Red Hat OpenStack Platform 13

Service Telemetry Framework 1.5

安装并部署服务 Telemetry Framework 1.5

Last Updated: 2023-10-26

Red Hat OpenStack Platform 13 Service Telemetry Framework 1.5

安装并部署服务 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服务 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。

使用直接文档反馈(DDF)功能

使用 **添加反馈** DDF 功能，用于特定句子、段落或代码块上的直接注释。

1. 以 *Multi-page HTML* 格式查看文档。
2. 请确定您看到文档右上角的 **反馈** 按钮。
3. 用鼠标指针高亮显示您想评论的文本部分。
4. 点 **添加反馈**。
5. 在**添加反馈**项中输入您的意见。
6. 可选：添加您的电子邮件地址，以便文档团队可以联系您以讨论您的问题。
7. 点 **Submit**。

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了解历史信息。
- 在仪表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。

监控数据可以是指标数据，也可以是事件：

指标

应用程序或系统的数字测量。

事件

在系统中发生的非法和离散性。

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到 Red Hat OpenShift Container Platform 上。

重要

STF 与 Red Hat OpenShift Container Platform 版本 4.10 兼容

其他资源

- 有关如何部署 Red Hat OpenShift Container Platform 的更多信息，请参阅 Red Hat OpenShift Container Platform 产品文档。
- 您可以在云平台或裸机上安装 Red Hat OpenShift Container Platform。有关 STF 性能和扩展的更多信息，请参阅 https://access.redhat.com/articles/4907241。
- 您可以在裸机或其他支持的云平台上安装 Red Hat OpenShift Container Platform。有关安装 Red Hat OpenShift Container Platform 的更多信息，请参阅 OpenShift Container Platform 4.10 文档。

1.1. 支持服务 TELEMETRY FRAMEWORK

红帽支持两个最新版本的服务 Telemetry Framework (STF)。不支持更早的版本。如需更多信息，请参阅 Service Telemetry Framework 支持的版本列表。

红帽支持核心 Operator 和工作负载，包括 AMQ Interconnect、Service Telemetry Operator 和 Smart Gateway Operator。红帽不支持社区 Operator 或工作负载组件，如 Elasticsearch、Prometheus、Alertmanager、Grafana 以及它们的 Operator。

您只能在完全连接的网络环境中部署 STF。您无法在 Red Hat OpenShift Container Platform 断开连接环境或网络代理环境中部署 STF。

1.2. 服务 TELEMETRY 框架架构

服务遥测框架(STF)使用客户端-服务器架构，其中 Red Hat OpenStack Platform (RHOSP)是客户端，而 Red Hat OpenShift Container Platform 是服务器。

STF 由以下组件组成：

- 数据收集
  - collectd：收集基础架构指标和事件。
  - Ceilometer：收集 RHOSP 指标和事件。
- 传输
  - AMQ Interconnect： AMQP 1.x 兼容消息传递总线，提供快速、可靠的数据传输以将指标传输到 STF 以进行存储。
  - Smart Gateway ：一个 Golang 应用程序，它从 AMQP 1.x 总线中获取指标数据和事件，以传送到 Elasticsearch 或 Prometheus。
- 数据存储
  - Prometheus：存储从智能网关收到的 STF 指标的时间序列数据存储。
  - Elasticsearch：事件数据存储存储从智能网关接收的 STF 事件。
- 观察
  - Alertmanager：一个警报工具，它使用 Prometheus 警报规则来管理警报。
  - 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。

下表描述了客户端和服务器组件的应用程序：

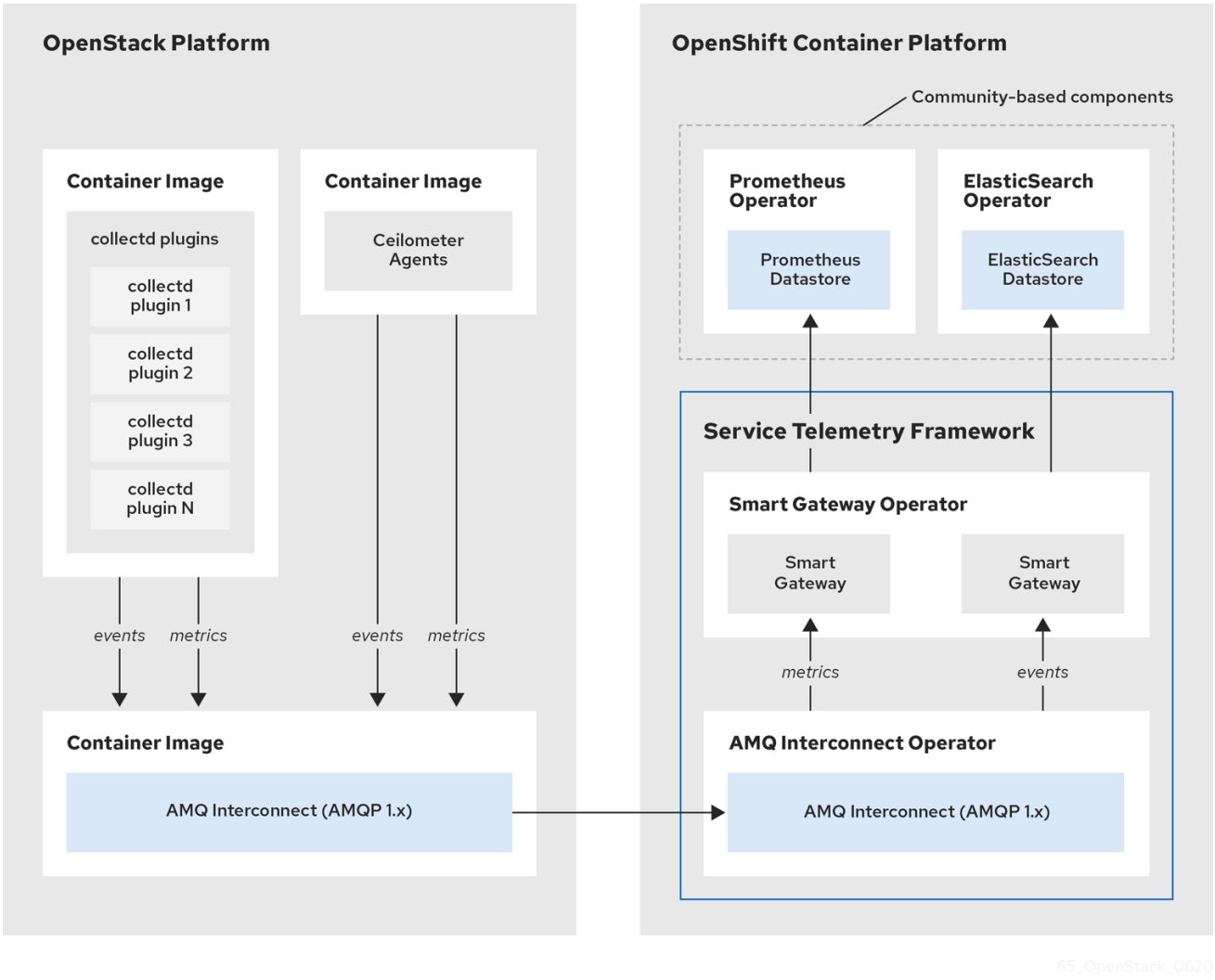
表 1.1. STF 的客户端和服务器组件

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
collectd	是	否
Ceilometer	是	否

重要

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

图 1.1. 服务 Telemetry 框架架构概述



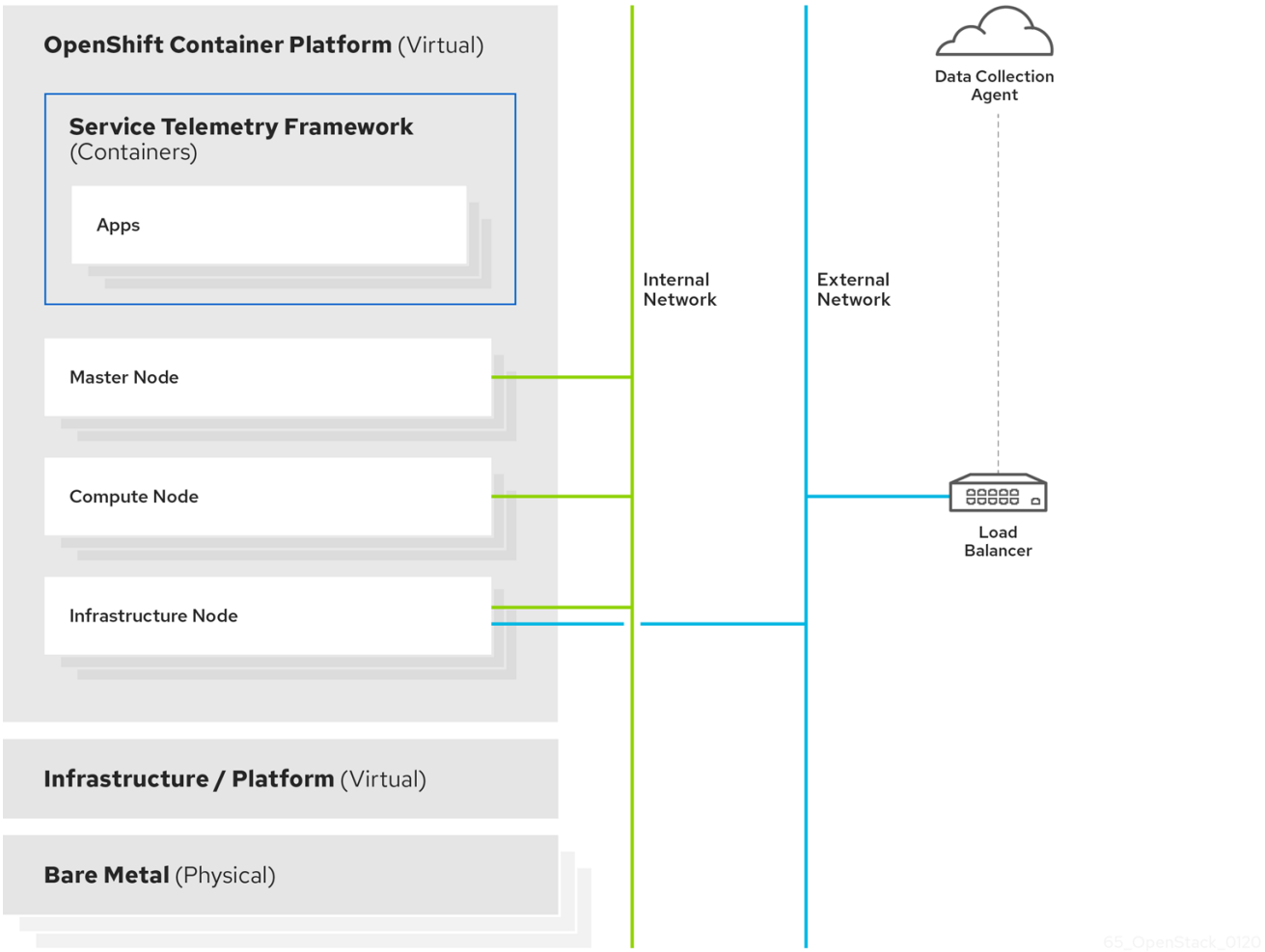
对于客户端侧指标，collectd 提供没有项目数据的基础架构指标，Ceilometer 则根据项目或用户工作负载提供 RHOSP 平台数据。Ceilometer 和 collectd 使用 AMQ Interconnect 传输将数据提供给 Prometheus，通过消息总线提供数据。在服务器端，名为 Smart Gateway 的 Golang 应用程序从总线中获取数据流，并将其作为 Prometheus 的本地提取端点公开。

如果您计划通过 AMQ Interconnect 传输来收集和存储事件，A collectd 和 Ceilometer 将事件数据提供给服务器端。另一个智能网关将数据写入 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 4.10
- 基础架构平台

图 1.2. Server-side STF 监控基础架构



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。
- 存储数据的时间长度。

Service Telemetry Framework (STF)安装取决于现有的 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* 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对应的安装文档。

第 2 章 为服务 TELEMETRY FRAMEWORK 准备 RED HAT OPENSHIFT CONTAINER PLATFORM 环境

要为服务 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 集群中有可用的持久性存储用于生产级部署。更多信息请参阅 第 2.2 节 “持久性卷（PV）”。
- 确保有足够的资源可用于运行 Operator 和应用程序容器。更多信息请参阅 第 2.3 节 “资源分配”。
- 确定您有完全连接的网络环境。更多信息请参阅 第 2.4 节 “Service Telemetry Framework 的网络注意事项”。

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

服务遥测框架(STF)不包括存储后端和警报工具。STF 使用社区操作器来部署 Prometheus、Alertmanager、Grafana 和 Elasticsearch。STF 向这些社区运营商提出请求，以创建配置为与 STF 配合使用的每个应用程序实例。

除了使用 Service Telemetry Operator 创建自定义资源请求外，您可以使用您自己的部署这些应用程序或其他兼容的应用程序，并提取指标智能网关以传送到您自己的 Prometheus 兼容系统进行遥测存储。如果您将 observability 策略设置为使用替代后端，则 STF 不需要持久性或临时存储。

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 和 Elasticsearch 可以存储指标和事件。

当您通过 Service Telemetry Operator 启用持久性存储时，在 STF 部署中请求的持久性卷声明(PVC)会导致 RWO (ReadWriteOnce)访问模式。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

- 如需有关为 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 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 中配置持久性存储的详情，请参考 “为 Prometheus 配置持久性存储”一节。
- 有关在 STF 中为 Elasticsearch 配置持久性存储的更多信息，请参阅 “为 Elasticsearch 配置持久性存储”一节。

2.2.1. 临时存储

您可以使用临时存储来运行服务 Telemetry Framework (STF)，而无需将数据存储在 Red Hat OpenShift Container Platform 集群中。

警告

如果使用临时存储，当 pod 重启、更新或重新调度到另一个节点上时，可能会遇到数据丢失。仅对开发或测试使用临时存储，而不适用于生产环境。

2.3. 资源分配

要在 Red Hat OpenShift Container Platform 基础架构中调度 pod，需要运行的组件的资源。如果您没有分配足够资源，pod 会保持 **Pending** 状态，因为它们无法调度。

运行 Service Telemetry Framework (STF)的资源数量取决于您的环境和要监控的节点和云。

其他资源

- 有关指标集合大小建议，请参阅服务 Telemetry Framework 性能和扩展。
- 有关 Elasticsearch 大小要求的详情，请参考 https://www.elastic.co/guide/en/cloud-on-k8s/current/k8s-managing-compute-resources.html。

2.4. SERVICE TELEMETRY FRAMEWORK 的网络注意事项

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

第 3 章 安装服务 TELEMETRY FRAMEWORK 的核心组件

您可以使用 Operator 来加载服务遥测框架(STF)组件和对象。Operator 管理以下 STF 核心和社区组件中的每个组件：

- cert-manager
- AMQ Interconnect
- 智能网关
- Prometheus 和 AlertManager
- Elasticsearch
- Grafana

先决条件

- Red Hat OpenShift Container Platform 版本 4.10 正在运行。
- 您已准备了 Red Hat OpenShift Container Platform 环境，并确保在 Red Hat OpenShift Container Platform 环境之上有持久性存储和足够资源来运行 STF 组件。如需更多信息，请参阅 Service Telemetry Framework 性能和扩展。
- 您的环境已完全连接。STF 无法在 Red Hat OpenShift Container Platform 断开连接环境或网络代理环境中工作。

重要

STF 与 Red Hat OpenShift Container Platform 版本 4.10 兼容

其他资源

- 如需有关 Operator 的更多信息，*请参阅了解 Operators* 指南。

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、存储和监控事件：

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 OperatorGroups。

3. 为 cert-manager Operator 创建命名空间：

```
$ oc create -f - <<EOF
apiVersion: project.openshift.io/v1
kind: Project
metadata:
  name: openshift-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: openshift-cert-manager-operator
  namespace: openshift-cert-manager-operator
spec: {}
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: openshift-cert-manager-operator
spec:
  channel: tech-preview
  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 的阶段显示为 **Succeeded** ：

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

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

7. 使用 redhat-operators CatalogSource 订阅 AMQ Interconnect Operator:

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

8. 验证您的 ClusterServiceVersion。确保 amq7-interconnect-operator.v1.10.10 显示阶段 **Succeeded** ：

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

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

9. 启用 OperatorHub.io Community Catalog Source 来安装数据存储和视觉化 Operator：

警告

红帽支持核心 Operator 和工作负载，包括 AMQ Interconnect、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、Prometheus、Alertmanager、Grafana 及其 Operator。

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

```
  displayName: OperatorHub.io Operators
  publisher: OperatorHub.io
EOF
```

10. 如果计划将指标存储在 Prometheus 中，则必须启用 Prometheus Operator。要启用 Prometheus Operator，请在 Red Hat OpenShift Container Platform 环境中创建以下清单：

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

11. 验证 Prometheus **Succeeded** 的 ClusterServiceVersion:

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

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

12. 如果计划将事件存储在 Elasticsearch 中，则必须在 Kubernetes (ECK) Operator 中启用 Elastic Cloud。要启用 ECK Operator，请在 Red Hat OpenShift Container Platform 环境中创建以下清单：

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

13. 验证 Kubernetes **Succeeded** 上的 Elastic Cloud 的 ClusterServiceVersion:

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

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

14. 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
```

15. 验证 Service Telemetry Operator 和依赖 Operator：

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

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

3.2. 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”。

流程

1. 要创建一个会生成使用默认值的 STF 部署的 **ServiceTelemetry** 对象，创建一个带有空 **spec** 参数的 **ServiceTelemetry** 对象：

```
$ oc apply -f - <<EOF
apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
```

```
metadata:
  name: default
  namespace: service-telemetry
spec: {}
EOF
```

要覆盖默认值，请定义您要覆盖的参数。在本例中，通过将 **enabled** 设置为 **true** 来启用 Elasticsearch：

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

使用空 **spec** 参数创建 **ServiceTelemetry** 对象会导致带有以下默认设置的 STF 部署：

```
apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
metadata:
  name: default
  namespace: service-telemetry
spec:
  alerting:
    alertmanager:
      receivers:
        snmpTraps:
          enabled: false
          target: 192.168.24.254
      storage:
        persistent:
          pvcStorageRequest: 20G
        strategy: persistent
    enabled: true
  backends:
    events:
      elasticsearch:
        enabled: false
        storage:
          persistent:
            pvcStorageRequest: 20Gi
          strategy: persistent
        version: 7.16.1
    logs:
      loki:
        enabled: false
        flavor: 1x.extra-small
        replicationFactor: 1
```

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

要覆盖这些默认值，请将配置添加到 **spec** 参数。

2. 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。

注意

如果将 backend.**events.elasticsearch.enabled** 参数设置为 **true**，则通知智能网关会在 Elasticsearch 启动前报告 **Error** 和 **CrashLoopBackOff** 错误消息。

```
$ oc get pods

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

3.2.1. ServiceTelemetry 对象的主要参数

ServiceTelemetry 对象包含以下主要配置参数：

- **警报**
- **后端**
- **clouds**
- **graphing**
- **highAvailability**
- **传输**

您可以配置每个配置参数，以在 STF 部署中提供不同的功能。

backends 参数
使用 **backends** 参数来控制哪些存储后端可用于存储指标和事件，并控制 **cloud** 参数所定义的智能网关启用。更多信息请参阅 “clouds 参数”一节。

目前，您可以使用 Prometheus 作为指标存储后端，Elasticsearch 作为事件存储后端。

为指标启用 Prometheus 作为存储后端

要启用 Prometheus 作为指标的存储后端，您必须配置 **ServiceTelemetry** 对象。

流程

- 配置 **ServiceTelemetry** 对象：

```
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

使用 **backends.metrics.prometheus.storage.persistent** 中定义的附加参数来为 Prometheus 配置持久性存储选项，如存储类和卷大小。

使用 **storageClass** 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** 参数来定义满足存储请求的最低所需卷大小。如果静态定义了卷，则可以使用大于请求的卷大小。默认情况下，Service Telemetry Operator 请求大小为 **20G** (20 千兆字节)。

流程

- 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
```

- 配置 **ServiceTelemetry** 对象：

```
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
```

启用 Elasticsearch 作为事件存储后端
要启用 Elasticsearch 作为事件的存储后端，您必须配置 **ServiceTelemetry** 对象。

流程

- 配置 **ServiceTelemetry** 对象：

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

为 Elasticsearch 配置持久性存储
使用 backend.**events.elasticsearch.storage.persistent** 中定义的附加参数，为 Elasticsearch 配置持久性存储选项，如存储类和卷大小。

使用 **storageClass** 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** 参数来定义满足存储请求的最低所需卷大小。如果静态定义了卷，则可以使用大于请求的卷大小。默认情况下，Service Telemetry Operator 请求大小为 **20Gi** (20 千兆字节)。

流程

- 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
```

- 配置 **ServiceTelemetry** 对象：

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

```
backends:
  events:
    elasticsearch:
      enabled: true
      version: 7.16.1
      storage:
        strategy: persistent
        persistent:
          storageClass: standard-csi
          pvcStorageRequest: 50G
```

clouds 参数

使用 **clouds** 参数定义部署了哪些智能网关对象，从而为多个监控的云环境提供接口以连接到 STF 实例。如果支持的后端可用，则创建默认云配置的指标和事件智能卡。默认情况下，Service Telemetry Operator 为 **cloud1** 创建智能网关。

您可以创建云对象列表，以控制为定义的云创建智能网关。每个云由数据类型和收集器组成。数据类型是 **指标** 或 **事件**。每一数据类型包含收集器列表、消息总线订阅地址列表，以及启用调试的参数。可用的指标收集器是 **collectd**、**ceilometer** 和 **sensubility**。事件的可用收集器是 **collectd** 和 **ceilometer**。确保这些收集器的订阅地址对于每一个云、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，该对象为 collectd、Ceilomeility data 收集和事件提供订阅和数据存储，用于特定云实例：

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

clouds 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**。**指标** **和事件** 参数代表该数据类型的存储的相应后端。**collectors** 参数指定由两个所需参数（即 **collectorType** 和 **subscriptionAddress** ）组成的对象列表，这些对象代表一个智能网关的实例。**collectorType** 参数指定由 collectd、Ceilometer 或 Sensubility 收集的数据。**subscriptionAddress** 参数提供 Smart Gateway 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 4.5.3 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 4.5 节 “配置多个云”。

警报参数
使用 **警报** 参数控制 Alertmanager 实例的创建以及存储后端的配置。默认情况下启用 **警报**。更多信息请参阅 第 5.3 节 “Service Telemetry Framework 中的警报”。

graphing 参数
使用 **graphing** 参数来控制 Grafana 实例的创建。默认情况下，**图形** 会被禁用。更多信息请参阅 第 5.1 节 “Service Telemetry Framework 中的仪表板”。

highAvailability 参数
使用 **highAvailability** 参数来控制 STF 组件的多个副本的实例化，以减少故障或重新调度组件的恢复时间。默认情况下禁用 **高可用性**。更多信息请参阅 第 5.5 节 “高可用性”。

transports 参数
使用 **transports** 参数来控制用于 STF 部署的消息总线启用。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 ingress 集群流量。

在 Service Telemetry Framework (STF)中，HTTPS 路由面向具有基于 Web 的界面的每个服务公开。这些路由受 Red Hat OpenShift Container Platform RBAC 和具有 **ClusterRoleBinding** 的任何用户保护，供他们查看 Red Hat OpenShift Container Platform 命名空间可以登录。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 Red Hat OpenShift Container Platform。
2. 进入 **service-telemetry** 命名空间：

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

3. 列出 **service-telemetry** 项目中可用的 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 Web 浏览器中，导航到 https://<route_address > 以访问对应服务的 Web 界面。

3.4. 配置备用可观察性策略

要将 STF 配置为跳过存储、视觉化和警报后端的部署，请将 **observabilityStrategy: none** 添加到 ServiceTelemetry spec。在这个模式中，只有 AMQ Interconnect 路由器和指标智能网关会被部署，您必须配置外部 Prometheus 兼容系统，以从 STF 智能网关收集指标。

注意

目前，只有将 **observabilityStrategy** 设置为 **none** 时支持指标。不会部署事件智能网关。

流程

1. 在 **spec** 参数中创建一个带有属性 **observabilityStrategy: none** 的 **ServiceTelemetry** 对象。清单显示的结果是采用 STF 的默认部署，适用于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. 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-meter-smartgateway-59c845d65b-gzhcs   3/3     Running   0          132m
default-cloud1-coll-meter-smartgateway-75bbd948b9-d5phm   3/3     Running   0          132m
default-cloud1-sens-meter-smartgateway-7fdbb57b6d-dh2g9   3/3     Running   0          132m
default-interconnect-668d5bbcd6-57b2l               1/1     Running   0          132m
interconnect-operator-b8f5bb647-tlp5t               1/1     Running   0          47h
service-telemetry-operator-566b9dd695-wkvjq         1/1     Running   0          156m
smart-gateway-operator-58d77dcf7-6xsq7              1/1     Running   0          47h
```

其他资源

有关配置其他云或更改支持的收集器集合的更多信息，请参阅 第 4.5.2 节 “部署智能网关”

第 4 章 为服务 TELEMETRY FRAMEWORK 配置 RED HAT OPENSTACK PLATFORM

要收集指标、事件或两个对象，并将其发送到服务遥测框架(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 可支持单一和多个云。RHOSP 和 STF 中的默认配置为单个云安装设置。

- 有关使用默认配置的单个 RHOSP overcloud 部署，请参阅 第 4.1 节 “为服务 Telemetry Framework 部署 Red Hat OpenStack Platform overcloud”。
- 要针对多个云规划 RHOSP 安装和配置 STF，请参阅 第 4.5 节 “配置多个云”。
- 作为 RHOSP overcloud 部署的一部分，您可能需要在您的环境中配置附加功能：
  - 要在 RHOSP 云节点上将数据收集和传输部署到使用路由 L3 域的 RHOSP 云节点上，如分布式计算节点(DCN)或 spine-leaf，请参阅 第 4.3 节 “部署到非标准网络拓扑”。
  - 要禁用数据收集器服务，请参阅 第 4.2 节 “禁用用于服务 Telemetry Framework 的 Red Hat OpenStack Platform 服务”。
  - 如果将容器镜像同步到本地 registry，您必须创建一个环境文件，并包含容器镜像的路径。更多信息请参阅 第 4.4 节 “将 Red Hat OpenStack Platform 容器镜像添加到 undercloud” 。

4.1. 为服务 TELEMETRY FRAMEWORK 部署 RED HAT OPENSTACK PLATFORM OVERCLOUD

作为 Red Hat OpenStack Platform (RHOSP) overcloud 部署的一部分，您必须配置数据收集器，并将数据传输配置为 Service Telemetry Framework (STF)。

流程

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

4.1.1. 从用于 overcloud 配置的服务 Telemetry Framework 获取 CA 证书

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

流程

1. 查看 Service Telemetry Framework 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 路由地址

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

流程

1. 登录您的 Red Hat OpenShift Container Platform 环境。
2. 在 **service-telemetry** 项目中，检索 AMQ Interconnect 路由地址：

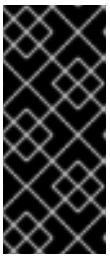
```
$ oc get routes -ogo-template='{{ range .items }}{{printf "%s\n" .spec.host }}{{ end }}' | grep "\-5671"
default-interconnect-5671-service-telemetry.apps.infra.watch
```

4.1.3. 为 STF 创建基本配置

要配置基础参数，以便为服务 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 Red Hat OpenStack Platform (RHOSP) undercloud。
2. 在 **/home/stack** 目录中创建一个名为 **enable-stf.yaml** 的配置文件。



重要

将 **EventPipelinePublishers** 和 **PipelinePublishers** 设置为空列表会导致事件或指标数据传递到 RHOSP 遥测组件，如 Gnocchi 或 Panko。如果您需要将数据发送到其他管道，CeiloConfig 轮询间隔为 30 秒（如 **ExtraConfig** 中所指定）可能会导致 RHOSP 遥测组件，且您必须将间隔增加到更大值，如 **300**。将该值增大到较长的轮询间隔会导致 STF 中的遥测分辨率减少。

enable-stf.yaml

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

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

  # enable Ceilometer metrics and events
```

```
    CeilometerQdrPublishMetrics: true
    CeilometerQdrPublishEvents: true

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

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

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

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

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

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

      # align defaults across OSP versions
      collectd::plugin::cpu::reportbycpu: true
      collectd::plugin::cpu::reportbystate: true
      collectd::plugin::cpu::reportnumcpu: false
      collectd::plugin::cpu::valuespercentage: true
      collectd::plugin::df::ignoreselected: true
```

```
    collectd::plugin::df::reportbydevice: true
    collectd::plugin::df::fstypes: ['xfs']
    collectd::plugin::load::reportrelative: true
    collectd::plugin::virt::extra_stats: "pcpu cpu_util vcpupin vcpu memory disk disk_err
disk_allocation disk_capacity disk_physical domain_state job_stats_background perf"
```

4.1.4. 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，其中包含用于 overcloud 的 AMQ Interconnect 的连接配置到 STF 部署。启用 STF 中事件的事件和存储的集合，并且部署 overcloud。默认配置适用于具有默认消息总线主题的单一云实例。有关配置多个云部署，请参阅 第 4.5 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 “从用于 overcloud 配置的服务 Telemetry Framework 获取 CA 证书”。
- 检索 AMQ Interconnect 路由地址。更多信息请参阅 第 4.1.2 节 “检索 AMQ Interconnect 路由地址”。

流程

1. 以 **stack** 用户身份登录到 RHOSP undercloud。
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，将 overcloud 上的 AMQ Interconnect 连接到 STF 部署。您可以在此文件中为 Sensubility、Ceilometer 和 collectd 配置主题地址，以匹配 STF 中的默认值。有关自定义主题和云配置的更多信息，请参阅 第 4.5 节 “配置多个云”。
    - **resource_registry** 配置直接加载 collectd 服务，因为您不为多个云部署包含 **collectd-write-qdr.yaml** 环境文件。
    - 将 **host** 参数替换为您在 第 4.1.2 节 “检索 AMQ Interconnect 路由地址” 中检索的 **HOST/PORT** 值。
    - 将 **caCertFileContent** 参数替换为在 第 4.1.1 节 “从用于 overcloud 配置的服务 Telemetry Framework 获取 CA 证书” 中检索的内容。
    - 将 **MetricsQdrConnectors 的主机** 子参数替换为您在 第 4.1.2 节 “检索 AMQ Interconnect 路由地址” 中检索的 **HOST/PORT** 的值。
    - 设置 **CeilometerQdrEventsConfig.topic**，以定义 Ceilometer 事件的主题。这个值的格式为 **anycast/ceilometer/cloud1-event.sample**。
    - 设置 **CeilometerQdrMetricsConfig.topic**，以定义 Ceilometer 指标的主题。这个值的格式为 **anycast/ceilometer/cloud1-metering.sample**。
    - 设置 **CollectdAmqpInstances**，以定义 collectd 事件的主题。这个值的格式为 **collectd/cloud1-notify**。
    - 设置 **CollectdAmqpInstances**，以定义 collectd 指标的主题。这个值的格式为 **collectd/cloud1-telemetry**。

stf-connectors.yaml

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

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

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

    CeilometerQdrEventsConfig:
      driver: amqp
      topic: cloud1-event

    CeilometerQdrMetricsConfig:
      driver: amqp
      topic: cloud1-metering

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

4.1.5. 部署 overcloud

使用所需的环境文件部署或更新 overcloud，以便收集数据并将其传输到服务遥测框架(STF)。

流程

1. 以 **stack** 用户身份登录 Red Hat OpenStack Platform (RHOSP) undercloud。
2. 提供身份验证文件：

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

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

3. 在 RHOSP director 部署中添加以下文件来配置数据收集和 AMQ Interconnect：
   - **ceilometer-write-qdr.yaml** 文件，以确保 Ceilometer 遥测和事件发送到 STF

- **qdr-edge-only.yaml** 文件，以确保消息总线被启用并连接到 STF 消息总线路由器
- **enable-stf.yaml** 环境文件，以确保正确配置了默认值
- **stf-connectors.yaml** 环境文件来定义到 STF 的连接

4. 部署 RHOSP overcloud：

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

4.1.6. 验证客户端安装

要验证 Service Telemetry Framework (STF)存储域的数据收集，请查询数据源以传送数据。要在 Red Hat OpenStack Platform (RHOSP)部署中验证单个节点，请使用 SSH 连接到控制台。

提示

某些遥测数据仅在 RHOSP 具有活跃工作负载时才可用。

流程

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

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

running
```

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

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

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

4. 返回到本地 AMQ Interconnect 的连接列表：

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

Connections
  id   host                                                container
```

```
role    dir  security                        authentication  tenant

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

有四个连接：

- 出站到 STF 的连接
- 来自 ceilometer 的入站连接
- 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。其他主机是与这个 AMQ Interconnect 的客户端连接的内部网络地址。

5. 为确保传递信息，列出链接，并在 **deliv** 列中查看用于传递消息的 **_edge** 地址：

```
$ sudo docker exec -it metrics_qdr qdstat --bus=172.17.1.44:5666 --links
Router Links
  type      dir  conn id  id    peer  class   addr               phs  cap  pri  undel  unsett  deliv
presett  psdrop  acc  rej  rel     mod  delay  rate

========================================================================================
========================================================================================
=====
  endpoint  out  1       5         local   _edge                250  0    0    0      2979926  0       0
0    0    2979926 0    0     0
  endpoint  in   1       6                                      250  0    0    0      0        0       0     0    0
0       0    0     0
  endpoint  in   1       7                                      250  0    0    0      0        0       0     0    0
0       0    0     0
  endpoint  out  1       8                                      250  0    0    0      0        0       0     0    0
0       0    0     0
  endpoint  in   1       9                                      250  0    0    0      0        0       0     0    0
0       0    0     0
  endpoint  out  1       10                                     250  0    0    0      911      911     0     0
0    0       0    911    0
  endpoint  in   1       11                                     250  0    0    0      0        911     0     0
0    0       0    0      0
  endpoint  out  12      32        local   temp.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. 要列出来自 RHOSP 节点到 STF 的地址，请连接到 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 default-interconnect-7458fd4d69-bgzfb -- qdstat --connections

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

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

========================================================================
========================================================================
=========================================
 5   10.129.0.110:48498  bridge-3f5                                      edge    in   no-security
anonymous-user          000:00:00:02  000:17:36:29
 6   10.129.0.111:43254  rcv[default-cloud1-ceil-meter-smartgateway-58f885c76d-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 **oc exec** 命令的每个地址：

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

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

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

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

4.2. 禁用用于服务 TELEMETRY FRAMEWORK 的 RED HAT OPENSTACK PLATFORM 服务

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

流程

1. 以 **stack** 用户身份登录到 RHOSP undercloud。
2. 提供身份验证文件：

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

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

3. 创建 **disable-stf.yaml** 环境文件：

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

4. 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。通过在列表的早期添加 **disable-stf.yaml**，其他环境文件可能会覆盖将禁用该服务的配置：

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

4.3. 部署到非标准网络拓扑

如果您的节点位于与默认 **InternalApi** 网络的独立网络中，您必须进行配置调整，以便 AMQ Interconnect 可以与 Service Telemetry Framework (STF)服务器实例传输数据。这个场景通常在 spine-leaf 或 DCN 拓扑中。有关 DCN 配置的更多信息，请参阅 Spine Leaf Networking 指南。

如果您将 STF 与 Red Hat OpenStack Platform (RHOSP) 13.0 搭配使用，并计划监控 Ceph、Block 或 Object Storage 节点，您必须进行与 spine-leaf 和 DCN 网络配置类似的配置更改。要监控 Ceph 节点，请使用 **CephStorageExtraConfig** 参数定义要加载到 AMQ Interconnect 和 collectd 配置文件中的网络接口。

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

同样，如果您的环境使用 Block 和 Object Storage 角色，则必须指定 **BlockStorageExtraConfig** 和 **ObjectStorageExtraConfig** 参数。

要部署自叶拓扑，您必须创建角色和网络，然后将这些网络分配到可用的角色。当您为 RHOSP 部署配置 STF 的数据收集和传输时，角色的默认网络为 **InternalApi**。对于 Ceph，块和对象存储角色是 **Storage**。由于 spine-leaf 配置可能会导致不同的网络被分配到不同的 Leaf 分组，且这些名称通常是唯一的，因此 RHOSP 环境文件的 **parameter_defaults** 部分中需要额外的配置。

流程

1. 记录每个 Leaf 角色都可用的网络。有关网络名称定义示例，请参阅 *Spine Leaf Networking* 指南中的创建网络 数据文件。有关创建 Leaf 分组（角色）并将网络分配给这些分组的更多信息，请参阅 *Spine Leaf Networking* 指南中的 角色数据文件。

2. 将以下配置示例添加到每个 leaf 角色的 **ExtraConfig** 部分。在本例中，**internal_api0** 是网络定义的 **name_lower** 参数中定义的值，是 **Compute0** leaf 角色所连接的网络。在本例中，0 的网络标识与 leaf 0 的 Compute 角色对应，它代表与默认内部 API 网络名称不同的值。
   对于 **Compute0** leaf 角色，指定额外的配置来执行 hiera lookup，以确定特定网络要分配给 collectd AMQP 主机参数的网络接口。对 AMQ Interconnect 监听器地址参数执行相同的配置。

   ```
   Compute0ExtraConfig:
     tripleo::profile::base::metrics::collectd::amqp_host: "%{hiera('internal_api0')}"
     tripleo::profile::base::metrics::qdr::listener_addr: "%{hiera('internal_api0')}"
   ```

   此示例配置位于 CephStorage leaf 角色上：

```
CephStorage0ExtraConfig:
  tripleo::profile::base::metrics::collectd::amqp_host: "%{hiera('storage0')}"
  tripleo::profile::base::metrics::qdr::listener_addr: "%{hiera('storage0')}"
```

4.4. 将 RED HAT OPENSTACK PLATFORM 容器镜像添加到 UNDERCLOUD

如果将容器镜像同步到本地 registry，您必须创建一个环境文件，并包含容器镜像的路径。

流程

1. 创建新环境文件，如 **container-images.yaml**，并插入以下容器镜像和路径：

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

   将 **<image-registry-path** > 替换为镜像 registry 的路径。

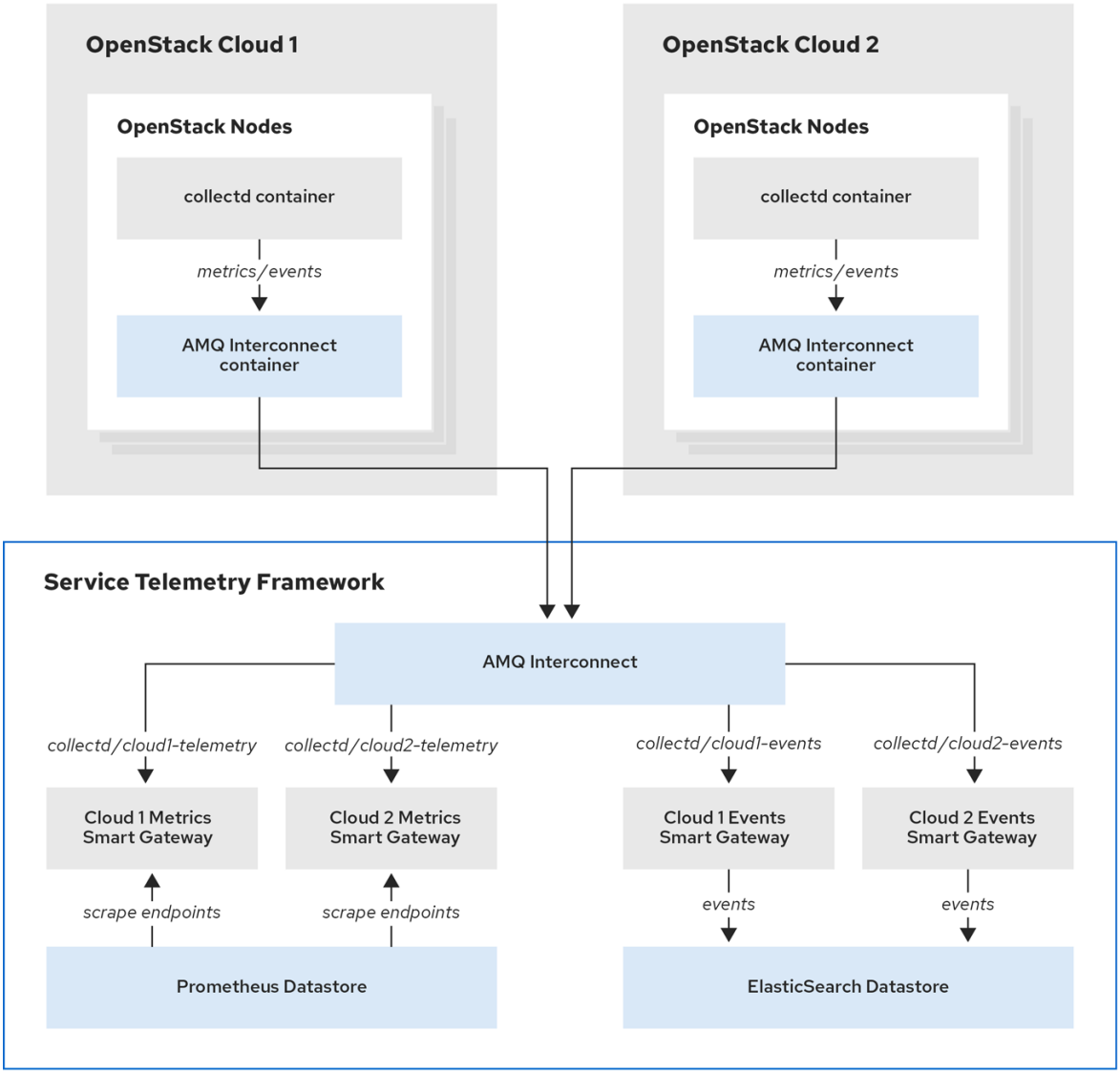
2. 要将 collectd 和 AMQ Interconnect 部署到 overcloud，请包含 **/usr/share/local-container-images/container-images.yaml** 环境文件，以便 Red Hat OpenStack Platform director 可以准备镜像。以下片段演示了如何包含此环境文件的示例：

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

4.5. 配置多个云

您可以将多个 Red Hat OpenStack Platform (RHOSP)云配置为以单个服务遥测框架(STF)实例为目标。当您配置多个云时，每个云都必须自行发送指标和事件。在 STF 部署中，智能网关实例侦听这些主题，以将信息保存到常见数据存储中。使用每个智能网关创建的元数据过滤在数据存储域中通过智能网关存储的数据过滤。

图 4.1. 两个 RHOSP 云连接到 STF



65_OpenStack_0120

要为多个云场景配置 RHOSP overcloud，请完成以下任务：

1. 规划要用于每个云的 AMQP 地址前缀。更多信息请参阅 第 4.5.1 节 “规划 AMQP 地址前缀”。
2. 为每一个云部署指标和事件消费者智能网关，以侦听对应的地址前缀。更多信息请参阅 第 4.5.2 节 “部署智能网关”。
3. 使用唯一域名配置每个云。更多信息请参阅 第 4.5.4 节 “设置唯一云域”。
4. 为 STF 创建基本配置。更多信息请参阅 第 4.1.3 节 “为 STF 创建基本配置”。
5. 配置每个云，使其指标和事件在正确的地址上发送到 STF。更多信息请参阅 第 4.5.5 节 “为多个云创建 Red Hat OpenStack Platform 环境文件”。

4.5.1. 规划 AMQP 地址前缀

默认情况下，Red Hat OpenStack Platform 节点通过两个数据收集器接收数据： collectd 和 Ceilometer。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 智能网关侦听用于监控数据的 AMQP 地址。要支持多个云并确定生成监控数据的云，请将每个云配置为将数据发送到唯一地址。为地址的第二部分添加云标识符前缀。以下列表显示了一些地址和标识符示例：

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

4.5.2. 部署智能网关

您必须为每个云部署每个数据收集类型的智能网关；一个用于 collectd 指标，一个用于 collectd 事件，一个用于 collectd 事件，一个用于 Ceilometer 指标，一个用于 Ceilometer 事件。配置每个智能网关，以侦听您为相应云定义的 AMQP 地址。要定义智能网关，请在 **ServiceTelemetry** 清单中配置 **cloud** 参数。

当您第一次部署 STF 时，会创建智能网关清单来为单个云定义初始智能网关。当您为多个云支持部署智能网关时，您可以为处理每个云指标和事件数据的数据收集类型部署多个智能网关。初始智能网关在 **cloud1** 中使用以下订阅地址定义：

collector	type	默认订阅地址
collectd	metrics	collectd/telemetry
collectd	事件	collectd/notify
Ceilometer	metrics	anycast/ceilometer/metering.sample
Ceilometer	事件	anycast/ceilometer/event.sample

先决条件

- 您已确定了云命名方案。有关确定您的命名方案的更多信息，请参阅 第 4.5.1 节 “规划 AMQP 地址前缀”。
- 您已创建了云对象列表。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 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 **配置添加一个 cloud** 参数：

警告

长云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，可以和 **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: ceilometer
        subscriptionAddress: anycast/ceilometer/cloud1-metering.sample
  - name: cloud2
    events:
      ...
```

4. 保存 ServiceTelemetry 对象。

5. 验证每个智能网关正在运行。这可能需要几分钟时间，具体取决于智能卡数量：

```
$ 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
```

4.5.3. 删除默认智能网关

为多个云配置 Service Telemetry Framework (STF)后，如果不再使用默认智能网关，可以删除默认智能网关。Service Telemetry Operator 可以移除创建的 **SmartGateway** 对象，但不再列在对象的 ServiceTelemetry **云** 列表中。要启用删除不是由 **clouds** 参数定义的 SmartGateway 对象，您必须在 **ServiceTelemetry** 清单中将 **cloudsRemoveOn** Mising 参数设置为 **true**。

提示

如果您不想部署任何智能网关，请使用 cloud **: [] 参数定义一个空云** 列表。

警告

cloudsRemoveOnMissing 参数会被默认禁用。如果启用 **cloudsRemoveOnMis** ing 参数，您可以删除当前命名空间中的手动创建的 **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 是否已删除智能卡。Operator 协调更改时可能需要几分钟时间：

```
$ oc get smartgateways
```

4.5.4. 设置唯一云域

要确保 AMQ 互连从 Red Hat OpenStack Platform (RHOSP)到 Service Telemetry Framework (STF)的连接是唯一的且没有冲突，请配置 **CloudDomain** 参数。

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. 将新环境文件添加到您的部署中。

其他资源

- 第 4.5.5 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*

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

要根据原始云标记流量，您必须创建具有特定云实例名称的配置。创建 **stf-connectors.yaml** 文件，并调整 **CeilometerQdrEventsConfig**、**CeilometerQdrMetricsConfig** 和 **CollectdAmqpInstances** 的值以匹配 AMQP 地址前缀方案。

先决条件

- 您已从 STF 部署的 AMQ Interconnect 检索了 CA 证书。更多信息请参阅 第 4.1.1 节 “从用于 overcloud 配置的服务 Telemetry Framework 获取 CA 证书”。
- 您已创建了云对象列表。有关为云参数创建内容的更多信息，请参阅 云配置参数。
- 您已获取 AMQ Interconnect 路由地址。更多信息请参阅 第 4.1.2 节 “检索 AMQ Interconnect 路由地址”。
- 您已为 STF 创建基本配置。更多信息请参阅 第 4.1.3 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 4.5.4 节 “设置唯一云域”。

流程

1. 以 **stack** 用户身份登录 Red Hat OpenStack Platform undercloud。
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 **CeilometerQdrEventsConfig**、**CeilometerQdrMetricsConfig** 和 **CollectdAmqpInstances** 主题值，以匹配此云部署的 AMQP 地址。
    - **resource_registry** 配置直接加载 collectd 服务，因为您不为多个云部署包含 **collectd-write-qdr.yaml** 环境文件。
    - 将 **MetricsQdrConnectors.host** 参数替换为您在 第 4.1.2 节 “检索 AMQ Interconnect 路由地址” 中检索的 **HOST/PORT** 的值。

- 将 **caCertFileContent** 参数替换为在 第 4.1.1 节 “从用于 overcloud 配置的服务 Telemetry Framework 获取 CA 证书” 中检索的内容。
- 设置 **CeilometerQdrEventsConfig.topic**，以定义 Ceilometer 事件的主题。这个值是 **anycast/ceilometer/cloud1-event.sample** 的地址格式。
- 设置 **CeilometerQdrMetricsConfig.topic**，以定义 Ceilometer 指标的主题。这个值是 **anycast/ceilometer/cloud1-metering.sample** 的地址格式。
- 设置 **CollectdAmqpInstances**，以定义 collectd 事件的主题。这个值是 **collectd/cloud1-notify** 的格式。
- 设置 **CollectdAmqpInstances**，以定义 collectd 指标的主题。这个值是 **collectd/cloud1-telemetry** 的格式。

stf-connectors.yaml

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

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

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

    CeilometerQdrEventsConfig:
      driver: amqp
      topic: cloud1-event

    CeilometerQdrMetricsConfig:
      driver: amqp
      topic: cloud1-metering

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

4. 确保 **stf-connectors.yaml** 文件中的命名约定与智能网关配置中的 **spec.bridge.amqpUrl** 字段一致。例如，将 **CeilometerQdrEventsConfig.topic** 字段配置为 **cloud1-event** 的值。

5. 提供身份验证文件：

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

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

6. 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** 参数配置加载 collectd 服务。

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

7. 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 4.1.6 节 “验证客户端安装”。

4.5.6. 从多个云查询指标数据

Prometheus 中存储的数据具有 **服务** 标签，它从其中提取的智能卡。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 使用服务 TELEMETRY FRAMEWORK 的操作功能

您可以使用以下操作功能为服务遥测框架(STF)提供额外功能：

- 配置仪表板
- 配置指标保留时间
- 配置警报
- 配置 SNMP 陷阱
- 配置高可用性
- 配置临时存储
- 配置备用可观察性策略

5.1. SERVICE TELEMETRY FRAMEWORK 中的仪表板

使用第三方应用 Grafana 来视觉化各个主机节点的数据流 collectd 和 Ceilometer 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 “为服务 Telemetry Framework 部署 Red Hat OpenStack Platform overcloud”。

5.1.1. 配置 Grafana 来托管仪表板

Grafana 不包含在默认的 Service Telemetry Framework (STF)部署中，因此您必须从 OperatorHub.io 部署 Grafana Operator。当您使用 Service Telemetry Operator 部署 Grafana 时，它会生成 Grafana 实例，以及本地 STF 部署的默认数据源配置。

流程

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

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

3. 部署 Grafana Operator：

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

4. 验证 Operator 是否已成功启动。在命令输出中，如果 **PHASE** 列的值为 **Succeeded**，Operator 可以成功启动：

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

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

5. 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:7'
```

6. 验证 Grafana 实例是否已部署：

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

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

7. 验证 Grafana 数据源是否已正确安装：

```
$ oc get grafanadatasources

NAME                  AGE
default-datasources     20h
```

8. 验证 Grafana 路由是否存在：

```
$ oc get route grafana-route

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

5.1.2. 获取和设置 Grafana 登录凭证

在启用 Grafana 时，服务 Telemetry Framework (STF)会设置默认登录凭证。您可以在 **ServiceTelemetry** 对象中覆盖凭证。

流程

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

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

3. 从 STF 对象检索默认用户名和密码：

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

4. 要通过 ServiceTelemetry 对象修改 Grafana 管理员用户名和密码的默认值，请使用 **graphing.grafana.adminUser** 和 **graphing.grafana.adminPassword** 参数。

5.2. SERVICE TELEMETRY FRAMEWORK 中的指标保留时间

Service Telemetry Framework (STF)中存储的指标的默认保留时间为 24 小时，为警报开发提供充足的趋势。

对于长期存储，请使用为长期数据保留而设计的系统，例如 Thanos。

其他资源

- 要调整 STF 的额外指标保留时间，请参阅 第 5.2.1 节 “编辑服务 Telemetry Framework 中的指标保留时间”。
- 有关 Prometheus 数据存储和估算存储空间的建议，请参考 https://prometheus.io/docs/prometheus/latest/storage/#operational-aspects
- 有关 Thanos 的更多信息，请参阅 https://thanos.io/

5.2.1. 编辑服务 Telemetry Framework 中的指标保留时间

您可以调整服务遥测框架(STF)以获得额外的指标保留时间。

流程

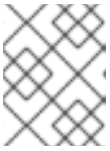
1. 登录 Red Hat OpenShift Container Platform。
2. 进入 service-telemetry 命名空间：

```
$ 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返回速度慢。

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

5. 保存您的更改，并关闭对象。

其他资源

- 有关指标保留时间的更多信息，请参阅 第 5.2 节 “Service Telemetry Framework 中的指标保留时间”。

5.3. SERVICE TELEMETRY FRAMEWORK 中的警报

您可以在 Alertmanager 中在 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、on-call 通知系统或聊天平台发送通知。

要创建警报，请完成以下步骤：

1. 在 Prometheus 中创建警报规则。更多信息请参阅 第 5.3.1 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 警报或通知的更多信息，请参阅 https://prometheus.io/docs/alerting/overview/

要查看可用于 Service Telemetry Framework (STF)的警报集合的示例，请参阅 https://github.com/infrawatch/service-telemetry-operator/tree/master/deploy/alerts

5.3.1. 在 Prometheus 中创建警报规则

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

流程

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

要更改规则，请编辑 **expr** 参数的值。

4. 要验证 Operator 是否将规则加载到 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 --user "internal:$(oc get secret default-prometheus-htpasswd -ogo-template='{{ .data.password | base64decode }}')" https://$(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

5.3.2. 配置自定义警报

您可以在您在 第 5.3.1 节 “在 Prometheus 中创建警报规则” 中创建的 **PrometheusRule** 对象中添加自定义警报。

流程

1. 使用 **oc edit** 命令：

```
$ oc edit prometheusrules 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

5.3.3. 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 管理。默认情况下，Service Telemetry Framework (STF)部署会导致任何接收器的基本配置：

```
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** 参数传递给 Service Telemetry Operator，以生成由 Prometheus Operator 管理的更新 secret。

注意

如果 **alertmanagerConfigManifest** 包含用于构造发送警报的标题和文本的自定义模板，请使用 base64 编码的配置部署 **alertmanagerConfigManifest** 的内容。更多信息请参阅 第 5.3.4 节 “在 Alertmanager 中创建带有模板模板的警报路由”。

流程

1. 登录 Red Hat OpenShift Container Platform。

2. 进入 **service-telemetry** 命名空间：

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

3. 为您的 STF 部署编辑 **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。要验证是否正确填充了更改，请更改值，返回 **alertmanager-default** secret，然后验证新值是否已加载到内存中。例如，将 **global.resolve_timeout** 参数的值从 **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 **curl** 命令，以检索 status 和 **configYAML** 内容，并验证提供的配置是否与 Alertmanager 中的配置匹配：

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

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

7. 验证 **configYAML** 字段是否包含您预期的更改。

8. 要清理环境，请删除 **curl** pod：

```
$ oc delete pod curl

pod "curl" deleted
```

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。

5.3.4. 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 管理。默认情况下，Service Telemetry Framework (STF)部署会导致任何接收器的基本配置：

```
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，请使用 base64 编码的配置部署 **alertmanagerConfigManifest** 的内容。

流程

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

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

3. 为您的 STF 部署编辑 **ServiceTelemetry** 对象：

```
$ oc edit stf default
```

4. 要使用 STF 部署自定义 Alertmanager 路由，您必须将 **alertmanagerConfigManifest** 参数传递给 Service Telemetry Operator，这会导致 Prometheus Operator 管理的更新 secret：

```
apiVersion: infra.watch/v1beta1
kind: ServiceTelemetry
metadata:
  name: default
  namespace: service-telemetry
spec:
  backends:
    metrics:
      prometheus:
        enabled: true
  alertmanagerConfigManifest: |
    apiVersion: v1
    kind: Secret
    metadata:
      name: 'alertmanager-default'
      namespace: 'service-telemetry'
    type: Opaque
    data:
      alertmanager.yaml:
Z2xvYmFsOgogIHJlc29sdmVfdGltZW91dDogMTBtCiAgc2xhY2tfYXBpX3VybDogPHNsYWNrX
2FwaV91cmw+CnJlY2VpdmVyczoKICAtIG5hbWU6IHNsYWNrCiAgICBzbGFja19jb25maWdzO
ogICAgLSBjaGFubmVsOiAjc3RmLWFsZXJ0cwogICAgICB0aXRsZTogfC0KICAgICAgICAuLi
4KICAgICAgdGV4dDogPi0KICAgICAgICAuLi4Kcm91dGU6CiAgZ3JvdXBfYnk6IFsnam9iJ10KI
CBncm91cF93YWl0OiAzMHMKICBncm91cF9pbnRlcnZhbDogNW0KICByZXBlYXRfaW50ZXJ2
YWw6IDEyaAogIHJlY2VpdmVyOiAnc2xhY2snCg==
```

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
  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 **curl** 命令，以检索 status 和 **configYAML** 内容，并验证提供的配置是否与 Alertmanager 中的配置匹配：

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

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

7. 验证 **configYAML** 字段是否包含您预期的更改。

8. 要清理环境，请删除 **curl** pod：

```
$ oc delete pod curl

pod "curl" deleted
```

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。

5.4. 配置 SNMP 陷阱

您可以将服务遥测框架(STF)与通过 SNMP 陷阱接收通知的现有基础架构监控平台集成。要启用 SNMP 陷阱，修改 **ServiceTelemetry** 对象并配置 **snmpTraps** 参数。

有关配置警报的详情请参考 第 5.3 节 “Service Telemetry Framework 中的警报”。

先决条件

- 了解您要发送警报的 SNMP 陷阱接收器的 IP 地址或主机名

流程

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

```
$ oc edit stf default
```

2. 设置 alert **.alertmanager.receivers.snmpTraps** 参数：

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

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

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

5.5. 高可用性

借助高可用性，服务遥测框架(STF)可以从组件服务中的故障快速恢复。虽然 Red Hat OpenShift Container Platform 会在节点可用时重启一个失败的 pod，但这个恢复过程可能需要一分钟以上的时间，其中事件和指标会丢失。高可用性配置包含 STF 组件的多个副本，可将恢复时间缩短为大约 2 秒。为了防止出现 Red Hat OpenShift Container Platform 节点故障，请将 STF 部署到具有三个或更多节点的 Red Hat OpenShift Container Platform 集群。

警告

STF 尚未是一个完全容错的系统。无法保证在恢复期间提供指标和事件。

启用高可用性有以下影响：

- 三个 Elasticsearch Pod 运行，而不是默认运行。
- 以下组件运行两个 pod，而不是默认 pod：
  - AMQ Interconnect
  - Alertmanager
  - Prometheus
  - 事件智能网关
  - 指标智能网关
- 在这些服务中丢失的 pod 恢复时间会减少到大约 2 秒。

5.5.1. 配置高可用性

要配置服务 Telemetry Framework (STF)以实现高可用性，请在 Red Hat OpenShift Container Platform 中的 ServiceTelemetry 对象中添加 **highAvailability.enabled: true**。您可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 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. 在 **spec** 部分添加 **highAvailability.enabled: true** ：

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

5. 保存您的更改，并关闭对象。

5.6. 临时存储

您可以使用临时存储来运行服务 Telemetry Framework (STF)，而无需将数据存储在 Red Hat OpenShift Container Platform 集群中。

警告

如果使用临时存储，当 pod 重启、更新或重新调度到另一个节点上时，可能会遇到数据丢失。仅对开发或测试使用临时存储，而不适用于生产环境。

5.6.1. 配置临时存储

要为临时存储配置 STF 组件，请将 **...storage.strategy: ephemeral** 添加到对应的参数。例如，要为 Prometheus 后端启用临时存储，请设置 **backends.metrics.prometheus.storage.strategy: ephemeral**。支持临时存储配置的组件包括 **alerting.alertmanager**, **backends.metrics.prometheus**, 和 **backends.events.elasticsearch**。您可以在安装时添加临时存储配置，或者已部署了 STF，请完成以下步骤：

流程

1. 登录 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 **...storage.strategy: ephemeral** 参数添加到相关组件的 **spec** 部分：

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

5. 保存您的更改，并关闭对象。

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

服务遥测框架(STF)不包括存储后端和警报工具。STF 使用社区操作器来部署 Prometheus、Alertmanager、Grafana 和 Elasticsearch。STF 向这些社区运营商提出请求，以创建配置为与 STF 配合使用的每个应用程序实例。

除了使用 Service Telemetry Operator 创建自定义资源请求外，您可以使用您自己的部署这些应用程序或其他兼容的应用程序，并提取指标智能网关以传送到您自己的 Prometheus 兼容系统进行遥测存储。如果您将 observability 策略设置为使用替代后端，则 STF 不需要持久性或临时存储。

5.7.1. 配置备用可观察性策略

要将 STF 配置为跳过存储、视觉化和警报后端的部署，请将 **observabilityStrategy: none** 添加到 ServiceTelemetry spec。在这个模式中，只有 AMQ Interconnect 路由器和指标智能网关会被部署，您必须配置外部 Prometheus 兼容系统，以从 STF 智能网关收集指标。

注意

目前，只有将 **observabilityStrategy** 设置为 **none** 时支持指标。不会部署事件智能网关。

流程

1. 在 **spec** 参数中创建一个带有属性 **observabilityStrategy: none** 的 **ServiceTelemetry** 对象。清单显示的结果是采用 STF 的默认部署，适用于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. 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-meter-smartgateway-59c845d65b-gzhcs   3/3     Running   0          132m
default-cloud1-coll-meter-smartgateway-75bbd948b9-d5phm   3/3     Running   0          132m
default-cloud1-sens-meter-smartgateway-7fdbb57b6d-dh2g9   3/3     Running   0          132m
default-interconnect-668d5bbcd6-57b2l               1/1     Running   0          132m
interconnect-operator-b8f5bb647-tlp5t               1/1     Running   0          47h
service-telemetry-operator-566b9dd695-wkvjq         1/1     Running   0          156m
smart-gateway-operator-58d77dcf7-6xsq7              1/1     Running   0          47h
```

其他资源

有关配置其他云或更改支持的收集器集合的更多信息，请参阅 第 4.5.2 节 “部署智能网关”

第 6 章 续订 AMQ INTERCONNECT 证书

当证书过期时，您必须定期更新保护 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。

6.1. 检查已过期的 AMQ INTERCONNECT CA 证书

当 CA 证书过期 AMQ Interconnect 连接时，但如果它们中断，则无法重新连接。最终，您会发现 Red Hat OpenStack Platform (RHOSP)路由器中的一些或全部连接失败，显示两端的错误，以及您的 CA 证书中的到期（或"Not After"字段将位于过去。

流程

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

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

3. 检查部分或所有路由器连接是否已失败：

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

4. 在 Red Hat OpenShift Container Platform 托管的 AMQ Interconnect 日志中检查这个错误：

```
$ 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. 在连接失败的节点的 RHOSP-hosted AMQ Interconnect 日志中检查这个错误：

```
$ 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
```

6.2. 更新 AMQ INTERCONNECT CA 证书

要更新 AMQ Interconnect 证书，您需要将其从 Red Hat OpenShift Container Platform 导出，并将其复制到 Red Hat OpenStack Platform (RHOSP)节点。

流程

1. 登录 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 （更多信息，请参阅 第 4.1.4 节 “为 overcloud 配置 STF 连接”）

注意

执行以下步骤后，您不需要执行 overcloud 部署。我们只编辑 stf-connectors.yaml 文件，以确保将来的部署不会覆盖新的 CA 证书。

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

第 7 章 从 RED HAT OPENSHIFT CONTAINER PLATFORM 环境中删除 SERVICE TELEMETRY FRAMEWORK

如果不再需要 STF 功能，请从 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. 删除目录源。
3. 删除 cert-manager Operator。

7.1. 删除命名空间

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

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.
```

7.2. 删除 CATALOGSOURCE

如果您不希望再次安装 Service Telemetry Framework (STF)，请删除 CatalogSource。当您删除 CatalogSource 时，与 STF 相关的 PackageManifests 将自动从 Operator Lifecycle Manager 目录中移除。

流程

1. 如果您在安装过程中启用了 OperatorHub.io Community Catalog Source，且不再需要这个目录源，请删除它：

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

其他资源

有关 OperatorHub.io 社区目录源的更多信息，请参阅 第 3.1 节 “在 Red Hat OpenShift Container Platform 环境中部署 Service Telemetry Framework”。

7.3. 删除 CERT-MANAGER OPERATOR

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

流程

1. 从 **openshift-cert-manager-operator** 命名空间中删除订阅：

```
$ oc delete --namespace=openshift-cert-manager-operator subscription openshift-cert-manager-operator

subscription.operators.coreos.com "openshift-cert-manager-operator" deleted
```

2. 检索安装的 ClusterServiceVersion 的版本号：

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

输出示例：

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

3. 从 **openshift-cert-manager-operator** 命名空间中删除 ClusterServiceVersion:

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

输出示例：

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

4. 获取 Operator 提供的 CustomResourceDefinitions 的当前列表，以便在删除 ClusterServiceVersion 后删除它们：

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

输出示例：

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

5. 删除与 cert-manager Operator 相关的 CustomResourceDefinitions：

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

输出示例：

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

6. 删除由 cert-manager Operator 拥有的命名空间：

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

输出示例：

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

附加信息

- 从集群中删除 Operator。