# Red Hat OpenStack Platform 13

## 서비스 원격 분석 프레임워크 1.5

Service Telemetry Framework 1.5 설치 및 배포

Last Updated: 2023-11-03

# Red Hat OpenStack Platform 13 서비스 원격 분석 프레임워크 1.5

Service Telemetry Framework 1.5 설치 및 배포

OpenStack Team
rhos-docs@redhat.com

## 법적 공지



## 초록

핵심 구성 요소를 설치하고 Service Telemetry Framework 1.5를 배포합니다.

# 차례

**보다 포괄적 수용을 위한 오픈 소스 용어 교체** ..... 3

**RED HAT 문서에 관한 피드백 제공** ..... 4

**1장. SERVICE TELEMETRY FRAMEWORK 1.5 소개** ..... 5
1.1. SERVICE TELEMETRY FRAMEWORK 지원 5
1.2. 서비스 TELEMETRY FRAMEWORK 아키텍처 5
1.3. RED HAT OPENSHIFT CONTAINER PLATFORM의 설치 크기 8

**2장. SERVICE TELEMETRY FRAMEWORK를 위한 RED HAT OPENSHIFT CONTAINER PLATFORM 환경 준비** 9
2.1. SERVICE TELEMETRY FRAMEWORK의 관찰 전략 9
2.2. 영구 볼륨 9
2.3. 리소스 할당 10
2.4. SERVICE TELEMETRY FRAMEWORK에 대한 네트워크 고려 사항 10

**3장. SERVICE TELEMETRY FRAMEWORK의 핵심 구성 요소 설치** ..... 11
3.1. RED HAT OPENSHIFT CONTAINER PLATFORM 환경에 SERVICE TELEMETRY FRAMEWORK 배포 11
3.2. RED HAT OPENSHIFT CONTAINER PLATFORM에서 SERVICETELEMETRY 오브젝트 생성 15
3.3. STF 구성 요소의 사용자 인터페이스에 액세스 22
3.4. 대체 관찰 전략 구성 23

**4장. SERVICE TELEMETRY FRAMEWORK용 RED HAT OPENSTACK PLATFORM 구성** ..... 25
4.1. SERVICE TELEMETRY FRAMEWORK용 RED HAT OPENSTACK PLATFORM 오버클라우드 배포 25
4.2. SERVICE TELEMETRY FRAMEWORK와 함께 사용되는 RED HAT OPENSTACK PLATFORM 서비스 비활성화 33
4.3. 비표준 네트워크 토폴로지에 배포 34
4.4. 언더클라우드에 RED HAT OPENSTACK PLATFORM 컨테이너 이미지 추가 36
4.5. 여러 클라우드 구성 36

**5장. SERVICE TELEMETRY FRAMEWORK의 운영 기능 사용** ..... 48
5.1. SERVICE TELEMETRY FRAMEWORK의 대시보드 48
5.2. SERVICE TELEMETRY FRAMEWORK의 지표 보존 기간 51
5.3. SERVICE TELEMETRY FRAMEWORK의 경고 52
5.4. SNMP 트랩 구성 61
5.5. 고가용성 61
5.6. 임시 스토리지 63
5.7. SERVICE TELEMETRY FRAMEWORK의 관찰 전략 65

**6장. AMQ INTERCONNECT 인증서 업데이트** ..... 67
6.1. 만료된 AMQ INTERCONNECT CA 인증서 확인 67
6.2. AMQ INTERCONNECT CA 인증서 업데이트 68

**7장. RED HAT OPENSHIFT CONTAINER PLATFORM 환경에서 SERVICE TELEMETRY FRAMEWORK 제거** 70
7.1. 네임스페이스 삭제 70
7.2. CATALOGSOURCE 제거 70
7.3. CERT-MANAGER OPERATOR 제거 71

# 보다 포괄적 수용을 위한 오픈 소스 용어 교체

Red Hat은 코드, 문서, 웹 속성에서 문제가 있는 용어를 교체하기 위해 최선을 다하고 있습니다. 먼저 마스터(master), 슬레이브(slave), 블랙리스트(blacklist), 화이트리스트(whitelist) 등 네 가지 용어를 교체하고 있습니다. 이러한 변경 작업은 작업 범위가 크므로 향후 여러 릴리스에 걸쳐 점차 구현할 예정입니다. 자세한 내용은 CTO Chris Wright의 메시지 를 참조하십시오.

# RED HAT 문서에 관한 피드백 제공

문서 개선을 위한 의견을 보내 주십시오. Red Hat이 어떻게 개선할 수 있는지 알려주십시오.

### DDF(직접 문서 피드백) 기능 사용

특정 문장, 단락 또는 코드 블록에 대한 직접 의견에 대한 **피드백 추가** DDF 함수를 사용합니다.

1. *Multi-page HTML* 형식으로 문서를 봅니다.
2. 문서의 오른쪽 상단에 **피드백** 버튼이 표시되는지 확인합니다.
3. 주석하려는 텍스트 부분을 강조 표시합니다.
4. **피드백 추가를** 클릭합니다.
5. 댓글을 사용하여 **피드백 추가** 필드를 작성합니다.
6. 선택 사항: 문제 해결을 위해 문서 팀에 문의할 수 있도록 이메일 주소를 추가하십시오.
7. **제출을** 클릭합니다.

# 1장. SERVICE TELEMETRY FRAMEWORK 1.5 소개

STF(Service Telemetry Framework)는 RHOSP(Red Hat OpenStack Platform) 또는 타사 노드에서 모니터링 데이터를 수집합니다. STF를 사용하여 다음 작업을 수행할 수 있습니다.

- 기록 정보에 대한 모니터링 데이터를 저장하거나 보관합니다.
- 대시보드에서 그래픽으로 모니터링 데이터를 봅니다.
- 모니터링 데이터를 사용하여 경고 또는 경고를 트리거합니다.

모니터링 데이터는 메트릭 또는 이벤트일 수 있습니다.

**메트릭**

애플리케이션 또는 시스템의 숫자 측정입니다.

**이벤트**

시스템에서 발생하는 비독점 및 개별 발생

STF의 구성 요소는 데이터 전송을 위해 메시지 버스를 사용합니다. 데이터를 수신하고 저장하는 다른 모듈식 구성 요소는 Red Hat OpenShift Container Platform에서 컨테이너로 배포됩니다.

**중요**

STF는 Red Hat OpenShift Container Platform 버전 4.10과 호환됩니다.

**추가 리소스**

- Red Hat OpenShift Container Platform 배포 방법에 대한 자세한 내용은 Red Hat OpenShift Container Platform 제품 설명서를 참조하십시오.
- 클라우드 플랫폼 또는 베어 메탈에 Red Hat OpenShift Container Platform을 설치할 수 있습니다. STF 성능 및 확장에 대한 자세한 내용은 https://access.redhat.com/articles/4907241 를 참조하십시오.
- 베어메탈 또는 기타 지원되는 클라우드 플랫폼에 Red Hat OpenShift Container Platform을 설치할 수 있습니다. Red Hat OpenShift Container Platform 설치에 대한 자세한 내용은 OpenShift Container Platform 4.10 문서를 참조하십시오.

## 1.1. SERVICE TELEMETRY FRAMEWORK 지원

Red Hat은 두 가지 최신 버전의 STF(Service Telemetry Framework)를 지원합니다. 이전 버전은 지원되지 않습니다. 자세한 내용은 Service Telemetry Framework 지원 버전 매트릭스 를 참조하십시오.

Red Hat은 AMQ Interconnect, Service Telemetry Operator 및 Smart Gateway Operator를 포함한 핵심 Operator 및 워크로드를 지원합니다. Red Hat은 Elasticsearch, Prometheus, Alertmanager, Grafana 및 해당 Operator와 같은 커뮤니티 Operator 또는 워크로드 구성 요소를 지원하지 않습니다.

완전히 연결된 네트워크 환경에서만 STF를 배포할 수 있습니다. Red Hat OpenShift Container Platform 관련 환경 또는 네트워크 프록시 환경에 STF를 배포할 수 없습니다.

## 1.2. 서비스 TELEMETRY FRAMEWORK 아키텍처

STF(Service Telemetry Framework)는 클라이언트-서버 아키텍처를 사용하며 RHOSP(Red Hat OpenStack Platform)는 클라이언트이고 Red Hat OpenShift Container Platform은 서버입니다.

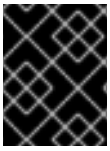
STF는 다음 구성 요소로 구성됩니다.

- 데이터 수집
  - collectd: 인프라 메트릭 및 이벤트를 수집합니다.
  - Ceilometer: RHOSP 지표 및 이벤트를 수집합니다.
- 전송
  - AMQ Interconnect: 스토리지를 위해 메트릭을 STF로 전송할 수 있는 빠르고 안정적인 데이터 전송을 제공하는 AMQP 1.x 호환 메시징 버스입니다.
  - Smart Gateway: AMQP 1.x 버스에서 지표 및 이벤트를 가져와서 Elasticsearch 또는 Prometheus에 전달하는 Golang 애플리케이션입니다.
- 데이터 스토리지
  - Prometheus: Smart Gateway에서 수신된 STF 지표를 저장하는 시계열 데이터 스토리지입니다.
  - Elasticsearch: Smart Gateway에서 수신된 STF 이벤트를 저장하는 이벤트 데이터 스토리지입니다.
- 가해진
  - Alertmanager: Prometheus 경고 규칙을 사용하여 경고를 관리하는 경고 툴입니다.
  - Grafana: 데이터를 쿼리, 시각화 및 탐색하는 데 사용할 수 있는 시각화 및 분석 애플리케이션입니다.

다음 표에서는 클라이언트 및 서버 구성 요소의 애플리케이션에 대해 설명합니다.

**표 1.1. STF의 클라이언트 및 서버 구성 요소**

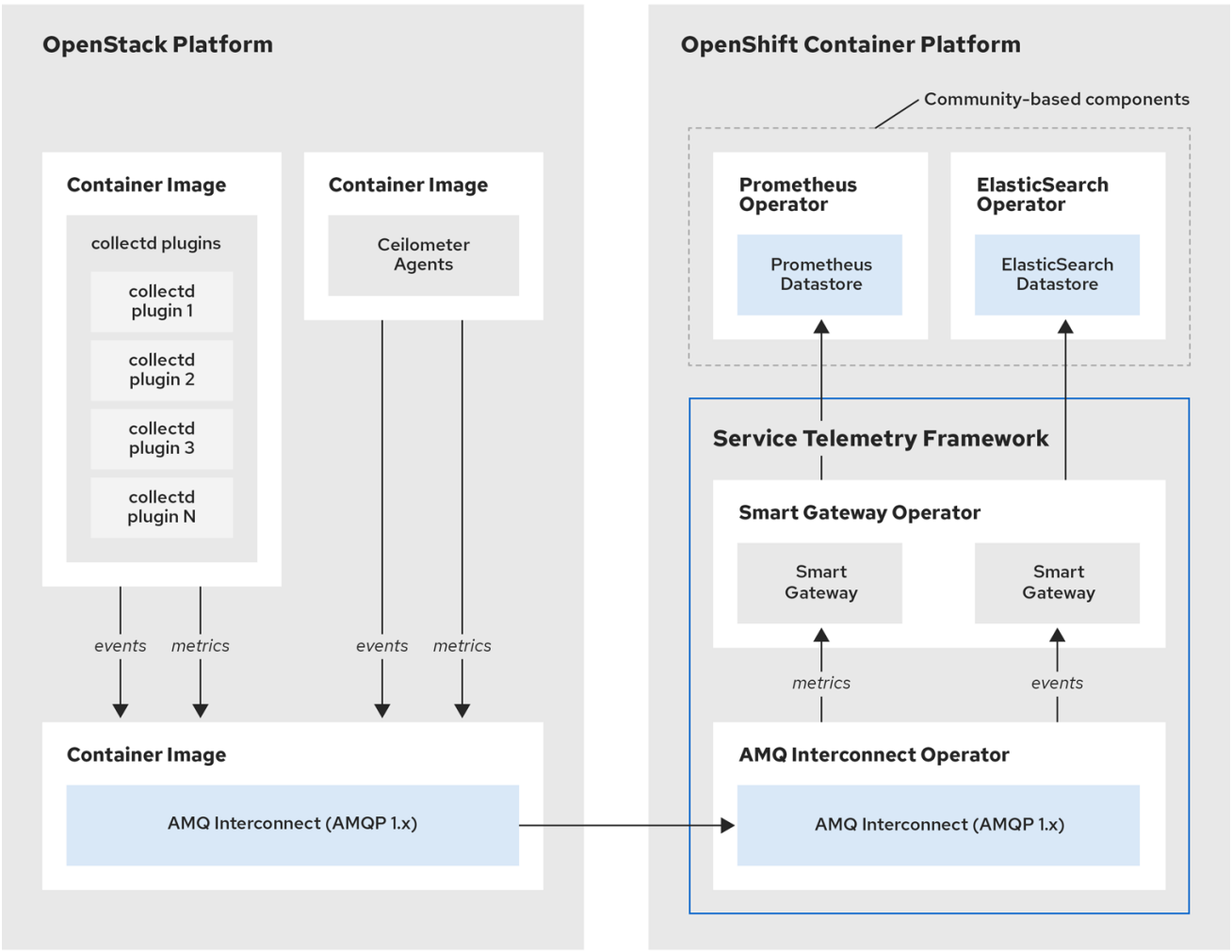
| 구성 요소 | 클라이언트 | 서버 |
|---|---|---|
| AMQP 1.x 호환 메시징 버스 | 제공됨 | 제공됨 |
| Smart Gateway | 제공되지 않음 | 제공됨 |
| Prometheus | 제공되지 않음 | 제공됨 |
| Elasticsearch | 제공되지 않음 | 제공됨 |
| collectd | 제공됨 | 제공되지 않음 |
| Ceilometer | 제공됨 | 제공되지 않음 |



**중요**

모니터링 플랫폼이 클라우드와 운영 문제를 보고할 수 있도록 하려면 모니터링 중인 동일한 인프라에 STF를 설치하지 마십시오.

**그림 1.1. Service Telemetry Framework 아키텍처 개요**



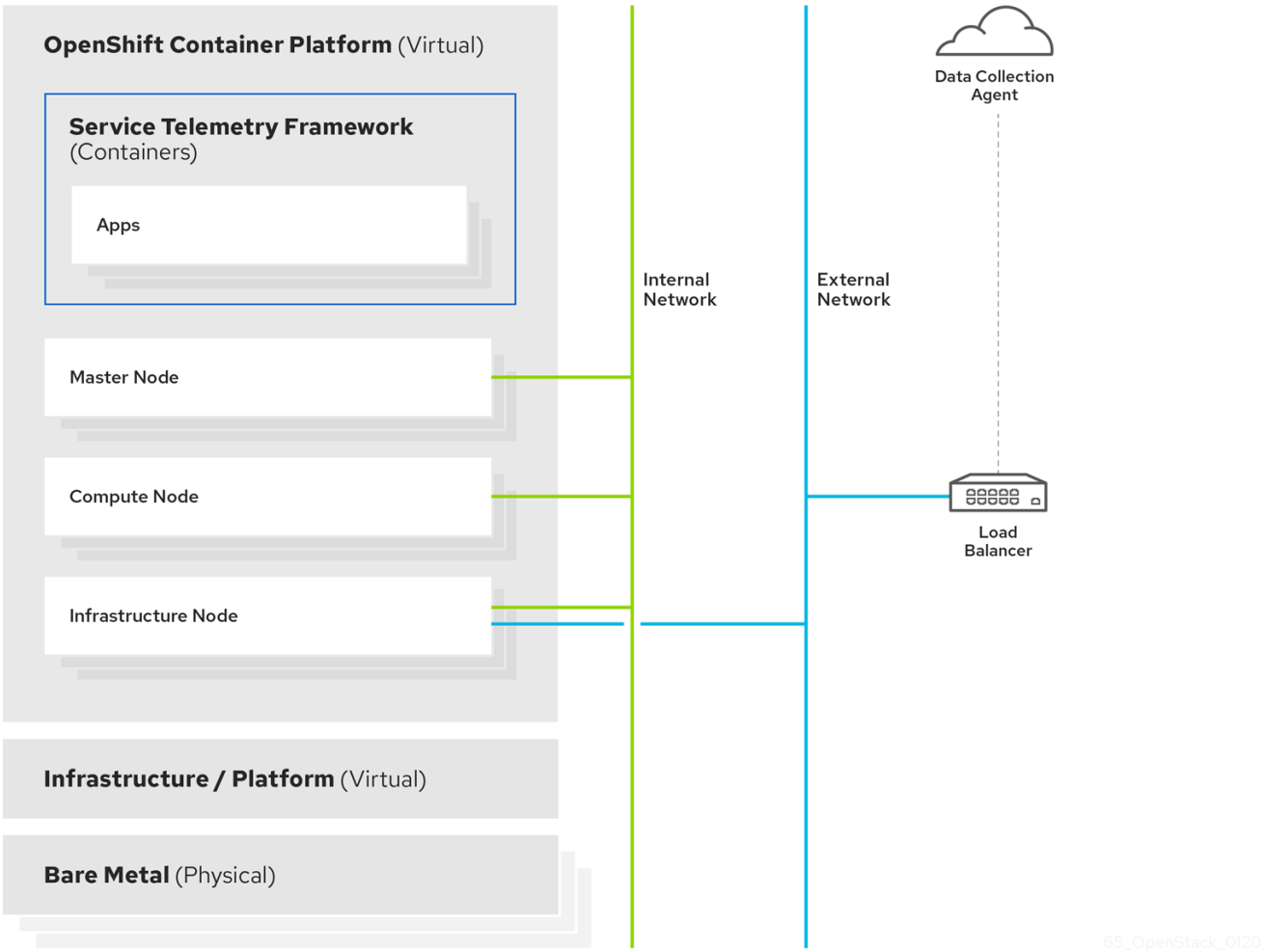

클라이언트 측 지표의 경우 collectd는 프로젝트 데이터가 없는 인프라 지표를 제공하며, Ceilometer는 프로젝트 또는 사용자 워크로드를 기반으로 RHOSP 플랫폼 데이터를 제공합니다. Ceilometer와 collectd 모두 AMQ Interconnect 전송을 사용하여 Prometheus에 데이터를 제공하여 메시지 버스를 통해 데이터를 전달합니다. 서버 측에서 Smart Gateway라는 Golang 애플리케이션은 버스에서 데이터 스트림을 가져와 Prometheus의 로컬 스크랩 끝점으로 표시합니다.

이벤트를 수집하고 저장하려면 AMQ Interconnect 전송을 사용하여 collectd 및 Ceilometer가 이벤트 데이터를 서버에 제공합니다. 또 다른 Smart Gateway는 Elasticsearch 데이터 저장소에 데이터를 씁니다.

서버 측 STF 모니터링 인프라는 다음 계층으로 구성됩니다.

- 서비스 원격 분석 프레임워크 1.5
- Red Hat OpenShift Container Platform 4.10
- 인프라 플랫폼

그림 1.2. 서버 측 STF 모니터링 인프라




## 1.3. RED HAT OPENSHIFT CONTAINER PLATFORM의 설치 크기

Red Hat OpenShift Container Platform 설치 크기는 다음과 같은 요인에 따라 다릅니다.

- 선택하는 인프라입니다.
- 모니터링할 노드 수입니다.
- 수집할 지표 수입니다.
- 메트릭의 해상도입니다.
- 데이터를 저장할 기간입니다.

STF(Service Telemetry Framework) 설치는 기존 Red Hat OpenShift Container Platform 환경에 따라 다릅니다.

baremetal에 Red Hat OpenShift Container Platform을 설치할 때 최소 리소스 요구 사항에 대한 자세한 내용은 *베어 메탈에 클러스터 설치* 가이드의 최소 리소스 요구 사항을 참조하십시오. 설치할 수 있는 다양한 퍼블릭 및 프라이빗 클라우드 플랫폼의 설치 요구 사항은 클라우드 플랫폼을 선택한 설치 설명서를 참조하십시오.

# 2장. SERVICE TELEMETRY FRAMEWORK를 위한 RED HAT OPENSHIFT CONTAINER PLATFORM 환경 준비

Red Hat OpenShift Container Platform 환경을 STF(Service Telemetry Framework)용으로 준비하려면 영구 스토리지, 적절한 리소스, 이벤트 스토리지 및 네트워크 고려 사항을 계획해야 합니다.

- Red Hat OpenShift Container Platform 클러스터에 프로덕션 수준의 배포를 위해 영구 스토리지를 사용할 수 있는지 확인합니다. 자세한 내용은 2.2절. "영구 볼륨"의 내용을 참조하십시오.
- Operator 및 애플리케이션 컨테이너를 실행하기에 충분한 리소스를 사용할 수 있는지 확인합니다. 자세한 내용은 2.3절. "리소스 할당"의 내용을 참조하십시오.
- 완전히 연결된 네트워크 환경이 있는지 확인합니다. 자세한 내용은 2.4절. "Service Telemetry Framework에 대한 네트워크 고려 사항"의 내용을 참조하십시오.

## 2.1. SERVICE TELEMETRY FRAMEWORK의 관찰 전략

STF(Service Telemetry Framework)에는 스토리지 백엔드 및 경고 툴이 포함되어 있지 않습니다. STF는 커뮤니티 Operator를 사용하여 Prometheus, Alertmanager, Grafana 및 Elasticsearch를 배포합니다. STF는 이러한 커뮤니티 운영자가 STF를 사용하도록 구성된 각 애플리케이션의 인스턴스를 생성하도록 요청합니다.

Service Telemetry Operator가 사용자 정의 리소스 요청을 생성하는 대신 이러한 애플리케이션 또는 기타 호환 애플리케이션에 대한 자체 배포를 사용하고 지표 Smart Gateway를 스크랩하여 Telemetry 스토리지를 위해 자체 Prometheus 호환 시스템으로 전달할 수 있습니다. 대신 대체 백엔드를 사용하도록 관찰 전략을 설정하면 STF에는 영구 또는 임시 스토리지가 필요하지 않습니다.

## 2.2. 영구 볼륨

Service Telemetry Framework(STF)는 Red Hat OpenShift Container Platform에서 영구 스토리지를 사용하여 영구 볼륨을 요청하여 Prometheus 및 Elasticsearch가 메트릭 및 이벤트를 저장할 수 있도록 합니다.

Service Telemetry Operator를 통해 영구 스토리지를 활성화하면 STF 배포에 요청된 PVC(영구 볼륨 클레임)에 RWO(ReadWriteOnce) 액세스 모드가 생성됩니다. 환경에 사전 프로비저닝된 영구 볼륨이 포함된 경우 Red Hat OpenShift Container Platform 기본 구성된 **storageClass** 에서 RWO 볼륨을 사용할 수 있는지 확인합니다.

**추가 리소스**

- Red Hat OpenShift Container Platform의 영구 스토리지 구성에 대한 자세한 내용은 영구 스토리지 이해를 참조하십시오.
- Red Hat OpenShift Container Platform에서 권장되는 구성 가능한 스토리지 기술에 대한 자세한 내용은 권장 설정 가능한 스토리지 기술을 참조하십시오.
- STF에서 Prometheus의 영구 스토리지를 구성하는 방법에 대한 자세한 내용은 "Prometheus의 영구 스토리지 구성"을 참조하십시오.
- STF에서 Elasticsearch의 영구 스토리지를 구성하는 방법에 대한 자세한 내용은 "Elasticsearch의 영구 스토리지 구성"을 참조하십시오.

### 2.2.1. 임시 스토리지

임시 스토리지를 사용하여 Red Hat OpenShift Container Platform 클러스터에 데이터를 영구적으로 저장하지 않고 STF(Service Telemetry Framework)를 실행할 수 있습니다.

**주의**

임시 스토리지를 사용하는 경우 Pod가 재시작, 업데이트 또는 다른 노드에 다시 예약되는 경우 데이터 손실이 발생할 수 있습니다. 임시 스토리지는 프로덕션 환경이 아닌 개발 또는 테스트에만 사용합니다.

## 2.3. 리소스 할당

Red Hat OpenShift Container Platform 인프라 내에서 Pod 예약을 활성화하려면 실행 중인 구성 요소에 대한 리소스가 필요합니다. 충분한 리소스를 할당하지 않으면 Pod를 예약할 수 없기 때문에 **Pending** 상태로 유지됩니다.

STF(Service Telemetry Framework)를 실행하는 데 필요한 리소스의 양은 사용자 환경과 모니터링할 노드 및 클라우드 수에 따라 다릅니다.

**추가 리소스**

- 지표 컬렉션의 크기 조정에 대한 권장 사항은 Service Telemetry Framework 성능 및 확장을 참조하십시오.
- Elasticsearch의 크기 조정에 대한 자세한 내용은 https://www.elastic.co/guide/en/cloud-on-k8s/current/k8s-managing-compute-resources.html 를 참조하십시오.

## 2.4. SERVICE TELEMETRY FRAMEWORK에 대한 네트워크 고려 사항

완전히 연결된 네트워크 환경에만 STF(Service Telemetry Framework)를 배포할 수 있습니다. Red Hat OpenShift Container Platform 관련 환경 또는 네트워크 프록시 환경에 STF를 배포할 수 없습니다.

# 3장. SERVICE TELEMETRY FRAMEWORK의 핵심 구성 요소 설치

Operator를 사용하여 STF(Service Telemetry Framework) 구성 요소 및 오브젝트를 로드할 수 있습니다. Operator는 다음 STF 코어 및 커뮤니티 구성 요소를 각각 관리합니다.

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

**사전 요구 사항**

- Red Hat OpenShift Container Platform 버전 4.10이 실행 중입니다.
- Red Hat OpenShift Container Platform 환경을 준비하고 Red Hat OpenShift Container Platform 환경 상단에 STF 구성 요소를 실행할 수 있는 영구 스토리지와 충분한 리소스가 있는지 확인합니다. 자세한 내용은 Service Telemetry Framework 성능 및 확장을 참조하십시오.
- 환경이 완전히 연결되어 있습니다. Red Hat OpenShift Container Platform이 분리된 환경 또는 네트워크 프록시 환경에서는 STF가 작동하지 않습니다.

**중요**

STF는 Red Hat OpenShift Container Platform 버전 4.10과 호환됩니다.

**추가 리소스**

- Operator에 대한 자세한 내용은 Operator *이해 가이드를* 참조하십시오.

## 3.1. RED HAT OPENSHIFT CONTAINER PLATFORM 환경에 SERVICE TELEMETRY FRAMEWORK 배포

STF(Service Telemetry Framework)를 배포하여 이벤트를 수집, 저장, 모니터링합니다.

**절차**

1. STF 구성 요소(예: **service-telemetry** )를 포함할 네임스페이스를 생성합니다.

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

2. Operator Pod를 예약할 수 있도록 네임스페이스에 OperatorGroup을 생성합니다.

```
$ 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
```

자세한 내용은 OperatorGroups 를 참조하십시오.

3. cert-manager Operator의 네임스페이스를 생성합니다.

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

4. cert-manager Operator에 대한 OperatorGroup을 생성합니다.

```
$ 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. redhat-operators CatalogSource를 사용하여 cert-manager Operator를 등록합니다.

```
$ 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. ClusterServiceVersion을 검증합니다. cert-manager Operator에 **Succeeded** 단계가 표시되는지 확인합니다.

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

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

7. redhat-operators CatalogSource를 사용하여 AMQ Interconnect Operator를 등록합니다.

```
$ 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. ClusterServiceVersion을 검증합니다. amq7-interconnect-operator.v1.10.10이 **Succeeded** 단계를 표시하는지 확인합니다.

```
$ 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. OperatorHub.io 커뮤니티 카탈로그 소스를 활성화하여 데이터 스토리지 및 시각화 Operator를 설치합니다.

**주의**

Red Hat은 AMQ Interconnect, Service Telemetry Operator 및 Smart Gateway Operator를 포함한 핵심 Operator 및 워크로드를 지원합니다. Red Hat은 Elasticsearch, Prometheus, Alertmanager, Grafana 및 해당 Operator를 포함한 커뮤니티 Operator 또는 워크로드 구성 요소를 지원하지 않습니다.

```
$ 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. Prometheus에 지표를 저장하려면 Prometheus Operator를 활성화해야 합니다. Prometheus Operator를 활성화하려면 Red Hat OpenShift Container Platform 환경에서 다음 매니페스트를 생성합니다.

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

11. Prometheus **Succeeded** 의 ClusterServiceVersion을 확인합니다.

```
$ 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. Elasticsearch에 이벤트를 저장하려면 ECK(Elastic Cloud on Kubernetes) Operator를 활성화해야 합니다. ECK Operator를 활성화하려면 Red Hat OpenShift Container Platform 환경에서 다음 매니페스트를 생성합니다.

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

13. Kubernetes **Succeeded** 에서 Elastic Cloud의 ClusterServiceVersion을 확인하십시오.

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

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

14. Service Telemetry Operator 서브스크립션을 생성하여 STF 인스턴스를 관리합니다.

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

15. Service Telemetry Operator 및 종속 Operator를 확인합니다.

```
$ 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. RED HAT OPENSHIFT CONTAINER PLATFORM에서 SERVICETELEMETRY 오브젝트 생성

Red Hat OpenShift Container Platform에서 **ServiceTelemetry** 오브젝트를 생성하여 Service Telemetry Operator에서 STF(Service Telemetry Framework) 배포에 대한 지원 구성 요소를 생성합니다. 자세한 내용은 3.2.1절. “ServiceTelemetry 오브젝트의 기본 매개변수” 의 내용을 참조하십시오.

**절차**

1. 기본값을 사용하는 STF 배포를 생성하는 **ServiceTelemetry** 오브젝트를 생성하려면 빈 **spec** 매개변수를 사용하여 **ServiceTelemetry** 오브젝트를 생성합니다.

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

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

기본값을 재정의하려면 재정의할 매개변수를 정의합니다. 이 예에서는 **enabled** 를 **true** 로 설정하여 Elasticsearch를 활성화합니다.

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

빈 **spec** 매개변수를 사용하여 **ServiceTelemetry** 오브젝트를 생성하면 다음과 같은 기본 설정으로 STF 배포가 생성됩니다.

```
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
```

이러한 기본값을 재정의하려면 **spec** 매개변수에 구성을 추가합니다.

2. Service Telemetry Operator에서 STF 배포 로그를 확인합니다.

```
$ 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
```

검증

- 모든 워크로드가 올바르게 작동하는지 확인하려면 Pod 및 각 Pod의 상태를 확인합니다.

**참고**

**backends.events.elasticsearch.enabled** 매개변수를 **true** 로 설정하면 Smart Gateways에서 Elasticsearch가 시작되기 전의 기간 동안 **Error** 및 **CrashLoopBackOff** 오류 메시지를 보고합니다.

```
$ 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 오브젝트의 기본 매개변수

**ServiceTelemetry** 오브젝트는 다음과 같은 주요 구성 매개변수로 구성됩니다.

- **경고**
- **백엔드**
- **클라우드**
- **그래프**
- **highAvailability**
- **전송**

STF 배포에 다양한 기능을 제공하도록 이러한 각 구성 매개변수를 구성할 수 있습니다.

**backends 매개변수**

**backends** 매개변수를 사용하여 지표 및 이벤트의 스토리지에 사용할 수 있는 스토리지 백엔드를 제어하고, **clouds** 매개 변수가 정의하는 Smart Gateway 사용을 제어합니다. 자세한 내용은 "clouds 매개변수"의 내용을 참조하십시오.

현재는 Prometheus를 지표 스토리지 백엔드로 사용하고 Elasticsearch를 이벤트 스토리지 백엔드로 사용할 수 있습니다.

**Prometheus를 메트릭의 스토리지 백엔드로 활성화**
Prometheus를 메트릭의 스토리지 백엔드로 활성화하려면 **ServiceTelemetry** 오브젝트를 구성해야 합니다.

**절차**

- **ServiceTelemetry** 오브젝트를 구성합니다.

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

**Prometheus의 영구 스토리지 구성**
**backends.metrics.prometheus.storage.persistent** 에 정의된 추가 매개변수를 사용하여 스토리지 클래스 및 볼륨 크기와 같은 Prometheus의 영구 스토리지 옵션을 구성합니다.

**storageClass** 를 사용하여 백엔드 스토리지 클래스를 정의합니다. 이 매개변수를 설정하지 않으면 Service Telemetry Operator에서 Red Hat OpenShift Container Platform 클러스터에 기본 스토리지 클래스를 사용합니다.

**pvcStorageRequest** 매개변수를 사용하여 스토리지 요청을 충족하기 위해 필요한 최소 볼륨 크기를 정의합니다. 볼륨이 정적으로 정의된 경우 요청된 볼륨보다 큰 볼륨 크기를 사용할 수 있습니다. 기본적으로 Service Telemetry Operator는 **20G** (20Gigabytes)의 볼륨 크기를 요청합니다.

**절차**

- 사용 가능한 스토리지 클래스를 나열합니다.

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

- **ServiceTelemetry** 오브젝트를 구성합니다.

```
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
```

**이벤트의 스토리지 백엔드로 Elasticsearch 활성화**

Elasticsearch를 이벤트의 스토리지 백엔드로 활성화하려면 **ServiceTelemetry** 오브젝트를 구성해야 합니다.

**절차**

- **ServiceTelemetry** 오브젝트를 구성합니다.

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

**Elasticsearch의 영구 스토리지 구성**

**backends.events.elasticsearch.storage.persistent** 에 정의된 추가 매개변수를 사용하여 스토리지 클래스 및 볼륨 크기와 같은 Elasticsearch의 영구 스토리지 옵션을 구성합니다.

**storageClass** 를 사용하여 백엔드 스토리지 클래스를 정의합니다. 이 매개변수를 설정하지 않으면 Service Telemetry Operator에서 Red Hat OpenShift Container Platform 클러스터에 기본 스토리지 클래스를 사용합니다.

**pvcStorageRequest** 매개변수를 사용하여 스토리지 요청을 충족하기 위해 필요한 최소 볼륨 크기를 정의합니다. 볼륨이 정적으로 정의된 경우 요청된 볼륨보다 큰 볼륨 크기를 사용할 수 있습니다. 기본적으로 Service Telemetry Operator는 볼륨 크기 **20Gi** (20Gibibytes)를 요청합니다.

**절차**

- 사용 가능한 스토리지 클래스를 나열합니다.

```
$ oc get storageclasses
NAME                 PROVISIONER                RECLAIMPOLICY   VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION   AGE
csi-manila-ceph      manila.csi.openstack.org   Delete          Immediate           false
20h
```

```
standard (default)   kubernetes.io/cinder       Delete          WaitForFirstConsumer   true
20h
standard-csi         cinder.csi.openstack.org   Delete          WaitForFirstConsumer   true
20h
```

- **ServiceTelemetry** 오브젝트를 구성합니다.

```
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 매개변수**
**clouds** 매개변수를 사용하여 배포되는 Smart Gateway 오브젝트를 정의하면 모니터링되는 여러 클라우드 환경에 대한 인터페이스를 제공하여 STF 인스턴스에 연결할 수 있습니다. 지원 백엔드를 사용할 수 있는 경우 기본 클라우드 구성에 대한 지표 및 이벤트 Smart Gateway가 생성됩니다. 기본적으로 Service Telemetry Operator는 **cloud1** 용 Smart Gateway를 생성합니다.

정의된 클라우드에 대해 생성된 Smart Gateway를 제어하는 클라우드 오브젝트 목록을 생성할 수 있습니다. 각 클라우드는 데이터 유형 및 수집기로 구성됩니다. 데이터 유형은 **메트릭** 또는 **이벤트** 입니다. 각 데이터 유형은 수집기 목록, 메시지 버스 서브스크립션 주소, 디버깅을 활성화하는 매개 변수로 구성됩니다. 메트릭에 사용 가능한 수집기는 **collectd**,**ceilometer** 및 **sensuability입니다**. 이벤트에 사용 가능한 수집기는 **collectd** 및 **ceilometer** 입니다. 이러한 각 컬렉터의 서브스크립션 주소가 모든 클라우드, 데이터 유형 및 수집기 조합에 대해 고유해야 합니다.

기본 **cloud1** 구성은 특정 클라우드 인스턴스에 대해 collectd, Ceilometer 및 Sensubility 데이터 수집기에 대한 메트릭 및 이벤트의 서브스크립션 및 데이터 스토리지를 제공하는 **ServiceTelemetry** 오브젝트로 표시됩니다.

```
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** 매개변수의 각 항목은 클라우드 인스턴스를 나타냅니다. 클라우드 인스턴스는 세 가지 최상위 매개변수( **이름**,**메트릭** 및 **이벤트** )로 구성됩니다. **metrics** 및 **events** 매개변수는 해당 데이터 유형의 스토리지에 대한 해당 백엔드를 나타냅니다. **collectors** 매개변수는 두 개의 필수 매개 변수인 **collectorType** 및 **subscriptionAddress** 로 구성된 오브젝트 목록을 지정하고, 이는 Smart Gateway 인스턴스를 나타냅니다. **collectorType** 매개변수는 collectd, Ceilometer 또는 Sensubility에서 수집한 데이터를 지정합니다. **subscriptionAddress** 매개변수는 Smart Gateway가 구독하는 AMQ Interconnect 주소를 제공합니다.

**collectors** 매개변수 내에서 선택적 부울 매개변수 **debugEnabled** 를 사용하여 실행중인 Smart Gateway Pod에서 추가 콘솔 디버깅을 활성화할 수 있습니다.

**추가 리소스**

- 기본 Smart Gateway 삭제에 대한 자세한 내용은 4.5.3절. "기본 Smart Gateway 삭제" 을 참조하십시오.
- 여러 클라우드를 구성하는 방법에 대한 자세한 내용은 4.5절. "여러 클라우드 구성" 을 참조하십시오.

**alert 매개변수**

alert **매개변수** 를 사용하여 Alertmanager 인스턴스 생성 및 스토리지 백엔드 구성을 제어합니다. 기본적으로 **경고가** 활성화됩니다. 자세한 내용은 5.3절. "Service Telemetry Framework의 경고"의 내용을 참조하십시오.

**graphing 매개변수**

graphing **매개변수** 를 사용하여 Grafana 인스턴스 생성을 제어합니다. 기본적으로 **그래프** 는 비활성화되어 있습니다. 자세한 내용은 5.1절. "Service Telemetry Framework의 대시보드"의 내용을 참조하십시오.

**highAvailability 매개변수**

**highAvailability** 매개변수를 사용하여 STF 구성 요소의 여러 사본 인스턴스화를 제어하여 실패하거나 다시 예약한 구성 요소의 복구 시간을 단축합니다. 기본적으로 **highAvailability** 는 비활성화되어 있습니다. 자세한 내용은 5.5절. "고가용성"의 내용을 참조하십시오.

**transports 매개변수**

**transports** 매개변수를 사용하여 STF 배포를 위한 메시지 버스의 사용을 제어합니다. 현재 지원되는 유일한 전송은 AMQ Interconnect입니다. 기본적으로 **qdr** 전송이 활성화됩니다.

## 3.3. STF 구성 요소의 사용자 인터페이스에 액세스

Red Hat OpenShift Container Platform에서 애플리케이션은 경로를 통해 외부 네트워크에 노출됩니다. 경로에 대한 자세한 내용은 Ingress 클러스터 트래픽 구성을 참조하십시오.

STF(Service Telemetry Framework)에서 웹 기반 인터페이스가 있는 각 서비스에 대해 HTTPS 경로가 노출됩니다. 이러한 경로는 Red Hat OpenShift Container Platform RBAC에 의해 보호되며 Red Hat OpenShift Container Platform 네임스페이스를 볼 수 있는 **ClusterRoleBinding** 이 있는 모든 사용자가 로그인할 수 있습니다. RBAC에 대한 자세한 내용은 RBAC를 사용하여 권한 정의 및 적용을 참조하십시오.

**절차**

1. Red Hat OpenShift Container Platform에 로그인합니다.
2. **service-telemetry** 네임스페이스로 변경합니다.

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

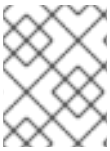
3. **service-telemetry** 프로젝트에서 사용 가능한 웹 UI 경로를 나열합니다.

```
$ 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. 웹 브라우저에서 https://<route_address> 로 이동하여 해당 서비스의 웹 인터페이스에 액세스합니다.

## 3.4. 대체 관찰 전략 구성

스토리지, 시각화 및 경고 백엔드 배포를 건너뛰도록 STF를 구성하려면 ServiceTelemetry 사양에 **observabilityStrategy: none** 을 추가합니다. 이 모드에서는 AMQ Interconnect 라우터 및 지표 Smart Gateway만 배포되며 STF Smart Gateway에서 지표를 수집하도록 외부 Prometheus 호환 시스템을 구성해야 합니다.



**참고**

현재 **observabilityStrategy** 를 **none** 으로 설정할 때 메트릭만 지원됩니다. Events Smart Gateways는 배포되지 않습니다.

**절차**

1. **spec** 매개변수에 **observabilityStrategy: none** 속성을 사용하여 **ServiceTelemetry** 오브젝트를 생성합니다. 매니페스트는 모든 지표 수집기 유형이 있는 단일 클라우드에서 Telemetry를 수신하는 데 적합한 STF의 기본 배포가 표시됩니다.

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

2. 모든 워크로드가 올바르게 작동하는지 확인하려면 Pod 및 각 Pod의 상태를 확인합니다.

```
$ 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
```

### 추가 리소스

추가 클라우드를 구성하거나 지원되는 컬렉터 집합을 변경하는 방법에 대한 자세한 내용은 을 참조하십시오. 4.5.2절. “스마트 게이트웨이 배포”

# 4장. SERVICE TELEMETRY FRAMEWORK용 RED HAT OPENSTACK PLATFORM 구성

지표, 이벤트 또는 둘 다를 수집하고 이를 STF(Service Telemetry Framework) 스토리지 도메인에 보내려면 데이터 수집 및 전송을 활성화하도록 RHOSP(Red Hat OpenStack Platform) 오버클라우드를 구성해야 합니다.

STF는 단일 및 다중 클라우드를 모두 지원할 수 있습니다. 단일 클라우드 설치에 대해 설정된 RHOSP 및 STF의 기본 구성입니다.

- 기본 구성이 포함된 단일 RHOSP 오버클라우드 배포의 경우 4.1절. "Service Telemetry Framework용 Red Hat OpenStack Platform 오버클라우드 배포" 을 참조하십시오.
- RHOSP 설치 및 구성 STF를 여러 클라우드에 계획하려면 4.5절. "여러 클라우드 구성" 을 참조하십시오.
- RHOSP 오버클라우드 배포의 일부로 해당 환경에서 추가 기능을 구성해야 할 수 있습니다.
  - 데이터 컬렉션을 배포하고 DCN(Distributed Compute Node) 또는 스파인-리프형과 같이 라우팅된 L3 도메인을 사용하는 RHOSP 클라우드 노드에서 STF로 전송하려면 4.3절. "비표준 네트워크 토폴로지에 배포" 을 참조하십시오.
  - 데이터 수집기 서비스를 비활성화하려면 4.2절. "Service Telemetry Framework와 함께 사용되는 Red Hat OpenStack Platform 서비스 비활성화" 을 참조하십시오.
  - 컨테이너 이미지를 로컬 레지스트리에 동기화한 경우 환경 파일을 생성하고 컨테이너 이미지의 경로를 포함해야 합니다. 자세한 내용은 4.4절. "언더클라우드에 Red Hat OpenStack Platform 컨테이너 이미지 추가"의 내용을 참조하십시오.

## 4.1. SERVICE TELEMETRY FRAMEWORK용 RED HAT OPENSTACK PLATFORM 오버클라우드 배포

RHOSP(Red Hat OpenStack Platform) 오버클라우드 배포의 일부로 데이터 수집기와 STF(Service Telemetry Framework)로 데이터 전송을 구성해야 합니다.

**절차**

1. 4.1.1절. "오버클라우드 구성을 위한 Service Telemetry Framework에서 CA 인증서 가져오기"
2. AMQ Interconnect 경로 주소 검색
3. STF의 기본 구성 생성
4. 오버클라우드의 STF 연결 구성
5. 오버클라우드 배포
6. 클라이언트 측 설치 검증

### 4.1.1. 오버클라우드 구성을 위한 Service Telemetry Framework에서 CA 인증서 가져오기

Red Hat OpenStack Platform 오버클라우드를 STF(Service Telemetry Framework)에 연결하려면 Service Telemetry Framework 내에서 실행되는 AMQ Interconnect의 CA 인증서를 검색하고 Red Hat OpenStack Platform 구성에서 인증서를 사용합니다.

**절차**

1. Service Telemetry Framework에서 사용 가능한 인증서 목록을 확인합니다.

```
$ oc get secrets
```

2. **default-interconnect-selfsigned** Secret의 콘텐츠를 검색하고 기록해 둡니다.

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

### 4.1.2. AMQ Interconnect 경로 주소 검색

STF(Service Telemetry Framework)용 RHOSP(Red Hat OpenStack Platform) 오버클라우드를 구성하는 경우 STF 연결 파일에 AMQ Interconnect 경로 주소를 제공해야 합니다.

**절차**

1. Red Hat OpenShift Container Platform 환경에 로그인합니다.
2. **service-telemetry** 프로젝트에서 AMQ Interconnect 경로 주소를 검색합니다.

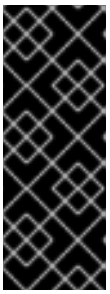
```
$ oc get routes -ogo-template='{{ range .items }}{{printf "%s\n" .spec.host }}{{ end }}' | grep "\-5671"
default-interconnect-5671-service-telemetry.apps.infra.watch
```

### 4.1.3. STF의 기본 구성 생성

STF(Service Telemetry Framework)에 대해 호환되는 데이터 수집 및 전송을 제공하도록 기본 매개변수를 구성하려면 기본 데이터 수집 값을 정의하는 파일을 만들어야 합니다.

**절차**

1. **stack** 사용자로 RHOSP(Red Hat OpenStack Platform) 언더클라우드에 로그인합니다.
2. **/home/stack** 디렉터리에 **enable-stf.yaml** 이라는 구성 파일을 생성합니다.



**중요**

**EventPipelinePublishers** 및 **PipelinePublishers** 를 표시하도록 설정하면 Gnocchi 또는 Panko와 같은 RHOSP Telemetry 구성 요소에 전달하는 이벤트 또는 지표 데이터가 발생하지 않습니다. 추가 파이프라인으로 데이터를 보내야 하는 경우 **ExtraConfig** 에 지정된 대로 Ceilometer 폴링 간격 30초가 RHOSP Telemetry 구성 요소를 초과할 수 있으며 간격을 **300** 과 같은 더 큰 값으로 늘려야 합니다. 값을 더 긴 폴링 간격으로 늘리면 STF에서 Telemetry 확인이 줄어듭니다.

**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. 오버클라우드의 STF 연결 구성

STF(Service Telemetry Framework) 연결을 구성하려면 오버클라우드의 AMQ Interconnect 연결 구성이 STF 배포에 포함된 파일을 생성해야 합니다. STF에서 이벤트 및 스토리지 컬렉션을 활성화하고 오버클라우드를 배포합니다. 기본 구성은 기본 메시지 버스 주제가 있는 단일 클라우드 인스턴스의 기본 구성입니다. 여러 클라우드 배포 구성은 4.5절. "여러 클라우드 구성" 에서 참조하십시오.

**사전 요구 사항**

- STF에서 배포한 AMQ Interconnect에서 CA 인증서를 검색합니다. 자세한 내용은 4.1.1절. "오버클라우드 구성을 위한 Service Telemetry Framework에서 CA 인증서 가져오기"의 내용을 참조하십시오.
- AMQ Interconnect 경로 주소를 검색합니다. 자세한 내용은 4.1.2절. "AMQ Interconnect 경로 주소 검색"의 내용을 참조하십시오.

**절차**

1. RHOSP 언더클라우드에 **stack** 사용자로 로그인합니다.
2. **/home/stack** 디렉터리에 **stf-connectors.yaml** 이라는 구성 파일을 생성합니다.
3. **stf-connectors.yaml** 파일에서 **MetricsQdrConnectors** 주소를 구성하여 오버클라우드의 AMQ Interconnect를 STF 배포에 연결합니다. STF의 기본값과 일치하도록 이 파일에서 Sensubility, Ceilometer, collectd에 대한 주제 주소를 구성합니다. 주제 및 클라우드 구성 사용자 지정에 대한 자세한 내용은 4.5절. "여러 클라우드 구성" 을 참조하십시오.
    - 여러 클라우드 배포에 대한 **collectd-write-qdr.yaml** 환경 파일이 포함되지 않으므로 **resource_registry** 구성이 collectd 서비스를 직접 로드합니다.
    - **host** 매개변수를 4.1.2절. "AMQ Interconnect 경로 주소 검색" 에서 검색한 **HOST/PORT** 값으로 바꿉니다.
    - **caCertFileContent** 매개변수를 4.1.1절. "오버클라우드 구성을 위한 Service Telemetry Framework에서 CA 인증서 가져오기" 에서 검색된 콘텐츠로 교체합니다.
    - **MetricsQdrConnectors** 의 **호스트** 하위 매개변수를 4.1.2절. "AMQ Interconnect 경로 주소 검색" 에서 검색한 **HOST/PORT** 값으로 바꿉니다.
    - Ceilometer 이벤트의 주제를 정의하려면 **CeilometerQdrEventsConfig.topic** 을 설정합니다. 이 값의 형식은 **anycast/ceilometer/cloud1-event.sample** 입니다.
    - **CeilometerQdrMetricsConfig.topic** 을 설정하여 Ceilometer 메트릭에 대한 주제를 정의합니다. 이 값의 형식은 **anycast/ceilometer/cloud1-metering.sample** 입니다.

- **CollectdAmqpInstances** 를 설정하여 collectd 이벤트에 대한 주제를 정의합니다. 이 값의 형식은 **collectd/cloud1-notify** 입니다.
- collectd 메트릭에 대한 주제를 정의하려면 **CollectdAmqpInstances** 를 설정합니다. 이 값의 형식은 **collectd/cloud1-telemetry** 입니다.

**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. 오버클라우드 배포

데이터를 수집하고 STF(Service Telemetry Framework)로 전송하도록 필요한 환경 파일을 사용하여 오버클라우드를 배포 또는 업데이트합니다.

**절차**

1. **stack** 사용자로 RHOSP(Red Hat OpenStack Platform) 언더클라우드에 로그인합니다.
2. 인증 파일을 소싱합니다.

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

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

3. RHOSP director 배포에 다음 파일을 추가하여 데이터 수집 및 AMQ Interconnect를 구성합니다.
   - Ceilometer Telemetry 및 이벤트가 STF로 전송되도록 **ceilometer-write-qdr.yaml** 파일
   - 메시지 버스가 활성화되어 STF 메시지 버스 라우터에 연결되어 있는지 확인하는 **qdr-edge-only.yaml** 파일
   - 기본값이 올바르게 구성되어 있는지 확인하는 **enable-stf.yaml** 환경 파일
   - STF에 대한 연결을 정의하는 **stf-connectors.yaml** 환경 파일
4. RHOSP 오버클라우드를 배포합니다.

```
(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. 클라이언트 측 설치 검증

STF(Service Telemetry Framework) 스토리지 도메인에서 데이터 수집을 검증하려면 전달된 데이터의 데이터 소스를 쿼리합니다. RHOSP(Red Hat OpenStack Platform) 배포에서 개별 노드를 확인하려면 SSH를 사용하여 콘솔에 연결합니다.

#### 작은 정보

일부 Telemetry 데이터는 RHOSP에 활성 워크로드가 있는 경우에만 사용할 수 있습니다.

#### 절차

1. 오버클라우드 노드에 로그인합니다(예: controller-0).
2. **metrics_qdr** 컨테이너가 노드에서 실행 중인지 확인합니다.

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

running
```

3. 포트 **5666** 에서 수신 대기하는 AMQ Interconnect가 실행 중인 내부 네트워크 주소를 반환합니다(예: 172.17 **.** 1.44).

```
$ 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. 로컬 AMQ Interconnect에 대한 연결 목록을 반환합니다.

```
$ 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
```

4개의 연결이 있습니다.

- STF에 대한 아웃 바운드 연결
- ceilometer에서 인바운드 연결
- collectd의 인바운드 연결
- **qdstat** 클라이언트의 인바운드 연결
  아웃바운드 STF 연결은 **MetricsQdrConnectors** 호스트 매개변수에 제공되며 STF 스토리지 도메인의 경로입니다. 다른 호스트는 이 AMQ Interconnect에 대한 클라이언트 연결의 내부 네트워크 주소입니다.

5. 메시지가 전달되도록 하려면 링크를 나열하고 메시지 전달을 위해 **deliv** 열의 **_edge** 주소를 확인합니다.

```
$ 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. RHOSP 노드에서 STF에 주소를 나열하려면 Red Hat OpenShift Container Platform에 연결하여 AMQ Interconnect Pod 이름을 검색하고 연결을 나열합니다. 사용 가능한 AMQ Interconnect Pod를 나열합니다.

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

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

7. Pod에 연결하고 알려진 연결을 나열합니다. 이 예에서는 연결 ID 22, 23 및 24인 RHOSP 노드에서 세 개의 **에지** 연결이 있습니다.

```
$ 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. 네트워크에서 제공하는 메시지 수를 보려면 **oc exec** 명령과 함께 각 주소를 사용합니다.

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

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

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

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

## 4.2. SERVICE TELEMETRY FRAMEWORK와 함께 사용되는 RED HAT OPENSTACK PLATFORM 서비스 비활성화

**RHOSP(Red Hat OpenStack Platform)**를 배포하고 이를 **STF(Service Telemetry Framework)**에 연결할 때 사용되는 서비스를 비활성화합니다. 서비스 비활성화의 일부로 로그 또는 생성된 구성 파일이 제거되지 않습니다.

절차

1. **RHOSP** 언더클라우드에 **stack** 사용자로 로그인합니다.

2. 인증 파일을 소싱합니다.

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

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

3. disable-stf.yaml 환경 파일을 생성합니다.

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

4. RHOSP director 배포에서 다음 파일을 제거합니다.

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

5. RHOSP 오버클라우드를 업데이트합니다. 환경 파일 목록 초기에 disable-stf.yaml 파일을 사용해야 합니다. 목록 초기에 disable-stf.yaml 을 추가하면 다른 환경 파일이 서비스를 비활성화하는 구성을 덮어쓸 수 있습니다.

```
(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. 비표준 네트워크 토폴로지에 배포

노드가 기본 InternalApi 네트워크와 별도의 네트워크에 있는 경우 AMQ Interconnect가 STF(Service Telemetry Framework) 서버 인스턴스로 데이터를 전송할 수 있도록 구성 조정을 수행해야 합니다. 이 시나리오는 스파인-리프형 또는 DCN 토폴로지에서 일반적입니다. DCN 구성에 대한 자세한 내용은 Spine Leaf Networking 가이드를 참조하십시오.

RHOSP(Red Hat OpenStack Platform) 13.0과 함께 STF를 사용하고 Ceph, Block 또는 Object Storage 노드를 모니터링하려는 경우 스파인-리프형 및 DCN 네트워크 구성과 관련된 구성 변경과 유사한 구성을 변경해야 합니다. Ceph 노드를 모니터링하려면 CephStorageExtraConfig 매개변수를 사용하여 AMQ Interconnect 및 collectd 설정 파일에 로드할 네트워크 인터페이스를 정의합니다.

```
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')}"
```

마찬가지로 환경에서 Block 및 Object Storage 역할을 사용하는 경우 BlockStorageExtraConfig 및 ObjectStorageExtraConfig 매개변수를 지정해야 합니다.

스파인-리프형 토폴로지를 배포하려면 역할 및 네트워크를 생성한 다음 해당 네트워크를 사용 가능한 역할에 할당해야 합니다. RHOSP 배포를 위해 STF에 대한 데이터 수집 및 전송을 구성할 때 역할의 기본 네트워크는 InternalApi 입니다. Ceph, 블록 및 오브젝트 스토리지 역할의 경우 기본 네트워크는 스토리지 입니다. 스파인-리프형 구성으로 인해 다른 Leaf 그룹화에 다른 네트워크가 할당될 수 있으며 이러한 이름은 일반적으로 고유하므로 RHOSP 환경 파일의 parameter_defaults 섹션에 추가 구성이 필요합니다.

절차

1. 각 Leaf 역할에 사용할 수 있는 네트워크를 문서화합니다. 네트워크 이름 정의의 예는 스파 인 *Leaf Networking 가이드에서 네트워크 데이터 파일 생성* 을 참조하십시오. 이러한 그룹화에 대한 Leaf 그룹화(roles) 및 네트워크 할당 생성에 대한 자세한 내용은 *스파인 Leaf Networking* 가이드의 역할 데이터 파일 생성 을 참조하십시오.

2. 각 리프 역할의 ExtraConfig 섹션에 다음 구성 예제를 추가합니다. 이 예제에서 internal_api0 은 네트워크 정의의 name_lower 매개변수에 정의된 값이며 Compute0 leaf 역할이 연결된 네트워크입니다. 이 경우 네트워크 식별 0은 leaf 0의 Compute 역할에 해당하며 기본 내부 API 네트워크 이름과 다른 값을 나타냅니다.

   Compute0 leaf 역할의 경우 hiera 조회를 수행하여 collectd AMQP 호스트 매개변수에 할당할 특정 네트워크의 네트워크 인터페이스를 결정하려면 추가 구성을 지정합니다. AMQ Interconnect 리스너 address 매개변수에 대해 동일한 구성을 수행합니다.

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

이 예제 구성은 **CephStorage** 리프 역할에 있습니다.

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

### 4.4. 언더클라우드에 RED HAT OPENSTACK PLATFORM 컨테이너 이미지 추가

컨테이너 이미지를 로컬 레지스트리에 동기화한 경우 환경 파일을 생성하고 컨테이너 이미지의 경로를 포함해야 합니다.

절차

1. 새 환경 파일(예: **container-images.yaml** )을 생성하고 다음 컨테이너 이미지 및 경로를 삽입합니다.

```
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
```

& lt;image-registry-path >를 이미지 레지스트리 경로로 바꿉니다.

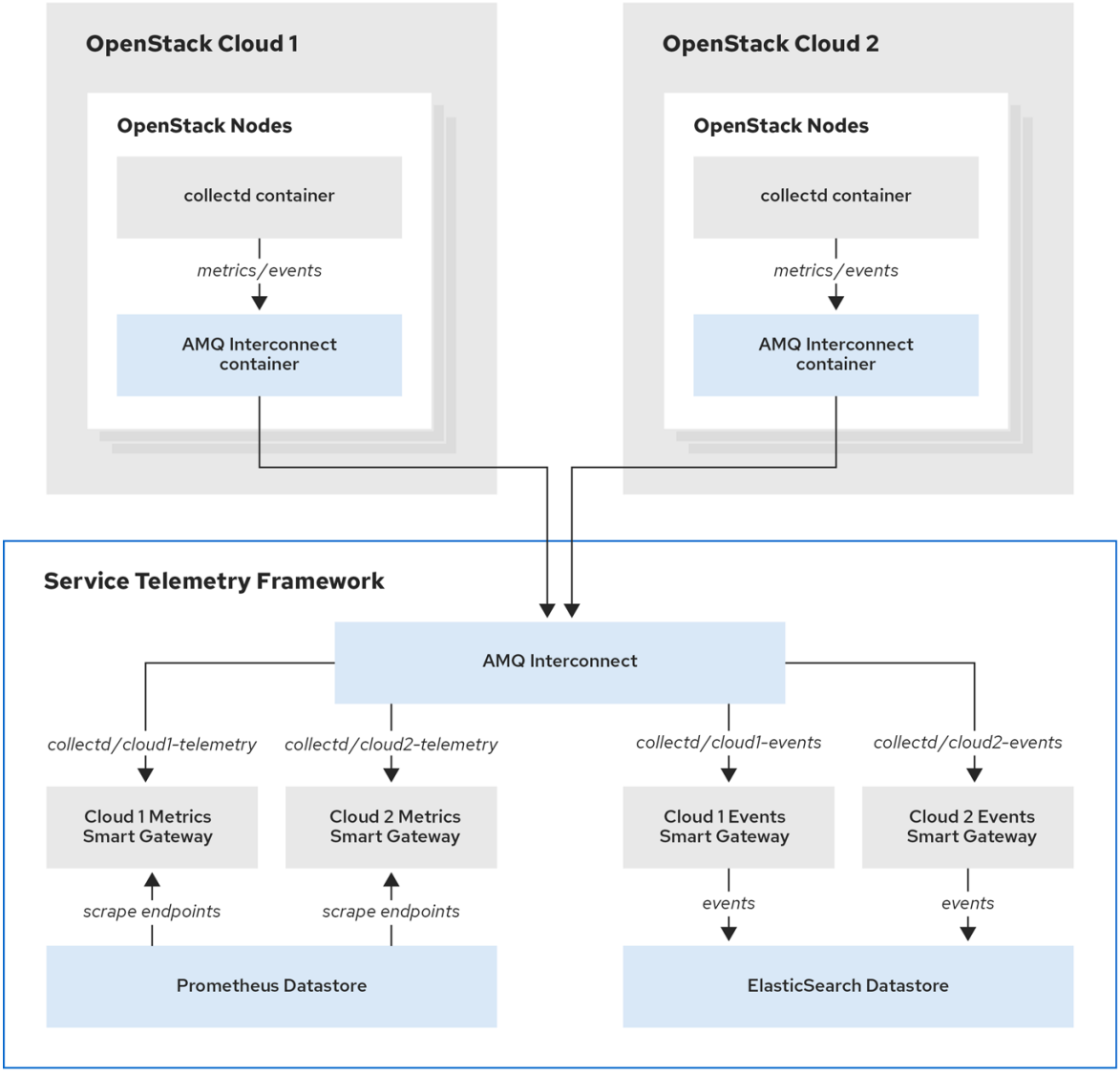
2. 오버클라우드에 **collectd** 및 **AMQ Interconnect**를 배포하려면 **Red Hat OpenStack Platform director**가 이미지를 준비할 수 있도록 **/usr/share/local-container-images/container-images.yaml** 환경 파일을 포함합니다. 다음 스니펫은 이 환경 파일을 포함하는 방법의 예입니다.

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

### 4.5. 여러 클라우드 구성

여러 개의 **RHOSP(Red Hat OpenStack Platform)** 클라우드를 구성하여 **Service Telemetry Framework(STF)**의 단일 인스턴스를 대상으로 할 수 있습니다. 여러 클라우드를 구성할 때 모든 클라우드는 고유한 메시지 버스 주제로 메트릭 및 이벤트를 보내야 합니다. **STF** 배포에서 **Smart Gateway** 인스턴스는 이러한 항목에서 수신 대기하여 공통 데이터 저장소에 정보를 저장합니다. 데이터 스토리지 도메인의 **Smart Gateway**가 저장하는 데이터는 **Smart Gateways**가 생성하는 메타데이터를 사용하여 필터링됩니다.

그림 **4.1.** 두 개의 **RHOSP** 클라우드가 **STF**에 연결




65_OpenStack_0120

여러 클라우드 시나리오에 대해 **RHOSP** 오버클라우드를 구성하려면 다음 작업을 완료합니다.

1. 각 클라우드에 사용하려는 **AMQP** 주소 접두사를 계획합니다. 자세한 내용은 **4.5.1**절. **“AMQP 주소 접두사 계획”**의 내용을 참조하십시오.

2. 각 클라우드에 대해 메트릭 및 이벤트 소비자 스마트 게이트웨이를 배포하여 해당 주소 접두

사에서 수신 대기합니다. 자세한 내용은 4.5.2절. “스마트 게이트웨이 배포”의 내용을 참조하십시오.

3. 고유한 도메인 이름으로 각 클라우드를 구성합니다. 자세한 내용은 4.5.4절. “고유한 클라우드 도메인 설정”의 내용을 참조하십시오.

4. STF의 기본 구성을 생성합니다. 자세한 내용은 4.1.3절. “STF의 기본 구성 생성”의 내용을 참조하십시오.

5. 올바른 주소에서 지표 및 이벤트를 STF에 보내도록 각 클라우드를 구성합니다. 자세한 내용은 4.5.5절. “여러 클라우드용 Red Hat OpenStack Platform 환경 파일 생성”의 내용을 참조하십시오.

### 4.5.1. AMQP 주소 접두사 계획

기본적으로 Red Hat OpenStack Platform 노드는 collectd 및 Ceilometer의 두 데이터 수집기를 통해 데이터를 수신합니다. 이러한 구성 요소는 Telemetry 데이터 또는 알림을 해당 AMQP 주소로 보냅니다(예: collectd/telemetry). STF Smart Gateways는 AMQP 주소에서 데이터 모니터링을 청취합니다. 여러 클라우드를 지원하고 모니터링 데이터를 생성한 클라우드를 식별하려면 각 클라우드를 구성하여 고유한 주소로 데이터를 보냅니다. 주소의 두 번째 부분에 클라우드 식별자 접두사를 추가합니다. 다음 목록은 일부 예제 주소 및 식별자를 보여줍니다.

- 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. 스마트 게이트웨이 배포

각 클라우드의 각 데이터 수집 유형에 대한 Smart Gateway를 배포해야 합니다. collectd 메트릭용이고, 하나는 collectd 이벤트용이고, 하나는 Ceilometer 이벤트용이고, 다른 하나는 Ceilometer 이벤트용입니다. 해당 클라우드에 대해 정의한 AMQP 주소에서 수신 대기하도록 각 Smart Gateway를 구성합니다. Smart Gateway를 정의하려면 ServiceTelemetry 매니페스트에서 clouds 매개변수를 구성합니다.

STF를 처음 배포할 때 단일 클라우드의 초기 스마트 게이트웨이를 정의하는 Smart Gateway 매니페스트가 생성됩니다. 여러 클라우드 지원을 위해 Smart Gateway를 배포할 때 각 클라우드의 지표 및 이벤트 데이터를 처리하는 각 데이터 수집 유형에 대해 여러 개의 Smart Gateway를 배포합니다. 초기 Smart Gateway는 다음과 같은 서브스크립션 주소를 사용하여 cloud1 에 정의됩니다.

| collector | type | 기본 서브스크립션 주소 |
|---|---|---|
| collectd | 메트릭 | collectd/telemetry |
| collectd | 이벤트 | collectd/notify |
| Ceilometer | 메트릭 | anycast/ceilometer/metering.sample |
| Ceilometer | 이벤트 | anycast/ceilometer/event.sample |

사전 요구 사항

- 클라우드 이름 지정 스키마가 결정되었습니다. 이름 지정 스키마를 결정하는 방법에 대한 자세한 내용은 4.5.1절. “AMQP 주소 접두사 계획” 을 참조하십시오.
- 클라우드 오브젝트 목록을 생성했습니다. clouds 매개변수의 콘텐츠 생성에 대한 자세한 내용은 “clouds 매개변수” 을 참조하십시오.

절차

1. Red Hat OpenShift Container Platform에 로그인합니다.

2. service-telemetry 네임스페이스로 변경합니다.

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

3. 기본 ServiceTelemetry 오브젝트를 편집하고 구성을 사용하여 clouds 매개변수를 추가합니다.

주의

긴 클라우드 이름은 63자의 최대 Pod 이름을 초과할 수 있습니다. ServiceTelemetry 이름 기본값 과 clouds.name 이 19자를 초과하지 않는지 확인합니다. 클라우드 이름은 - 과 같은 특수 문자를 포함할 수 없습니다. 클라우드 이름을 영숫자(a-z, 0-9)로 제한합니다.

주제 주소는 문자 제한이 없으며 clouds.name 값과 다를 수 있습니다.

```
$ 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. **ServiceTelemetry** 오브젝트를 저장합니다.

5. 각 **Smart Gateway**가 실행되고 있는지 확인합니다. **Smart Gateway** 수에 따라 몇 분이 걸릴 수 있습니다.

```
$ 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. 기본 Smart Gateway 삭제

여러 클라우드에 대해 **STF(Service Telemetry Framework)**를 구성한 후 더 이상 사용하지 않는 경우 기본 **Smart Gateway**를 삭제할 수 있습니다. **Service Telemetry Operator**는 생성된 **SmartGateway** 오브젝트를 제거할 수 있지만 더 이상 **ServiceTelemetry** 클라우드 목록에 나열되지 않습니다. **clouds** 매개변수에 정의되지 않은 **SmartGateway** 오브젝트를 제거하려면 **ServiceTelemetry** 매니페스트에서 **cloudsRemoveOnMissing** 매개변수를 **true** 로 설정해야 합니다.

작은 정보

**Smart Gateway**를 배포하지 않으려면 클라우드 **[]** 매개변수를 사용하여 빈 클라우드 목록을 정의합니다.

주의

**cloudsRemoveOnMising** 매개변수는 기본적으로 비활성화되어 있습니다. **cloudsRemoveOnMissing** 매개변수를 활성화하면 복원할 수 없이 현재 네임스페이스에서 수동으로 생성된 **SmartGateway** 오브젝트를 제거합니다.

절차

1. Service Telemetry Operator에서 관리할 클라우드 오브젝트 목록을 사용하여 clouds 매개변수를 정의합니다. 자세한 내용은 "clouds 매개변수"의 내용을 참조하십시오.

2. ServiceTelemetry 오브젝트를 편집하고 cloudsRemoveOnMissing 매개변수를 추가합니다.

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

3. 수정 사항을 저장합니다.

4. Operator가 Smart Gateways를 삭제했는지 확인합니다. Operator가 변경 사항을 조정하는 동안 몇 분이 걸릴 수 있습니다.

```
$ oc get smartgateways
```

### 4.5.4. 고유한 클라우드 도메인 설정

RHOSP(Red Hat OpenStack Platform)에서 STF(Service Telemetry Framework)로의 AMQ Interconnect 라우터 연결이 고유하고 충돌하지 않도록 CloudDomain 매개변수를 구성합니다.

절차

1. 새 환경 파일(예: hostnames .yaml) 을 생성합니다.

2. 다음 예와 같이 환경 파일에서 CloudDomain 매개변수를 설정합니다.

hostnames.yaml

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

3. 배포에 새 환경 파일을 추가합니다.

추가 리소스

- 4.5.5절. “여러 클라우드용 Red Hat OpenStack Platform 환경 파일 생성”
- Overcloud 매개변수 가이드의 *core Overcloud 매개변수*

### 4.5.5. 여러 클라우드용 Red Hat OpenStack Platform 환경 파일 생성

원본 클라우드에 따라 트래픽에 레이블을 지정하려면 클라우드별 인스턴스 이름을 사용하여 구성을 생성해야 합니다. **stf-connectors.yaml** 파일을 생성하고 **CeilometerQdrEventsConfig,CeilometerQdrMetricsConfig** 및 **CollectdAmqpInstances** 값을 **AMQP** 주소 접두사 체계와 일치하도록 조정합니다.

사전 요구 사항

- **STF**에서 배포한 **AMQ Interconnect**에서 **CA** 인증서를 검색했습니다. 자세한 내용은 4.1.1절. “오버클라우드 구성을 위한 Service Telemetry Framework에서 CA 인증서 가져오기”의 내용을 참조하십시오.
- 클라우드 오브젝트 목록을 생성했습니다. **clouds** 매개변수의 콘텐츠를 생성하는 방법에 대한 자세한 내용은 clouds 구성 매개변수 를 참조하십시오.
- **AMQ Interconnect** 경로 주소를 검색했습니다. 자세한 내용은 4.1.2절. “AMQ Interconnect 경로 주소 검색”의 내용을 참조하십시오.
- **STF**의 기본 구성을 생성했습니다. 자세한 내용은 4.1.3절. “STF의 기본 구성 생성”의 내용을 참조하십시오.

- 고유한 도메인 이름 환경 파일을 생성했습니다. 자세한 내용은 4.5.4절. “고유한 클라우드 도메인 설정”의 내용을 참조하십시오.

절차

1. Red Hat OpenStack Platform 언더클라우드에 stack 사용자로 로그인합니다.

2. /home/stack 디렉터리에 stf-connectors.yaml 이라는 구성 파일을 생성합니다.

3. stf-connectors.yaml 파일에서 MetricsQdrConnectors 주소를 구성하여 오버클라우드 배포의 AMQ Interconnect에 연결합니다. 이 클라우드 배포에 필요한 AMQP 주소와 일치하도록 CeilometerQdrEventsConfig,CeilometerQdrMetricsConfig 및 CollectdAmqpInstances 주제 값을 구성합니다.

   - 여러 클라우드 배포에 대한 collectd-write-qdr.yaml 환경 파일이 포함되지 않으므로 resource_registry 구성이 collectd 서비스를 직접 로드합니다.

   - MetricsQdrConnectors.host 매개변수를 4.1.2절. “AMQ Interconnect 경로 주소 검색” 에서 검색한 HOST/PORT 값으로 교체합니다.

   - caCertFileContent 매개변수를 4.1.1절. “오버클라우드 구성을 위한 Service Telemetry Framework에서 CA 인증서 가져오기” 에서 검색된 콘텐츠로 교체합니다.

   - Ceilometer 이벤트의 주제를 정의하려면 CeilometerQdrEventsConfig.topic 을 설정합니다. 이 값은 anycast/ceilometer/cloud1-event.sample 의 주소 형식입니다.

   - CeilometerQdrMetricsConfig.topic 을 설정하여 Ceilometer 지표의 주제를 정의합니다. 이 값은 anycast/ceilometer/cloud1-metering.sample 의 주소 형식입니다.

   - CollectdAmqpInstances 를 설정하여 collectd 이벤트에 대한 주제를 정의합니다. 이 값은 collectd/cloud1-notify 의 형식입니다.

   - CollectdAmqpInstances 를 설정하여 collectd 메트릭에 대한 주제를 정의합니다. 이 값은 collectd/cloud1-telemetry 형식입니다.

**stf-connectors.yaml**

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

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

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

    CeilometerQdrEventsConfig:
      driver: amqp
      topic: cloud1-event

    CeilometerQdrMetricsConfig:
      driver: amqp
      topic: cloud1-metering

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

4. **stf-connectors.yaml** 파일의 이름 지정 규칙이 **Smart Gateway** 구성의 **spec.bridge.amqpUrl** 필드와 일치하는지 확인합니다. 예를 들어 **CeilometerQdrEventsConfig.topic** 필드를 **cloud1-event** 값으로 구성합니다.

5. 인증 파일을 소싱합니다.

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

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

6. **openstack overcloud deployment** 명령에 **stf-connectors.yaml** 파일과 고유한 도메인 이름 환경 파일 **hostnames.yaml** 을 포함하고 해당 환경과 관련된 다른 환경 파일을 포함합니다.

주의

사용자 지정 **CollectdAmqpInstances** 매개변수와 함께 **collectd-write-qdr.yaml** 파일을 사용하는 경우 사용자 지정 및 기본 항목에 데이터가 게시됩니다. 여러 클라우드 환경에서 **stf-connectors.yaml** 파일의 **resource_registry** 매개변수 설정에 **collectd** 서비스를 로드합니다.

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

7. **Red Hat OpenStack Platform** 오버클라우드를 배포합니다.

추가 리소스

- 배포의 유효성을 확인하는 방법에 대한 자세한 내용은 4.1.6절. “클라이언트 측 설치 검증” 을 참조하십시오.

### 4.5.6. 여러 클라우드에서 메트릭 데이터 쿼리

**Prometheus**에 저장된 데이터에는 스크랩된 **Smart Gateway**에 따라 서비스 레이블이 있습니다. 이 레이블을 사용하여 특정 클라우드의 데이터를 쿼리할 수 있습니다.

특정 클라우드에서 데이터를 쿼리하려면 관련 서비스 레이블과 일치하는 **Prometheus *promql*** 쿼리를 사용합니다(예: **collectd_uptime{service="default-cloud1-coll-meter"}** ).

# 5장. SERVICE TELEMETRY FRAMEWORK의 운영 기능 사용

다음 운영 기능을 사용하여 STF(Service Telemetry Framework)에 추가 기능을 제공할 수 있습니다.

- 대시보드 구성
- 메트릭 보존 기간 구성
- 경고 구성
- SNMP 트랩 구성
- 고가용성 구성
- 임시 스토리지 구성
- 대체 관찰 전략 구성

## 5.1. SERVICE TELEMETRY FRAMEWORK의 대시보드

타사 애플리케이션 Grafana를 사용하여 데이터 수집기 및 Ceilometer가 각 개별 호스트 노드에 대해 수집하는 시스템 수준 지표를 시각화합니다.

데이터 수집기 구성에 대한 자세한 내용은 4.1절. “Service Telemetry Framework용 Red Hat OpenStack Platform 오버클라우드 배포” 을 참조하십시오.

### 5.1.1. 대시보드를 호스팅하도록 Grafana 구성

Grafana는 기본 STF(Service Telemetry Framework) 배포에 포함되지 않으므로 OperatorHub.io에서 Grafana Operator를 배포해야 합니다. Service Telemetry Operator를 사용하여 Grafana를 배포하면 Grafana 인스턴스와 로컬 STF 배포에 대한 기본 데이터 소스 구성이 생성됩니다.

절차

1. Red Hat OpenShift Container Platform에 로그인합니다.

2. service-telemetry 네임스페이스로 변경합니다.

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

3. Grafana Operator를 배포합니다.

```
$ 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. Operator가 시작되었는지 확인합니다. 명령 출력에서 PHASE 열의 값이 Succeeded 이면 Operator가 성공적으로 시작됩니다.

```
$ 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. Grafana 인스턴스를 시작하려면 ServiceTelemetry 오브젝트를 생성하거나 수정합니다. graphing.enabled 및 graphing .grafana.ingressEnabled를 true 로 설정합니다. 선택적으로 graphing.grafana.baseImage 값을 배포할 Grafana 워크로드 컨테이너 이미지로 설정합니다.

```
$ 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. Grafana 인스턴스가 배포되었는지 확인합니다.

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

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

7. Grafana 데이터 소스가 올바르게 설치되었는지 확인합니다.

```
$ oc get grafanadatasources

NAME                  AGE
default-datasources   20h
```

8. Grafana 경로가 있는지 확인합니다.

```
$ oc get route grafana-route

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

### 5.1.2. Grafana 로그인 자격 증명 검색 및 설정

STF(Service Telemetry Framework)는 Grafana가 활성화되면 기본 로그인 자격 증명을 설정합니다. ServiceTelemetry 오브젝트의 인증 정보를 덮어쓸 수 있습니다.

절차

1. Red Hat OpenShift Container Platform에 로그인합니다.

2. service-telemetry 네임스페이스로 변경합니다.

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

3. STF 오브젝트에서 기본 사용자 이름 및 암호를 검색합니다.

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

4. ServiceTelemetry 오브젝트를 통해 Grafana 관리자 사용자 이름 및 암호의 기본값을 수정하려면 graphing.grafana.adminUser 및 graphing.grafana.adminPassword 매개변수를 사용합니다.

## 5.2. SERVICE TELEMETRY FRAMEWORK의 지표 보존 기간

STF(Service Telemetry Framework)에 저장된 메트릭의 기본 보존 시간은 24시간이며 경고 목적으로 개발하기 위한 추세에 충분한 데이터를 제공합니다.

장기 스토리지의 경우 장기 데이터 보존용으로 설계된 시스템을 사용합니다(예: Thanos).

추가 리소스

- 추가 지표 보존 시간을 위해 STF를 조정하려면 5.2.1절. “Service Telemetry Framework에서 메트릭 보존 기간 편집” 을 참조하십시오.
- Prometheus 데이터 스토리지 및 스토리지 공간 추정에 대한 권장 사항은 https://prometheus.io/docs/prometheus/latest/storage/#operational-aspects을 참조하십시오.
- Thanos에 대한 자세한 내용은 https://thanos.io/를 참조하십시오.

### 5.2.1. Service Telemetry Framework에서 메트릭 보존 기간 편집

추가 메트릭 보존 시간에 대해 STF(Service Telemetry Framework)를 조정할 수 있습니다.

절차

1. Red Hat OpenShift Container Platform에 로그인합니다.

2. service-telemetry 네임스페이스로 변경합니다.

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

3. ServiceTelemetry 오브젝트를 편집합니다.

```
$ oc edit stf default
```

4. 보존 기간을 backends.metrics.prometheus.storage의 storage 섹션에 7d 를 추가하여 보존 기간을 7일로 늘립니다.

참고

긴 보존 기간을 설정하면 고도로 채워진 Prometheus 시스템에서 데이터를 검색하면 결과 반환 속도가 느려질 수 있습니다.

```
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. 변경 사항을 저장하고 오브젝트를 닫습니다.

추가 리소스

- 지표 보존 시간에 대한 자세한 내용은 5.2절. “Service Telemetry Framework의 지표 보존 기간” 을 참조하십시오.

### 5.3. SERVICE TELEMETRY FRAMEWORK의 경고

Prometheus 및 Alertmanager에서 경고 경로를 통해 경고 규칙을 생성합니다. Prometheus 서버의 경고 규칙은 경고를 관리하는 Alertmanager에 경고를 보냅니다. Alertmanager는 알림을 음소거, 억제 또는 집계하고 이메일, 전화 알림 시스템 또는 채팅 플랫폼을 사용하여 알림을 보낼 수 있습니다.

경고를 생성하려면 다음 작업을 완료합니다.

1. Prometheus에서 경고 규칙을 생성합니다. 자세한 내용은 5.3.1절. “Prometheus에서 경고 규칙 생성”의 내용을 참조하십시오.

2. Alertmanager에서 경고 경로를 만듭니다. 경고 경로를 생성하는 방법에는 다음 두 가지가 있습니다.

   - Alertmanager에서 표준 경고 경로 생성
   - Alertmanager에서 템플릿으로 경고 경로 생성

추가 리소스

Prometheus 및 Alertmanager를 사용한 경고 또는 알림에 대한 자세한 내용은 https://prometheus.io/docs/alerting/overview/을 참조하십시오.

STF(Service Telemetry Framework)에서 사용할 수 있는 경고 집합을 보려면 https://github.com/infrawatch/service-telemetry-operator/tree/master/deploy/alerts을 참조하십시오.

### 5.3.1. Prometheus에서 경고 규칙 생성

Prometheus는 알림을 트리거하기 위해 경고 규칙을 평가합니다. 규칙 조건이 빈 결과 집합을 반환하는 경우 조건은 false입니다. 그렇지 않으면 규칙이 true이고 경고를 트리거합니다.

절차

1. Red Hat OpenShift Container Platform에 로그인합니다.

2. service-telemetry 네임스페이스로 변경합니다.

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

3. 경고 규칙이 포함된 PrometheusRule 오브젝트를 생성합니다. Prometheus Operator는 규칙을 Prometheus에 로드합니다.

```
$ 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
```

규칙을 변경하려면 expr 매개변수 값을 편집합니다.

4. Operator가 규칙을 Prometheus에 로드했는지 확인하려면 기본 인증을 사용하여 default-prometheus-proxy 경로에 대해 curl 명령을 실행합니다.

```
$ 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"}]}}
```

추가 리소스

- 

경고에 대한 자세한 내용은 https://github.com/coreos/prometheus-operator/blob/master/Documentation/user-guides/alerting.md을 참조하십시오.

### 5.3.2. 사용자 정의 경고 구성

5.3.1절. “Prometheus에서 경고 규칙 생성” 에서 생성한 PrometheusRule 오브젝트에 사용자 정의 경고를 추가할 수 있습니다.

절차

1. oc edit 명령을 사용합니다.

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

2. PrometheusRules 매니페스트를 편집합니다.

3. 매니페스트를 저장하고 닫습니다.

추가 리소스

- 경고 규칙을 구성하는 방법에 대한 자세한 내용은 https://prometheus.io/docs/prometheus/latest/configuration/alerting_rules/ 를 참조하십시오.

- PrometheusRules 오브젝트에 대한 자세한 내용은 https://github.com/coreos/prometheus-operator/blob/master/Documentation/user-guides/alerting.md을 참조하십시오.

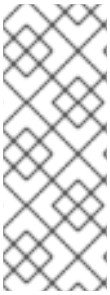
### 5.3.3. Alertmanager에서 표준 경고 경로 생성

Alertmanager를 사용하여 이메일, IRC 또는 기타 알림 채널과 같은 외부 시스템에 경고를 전달합니다. Prometheus Operator는 Alertmanager 설정을 Red Hat OpenShift Container Platform 시크릿으로 관리합니다. 기본적으로 STF(Service Telemetry Framework)는 수신자가 발생하지 않는 기본 구성을 배포합니다.

```
alertmanager.yaml: |-
  global:
    resolve_timeout: 5m
  route:
```

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

STF를 사용하여 사용자 정의 Alertmanager 경로를 배포하려면 alertmanagerConfigManifest 매개변수를 Service Telemetry Operator에 전달하여 Prometheus Operator에서 관리하는 업데이트된 보안을 제공해야 합니다.



참고

alertmanagerConfigManifest 에 전송된 경고의 제목과 텍스트를 구성하는 사용자 지정 템플릿이 포함된 경우 base64로 인코딩된 구성을 사용하여 alertmanagerConfigManifest 의 콘텐츠를 배포합니다. 자세한 내용은 5.3.4절. "Alertmanager에서 템플릿으로 경고 경로 생성"의 내용을 참조하십시오.

절차

1. Red Hat OpenShift Container Platform에 로그인합니다.

2. service-telemetry 네임스페이스로 변경합니다.

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

3. STF 배포에 대한 ServiceTelemetry 오브젝트를 편집합니다.

```
$ oc edit stf default
```

4. 새 매개변수 alertmanagerConfigManifest 및 Secret 오브젝트 콘텐츠를 추가하여 Alertmanager에 대한 alertmanager.yaml 구성을 정의합니다.

참고

이 단계에서는 **Service Telemetry Operator**가 관리하는 기본 템플릿을 로드합니다. 변경 사항이 올바르게 채워지는지 확인하려면 값을 변경하고 **alertmanager-default** 보안을 반환하고 새 값이 메모리에 로드되었는지 확인합니다. 예를 들어, **global.resolve_timeout** 매개 변수의 값을 **5m** 에서 **10m** 로 변경합니다.

```
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. 구성이 보안에 적용되었는지 확인합니다.

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

global:
  resolve_timeout: 10m
route:
  group_by: ['job']
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 12h
```

```
    receiver: 'null'
  receivers:
  - name: 'null'
```

6. alertmanager-proxy 서비스에 대해 curl 명령을 실행하여 status 및 configYAML 콘텐츠를 검색하고 제공된 구성이 Alertmanager의 구성과 일치하는지 확인합니다.

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

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

7. configYAML 필드에 예상 변경 사항이 포함되어 있는지 확인합니다.

8. 환경을 정리하려면 curl 포드를 삭제합니다.

```
$ oc delete pod curl

pod "curl" deleted
```

추가 리소스

- Red Hat OpenShift Container Platform 시크릿 및 Prometheus Operator에 대한 자세한 내용은 경고에 대한 Prometheus 사용자 가이드를 참조하십시오.

### 5.3.4. Alertmanager에서 템플릿으로 경고 경로 생성

Alertmanager를 사용하여 이메일, IRC 또는 기타 알림 채널과 같은 외부 시스템에 경고를 전달합니다. Prometheus Operator는 Alertmanager 설정을 Red Hat OpenShift Container Platform 시크릿으로 관리합니다. 기본적으로 STF(Service Telemetry Framework)는 수신자가 발생하지 않는 기본 구성을 배포합니다.

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

```
  receiver: 'null'
receivers:
- name: 'null'
```

**alertmanagerConfigManifest** 매개변수에 전송된 경고의 제목과 텍스트를 구성하기 위한 사용자 지정 템플릿이 포함된 경우 **base64**로 인코딩된 구성을 사용하여 **alertmanagerConfigManifest** 의 콘텐츠를 배포합니다.

절차

1. **Red Hat OpenShift Container Platform**에 로그인합니다.

2. **service-telemetry** 네임스페이스로 변경합니다.

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

3. **STF** 배포에 대한 **ServiceTelemetry** 오브젝트를 편집합니다.

```
$ oc edit stf default
```

4. **STF**를 사용하여 사용자 정의 **Alertmanager** 경로를 배포하려면 **alertmanagerConfigManifest** 매개변수를 **Service Telemetry Operator**에 전달하여 **Prometheus Operator**에서 관리하는 업데이트된 보안을 생성해야 합니다.

```
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:
Z2xvYmFsOgogIHJlc29sdmVfdGltZW91dDogMTBtCiAgc2xhY2tfYXBpX3VybDogPHNs
YWNrX2FwaV91cmw+CnJlY2VpdmVyczoKICAtIG5hbWU6IHNsYWNrCiAgICBzbGFja19j
```

```
b25maWdzOgogICAgLSBjaGFubmVsOiAjc3RmLWFsZXJ0cwogICAgICB0aXRsZTogfC
0KICAgICAgICAuLi4KICAgICAgdGV4dDogPi0KICAgICAgICAuLi4Kcm91dGU6CiAgZ3J
vdXBfYnk6IFsnam9iJ10KICBncm91cF93YWl0OiAzMHMKICBncm91cF9pbnRlcnZhbDo
gNW0KICByZXBlYXRfaW50ZXJ2YWw6IDEyaAogIHJlY2VpdmVyOiAnc2xhY2snCg==
```

5. 구성이 보안에 적용되었는지 확인합니다.

```
$ 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. alertmanager-proxy 서비스에 대해 curl 명령을 실행하여 status 및 configYAML 콘텐츠를 검색하고 제공된 구성이 Alertmanager의 구성과 일치하는지 확인합니다.

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

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

7. configYAML 필드에 예상 변경 사항이 포함되어 있는지 확인합니다.

8. 환경을 정리하려면 curl 포드를 삭제합니다.

```
$ oc delete pod curl

pod "curl" deleted
```

추가 리소스

- Red Hat OpenShift Container Platform 시크릿 및 Prometheus Operator에 대한 자세한 내용은 경고에 대한 Prometheus 사용자 가이드를 참조하십시오.

## 5.4. SNMP 트랩 구성

STF(Service Telemetry Framework)를 SNMP 트랩을 통해 알림을 수신하는 기존 인프라 모니터링 플랫폼과 통합할 수 있습니다. SNMP 트랩을 활성화하려면 ServiceTelemetry 오브젝트를 수정하고 snmpTraps 매개변수를 구성합니다.

경고 구성에 대한 자세한 내용은 5.3절. “Service Telemetry Framework의 경고” 을 참조하십시오.

사전 요구 사항

- 경고를 보낼 SNMP 트랩 수신자의 IP 주소 또는 호스트 이름을 알고

절차

1. SNMP 트랩을 활성화하려면 ServiceTelemetry 오브젝트를 수정합니다.

```
$ oc edit stf default
```

2. alert.alertmanager.receivers.snmpTraps 매개변수를 설정합니다.

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

3. target 값을 SNMP 트랩 수신자의 IP 주소 또는 호스트 이름으로 설정해야 합니다.

## 5.5. 고가용성

고가용성을 통해 STF(Service Telemetry Framework)는 구성 요소 서비스의 장애에서 빠르게 복구할 수 있습니다. Red Hat OpenShift Container Platform은 노드를 사용하여 워크로드를 예약할 수 있는 경우 실패한 Pod를 재시작하지만 이 복구 프로세스에는 이벤트 및 메트릭이 손실되는 데 1분 이상 걸릴 수 있습니다. 고가용성 구성에는 STF 구성 요소의 여러 사본이 포함되어 있어 복구 시간이 약 2초로 단축됩니다. Red Hat OpenShift Container Platform 노드 장애로부터 보호하려면 3개 이상의 노드가 있는 Red Hat OpenShift Container Platform 클러스터에 STF를 배포합니다.

주의

STF는 아직 완전한 내결함성 시스템이 아닙니다. 복구 기간 중 메트릭 및 이벤트 제공은 보장되지 않습니다.

고가용성을 활성화하면 다음과 같은 이점이 있습니다.

- 기본 Pod 대신 3개의 Elasticsearch Pod가 실행됩니다.
- 다음 구성 요소는 기본 Pod 대신 두 개의 Pod를 실행합니다.
  - AMQ Interconnect
  - Alertmanager
  - Prometheus
  - 이벤트 스마트 게이트웨이
  - 지표 스마트 게이트웨이
- 이러한 서비스 중 하나에서 손실된 포드에서 복구 시간을 약 2초로 줄입니다.

### 5.5.1. 고가용성 구성

고가용성을 위해 **STF(Service Telemetry Framework)**를 구성하려면 **Red Hat OpenShift Container Platform**의 **ServiceTelemetry** 오브젝트에 **highAvailability.enabled: true** 를 추가합니다. 설치시 이 매개변수를 설정하거나 **STF**를 이미 배포한 경우 다음 단계를 완료할 수 있습니다.

**절차**

1. **Red Hat OpenShift Container Platform**에 로그인합니다.

2. **service-telemetry** 네임스페이스로 변경합니다.

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

3. **oc** 명령을 사용하여 **ServiceTelemetry** 오브젝트를 편집합니다.

```
$ oc edit stf default
```

4. **spec** 섹션에 **highAvailability.enabled: true** 를 추가합니다.

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

5. 변경 사항을 저장하고 오브젝트를 닫습니다.

## 5.6. 임시 스토리지

임시 스토리지를 사용하여 **Red Hat OpenShift Container Platform** 클러스터에 데이터를 영구적으로 저장하지 않고 **STF(Service Telemetry Framework)**를 실행할 수 있습니다.

주의

임시 스토리지를 사용하는 경우 Pod가 재시작, 업데이트 또는 다른 노드에 다시 예약되는 경우 데이터 손실이 발생할 수 있습니다. 임시 스토리지는 프로덕션 환경이 아닌 개발 또는 테스트에만 사용합니다.

### 5.6.1. 임시 스토리지 구성

임시 스토리지에 대한 STF 구성 요소를 구성하려면 해당 매개변수에 ...storage.strategy: ephemeral 를 추가합니다. 예를 들어 Prometheus 백엔드의 임시 스토리지를 활성화하려면 backends.metrics.prometheus.storage.strategy: ephemeral. 임시 스토리지 구성을 지원하는 구성 요소에는 alert.alertmanager,backends.metrics.prometheus, backends.events.elasticsearch 가 포함됩니다. 설치시 임시 스토리지 구성을 추가하거나 STF를 이미 배포한 경우 다음 단계를 완료할 수 있습니다.

절차

1. Red Hat OpenShift Container Platform에 로그인합니다.

2. service-telemetry 네임스페이스로 변경합니다.

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

3. ServiceTelemetry 오브젝트를 편집합니다.

```
$ oc edit stf default
```

4. ...storage.strategy: ephemeral 매개변수를 관련 구성 요소의 spec 섹션에 추가합니다.

```
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. 변경 사항을 저장하고 오브젝트를 닫습니다.

## 5.7. SERVICE TELEMETRY FRAMEWORK의 관찰 전략

STF(Service Telemetry Framework)에는 스토리지 백엔드 및 경고 툴이 포함되어 있지 않습니다. STF는 커뮤니티 Operator를 사용하여 Prometheus, Alertmanager, Grafana 및 Elasticsearch를 배포합니다. STF는 이러한 커뮤니티 운영자가 STF를 사용하도록 구성된 각 애플리케이션의 인스턴스를 생성하도록 요청합니다.

Service Telemetry Operator가 사용자 정의 리소스 요청을 생성하는 대신 이러한 애플리케이션 또는 기타 호환 애플리케이션에 대한 자체 배포를 사용하고 지표 Smart Gateway를 스크랩하여 Telemetry 스토리지를 위해 자체 Prometheus 호환 시스템으로 전달할 수 있습니다. 대신 대체 백엔드를 사용하도록 관찰 전략을 설정하면 STF에는 영구 또는 임시 스토리지가 필요하지 않습니다.

### 5.7.1. 대체 관찰 전략 구성

스토리지, 시각화 및 경고 백엔드 배포를 건너뛰도록 STF를 구성하려면 ServiceTelemetry 사양에 observabilityStrategy: none 을 추가합니다. 이 모드에서는 AMQ Interconnect 라우터 및 지표 Smart Gateway만 배포되며 STF Smart Gateway에서 지표를 수집하도록 외부 Prometheus 호환 시스템을 구성해야 합니다.

참고

현재 observabilityStrategy 를 none 으로 설정할 때 메트릭만 지원됩니다. Events Smart Gateways는 배포되지 않습니다.

절차

1. spec 매개변수에 observabilityStrategy: none 속성을 사용하여 ServiceTelemetry 오브젝

트를 생성합니다. 매니페스트는 모든 지표 수집기 유형이 있는 단일 클라우드에서 **Telemetry**를 수신하는 데 적합한 **STF**의 기본 배포가 표시됩니다.

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

2. 모든 워크로드가 올바르게 작동하는지 확인하려면 **Pod** 및 각 **Pod**의 상태를 확인합니다.

```
$ 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
```

추가 리소스

추가 클라우드를 구성하거나 지원되는 컬렉터 집합을 변경하는 방법에 대한 자세한 내용은 을 참조하십시오. 4.5.2절. “스마트 게이트웨이 배포”

# 6장. AMQ INTERCONNECT 인증서 업데이트

인증서가 만료될 때 RHOSP(Red Hat OpenStack Platform)와 STF(Service Telemetry Framework) 간에 AMQ Interconnect 연결을 보호하는 CA 인증서를 정기적으로 갱신해야 합니다. 갱신은 Red Hat OpenShift Container Platform의 cert-manager 구성 요소에서 자동으로 처리되지만 갱신된 인증서를 RHOSP 노드에 수동으로 복사해야 합니다.

## 6.1. 만료된 AMQ INTERCONNECT CA 인증서 확인

CA 인증서가 만료되면 AMQ Interconnect 연결이 계속 작동되지만 중단된 경우 다시 연결할 수 없습니다. 결국 RHOSP(Red Hat OpenStack Platform) 라우터의 일부 또는 모든 연결이 실패하여 양 쪽에 오류가 표시되고 CA 인증서의 만료(또는 "비행 안 함") 필드가 과거임을 알 수 있습니다.

절차

1. Red Hat OpenShift Container Platform에 로그인합니다.

2. service-telemetry 네임스페이스로 변경합니다.

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

3. 일부 또는 모든 라우터 연결이 실패했는지 확인합니다.

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

4. Red Hat OpenShift Container Platform-hosted AMQ Interconnect 로그에서 이 오류가 있는지 확인합니다.

```
$ oc logs -l application=default-interconnect | tail
[...]
2022-11-10 20:51:22.863466 +0000 SERVER (info) [C261] Connection from
10.10.10.10:34570 (to 0.0.0.0:5671) failed: amqp:connection:framing-error SSL Failure:
error:140940E5:SSL routines:ssl3_read_bytes:ssl handshake failure
```

5. RHOSP 언더클라우드에 로그인합니다.

6.

연결이 실패한 노드의 **RHOSP** 호스트 **AMQ Interconnect** 로그에서 이 오류가 있는지 확인합니다.

```
$ ssh controller-0.ctlplane -- sudo tail /var/log/containers/metrics_qdr/metrics_qdr.log
[...]
2022-11-10 20:50:44.311646 +0000 SERVER (info) [C137] Connection to default-interconnect-5671-service-telemetry.apps.mycluster.com:443 failed: amqp:connection:framing-error SSL Failure: error:0A000086:SSL routines::certificate verify failed
```

7. **RHOSP** 노드에서 파일을 검사하여 **CA** 인증서가 만료되었는지 확인합니다.

```
$ ssh controller-0.ctlplane -- cat /var/lib/config-data/puppet-generated/metrics_qdr/etc/pki/tls/certs/CA_sslProfile.pem | openssl x509 -text | grep "Not After"
        Not After : Nov 10 20:31:16 2022 GMT

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

### 6.2. AMQ INTERCONNECT CA 인증서 업데이트

**AMQ Interconnect** 인증서를 업데이트하려면 **Red Hat OpenShift Container Platform**에서 내보내고 **RHOSP(Red Hat OpenStack Platform)** 노드에 복사해야 합니다.

절차

1. **Red Hat OpenShift Container Platform**에 로그인합니다.

2. **service-telemetry** 네임스페이스로 변경합니다.

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

3. **CA** 인증서를 **STFCA.pem** 으로 내보냅니다.

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

4. **STFCA.pem**을 **RHOSP** 언더클라우드에 복사합니다.

5. RHOSP 언더클라우드에 로그인합니다.

6. stf-connectors.yaml 파일을 편집하여 새 caCertFileContent를 포함합니다. 자세한 내용은 4.1.4절. “오버클라우드의 STF 연결 구성”을 참조하십시오.

참고

아래 단계를 수행한 후에는 오버클라우드 배포를 수행할 필요가 없습니다. 향후 배포에서는 새 CA 인증서를 덮어쓰지 않도록 stf-connectors.yaml 파일을 편집합니다.

7. STFCA.pem 파일을 각 RHOSP 오버클라우드 노드에 복사합니다.

```
[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. 각 RHOSP 오버클라우드 노드에서 metrics_qdr 컨테이너를 다시 시작합니다.

```
[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"
```

# 7장. RED HAT OPENSHIFT CONTAINER PLATFORM 환경에서 SERVICE TELEMETRY FRAMEWORK 제거

더 이상 STF 기능이 필요하지 않은 경우 Red Hat OpenShift Container Platform 환경에서 STF(Service Telemetry Framework)를 제거합니다.

Red Hat OpenShift Container Platform 환경에서 STF를 제거하려면 다음 작업을 수행해야 합니다.

1. 네임스페이스를 삭제합니다.
2. 카탈로그 소스를 제거합니다.
3. cert-manager Operator를 제거합니다.

## 7.1. 네임스페이스 삭제

Red Hat OpenShift Container Platform에서 STF의 운영 리소스를 제거하려면 네임스페이스를 삭제합니다.

절차

1. oc delete 명령을 실행합니다.

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

2. 리소스가 네임스페이스에서 삭제되었는지 확인합니다.

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

## 7.2. CATALOGSOURCE 제거

STF(Service Telemetry Framework)를 다시 설치할 것으로 예상하지 않으면 CatalogSource를 삭제합니다. CatalogSource를 제거하면 STF와 관련된 PackageManifests가 Operator Lifecycle Manager

카탈로그에서 자동으로 제거됩니다.

절차

1. 설치 프로세스 중에 OperatorHub.io 커뮤니티 카탈로그 소스를 활성화했으며 더 이상 이 카탈로그 소스가 필요하지 않은 경우 삭제합니다.

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

추가 리소스

OperatorHub.io 커뮤니티 카탈로그 소스에 대한 자세한 내용은 3.1절. “Red Hat OpenShift Container Platform 환경에 Service Telemetry Framework 배포” 을 참조하십시오.

### 7.3. CERT-MANAGER OPERATOR 제거

다른 애플리케이션에 cert-manager Operator를 사용하지 않는 경우 Subscription, ClusterServiceVersion 및 CustomResourceDefinitions를 삭제합니다.

절차

1. openshift-cert-manager-operator 네임스페이스에서 서브스크립션을 삭제합니다.

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

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

2. 설치된 ClusterServiceVersion의 버전 번호를 검색합니다.

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

출력 예:

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

3. openshift-cert-manager-operator 네임스페이스에서 ClusterServiceVersion을 삭제합니

다.

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

출력 예:

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

4. Operator에서 제공하는 CustomResourceDefinitions의 현재 목록을 가져와서 ClusterServiceVersion을 제거한 후 삭제할 수 있습니다.

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

출력 예:

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

5. cert-manager Operator와 관련된 CustomResourceDefinitions를 삭제합니다.

```
$ 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
```

출력 예:

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

```
deleted
customresourcedefinition.apiextensions.k8s.io "clusterissuers.cert-manager.io"
deleted
customresourcedefinition.apiextensions.k8s.io "issuers.cert-manager.io" deleted
customresourcedefinition.apiextensions.k8s.io "orders.acme.cert-manager.io" deleted
```

6. cert-manager Operator가 소유한 네임스페이스를 삭제합니다.

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

출력 예:

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

추가 정보

- 클러스터에서 Operator 삭제.