Red Hat OpenStack Platform 16.1

모니터링 툴 구성 가이드

OpenStack 로깅 및 모니터링 툴 가이드

Last Updated: 2023-12-06

Red Hat OpenStack Platform 16.1 모니터링 툴 구성 가이드

OpenStack 로깅 및 모니터링 툴 가이드

OpenStack Team
rhos-docs@redhat.com

법적 공지

초록

이 가이드에서는 Red Hat OpenStack Platform 환경의 로깅 및 모니터링을 구성하는 방법에 대해 설명합니다.

차례

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 이를 개선하는지 알려주십시오.

DDF(직접 문서 피드백) 기능 사용

특정 문장, 단락 또는 코드 블록에 대한 직접 주석은 **피드백 추가** DDF 기능을 사용하십시오.

1. *다중 페이지 HTML* 형식으로 설명서를 봅니다.
2. 문서 오른쪽 상단에 **Feedback** (피드백) 버튼이 표시되는지 확인합니다.
3. 주석 처리하려는 텍스트 부분을 강조 표시합니다.
4. **피드백 추가**를 클릭합니다.
5. 주석을 사용하여 **Add Feedback** (피드백 추가) 필드를 작성합니다.
6. 선택 사항: 설명서 팀이 문제에 대한 자세한 내용을 문의할 수 있도록 이메일 주소를 추가하십시오.
7. **Submit(제출)**을 클릭합니다.

1장. RED HAT OPENSTACK PLATFORM 모니터링 툴 소개

모니터링 툴은 운영자가 OpenStack 환경을 유지 관리할 수 있도록 설계된 선택적 툴 제품군입니다. 도구는 다음 기능을 수행합니다.

- 중앙 집중식 로깅: 하나의 중앙 위치에 있는 OpenStack 환경의 모든 구성 요소에서 로그를 수집합니다. 모든 노드 및 서비스에서 문제를 식별하고 필요한 경우 문제 진단 지원을 위해 로그 데이터를 Red Hat으로 내보낼 수 있습니다.
- 가용성 모니터링: OpenStack 환경의 모든 구성 요소를 모니터링하고 현재 중단이 발생하고 있거나 다른 구성 요소가 작동하지 않는지 확인합니다. 문제가 식별될 때 경고를 표시하도록 시스템을 구성할 수도 있습니다.

1.1. 모니터링 구성 요소의 상태 지원

지원 상태 표를 사용하여 Red Hat OpenStack Platform에서 구성 요소의 지원 상태를 확인합니다.

표 1.1. 지원 상태

구성 요소	이후 완전하게 지원됨	더 이상 사용되지 않음	이후 삭제	참고
Aodh	OSP 9	OSP 15		16.1의 자동 스케일링에 사용됩니다.
Ceilometer	OSP 4			
collectd	OSP 11			
Gnocchi	OSP 9	OSP 15		
Panko	OSP 11	OSP 12, OSP 14 이후 기본적으로 설치되지 않음	OSP 16.1	Cloudforms에 대한 필수 16.1까지
osops-tools-monitoring-oscheck		OSP 14		

2장. 아키텍처 모니터링

모니터링 툴은 Red Hat OpenStack Platform Overcloud 노드에 배포된 클라이언트와 함께 클라이언트-서버 모델을 사용합니다. Rsyslog 서비스는 클라이언트 측 중앙 집중식 로깅(CL)을 제공하며, 활성화된 의미 플러그인이 있는 collectd는 클라이언트 측 가용성 모니터링(AM)을 제공합니다.

2.1. 중앙 집중식 로깅

Red Hat OpenStack 환경에서 하나의 중앙 위치에서 모든 서비스에서 로그를 수집하면 디버깅 및 관리를 단순화합니다. 이러한 로그는 syslog 및 감사 로그 파일, RabbitMQ 및 MariaDB와 같은 인프라 구성 요소, ID, 계산 등의 OpenStack 서비스와 같은 운영 체제에서 가져옵니다.

중앙 집중식 로깅 툴체인은 다음 구성 요소로 구성됩니다.

- 로그 컬렉션 에이전트(Rsyslog)
- 데이터 저장소 (Elasticsearch)
- API/Presentation Layer(Kibana)

참고

Red Hat OpenStack Platform director는 중앙 집중식 로깅을 위한 서버 측 구성 요소를 배포하지 않습니다. Red Hat은 Elasticsearch 데이터베이스 및 Kibana를 포함한 서버 측 구성 요소를 지원하지 않습니다.

2.2. 가용성 모니터링

가용성 모니터링을 통해 전체 OpenStack 환경에서 모든 구성 요소의 높은 수준의 기능을 모니터링할 수 있는 하나의 중앙 지점이 있습니다.

가용성 모니터링 툴체인은 다음과 같은 여러 구성 요소로 구성됩니다.

- 모니터링 에이전트(활성화된 의미 플러그인으로 수집)
- 모니터링 중계/Proxy(RabbitMQ)
- 컨트롤러/서버 모니터링(Sensu 서버)
- API/Presentation Layer(Uchiwa)

참고

Red Hat OpenStack Platform director는 가용성 모니터링을 위한 서버 측 구성 요소를 배포하지 않습니다. Red Hat은 Uchiwa, Sensu Server, Sensu API 및 RabbitMQ, 모니터링 노드에서 실행되는 Redis 인스턴스를 비롯한 서버 측 구성 요소를 지원하지 않습니다.

가용성 모니터링 구성 요소 및 상호 작용은 다음 다이어그램에 설명되어 있습니다.

참고

파란색으로 표시된 항목은 Red Hat 지원 구성 요소를 나타냅니다.

그림 2.1. 높은 수준의 가용성 모니터링 아키텍처

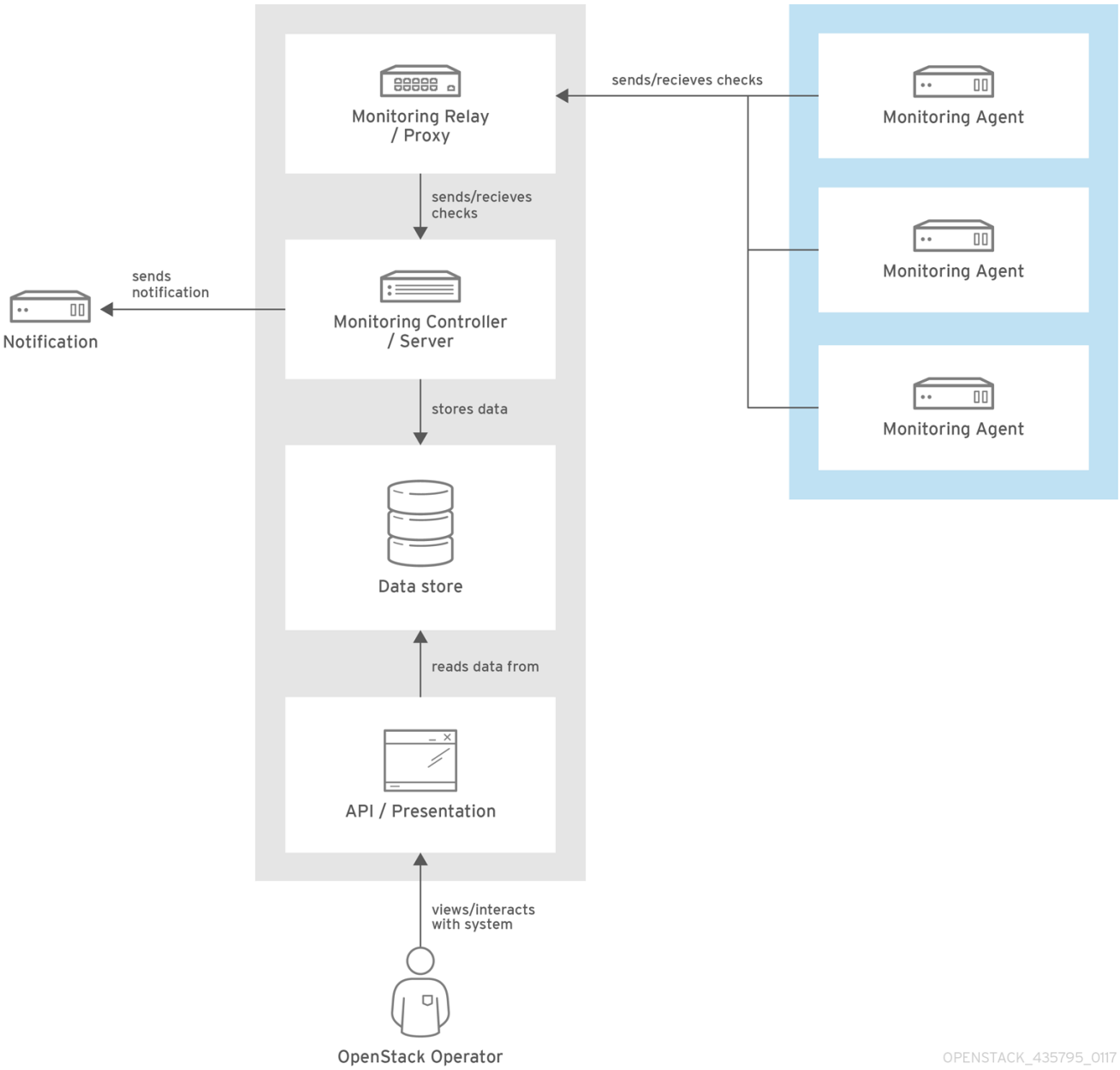


그림 2.2. Red Hat OpenStack Platform용 단일 노드 배포

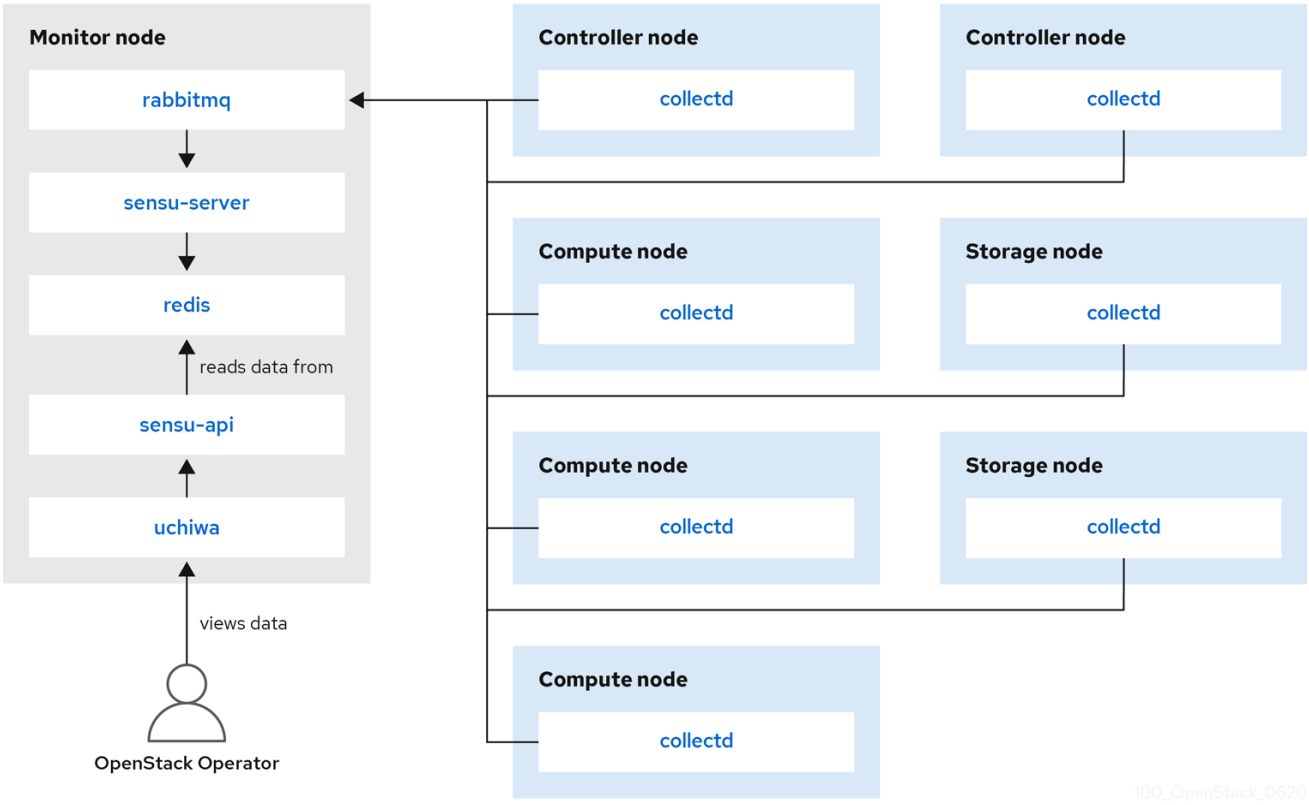
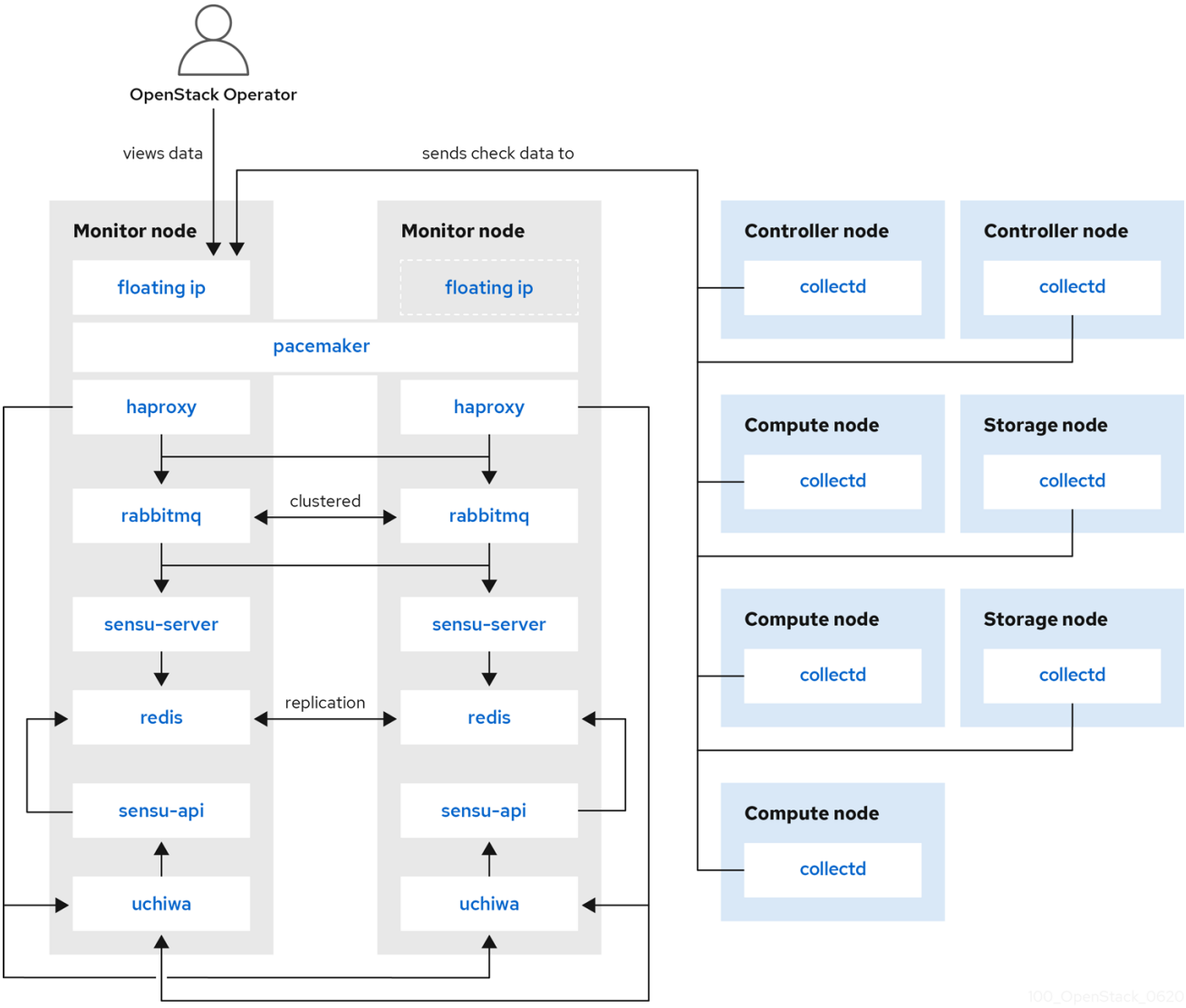


그림 2.3. Red Hat OpenStack Platform용 HA 배포



3장. 클라이언트측 도구 설치

오버클라우드를 배포하기 전에 각 클라이언트에 적용할 구성 설정을 결정해야 합니다. heat 템플릿 컬렉션의 예제 환경 파일을 복사하고 환경에 맞게 파일을 수정합니다.

3.1. 중앙 집중식 로깅 클라이언트 매개변수 설정

자세한 내용은 *로깅 , 모니터링 및 문제 해결 가이드에서 Elasticsearch를 사용하여 중앙 집중식 로깅 활성화* 를 참조하십시오.

3.2. 모니터링 클라이언트 매개변수 설정

모니터링 솔루션은 시스템 정보를 정기적으로 수집하고 데이터 수집 에이전트를 사용하여 다양한 방식으로 값을 저장하고 모니터링하는 메커니즘을 제공합니다. Red Hat은 수집 에이전트로 collectd을 지원합니다. collectd-sensubility는 collectd의 범위이며 RabbitMQ를 통해 Sensu 서버 측과 통신합니다.
STF(Service Telemetry Framework)를 사용하여 데이터를 저장하고, 시스템을 모니터링하고, 성능 병목을 찾고, 향후 시스템 부하를 예측할 수 있습니다. Service Telemetry Framework에 대한 자세한 내용은 Service Telemetry Framework 1.3 가이드를 참조하십시오.

collectd 및 collectd-sensubility를 구성하려면 다음 단계를 완료합니다.

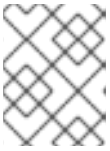
1. 홈 디렉터리(예: **/home/templates/custom**)에 **config.yaml** 을 생성하고 STF 서버 측을 가리키도록 **MetricsQdrConnectors** 매개변수를 구성합니다.

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

2. **config.yaml** 파일의 **CollectdExtraPlugins** 아래에 사용할 플러그인을 나열합니다. **ExtraConfig** 섹션에서 매개 변수를 제공할 수도 있습니다. 기본적으로 collectd는 **cpu,df, disk,hugepages,인터페이스 , 로드,메모리,프로세스,tcpconns,unixsock** 및 **가동 시간** 플러그인과 함께 제공됩니다. **CollectdExtraPlugins** 매개변수를 사용하여 추가 플러그인을 추가할 수 있습니다. **ExtraConfig** 옵션을 사용하여 **CollectdExtraPlugins** 에 대한 추가 구성 정보를 제공할 수도 있습니다. 예를 들어 **virt** 플러그인을 활성화하고 연결 문자열과 호스트 이름 형식을 구성하려면 다음 구문을 사용하십시오.

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

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



참고

unixsock 플러그인을 제거하지 마십시오. 제거는 비정상으로 collectd 컨테이너를 영구적으로 표시합니다.

3. 선택 사항: AMQ 상호 연결을 통해 지표 및 이벤트 데이터를 수집하려면 **config.yaml 파일에 MetricsQdrExternalEndpoint: true 행을 추가합니다.**

```
parameter_defaults:
  MetricsQdrExternalEndpoint: true
```

4. collectd-sensubility를 활성화하려면 **config.yaml 파일에 다음 환경 구성을 추가합니다.**

```
parameter_defaults:
  CollectdEnableSensubility: true

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

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

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

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

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

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

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

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

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

5. Overcloud를 배포합니다. 오버클라우드 배포 명령에 **config.yaml**,**collectd-write-qdr.yaml** 및 **qdr-*.yaml** 파일 중 하나를 포함합니다.

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

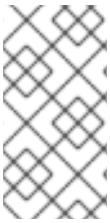
6. 선택 사항: Overcloud RabbitMQ 모니터링을 활성화하려면 **overcloud deploy** 명령에 **collectd-read-rabbitmq.yaml** 파일을 포함합니다.

추가 리소스

- YAML 파일에 대한 자세한 내용은 3.5절. "YAML 파일" 을 참조하십시오.
- collectd 플러그인에 대한 자세한 내용은 3.4절. "collectd 플러그인 구성" 의 내용을 참조하십시오.
- Service Telemetry Framework에 대한 자세한 내용은 Service Telemetry Framework 1.3 가이드를 참조하십시오.

3.3. AMQ INTERCONNECT를 통한 데이터 수집

지표 및 이벤트 데이터 사용량에 사용할 수 있는 AMQ 상호 연결 주소를 구독하려면 환경 파일을 생성하여 클라이언트 연결에 대해 AMQ 상호 연결을 노출하고 Overcloud를 배포합니다.



참고

Service Telemetry Operator는 단일 클라우드 배포를 위해 모든 데이터 수집 및 데이터 스토리지 구성 요소를 간단하게 배포할 수 있습니다. 데이터 스토리지 도메인을 여러 클라우드와 공유하려면 *Service Telemetry Framework 1.3* 가이드에서 여러 클라우드 구성을 참조하십시오.

주의

STF(Service Telemetry Framework)에서 사용하는 QDR 메시 모드와 QDR 엣지 모드는 전환할 수 없습니다. 또한 STF에 대한 데이터 수집을 활성화하는 경우 QDR 메시 모드를 사용할 수 없습니다.

절차

1. **stack** 사용자로 Red Hat OpenStack Platform 언더클라우드에 로그인합니다.
2. **/home/stack** 디렉터리에 **data-collection.yaml** 이라는 구성 파일을 생성합니다.
3. 외부 끝점을 활성화하려면 **MetricsQdrExternalEndpoint: true** 매개변수를 **data-collection.yaml** 파일에 추가합니다.

```
parameter_defaults:
  MetricsQdrExternalEndpoint: true
```

4. collectd 및 AMQ 상호 연결을 활성화하려면 Red Hat OpenStack Platform director 배포에 다음 파일을 추가합니다.
   - **data-collection.yaml** 환경 파일
   - 클라이언트 측 AMQ 상호 연결을 통해 외부 엔드포인트에 연결할 수 있는 **qdr-form-controller-mesh.yaml** 파일

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

5. 선택 사항: Ceilometer 및 collectd 이벤트를 수집하려면 **오버클라우드 배포** 명령에 **ceilometer-write-qdr.yaml** 및 **collectd-write-qdr.yaml** 파일을 포함합니다.
6. Overcloud를 배포합니다.

추가 리소스

- YAML 파일에 대한 자세한 내용은 3.5절. "YAML 파일" 을 참조하십시오.

3.4. COLLECTD 플러그인 구성

Red Hat OpenStack Platform director의 구성 가능성은 다양합니다. 환경에 맞게 여러 collectd 플러그인을 구성할 수 있습니다. 문서화된 각 플러그인에는 설명 및 예제 구성이 있습니다. 일부 플러그인에는 Grafana 또는 Prometheus에서 쿼리할 수 있는 메트릭 테이블과 가능한 경우 구성할 수 있는 옵션 목록이 있습니다.

추가 리소스

- collectd 플러그인 옵션의 전체 목록을 보려면 *Service Telemetry Framework* 가이드의 collectd 플러그인을 참조하십시오.

3.5. YAML 파일

collectd를 구성할 때 **오버클라우드 배포** 명령에 다음 YAML 파일을 포함할 수 있습니다.

- **collectd-read-rabbitmq.yaml**: Overcloud RabbitMQ 인스턴스를 모니터링하도록 **python-collect-rabbitmq** 를 활성화하고 구성합니다.

- **collectd-write-qdr.yaml**: collectd가 AMQ Interconnect를 통해 원격 분석 및 알림 데이터를 보낼 수 있도록 합니다.

- **qdr-edge-only.yaml**: AMQ Interconnect를 배포할 수 있습니다. 각 오버클라우드 노드에는 엣지 모드에서 실행 중이고 작동 중인 하나의 로컬 qdrouterd 서비스가 있습니다. 예를 들어 수신된 데이터를 정의된 **MetricsQdrConnectors** 로 직접 보냅니다.

- **qdr-form-controller-mesh.yaml**: AMQ Interconnect를 배포할 수 있습니다. 각 오버클라우드 노드에는 메시 토폴로지를 구성하는 하나의 로컬 qdrouterd 서비스가 있습니다. 예를 들어 컨트롤러의 AMQ 상호 연결 라우터는 정의된 **MetricsQdrConnectors** 및 다른 노드 유형의 AMQ 상호 연결 라우터와 함께 내부 라우터 모드에서 작동하며 컨트롤러에서 실행되는 내부 라우터에 에지 모드로 연결합니다.

추가 리소스

collectd 구성에 대한 자세한 내용은 3.2절. "모니터링 클라이언트 매개변수 설정" 의 내용을 참조하십시오.