はデフォルトの Service Telemetry Framework (STF) のデプロイメントに含まれていないため、Operator Hub.io から Grafana Operator をデプロイする必要があります。Service Telemetry Operator を使用して Grafana をデプロイすると、Grafana インスタンスとローカル STF デプロイメントのデフォルトデータソースの設定が作成されます。

手順

1. Red Hat OpenShift Container Platform にログインします。
2. **service-telemetry** namespace に切り替えます。

```
$ oc project service-telemetry
```

3. Grafana オペレーターをデプロイします。

```
$ oc apply -f - <<EOF
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: grafana-operator
  namespace: service-telemetry
spec:
  channel: v4
  installPlanApproval: Automatic
  name: grafana-operator
```

```
source: operatorhubio-operators
sourceNamespace: openshift-marketplace
EOF
```

- Operator が正常に起動したことを確認します。コマンド出力で、**PHASE** 列の値が **Succeeded** の場合には、Operator は正常に起動されています。

```
$ oc get csv --selector operators.coreos.com/grafana-operator.service-telemetry

NAME                DISPLAY          VERSION  REPLACES          PHASE
grafana-operator.v4.6.0 Grafana Operator 4.6.0   grafana-operator.v4.5.1 Succeeded
```

- Grafana インスタンスを起動するには、**ServiceTelemetry** オブジェクトを作成または変更します。**graphing.enabled** および **graphing.grafana.ingressEnabled** を **true** に設定します。オプションで、**graphing.grafana.baselmage** の値を、デプロイされる Grafana ワークロードコンテナイメージに設定します。

```
$ oc edit stf default

apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
...
spec:
...
graphing:
  enabled: true
grafana:
  ingressEnabled: true
  baselmage: 'registry.redhat.io/rhel8/grafana:7'
```

- Grafana インスタンスがデプロイされたことを確認します。

```
$ oc get pod -l app=grafana

NAME                                READY  STATUS   RESTARTS  AGE
grafana-deployment-7fc7848b56-sbkhv 1/1    Running  0          1m
```

- Grafana のデータソースが正しくインストールされたことを確認します。

```
$ oc get grafanadatasources

NAME                AGE
default-datasources 20h
```

- Grafana のルートが存在することを確認します。

```
$ oc get route grafana-route

NAME          HOST/PORT          PATH  SERVICES  PORT
TERMINATION  WILDCARD
grafana-route grafana-route-service-telemetry.apps.infra.watch grafana-service 3000
edge         None
```

5.1.2. Grafana のログイン認証情報の取得および設定

Service Telemetry Framework (STF) は、Grafana が有効な場合にデフォルトのログイン認証情報を設定します。**ServiceTelemetry** オブジェクトで認証情報を上書きできます。

手順

1. Red Hat OpenShift Container Platform にログインします。

2. **service-telemetry** namespace に切り替えます。

```
$ oc project service-telemetry
```

3. STF オブジェクトからデフォルトのユーザー名とパスワードを取得します。

```
$ oc get stf default -o jsonpath="{.spec.graphing.grafana['adminUser','adminPassword']}"
```

4. ServiceTelemetry オブジェクトを介して Grafana 管理者のユーザー名およびパスワードのデフォルト値を変更するには、**graphing.grafana.adminUser** および **graphing.grafana.adminPassword** パラメーターを使用します。

5.2. SERVICE TELEMETRY FRAMEWORK でのメトリクスの保持期間

Service Telemetry Framework (STF) に保存されるメトリクスのデフォルトの保持期間は 24 時間となっており、アラート目的で傾向を把握するのに十分なデータが提供されます。

長期ストレージの場合は、Thanos など、長期のデータ保持用に設計されたシステムを使用します。

関連情報

- 追加のメトリクスの保持時間に合わせて STF を調整するには、「[Service Telemetry Framework でのメトリクスの保持期間の編集](#)」を参照してください。
- Prometheus のデータ保存に関する推奨事項や、ストレージ容量の見積もりについては、<https://prometheus.io/docs/prometheus/latest/storage/#operational-aspects> を参照してください。
- Thanos の詳細は、<https://thanos.io/> を参照してください。

5.2.1. Service Telemetry Framework でのメトリクスの保持期間の編集

追加のメトリクスの保持時間に対応するために、Service Telemetry Framework (STF) を調整できます。

手順

1. Red Hat OpenShift Container Platform にログインします。

2. **service-telemetry** namespace に切り替えます。

```
$ oc project service-telemetry
```

3. Service Telemetry オブジェクトを編集します。

```
$ oc edit stf default
```

4. **retention: 7d** を `backends.metrics.prometheus.storage` の `storage` セクションに追加し、保持期間を7日間増やします。



注記

保持期間を長く設定すると、設定された Prometheus システムからデータを取得すると、クエリーで結果の速度が遅くなる可能性があります。

```
apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
metadata:
  name: stf-default
  namespace: service-telemetry
spec:
  ...
  backends:
    metrics:
      prometheus:
        enabled: true
        storage:
          strategy: persistent
          retention: 7d
  ...
```

5. 変更内容を保存し、オブジェクトを閉じます。

関連情報

- メトリクスの保持期間の詳細は、[「Service Telemetry Framework でのメトリクスの保持期間」](#)を参照してください。

5.3. SERVICE TELEMETRY FRAMEWORK でのアラート

Prometheus ではアラートルールを作成し、Alertmanager ではアラートルートを作成します。Prometheus サーバーのアラートルールは、アラートを管理する Alertmanager にアラートを送信します。Alertmanager は通知をオフにしたり、アラートを集約してメール (on-call 通知システムまたはチャットプラットフォーム) で通知を送信できます。

アラートを作成するには、以下のタスクを行います。

1. Prometheus でアラートルールを作成します。詳細は、[「Prometheus でのアラートルールの作成」](#)を参照してください。
2. Alertmanager でアラートルートを作成します。アラートルートを作成するには、2つの方法があります。
 - [Alertmanager での標準的なアラートルートの作成。](#)
 - [Alertmanager のテンプレート化によるアラートルートの作成。](#)

関連情報

Prometheus と Alertmanager によるアラートまたは通知の詳細については、<https://prometheus.io/docs/alerting/overview/> を参照してください。

Service Telemetry Framework (STF) で使用できるアラートの例を見るには、<https://github.com/infracore/service-telemetry-operator/tree/master/deploy/alerts> を参照してください。

5.3.1. Prometheus でのアラートルールの作成

Prometheus はアラートルールを評価して通知を行います。ルール条件が空の結果セットを返す場合は、条件は偽となります。それ以外の場合は、ルールが真となり、アラートが発生します。

手順

1. Red Hat OpenShift Container Platform にログインします。
2. **service-telemetry** namespace に切り替えます。

```
$ oc project service-telemetry
```

3. アラートルールを含む **PrometheusRule** オブジェクトを作成します。Prometheus Operator は、ルールを Prometheus に読み込みます。

```
$ oc apply -f - <<EOF
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  creationTimestamp: null
  labels:
    prometheus: default
    role: alert-rules
  name: prometheus-alarm-rules
  namespace: service-telemetry
spec:
  groups:
  - name: ./openstack.rules
    rules:
    - alert: Collectd metrics receive rate is zero
      expr: rate(sg_total_collectd_msg_received_count[1m]) == 0
EOF
```

ルールを変更するには、**expr** パラメーターの値を編集します。

4. Operator がルールを Prometheus に読み込んだことを確認するには、Basic 認証で default-prometheus-proxy ルートに対して **curl** コマンドを実行します。

```
$ curl -k --user "internal:$(oc get secret default-prometheus-htpasswd -o go-template='{{ .data.password | base64decode }}')" https://$(oc get route default-prometheus-proxy -o go-template='{{ .spec.host }}')/api/v1/rules
```

```
{
  "status": "success",
  "data": {
    "groups": [
      {
        "name": "./openstack.rules",
        "file": "/etc/prometheus/rules/prometheus-default-rulefiles-0/service-telemetry-prometheus-alarm-rules.yaml",
        "rules": [
          {
            "state": "inactive",
            "name": "Collectd metrics receive count is zero",
            "query": "rate(sg_total_collectd_msg_received_count[1m]) == 0",
            "duration": "0",
            "labels": {}
          }
        ]
      }
    ]
  }
}
```

```
{}, "annotations": {}, "alerts":
[], "health": "ok", "evaluationTime": 0.00034627, "lastEvaluation": "2021-12-
07T17:23:22.160448028Z", "type": "alerting"}], "interval": 30, "evaluationTime": 0.000353787, "last
Evaluation": "2021-12-07T17:23:22.160444017Z"}]]}]}
```

関連情報

- アラートの詳細については、<https://github.com/coreos/prometheus-operator/blob/master/Documentation/user-guides/alerting.md> を参照してください。

5.3.2. カスタムアラートの設定

カスタムアラートは、「[Prometheus でのアラートルールの作成](#)」で作成した **PrometheusRule** オブジェクトに追加できます。

手順

1. **oc edit** コマンドを使用します。

```
$ oc edit prometheusrules prometheus-alarm-rules
```

2. **PrometheusRules** マニフェストを編集します。
3. マニフェストを保存し、終了します。

関連情報

- アラートルールの設定方法は、https://prometheus.io/docs/prometheus/latest/configuration/alerting_rules/ を参照してください。
- Prometheus Rules オブジェクトの詳細については、<https://github.com/coreos/prometheus-operator/blob/master/Documentation/user-guides/alerting.md> を参照してください。

5.3.3. Alertmanager での標準的なアラートルートの作成

Alertmanager を使用して、電子メール、IRC、その他の通知チャンネルなどの外部システムにアラートを配信します。Prometheus Operator は、Alertmanager 設定を Red Hat OpenShift Container Platform シークレットとして管理します。デフォルトで、Service Telemetry Framework (STF) は、受信側を持たない基本的な設定をデプロイします。

```
alertmanager.yaml: |-
  global:
    resolve_timeout: 5m
  route:
    group_by: ['job']
    group_wait: 30s
    group_interval: 5m
    repeat_interval: 12h
    receiver: 'null'
  receivers:
  - name: 'null'
```


STF を使用してカスタム Alertmanager ルートをデプロイするには、**alertmanagerConfigManifest** パラメーターを Service Telemetry Operator に渡す必要があります。これにより、更新されたシークレットが作成され、Prometheus Operator の管理対象となります。



注記

alertmanagerConfigManifest に、送信されるアラートのタイトルとテキストを設定するカスタムテンプレートが含まれている場合は、Base64 エンコードされた設定を使用し、**alertmanagerConfigManifest** のコンテンツをデプロイします。詳細は、「[Alertmanager のテンプレート化によるアラートルートの作成](#)」を参照してください。

手順

1. Red Hat OpenShift Container Platform にログインします。
2. **service-telemetry** namespace に切り替えます。

```
$ oc project service-telemetry
```

3. STF デプロイメントの **ServiceTelemetry** オブジェクトを編集します。

```
$ oc edit stf default
```

4. 新規パラメーター **alertmanagerConfigManifest** および **Secret** オブジェクトの内容を追加し、Alertmanager の **alertmanager.yaml** 設定を定義します。



注記

この手順では、Service Telemetry Operator が管理するデフォルトのテンプレートを読み込みます。変更が正しく入力されていることを確認するには、値を変更して **alertmanager-default** シークレットを返し、新しい値がメモリーに読み込まれていることを確認します。たとえば、パラメーター **global.resolve_timeout** の値を **5m** から **10m** に変更します。

```
apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
metadata:
  name: default
  namespace: service-telemetry
spec:
  backends:
    metrics:
      prometheus:
        enabled: true
  alertmanagerConfigManifest: |
    apiVersion: v1
    kind: Secret
    metadata:
      name: 'alertmanager-default'
      namespace: 'service-telemetry'
    type: Opaque
    stringData:
      alertmanager.yaml: |-
```

```

global:
  resolve_timeout: 10m
route:
  group_by: ['job']
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 12h
  receiver: 'null'
receivers:
- name: 'null'

```

5. 設定がシークレットに適用されたことを確認します。

```
$ oc get secret alertmanager-default -o go-template='{{index .data "alertmanager.yaml" | base64decode }}'
```

```

global:
  resolve_timeout: 10m
route:
  group_by: ['job']
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 12h
  receiver: 'null'
receivers:
- name: 'null'

```

6. **alertmanager-proxy** サービスに対して **curl** コマンドを実行して、ステータスおよび **configYAML** コンテンツを取得し、指定された設定が Alertmanager の設定と一致することを確認します。

```
$ oc run curl -it --serviceaccount=prometheus-k8s --restart='Never' --
image=radial/busyboxplus:curl -- sh -c "curl -k -H \"Content-Type: application/json\" -H
\"Authorization: Bearer \$(cat /var/run/secrets/kubernetes.io/serviceaccount/token)\"
https://default-alertmanager-proxy:9095/api/v1/status"

{"status": "success", "data": {"configYAML": "...", ...}}
```

7. **configYAML** フィールドに予想される変更が含まれることを確認します。
8. 環境を消去するには、**curl** Pod を削除します。

```
$ oc delete pod curl

pod "curl" deleted
```

関連情報

- Red Hat Open Shift Container Platform のシークレットと Prometheus オペレーターの詳細については、[Prometheus user guide on alerting](#) を参照してください。

5.3.4. Alertmanager のテンプレート化によるアラートルートの作成

Alertmanager を使用して、電子メール、IRC、その他の通知チャンネルなどの外部システムにアラートを

配信します。Prometheus Operator は、Alertmanager 設定を Red Hat OpenShift Container Platform シークレットとして管理します。デフォルトで、Service Telemetry Framework (STF) は、受信側を持たない基本的な設定をデプロイします。

```
alertmanager.yaml: |-
  global:
    resolve_timeout: 5m
  route:
    group_by: ['job']
    group_wait: 30s
    group_interval: 5m
    repeat_interval: 12h
    receiver: 'null'
  receivers:
  - name: 'null'
```

alertmanagerConfigManifest パラメーターに、送信されたアラートのタイトルとテキストを設定するためのカスタムテンプレートなどが含まれている場合、Base64 エンコードされた設定を使用して、**alertmanagerConfigManifest** のコンテンツをデプロイします。

手順

1. Red Hat OpenShift Container Platform にログインします。
2. **service-telemetry** namespace に切り替えます。

```
$ oc project service-telemetry
```

3. STF デプロイメントの **ServiceTelemetry** オブジェクトを編集します。

```
$ oc edit stf default
```

4. STF を使用してカスタム Alertmanager ルートをデプロイするには、**alertmanagerConfigManifest** パラメーターを Service Telemetry Operator に渡す必要があります。これにより、更新されたシークレットが作成され、Prometheus Operator の管理対象となります。

```
apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
metadata:
  name: default
  namespace: service-telemetry
spec:
  backends:
    metrics:
      prometheus:
        enabled: true
  alertmanagerConfigManifest: |
    apiVersion: v1
    kind: Secret
    metadata:
      name: 'alertmanager-default'
      namespace: 'service-telemetry'
    type: Opaque
    data:
```

```

alertmanager.yaml:
Z2xvYmFsOgogIHJlc29sdmVfdGltZW91dDogMTBtCiAgc2xhY2tfYXBpX3VybDogPHNsYWNRX
2FwaV91cmw+CnJlY2VpdmVyczoKICAtIG5hbWU6IHNsYWNRciAgICBzbGFja19jb25maWdzO
gogICAgLSBjaGFubmVsOiAjc3RmLWFsZXJ0cwogICAgICB0aXRzZTogfC0KICAgICAgICAgLi
4KICAgICAgdGV4dDogPi0KICAgICAgICAgLi4Kcm91dGU6CiAgZ3JvdXBfYnk6IFsnam9iJ10Kl
CBncm91cF93YWl0OiAzMHMKICBncm91cF9pbnRlcnZhbDogNW0KICByZXBIYXRfaW50ZXJ2
YWw6IDEyaAogIHJlY2VpdmVyOiAnc2xhY2snCg==

```

- 設定がシークレットに適用されたことを確認します。

```

$ oc get secret alertmanager-default -o go-template='{{index .data "alertmanager.yaml" |
base64decode }}'

global:
  resolve_timeout: 10m
  slack_api_url: <slack_api_url>
receivers:
- name: slack
  slack_configs:
  - channel: #stf-alerts
    title: |-
      ...
    text: >-
      ...
route:
  group_by: ['job']
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 12h
  receiver: 'slack'

```

- alertmanager-proxy** サービスに対して **curl** コマンドを実行して、ステータスおよび **configYAML** コンテンツを取得し、指定された設定が Alertmanager の設定と一致することを確認します。

```

$ oc run curl -it --serviceaccount=prometheus-k8s --restart='Never' --
image=radial/busyboxplus:curl -- sh -c "curl -k -H \"Content-Type: application/json\" -H
\"Authorization: Bearer \$(cat /var/run/secrets/kubernetes.io/serviceaccount/token)\"
https://default-alertmanager-proxy:9095/api/v1/status"

{"status":"success","data":{"configYAML":"...","..."}}

```

- configYAML** フィールドに予想される変更が含まれることを確認します。
- 環境を消去するには、**curl** Pod を削除します。

```

$ oc delete pod curl

pod "curl" deleted

```

関連情報

- Red Hat Open Shift Container Platform のシークレットと Prometheus オペレーターの詳細については、[Prometheus user guide on alerting](#) を参照してください。

5.4. SNMP トラップの設定

Service Telemetry Framework (STF) は、SNMP トラップで通知を受信する既存のインフラストラクチャーモニタリングプラットフォームと統合できます。SNMP トラップを有効にするには、**ServiceTelemetry** オブジェクトを変更し、**snmpTraps** パラメーターを設定します。

アラートの設定については、「[Service Telemetry Framework でのアラート](#)」を参照してください。

前提条件

- アラートを送信する SNMP トラップ受信機の IP アドレスまたはホスト名が判明しています。

手順

- SNMP トラップを有効にするには、**ServiceTelemetry** オブジェクトを変更します。

```
$ oc edit stf default
```

- alerting.alertmanager.receivers.snmpTraps** パラメーターを設定します。

```
apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
...
spec:
...
  alerting:
    alertmanager:
      receivers:
        snmpTraps:
          enabled: true
          target: 10.10.10.10
```

- target** の値は、SNMP トラップレシーバーの IP アドレスまたはホスト名に設定するようにしてください。

5.5. 高可用性

高可用性により、Service Telemetry Framework (STF) はコンポーネントサービスの障害から迅速に復旧できます。Red Hat OpenShift Container Platform は、ワークロードをスケジュールするノードが利用可能な場合に、障害のある Pod を再起動しますが、この復旧プロセスではイベントとメトリクスが失われる可能性があります。高可用性設定には、複数の STF コンポーネントのコピーが含まれており、復旧時間が約 2 秒に短縮されます。Red Hat OpenShift Container Platform ノードの障害から保護するには、3 つ以上のノードで STF を Red Hat OpenShift Container Platform クラスターにデプロイします。



警告

STF はまだ完全なフォールトトレラントシステムではありません。復旧期間中のメトリクスやイベントの配信は保証されません。

高可用性を有効にすると、以下のような効果があります。

- デフォルトの1つではなく、3つの Elasticsearch Pod が実行されます。
- 以下のコンポーネントは、デフォルトの1つの Pod ではなく、2つの Pod を実行します。
 - AMQ Interconnect
 - Alertmanager
 - Prometheus
 - Events Smart Gateway
 - Metrics Smart Gateway
- これらのサービスのいずれにおいても、Pod の紛失からの復旧時間は約 2 秒に短縮されます。

5.5.1. 高可用性の設定

高可用性のために Service Telemetry Framework (STF) を設定するには、Red Hat OpenShift Container Platform の ServiceTelemetry オブジェクトに **highAvailability.enabled: true** を追加します。このパラメーターはインストール時に設定できます。またはすでに STF をデプロイしている場合には、以下の手順を実行します。

手順

1. Red Hat OpenShift Container Platform にログインします。
2. **service-telemetry** namespace に切り替えます。

```
$ oc project service-telemetry
```

3. oc コマンドで ServiceTelemetry オブジェクトを編集します。

```
$ oc edit stf default
```

4. **highAvailability.enabled: true** を **spec** セクションに追加します。

```
apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
...
spec:
  ...
  highAvailability:
    enabled: true
```

5. 変更内容を保存し、オブジェクトを閉じます。

5.6. 一時ストレージ

一時ストレージを使用して、Red Hat OpenShift Container Platform クラスターにデータを永続的に保存せずに Service Telemetry Framework (STF) を実行できます。



警告

一時ストレージを使用している場合は、Pod が別のノードで再起動、更新、再スケジューリングされると、データが失われる可能性があります。一時ストレージは、本番環境ではなく、開発やテストにのみ使用してください。

5.6.1. 一時ストレージの設定

一時ストレージ用に STF コンポーネントを設定するには、対応するパラメーターに **...storage.strategy: ephemeral** を追加します。たとえば、Prometheus バックエンドの一時ストレージを有効にするには、**backends.metrics.prometheus.storage.strategy: ephemeral** を設定します。一時ストレージの設定をサポートするコンポーネントには、**alerting.alertmanager**、**backends.metrics.prometheus**、および **backends.events.elasticsearch** が含まれます。一時ストレージの設定は、インストール時に追加することもできますが、すでに STF を導入している場合は、以下の手順で追加することができます。

手順

1. Red Hat OpenShift Container Platform にログインします。
2. **service-telemetry** namespace に切り替えます。

```
$ oc project service-telemetry
```

3. Service Telemetry オブジェクトを編集します。

```
$ oc edit stf default
```

4. **...storage.strategy: ephemeral** パラメーターを関連コンポーネントの **spec** セクションに追加します。

```
apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
metadata:
  name: stf-default
  namespace: service-telemetry
spec:
  alerting:
    enabled: true
    alertmanager:
      storage:
        strategy: ephemeral
  backends:
    metrics:
      prometheus:
        enabled: true
        storage:
          strategy: ephemeral
    events:
      elasticsearch:
```

```
enabled: true
storage:
  strategy: ephemeral
```

5. 変更内容を保存し、オブジェクトを閉じます。

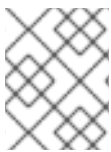
5.7. SERVICE TELEMETRY FRAMEWORK の可観測性ストラテジー

Service Telemetry Framework (STF) には、ストレージバックエンドおよびアラートツールは含まれません。STF はコミュニティ Operator を使用して Prometheus、Alertmanager、Grafana、および Elasticsearch をデプロイします。STF は、これらのコミュニティ Operator にリクエストを行い、STF と連携するように設定された各アプリケーションのインスタンスを作成します。

Service Telemetry Operator がカスタムリソース要求を作成する代わりに、これらのアプリケーションまたは他の互換性のあるアプリケーションの独自のデプロイメントを使用し、Telemetry ストレージ用の独自の Prometheus 互換システムに配信するためにメトリクス Smart Gateways を収集できます。代わりに別のバックエンドを使用するように可観測性ストラテジーを設定する場合、STF には永続ストレージまたは一時ストレージは必要ありません。

5.7.1. 代替可観測性ストラテジーの設定

ストレージ、可視化、およびアラートバックエンドのデプロイメントを省略するように STF を設定するには、ServiceTelemetry 仕様に **observabilityStrategy: none** を追加します。このモードでは、AMQ Interconnect ルーターおよびメトリクス Smart Gateway のみがデプロイされ、外部の Prometheus 互換システムを設定して、STF Smart Gateways からメトリクスを収集する必要があります。



注記

現在、**observabilityStrategy** を **none** に設定すると、メトリクスのみがサポートされません。Events Smart Gateways はデプロイされません。

手順

1. **spec** パラメーターに **observabilityStrategy: none** プロパティを指定して **ServiceTelemetry** オブジェクトを作成します。マニフェストは、すべてのメトリクスコレクタータイプを持つ単一のクラウドから Telemetry を受け取るのに適した STF のデフォルトデプロイメントを示しています。

```
$ oc apply -f - <<EOF
apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
metadata:
  name: default
  namespace: service-telemetry
spec:
  observabilityStrategy: none
EOF
```

2. すべてのワークロードが適切に動作していることを確認するには、Pod および各 Pod のステータスを確認します。

```
$ oc get pods
NAME                                     READY STATUS RESTARTS AGE
default-cloud1-ceil-meter-smartgateway-59c845d65b-gzhcs 3/3 Running 0 132m
```


default-cloud1-coll-meter-smartgateway-75bbd948b9-d5phm	3/3	Running	0	132m
default-cloud1-sens-meter-smartgateway-7fdbb57b6d-dh2g9	3/3	Running	0	132m
default-interconnect-668d5bbcd6-57b2l	1/1	Running	0	132m
interconnect-operator-b8f5bb647-tlp5t	1/1	Running	0	47h
service-telemetry-operator-566b9dd695-wkvjq	1/1	Running	0	156m
smart-gateway-operator-58d77dcf7-6xsq7	1/1	Running	0	47h

関連情報

追加のクラウドの設定、またはサポート対象のコレクターのセットの変更に関する詳細は、[「Smart Gateway の導入」](#) を参照してください。

第6章 AMQ INTERCONNECT 証明書の更新

証明書の有効期限が切れたときに、Red Hat OpenStack Platform (RHOSP) と Service Telemetry Framework (STF) の間の AMQ Interconnect 接続を保護する CA 証明書を定期的に更新する必要があります。更新は Red Hat OpenShift Container Platform の cert-manager コンポーネントによって自動的に処理されますが、更新された証明書を手動で RHOSP ノードにコピーする必要があります。

6.1. 期限切れの AMQ INTERCONNECT CA 証明書の確認

CA 証明書の有効期限が切れると、AMQ Interconnect 接続は維持されますが、中断された場合は再接続できなくなります。最終的に、Red Hat OpenStack Platform (RHOSP) ルーターからの接続の一部またはすべてが失敗し、両側でエラーが表示され、CA 証明書の有効期限 (または後で) フィールドが過去のものになることがわかります。

手順

1. Red Hat OpenShift Container Platform にログインします。
2. **service-telemetry** namespace に切り替えます。

```
$ oc project service-telemetry
```

3. 一部またはすべてのルーター接続が失敗したことを確認します。

```
$ oc exec -it $(oc get po -l application=default-interconnect -o
  jsonpath='{.items[0].metadata.name}') -- qdstat --connections | grep Router | wc
  0 0 0
```

4. Red Hat OpenShift Container Platform がホストする AMQ Interconnect ログで、次のエラーを確認します。

```
$ oc logs -l application=default-interconnect | tail
[...]
```

```
2022-11-10 20:51:22.863466 +0000 SERVER (info) [C261] Connection from
10.10.10.10:34570 (to 0.0.0.0:5671) failed: amqp:connection:framing-error SSL Failure:
error:140940E5:SSL routines:ssl3_read_bytes:ssl handshake failure
```

5. RHOSP アンダークラウドにログインします。
6. 接続が失敗したノードの RHOSP がホストする AMQ Interconnect ログで、次のエラーを確認します。

```
$ ssh controller-0.ctlplane -- sudo tail /var/log/containers/metrics_qdr/metrics_qdr.log
[...]
```

```
2022-11-10 20:50:44.311646 +0000 SERVER (info) [C137] Connection to default-
interconnect-5671-service-telemetry.apps.mycluster.com:443 failed:
amqp:connection:framing-error SSL Failure: error:0A000086:SSL routines::certificate verify
failed
```

7. RHOSP ノードのファイルを調べて、CA 証明書の有効期限が切れていることを確認します。

```
$ ssh controller-0.ctlplane -- cat /var/lib/config-data/puppet-
generated/metrics_qdr/etc/pki/tls/certs/CA_sslProfile.pem | openssl x509 -text | grep "Not
```

```
After"
```

```
Not After : Nov 10 20:31:16 2022 GMT
```

```
$ date
```

```
Mon Nov 14 11:10:40 EST 2022
```

6.2. AMQ INTERCONNECT CA 証明書の更新

AMQ Interconnect 証明書を更新するには、Red Hat OpenShift Container Platform から証明書をエクスポートし、Red Hat OpenStack Platform (RHOSP) ノードにコピーする必要があります。

手順

1. Red Hat OpenShift Container Platform にログインします。
2. **service-telemetry** namespace に切り替えます。

```
$ oc project service-telemetry
```

3. CA 証明書を **STFCA.pem** にエクスポートします。

```
$ oc get secret/default-interconnect-selfsigned -o jsonpath='{.data.ca.crt}' | base64 -d > STFCA.pem
```

4. STFCA.pem を RHOSP アンダークラウドにコピーします。
5. RHOSP アンダークラウドにログインします。
6. stf-connectors.yaml ファイルを編集して、新しい caCertFileContent を含めます (詳細は、「[オーバークラウドの STF 接続の設定](#)」)



注記

以下の手順を実行した後、オーバークラウドのデプロイを実行する必要はありません。stf-connectors.yaml ファイルを編集するのは、今後のデプロイメントで新しい CA 証明書が上書きされないようにするためだけです。

7. STFCA.pem ファイルを各 RHOSP オーバークラウドノードにコピーします。

```
[stack@undercloud-0 ~]$ ansible -i overcloud-deploy/overcloud/tripleo-ansible-inventory.yaml allovercloud -b -m copy -a "src=STFCA.pem dest=/var/lib/config-data/puppet-generated/metrics_qdr/etc/pki/tls/certs/CA_sslProfile.pem"
```

8. 各 RHOSP オーバークラウドノードで metrics_qdr コンテナを再起動します。

```
[stack@undercloud-0 ~]$ tripleo-ansible-inventory --static-yaml-inventory ./tripleo-ansible-inventory.yaml
[stack@undercloud-0 ~]$ ansible -i tripleo-ansible-inventory.yaml allovercloud -m shell -a "sudo podman restart metrics_qdr"
```

第7章 RED HAT OPENSIFT CONTAINER PLATFORM 環境からの SERVICE TELEMETRY FRAMEWORK の削除

STF 機能を必要としなくなった場合に、Red Hat OpenShift Container Platform 環境から Service Telemetry Framework (STF) を削除します。

Red Hat OpenShift Container Platform 環境から STF を削除するには、以下のタスクを実行する必要があります。

1. namespace を削除します。
2. カタログソースを削除します。
3. cert-manager Operator を削除します。

7.1. NAMESPACE の削除

Red Hat OpenShift Container Platform から STF の操作リソースを削除するには、namespace を削除します。

手順

1. **oc delete** コマンドを実行します。

```
$ oc delete project service-telemetry
```

2. ネームスペースからリソースが削除されたことを確認します。

```
$ oc get all  
No resources found.
```

7.2. CATALOGSOURCE の削除

Service Telemetry Framework (STF) を再びインストールしない場合には、CatalogSource を削除します。CatalogSource を削除すると、STF に関連する PackageManifest は Operator Lifecycle Manager カタログから自動的に削除されます。

手順

1. インストール時に OperatorHub.io Community Catalog Source を有効にしている、このカタログソースが不要になった場合は、削除してください。

```
$ oc delete --namespace=openshift-marketplace catalogsource operatorhubio-operators  
catalogsource.operators.coreos.com "operatorhubio-operators" deleted
```

関連情報

Operator Hub.io Community Catalog Source の詳細については、「[Service Telemetry Framework の Red Hat OpenShift Container Platform 環境へのデプロイ](#)」を参照してください。

7.3. CERT-MANAGER OPERATOR の削除

他のアプリケーションに cert-manager Operator を使用していない場合は、Subscription、ClusterServiceVersion、および CustomResourceDefinitions を削除します。

手順

1. **openshift-cert-manager-operator** namespace からサブスクリプションを削除します。

```
$ oc delete --namespace=openshift-cert-manager-operator subscription openshift-cert-manager-operator  
  
subscription.operators.coreos.com "openshift-cert-manager-operator" deleted
```

2. インストールされた ClusterServiceVersion のバージョン番号を取得します。

```
$ oc get --namespace=openshift-cert-manager-operator subscription openshift-cert-manager-operator -oyaml | grep currentCSV
```

出力例:

```
currentCSV: openshift-cert-manager.v1.7.1
```

3. **openshift-cert-manager-operator** namespace から ClusterServiceVersion を削除します。

```
$ oc delete --namespace=openshift-cert-manager-operator csv openshift-cert-manager.v1.7.1
```

出力例:

```
clusterserviceversion.operators.coreos.com "openshift-cert-manager.v1.7.1" deleted
```

4. Operator によって提供される CustomResourceDefinitions の現在のリストを取得して、ClusterServiceVersion の削除後に削除できるようにします。

```
$ oc get csv -n openshift-cert-manager-operator openshift-cert-manager.v1.7.1 -oyaml | grep "kind: CustomResourceDefinition" -A2 | grep name | awk '{print $2}'
```

出力例:

```
certificaterequests.cert-manager.io  
certificates.cert-manager.io  
certmanagers.config.openshift.io  
certmanagers.operator.openshift.io  
challenges.acme.cert-manager.io  
clusterissuers.cert-manager.io  
issuers.cert-manager.io  
orders.acme.cert-manager.io
```

5. cert-Manager Operator に関連する CustomResourceDefinitions を削除します。

```
$ oc delete crd certificaterequests.cert-manager.io certificates.cert-manager.io certmanagers.config.openshift.io certmanagers.operator.openshift.io challenges.acme.cert-manager.io clusterissuers.cert-manager.io issuers.cert-manager.io orders.acme.cert-manager.io
```

出力例:

```
customresourcedefinition.apiextensions.k8s.io "certificaterequests.cert-manager.io" deleted
customresourcedefinition.apiextensions.k8s.io "certificates.cert-manager.io" deleted
customresourcedefinition.apiextensions.k8s.io "certmanagers.config.openshift.io" deleted
customresourcedefinition.apiextensions.k8s.io "certmanagers.operator.openshift.io" deleted
customresourcedefinition.apiextensions.k8s.io "challenges.acme.cert-manager.io" deleted
customresourcedefinition.apiextensions.k8s.io "clusterissuers.cert-manager.io" deleted
customresourcedefinition.apiextensions.k8s.io "issuers.cert-manager.io" deleted
customresourcedefinition.apiextensions.k8s.io "orders.acme.cert-manager.io" deleted
```

6. cert-manager Operator が所有する namespace を削除します。

```
$ oc delete project openshift-cert-manager openshift-cert-manager-operator
```

出力例:

```
project.project.openshift.io "openshift-cert-manager" deleted
project.project.openshift.io "openshift-cert-manager-operator" deleted
```

追加情報

- [クラスターからの Operator の削除](#)

第8章 SERVICE TELEMETRY FRAMEWORK のバージョン 1.5 へのアップグレード

Service Telemetry Framework (STF) 1.4 を STF 1.5 にアップグレードするには、次の手順を完了する必要があります。

- AMQ Certificate Manager を Certificate Manager に置き換えます。
- Red Hat OpenShift Container Platform 環境の **service-telemetry** 名前空間にある Smart Gateway Operator および Service Telemetry Operator の **ClusterServiceVersion** および **Subscription** オブジェクトを削除します。
- Red Hat OpenShift Container Platform を 4.8 から 4.10 にアップグレードします。
- 削除したオペレーターを再度有効にします。
- Red Hat OpenStack Platform (RHOSP) で AMQ Interconnect CA 証明書を更新します。

前提条件

- データのバックアップを作成しています。Red Hat OpenShift Container Platform のアップグレード中に停止します。Operator の交換中に **ServiceTelemetry** および **SmartGateway** オブジェクトを再設定することはできません。
- Red Hat OpenShift Container Platform 4.8 からサポートされているバージョン 4.10 にアップグレードする環境を準備しました。
- Red Hat OpenShift Container Platform クラスターは完全に接続されています。STF は、切断されたクラスターまたはネットワークが制限されたクラスターをサポートしません。

8.1. SERVICE TELEMETRY FRAMEWORK 1.4 オペレーターの削除

Service Telemetry Framework (STF) 1.4 Operator と AMQ Certificate Manager Operator を Red Hat OpenShift Container Platform 4.8 から削除します。

手順

1. Service Telemetry Operator を削除します。
2. スマートゲートウェイオペレーターを削除します。
3. AMQ Certificate Manager Operator の削除
4. Grafana オペレーターを削除します。

関連情報

- Red Hat OpenShift Container Platform からの Operator の削除に関する詳細は、[Deleting Operators from a cluster](#) を参照してください。

8.1.1. サービステレメトリーオペレーターの削除

Service Telemetry Framework (STF) インストールのアップグレードの一環として、Red Hat OpenShift Container Platform 環境の **service-telemetry** 名前空間で Service Telemetry Operator を削除する必要があります。

手順

1. **service-telemetry** プロジェクトに変更します。

```
$ oc project service-telemetry
```

2. Service Telemetry Operator サブスクリプションを削除します。

```
$ oc delete sub --selector=operators.coreos.com/service-telemetry-operator.service-telemetry
```

```
subscription.operators.coreos.com "service-telemetry-operator" deleted
```

3. Service Telemetry Operator **ClusterServiceVersion** を削除します。

```
$ oc delete csv --selector=operators.coreos.com/service-telemetry-operator.service-telemetry
```

```
clusterserviceversion.operators.coreos.com "service-telemetry-operator.v1.4.1669718959" deleted
```

検証

1. Service Telemetry Operator のデプロイが実行されていないことを確認します。

```
$ oc get deploy --selector=operators.coreos.com/service-telemetry-operator.service-telemetry
```

```
No resources found in service-telemetry namespace.
```

2. Service Telemetry Operator サブスクリプションが存在しないことを確認します。

```
$ oc get sub --selector=operators.coreos.com/service-telemetry-operator.service-telemetry
```

```
No resources found in service-telemetry namespace.
```

3. Service Telemetry Operator ClusterServiceVersion が存在しないことを確認します。

```
$ oc get csv --selector=operators.coreos.com/service-telemetry-operator.service-telemetry
```

```
No resources found in service-telemetry namespace.
```

8.1.2. Smart Gateway オペレーターの削除

Service Telemetry Framework (STF) インストールのアップグレードの一環として、Red Hat OpenShift Container Platform 環境の **service-telemetry** 名前空間にある Smart Gateway Operator を削除する必要があります。

手順

1. **service-telemetry** プロジェクトに変更します。

```
$ oc project service-telemetry
```

2. Smart Gateway Operator サブスクリプションを削除します。

```
$ oc delete sub --selector=operators.coreos.com/smart-gateway-operator.service-telemetry
subscription.operators.coreos.com "smart-gateway-operator-stable-1.4-redhat-operators-
openshift-marketplace" deleted
```

3. Smart Gateway Operator **ClusterServiceVersion** を削除します。

```
$ oc delete csv --selector=operators.coreos.com/smart-gateway-operator.service-telemetry
clusterserviceversion.operators.coreos.com "smart-gateway-operator.v4.0.1669718962"
deleted
```

検証

1. Smart Gateway Operator デプロイメントが実行されていないことを確認します。

```
$ oc get deploy --selector=operators.coreos.com/smart-gateway-operator.service-telemetry
No resources found in service-telemetry namespace.
```

2. Smart Gateway Operator サブスクリプションが存在しないことを確認します。

```
$ oc get sub --selector=operators.coreos.com/smart-gateway-operator.service-telemetry
No resources found in service-telemetry namespace.
```

3. Smart Gateway Operator ClusterServiceVersion が存在しないことを確認します。

```
$ oc get csv --selector=operators.coreos.com/smart-gateway-operator.service-telemetry
No resources found in service-telemetry namespace.
```

8.1.3. AMQ Certificate Manager Operator の削除

手順

1. AMQ Certificate Manager Operator サブスクリプションを削除します。

```
$ oc delete sub --namespace openshift-operators --selector=operators.coreos.com/amq7-
cert-manager-operator.openshift-operators
subscription.operators.coreos.com "amq7-cert-manager-operator" deleted
```

2. AMQ Certificate Manager Operator **ClusterServiceVersion** を削除します。

```
$ oc delete csv --namespace openshift-operators --selector=operators.coreos.com/amq7-
cert-manager-operator.openshift-operators

clusterserviceversion.operators.coreos.com "amq7-cert-manager.v1.0.11" deleted
```

検証

1. AMQ Certificate Manager Operator デプロイメントが実行されていないことを確認します。

```
$ oc get deploy --namespace openshift-operators --selector=operators.coreos.com/amq7-
cert-manager-operator.openshift-operators

No resources found in openshift-operators namespace.
```

2. AMQ Certificate Manager Operator サブスクリプションが存在しないことを確認します。

```
$ oc get sub --namespace openshift-operators --selector=operators.coreos.com/amq7-cert-
manager-operator.service-telemetry

No resources found in openshift-operators namespace.
```

3. AMQ Certificate Manager オペレータークラスターサービスバージョンが存在しないことを確認します。

```
$ oc get csv --namespace openshift-operators --selector=operators.coreos.com/amq7-cert-
manager-operator.openshift-operators

No resources found in openshift-operators namespace.
```

8.1.4. Grafana オペレーターの削除

手順

1. Grafana オペレーターのサブスクリプションを削除します。

```
$ oc delete sub --selector=operators.coreos.com/grafana-operator.service-telemetry
subscription.operators.coreos.com "grafana-operator" deleted
```

2. Grafana オペレーター **ClusterServiceVersion** を削除します。

```
$ oc delete csv --selector=operators.coreos.com/grafana-operator.service-telemetry
clusterserviceversion.operators.coreos.com "grafana-operator.v3.10.3" deleted
```

検証

1. Grafana Operator デプロイメントが実行されていないことを確認します。

```
$ oc get deploy --selector=operators.coreos.com/grafana-operator.service-telemetry

No resources found in service-telemetry namespace.
```

2. Grafana Operator サブスクリプションが存在しないことを確認します。

```
$ oc get sub --selector=operators.coreos.com/grafana-operator.service-telemetry
```

No resources found in service-telemetry namespace.

3. Grafana オペレータークラスターサービスバージョンが存在しないことを確認します。

```
$ oc get csv --selector=operators.coreos.com/grafana-operator.service-telemetry
```

No resources found in service-telemetry namespace.

8.2. RED HAT OPENSIFT CONTAINER PLATFORM の 4.10 へのアップグレード

Service Telemetry Framework (STF) 1.5 は、Red Hat OpenShift Container Platform 4.10 とのみ互換性があります。Red Hat OpenShift Container Platform を 4.8 から 4.10 にアップグレードする方法の詳細については、[クラスターの更新の概要](#) を参照してください。

前提条件

- STF 1.4 オペレーターを削除しました。
- AMQ Certificate Manager Operator と Grafana Operator を削除しました。Operator API は 4.10 と互換性がないため、Red Hat OpenShift Container Platform をアップグレードする前に Operator を削除する必要があります。Red Hat OpenShift Container Platform を 4.8 から 4.10 にアップグレードする準備の詳細については、[OpenShift Container Platform の更新について](#) を参照してください。
- Red Hat OpenShift Container Platform アップグレードの適合性を確認します。

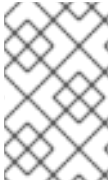
```
$ oc adm upgrade
```

次のエラーメッセージが表示された場合は、クラスターをアップグレードできません。

```
Cluster operator operator-lifecycle-manager should not be upgraded between minor versions:
ClusterServiceVersions blocking cluster upgrade: service-telemetry/grafana-operator.v3.10.3
is incompatible with OpenShift minor versions greater than 4.8,openshift-operators/amq7-
cert-manager.v1.0.11 is incompatible with OpenShift minor versions greater than 4.8
```

8.3. SERVICE TELEMETRY FRAMEWORK 1.5 オペレーターのインストール

Red Hat OpenShift Container Platform 4.10 環境に Service Telemetry Framework (STF) 1.5 Operator と OpenShift Operator 用の Certificate Manager をインストールします。STF 1.5 は、Red Hat OpenShift Container Platform 4.10 のみをサポートします。切断されたネットワーク環境または制限されたネットワーク環境への STF のインストールはサポートされていません。



注記

STF 1.5 のインストールが正常に完了したら、AMQ Interconnect CA 証明書を取得して Red Hat OpenStack Platform 環境に適用する必要があります。そうしないと、トランスポート層とテレメトリデータが使用できなくなります。

AMQ Interconnect CA 証明書の更新の詳細については、次を参照してください。 [「Red Hat OpenStack Platform での AMQ Interconnect CA 証明書の更新」](#)。

前提条件

- Red Hat OpenShift Container Platform 環境を 4.10 にアップグレードしている。Red Hat OpenShift Container Platform のアップグレードの詳細については、次を参照してください。 [「Red Hat OpenShift Container Platform の 4.10 へのアップグレード」](#)。
- Red Hat OpenShift Container Platform 環境のネットワークは完全に接続されています。

手順

1. **service-telemetry** プロジェクトに変更します。

```
$ oc project service-telemetry
```

2. **cert-manager** Operator の **namespace** を作成します。

```
$ oc create -f - <<EOF
apiVersion: project.openshift.io/v1
kind: Project
metadata:
  name: openshift-cert-manager-operator
spec:
  finalizers:
    - kubernetes
EOF
```

3. cert-manager Operator の **OperatorGroup** を作成します。

```
$ oc create -f - <<EOF
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: openshift-cert-manager-operator
  namespace: openshift-cert-manager-operator
spec: {}
EOF
```

4. **redhat-operators** CatalogSource で **cert-manager** Operator にサブスクライブします。

```
$ oc create -f - <<EOF
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: openshift-cert-manager-operator
  namespace: openshift-cert-manager-operator
```

```
spec:
  channel: tech-preview
  installPlanApproval: Automatic
  name: openshift-cert-manager-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
EOF
```

5. **ClusterServiceVersion** を検証します。**cert-manager** Operator のフェーズが **Succeeded** であることを確認します。

```
$ oc get csv --namespace openshift-cert-manager-operator --
selector=operators.coreos.com/openshift-cert-manager-operator.openshift-cert-manager-
operator
```

NAME	DISPLAY	VERSION	REPLACES	PHASE
openshift-cert-manager.v1.7.1	cert-manager Operator	for Red Hat OpenShift	1.7.1-1	Succeeded

6. オプション: Grafana Operator に再サブスクリブします。詳細は、テスト「[ダッシュボードをホストするための Grafana の設定](#)」を参照してください。
7. Service Telemetry Operator サブスクリプションを作成し、STF インスタンスを管理します。

```
$ oc create -f - <<EOF
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: service-telemetry-operator
  namespace: service-telemetry
spec:
  channel: stable-1.5
  installPlanApproval: Automatic
  name: service-telemetry-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
EOF
```

8. Service Telemetry Operator および依存する Operator を検証します。

```
$ oc get csv --namespace service-telemetry
```

NAME	DISPLAY	VERSION
REPLACES	PHASE	
amq7-interconnect-operator.v1.10.13	Red Hat Integration - AMQ Interconnect	
1.10.13	amq7-interconnect-operator.v1.10.4	Succeeded
elasticsearch-eck-operator-certified.v2.5.0	Elasticsearch (ECK) Operator	2.5.0
elasticsearch-eck-operator-certified.v2.4.0	Succeeded	
openshift-cert-manager.v1.7.1	cert-manager Operator	for Red Hat OpenShift
1.7.1-1	Succeeded	
prometheusoperator.0.47.0	Prometheus Operator	0.47.0
prometheusoperator.0.37.0	Succeeded	
service-telemetry-operator.v1.5.1669950702	Service Telemetry Operator	

```

1.5.1669950702           Succeeded
smart-gateway-operator.v5.0.1669950681 Smart Gateway Operator
5.0.1669950681           Succeeded

```

検証

- Service Telemetry Operator が正常に調整されたことを確認します。

```

$ oc logs -f --selector=name=service-telemetry-operator

[...]
---- Ansible Task Status Event StdOut (infra.watch/v1beta1, Kind=ServiceTelemetry,
default/service-telemetry) ----

PLAY RECAP *****
localhost           : ok=115 changed=0  unreachable=0  failed=0  skipped=21
rescued=0  ignored=0

$ oc get pods
NAME                                READY STATUS  RESTARTS   AGE
alertmanager-default-0             3/3   Running  0          20h
default-cloud1-ceil-event-smartgateway-6d57ffbdbc-5mrj8  2/2   Running  1 (3m42s ago)
4m21s
default-cloud1-ceil-meter-smartgateway-67684d88c8-62mp7  3/3   Running  1 (3m43s
ago) 4m20s
default-cloud1-coll-event-smartgateway-66cddd5567-qb6p2  2/2   Running  1 (3m42s ago)
4m19s
default-cloud1-coll-meter-smartgateway-76d5ff6db5-z5ch8  3/3   Running  0
4m20s
default-cloud1-sens-meter-smartgateway-7b59669fdd-c42zg  3/3   Running  1 (3m43s
ago) 4m20s
default-interconnect-845c4b647c-wwfcm                    1/1   Running  0          4m10s
elastic-operator-57b57964c5-6q88r                       1/1   Running  8 (17h ago) 20h
elasticsearch-es-default-0                              1/1   Running  0          21h
grafana-deployment-59c54f7d7c-zjnhm                    2/2   Running  0          20h
interconnect-operator-848889bf8b-bq2zx                  1/1   Running  0          20h
prometheus-default-0                                    3/3   Running  1 (20h ago) 20h
prometheus-operator-5d7b69fd46-c2xlv                   1/1   Running  0          20h
service-telemetry-operator-79fb664dfb-nj8jn             1/1   Running  0          5m11s
smart-gateway-operator-79557664f8-ql7xr                 1/1   Running  0          5m7s

```

8.4. RED HAT OPENSTACK PLATFORM での AMQ INTERCONNECT CA 証明書の更新

Service Telemetry Framework (STF) v1.5 にアップグレードすると、AMQ Interconnect の CA 証明書が再生成されます。STF v1.4 では、AMQ Interconnect の CA 証明書は 3 か月間有効であり、Red Hat OpenStack Platform (RHOSP) で定期的に更新する必要があります。STF v1.5 では、生成された証明書は RHOSP ライフサイクルの有効期間 (デフォルトでは 70080 時間) 有効です。

前提条件

- STF v1.5 が正常にインストールされ、AMQ Interconnect の CA 証明書が更新されました。

手順

- RHOSP 環境で CA 証明書を更新する方法の詳細については、次を参照してください。[6章AMQ Interconnect 証明書の更新](#)