Red Hat OpenStack Platform 17.1

Service Telemetry Framework 1.5

安装和部署 Service Telemetry Framework 1.5

Last Updated: 2024-06-22

Red Hat OpenStack Platform 17.1 Service Telemetry Framework 1.5

安装和部署 Service Telemetry Framework 1.5

OpenStack Team
rhos-docs@redhat.com

法律通告

摘要

安装核心组件并部署 Service Telemetry Framework 1.5

目录

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看 CTO Chris Wright 的信息。

对红帽文档提供反馈

我们感谢您对文档提供反馈信息。与我们分享您的成功秘诀。

在 JIRA 中提供文档反馈

使用 Create Issue 表单对文档提供反馈。JIRA 问题将在 Red Hat OpenStack Platform Jira 项目中创建，您可以在其中跟踪您的反馈进度。

1. 确保您已登录到 JIRA。如果您没有 JIRA 帐户，请创建一个帐户来提交反馈。
2. 点击以下链接打开 **Create Issue** 页面： Create Issue
3. 完成 **Summary** 和 **Description** 字段。在 **Description** 字段中，包含文档 URL、章节或章节号以及问题的详细描述。不要修改表单中的任何其他字段。
4. 点 **Create**。

第 1 章 SERVICE TELEMETRY FRAMEWORK 1.5 简介

Service Telemetry Framework (STF)从 Red Hat OpenStack Platform (RHOSP)或第三方节点收集监控数据。您可以使用 STF 执行以下任务：

- 存储或归档监控数据以获取历史信息。
- 在控制面板中以图形方式查看监控数据。
- 使用监控数据来触发警报或警告。

监控数据可以是 metric 或 event：

指标

应用程序或系统的数字测量。

事件

系统中发生异常和离散性发生。

STF 的组件使用消息总线进行数据传输。接收和存储数据的其他模块化组件作为容器部署在 Red Hat OpenShift Container Platform 上。

重要

STF 与 Red Hat OpenShift Container Platform Extended Update Support (EUS)版本 4.12 和 4.14 兼容。

其他资源

- Red Hat OpenShift Container Platform 产品文档
- Service Telemetry Framework 性能和扩展
- OpenShift Container Platform 4.14 文档
- Red Hat OpenShift Container Platform 生命周期政策

1.1. 支持 SERVICE TELEMETRY FRAMEWORK

红帽支持核心 Operator 和工作负载，包括 AMQ Interconnect、Cluster Observability Operator (Prometheus、Alertmanager)、Service Telemetry Operator 和 Smart Gateway Operator。红帽不支持社区 Operator 或工作负载组件，包括 Elasticsearch、Grafana 及其 Operator。

您可以在完全连接的网络环境或 Red Hat OpenShift Container Platform 断开连接环境中部署 Service Telemetry Framework (STF)。您无法在网络代理环境中部署 STF。

有关 STF 生命周期和支持状态的更多信息，请参阅 Service Telemetry Framework 支持的版本列表。

1.2. SERVICE TELEMETRY FRAMEWORK 架构

Service Telemetry Framework (STF)使用客户端-服务器架构，其中的 Red Hat OpenStack Platform (RHOSP)是客户端，Red Hat OpenShift Container Platform 是服务器。

默认情况下，STF 会收集、传输和存储指标信息。

您可以收集 RHOSP 事件数据，使用消息总线传输它，并将其转发到智能网关中的用户提供的 Elasticsearch，但这个选项已弃用。

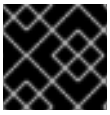
STF 由以下组件组成：

- 数据收集
  - collectd：在 RHOSP 上收集基础架构指标和事件。
  - Ceilometer：在 RHOSP 上收集 RHOSP 指标和事件。
- 传输
  - AMQ Interconnect ：一个 AMQP 1.x 兼容消息传递总线，它提供快速可靠的数据传输，将指标从 RHOSP 传送到 STF 以进行存储或转发。
  - 智能网关： Golang 应用从 AMQP 1.x 总线获取指标和事件，以提供给 Prometheus 或外部 Elasticsearch。
- 数据存储
  - Prometheus：存储从智能网关接收的 STF 指标的时间序列数据存储。
  - Alertmanager：使用 Prometheus 警报规则管理警报的警报工具。
- 用户提供的组件
  - Grafana：可用于查询、视觉化和探索数据的视觉化和分析应用程序。
  - Elasticsearch：存储由智能网关接收和转发 RHOSP 事件的事件数据存储。

下表描述了客户端和服务器组件的应用程序：

表 1.1. STF 的客户端和服务器组件

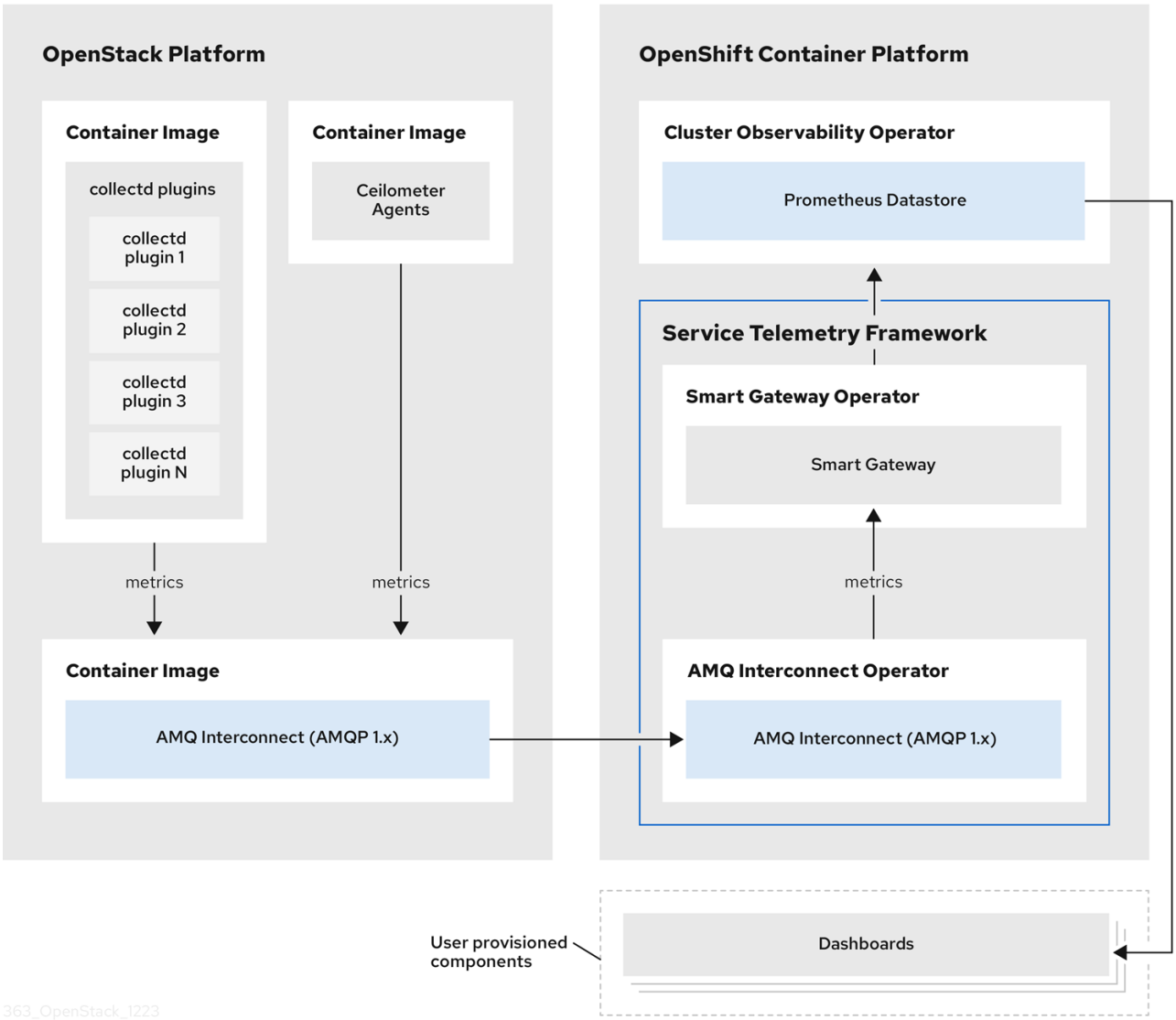
组件	客户端	Server
AMQP 1.x 兼容消息传递总线	是	是
智能网关	否	是
Prometheus	否	是
Elasticsearch	否	是
Grafana	否	是
collectd	是	否
ilo	是	否



重要

为确保监控平台可以报告云中的操作问题，请不要在您监控的同一基础架构上安装 STF。

图 1.1. Service Telemetry Framework 架构概述



对于客户端侧指标，collectd 提供没有项目数据的基础架构指标，Ceilometer 根据项目或用户工作负载提供 RHOSP 平台数据。Ceilometer 和 collectd 通过使用 AMQ Interconnect 传输将数据提供给 Prometheus，并通过消息总线提供数据。在服务器端，名为 Smart Gateway 的 Golang 应用从总线获取数据流，并将其公开为 Prometheus 的本地提取端点。

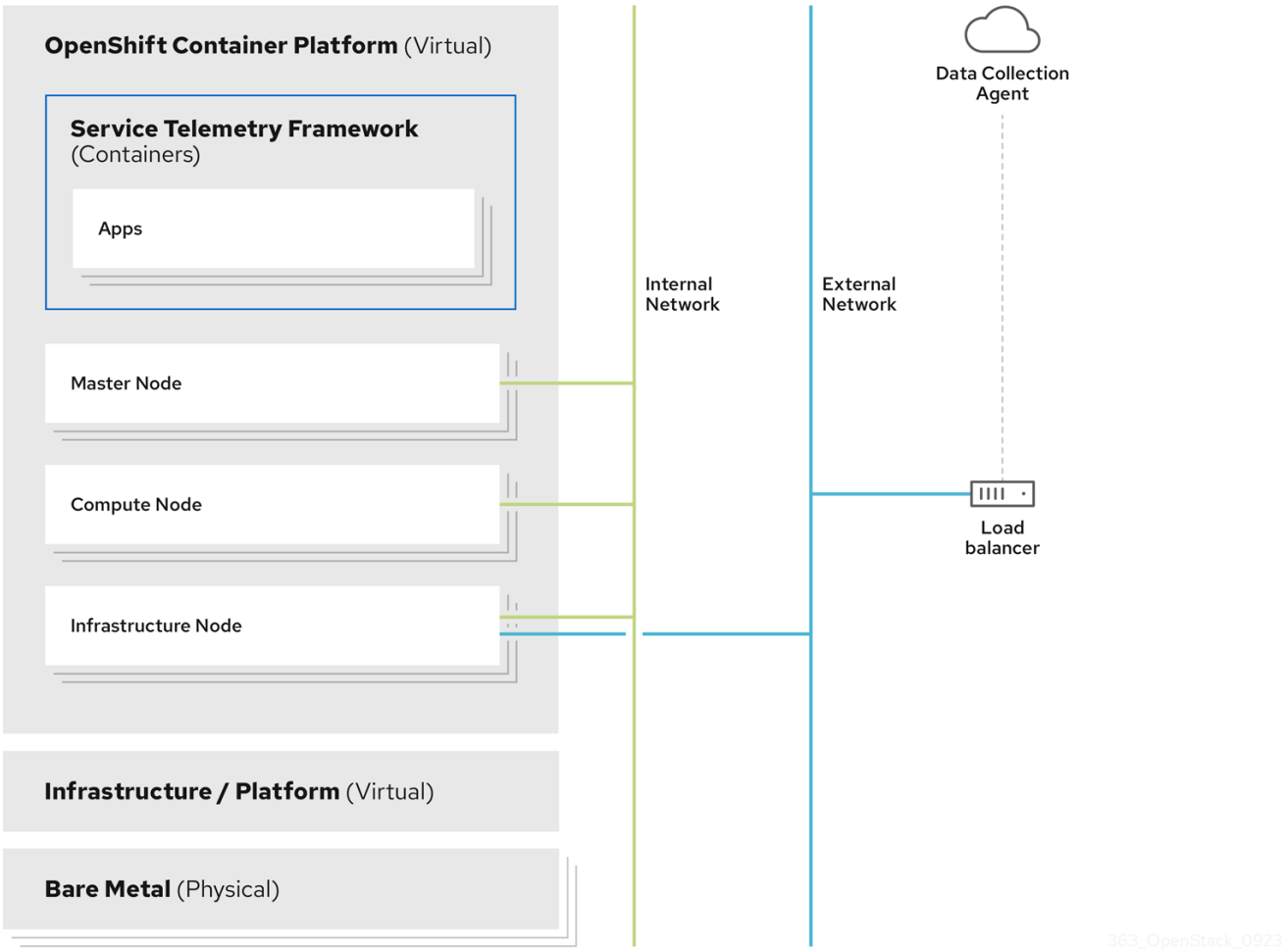
当您收集和存储事件时，collectd 和 Ceilometer 使用 AMQ Interconnect 传输向服务器端发送事件数据。另一个智能网关将数据转发到用户提供的 Elasticsearch 数据存储。

服务器端 STF 监控基础架构由以下层组成：

- Service Telemetry Framework 1.5
- Red Hat OpenShift Container Platform Extended Update Support (EUS)版本 4.12 和 4.14
- 基础架构平台

如需有关 Red Hat OpenShift Container Platform EUS 版本的更多信息，请参阅 Red Hat OpenShift Container Platform 生命周期政策。

图 1.2. 服务器端 STF 监控基础架构



1.2.1. STF 架构变化

在 1.5.3 之前的 STF 版本中，Service Telemetry Operator 从 Kubernetes (ECK) Operator 上请求 Elasticsearch 实例。STF 现在使用转发模型，其中事件从智能网关实例转发到用户提供的 Elasticsearch 实例。

注意

通过 Service Telemetry Operator 管理 Elasticsearch 实例已弃用。

在新的 **ServiceTelemetry** 部署中，**observabilityStrategy** 参数的值为 **use_redhat**，它不从 ECK 请求 Elasticsearch 实例。带有 STF 版本 1.5.2 或更早版本的 **ServiceTelemetry** 的部署，并更新至 1.5.3，将 **observabilityStrategy** 参数设置为 **use_community**，它与前面的架构匹配。

如果用户之前使用 STF 部署 Elasticsearch 实例，Service Telemetry Operator 会更新 **ServiceTelemetry** 自定义资源对象，将 **observabilityStrategy** 参数设置为 **use_community**，以及与之前的版本类似的功能。有关可观察性策略的更多信息，请参阅 第 2.1 节 “Service Telemetry Framework 中的可观察性策略”。

建议 STF 用户迁移到 **use_redhat** observability 策略。有关迁移到 **use_redhat** observability 策略的更多信息，请参阅红帽知识库文章 将 Service Telemetry Framework 迁移到完全支持的 operator。

1.3. RED HAT OPENSHIFT CONTAINER PLATFORM 的安装大小

Red Hat OpenShift Container Platform 安装的大小取决于以下因素：

- 您选择的基础架构。
- 要监控的节点数量。
- 要收集的指标数量。
- 指标的解析。
- 要存储数据的时间长度。

Service Telemetry Framework (STF)的安装依赖于现有的 Red Hat OpenShift Container Platform 环境。

有关 *在裸机上安装 Red Hat OpenShift Container Platform 时 最低资源要求 的更多信息，请参阅在裸机上安装集群指南中的最低资源要求*。有关您可以安装的各种公共和私有云平台的安装要求，请参阅您选择的云平台的相应安装文档。

第 2 章 为 SERVICE TELEMETRY FRAMEWORK 准备 RED HAT OPENSHIFT CONTAINER PLATFORM 环境

要为 Service Telemetry Framework (STF)准备 Red Hat OpenShift Container Platform 环境，您必须规划持久性存储、适当的资源、事件存储和网络注意事项：

- 确保 Red Hat OpenShift Container Platform 集群中为生产环境级部署提供持久性存储。更多信息请参阅 第 2.2 节 “持久性卷（PV）”。
- 确保有足够的资源来运行 Operator 和应用程序容器。更多信息请参阅 第 2.3 节 “资源分配”。

2.1. SERVICE TELEMETRY FRAMEWORK 中的可观察性策略

Service Telemetry Framework (STF)不包括事件存储后端或仪表板工具。STF 可以使用社区操作器为 Grafana 创建数据源配置，以提供仪表板接口。

除了让 Service Telemetry Operator 创建自定义资源请求外，您可以使用您自己的应用程序部署或其他兼容应用程序，并提取指标 智能网关以发送到您自己的 Prometheus 兼容系统以进行遥测存储。如果将 **observabilityStrategy** 设置为 **none**，则不会部署存储后端，因此 STF 不需要持久性存储。

使用 STF 对象中的 observabilityStrategy 属性来指定要部署的可观察性组件类型。

可用的值如下：

value	含义
use_redhat	红帽支持的组件由 STF 请求。这包括 Cluster Observability Operator 中的 Prometheus 和 Alertmanager，但没有对 Kubernetes (ECK) Operator 上的 Elastic Cloud 的请求。如果启用，也会从 Grafana Operator （社区组件）请求资源。
use_hybrid	除了红帽支持的组件外，还会请求 Elasticsearch 和 Grafana 资源（如果在 ServiceTelemetry 对象中指定）
use_community	使用 Prometheus Operator 的社区版本，而不是 Cluster Observability Operator。还请求 Elasticsearch 和 Grafana 资源（如果在 ServiceTelemetry 对象中指定）
none	没有部署存储或警报组件

注意

新部署的 STF 环境为 1.5.3 默认为 **use_redhat**。在 1.5.3 默认之前创建的现有 STF 部署 **以使用_community**。

要将现有 STF 部署迁移到 **use_redhat**，请参阅红帽知识库文章将服务 Telemetry Framework 迁移到完全支持的操作器。

2.2. 持久性卷（PV）

Service Telemetry Framework (STF)使用 Red Hat OpenShift Container Platform 中的持久性存储来请求持久性卷，以便 Prometheus 可以存储指标数据。

当您通过 Service Telemetry Operator 启用持久性存储时，STF 部署中请求的持久性卷声明(PVC)会导致访问模式为 RWO (ReadWriteOnce)。如果您的环境包含预置备的持久性卷，请确保 Red Hat OpenShift Container Platform 默认配置的 **storageClass** 中提供了 RWO 卷。

其他资源

- 有关为 Red Hat OpenShift Container Platform 配置持久性存储的更多信息，请参阅了解持久性存储。
- 有关在 Red Hat OpenShift Container Platform 中推荐的可配置存储技术的更多信息，请参阅 推荐的可配置存储技术。
- 有关在 STF 中为 Prometheus 配置持久性存储的更多信息，请参阅 "为 Prometheus 配置持久性存储"一节。

2.3. 资源分配

要启用在 Red Hat OpenShift Container Platform 基础架构中调度 pod，您需要运行的组件的资源。如果您没有分配足够的资源，pod 会一直处于 **Pending** 状态，因为它们无法调度。

运行 Service Telemetry Framework (STF)所需的资源数量取决于您的环境以及要监控的节点数量。

其他资源

- 有关指标集合大小的建议，请参阅红帽知识库文章 Service Telemetry Framework 性能和扩展。

2.4. SERVICE TELEMETRY FRAMEWORK 的网络注意事项

您可以在完全连接的网络环境或 Red Hat OpenShift Container Platform 断开连接环境中部署 Service Telemetry Framework (STF)。您无法在网络代理环境中部署 STF。

2.5. 在 RED HAT OPENSHIFT CONTAINER PLATFORM 断开连接环境中部署 STF

自 Service Telemetry Framework (STF)版本 1.5.4 开始，您可以在 Red Hat OpenShift Container Platform 断开连接环境中部署 STF。

先决条件

- 在受限网络中部署的 Red Hat OpenShift Container Platform Extended Update Support (EUS)版本 4.12 或 4.14。
- 一个镜像 registry，以便 Red Hat OpenShift Container Platform 集群可以访问所需的镜像。如需有关镜像 registry 的更多信息，请参阅 Red Hat OpenShift Container Platform 安装指南中的 断开连接的安装镜像。
- *Red Hat OpenShift Container Platform 集群镜像 registry 中提供了所有 STF 依赖项。*

在镜像 registry 中添加 STF 依赖项

*您可以使用 **oc-mirror** 插件获取 STF 依赖项，并将其添加到 Red Hat OpenShift Container Platform 集群镜像 registry 中。有关安装 **oc-mirror** 插件的更多信息，请参阅 Red Hat OpenShift Container Platform 安装指南中的使用 oc-mirror 插件为断开连接的安装镜像镜像。*

流程

1. *在本地工作目录中创建 **imagesetconfig.yaml** 文件：*

   imagesetconfig.yaml

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  local:
    path: ./
mirror:
  operators:
  - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.14
    packages:
      - name: service-telemetry-operator
        channels:
          - name: stable-1.5
      - name: openshift-cert-manager-operator
        channels:
          - name: stable-v1
      - name: amq7-interconnect-operator
        channels:
          - name: 1.10.x
      - name: smart-gateway-operator
        channels:
          - name: stable-1.5
      - name: cluster-observability-operator
        channels:
          - name: development
```

2. *（可选）如果镜像 registry 无法访问，您可以保存通过 **oc-mirror** 获取的清单和镜像，并将其物理传输到镜像 registry 和 Red Hat OpenShift Container Platform 集群。否则，您可以运行 **oc-mirror** 并指向镜像 registry。*
   *您可以根据您的环境的不同使用 **oc-mirror** 插件，例如：*

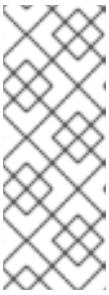
   - *镜像之间的镜像(mirror)。*
   - *从 mirror 到磁盘进行镜像(mirror)。*
   - *从磁盘镜像到镜像。*
     *有关不同 **oc-mirror** 场景的更多信息，请参阅 Red Hat OpenShift Container Platform 安装指南中的 完全断开连接的环境中镜像镜像集。*

3. *从镜像 registry 中推送 STF operator 及其依赖项，并为 Red Hat OpenShift Container Platform 集群生成清单。*

```
$ oc-mirror --config imagesetconfig.yaml <mirror_registry_location>
```

   - *将 <mirror_registry_location> 替换为您要使用的镜像 registry 的文件路径。*

4. 找到生成的清单，并将其应用到目标 Red Hat OpenShift Container Platform 集群。如需更多信息，请参阅 Red Hat OpenShift Container Platform 安装指南中的将集群配置为使用 oc-mirror 生成的资源。



注意

使用 **oc-mirror** 生成的清单使用完整索引名称生成目录，如 **redhat-operator-index** 而不是 **redhat-operators** 用于 **CatalogSource**。确保为 STF 订阅使用正确的索引名称。如需更多信息，请参阅 第 3.1 节 “在 Red Hat OpenShift Container Platform 环境中部署 Service Telemetry Framework”。有关使用 oc mirror 自定义 Operator 的更多信息，请参阅红帽知识库解决方案 如何使用 oc mirror 插件自定义 Operator 的目录名称和标签镜像到镜像 registry。

验证

- 检查是否应用了目录源。您可以返回引用 STF operator 及其依赖项的新目录条目：

```
$ oc get catalogsources
```

- 您已在断开连接的 Red Hat OpenShift Container Platform 集群中部署了 STF，因此无法访问外部网络。

第 3 章 安装 SERVICE TELEMETRY FRAMEWORK 的核心组件

您可以使用 Operator 加载 Service Telemetry Framework (STF) 组件和对象。Operator 管理以下每个 STF 核心组件：

- 证书管理
- AMQ Interconnect
- 智能网关
- Prometheus 和 Alertmanager

Service Telemetry Framework (STF) 使用其他支持的 Operator 作为部署的一部分。STF 可以自动解决大多数依赖项，但您需要预安装一些 Operator，如 Cluster Observability Operator，它提供 Prometheus 和 Alertmanager 实例，以及 cert-manager for Red Hat OpenShift，它提供管理证书。

先决条件

- 一个 Red Hat OpenShift Container Platform 延长更新支持(EUS) 发行版本 4.12 或 4.14 正在运行。
- 您已准备了 Red Hat OpenShift Container Platform 环境，并确保有持久性存储和充足的资源可在 Red Hat OpenShift Container Platform 环境上运行 STF 组件。有关 STF 性能的更多信息，请参阅红帽知识库文章 Service Telemetry Framework 性能和扩展。
- 您已在完全连接或 Red Hat OpenShift Container Platform 的情况下部署了 STF。在网络代理环境中无法使用 STF。

重要

STF 与 Red Hat OpenShift Container Platform 版本 4.12 和 4.14 兼容。

其他资源

- 如需有关 Operator 的更多信息，请参阅了解 Operator 指南。
- 如需有关 Operator 目录的更多信息，请参阅红帽提供的 Operator 目录。
- 有关 Red Hat 的 cert-manager Operator 的更多信息，请参阅 cert-manager Operator for Red Hat OpenShift overview。
- 如需有关 Cluster Observability Operator 的更多信息，请参阅 Cluster Observability Operator 概述。
- 如需有关 OpenShift 生命周期政策和延长更新支持(EUS) 的更多信息，请参阅 Red Hat OpenShift Container Platform 生命周期政策。

3.1. 在 RED HAT OPENSHIFT CONTAINER PLATFORM 环境中部署 SERVICE TELEMETRY FRAMEWORK

部署 Service Telemetry Framework (STF) 以收集和存储 Red Hat OpenStack Platform (RHOSP) 遥测。

3.1.1. 部署 Cluster Observability Operator

如果 **observabilityStrategy** 设置为 **use_redhat**，且在 ServiceTelemetry 对象中将 backend. **metrics.prometheus.enabled** 设置为 **true**，则必须在创建 **Service Telemetry** Framework (STF) 实例前安装 Cluster Observability Operator (COO)。如需有关 COO 的更多信息，请参阅 OpenShift Container Platform 文档中的 Cluster Observability Operator 概述。

流程

1. 登录到托管 STF 的 Red Hat OpenShift Container Platform 环境。
2. 要在 Prometheus 中存储指标，请使用 **redhat-operators** CatalogSource 启用 Cluster Observability Operator：

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

3. 验证 Cluster Observability Operator 的 **ClusterServiceVersion** 的状态是否为 **Succeeded**：

```
$ oc wait --for jsonpath="{.status.phase}"=Succeeded csv --namespace=openshift-operators
-l operators.coreos.com/cluster-observability-operator.openshift-operators

clusterserviceversion.operators.coreos.com/observability-operator.v0.0.26 condition met
```

3.1.2. 为 Red Hat OpenShift 部署 cert-manager

在创建 Service Telemetry Framework (STF) 实例之前，必须先预安装 Red Hat OpenShift (cert-manager) Operator 的 cert-manager。有关 cert-manager 的更多信息，请参阅 cert-manager for Red Hat OpenShift 概述。

在以前的 STF 版本中，唯一可用的 cert-manager 频道 **是技术预览**，直到 Red Hat OpenShift Container Platform v4.12 为止。在 Red Hat OpenShift Container Platform v4.14 及更新版本上安装 cert-manager 必须从 **stable-v1** 频道安装。对于 STF 的新安装，建议从 **stable-v1** 频道安装 cert-manager。

警告

每个 Red Hat OpenShift Container Platform 集群只能安装一个 cert-manager 部署。在多个项目中订阅 cert-manager 会导致部署相互冲突。

流程

1. 登录到托管 STF 的 Red Hat OpenShift Container Platform 环境。
2. 验证 Red Hat OpenShift Container Platform 集群中尚未安装 cert-manager。如果返回任何结果，请不要安装另一个 cert-manager 实例：

```
$ oc get sub --all-namespaces -o json | jq '.items[] | select(.metadata.name | match("cert-manager")) | .metadata.name'
```

3. 为 cert-manager Operator 创建命名空间：

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

4. 为 cert-manager Operator 创建 OperatorGroup：

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

5. 使用 redhat-operators CatalogSource 订阅 cert-manager Operator：

```
$ oc create -f - <<EOF
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: openshift-cert-manager-operator
  namespace: cert-manager-operator
  labels:
    operators.coreos.com/openshift-cert-manager-operator.cert-manager-operator: ""
spec:
  channel: stable-v1
  installPlanApproval: Automatic
  name: openshift-cert-manager-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
EOF
```

6. 验证您的 ClusterServiceVersion。确保 cert-manager Operator 显示 **Succeeded** 的阶段：

```
oc wait --for jsonpath="{.status.phase}"=Succeeded csv --namespace=cert-manager-operator --selector=operators.coreos.com/openshift-cert-manager-operator.cert-manager-operator

clusterserviceversion.operators.coreos.com/cert-manager-operator.v1.12.1 condition met
```

3.1.3. 部署 Service Telemetry Operator

在 Red Hat OpenShift Container Platform 上部署 Service Telemetry Operator，为创建 Service Telemetry Framework (STF)实例提供支持，以监控 Red Hat OpenStack Platform (RHOSP)云平台。

先决条件

- 已安装 Cluster Observability Operator 以允许存储指标。如需更多信息，请参阅 第 3.1.1 节 "部署 Cluster Observability Operator"。
- 您已安装了 Red Hat OpenShift 的 cert-manager 以允许证书管理。如需更多信息，请参阅 第 3.1.2 节 "为 Red Hat OpenShift 部署 cert-manager"。

流程

1. 创建一个命名空间来包含 STF 组件，如 **service-telemetry**：

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

2. 在命名空间中创建 OperatorGroup，以便您可以调度 Operator pod：

```
$ 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
```

如需更多信息，请参阅 OperatorGroup。

3. 创建 Service Telemetry Operator 订阅来管理 STF 实例：

```
$ 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
```

4. 验证 Service Telemetry Operator 和依赖 Operator 的阶段为 Succeeded ：

```
$ oc wait --for jsonpath="{.status.phase}"=Succeeded csv --namespace=service-telemetry -l operators.coreos.com/service-telemetry-operator.service-telemetry ; oc get csv --namespace service-telemetry

clusterserviceversion.operators.coreos.com/service-telemetry-operator.v1.5.1700688542 condition met

NAME                                         DISPLAY                          VERSION        REPLACES                                     PHASE
amq7-interconnect-operator.v1.10.17          Red Hat Integration - AMQ Interconnect   1.10.17   amq7-interconnect-operator.v1.10.4   Succeeded
observability-operator.v0.0.26               Cluster Observability Operator           0.1.0                                                Succeeded
service-telemetry-operator.v1.5.1700688542   Service Telemetry Operator               1.5.1700688542                                       Succeeded
smart-gateway-operator.v5.0.1700688539       Smart Gateway Operator                   5.0.1700688539                                       Succeeded
```

3.2. 在 RED HAT OPENSHIFT CONTAINER PLATFORM 中创建 SERVICETELEMETRY 对象

在 Red Hat OpenShift Container Platform 中创建 **ServiceTelemetry** 对象，以便 Service Telemetry Operator 为 Service Telemetry Framework (STF) 部署创建支持组件。更多信息请参阅 第 3.2.1 节 “ServiceTelemetry 对象的主要参数”。

先决条件

- 您已部署了 STF 和支持的 operator。如需更多信息，请参阅 第 3.1 节 “在 Red Hat OpenShift Container Platform 环境中部署 Service Telemetry Framework”。
- 已安装 Cluster Observability Operator 以允许存储指标。如需更多信息，请参阅 第 3.1.1 节 “部署 Cluster Observability Operator”。
- 您已安装了 Red Hat OpenShift 的 cert-manager 以允许证书管理。如需更多信息，请参阅 第 3.1.2 节 “为 Red Hat OpenShift 部署 cert-manager” 。

流程

1. 登录到托管 STF 的 Red Hat OpenShift Container Platform 环境。
2. 要部署 STF 会导致要配置指标交付的核心组件，请创建一个 **ServiceTelemetry** 对象：

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

```
spec:
  alerting:
    alertmanager:
      storage:
        persistent:
          pvcStorageRequest: 20G
        strategy: persistent
    enabled: true
  backends:
    metrics:
      prometheus:
        enabled: true
        scrapeInterval: 30s
        storage:
          persistent:
            pvcStorageRequest: 20G
          retention: 24h
          strategy: persistent
  clouds:
  - metrics:
      collectors:
      - bridge:
          ringBufferCount: 15000
          ringBufferSize: 16384
          verbose: false
        collectorType: collectd
        debugEnabled: false
        subscriptionAddress: collectd/cloud1-telemetry
      - bridge:
          ringBufferCount: 15000
          ringBufferSize: 16384
          verbose: false
        collectorType: ceilometer
        debugEnabled: false
        subscriptionAddress: anycast/ceilometer/cloud1-metering.sample
      - bridge:
          ringBufferCount: 15000
          ringBufferSize: 65535
          verbose: false
        collectorType: sensubility
        debugEnabled: false
        subscriptionAddress: sensubility/cloud1-telemetry
    name: cloud1
  observabilityStrategy: use_redhat
  transports:
    qdr:
      auth: basic
      certificates:
        caCertDuration: 70080h
        endpointCertDuration: 70080h
      enabled: true
      web:
        enabled: false
EOF
```

要覆盖这些默认值，请将配置添加到 ***spec*** *参数中。*

3. 在 Service Telemetry Operator 中查看 STF 部署日志：

```
$ 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
```

验证

- 要确定所有工作负载是否都正常运行，请查看 pod 和每个 pod 的状态。

```
$ oc get pods

NAME                                              READY   STATUS    RESTARTS   AGE
alertmanager-default-0                            3/3     Running   0          123m
default-cloud1-ceil-meter-smartgateway-7dfb95fcb6-bs6jl    3/3     Running   0          122m
default-cloud1-coll-meter-smartgateway-674d88d8fc-858jk    3/3     Running   0          122m
default-cloud1-sens-meter-smartgateway-9b869695d-xcssf     3/3     Running   0          122m
default-interconnect-6cbf65d797-hk7l6             1/1     Running   0          123m
interconnect-operator-7bb99c5ff4-l6xc2            1/1     Running   0          138m
prometheus-default-0                              3/3     Running   0          122m
service-telemetry-operator-7966cf57f-g4tx4        1/1     Running   0          138m
smart-gateway-operator-7d557cb7b7-9ppls           1/1     Running   0          138m
```

3.2.1. ServiceTelemetry 对象的主要参数

您可以设置 **ServiceTelemetry** 对象的以下主要配置参数来配置 STF 部署：

- **警报**
- **后端**
- **云**
- **图表**
- **HighAvailability**
- **传输**

backend 参数
设置 backend **参数的值**，为指标和事件分配存储后端，并启用 **clouds** 参数定义的智能网关。如需更多信息，请参阅 "clouds 参数"一节。

您可以使用 Prometheus 作为指标存储后端，Elasticsearch 作为事件存储后端。Service Telemetry Operator 可以创建 Prometheus Operator 监视的自定义资源对象来创建 Prometheus 工作负载。您需要外部部署 Elasticsearch 来存储事件。

将 Prometheus 启用为指标的存储后端
要将 Prometheus 启用为指标的存储后端，您必须配置 **ServiceTelemetry** 对象。

流程

1. 编辑 **ServiceTelemetry** 对象：

```
$ oc edit stf default
```

2. 将 backends.metrics.prometheus.enabled 参数的值设置为 **true** ：

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

为 Prometheus 配置持久性存储

在 backend. **metrics.prometheus.storage.persistent** 中设置附加参数，以配置 Prometheus 的持久性存储选项，如存储类和卷大小。

使用 **storageClass** 参数定义后端存储类。如果没有设置此参数，Service Telemetry Operator 将使用 Red Hat OpenShift Container Platform 集群的默认存储类。

使用 **pvcStorageRequest** 参数定义存储请求所需的最小卷大小。默认情况下，Service Telemetry Operator 请求大小为 **20G** (20 Gigabytes)。

流程

1. 列出可用的存储类：

```
$ 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
```

2. 编辑 **ServiceTelemetry** 对象：

```
$ oc edit stf default
```

3. 将 **backends.metrics.prometheus.enabled** 参数的值设置为 **true**，将 **backends.metrics.prometheus.storage.strategy** 的值设置为 **persistent** ：

```
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
```

为事件启用 Elasticsearch 作为存储后端

注意

STF 的早期版本在 Kubernetes Operator (ECK)上支持社区支持的 Elastic Cloud。在 STF 1.5.3 中弃用了 Elasticsearch 管理功能。您仍然可以转发到使用 ECK 部署和管理的现有 Elasticsearch 实例，但您无法管理 Elasticsearch 对象的创建。当您升级 STF 部署时，现有的 Elasticsearch 对象和部署会保留，但不再由 STF 管理。

有关在 STF 中使用 Elasticsearch 的更多信息，请参阅 在 Elasticsearch 中使用 Service Telemetry Framework 的红帽知识库文章。

*要启用转发到 Elasticsearch 作为存储后端的事件，您必须配置 **ServiceTelemetry** 对象。*

流程

1. *编辑 **ServiceTelemetry** 对象：*

```
$ oc edit stf default
```

2. *将 **backends.events.elasticsearch.enabled** 参数的值设置为 **true**，并使用相关的 Elasticsearch 实例配置 **hostUrl** ：*

```
apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
metadata:
  name: default
  namespace: service-telemetry
spec:
  [...]
  backends:
    events:
      elasticsearch:
        enabled: true
        forwarding:
          hostUrl: https://external-elastic-http.domain:9200
          tlsServerName: ""
          tlsSecretName: elasticsearch-es-cert
          userSecretName: elasticsearch-es-elastic-user
          useBasicAuth: true
          useTls: true
```

3. 在 **userSecretName** 参数中创建名为 的secret，以存储 **基本身份验证** 凭证

```
$ oc create secret generic elasticsearch-es-elastic-user --from-
literal=elastic='<PASSWORD>'
```

4. 将 CA 证书复制到名为 **EXTERNAL-ES-CA.pem** 的文件中，然后在 **tlsSecretName** 参数中创建名为 的secret，使其可供STF 使用

```
$ cat EXTERNAL-ES-CA.pem
-----BEGIN CERTIFICATE-----
[...]
-----END CERTIFICATE-----

$ oc create secret generic elasticsearch-es-cert --from-file=ca.crt=EXTERNAL-ES-CA.pem
```

clouds 参数

配置 **clouds** 参数，以定义哪些智能网关对象部署并提供监控云环境的接口，以连接到STF 实例。如果支持后端可用，则会为默认云配置创建指标和事件智能网关。默认情况下，Service Telemetry Operator 为 **cloud1** 创建智能网关。

您可以创建一个云对象列表来控制为定义的云创建哪个智能网关。每个云都由数据类型和收集器组成。数据类型是 **指标** 或 **事件**。每种数据类型由一个收集器列表、消息总线订阅地址和一个参数组成，以启用调试。可用的指标收集器有 **collectd**、**ceilometer** 和 **sensubility**。适用于事件的可用收集器是 **collectd** 和 **ceilometer**。确保每个收集器的订阅地址对于每个云、数据类型和收集器组合都是唯一的。

默认 **cloud1** 配置由以下 **ServiceTelemetry** 对象表示，它为特定云实例提供 collectd、Ceilometer 和 Sensubility 数据收集器的指标和数据存储：

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

clouds 参数的每个项目代表一个云实例。云实例包含三个顶级参数：**名称**、**指标** **和事件**。**metrics** 和 **events** 参数代表该数据类型存储的对应后端。**collectors** 参数指定由两个所需参数(**collectorType** 和 **subscriptionAddress**)组成的对象列表，它们代表智能网关的实例。**collectorType** 参数指定 collectd、

Ceilometer 或 Sensubility 收集的数据。**subscriptionAddress** 参数提供智能网关订阅的 AMQ Interconnect 地址。

您可以使用 **collectors** 参数中的一个可选布尔值参数 **debugEnabled** 在运行的 Smart Gateway pod 中启用额外的控制台调试功能。

其他资源

- 有关删除默认智能网关的详情，请参考 第 4.3.3 节 "删除默认智能网关"。
- 有关如何配置多个云的详情，请参考 第 4.3 节 "配置多个云"。

警报参数
设置 **警报** 参数，以创建 Alertmanager 实例和存储后端。默认情况下启用 **警报**。更多信息请参阅 第 6.3 节 "Service Telemetry Framework 中的警报"。

graphing 参数
设置 **graphing** 参数来创建 Grafana 实例。默认情况下禁用 **图形**。更多信息请参阅 第 6.1 节 "Service Telemetry Framework 中的仪表板"。

highAvailability 参数

警告

STF 高可用性(HA)模式已弃用，在生产环境中不被支持。Red Hat OpenShift Container Platform 是一个高可用性平台，如果启用了 HA 模式，您可以在 STF 中造成问题和复杂调试。

设置 **highAvailability** 参数，以实例化 STF 组件的多个副本，以减少失败或被重新调度的组件恢复时间。默认情况下禁用 **highAvailability**。更多信息请参阅 第 6.6 节 "高可用性"。

transports 参数
设置 **transports** 参数，为 STF 部署启用消息总线。目前唯一支持的传输是 AMQ Interconnect。默认情况下启用 **qdr** 传输。

3.3. 访问 STF 组件的用户界面

在 Red Hat OpenShift Container Platform 中，应用程序通过路由公开给外部网络。有关路由的更多信息，请参阅配置入口集群流量。

在 Service Telemetry Framework (STF) 中，为具有基于 Web 的界面的每个服务公开 HTTPS 路由，并由 Red Hat OpenShift Container Platform 基于角色的访问控制(RBAC)进行保护。

您需要以下权限来访问对应的组件 UI：

```
{"namespace":"service-telemetry", "resource":"grafana", "group":"grafana.integreatly.org",
"verb":"get"}
{"namespace":"service-telemetry", "resource":"prometheus", "group":"monitoring.rhobs", "verb":"get"}
{"namespace":"service-telemetry", "resource":"alertmanager", "group":"monitoring.rhobs",
"verb":"get"}
```

有关 RBAC 的更多信息，请参阅使用 RBAC 定义和应用权限。

流程

1. 登录到 Red Hat OpenShift Container Platform。
2. 进入 **service-telemetry** 命名空间：

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

3. 列出 **service-telemetry** 项目中的可用 Web UI 路由：

```
$ 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. 在网页浏览器中，进入到 https://<route_address& gt; 以访问相应服务的 Web 界面。

3.4. 配置备用可观察性策略

要跳过存储、视觉化和警报后端的部署，请将 **observabilityStrategy: none** 添加到 ServiceTelemetry spec 中。在这个模式中，您仅部署 AMQ Interconnect 路由器和智能网关，您必须配置与 Prometheus 兼容的外部系统，以从 STF 智能网关和外部 Elasticsearch 收集指标，以接收转发的事件。

流程

1. 在 **spec** 参数中，使用属性 **observabilityStrategy: none** 创建一个 **ServiceTelemetry** 对象。清单显示会导致 STF 的默认部署，该部署适合从具有所有指标收集器类型的单个云接收遥测。

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

2. 删除由社区操作器管理的剩余对象

```
$ for o in alertmanagers.monitoring.rhobs/default prometheuses.monitoring.rhobs/default elasticsearch/elasticsearch grafana/default-grafana; do oc delete $o; done
```

3. 要验证所有工作负载是否都正常运行，请查看 pod 和每个 pod 的状态：

```
$ oc get pods
NAME                                                READY   STATUS    RESTARTS   AGE
default-cloud1-ceil-event-smartgateway-6f8547df6c-p2db5   3/3     Running   0          132m
default-cloud1-ceil-meter-smartgateway-59c845d65b-gzhcs   3/3     Running   0          132m
default-cloud1-coll-event-smartgateway-bf859f8d77-tzb66   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
```

其他资源

- 有关配置附加云或更改支持的收集器集合的更多信息，请参阅 第 4.3.2 节 "部署智能网关"。
- 要将现有 STF 部署迁移到 **use_redhat**，请参阅红帽知识库文章将服务 Telemetry Framework 迁移到完全支持的操作器。

第 4 章 为 SERVICE TELEMETRY FRAMEWORK 配置 RED HAT OPENSTACK PLATFORM DIRECTOR

要收集指标、事件或两者，并将它们发送到 Service Telemetry Framework (STF) 存储域，您必须配置 Red Hat OpenStack Platform (RHOSP) overcloud 以启用数据收集和传输。

STF 可以同时支持单云和多个云。为单个云安装设置 RHOSP 和 STF 中的默认配置。

- 有关使用默认配置的单个 RHOSP overcloud 部署，请参阅 第 4.1 节 “使用 director 为 Service Telemetry Framework 部署 Red Hat OpenStack Platform overcloud”。
- 要为多个云规划您的 RHOSP 安装和配置 STF，请参阅 第 4.3 节 “配置多个云”。
- 作为 RHOSP overcloud 部署的一部分，您可能需要在环境中配置额外的功能：
  - 要禁用数据收集器服务，请参阅 第 4.2 节 “禁用用于 Service Telemetry Framework 的 Red Hat OpenStack Platform 服务”。

4.1. 使用 DIRECTOR 为 SERVICE TELEMETRY FRAMEWORK 部署 RED HAT OPENSTACK PLATFORM OVERCLOUD

作为使用 director 的 Red Hat OpenStack Platform (RHOSP) overcloud 部署的一部分，您必须配置数据收集器和数据传输到 Service Telemetry Framework (STF)。

流程

1. 第 4.1.1 节 “从 Service Telemetry Framework 获取用于 overcloud 配置的 CA 证书”
2. 检索 AMQ Interconnect 密码
3. 检索 AMQ Interconnect 路由地址
4. 为 STF 创建基本配置
5. 为 overcloud 配置 STF 连接
6. 部署 overcloud
7. 验证客户端安装

其他资源

- 有关使用 director 部署 OpenStack 云的更多信息，请参阅使用 director 安装和管理 Red Hat OpenStack Platform。
- 要通过 AMQ Interconnect 收集数据，请查看 amqp1 插件。

4.1.1. 从 Service Telemetry Framework 获取用于 overcloud 配置的 CA 证书

要将 Red Hat OpenStack Platform (RHOSP) overcloud 连接到 Service Telemetry Framework (STF)，检索在 STF 中运行的 AMQ Interconnect 的 CA 证书，并在 RHOSP 配置中使用证书。

流程

1. 查看 STF 中的可用证书列表：

```
$ oc get secrets
```

2. 检索并记录 **default-interconnect-selfsigned** Secret 的内容：

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

4.1.2. 检索 AMQ Interconnect 密码

当您为 Service Telemetry Framework (STF)配置 Red Hat OpenStack Platform (RHOSP) overcloud 时，您必须在 STF 连接文件中提供 AMQ Interconnect 密码。

您可以通过将 ServiceTelemetry spec 的 **transports.qdr.auth** 参数的值设置为 **none**，在 AMQ Interconnect 连接中禁用基本身份验证。在 STF 的 1.5.3 之前的版本中没有 **transports.qdr.auth** 参数，因此默认行为是禁用基本身份验证。在 STF 1.5.3 或更高版本的新安装中，**transports.qdr.auth** 的默认值是 **basic**，但如果您升级到 STF 1.5.3，则 **transports.qdr.auth** 的默认值为 **none**。

流程

1. 登录到托管 STF 的 Red Hat OpenShift Container Platform 环境。
2. 进入 **service-telemetry** 项目：

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

3. 检索 AMQ Interconnect 密码：

```
$ oc get secret default-interconnect-users -o json | jq -r .data.guest | base64 -d
```

4.1.3. 检索 AMQ Interconnect 路由地址

当您为 Service Telemetry Framework (STF)配置 Red Hat OpenStack Platform (RHOSP) overcloud 时，您必须在 STF 连接文件中提供 AMQ Interconnect 路由地址。

流程

1. 登录到托管 STF 的 Red Hat OpenShift Container Platform 环境。
2. 进入 **service-telemetry** 项目：

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

3. 检索 AMQ Interconnect 路由地址：

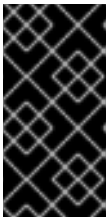
```
$ oc get routes -ogo-template='{{ range .items }}{{printf "%s\n" .spec.host }}{{ end }}' | grep "\-5671"
default-interconnect-5671-service-telemetry.apps.infra.watch
```

4.1.4. 为 STF 创建基本配置

要配置基本参数，以便为 Service Telemetry Framework (STF)提供兼容数据收集和传输，您必须创建一个定义默认数据收集值的文件。

流程

1. *以 **stack** 用户身份登录 undercloud 主机。*
2. *在 **/home/stack** 目录中创建一个名为 **enable-stf.yaml** 的配置文件。*



重要

*将 **PipelinePublishers** 设置为空列表会导致没有指标数据传递给 RHOSP 遥测组件，如 Gnocchi 或 Panko。如果您需要将数据发送到其他管道，Ceilometer 轮询间隔为 **30** 秒，您在 **ExtraConfig** 中指定，可能会认为 RHOSP 遥测组件。您必须将间隔增加到较大的值，如 **300**，这会减少 STF 中的遥测解析。*

enable-stf.yaml

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

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

  # enable Ceilometer metrics
  CeilometerQdrPublishMetrics: true

  # enable collection of API status
  CollectdEnableSensubility: true
  CollectdSensubilityTransport: amqp1

  # enable collection of containerized service metrics
  CollectdEnableLibpodstats: true

  # set collectd overrides for higher telemetry resolution and extra plugins
  # to load
  CollectdConnectionType: amqp1
  CollectdAmqpInterval: 30
  CollectdDefaultPollingInterval: 30
  # to collect information about the virtual memory subsystem of the kernel
  # CollectdExtraPlugins:
  # - vmem

  # set standard prefixes for where metrics 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
    - memory.usage

    # to avoid filling the memory buffers if disconnected from the message bus
    # note: this may need an adjustment if there are many metrics to be sent.
    collectd::plugin::amqp1::send_queue_limit: 5000

    # to receive extra information about virtual memory, you must enable vmem plugin in
CollectdExtraPlugins
    # collectd::plugin::vmem::verbose: true

    # provide name and uuid in addition to hostname for better correlation
    # to ceilometer data
    collectd::plugin::virt::hostname_format: "name uuid hostname"

    # to capture all extra_stats metrics, comment out below config
    collectd::plugin::virt::extra_stats: cpu_util vcpu disk

    # provide the human-friendly name of the virtual instance
    collectd::plugin::virt::plugin_instance_format: metadata

    # 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

    # report root filesystem storage metrics
    collectd::plugin::df::ignoreselected: false
```

4.1.5. 为 overcloud 配置 STF 连接

要配置 Service Telemetry Framework (STF)连接，您必须创建一个文件，其中包含 overcloud 到 STF 部署的 AMQ Interconnect 的连接配置。启用 STF 中指标和存储的指标集合，并部署 overcloud。默认配置是用于具有默认消息总线主题的单个云实例。有关多个云部署的配置，请参阅 第 4.3 节 “配置多个云”。

先决条件

- 从由 STF 部署的 AMQ Interconnect 检索 CA 证书。更多信息请参阅 第 4.1.1 节 “从 Service Telemetry Framework 获取用于 overcloud 配置的 CA 证书”。
- 检索 AMQ Interconnect 密码。如需更多信息，请参阅 第 4.1.2 节 “检索 AMQ Interconnect 密码”。
- 检索 AMQ Interconnect 路由地址。更多信息请参阅 第 4.1.3 节 “检索 AMQ Interconnect 路由地址”。

流程

1. 以 **stack** 用户身份登录 undercloud 主机。
2. 在 **/home/stack** 目录中创建一个名为 **stf-connectors.yaml** 的配置文件。

3. 在 **stf-connectors.yaml** 文件中，配置 **MetricsQdrConnectors** 地址，以将 overcloud 上的 AMQ Interconnect 连接到 STF 部署。您可以在此文件中配置 Sensubility、Ceilometer 和 collectd 的主题地址，以匹配 STF 中的默认值。有关自定义主题和云配置的详情，请参考 第 4.3 节 "配置多个云"。

stf-connectors.yaml

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

parameter_defaults:
    ExtraConfig:
        qdr::router_id: "%{::hostname}.cloud1"

    MetricsQdrConnectors:
        - host: default-interconnect-5671-service-telemetry.apps.infra.watch
          port: 443
          role: edge
          verifyHostname: false
          sslProfile: sslProfile
          saslUsername: guest@default-interconnect
          saslPassword: <password_from_stf>

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

    CeilometerQdrMetricsConfig:
        driver: amqp
        topic: cloud1-metering

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

    CollectdSensubilityResultsChannel: sensubility/cloud1-telemetry
```

- **qdr::router_id** 配置是覆盖使用主机的完全限定域名(FQDN)的默认值。在某些情况下，FQDN 可能会导致路由器 ID 长度大于 61 个字符，这会导致 QDR 连接失败。对于带有较短的 FQDN 值的部署，这不是必要条件。
- **resource_registry** 配置直接加载 collectd 服务，因为您没有为多个云部署包含 **collectd-write-qdr.yaml** 环境文件。
- 将 **MetricsQdrConnectors 的主机** 子参数替换为您在 第 4.1.3 节 "检索 AMQ Interconnect 路由地址" 中检索的值。
- 将 **MetricsQdrConnector s 的 saslPassword 子参数中的 <password_from_stf** > 部分替换为您在 第 4.1.2 节 "检索 AMQ Interconnect 密码" 中检索的值。

- 将 **caCertFileContent** 参数替换为 第 4.1.1 节 "从 Service Telemetry Framework 获取用于 overcloud 配置的 CA 证书" 中检索的内容。
- 设置 **CeilometerQdrMetricsConfig.topic** 的 **topic** 值，以定义 Ceilometer 指标的主题。该值是云的唯一标识符，如 **cloud1-metering**。
- 设置 **CollectdAmqpInstances** 子参数，以定义 collectd 指标的主题。部分名称是云的唯一标识符，如 **cloud1-telemetry**。
- 设置 **CollectdSensubilityResultsChannel**，以定义 collectd-sensubility 事件的主题。该值是云的唯一主题标识符，如 **sensubility/cloud1-telemetry**。

注意

当您为 collectd 和 Ceilometer 定义主题时，您提供的值将转换为智能网关客户端用于侦听消息的完整主题。

Ceilometer 主题值被转换为主题地址 **anycast/ceilometer/<TOPIC>.sample**，collectd 主题值被转换为主题地址 **collectd/<TOPIC>**。sensubility 的值是完整主题路径，且没有从主题值转换为主题地址。

有关 **ServiceTelemetry** 对象中的云配置示例，请参阅 "clouds 参数" 一节。

4.1.6. 部署 overcloud

使用所需的环境文件部署或更新 overcloud，以便收集数据并传输到 Service Telemetry Framework (STF)。

流程

1. 以 **stack** 用户身份登录 undercloud 主机。
2. 查找 **stackrc** undercloud 凭证文件：

```
$ source ~/stackrc
```

3. 使用其他环境文件将数据收集和 AMQ Interconnect 环境文件添加到堆栈中，并部署 overcloud：

```
(undercloud)$ openstack overcloud deploy --templates \
 -e [your environment files] \
 -e /usr/share/openstack-tripleo-heat-templates/environments/metrics/ceilometer-write-qdr.yaml \
 -e /usr/share/openstack-tripleo-heat-templates/environments/metrics/qdr-edge-only.yaml \
 -e /home/stack/enable-stf.yaml \
 -e /home/stack/stf-connectors.yaml
```

- 包含 **ceilometer-write-qdr.yaml** 文件，以确保将 Ceilometer 遥测发送到 STF。
- 包含 **qdr-edge-only.yaml** 文件，以确保消息总线已启用并连接到 STF 消息总线路由器。
- 包含 **enable-stf.yaml** 环境文件，以确保正确配置了默认值。
- 包含 **stf-connectors.yaml** 环境文件以定义与 STF 的连接。

4.1.7. 验证客户端安装

要验证来自 Service Telemetry Framework (STF) 存储域的数据收集，请查询数据源以获取交付数据。要验证 Red Hat OpenStack Platform (RHOSP) 部署中的单个节点，请使用 SSH 连接到控制台。

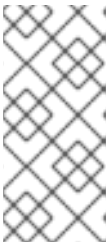
提示

只有在 RHOSP 有活跃工作负载时，一些遥测数据才可用。

流程

1. *登录 overcloud 节点，如 controller-0。*
2. *确保 **metrics_qdr** 和集合代理容器在节点上运行：*

```
$ sudo podman container inspect --format '{{.State.Status}}' metrics_qdr collectd
ceilometer_agent_notification ceilometer_agent_central
running
running
running
running
```



注意

在计算节点上使用此命令：

```
$ sudo podman container inspect --format '{{.State.Status}}' metrics_qdr
collectd ceilometer_agent_compute
```

3. *返回运行 AMQ Interconnect 的内部网络地址，例如 **172.17.1.44** 侦听端口 **5666** ：*

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

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

4. *返回到本地 AMQ Interconnect 的连接列表：*

```
$ sudo podman 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
```

有四个连接：

- 到 STF 的出站连接
- 来自 ceilometer 的入站连接
- 来自 collectd 的入站连接
- 来自 **qdstat** 客户端的入站连接
  出站 STF 连接提供给 **MetricsQdrConnectors** 主机参数，是 STF 存储域的路由。其他主机是连接到这个 AMQ Interconnect 的客户端连接的内部网络地址。

5. 要确保传递消息，列出链接，并查看 **deliv** 列中用于传递消息的 **_edge** 地址：

```
$ sudo podman 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
  endpoint  in   1       7                                      250  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
  endpoint  in   1       9                                      250  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.lSY6Mcicol4J2Kp      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.9Ok2resl9tmt+CT      250  0    0    0      0
0       0      0    0    0      0    0     0
```

6. 要列出 RHOSP 节点到 STF 的地址，连接到 Red Hat OpenShift Container Platform 以检索 AMQ Interconnect pod 名称并列出连接。列出可用的 AMQ Interconnect pod:

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

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

7. 连接到 pod 并列出已知的连接。在本例中，有来自 RHOSP 节点的三个 **边缘** 连接，其连接 **id** 为 22, 23, 和 24。

```
$ oc exec -it deploy/default-interconnect -- 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-xmxwn]
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. 要查看网络发送的消息数量，请使用每个地址与 **oc exec** 命令：

```
$ oc exec -it deploy/default-interconnect -- 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. 禁用用于 SERVICE TELEMETRY FRAMEWORK 的 RED HAT OPENSTACK PLATFORM 服务

禁用部署 Red Hat OpenStack Platform (RHOSP)时使用的服务，并将其连接到 Service Telemetry Framework (STF)。没有删除日志或生成的配置文件，作为服务禁用的一部分。

流程

1. 以 **stack** 用户身份登录 undercloud 主机。
2. 查找 **stackrc** undercloud 凭证文件：

```
$ source ~/stackrc
```

3. 创建 **disable-stf.yaml** 环境文件：

```
$ cat > ~/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. 从 RHOSP director 部署中删除以下文件：
   - **ceilometer-write-qdr.yaml**
   - **qdr-edge-only.yaml**
   - **enable-stf.yaml**
   - **stf-connectors.yaml**
5. 更新 RHOSP overcloud。确保在环境文件列表早期使用 **disable-stf.yaml** 文件。通过在列表早期添加 **disable-stf.yaml**，其他环境文件可能会覆盖禁用该服务的配置：

```
(undercloud)$ openstack overcloud deploy --templates \
-e /home/stack/disable-stf.yaml \
-e [your environment files]
```

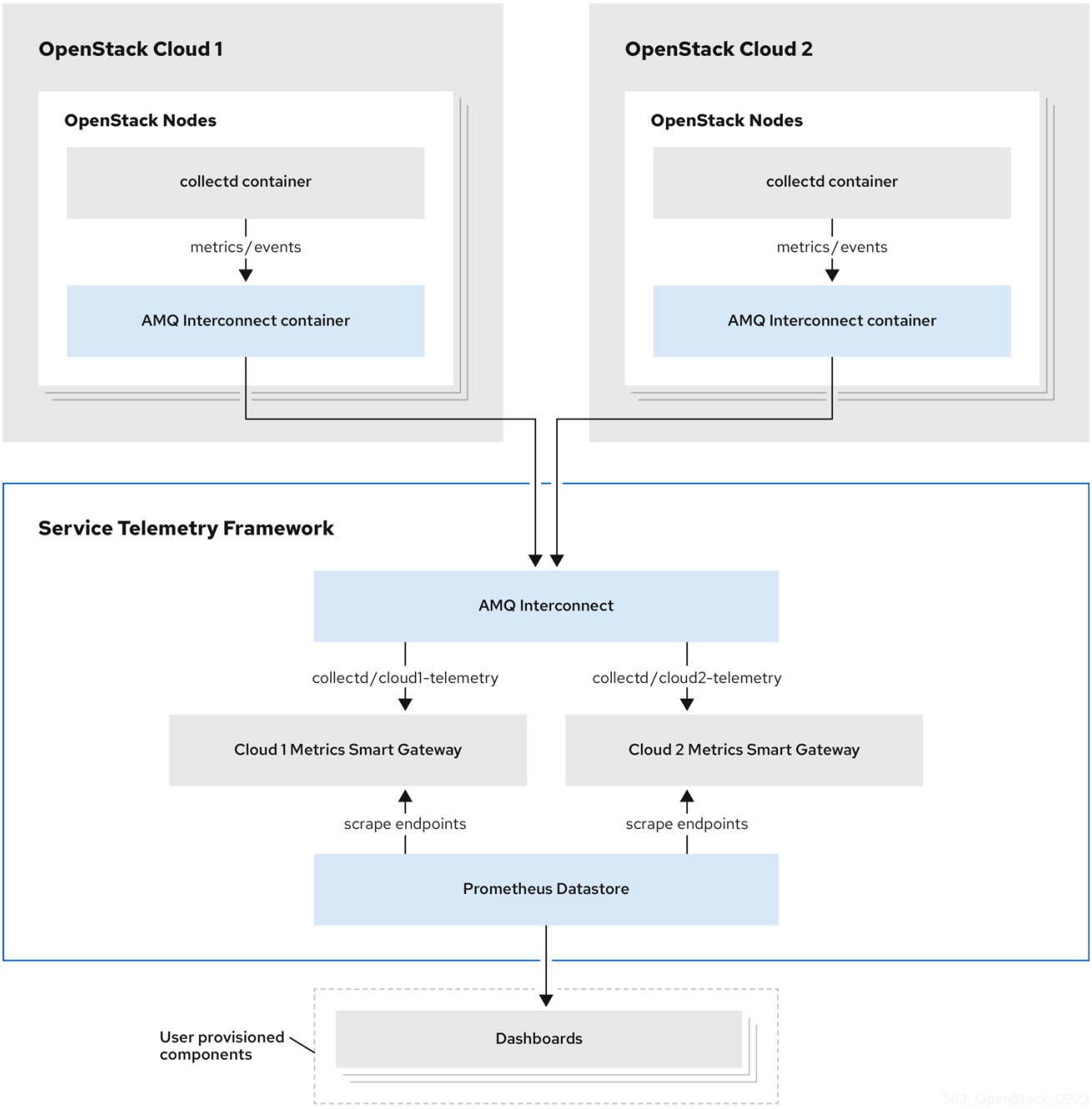
4.3. 配置多个云

您可以配置多个 Red Hat OpenStack Platform (RHOSP) 云以单一 Service Telemetry Framework (STF) 实例为目标。当您配置多个云时，每个云都必须在其自己唯一的消息总线主题上发送指标和事件。在 STF 部署中，智能网关实例侦听这些主题，以将信息保存到通用数据存储中。在数据存储域中由智能网关存储的数据通过利用每个智能网关创建的元数据进行过滤。

警告

确保您使用唯一的云域配置部署每个云。有关为您的云部署配置域的更多信息，请参阅 第 4.3.4 节 “设置唯一的云域”。

图 4.1. 两个 RHOSP 云连接到 STF



要为多个云场景配置 RHOSP overcloud，请完成以下任务：

1. *规划您要为每个云使用的 AMQP 地址前缀。更多信息请参阅 第 4.3.1 节 "规划 AMQP 地址前缀"。*
2. *为每个云部署指标和事件使用者智能网关，以侦听对应的地址前缀。更多信息请参阅 第 4.3.2 节 "部署智能网关"。*
3. *使用唯一的域名配置每个云。更多信息请参阅 第 4.3.4 节 "设置唯一的云域"。*
4. *为 STF 创建基础配置。更多信息请参阅 第 4.1.4 节 "为 STF 创建基本配置"。*
5. *配置每个云，使其指标和事件发送到正确地址上的 STF。更多信息请参阅 第 4.3.5 节 "为多个云创建 Red Hat OpenStack Platform 环境文件"。*

4.3.1. 规划 AMQP 地址前缀

默认情况下，Red Hat OpenStack Platform (RHOSP) 节点通过两个数据收集器检索数据：collectd 和 Ceilometer。collectd-sensubility 插件需要唯一的地址。这些组件将遥测数据或通知发送到对应的 AMQP 地址，如 **collectd/telemetry**。STF 智能网关侦听这些用于数据的 AMQP 地址。要支持多个云并识别哪个云生成监控数据，请配置每个云以将数据发送到唯一的地址。将云标识符前缀添加到地址的第二部分。以下列表显示了一些地址和标识符示例：

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

4.3.2. 部署智能网关

您必须为每个云的每个数据收集类型部署智能网关；一个用于 collectd 指标，一个用于 collectd 指标，一个用于 Ceilometer 指标，一个用于 Ceilometer 事件，另一个用于 collectd-sensubility 指标。将每个智能网关配置为侦听您为对应云定义的 AMQP 地址。要定义智能网关，请在 **ServiceTelemetry** 清单中配置 **clouds** 参数。

当您首次部署 STF 时，会创建智能网关清单，以定义单个云的初始智能网关。当您为多个云支持部署智能网关时，您可以为处理指标和每个云的事件数据的每个数据收集类型部署多个智能网关。初始智能网关在 **cloud1** 中定义，包括以下订阅地址：

collector	type	默认订阅地址
collectd	metrics	collectd/telemetry
collectd	Events	collectd/notify
collectd-sensubility	metrics	sensubility/telemetry
ilo	metrics	anycast/ceilometer/metering.sample
ilo	Events	anycast/ceilometer/event.sample

前提条件

- 您已确定了云命名方案。有关确定命名方案的详情请参考 第 4.3.1 节 "规划 AMQP 地址前缀"。
- 您已创建了 clouds 对象列表。有关为 **clouds** 参数创建内容的更多信息，请参阅 "clouds 参数" 一节。

流程

1. 登录到 Red Hat OpenShift Container Platform。
2. 进入 **service-telemetry** 命名空间：

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

3. 编辑默认 ServiceTelemetry 对象，并使用您的配置添加 **clouds** 参数：

警告

较长的节点名称可能会超过最大 pod 名称 63 个字符。确保 **ServiceTelemetry** 名称 **default** 和 **clouds.name** 的组合没有超过 19 个字符。云名称不能包含任何特殊字符，如 **-**。将云名称限制为字母数字(a-z、0-9)。

主题地址没有字符限制，可以与 **clouds.name** 值不同。

```
$ 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: sensubility
        subscriptionAddress: sensubility/cloud1-telemetry
      - collectorType: ceilometer
        subscriptionAddress: anycast/ceilometer/cloud1-metering.sample
```

```
  - name: cloud2
    events:
      ...
```

4. 保存 ServiceTelemetry 对象。

5. 验证每个智能网关是否正在运行。这可能需要几分钟，具体取决于智能网关的数量：

```
$ 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-rl9bd   2/2     Running   0          13h
default-cloud1-coll-meter-smartgateway-7c6fc495c4-jn728   2/2     Running   0          13h
default-cloud1-sens-meter-smartgateway-8h4tc445a2-mm683   2/2     Running   0          13h
```

4.3.3. 删除默认智能网关

为多个云配置 Service Telemetry Framework (STF)后，如果不再使用它们，您可以删除默认的智能网关。Service Telemetry Operator 可以删除创建的 **SmartGateway** 对象，但不再列在 ServiceTelemetry **云** 对象列表中。要启用删除由 **clouds** 参数定义的 SmartGateway 对象，您必须在 **ServiceTelemetry** 清单中将 **cloudsRemoveOnMissing** 参数设置为 **true**。

提示

如果您不想部署任何智能网关，请使用 **clouds: []** 参数定义空云列表。

警告

cloudsRemoveOnMissing 参数默认为禁用。如果启用了 **cloudsRemoveOnMissing** 参数，您可以在当前命名空间中删除所有手动创建的 **SmartGateway** 对象，而无需恢复。

流程

1. 使用您要管理 Service Telemetry Operator 的云对象列表定义您的 **clouds** 参数。更多信息请参阅 "clouds 参数"一节。

2. 编辑 ServiceTelemetry 对象并添加 **cloudsRemoveOnMissing** 参数：

```
apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
metadata:
  ...
spec:
  ...
  cloudsRemoveOnMissing: true
  clouds:
    ...
```

3. 保存修改。
4. 验证 Operator 是否删除了 Smart Gateways。Operator 协调更改时可能需要几分钟时间：

```
$ oc get smartgateways
```

4.3.4. 设置唯一的云域

为确保从不同 Red Hat OpenStack Platform (RHOSP) 云到 Service Telemetry Framework (STF) 的遥测可以唯一标识且不冲突，请配置 **CloudDomain** 参数。

警告

确保您不会在现有部署中更改主机或域名。只有新的云部署支持主机和域名配置。

流程

1. 创建新的环境文件，如 hostname **.yaml**。
2. 在环境文件中设置 **CloudDomain** 参数，如下例所示：

hostnames.yaml

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

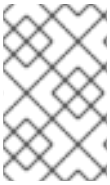
3. 将新的环境文件添加到您的部署中。

其他资源

- 第 4.3.5 节 "为多个云创建 Red Hat OpenStack Platform 环境文件"
- Overcloud 参数指南中的 核心 Overcloud 参数

4.3.5. 为多个云创建 Red Hat OpenStack Platform 环境文件

要根据原始云标记流量，您必须使用特定于云的实例名称创建配置。创建一个 **stf-connectors.yaml** 文件，并调整 **CeilometerQdrMetricsConfig** 和 **CollectdAmqpInstances** 的值，以匹配 AMQP 地址前缀方案。



注意

如果启用了容器健康和 API 状态监控，还必须修改 **CollectdSensubilityResultsChannel** 参数。更多信息请参阅 第 6.9 节 "Red Hat OpenStack Platform API 状态和容器化服务健康状况"。

先决条件

- 您已从 STF 部署的 AMQ Interconnect 中检索 CA 证书。更多信息请参阅 第 4.1.1 节 "从 Service Telemetry Framework 获取用于 overcloud 配置的 CA 证书"。
- 您已创建了 clouds 对象列表。有关为 clouds 参数创建内容的更多信息，请参阅 clouds 配置参数。
- 您已检索了 AMQ Interconnect 路由地址。更多信息请参阅 第 4.1.3 节 "检索 AMQ Interconnect 路由地址"。
- 您已为 STF 创建了基础配置。更多信息请参阅 第 4.1.4 节 "为 STF 创建基本配置"。
- 您已创建了唯一的域名环境文件。更多信息请参阅 第 4.3.4 节 "设置唯一的云域"。

流程

1. 以 **stack** 用户身份登录 undercloud 主机。
2. 在 **/home/stack** 目录中创建一个名为 **stf-connectors.yaml** 的配置文件。
3. 在 **stf-connectors.yaml** 文件中，配置 **MetricsQdrConnectors** 地址，以连接到 overcloud 部署上的 AMQ Interconnect。配置 **CeilometerQdrMetricsConfig**、**CollectdAmqpInstances** 和 **CollectdSensubilityResultsChannel** 主题值，以匹配您想要进行此云部署的 AMQP 地址。

   stf-connectors.yaml

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

parameter_defaults:
    ExtraConfig:
      qdr::router_id: %{::hostname}.cloud1

    MetricsQdrConnectors:
      - host: 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-----

    CeilometerQdrMetricsConfig:
      driver: amqp
      topic: cloud1-metering

    CollectdAmqpInstances:
      cloud1-telemetry:
        format: JSON
```

```
        presettle: false

    CollectdSensubilityResultsChannel: sensubility/cloud1-telemetry
```

- **qdr::router_id** 配置是覆盖使用主机的完全限定域名(FQDN)的默认值。在某些情况下，FQDN 可能会导致路由器 ID 长度大于 61 个字符，这会导致 QDR 连接失败。对于带有较短的 FQDN 值的部署，这不是必要条件。
- **resource_registry** 配置直接加载 collectd 服务，因为您没有为多个云部署包含 **collectd-write-qdr.yaml** 环境文件。
- 将 **host** 参数替换为您在 第 4.1.3 节 "检索 AMQ Interconnect 路由地址" 中检索的值。
- 将 **caCertFileContent** 参数替换为 第 4.1.1 节 "从 Service Telemetry Framework 获取用于 overcloud 配置的 CA 证书" 中检索的内容。
- 将 **MetricsQdrConnectors 的主机** 子参数替换为您在 第 4.1.3 节 "检索 AMQ Interconnect 路由地址" 中检索的值。
- 设置 **CeilometerQdrMetricsConfig.topic** 的 **topic** 值，以定义 Ceilometer 指标的主题。该值是云的唯一标识符，如 **cloud1-metering**。
- 设置 **CollectdAmqpInstances** 子参数，以定义 collectd 指标的主题。部分名称是云的唯一标识符，如 **cloud1-telemetry**。
- 设置 **CollectdSensubilityResultsChannel**，以定义 collectd-sensubility 事件的主题。该值是云的唯一主题标识符，如 **sensubility/cloud1-telemetry**。

注意

当您为 collectd 和 Ceilometer 定义主题时，您提供的值将转换为智能网关客户端用于侦听消息的完整主题。

Ceilometer 主题值被转换为主题地址 **anycast/ceilometer/<TOPIC>.sample**，collectd 主题值被转换为主题地址 **collectd/<TOPIC>**。sensubility 的值是完整主题路径，且没有从主题值转换为主题地址。

有关 **ServiceTelemetry** 对象中的云配置示例，请参阅 "clouds 参数" 一节。

4. 确保 **stf-connectors.yaml** 文件中的命名约定与 Smart Gateway 配置中的 **spec.bridge.amqpUrl** 字段一致。例如，将 **CeilometerQdrMetricsConfig.topic** 字段配置为 **cloud1-metering** 的值。
5. 以 **stack** 用户身份登录 undercloud 主机。
6. 查找 **stackrc** undercloud 凭证文件：

```
$ source stackrc
```

7. 在 **openstack overcloud deployment** 命令中包含 **stf-connectors.yaml** 文件以及唯一域名环境文件 **hostname.yaml**，以及其他与您环境相关的环境文件：

警告

如果您使用带有自定义 **CollectdAmqpInstances** 参数的 **collectd-write-qdr.yaml** 文件，数据会发布到自定义和默认主题。在多个云环境中，**stf-connectors.yaml** 文件中的 **resource_registry** 参数的配置会加载 collectd 服务。

```
(undercloud)$ openstack overcloud deploy --templates \
-e [your environment files] \
-e /usr/share/openstack-tripleo-heat-templates/environments/metrics/ceilometer-write-qdr.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/metrics/qdr-edge-only.yaml \
-e /home/stack/hostnames.yaml \
-e /home/stack/enable-stf.yaml \
-e /home/stack/stf-connectors.yaml
```

8. 部署 Red Hat OpenStack Platform overcloud。

其他资源

- 有关如何验证部署的详情，请参考 第 4.1.7 节 "验证客户端安装"。

4.3.6. 从多个云查询指标数据

Prometheus 中存储的数据根据从中提取的智能网关具有 **服务** 标签。您可以使用此标签从特定云查询数据。

要查询特定云的数据，请使用与 **service** 标签相关的匹配的 Prometheus promql 查询；例如：**collectd_uptime{service="default-cloud1-coll-meter"}**。

第 5 章 为 SERVICE TELEMETRY FRAMEWORK 配置 RED HAT OPENSTACK PLATFORM DIRECTOR OPERATOR

要收集指标、事件或两者，并将它们发送到 Service Telemetry Framework (STF) 存储域，您必须配置 Red Hat OpenStack Platform (RHOSP) overcloud 以启用数据收集和传输。

STF 可以同时支持单云和多个云。为单个云安装设置 RHOSP 和 STF 中的默认配置。

- 有关使用带有默认配置的 director Operator 的单一 RHOSP overcloud 部署，请参阅 第 5.1 节 "使用 director Operator 为 Service Telemetry Framework 部署 Red Hat OpenStack Platform overcloud"。

5.1. 使用 DIRECTOR OPERATOR 为 SERVICE TELEMETRY FRAMEWORK 部署 RED HAT OPENSTACK PLATFORM OVERCLOUD

使用 director Operator 部署 Red Hat OpenStack Platform (RHOSP) overcloud 部署时，您必须配置数据收集器以及 Service Telemetry Framework (STF) 的数据传输。

先决条件

- 熟悉使用 RHOSP director Operator 部署和管理 RHOSP。

流程

1. 第 4.1.1 节 "从 Service Telemetry Framework 获取用于 overcloud 配置的 CA 证书"
2. 检索 AMQ Interconnect 路由地址
3. 为 STF 创建 director Operator 的基本配置
4. 为 overcloud 配置 STF 连接
5. 为 director operator 部署 overcloud

其他资源

- 有关使用 director Operator 部署 OpenStack 云的更多信息，请参阅 https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/17.1/html/deploying_an_overcloud_in_a_red_hat_openshift_contain
- 要通过 AMQ Interconnect 收集数据，请查看 amqp1 插件。

5.1.1. 从 Service Telemetry Framework 获取用于 overcloud 配置的 CA 证书

要将 Red Hat OpenStack Platform (RHOSP) overcloud 连接到 Service Telemetry Framework (STF)，检索在 STF 中运行的 AMQ Interconnect 的 CA 证书，并在 RHOSP 配置中使用证书。

流程

1. 查看 STF 中的可用证书列表：

```
$ oc get secrets
```

2. 检索并记录 **default-interconnect-selfsigned** Secret 的内容：

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

5.1.2. 检索 AMQ Interconnect 路由地址

当您为 Service Telemetry Framework (STF)配置 Red Hat OpenStack Platform (RHOSP) overcloud 时，您必须在 STF 连接文件中提供 AMQ Interconnect 路由地址。

流程

1. 登录到托管 STF 的 Red Hat OpenShift Container Platform 环境。
2. 进入 **service-telemetry** 项目：

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

3. 检索 AMQ Interconnect 路由地址：

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

5.1.3. 为 STF 创建 director Operator 的基本配置

编辑 **heat-env-config-deploy** ConfigMap，将基本 Service Telemetry Framework (STF)配置添加到 overcloud 节点。

流程

1. 登录到部署了 RHOSP director Operator 的 Red Hat OpenShift Container Platform 环境，并更改到托管 RHOSP 部署的项目：

```
$ oc project openstack
```

2. 打开 **heat-env-config-deploy ConfigMap** CR 进行编辑：

```
$ oc edit heat-env-config-deploy
```

3. 将 **enable-stf.yaml** 配置添加到 **heat-env-config-deploy** ConfigMap 中，保存您的编辑并关闭该文件：

enable-stf.yaml

```
apiVersion: v1
data:
  [...]
  enable-stf.yaml: |
    parameter_defaults:
      # only send to STF, not other publishers
      PipelinePublishers: []
```

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

    # enable Ceilometer metrics and events
    CeilometerQdrPublishMetrics: true

    # enable collection of API status
    CollectdEnableSensubility: true
    CollectdSensubilityTransport: amqp1

    # enable collection of containerized service metrics
    CollectdEnableLibpodstats: true

    # set collectd overrides for higher telemetry resolution and extra plugins
    # to load
    CollectdConnectionType: amqp1
    CollectdAmqpInterval: 30
    CollectdDefaultPollingInterval: 30
    # to collect information about the virtual memory subsystem of the kernel
    # CollectdExtraPlugins:
    # - vmem

    # set standard prefixes for where metrics 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
      - memory.usage

      # to avoid filling the memory buffers if disconnected from the message bus
      # note: this may need an adjustment if there are many metrics to be sent.
      collectd::plugin::amqp1::send_queue_limit: 5000

      # to receive extra information about virtual memory, you must enable vmem plugin in
CollectdExtraPlugins
      # collectd::plugin::vmem::verbose: true

      # provide name and uuid in addition to hostname for better correlation
      # to ceilometer data
      collectd::plugin::virt::hostname_format: "name uuid hostname"

      # to capture all extra_stats metrics, comment out below config
      collectd::plugin::virt::extra_stats: cpu_util vcpu disk

      # provide the human-friendly name of the virtual instance
      collectd::plugin:ConfigMap :virt::plugin_instance_format: metadata

      # 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

    # report root filesystem storage metrics
    collectd::plugin::df::ignoreselected: false
```

其他资源

- 有关配置 **enable-stf.yaml** 环境文件的更多信息，请参阅 第 4.1.4 节 "为 STF 创建基本配置"
- 有关在 Red Hat OpenStack Platform director Operator 部署中添加 heat 模板的更多信息，请参阅使用 director Operator 添加 heat 模板和环境文件

5.1.4. 为 overcloud 为 director Operator 配置 STF 连接

编辑 **heat-env-config-deploy** ConfigMap，以创建从 Red Hat OpenStack Platform (RHOSP) 到 Service Telemetry Framework 的连接。

流程

1. 登录到部署了 RHOSP director Operator 的 Red Hat OpenShift Container Platform 环境，并更改到托管 RHOSP 部署的项目：

```
$ oc project openstack
```

2. 打开 **heat-env-config-deploy** ConfigMap 进行编辑：

```
$ oc edit configmap heat-env-config-deploy
```

3. 将 **stf-connectors.yaml** 配置添加到 **heat-env-config-deploy** ConfigMap 中，适用于您的环境，保存您的编辑并关闭该文件：

stf-connectors.yaml

```
apiVersion: v1
data:
  [...]
  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: default-interconnect-5671-service-telemetry.apps.ostest.test.metalkube.org
          port: 443
          role: edge
          verifyHostname: false
          sslProfile: sslProfile
          saslUsername: guest@default-interconnect
          saslPassword: <password_from_stf>
```

```
    MetricsQdrSSLProfiles:
      - name: sslProfile

    CeilometerQdrMetricsConfig:
      driver: amqp
      topic: cloud1-metering

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

    CollectdSensubilityResultsChannel: sensubility/cloud1-telemetry
```

- ***resource_registry*** *配置直接加载 collectd 服务，因为您没有为多个云部署包含* ***collectd-write-qdr.yaml*** *环境文件。*
- *将* ***MetricsQdrConnectors 的主机*** *子参数替换为您在 第 4.1.3 节 "检索 AMQ Interconnect 路由地址" 中检索的值。*
- *将* ***MetricsQdrConnector s 的 saslPassword 子参数中的 <password_from_stf>*** *部分替换为您在 第 4.1.2 节 "检索 AMQ Interconnect 密码" 中检索的值。*
- *将* ***caCertFileContent*** *参数替换为 第 4.1.1 节 "从 Service Telemetry Framework 获取用于 overcloud 配置的 CA 证书" 中检索的内容。*
- *设置* ***CeilometerQdrMetricsConfig.topic*** *的* ***topic*** *值，以定义 Ceilometer 指标的主题。该值是云的唯一标识符，如* ***cloud1-metering***。
- *设置* ***CollectdAmqpInstances*** *子参数，以定义 collectd 指标的主题。部分名称是云的唯一标识符，如* ***cloud1-telemetry***。
- *设置* ***CollectdSensubilityResultsChannel***，*以定义 collectd-sensubility 事件的主题。该值是云的唯一主题标识符，如* ***sensubility/cloud1-telemetry***。

其他资源

- *有关* ***stf-connectors.yaml*** *环境文件的更多信息，请参阅 第 4.1.5 节 "为 overcloud 配置 STF 连接"。*
- *有关在 RHOSP director Operator 部署中添加 heat 模板的更多信息，请参阅使用 director Operator 添加 heat 模板和环境文件*

5.1.5. 为 director Operator 部署 overcloud

使用所需的环境文件部署或更新 overcloud，以便收集数据并传输到 Service Telemetry Framework (STF)。

流程

1. *登录到部署了 RHOSP director Operator 的 Red Hat OpenShift Container Platform 环境，并更改到托管 RHOSP 部署的项目：*

```
$ oc project openstack
```

2. 打开 **OpenStackConfigGenerator** 自定义资源进行编辑：

```
$ oc edit OpenStackConfigGenerator
```

3. 添加 **metrics/ceilometer-write-qdr.yaml** 和 **metrics/qdr-edge-only.yaml** 环境文件，作为 **heatEnvs** 参数的值。保存编辑并关闭 **OpenStackConfigGenerator** 自定义资源：

注意

如果您已经使用 director Operator 部署 Red Hat OpenStack Platform 环境，您必须删除现有 **OpenStackConfigGenerator** 并创建带有完整配置的新对象，才能重新生成 **OpenStackConfigVersion**。

OpenStackConfigGenerator

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackConfigGenerator
metadata:
  name: default
  namespace: openstack
spec:
  heatEnvConfigMap: heat-env-config-deploy
  heatEnvs:
  - <existing_environment_file_references>
  - metrics/ceilometer-write-qdr.yaml
  - metrics/qdr-edge-only.yaml
```

4. 如果您已经使用 director Operator 部署 Red Hat OpenStack Platform 环境，并生成新的 **OpenStackConfigVersion**，请编辑部署的 **OpenStackDeploy** 对象，并将 **spec.configVersion** 的值设置为新的 **OpenStackConfigVersion** 以更新 overcloud 部署。

其他资源

- 有关获取最新的 **OpenStackConfigVersion** 的更多信息，请参阅 *获取最新的 OpenStackConfigVersion*
- 有关使用 director Operator 应用 overcloud 配置的更多信息，请参阅使用 director Operator *应用 overcloud 配置*

第 6 章 使用 SERVICE TELEMETRY FRAMEWORK 的操作功能

您可以使用以下操作功能为 Service Telemetry Framework (STF)提供额外的功能：

- 配置仪表板
- 配置指标保留时间段
- 配置警报
- 配置 SNMP 陷阱
- 配置高可用性
- 配置备用可观察性策略
- 监控 OpenStack 服务的资源使用
- 监控容器健康和 API 状态

6.1. SERVICE TELEMETRY FRAMEWORK 中的仪表板

使用第三方应用程序 Grafana 来视觉化数据收集器为各个主机节点收集的系统级指标。

有关配置数据收集器的更多信息，请参阅 第 4.1 节 "使用 director 为 Service Telemetry Framework 部署 Red Hat OpenStack Platform overcloud"。

您可以使用仪表板来监控云：

基础架构仪表板

使用基础架构仪表板查看单个节点的指标。从控制面板的左上角选择一个节点。

Cloud view 仪表板

使用云视图仪表板查看面板，以监控服务资源使用情况、API 统计和云事件。您必须启用 API 健康监控和服务监控，以便为此仪表板提供数据。在 STF 基础配置中默认启用 API 健康监控。更多信息请参阅 第 4.1.4 节 "为 STF 创建基本配置"。

- 有关 API 健康监控的更多信息，请参阅 第 6.9 节 "Red Hat OpenStack Platform API 状态和容器化服务健康状况"。
- 有关 RHOSP 服务监控的更多信息，请参阅 第 6.8 节 "Red Hat OpenStack Platform 服务的资源使用"。

虚拟机视图仪表板

使用虚拟机视图仪表板查看面板，以监控虚拟机基础架构使用情况。从控制面板的左上角选择云和项目。如果要在此仪表板上启用事件注解，则必须启用事件存储。如需更多信息，请参阅 第 3.2 节 "在 Red Hat OpenShift Container Platform 中创建 ServiceTelemetry 对象"。

Memcached view 仪表板

使用 memcached 视图仪表板查看面板，以监控连接、可用性、系统指标和缓存性能。从控制面板的左上角选择云。

6.1.1. 配置 Grafana 以托管仪表板

Grafana 不包含在默认的 Service Telemetry Framework (STF) 部署中，因此您必须从 community-operators CatalogSource 部署 Grafana Operator。如果使用 Service Telemetry Operator 部署 Grafana，它会生成 Grafana 实例，并配置本地 STF 部署的默认数据源。

流程

1. *登录到托管 STF 的 Red Hat OpenShift Container Platform 环境。*
2. *使用 community-operators CatalogSource 订阅 Grafana Operator：*

警告

社区 Operator 是尚未被红帽审查或验证的 Operator。社区 Operator 应谨慎使用，因为其稳定性未知。红帽不支持社区 Operator。

了解更多有关红帽的第三方软件支持政策

```
$ oc apply -f - <<EOF
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  labels:
    operators.coreos.com/grafana-operator.openshift-operators: ""
  name: grafana-operator
  namespace: openshift-operators
spec:
  channel: v5
  installPlanApproval: Automatic
  name: grafana-operator
  source: community-operators
  sourceNamespace: openshift-marketplace
EOF
```

3. *验证 Operator 是否已成功启动。在命令输出中，如果 **PHASE** 列的值是 **Succeeded**，Operator 会成功启动：*

```
$ oc wait --for jsonpath="{.status.phase}"=Succeeded csv --namespace openshift-operators -l operators.coreos.com/grafana-operator.openshift-operators

clusterserviceversion.operators.coreos.com/grafana-operator.v5.6.0 condition met
```

4. *要启动 Grafana 实例，请创建或修改 **ServiceTelemetry** 对象。将 **graphing.enabled** 和 **graphing.grafana.ingressEnabled** 设置为 **true**。另外，还可将 **graphing.grafana.baseImage** 的值设置为要部署的 Grafana 工作负载容器镜像：*

```
$ oc edit stf default

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

```
spec:
  ...
  graphing:
    enabled: true
    grafana:
      ingressEnabled: true
      baseImage: 'registry.redhat.io/rhel8/grafana:9'
```

5. 验证 Grafana 实例是否已部署：

```
$ oc wait --for jsonpath="{.status.phase}"=Running pod -l app=default-grafana --
timeout=600s

pod/default-grafana-deployment-669968df64-wz5s2 condition met
```

6. 验证 Grafana 数据源是否已正确安装：

```
$ oc get grafanadatasources.grafana.integreatly.org

NAME                     NO MATCHING INSTANCES   LAST RESYNC   AGE
default-ds-stf-prometheus                        2m35s         2m56s
```

7. 验证 Grafana 路由是否存在：

```
$ oc get route default-grafana-route

NAME                  HOST/PORT                                          PATH   SERVICES
PORT   TERMINATION   WILDCARD
default-grafana-route   default-grafana-route-service-telemetry.apps.infra.watch
default-grafana-service   web    reencrypt     None
```

6.1.2. 启用仪表板

Grafana Operator 可以通过创建 **GrafanaDashboard** 对象来导入和管理仪表板。Service Telemetry Operator 可以启用一组默认仪表板，创建将仪表板加载到 Grafana 实例中的 **GrafanaDashboard** 对象。

将 **graphing.grafana.dashboards.enabled** 的值设置为 **true**，将以下仪表板加载到 Grafana 中：

- 基础架构仪表板
- Cloud view 仪表板
- 虚拟机视图仪表板
- Memcached view 仪表板

您可以使用 GrafanaDashboard **对象来创建** 并载入额外的仪表板到 Grafana 中。有关使用 Grafana Operator 管理仪表板的更多信息，请参阅 Grafana Operator 项目文档中的 Dashboards。

先决条件

- 您在 **ServiceTelemetry** 对象中启用了图形。有关图形的详情请参考 第 6.1.1 节 "配置 Grafana 以托管仪表板"。

流程

流程

1. 要启用受管仪表板，请创建或修改 **ServiceTelemetry** 对象。将 **graphing.grafana.dashboards.enabled** 设置为 **true** ：

```
$ oc edit stf default

apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
...
spec:
  ...
  graphing:
    enabled: true
    grafana:
      dashboards:
        enabled: true
```

2. 验证 Grafana 仪表板是否已创建。创建仪表板的 Service Telemetry Operator 的过程可能需要一些时间。

```
$ oc get grafanadashboards.grafana.integreatly.org

NAME                          NO MATCHING INSTANCES   LAST RESYNC   AGE
memcached-dashboard-1                                 38s           38s
rhos-cloud-dashboard-1                                39s           39s
rhos-dashboard-1                                      39s           39s
virtual-machine-dashboard-1                           37s           37s
```

3. 检索 Grafana 路由地址：

```
$ oc get route default-grafana-route -ojsonpath='{.spec.host}'

default-grafana-route-service-telemetry.apps.infra.watch
```

4. 在 Web 浏览器中，进入到 https://<grafana_route_address>。使用您在上一步中获得的值替换 <grafana_route_address>。

5. 使用 OpenShift 凭据登录。有关登录的详情请参考 第 3.3 节 “访问 STF 组件的用户界面”。

6. 要查看仪表板，请点 **Dashboards** 和 **Browse**。受管仪表板位于 service-telemetry 文件夹中。

6.1.3. 连接外部仪表板系统

可以配置第三方视觉化工具，以连接到 STF Prometheus 以进行指标检索。访问通过 OAuth 令牌控制，并且已创建了具有所需权限的 ServiceAccount。可以为此帐户生成一个新的 OAuth 令牌，供外部系统使用。

若要使用身份验证令牌，必须将第三方工具配置为提供 HTTP Bearer Token Authorization 标头，如 RFC6750 所述。有关如何配置此标头的信息，请参阅第三方工具的文档。例如，在 Grafana 文档中的配置 Prometheus - 自定义 HTTP 标头。

流程

1. 登录到 Red Hat OpenShift Container Platform。

2. 进入 **service-telemetry** 命名空间：

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

3. 为 stf-prometheus-reader 服务帐户创建新令牌 secret

```
$ oc create -f - <<EOF
apiVersion: v1
kind: Secret
metadata:
  name: my-prometheus-reader-token
  namespace: service-telemetry
  annotations:
    kubernetes.io/service-account.name: stf-prometheus-reader
type: kubernetes.io/service-account-token
EOF
```

4. 从 secret 检索令牌

```
$ TOKEN=$(oc get secret my-prometheus-reader-token -o template='{{.data.token}}' |
base64 -d)
```

5. 检索 Prometheus 主机名

```
$ PROM_HOST=$(oc get route default-prometheus-proxy -ogo-template='{{ .spec.host }}')
```

6. 测试访问令牌

```
$ curl -k -H "Authorization: Bearer ${TOKEN}" https://${PROM_HOST}/api/v1/query?
query=up

{"status":"success",[...]
```

7. 使用以上 PROM_HOST 和 TOKEN 值配置第三方工具

```
$ echo $PROM_HOST
$ echo $TOKEN
```

8. 只要 secret 存在，令牌就会保持有效。您可以通过删除 secret 来撤销令牌。

```
$ oc delete secret my-prometheus-reader-token
secret "my-prometheus-reader-token" deleted
```

附加信息

如需有关服务帐户令牌 secret 的更多信息，请参阅 OpenShift Container Platform 文档中的创建服务帐户令牌 secret。

6.2. SERVICE TELEMETRY FRAMEWORK 中的指标保留时间段

在 Service Telemetry Framework (STF) 中存储的指标的默认保留时间为 24 小时，它为警报目的提供了足够的数据。

对于长期存储，请使用专为长期数据保留而设计的系统，例如 Thanos。

其他资源

- 要为额外的指标保留时间调整 STF，请参阅 第 6.2.1 节 "在 Service Telemetry Framework 中编辑指标保留时间段"。
- 有关 Prometheus 数据存储和估算存储空间的建议，请参阅 https://prometheus.io/docs/prometheus/latest/storage/#operational-aspects
- 有关 Thanos 的更多信息，请参阅 https://thanos.io/

6.2.1. 在 Service Telemetry Framework 中编辑指标保留时间段

您可以调整 Service Telemetry Framework (STF) 以提供额外的指标保留时间。

流程

1. 登录到 Red Hat OpenShift Container Platform。
2. 进入 service-telemetry 命名空间：

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

3. 编辑 ServiceTelemetry 对象：

```
$ oc edit stf default
```

4. 将 **retention: 7d** 添加到 backends.metrics.prometheus.storage 的 storage 部分，将保留周期增加到 7 天：

注意

如果您设置了较长的保留周期，从大量填充的 Prometheus 系统检索数据可能会导致查询缓慢返回结果。

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

5. 保存您的更改并关闭对象。

6. 等待 prometheus 使用新设置重启。

```
$ oc get po -l app.kubernetes.io/name=prometheus -w
```

7. 通过检查 pod 中使用的命令行参数来验证新的保留设置。

```
$ oc describe po prometheus-default-0 | grep retention.time
    --storage.tsdb.retention.time=24h
```

其他资源

- 有关指标保留时间的更多信息，请参阅 第 6.2 节 "Service Telemetry Framework 中的指标保留时间段"。

6.3. SERVICE TELEMETRY FRAMEWORK 中的警报

您可以在 Alertmanager 中的 Prometheus 和警报路由中创建警报规则。Prometheus 服务器中的警报规则将警报发送到管理警报的 Alertmanager。Alertmanager 可以静默、禁止或聚合警报，并使用电子邮件、调用通知系统或聊天平台发送通知。

要创建警报，请完成以下任务：

1. 在 Prometheus 中创建警报规则。更多信息请参阅 第 6.3.1 节 "在 Prometheus 中创建警报规则"。
2. 在 Alertmanager 中创建警报路由。您可以通过两种方式来创建警报路由：
   - 在 Alertmanager 中创建标准警报路由。
   - 在 Alertmanager 中创建带有模板模板的警报路由。

其他资源

有关使用 Prometheus 和 Alertmanager 的警报或通知的更多信息，请参阅 https://prometheus.io/docs/alerting/overview/

要查看可与 Service Telemetry Framework (STF) 搭配使用的警报示例集合，请参阅 https://github.com/infrawatch/service-telemetry-operator/tree/master/deploy/alerts

6.3.1. 在 Prometheus 中创建警报规则

Prometheus 评估警报规则以触发通知。如果规则条件返回空结果集，则条件为 false。否则，该规则为 true，它会触发警报。

流程

1. 登录到 Red Hat OpenShift Container Platform。
2. 进入 **service-telemetry** 命名空间：

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

3. 创建包含警报规则的 **PrometheusRule** 对象。Prometheus Operator 将规则加载到 Prometheus 中：

```
$ oc apply -f - <<EOF
apiVersion: monitoring.rhobs/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
```

要更改规则，请编辑 **expr** 参数的值。

4. 要验证 Operator 是否已将规则加载到 Prometheus 中，请使用基本身份验证针对 default-prometheus-proxy 路由运行 **curl** 命令：

```
$ curl -k -H "Authorization: Bearer $(oc create token stf-prometheus-reader)" https://$(oc get route default-prometheus-proxy -ogo-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,"lastEvaluation":"2021-12-07T17:23:22.160444017Z"}]}}
```

其他资源

- 有关警报的更多信息，请参阅 https://github.com/coreos/prometheus-operator/blob/master/Documentation/user-guides/alerting.md

6.3.2. 配置自定义警报

您可以将自定义警报添加到您在 第 6.3.1 节 “在 Prometheus 中创建警报规则” 中创建的 **PrometheusRule** 对象中。

流程

1. 使用 **oc edit** 命令：

```
$ oc edit prometheusrules.monitoring.rhobs prometheus-alarm-rules
```

2. 编辑 **PrometheusRules** 清单。

3. 保存并关闭清单。

其他资源

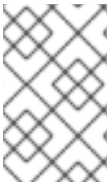
- 有关如何配置警报规则的更多信息，请参阅 https://prometheus.io/docs/prometheus/latest/configuration/alerting_rules/。
- 有关 PrometheusRules 对象的更多信息，请参阅 https://github.com/coreos/prometheus-operator/blob/master/Documentation/user-guides/alerting.md

6.3.3. 在 Alertmanager 中创建标准警报路由

使用 Alertmanager 向外部系统发送警报，如电子邮件、IRC 或其他通知频道。Prometheus Operator 将 Alertmanager 配置作为 Red Hat OpenShift Container Platform secret 进行管理。默认情况下，Service Telemetry Framework (STF) 部署了一个基本配置，它没有接收器：

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

要使用 STF 部署自定义 Alertmanager 路由，您必须将 **alertmanagerConfigManifest** 参数添加到生成由 Prometheus Operator 管理的 Service Telemetry Operator 中。



注意

如果您的 **alertmanagerConfigManifest** 包含自定义模板，例如构建发送警报的标题和文本，则必须使用 base64 编码的配置部署 **alertmanagerConfigManifest** 的内容。如需更多信息，请参阅 第 6.3.4 节 "在 Alertmanager 中创建带有模板模板的警报路由"。

流程

1. 登录到 Red Hat OpenShift Container Platform。
2. 进入 **service-telemetry** 命名空间：

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

3. 编辑 STF 部署的 **ServiceTelemetry** 对象：

```
$ oc edit stf default
```

4. 添加新参数 **alertmanagerConfigManifest** 和 **Secret** 对象内容，以定义 Alertmanager 的 **alertmanager.yaml** 配置：

注意

此步骤加载 Service Telemetry Operator 管理的默认模板。要验证更改是否已正确填充，更改值，返回 **alertmanager-default** secret，并验证新值是否已加载到内存中。例如，将参数 **global.resolve_timeout** 的值从 **5m** 改为 **10m**。

```
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. 验证配置是否已应用到 secret：

```
$ 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. 针对 **alertmanager-proxy** 服务运行 prometheus pod 中的 **wget** 命令，以检索状态和 **configYAML** 内容，并验证提供的配置是否与 Alertmanager 中的配置匹配：

```
$ oc exec -it prometheus-default-0 -c prometheus -- sh -c "wget --header \"Authorization: Bearer \$(cat /var/run/secrets/kubernetes.io/serviceaccount/token)\" https://default-alertmanager-proxy:9095/api/v1/status -q -O -"

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

7. 验证 **configYAML** 字段是否包含您预期的更改。

其他资源

- 如需有关 Red Hat OpenShift Container Platform secret 和 Prometheus operator 的更多信息，请参阅有关 警报的 Prometheus 用户指南。

6.3.4. 在 Alertmanager 中创建带有模板模板的警报路由

使用 Alertmanager 向外部系统发送警报，如电子邮件、IRC 或其他通知频道。Prometheus Operator 将 Alertmanager 配置作为 Red Hat OpenShift Container Platform secret 进行管理。默认情况下，Service Telemetry Framework (STF) 部署了一个基本配置，它没有接收器：

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

如果 **alertmanagerConfigManifest** 参数包含自定义模板，例如构建发送警报的标题和文本，则必须使用 base64 编码的配置部署 **alertmanagerConfigManifest** 的内容。

流程

1. 登录到 Red Hat OpenShift Container Platform。
2. 进入 **service-telemetry** 命名空间：

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

3. 在名为 alertmanager.yaml 的文件中创建所需的 alertmanager 配置，例如：

```
$ cat > alertmanager.yaml <<EOF
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'
EOF
```

4. 生成配置清单，并将其添加到 STF 部署的 **ServiceTelemetry** 对象中：

```
$ CONFIG_MANIFEST=$(oc create secret --dry-run=client generic alertmanager-default --from-file=alertmanager.yaml -o json)
$ oc patch stf default --type=merge -p '{"spec":{"alertmanagerConfigManifest":"'"$CONFIG_MANIFEST"'"}}'
```

5. 验证配置是否已应用到 secret：

注意

当操作员更新每个对象时，会有一个短暂的延迟

```
$ 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'
```

6. 针对 **alertmanager-proxy** 服务运行 prometheus pod 中的 **wget** 命令，以检索状态和 **configYAML** 内容，并验证提供的配置是否与 Alertmanager 中的配置匹配：

```
$ oc exec -it prometheus-default-0 -c prometheus -- /bin/sh -c "wget --header \"Authorization: Bearer \$(cat /var/run/secrets/kubernetes.io/serviceaccount/token)\" https://default-alertmanager-proxy:9095/api/v1/status -q -O -"

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

7. 验证 **configYAML** 字段是否包含您预期的更改。

其他资源

- 如需有关 Red Hat OpenShift Container Platform secret 和 Prometheus operator 的更多信息，请参阅有关 警报的 Prometheus 用户指南。

6.4. 将警报作为 SNMP 陷阱发送

要启用 SNMP 陷阱，请修改 **ServiceTelemetry** 对象并配置 **snmpTraps** 参数。SNMP 陷阱使用版本 2c 发送。

6.4.1. snmpTraps 的配置参数

snmpTraps 参数包含以下子参数来配置警报接收器：

enabled

将此子参数的值设为 true 以启用 SNMP 陷阱警报接收器。默认值为 false。

target

发送 SNMP 陷阱的目标地址。value 是一个字符串。默认为 **192.168.24.254**。

端口

发送 SNMP 陷阱的目标端口。value 是一个整数。默认为 **162**。

community

将 SNMP 陷阱发送到的目标社区。value 是一个字符串。默认为 **公共**。

retries

SNMP 陷入重试交付限制。value 是一个整数。默认值为 **5**。

timeout

SNMP 陷阱交付超时（以秒为单位）。value 是一个整数。默认为 **1**。

alertOidLabel

警报中的标签名称，用于定义 OID 值以发送 SNMP 陷阱。value 是一个字符串。默认为 **oid**。

trapOidPrefix

SNMP 陷入变量绑定的 OID 前缀。value 是一个字符串。默认为 **1.3.6.1.4.1.50495.15**。

trapDefaultOid

如果没有通过警报指定警报 OID 标签，则 SNMP 陷阱 OID。value 是一个字符串。默认为 **1.3.6.1.4.1.50495.15.1.2.1**。

trapDefaultSeverity

如果没有设置警报严重性时，SNM 陷阱严重性。value 是一个字符串。默认为空字符串。

将 **snmpTraps** 参数配置为 **ServiceTelemetry** 对象中的 **alert.alertmanager.receivers** 定义的一部分：

```
apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
metadata:
  name: default
  namespace: service-telemetry
spec:
  alerting:
    alertmanager:
      receivers:
        snmpTraps:
```

```
        alertOidLabel: oid
        community: public
        enabled: true
        port: 162
        retries: 5
        target: 192.168.25.254
        timeout: 1
        trapDefaultOid: 1.3.6.1.4.1.50495.15.1.2.1
        trapDefaultSeverity: ""
        trapOidPrefix: 1.3.6.1.4.1.50495.15
...
```

6.4.2. MIB 定义概述

SNMP 陷阱默认使用对象标识符(OID)值 **1.3.6.1.4.1.50495.15.1.2.1**。管理信息基础(MIB)模式位于 https://github.com/infrawatch/prometheus-webhook-snmp/blob/master/PROMETHEUS-ALERT-CEPH-MIB.txt。

OID 数量由以下组件值组成：* 值 **1.3.6.1.4.1** 是为私有企业定义的全局 OID。* 下一个标识符 **50495** 是 Ceph 机构的 IANA 分配的专用企业编号。* 其他值是父值的子 OID。

15

Prometheus 对象

15.1

Prometheus 警报

15.1.2

Prometheus 警报陷阱

15.1.2.1

Prometheus 警报陷阱默认

prometheus alert trap 默认是一个由几个其他子对象到 OID **1.3.6.1.4.1.50495.15** 的对象， 它由 alert. **alertmanager.receivers.snmpTraps.trapOidPrefix** 参数定义：

<trapOidPrefix>.1.1.1

警报名称

<trapOidPrefix>.1.1.2

status

<trapOidPrefix>.1.1.3

严重性

<trapOidPrefix>.1.1.4

实例

<trapOidPrefix>.1.1.5

job

<trapOidPrefix>.1.1.6

description

<trapOidPrefix>.1.1.7

labels

<trapOidPrefix>.1.1.8

timestamp

<trapOidPrefix>.1.1.9

rawdata

以下是输出一个简单的SNMP 陷阱接收器的输出示例，该接收器将收到的陷阱输出到控制台：

```
SNMPv2-MIB::snmpTrapOID.0 = OID: SNMPv2-SMI::enterprises.50495.15.1.2.1
SNMPv2-SMI::enterprises.50495.15.1.1.1 = STRING: "TEST ALERT FROM PROMETHEUS
PLEASE ACKNOWLEDGE"
SNMPv2-SMI::enterprises.50495.15.1.1.2 = STRING: "firing"
SNMPv2-SMI::enterprises.50495.15.1.1.3 = STRING: "warning"
SNMPv2-SMI::enterprises.50495.15.1.1.4 = ""
SNMPv2-SMI::enterprises.50495.15.1.1.5 = ""
SNMPv2-SMI::enterprises.50495.15.1.1.6 = STRING: "TEST ALERT FROM "
SNMPv2-SMI::enterprises.50495.15.1.1.7 = STRING: "{\"cluster\": \"TEST\", \"container\": \"sg-
core\", \"endpoint\": \"prom-https\", \"prometheus\": \"service-telemetry/default\", \"service\": \"default-
cloud1-coll-meter\", \"source\": \"SG\"}"
SNMPv2-SMI::enterprises.50495.15.1.1.8 = Timeticks: (1676476389) 194 days, 0:52:43.89
SNMPv2-SMI::enterprises.50495.15.1.1.9 = STRING: "{\"status\": \"firing\", \"labels\": {\"cluster\":
\"TEST\", \"container\": \"sg-core\", \"endpoint\": \"prom-https\", \"prometheus\": \"service-
telemetry/default\", \"service\": \"default-cloud1-coll-meter\", \"source\": \"SG\"}, \"annotations\":
{\"action\": \"TESTING PLEASE ACKNOWLEDGE, NO FURTHER ACTION REQUIRED ONLY A
TEST\"}, \"startsAt\": \"2023-02-15T15:53:09.109Z\", \"endsAt\": \"0001-01-01T00:00:00Z\",
\"generatorURL\": \"http://prometheus-default-0:9090/graph?
g0.expr=sg_total_collectd_msg_received_count+%3E+1&g0.tab=1\", \"fingerprint\":
\"feefeb77c577a02f\"}"
```

6.4.3. 配置 SNMP 陷阱

先决条件

- *确保您知道要将警报发送到的SNMP 陷阱接收器的IP 地址或主机名。*

流程

1. *登录到 Red Hat OpenShift Container Platform。*
2. *进入 **service-telemetry** 命名空间：*

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

3. *要启用 SNMP 陷阱，修改 **ServiceTelemetry** 对象：*

```
$ oc edit stf default
```

4. *设置 **alerting.alertmanager.receivers.snmpTraps** 参数：*

```
apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
...
spec:
  ...
  alerting:
```

```
    alertmanager:
      receivers:
        snmpTraps:
          enabled: true
          target: 10.10.10.10
```

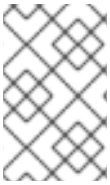
5. 确保将 **target** 的值设置为 SNMP 陷阱接收器的 IP 地址或主机名。

其它信息

有关 **snmpTraps** 可用参数的详情，请参考 第 6.4.1 节 "snmpTraps 的配置参数"。

6.4.4. 为 SNMP 陷阱创建警报

您可以通过添加由 prometheus-webhook-snmp 中间件解析的标签来创建由 SNMP 陷阱配置的警报，以定义陷阱信息和交付对象标识符(OID)。只有在需要更改特定警报定义的默认值时，才需要添加 **oid** 或 **severity** 标签。



注意

当您设置 oid 标签时，顶级 SNMP 陷阱 OID 更改，但 sub-OID 仍由全局 **trapOidPrefix** 值定义，加上子 OID 值 **.1.1.1** 到 **.1.1.9**。有关 MIB 定义的详情，请参考 第 6.4.2 节 "MIB 定义概述"。

流程

1. 登录到 Red Hat OpenShift Container Platform。
2. 进入 **service-telemetry** 命名空间：

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

3. 创建包含警报规则和包含 SNMP trap OID 覆盖值的 **oid** 标签的 **PrometheusRule** 对象：

```
$ oc apply -f - <<EOF
apiVersion: monitoring.rhobs/v1
kind: PrometheusRule
metadata:
  creationTimestamp: null
  labels:
    prometheus: default
    role: alert-rules
  name: prometheus-alarm-rules-snmp
  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
          labels:
            oid: 1.3.6.1.4.1.50495.15.1.2.1
            severity: critical
EOF
```

附加信息

有关配置警报的更多信息，请参阅 第 6.3 节 "Service Telemetry Framework 中的警报"。

6.5. 为 TLS 证书配置持续时间

要在 Service Telemetry Framework (STF) 中配置用于 AMQ Interconnect 连接的 TLS 证书的持续时间，请修改 **ServiceTelemetry** 对象并配置 certificate 参数。

6.5.1. TLS 证书的配置参数

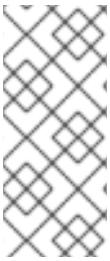
您可以使用 certificate 参数的以下子参数配置 证书 的持续时间：

endpointCertDuration

端点证书请求的持续时间或生命周期。最少接受的持续时间为 1 小时。值必须采用 Go time.ParseDuration https://golang.org/pkg/time/#ParseDuration 接受的单位。默认值为 70080h。

caCertDuration

CA 证书请求的 持续时间或生命周期。最少接受的持续时间为 1 小时。值必须采用 Go time.ParseDuration https://golang.org/pkg/time/#ParseDuration 接受的单位。默认值为 70080h。



注意

证书的默认持续时间比较长，因为在证书续订时，通常在 Red Hat OpenStack Platform 部署中复制部分证书。有关 QDR CA 证书续订过程的更多信息，请参阅 第 7 章 续订 AMQ Interconnect 证书。

您可以为 QDR 配置 certificate 参数，它是 ServiceTelemetry 对象中的 transports.qdr 定义的一部分：

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

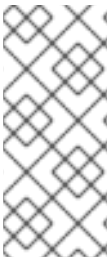
```
    certificates:
      endpointCertDuration: 70080h
      caCertDuration: 70080h
...
```

6.5.2. 配置 TLS 证书持续时间

要配置与 Service Telemetry Framework (STF)一起使用的 TLS 证书的持续时间，请修改 ServiceTelemetry 对象并配置 certificate 参数。

先决条件

- 您尚未部署 Service Telemetry Operator 实例。



注意

在创建 ServiceTelemetry 对象时，也会为 STF 创建所需的证书及其 secret。有关如何修改证书和 secret 的更多信息，请参阅： 第 7 章 续订 AMQ Interconnect 证书。以下流程对新的 STF 部署有效。

流程

1. 要编辑 TLS 证书的持续时间，您可以设置 QDR caCertDuration，例如 87600h 的 10 年：

```
$ oc apply -f - <<EOF
apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
metadata:
  name: default
  namespace: service-telemetry
spec:
  transport:
    qdr:
      enabled: true
      certificates:
        caCertDuration: 87600h
EOF
```

验证

1. 验证证书的到期日期是否正确：

```
$ oc get secret  default-interconnect-selfsigned -o jsonpath='{.data.tls\.crt}' | base64 -d
| openssl x509 -in - -text | grep "Not After"
        Not After : Mar  9 21:00:16 2033 GMT
```

6.6. 高可用性

警告

STF 高可用性(HA)模式已弃用，在生产环境中不被支持。Red Hat OpenShift Container Platform 是一个高可用性平台，如果启用了 HA 模式，您可以在 STF 中造成问题和复杂调试。

借助高可用性，Service Telemetry Framework (STF)可从组件服务中的故障中快速恢复。虽然如果节点可用于调度工作负载，Red Hat OpenShift Container Platform 会重启一个失败的 pod，但这个恢复过程可能需要一分钟以上，在此期间事件和指标会丢失。高可用性配置包括 STF 组件的多个副本，这可将恢复时间缩短到大约 2 秒。要防止 Red Hat OpenShift Container Platform 节点失败，请使用三个或更多节点将 STF 部署到 Red Hat OpenShift Container Platform 集群中。

启用高可用性有以下影响：

- 以下组件运行两个 pod，而不是默认的 pod：
  - AMQ Interconnect
  - Alertmanager
  - Prometheus
  - 事件智能网关
  - 指标智能网关

- 恢复其中任何服务丢失的 pod 的时间会减少到大约 2 秒。

6.6.1. 配置高可用性

要配置 Service Telemetry Framework (STF)以实现高可用性，请将 highAvailability.enabled: true 添加到 Red Hat OpenShift Container Platform 中的 ServiceTelemetry 对象中。您可以在安装时设置此参数，或者如果您已经部署了 STF，请完成以下步骤：

流程

1. 登录到 Red Hat OpenShift Container Platform。
2. 进入 service-telemetry 命名空间：

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

3. 使用 oc 命令编辑 ServiceTelemetry 对象：

```
$ oc edit stf default
```

4. 将 highAvailability.enabled: true 添加到 spec 部分：

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

5. 保存您的更改并关闭对象。

6.7. SERVICE TELEMETRY FRAMEWORK 中的可观察性策略

Service Telemetry Framework (STF)不包括事件存储后端或仪表板工具。STF 可以使用社区操作器为 Grafana 创建数据源配置，以提供仪表板接口。

除了让 Service Telemetry Operator 创建自定义资源请求外，您可以使用您自己的应用程序部署或其他兼容应用程序，并提取指标 智能网关以发送到您自己的 Prometheus 兼容系统以进行遥测存储。如果将 observabilityStrategy 设置为 none，则不会部署存储后端，因此 STF 不需要持久性存储。

使用 STF 对象中的 observabilityStrategy 属性来指定要部署的可观察性组件类型。

可用的值如下：

value	含义
use_redhat	红帽支持的组件由 STF 请求。这包括 Cluster Observability Operator 中的 Prometheus 和 Alertmanager，但没有对 Kubernetes (ECK) Operator 上的 Elastic Cloud 的请求。如果启用，也会从 Grafana Operator （社区组件）请求资源。
use_hybrid	除了红帽支持的组件外，还会请求 Elasticsearch 和 Grafana 资源（如果在 ServiceTelemetry 对象中指定）
use_community	使用 Prometheus Operator 的社区版本，而不是 Cluster Observability Operator。还请求 Elasticsearch 和 Grafana 资源（如果在 ServiceTelemetry 对象中指定）
none	没有部署存储或警报组件

注意

新部署的 STF 环境为 1.5.3 默认为 use_redhat。在 1.5.3 默认之前创建的现有 STF 部署 以使用_community。

要将现有 STF 部署迁移到 use_redhat，请参阅红帽知识库文章将服务 Telemetry Framework 迁移到完全支持的操作器。

6.7.1. 配置备用可观察性策略

要跳过存储、视觉化和警报后端的部署，请将 observabilityStrategy: none 添加到 ServiceTelemetry spec 中。在这个模式中，您仅部署 AMQ Interconnect 路由器和智能网关，您必须配

置与 Prometheus 兼容的外部系统，以从 STF 智能网关和外部 Elasticsearch 收集指标，以接收转发的事件。

流程

1. *在 spec 参数中，使用属性 observabilityStrategy: none 创建一个 ServiceTelemetry 对象。清单显示会导致 STF 的默认部署，该部署适合从具有所有指标收集器类型的单个云接收遥测。*

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

2. *删除由社区操作器管理的剩余对象*

```
$ for o in alertmanagers.monitoring.rhobs/default
prometheuses.monitoring.rhobs/default elasticsearch/elasticsearch grafana/default-
grafana; do oc delete $o; done
```

3. *要验证所有工作负载是否都正常运行，请查看 pod 和每个 pod 的状态：*

```
$ oc get pods
NAME                                          READY   STATUS    RESTARTS   AGE
default-cloud1-ceil-event-smartgateway-6f8547df6c-p2db5   3/3     Running   0
132m
default-cloud1-ceil-meter-smartgateway-59c845d65b-gzhcs   3/3     Running   0
132m
default-cloud1-coll-event-smartgateway-bf859f8d77-tzb66   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
```

其他资源

- 有关配置附加云或更改支持的收集器集合的更多信息，请参阅 第 4.3.2 节 “部署智能网关”。

- 要将现有 STF 部署迁移到 use_redhat，请参阅红帽知识库文章将服务 Telemetry Framework 迁移到完全支持的操作器。

6.8. RED HAT OPENSTACK PLATFORM 服务的资源使用

您可以通过显示没有计算功能的服务来监控 Red Hat OpenStack Platform (RHOSP)服务的资源使用情况，如 API 和其他基础架构进程，以识别 overcloud 中的瓶颈。默认启用资源使用情况监控。

其他资源

- 要禁用资源使用情况监控，请参阅 第 6.8.1 节 “禁用 Red Hat OpenStack Platform 服务的资源使用情况监控”。

6.8.1. 禁用 Red Hat OpenStack Platform 服务的资源使用情况监控

要禁用 RHOSP 容器化服务资源使用情况的监控，您必须将 CollectdEnableLibpodstats 参数设置为 false。

前提条件

- 您已创建了 stf-connectors.yaml 文件。如需更多信息，请参阅 第 4.1 节 “使用 director 为 Service Telemetry Framework 部署 Red Hat OpenStack Platform overcloud”。

- 您使用最新版本的 Red Hat OpenStack Platform (RHOSP) 17.1。

流程

1. 打开 stf-connectors.yaml 文件，并添加 CollectdEnableLibpodstats 参数来覆盖 enable-stf.yaml 中的设置。在 enable-stf.yaml 后，确保从 openstack overcloud deploy 命令调用 stf-connectors.yaml ：

```
CollectdEnableLibpodstats: false
```

2. 继续 overcloud 部署过程。更多信息请参阅 第 4.1.6 节 “部署 overcloud”。

6.9. RED HAT OPENSTACK PLATFORM API 状态和容器化服务健康状况

您可以通过定期运行健康检查脚本，使用 OCI （开放容器项目）标准来评估每个 Red Hat OpenStack Platform (RHOSP)服务的容器健康状况。大多数 RHOSP 服务实施健康检查，用于记录问题并返回二进制状态。对于 RHOSP API，健康检查会查询根端点，并根据响应时间确定健康状况。

默认启用对 RHOSP 容器健康和 API 状态的监控。

其他资源

- 要禁用 RHOSP 容器健康和 API 状态监控，请参阅 第 6.9.1 节 “禁用容器健康和 API 状态监控”。

6.9.1. 禁用容器健康和 API 状态监控

要禁用 RHOSP 容器化服务健康和 API 状态监控，您必须将 CollectdEnableSensubility 参数设置为 false。

前提条件

- 您已在 templates 目录中创建了 stf-connectors.yaml 文件。如需更多信息，请参阅 第 4.1 节 “使用 director 为 Service Telemetry Framework 部署 Red Hat OpenStack Platform overcloud”。
- 您使用最新版本的 Red Hat OpenStack Platform (RHOSP) 17.1。

流程

1. 打开 stf-connectors.yaml，并添加 CollectdEnableSensubility 参数来覆盖 enable-stf.yaml 中的设置。在 enable-stf.yaml 后，确保从 openstack overcloud deploy 命令调用 stf-connectors.yaml ：

```
CollectdEnableSensubility: false
```

2. 继续 overcloud 部署过程。更多信息请参阅 第 4.1.6 节 “部署 overcloud”。

其他资源

- *有关多个云地址的详情请参考 第 4.3 节 “配置多个云”。*

第 7 章 续订 AMQ INTERCONNECT 证书

当证书过期时，您必须续订保护 Red Hat OpenStack Platform (RHOSP)和 Service Telemetry Framework (STF)之间的 AMQ Interconnect 连接的 CA 证书。续订由 Red Hat OpenShift Container Platform 中的 cert-manager 组件自动处理，但您必须手动将更新的证书复制到 RHOSP 节点。

7.1. 检查已过期的 AMQ INTERCONNECT CA 证书

当 CA 证书过期时，AMQ Interconnect 连接会保留，但如果它们中断，则无法重新连接。最后，来自 Red Hat OpenStack Platform (RHOSP)分配路由器的一些或所有连接都失败，在两端显示错误，以及 CA 证书中的 expiry 或 Not After 字段。

流程

1. 登录到 Red Hat OpenShift Container Platform。

2. 进入 service-telemetry 命名空间：

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

3. 验证部分或所有分配路由器连接是否都失败：

```
$ oc exec -it deploy/default-interconnect -- qdstat --connections | grep Router | wc
      0       0       0
```

4. 检查 Red Hat OpenShift Container Platform-hosted AMQ Interconnect 日志中出现的这个错误：

```
$ 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. 登录到您的 RHOSP undercloud。

6. 检查在带有失败连接的节点的 RHOSP 托管 AMQ Interconnect 日志中出现这个错误：

```
$ 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. 通过检查 RHOSP 节点上的文件来确认 CA 证书已过期：

```
$ 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
```

7.2. 更新 AMQ INTERCONNECT CA 证书

要更新 AMQ Interconnect 证书，您必须从 Red Hat OpenShift Container Platform 导出它，并将其复制到 Red Hat OpenStack Platform (RHOSP)节点。

流程

1. 登录到 Red Hat OpenShift Container Platform。
2. 进入 service-telemetry 命名空间：

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

3. 将 CA 证书导出到 STFCA.pem ：

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

4. 将 STFCA.pem 复制到您的 RHOSP undercloud。
5. 登录到您的 RHOSP undercloud。
6.

编辑 stf-connectors.yaml 文件，使其包含新的 caCertFileContent。如需更多信息，请参阅 第 4.1.5 节 “为 overcloud 配置 STF 连接”。

7. 将 STFCA.pem 文件复制到每个 RHOSP overcloud 节点：

```
[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. 重启每个 RHOSP overcloud 节点上的 metrics_qdr 容器：

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

注意

在复制 STFCA.pem 文件并重启 metrics_qdr 容器后，您不需要部署 overcloud。您可以编辑 stf-connectors.yaml 文件，以便将来的部署不会覆盖新的 CA 证书。

第 8 章 从 RED HAT OPENSHIFT CONTAINER PLATFORM 环境中删除 SERVICE TELEMETRY FRAMEWORK

如果您不再需要 STF 功能，从 Red Hat OpenShift Container Platform 环境中删除 Service Telemetry Framework (STF)。

要从 Red Hat OpenShift Container Platform 环境中删除 STF，您必须执行以下任务：

1. 删除命名空间。
2. 删除 cert-manager Operator。
3. 删除 Cluster Observability Operator。

8.1. 删除命名空间

要从 Red Hat OpenShift Container Platform 中删除 STF 的操作资源，请删除命名空间。

流程

1. 运行 oc delete 命令：

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

2. 验证资源是否已从命名空间中删除：

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

8.2. 为 RED HAT OPENSHIFT 删除 CERT-MANAGER OPERATOR

如果您没有将 cert-manager Operator for Red Hat OpenShift 用于任何其他应用程序，请删除 Subscription、ClusterServiceVersion 和 CustomResourceDefinitions。

有关为 Red Hat OpenShift Operator 删除 cert-manager 的更多信息，请参阅 OpenShift Container Platform 文档中的为 Red Hat OpenShift 删除 cert-manager Operator。

其他资源

- 从集群中删除 Operator。

8.3. 删除 CLUSTER OBSERVABILITY OPERATOR

如果您没有将 Cluster Observability Operator 用于任何其他应用程序，请删除 Subscription、ClusterServiceVersion 和 CustomResourceDefinitions。

有关删除 Cluster Observability Operator 的更多信息，请参阅 OpenShift Container Platform 文档中的使用 Web 控制台卸载 Cluster Observability Operator。

其他资源

- 从集群中删除 Operator。