Red Hat OpenStack Platform 13

操作测量

跟踪物理和虚拟资源并收集指标

Last Updated: 2023-10-20

Red Hat OpenStack Platform 13 操作测量

跟踪物理和虚拟资源并收集指标

OpenStack Team
rhos-docs@redhat.com

法律通告

Copyright © 2023 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

摘要

使用操作工具帮助您测量和维护 Red Hat OpenStack Platform 环境。

目录

第 1 章 操作测量简介

在 Red Hat OpenStack Platform 环境中使用遥测服务的组件，您可以跟踪物理和虚拟资源，并使用在 Gnocchi 后端上存储聚合的数据集合守护进程收集部署中 CPU 使用率和资源可用性等指标。

1.1. 了解运营测量

使用操作工具帮助您测量和维护 Red Hat OpenStack Platform 环境。这些测量工具执行以下功能：

- **可用性监控：监控** Red Hat OpenStack Platform (RHOSP)环境中的所有组件，并确定任何组件当前出现停机或无法正常工作。您还可以将系统配置为在发现问题时提醒您。
- **性能监控** ：定期收集系统信息并提供一种机制，使用数据收集守护进程以各种方式存储和监控值。此守护进程存储它收集的数据，如操作系统和日志文件，或者通过网络提供数据。您可以使用从数据收集的统计信息来监控系统、查找性能瓶颈和预测将来的系统负载。

1.2. TELEMETRY 架构

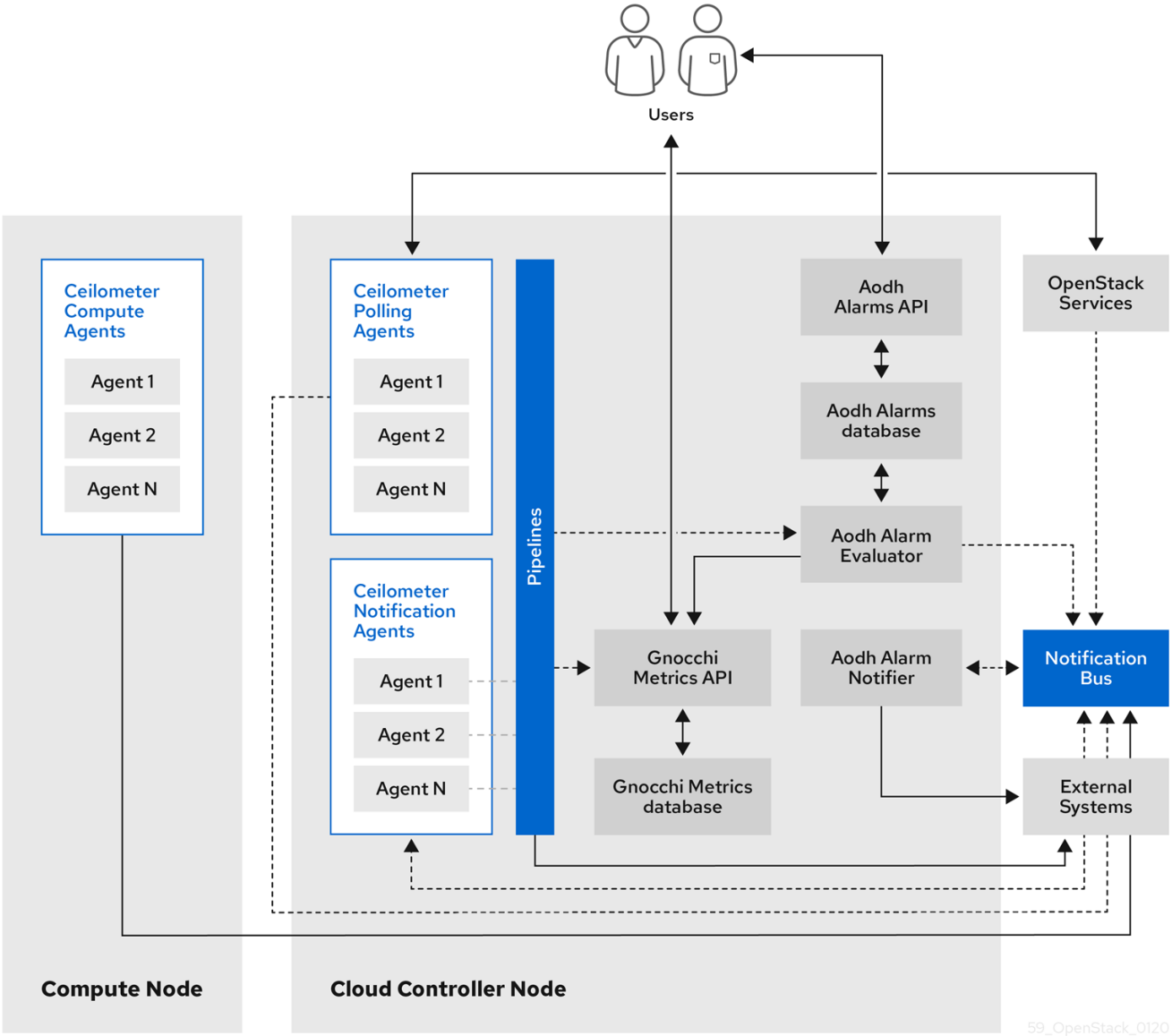
Red Hat OpenStack Platform (RHOSP) Telemetry 为基于 OpenStack 的云提供用户级使用情况数据。您可以使用这些数据进行客户计帐、系统监控或警报。您可以配置 Telemetry 组件从现有 RHOSP 组件发送的通知中收集数据，如计算使用情况事件，或者轮询 RHOSP 基础架构资源，如 libvirt。Telemetry 将收集的数据发布到包括数据存储和消息队列的各种目标。

Telemetry 由以下组件组成：

- **数据收集** ：Telemetry 使用 Ceilometer 收集指标和事件数据。更多信息请参阅 第 1.3.1 节 “Ceilometer”。
- **存储** ：Telemetry 将指标数据存储在 Gnocchi 中，并将事件数据存储在 Panko 中。更多信息请参阅 第 1.4 节 “使用 Gnocchi 存储”。
- **警报服务** ：Telemetry 使用 Aodh 来基于定义的规则根据 Ceilometer 收集的指标或事件数据来触发操作。

收集数据时，您可以使用第三方工具（如 Red Hat Cloudforms）来显示和分析指标数据，您可以使用警报服务 Aodh 配置事件的警报。

图 1.1. Telemetry 架构



1.3. 数据收集

Red Hat OpenStack Platform (RHOSP)支持两种类型的数据收集：

- collectd 用于基础架构监控。更多信息请参阅 第 1.3.2 节 “collectd”。
- Ceilometer 用于 OpenStack 组件级监控。更多信息请参阅 第 1.3.1 节 “Ceilometer”。

1.3.1. Ceilometer

Ceilometer 是 OpenStack 遥测服务的默认数据收集组件，能够在所有 OpenStack 核心组件中规范化和转换数据。Ceilometer 收集与 OpenStack 服务相关的计量和事件数据。用户可以根据部署配置访问收集的数据。

Ceilometer 服务使用三个代理从 Red Hat OpenStack Platform (RHOSP)组件收集数据：

- **计算代理(ceilometer-agent-compute)**：在每个 Compute 节点上运行，并轮询资源利用率统计。此代理与轮询代理 **ceilometer-polling** 与使用参数 **--polling namespace-compute** 一起运行相同。

- **中央代理(ceilometer-agent-central)**：在中央管理服务器上运行，以轮询不与实例或计算节点关联的资源利用率统计。您可以启动多个代理来横向扩展服务。这与轮询代理 **ceilometer-polling** 使用参数 **--polling namespace-central** 运行。

- **通知代理(ceilometer-agent-notification)**：在中央管理服务器上运行，并使用来自消息队列的消息来构建事件和计量数据。然后将数据发布到定义的目标。默认情况下，数据被推送到 Gnocchi。这些服务使用 RHOSP 通知总线进行通信。

Ceilometer 代理使用发布者将数据发送到对应的端点，如 Gnocchi。您可以在 **pipeline.yaml** 文件中配置这些信息。

其他资源

- 有关发布程序的更多信息，请参阅 第 1.3.1.1 节 "publishers"。

1.3.1.1. publishers

遥测服务提供多种传输方法，用于将收集的数据转让给外部系统。此数据的使用者则不同，例如，监控系统可接受数据丢失，以及计费系统，这需要可靠的数据传输。Telemetry 提供了一种方法来减轻这两种系统类型的要求。您可以使用服务的发布组件将数据通过消息总线将数据保存到持久性存储中，或将其发送到一个或多个外部用户。个链可以包含多个发布程序。

支持以下发布程序类型：

- **Gnocchi （默认）**：当启用 Gnocchi 发布程序时，指标和资源信息将推送到 Gnocchi，以优化时间序列。确保您在身份服务中注册了 Gnocchi，因为 Ceilometer 通过身份服务发现确切的路径。

- Panko：您可以在 **panko** 中存储 Ceilometer 的事件数据，该界面提供了一个 HTTP REST 接口，以在 Red Hat OpenStack Platform 中查询系统事件。要将数据推送到 panko，请将发布器设置为 **direct://?dispatcher=panko**。

1.3.1.1.1. 配置发布者参数

您可以为遥测服务内每个数据点配置多发布程序，使相同的技术计量或事件多次发布到多个目的地，每个都可能使用不同的传输方法。

流程

1. 创建一个 YAML 文件来描述可能的发布程序参数和默认值，如 **ceilometer-publisher.yaml**。在 **parameter_defaults** 中插入以下参数：

```
parameter_defaults:

ManagePipeline: true
ManageEventPipeline: true
EventPipelinePublishers:
- gnocchi://?archive_policy=high
PipelinePublishers:
- gnocchi://?archive_policy=high
```

2. 部署您的定制 overcloud。部署 overcloud 的方法有两种：

   - 在 **openstack overcloud deploy** 命令中包含修改后的 YAML 文件，以定义发布者。在以下示例中，将 &**lt;environment-files** > 替换为您要包含在部署中的其他 YAML 文件：

```
$ openstack overcloud  deploy
--templates \
-e /home/custom/ceilometer-publisher.yaml
-e <environment-files>
```

- 创建一个 YAML 文件来包括所有本地修改，例如 **local_modifications.yaml**。您可以使用脚本执行部署，如下例所示：

```
$ sh deploy.sh:

#!/bin/bash
openstack overcloud deploy
-e <environment-files> \
-e local_modifications.yaml \
....
```

其他资源

- 有关参数的更多信息，请参阅《*高级 Overcloud 自定义指南》中的 Overcloud* 参数 和参数中的 遥测 参数。

1.3.2. collectd

性能监控会定期收集系统信息，并提供一种机制，使用收集的数据代理以各种方式存储和监控值。红帽支持 collectd 守护进程作为数据收集代理。此守护进程将数据存储在时间序列数据库中。红帽支持的数据库之一称为 Gnocchi。您可以使用此存储的数据来监控系统、查找性能瓶颈和预测将来的系统负载。

其他资源

- 有关 Gnocchi 的更多信息，请参阅 第 1.4 节 “使用 Gnocchi 存储”。
- 有关 collectd 的更多信息，请参阅 第 3.1 节 “安装 collectd”。

1.4. 使用 GNOCCHI 存储

Gnocchi 是开源时间序列数据库。它以非常大的规模存储指标，并提供对操作器和用户的指标和资源的访问权限。Gnocchi 使用归档策略来定义计算和要保留的聚合数；以及索引器驱动程序，以存储所有资源、归档策略和指标的索引。

1.4.1. 归档策略：存储时间序列数据库中的短期和长期数据

归档策略定义了计算的聚合以及要保留的聚合数量。Gnocchi 支持不同的聚合方法，如最小值、最大值、平均数、Nthile 和标准偏差。这些聚合会在特定时间段的时间段内计算名为 granularity 并保留的。

归档策略定义了指标的聚合方式，以及它们的存储时间。每个归档策略定义为 timespan 的点数。

例如，如果您的归档策略定义了粒度为 1 秒的 10 点策略，则时间序列存档最多保留 10 秒，各自代表一个超过 1 秒的聚合。这意味着，时间序列的最大值为上限，在最近点和旧点之间保留 10 秒的数据。

归档策略也定义使用哪些聚合方法。默认设置为参数 **default_aggregation_methods**，其默认值为 mean、min、max. sum、std、count。因此，根据用例，归档策略和粒度会有所不同。

其他资源

- 有关归档策略的更多信息，请参阅 规划和管理归档策略。

1.4.2. 索引器驱动程序

索引器负责存储所有资源、归档策略和指标的索引及其定义、类型和属性。它还负责将资源与指标链接。Red Hat OpenStack Platform director 默认安装 indexer 驱动程序。您需要一个数据库来索引 Gnocchi 处理的所有资源和指标。支持的驱动程序是 MySQL。

1.4.3. Gnocchi Metric-as-a-Service 术语

下表包含 Metric-as-a-Service 功能常用术语的定义。

表 1.1. metric-as-a-Service 术语

术语	定义
聚合方法	种函数用于将多个测量结果聚合到聚合中。例如，min 聚合方法将不同测量结果的值聚合到时间范围中所有测量结果的最小值。
aggregate	根据归档策略，从多个测量结果生成的数据点元数据。聚合由时间戳和值组成。
归档策略	附加到指标的聚合存储策略。归档策略决定聚合在指标中保留的时长，以及如何聚合聚合（聚合方法）。
granularity	在一系列指标的聚合中两个聚合之间的时间。
测量	API 发送到时间序列数据库的传入数据点元。测量由时间戳和值组成。
指标	由 UUID 标识的实体。指标可以使用名称附加到资源。指标存储其聚合的方式由与指标关联的归档策略定义。
资源	代表您与指标关联的任何实体。资源由唯一 ID 标识，可以包含属性。
时间序列	按时间排序的聚合列表。
timespan	指标保留其聚合的时间周期。它用于归档策略的上下文。

1.5. 显示指标数据

您可以使用以下工具显示和分析指标数据：

- **Grafana** ：开源指标分析和视觉化套件。Grafana 最常用于可视化时间序列数据以进行基础架构和应用程序分析。
- **红帽 CloudForms** ：IT 部门用来控制用户自助式服务功能的基础架构管理平台，以调配、管理并确保虚拟机及私有云中的合规性。

其他资源

- 有关 Grafana 的更多信息，请参阅 第 1.5.1 节 “使用并连接 Grafana 来显示数据”。

- 有关 Red Hat Cloudforms 的更多信息，请参阅 产品文档。

1.5.1. 使用并连接 Grafana 来显示数据

您可以使用第三方软件（如 Grafana）来查看收集和存储的指标的图形化表示。

Grafana 是一个开源指标分析、监控和视觉化套件。要安装和配置 Grafana，请参阅官方 Grafana 文档。

第 2 章 规划运营测量

您监控的资源取决于您的业务需求。您可以使用 Ceilometer 或 collectd 来监控您的资源。

- 如需有关 collectd 测量的更多信息，请参阅 第 2.2 节 “collectd 测量”。
- 有关 Ceilometer 测量的更多信息，请参阅 第 2.1 节 “Ceilometer 测量”。

2.1. CEILOMETER 测量

有关 Ceilometer 测量的完整列表，请参阅
https://docs.openstack.org/ceilometer/queens/admin/telemetry-measurements.html

2.2. COLLECTD 测量

以下测量是最常用的 collectd 指标：

- disk
- interface
- load
- 内存
- 进程
- tcpconns

有关测量结果的完整列表，请参阅 collectd 指标和事件。

2.3. 监控 GNOCCHI 和 CEILOMETER 性能

您可以使用 **openstack metric** 命令管理部署的归档策略、基准测试、测量、指标和资源。

流程

- 在命令行中输入 **openstack metric status**，以监控部署中 Gnocchi 安装并检查测量的状态：

```
(overcloud) [stack@undercloud-0 ~]$ openstack metric status
+-----------------------------------------------------+-------+
| Field                                               | Value |
+-----------------------------------------------------+-------+
| storage/number of metric having measures to process | 0     |
| storage/total number of measures to process         | 0     |
+-----------------------------------------------------+-------+
```

2.4. 规划数据存储

Gnocchi 存储数据点的集合，其中每个数据点都是一个聚合的。存储格式使用不同的技术进行压缩。因此，为了计算时间序列数据库的大小，您可以根据最糟糕的情况来估算大小。

流程

1. 计算数据点数：
   point 数量 = timespan / granularity

   例如，如果要使用一分钟分辨率保留一个年的数据，请使用公式：

   数据点数 = (365 天 24 小时 X 60 分钟)/1 分钟的数据点数 = 525600

2. 计算时间序列数据库的大小：
   字节大小 = 数据点 X 8 字节数

   如果您将这个公式应用到示例，则结果为 4.1 MB：

   字节大小 = 525600 点 X 8 字节 = 4204800 字节 = 4.1 MB

   这个值是一个单一聚合时间序列数据库的估算存储要求。如果您的归档策略使用多个聚合方法 (min、max、mean、ided、std、std 和 count)，则按您使用的聚合方法数乘以该值。

其他资源

- 更多信息请参阅 第 1.4.1 节 “归档策略：存储时间序列数据库中的短期和长期数据”。

2.5. 规划和管理归档策略

归档策略定义了您如何聚合指标，以及将指标存储在时间序列数据库中的时长。归档策略定义为 timespan 的点数。

如果您的归档策略定义了粒度为 1 秒的 10 点策略，则时间序列存档最多保留 10 秒，各自代表 1 秒的聚合。这意味着，时间序列在最近点和旧点之间保留最多 10 秒的数据。归档策略也定义要使用的聚合方法。默认设置为参数 **default_aggregation_methods**，其中默认值设置为 **mean**、**min**、**max**。总、**std**、**count**.因此，根据用例，归档策略和粒度可能会有所不同。

要计划归档策略，请确定您熟悉以下概念：

- 指标.更多信息请参阅 第 2.5.1 节 “指标”。
- 测量结果。更多信息请参阅 第 2.5.2 节 “创建自定义测量结果”。
- 聚合.更多信息请参阅 第 2.5.4 节 “计算时间序列聚合的大小”。
- 指标的 worker。更多信息请参阅 第 2.5.5 节 “metricd worker”。

要创建和管理归档策略，请完成以下任务：

1. 创建归档策略。更多信息请参阅 第 2.5.6 节 “创建归档策略”。
2. 管理归档策略。更多信息请参阅 第 2.5.7 节 “管理归档策略”。
3. 创建归档策略规则。更多信息请参阅 第 2.5.8 节 “创建归档策略规则”。

2.5.1. 指标

Gnocchi 提供名为 *metric* 的对象类型。指标是您可以测量的任何内容，例如，服务器的 CPU 使用量、空间温度或网络接口发送的字节数。指标具有以下属性：

- 用于标识它的 UUID

- 一个名称
- 用于存储和聚合测量结果的归档策略

其他资源

- 有关术语定义，请参阅 Gnocchi Metric-as-a-Service 术语。

2.5.1.1. 创建指标

流程

1. 创建资源。将 <resource_name> 替换为资源名称：

```
$ openstack metric resource create <resource_name>
```

2. 创建指标。将 <resource_name> 替换为资源名称，将 <metric_name> 替换为指标的名称：

```
$ openstack metric metric create -r <resource_name> <metric_name>
```

在创建指标时，归档策略属性会被修复并不可更改。您可以通过 **archive_policy** 端点更改归档策略的定义。

2.5.2. 创建自定义测量结果

测量 API 发送到 Gnocchi 的传入数据点数。它由一个时间戳和值组成。您可以创建自己的自定义测量结果。

流程

- 创建自定义测量：

```
$ openstack metric measures add -m <MEASURE1> -m <MEASURE2> .. -r
<RESOURCE_NAME> <METRIC_NAME>
```

2.5.3. 默认归档策略

默认情况下，Gnocc 具有以下归档策略：

- 低
  - 5 分钟粒度超过 30 天
  - 使用的聚合方法： **default_aggregation_methods**
  - 每个指标的最大估算大小： 406 KiB
- 中
  - 1 分钟粒度超过 7 天
  - 365 天 1 小时粒度
  - 使用的聚合方法： **default_aggregation_methods**

- 每个指标的最大估算大小：887 KiB

- high
  - 1 小时第二个粒度
  - 1 周的 1 分钟粒度
  - 1 小时粒度超过 1 年
  - 使用的聚合方法： **default_aggregation_methods**
  - 每个指标的最大预计大小：1 057 KiB

- bool
  - 1 年超过 1 秒的粒度
  - 使用的聚合方法：最后
  - 每个指标的最大优化大小：1 539 KiB
  - 每个指标的最大 pessimistic 大小：277 172 KiB

2.5.4. 计算时间序列聚合的大小

Gnocchi 存储一系列数据点，其中每个点都是一个聚合的。存储格式使用不同的技术进行压缩。因此，根据最糟糕的情况，计算时间序列的大小估计是估算的，如下例所示。

流程

1. 使用此公式计算点数：
   point 数量 = timespan / granularity

   例如，如果要使用一分钟分辨率保留一年的数据：

   点数 = (365 天 X 24 小时 X 60 分钟)/1 分钟

   点数 = 525600

2. 要计算点大小（以字节为单位），请使用这个公式：
   字节大小（以字节为单位） = 点 X 8 字节数

   字节大小 = 525600 点 X 8 字节 = 4204800 字节 = 4.1 MB

   这个值是一个单一聚合时间序列的估算存储要求。如果您的归档策略使用多个聚合方法 - 分钟、max、mean、sted、std、count - 按您使用的聚合方法数乘以该值。

2.5.5. metricd worker

您可以使用 metricd 守护进程来处理测量结果，创建聚合、存储测量结果，以及删除指标。metricd 守护进程负责 Gnocchi 中大多数 CPU 用量和 I/O 作业。每个指标的归档策略决定指标守护进程执行的速度。metricd 会定期检查传入的存储是否有新的测量结果。要配置每个检查之间的延迟，您可以使用 **[metricd]metric_processing_delay 配置**选项。

2.5.6. 创建归档策略

流程

- 创建归档策略。将 <archive-policy-name> 替换为策略的名称，将 <aggregation-method> 替换为聚合的方法。

```
# openstack metric archive policy create <archive-policy-name> --definition <definition> \
--aggregation-method <aggregation-method>
```

注意

<definition> 是策略定义。使用逗号分隔多个属性（、）。使用冒号(:)分隔归档策略定义的 name 和值。

2.5.7. 管理归档策略

- 删除归档策略：

```
openstack metric archive policy delete <archive-policy-name>
```

- 查看所有归档策略：

```
# openstack metric archive policy list
```

- 查看归档策略的详情：

```
# openstack metric archive-policy show <archive-policy-name>
```

2.5.8. 创建归档策略规则

归档策略规则定义了指标和归档策略之间的映射。这样，用户可以预定义规则，以便基于匹配模式将归档策略分配给指标。

流程

- 创建归档策略规则。将 <rule-name> 替换为规则的名称，<archive-policy-name> 替换为归档策略的名称：

```
# openstack metric archive-policy-rule create <rule-name> /
--archive-policy-name  <archive-policy-name>
```

第 3 章 安装和配置操作测量工具

您必须安装收集的数据收集代理、collectd 和时间序列数据库 Gnocchi。

3.1. 安装 COLLECTD

安装 collectd 时，您可以配置多个 collectd 插件来适合您的环境。

流程

1. 将文件 **/usr/share/openstack-tripleo-heat-templates/environments/collectd-environment.yaml** 复制到本地目录。
2. 打开 **collectd-environment.yaml** 并列出您要在 **CollectdExtraPlugins** 下的插件。您还可以在 **ExtraConfig** 部分提供参数：

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

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

   默认情况下，collectd 附带了 **磁盘**、**接口**、**加载**、**内存**、**进程** 和 **tcpconns** 插件。您可以使用 **CollectdExtraPlugins** 参数添加额外的插件。您还可以使用 **ExtraConfig** 选项为 CollectdExtraPlugins 提供额外的配置信息，如下所示。这个示例添加了 **virt** 插件，并配置连接字符串和主机名格式。

3. 在 **openstack overcloud deploy** 命令中包含修改后的 YAML 文件，以便在所有 overcloud 节点上安装 collectd 守护进程：

```
$ openstack overcloud deploy
--templates \/home/templates/environments/collectd.yaml \
-e /path-to-copied/collectd-environment.yaml
```

其他资源

- 有关 collectd 的更多信息，请参阅 第 1.3.2 节 “collectd”。
- 要查看 collectd 插件和配置，请参阅 *Service Telemetry Framework* 指南中的 collectd 插件。

3.2. 安装 GNOCCHI

默认情况下，在 undercloud 上不启用 Gnocchi。红帽不推荐在 undercloud 上启用 Telemetry，因为它会生成很多 undercloud 无法处理的数据，因为资源和单点故障。

默认情况下，Telemetry 和 Gnocchi 安装在控制器和 Compute 节点上。Gnocchi 的默认存储后端为 file。

您可以通过以下两种方式之一在 overcloud 上部署 Gnocchi：

- 内部.更多信息请参阅 第 3.2.1 节 “在内部部署 Gnocchi”。
- 外部.更多信息请参阅 第 3.2.2 节 “在外部部署 Gnocchi”。

3.2.1. 在内部部署 Gnocchi

默认部署为 internal。

流程

- 要部署 collectd 以向内部 Gnocchi 发送指标数据，请将 **/usr/share/openstack-tripleo-heat-templates/environments/services/collectd.yaml** 添加到 **overcloud deploy** 命令。

其他资源

- 更多信息请参阅 第 3.1 节 “安装 collectd”。

3.2.2. 在外部部署 Gnocchi

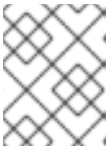
流程

1. 在本地目录中创建自定义 YAML 文件，如 **ExternalGnocchi.yaml**，并确保您包含以下详情：

```
CollectdGnocchiServer: <IPofExternalServer>
CollectdGnocchiUser: admin
CollectdGnocchiAuth: basic
```

2. 若要部署 Gnocchi，请将自定义 YAML 文件添加到 **overcloud deploy** 命令中。使用 **<existing_overcloud_environment_files>** 属于现有部署一部分的环境文件列表替换。

```
openstack overcloud deploy \
-e <existing_overcloud_environment_files> \
-e /usr/share/openstack-tripleo-heat-templates/environments/services/collectd.yaml \
-e /home/templates/environments/ExternalGnocchi.yaml \
...
```



注意

您可以在以下 YAML 文件中找到所有 Gnocchi 参数：**/usr/share/openstack-tripleo-heat-templates/puppet/services/metrics/collectd.yaml**

3.2.3. 验证 Gnocchi 部署

流程

- 列出新资源和指标：

```
$ (overcloud) [stack@undercloud-0 ~]$ openstack metric  metric list |more
+--------------------------------------+--------------------+-------------------------------------------
-+------+----------------------------------+
| id                                   | archive_policy/name | name                                      | unit | resource_id
|
```

```
+--------------------------------------+-------------------+-----------------------------------------------
-+------+--------------------------------------+
| 001715fe-8ad1-4f21-b0fa-1dee2b1c8201 | low   | interface-eth4@if_packets-rx  | None |
e1357192-e563-5524-b0f0-b30fd11ea8df |
| 003057df-0271-4c59-b793-f885fe09668a | low   | pconns-6000-local@tcp_connections-
LISTEN   | None | 33ac72d9-ab4d-54f1-9fcf-56e02f770b70 |
| 0030a5ba-0e4a-4cce-a48b-4122bfbc4e7a | low   | tcpconns-8004-local@tcp_connections-
CLOSING   | None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 0030e0a7-2cea-42db-860c-fa9ab175c8e1 | low   | tcpconns-25-local@tcp_connections-
CLOSING    | None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 007ec848-f16a-48b8-8706-742a28e2efcf | low    | memcached-
local@memcached_command-get         | None | 8900c37d-501a-565d-b38c-85e5a07a0463 |
| 009d08c7-7483-4e0a-b1bd-0b1451a3b2ab | low | tcpconns-8041-local@tcp_connections-
TIME_WAIT | None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 00a8183c-910e-4849-8d08-9401fbe82029 | low   | tcpconns-11211-local@tcp_connections-
CLOSING  | None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 010af93c-62be-442a-94da-5cb426053fe8 | low   | tcpconns-4567-local@tcp_connections-
FIN_WAIT2 | None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 0113b4f3-786d-4a0a-bb4b-59417d63b60f | low   | tcpconns-38857-local@tcp_connections-
LAST_ACK | None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 0128dac7-b237-4e4c-8d5f-83f6e53771b4 | low   | tcpconns-37871-local@tcp_connections-
SYN_SENT | None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 014ca1f7-565d-4ce7-b35f-52fd916a8b06 | low    | tcpconns-43752-local@tcp_connections-
TIME_WAIT   | None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 0154f901-18c2-4dd9-a593-db26d356611a | low   | tcpconns-1993-local@tcp_connections-
CLOSED    | None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 0158d618-d3ba-45e8-bce8-33411d6f35e3 | low    | tcpconns-111-local@tcp_connections-
CLOSED    | None | e1357192-e563-5524-b0f0-b30fd11ea8df |
| 016fe93f-2794-490e-8057-fd5ab75eb4ec | low      | tcpconns-6001-local@tcp_connections-
SYN_RECV  | None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 01ce3b82-98ad-4f61-88d0-ba46b9862688 | low    | interface-br-tenant@if_dropped-rx
| None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 01d4d186-cf26-4264-87a0-9f5deb8872b5 | low     | tcpconns-8774-local@tcp_connections-
ESTABLISHED  | None | 33ac72d9-ab4d-54f1-9fcf-56e02f770b70 |
| 01f19617-3ef6-43e8-9ad8-8c84b923f13f | low      | tcpconns-43394-
local@tcp_connections-FIN_WAIT2   | None | 33ac72d9-ab4d-54f1-9fcf-56e02f770b70 |
| 020646e9-2d50-4126-9a63-a5f36180cc0b | low     | tcpconns-6000-local@tcp_connections-
CLOSE_WAIT   | None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 02155fd3-0d68-4eec-8cd5-8b158f5f03a4 | low       | tcpconns-6633-
local@tcp_connections-LISTEN    | None | 33ac72d9-ab4d-54f1-9fcf-56e02f770b70 |
| 0236355e-3415-4a72-99f2-38ada7b3db68 | low      | tcpconns-43806-
local@tcp_connections-FIN_WAIT2   | None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 024e89d2-aa77-49c0-ae6e-65a665db01b3 | low      | tcpconns-35357-
local@tcp_connections-FIN_WAIT2   | None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 027adb62-272f-4331-8167-28c3489d3b44 | low       | tcpconns-9292-
local@tcp_connections-LISTEN    | None | 33ac72d9-ab4d-54f1-9fcf-56e02f770b70 |
| 0290d15e-7687-4683-bd25-4c030cad12cf | low       | tcpconns-37378-
local@tcp_connections-CLOSE_WAIT  | None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 02a5383f-b061-4422-9570-bfd3b2532832 | low       | processes@ps_state-zombies
| None | 33ac72d9-ab4d-54f1-9fcf-56e02f770b70 |
| 02c959d5-8ae2-4d14-a140-189530b4e8f6 | low       | disk-vda2@disk_merged-write
| None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 02d174b5-6783-4db5-bfe7-8d45931aa0b0 | low       | interface-br-tun@if_octets-rx
| None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 02d8c001-90da-4997-bfa5-e5d3afe599fa | low         | tcpconns-25672-
local@tcp_connections-CLOSING  | None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
```

```
| 02d932f0-5745-4694-86d9-84e365789a7d | low       | tcpconns-9292-
local@tcp_connections-CLOSING   | None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 02e1a5e2-194d-4e49-8b36-a843b5dbdc3d | low       | tcpconns-45228-
local@tcp_connections-CLOSE_WAIT  | None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 02e5dcec-f5b7-41e9-9f3c-714ade502836 | low       | load@load-5min                      |
None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 02fe05ed-e4a5-4a18-ad6c-64f8787225c9 | low       | tcpconns-8774-
local@tcp_connections-SYN_SENT  | None | 33ac72d9-ab4d-54f1-9fcf-56e02f770b70 |
| 03129089-7bbd-47f3-ab14-9cd583d775ae | low       | tcpconns-6379-
local@tcp_connections-LAST_ACK  | None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 032b1771-93c4-4051-bbec-9115c9d329c4 | low       | tcpconns-8042-
local@tcp_connections-TIME_WAIT | None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 033516d5-ac15-4767-b005-cff52276badd | low       | tcpconns-22-local@tcp_connections-
CLOSED      | None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 03589183-efa7-43ed-aea6-9d3c8accf97a | low       | interface-br-tun@if_errors-tx
| None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 0360077f-729d-4591-a9fc-d96fbb7e0326 | low       | tcpconns-6080-
local@tcp_connections-CLOSING   | None | 33ac72d9-ab4d-54f1-9fcf-56e02f770b70 |
| 0365f5a1-4a98-435e-a809-c8706b0671bd | low       | tcpconns-34296-
local@tcp_connections-LAST_ACK | None | 33ac72d9-ab4d-54f1-9fcf-56e02f770b70 |
| 039c57b7-ae5b-4f13-846e-8e5f184cdc4d | low       | tcpconns-111-
local@tcp_connections-CLOSE_WAIT | None | 926d87fe-b388-501d-afa6-dc6af55ce4ad |
| 04169046-80c4-478e-b263-b9529b5ca739 | low       | interface-br-tenant@if_packets-tx
| None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 0432314d-eb8b-4bf9-ab8f-5ecc5b30592d | low       | tcpconns-37415-
local@tcp_connections-CLOSE_WAIT  | None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 043d9d6e-5db0-40e4-83b1-b8120506e9ec | low       | tcpconns-6379-
local@tcp_connections-ESTABLISHED  | None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 04639fe9-6dc1-46eb-8b86-9b39e8fd782d | low       | tcpconns-35357-
local@tcp_connections-SYN_RECV | None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 04871c86-b176-45ed-9f37-bb6fae27e930 | low       | tcpconns-8042-
local@tcp_connections-CLOSE_WAIT   | None | caac8e14-0a2f-5e31-b4bd-e07a14361d81 |
| 049c50c4-bf5e-4098-988a-fb867649ee17 | low       | tcpconns-11211-
local@tcp_connections-SYN_SENT | None | 54b1ecb3-5ccf-533f-84ab-e86eb426606f |
| 04e37efb-fc67-418e-b53b-6b8055967a2a | low       | tcpconns-8004-
local@tcp_connections-CLOSING   | None | 33ac72d9-ab4d-54f1-9fcf-56e02f770b70 |
| 04ef0d0f-db22-4501-ae4a-4bc9ee719075 | low       | tcpconns-9200-
local@tcp_connections-SYN_RECV  | None | 33ac72d9-ab4d-54f1-9fcf-56e02f770b70 |
```

第 4 章 管理操作测量

4.1. 根据您的部署修改环境变量

流程

1. 将 **/usr/share/openstack-tripleo-heat-templates/environments/gnocchi-environment.yaml** 文件复制到您的主目录。
2. 修改参数以符合您的环境。您可以在 YAML 文件中修改以下主要参数：
   - GnocchiIndexerBackend：要使用的数据库索引程序后端，如 **mysql**。请查看 https://github.com/openstack/tripleo-heat-templates/blob/stable/queens/puppet/services/gnocchi-base.yaml#L33
   - GnocchiBackend：临时存储的类型。该值可以是 **rbd**、**swift** 或 **file** (ceph)。更多信息请参阅 https://github.com/openstack/tripleo-heat-templates/blob/stable/queens/environments/storage-environment.yaml#L29-L30
   - NumberOfStorageSacks：存储黑名单的数量。更多信息请参阅 第 4.1.2 节 “sacks 的数量”。
3. 将 **gnocchi-environment.yaml** 添加到 **overcloud deploy** 命令中，以及与您的环境和部署相关的任何其他环境文件。使用 **<existing_overcloud_environment_files>** 属于现有部署一部分的环境文件列表替换：

```
$ openstack overcloud deploy \
<existing_overcloud_environment_files> \
-e ~gnocchi-environment.yaml \
...
```

4.1.1. 运行 metricd worker

默认情况下，**gnocchi-metricd** 守护进程会跨越 CPU 电源，以最大化计算指标聚合时的 CPU 使用。

流程

- 使用 **openstack metric status** 命令查询 HTTP API 并检索用于指标处理的状态：

```
# openstack metric status
```

命令输出显示 **gnocchi-metrid** 守护进程的处理 backlog。只要此积压不持续增加，**gnocchi-metricd** 就可以与要收集的指标数量保持同步。如果持续增加进程的测量结果数量，请增加 **gnocchi-metricd** 守护进程的数量。您可以在任意数量的服务器上运行任意数量的指标守护进程。

4.1.2. sacks 的数量

Gnocchi 中传入的指标数据被推送到不同的 sacks，每个 sack 被分配到一个或多个 **gnocchi-metricd** 守护进程进行处理。攻击的数量取决于系统捕获的活动指标。

红帽建议 sacks 的数量大于活跃 **gnocchi-metricd** worker 总数。

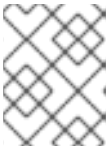
4.1.3. 更改 sack 大小

如果想收集更多指标而不是最初的预计，您可以更改 sack 大小。

推送到 Gnocchi 的测量数据被分成多个 sack 以更好地分发。传入的指标被推送到特定的 sack，每个 sack 都会被分配给一个或多个 **gnocchi-metricd** 守护进程进行处理。要设置 sacks 的数量，请使用系统捕获的活动指标数量。sacks 的数量应该大于活跃 **gnocchi-metricd** worker 的总数。

流程

- 要确定要设置的适当 sacks 值，请使用以下内容：
  sacks value = **活跃** 指标的数量/ 300



注意

如果估算的指标数量是绝对最大值，请将值除以 500 个。如果活跃指标的预计数量是保守和预期增长，请将值除以容纳增长。

4.2. 监控时间序列数据库服务

HTTP API 的 **/v1/status** 端点返回信息，如要处理的测量结果数量(measures backlog)，您可以监控这些信息。以下条件表示一个健康的系统：

- HTTP 服务器和 **gnocchi-metricd** 正在运行
- HTTP 服务器和 **gnocchi-metricd** 不会将错误消息写入到日志文件中。

流程

- 查看时间序列数据库的状态：

```
# openstack metric status
```

输出显示时间序列数据库的状态以及要处理的指标数量。这里的数值越好，最好是接近 0。

4.3. 备份和恢复时间序列数据库

要从意外事件中恢复，请备份索引和存储。您必须使用 PostgreSQL 或 MySQL 创建数据库转储，并使用 Ceph、Swift 或您的文件系统获取数据存储的快照或副本。

流程

1. 恢复您的索引和存储备份。
2. 如有必要，重新安装 Gnocchi。
3. 重新启动 Gnocchi.

4.4. 查看测量结果

您可以查看特定资源的测量结果列表：

流程

1. 使用 **metric measures** 命令：

```
# openstack metric measures show --resource-id UUID <METER_NAME>
```

2. 列出一系列时间戳中特定资源的测量结果：

```
# openstack metric measures show --aggregation mean --start <START_TIME> --stop
<STOP_TIME> --resource-id UUID <METER_NAME>
```

时间戳变量 <START_TIME> 和 <END_TIME> 使用格式 iso-dateThh:mm:ss。

4.5. 管理资源类型

您可以创建、查看和删除资源类型。默认资源类型是通用的，但您可以使用任意数量的属性创建自己的资源类型。

流程

1. 创建新资源类型：

```
$ openstack metric resource-type create testResource01 -a bla:string:True:min_length=123
+-------------+-----------------------------------------------------------------+
| Field       | Value                                                           |
+-------------+-----------------------------------------------------------------+
| attributes/bla | max_length=255, min_length=123, required=True, type=string |
| name        | testResource01                                                  |
| state       | active                                                          |
+-------------+-----------------------------------------------------------------+
```

2. 查看资源类型的配置：

```
$ openstack metric resource-typegnocchi resource-type show testResource01
+-------------+-----------------------------------------------------------------+
| Field       | Value                                                           |
+-------------+-----------------------------------------------------------------+
| attributes/bla | max_length=255, min_length=123, required=True, type=string |
| name        | testResource01                                                  |
| state       | active                                                          |
+-------------+-----------------------------------------------------------------+
```

3. 删除资源类型：

```
$ openstack metric resource-type delete testResource01
```

注意

如果资源正在使用，则无法删除资源类型。

4.6. 查看云使用量措施

流程

- 查看每个项目所有实例的平均内存用量：

```
openstack metrics measures aggregation --resource-type instance --groupby project_id -m
“memoryView L3” --resource-id UUID
```

4.6.1. 启用 L3 缓存监控

如果您的 Intel 硬件和 **libvirt** 版本支持缓存监控技术(CMT)，您可以使用 **cpu_l3_cache** 量表来监控实例使用的 L3 缓存量。

要监控 L3 缓存，必须具有以下参数和文件：

- **LibvirtEnabledPerfEvents** 参数中的 CMT。
- **gnocchi_resources.yaml 文件中的 cpu_l3_cache。**
- **Ceilometer poll .yaml 文件中的 cpu_l3_cache。**

流程

1. 为 **Telemetry** 创建一个 **YAML** 文件，如 **ceilometer-environment.yaml。**
2. **在 ceilometer-environment.yaml 文件中，将 cmt 添加到 LibvirtEnabledPerfEvents 参数。如需更多信息，请参阅 /usr/share/openstack-triple-heat-templates/puppet/services/nova_libvirt.yaml。**
3. **使用此 YAML 文件部署 overcloud。使用 <existing_overcloud_environment_files> 属于现有部署一部分的环境文件列表替换：**

```
#!/bin/bash

openstack overcloud deploy \
--templates \
<existing_overcloud_environment_files> \
-e  /home/stack/ceilometer-environment.yaml \
...
```

4. 验证 **Compute** 节点上的 **Gnocchi** 中是否启用了 **cpu_l3_cache。**

```
$ sudo -i
# docker exec -ti ceilometer_agent_compute cat /etc/ceilometer/gnocchi_resources.yaml |
grep cpu_l3_cache
//Verify that cpu_l3_cache is enabled for Telemetry polling.
# docker exec -ti ceilometer_agent_compute cat /etc/ceilometer/polling.yaml  | grep
```

```
cpu_l3_cache
//If cpu_l3_cache is not enabled for Telemetry, enable it and restart the service.
# docker exec -ti ceilometer_agent_compute echo "        - cpu_l3_cache" >>
/etc/ceilometer/polling.yaml
# docker exec -ti ceilometer_agent_compute pkill -HUP -f "ceilometer.*master process"
```

注意

更改容器镜像中的设置不会在重新引导后保留。

5. **在此 Compute 节点上启动客户端实例后，监控 CMT 指标：**

```
(overcloud) [stack@undercloud-0 ~]$ openstack metric measures show --resource-id
a6491d92-b2c8-4f6d-94ba-edc9dfde23ac cpu_l3_cache
+---------------------------+-------------+-----------+
| timestamp                 | granularity |     value |
+---------------------------+-------------+-----------+
| 2017-10-25T09:40:00+00:00 |       300.0 | 1966080.0 |
| 2017-10-25T09:45:00+00:00 |       300.0 | 1933312.0 |
| 2017-10-25T09:50:00+00:00 |       300.0 | 2129920.0 |
| 2017-10-25T09:55:00+00:00 |       300.0 | 1966080.0 |
| 2017-10-25T10:00:00+00:00 |       300.0 | 1933312.0 |
| 2017-10-25T10:05:00+00:00 |       300.0 | 2195456.0 |
| 2017-10-25T10:10:00+00:00 |       300.0 | 1933312.0 |
| 2017-10-25T10:15:00+00:00 |       300.0 | 1998848.0 |
| 2017-10-25T10:20:00+00:00 |       300.0 | 2097152.0 |
| 2017-10-25T10:25:00+00:00 |       300.0 | 1933312.0 |
| 2017-10-25T10:30:00+00:00 |       300.0 | 1966080.0 |
| 2017-10-25T10:35:00+00:00 |       300.0 | 1933312.0 |
| 2017-10-25T10:40:00+00:00 |       300.0 | 1933312.0 |
| 2017-10-25T10:45:00+00:00 |       300.0 | 1933312.0 |
| 2017-10-25T10:50:00+00:00 |       300.0 | 2850816.0 |
| 2017-10-25T10:55:00+00:00 |       300.0 | 2359296.0 |
| 2017-10-25T11:00:00+00:00 |       300.0 | 2293760.0 |
+---------------------------+-------------+-----------+
```

4.7. 升级 GNOCCHI

默认情况下，使用 Red Hat OpenStack Platform director 升级 Gnocchi 升级您的部署。有关升级部署的详情，请参阅升级 Red Hat OpenStack Platform。如果您使用 Red Hat OpenStack Platform 10 并希望升级到 Red Hat OpenStack Platform 13，请参阅 快进升级。

第 5 章 管理警报

您可以使用名为 aodh 的警报服务根据定义的规则根据 Ceilometer 或 Gnocchi 收集的指标数据来触发操作。

5.1. 查看现有警报

流程

1. 列出现有的 Telemetry 警报：

```
# openstack alarm list
+--------------------------------------+-----------------------------------------+-----------------------
--+------------------+----------+---------+
| alarm_id                             | type                                    | name                  | state
| severity | enabled |
+--------------------------------------+-----------------------------------------+-----------------------
--+------------------+----------+---------+
| 922f899c-27c8-4c7d-a2cf-107be51ca90a |
gnocchi_aggregation_by_resources_threshold | iops-monitor-read-requests |
insufficient data | low      | True    |
+--------------------------------------+-----------------------------------------+-----------------------
--+------------------+----------+---------+
```

2. 要列出分配给资源的计量，请指定资源的 UUID。例如：

```
# openstack resource show 5e3fcbe2-7aab-475d-b42c-a440aa42e5ad
```

5.2. 创建警报

您可以使用 aodh 创建在达到阈值时激活的警报。在本例中，警报激活并在单个实例的平均 CPU 利用率超过 80% 时添加一个日志条目。

流程

1. 创建警报并使用查询来隔离实例的特定 ID (94619081-abf5-4f1f-81c7-9cedaa872403)以进行监控：

```
# openstack alarm create --type gnocchi_aggregation_by_resources_threshold --name
cpu_usage_high --metric cpu_util --threshold 80 --aggregation-method sum --
resource-type instance --query '{"=": {"id": "94619081-abf5-4f1f-81c7-9cedaa872403"}}'
--alarm-action 'log://'
```

```
+---------------------------+-------------------------------------------------------+
| Field                     | Value                                                 |
+---------------------------+-------------------------------------------------------+
| aggregation_method        | sum                                                   |
| alarm_actions             | [u'log://']                                           |
| alarm_id                  | b794adc7-ed4f-4edb-ace4-88cbe4674a94                  |
| comparison_operator       | eq                                                    |
| description               | gnocchi_aggregation_by_resources_threshold alarm rule |
| enabled                   | True                                                  |
| evaluation_periods        | 1                                                     |
| granularity               | 60                                                    |
| insufficient_data_actions | []                                                    |
| metric                    | cpu_util                                              |
| name                      | cpu_usage_high                                        |
| ok_actions                | []                                                    |
| project_id                | 13c52c41e0e543d9841a3e761f981c20                      |
| query                     | {"=": {"id": "94619081-abf5-4f1f-81c7-9cedaa872403"}} |
| repeat_actions            | False                                                 |
| resource_type             | instance                                              |
| severity                  | low                                                   |
| state                     | insufficient data                                     |
| state_timestamp           | 2016-12-09T05:18:53.326000                            |
| threshold                 | 80.0                                                  |
| time_constraints          | []                                                    |
| timestamp                 | 2016-12-09T05:18:53.326000                            |
| type                      | gnocchi_aggregation_by_resources_threshold            |
| user_id                   | 32d3f2c9a234423cb52fb69d3741dbbc                      |
+---------------------------+-------------------------------------------------------+
```

2. 若要编辑现有阈值警报，可使用 aodh alarm update 命令。例如，要将警报阈值增加到 75%，请使用以下命令：

```
# openstack alarm update --name cpu_usage_high --threshold 75
```

5.3. 禁用警报

流程

- 要禁用警报，请输入以下命令：

```
# openstack alarm update --name cpu_usage_high --enabled=false
```

5.4. 删除警报

流程

- 要删除警报，请输入以下命令：

```
# openstack alarm delete --name cpu_usage_high
```

5.5. 示例：监控实例的磁盘活动

以下示例演示了如何使用 aodh 警报来监控特定项目中包含的所有实例的累积磁盘活动。

流程

1. 检查现有项目，再选择您要监控的项目的适当 UUID。本例使用 admin 租户：

```
$ openstack project list
+----------------------------------+----------+
| ID                               | Name     |
+----------------------------------+----------+
| 745d33000ac74d30a77539f8920555e7 | admin    |
| 983739bb834a42ddb48124a38def8538 | services |
| be9e767afd4c4b7ead1417c6dfedde2b | demo     |
+----------------------------------+----------+
```

2. 使用项目 UUID 创建警报，以分析 admin 租户中实例生成的所有读取请求的 sum （）。您可以使用 --query 参数进一步限制查询：

```
# openstack alarm create --type gnocchi_aggregation_by_resources_threshold --name
iops-monitor-read-requests --metric disk.read.requests.rate --threshold 42000 --
aggregation-method sum --resource-type instance --query '{"=": {"project_id":
"745d33000ac74d30a77539f8920555e7"}}'
+---------------------------+------------------------------------------------------------+
| Field                     | Value                                                      |
+---------------------------+------------------------------------------------------------+
| aggregation_method        | sum                                                        |
| alarm_actions             | []                                                         |
| alarm_id                  | 192aba27-d823-4ede-a404-7f6b3cc12469                       |
| comparison_operator       | eq                                                         |
| description               | gnocchi_aggregation_by_resources_threshold alarm rule      |
| enabled                   | True                                                       |
| evaluation_periods        | 1                                                          |
| granularity               | 60                                                         |
| insufficient_data_actions | []                                                         |
| metric                    | disk.read.requests.rate                                    |
| name                      | iops-monitor-read-requests                                 |
| ok_actions                | []                                                         |
| project_id                | 745d33000ac74d30a77539f8920555e7                           |
| query                     | {"=": {"project_id": "745d33000ac74d30a77539f8920555e7"}}  |
| repeat_actions            | False                                                      |
| resource_type             | instance                                                   |
| severity                  | low                                                        |
| state                     | insufficient data                                          |
| state_timestamp           | 2016-11-08T23:41:22.919000                                 |
```

```
| threshold              | 42000.0                                                  |
| time_constraints       | []                                                       |
| timestamp              | 2016-11-08T23:41:22.919000                               |
| type                   | gnocchi_aggregation_by_resources_threshold               |
| user_id                | 8c4aea738d774967b4ef388eb41fef5e                         |
+------------------------+----------------------------------------------------------+
```

5.6. 示例：监控 CPU 使用

要监控实例的性能，可检查 Gnocchi 数据库以确定您可以监控哪些指标，如内存或 CPU 使用量。

流程

1. 输入 **openstack metric resource show** 命令和实例 UUID，以识别您可以监控的指标：

```
$ openstack metric resource show --type instance d71cdf9a-51dc-4bba-8170-
9cd95edd3f66
+----------------------+--------------------------------------------------------------------------+
| Field                | Value                                                                    |
+----------------------+--------------------------------------------------------------------------+
| created_by_project_id | 44adccdc32614688ae765ed4e484f389                                        |
| created_by_user_id    | c24fa60e46d14f8d847fca90531b43db                                        |
| creator               |
c24fa60e46d14f8d847fca90531b43db:44adccdc32614688ae765ed4e484f389   |
| display_name          | test-instance                                                           |
| ended_at              | None                                                                    |
| flavor_id             | 14c7c918-df24-481c-b498-0d3ec57d2e51                                    |
| flavor_name           | m1.tiny                                                                 |
| host                  | overcloud-compute-0                                                     |
| id                    | d71cdf9a-51dc-4bba-8170-9cd95edd3f66                                    |
| image_ref             | e75dff7b-3408-45c2-9a02-61fbfbf054d7                                    |
| metrics               | compute.instance.booting.time: c739a70d-2d1e-45c1-8c1b-
4d28ff2403ac |
|                       | cpu.delta: 700ceb7c-4cff-4d92-be2f-6526321548d6                         |
|                       | cpu: 716d6128-1ea6-430d-aa9c-ceaff2a6bf32                               |
|                       | cpu_l3_cache: 3410955e-c724-48a5-ab77-c3050b8cbe6e                      |
|                       | cpu_util: b148c392-37d6-4c8f-8609-e15fc15a4728                          |
|                       | disk.allocation: 9dd464a3-acf8-40fe-bd7e-3cb5fb12d7cc                   |
|                       | disk.capacity: c183d0da-e5eb-4223-a42e-855675dd1ec6                     |
|                       | disk.ephemeral.size: 15d1d828-fbb4-4448-b0f2-2392dcfed5b6               |
|                       | disk.iops: b8009e70-daee-403f-94ed-73853359a087                         |
|                       | disk.latency: 1c648176-18a6-4198-ac7f-33ee628b82a9                      |
|                       | disk.read.bytes.rate: eb35828f-312f-41ce-b0bc-cb6505e14ab7              |
|                       | disk.read.bytes: de463be7-769b-433d-9f22-f3265e146ec8                   |
|                       | disk.read.requests.rate: 588ca440-bd73-4fa9-a00c-8af67262f4fd           |
|                       | disk.read.requests: 53e5d599-6cad-47de-b814-5cb23e8aaf24                |
|                       | disk.root.size: cee9d8b1-181e-4974-9427-aa7adb3b96d9                    |
|                       | disk.usage: 4d724c99-7947-4c6d-9816-abbbc166f6f3                        |
|                       | disk.write.bytes.rate: 45b8da6e-0c89-4a6c-9cce-c95d49d9cc8b             |
|                       | disk.write.bytes: c7734f1b-b43a-48ee-8fe4-8a31b641b565                  |
|                       | disk.write.requests.rate: 96ba2f22-8dd6-4b89-b313-1e0882c4d0d6          |
```

```
|                     | disk.write.requests: 553b7254-be2d-481b-9d31-b04c93dbb168          |
|                     | memory.bandwidth.local: 187f29d4-7c70-4ae2-86d1-191d11490aad       |
|                     | memory.bandwidth.total: eb09a4fc-c202-4bc3-8c94-aa2076df7e39       |
|                     | memory.resident: 97cfb849-2316-45a6-9545-21b1d48b0052              |
|                     | memory.swap.in: f0378d8f-6927-4b76-8d34-a5931799a301               |
|                     | memory.swap.out: c5fba193-1a1b-44c8-82e3-9fdc9ef21f69              |
|                     | memory.usage: 7958d06d-7894-4ca1-8c7e-72ba572c1260                 |
|                     | memory: a35c7eab-f714-4582-aa6f-48c92d4b79cd                       |
|                     | perf.cache.misses: da69636d-d210-4b7b-bea5-18d4959e95c1            |
|                     | perf.cache.references: e1955a37-d7e4-4b12-8a2a-51de4ec59efd        |
|                     | perf.cpu.cycles: 5d325d44-b297-407a-b7db-cc9105549193              |
|                     | perf.instructions: 973d6c6b-bbeb-4a13-96c2-390a63596bfc            |
|                     | vcpus: 646b53d0-0168-4851-b297-05d96cc03ab2                        |
| original_resource_id  | d71cdf9a-51dc-4bba-8170-9cd95edd3f66                               |
| project_id            | 3cee262b907b4040b26b678d7180566b                                   |
| revision_end          | None                                                               |
| revision_start        | 2017-11-16T04:00:27.081865+00:00                                   |
| server_group          | None                                                               |
| started_at            | 2017-11-16T01:09:20.668344+00:00                                   |
| type                  | instance                                                           |
| user_id               | 1dbf5787b2ee46cf9fa6a1dfea9c9996                                   |
+-----------------------+--------------------------------------------------------------------+
```

因此，指标值列出了您可以使用 aodh 警报监控的组件，如 cpu_util。

2. 要监控 CPU 用量，请使用 cpu_util 指标：

```
$ openstack metric show --resource-id d71cdf9a-51dc-4bba-8170-9cd95edd3f66
cpu_util
+-----------------------------------+--------------------------------------------------------------------+
| Field                             | Value                                                              |
+-----------------------------------+--------------------------------------------------------------------+
| archive_policy/aggregation_methods | std, count, min, max, sum, mean
|
| archive_policy/back_window        | 0                                                                  |
| archive_policy/definition         | - points: 8640, granularity: 0:05:00, timespan: 30
days, 0:00:00  |
| archive_policy/name               | low                                                                |
| created_by_project_id             | 44adccdc32614688ae765ed4e484f389
|
| created_by_user_id                | c24fa60e46d14f8d847fca90531b43db
|
| creator                           |
c24fa60e46d14f8d847fca90531b43db:44adccdc32614688ae765ed4e484f389 |
| id                                | b148c392-37d6-4c8f-8609-e15fc15a4728                               |
| name                              | cpu_util                                                           |
| resource/created_by_project_id    | 44adccdc32614688ae765ed4e484f389
|
| resource/created_by_user_id       | c24fa60e46d14f8d847fca90531b43db
|
| resource/creator                  |
c24fa60e46d14f8d847fca90531b43db:44adccdc32614688ae765ed4e484f389 |
```

```
| resource/ended_at                | None                                                                 |
| resource/id                      | d71cdf9a-51dc-4bba-8170-9cd95edd3f66                                 |
| resource/original_resource_id    | d71cdf9a-51dc-4bba-8170-9cd95edd3f66
|
| resource/project_id              | 3cee262b907b4040b26b678d7180566b
|
| resource/revision_end            | None                                                                 |
| resource/revision_start          | 2017-11-17T00:05:27.516421+00:00
|
| resource/started_at              | 2017-11-16T01:09:20.668344+00:00                                     |
| resource/type                    | instance                                                             |
| resource/user_id                 | 1dbf5787b2ee46cf9fa6a1dfea9c9996                                     |
| unit                             | None                                                                 |
+----------------------------------+----------------------------------------------------------------------+
```

- archive_policy：定义计算 std、count、min、max、sum 和 mean 值的聚合间隔。

3. 使用 aodh 创建查询 cpu_util 的监控任务。此任务会根据您指定的设置触发事件。例如，当实例的 CPU 高峰持续时间超过 80% 时，要引发日志条目，请使用以下命令：

```
$ openstack alarm create \
  --project-id 3cee262b907b4040b26b678d7180566b \
  --name high-cpu \
  --type gnocchi_resources_threshold \
  --description 'High CPU usage' \
  --metric cpu_util \
  --threshold 80.0 \
  --comparison-operator ge \
  --aggregation-method mean \
  --granularity 300 \
  --evaluation-periods 1 \
  --alarm-action 'log://' \
  --ok-action 'log://' \
  --resource-type instance \
  --resource-id d71cdf9a-51dc-4bba-8170-9cd95edd3f66
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| aggregation_method        | mean                                 |
| alarm_actions             | [u'log://']                          |
| alarm_id                  | 1625015c-49b8-4e3f-9427-3c312a8615dd |
| comparison_operator       | ge                                   |
| description               | High CPU usage                       |
| enabled                   | True                                 |
| evaluation_periods        | 1                                    |
| granularity               | 300                                  |
| insufficient_data_actions | []                                   |
| metric                    | cpu_util                             |
| name                      | high-cpu                             |
| ok_actions                | [u'log://']                          |
| project_id                | 3cee262b907b4040b26b678d7180566b     |
| repeat_actions            | False                                |
```

```
| resource_id             | d71cdf9a-51dc-4bba-8170-9cd95edd3f66 |
| resource_type           | instance                             |
| severity                | low                                  |
| state                   | insufficient data                    |
| state_reason            | Not evaluated yet                    |
| state_timestamp         | 2017-11-16T05:20:48.891365           |
| threshold               | 80.0                                 |
| time_constraints        | []                                   |
| timestamp               | 2017-11-16T05:20:48.891365           |
| type                    | gnocchi_resources_threshold          |
| user_id                 | 1dbf5787b2ee46cf9fa6a1dfea9c9996     |
+-------------------------+--------------------------------------+
```

- comparison-operator：如果 CPU 使用率大于或等于 80%，ge 运算符定义了警报触发。
- granularity：指标关联有一个归档策略，策略可以具有各种粒度。例如，每月 1 小时的 5 分钟聚合为 1 小时聚合。granularity 值必须与归档策略中描述的持续时间匹配。
- 评估-periods：在警报触发前需要传递的粒度周期数。例如，如果您将此值设置为 2，则在警报触发前，CPU 用量需要超过 80% 才能触发两个轮询周期。
- [U'log://'] ：当您将 alarm_actions 或 ok_actions 设置为 [u'log://'] 时，事件会被触发或返回到普通的状态，则会记录到 aodh 日志文件。

注意

您可以定义在警报触发时运行的不同操作(alarm_actions)，并在返回正常状态(ok_actions)（如 Webhook URL）时运行。

5.7. 查看警报历史记录

若要检查是否触发了警报，您可以查询警报历史记录。

流程

- 使用 openstack alarm-history show 命令：

```
openstack alarm-history show 1625015c-49b8-4e3f-9427-3c312a8615dd --fit-width
+--------------------------+-----------------+-------------------------------------------------------------------
-------------------------------------------------------------------------------+----------------------
```

```
------------+
| timestamp                  | type            | detail
| event_id                         |
+----------------------------+-----------------+-------------------------------------------------------------------------------
-------------------------------------------------------------------------------+------------------------
------------+
| 2017-11-16T05:21:47.850094 | state transition | {"transition_reason": "Transition to ok
due to 1 samples inside threshold, most recent: 0.0366665763", "state": "ok"}
| 3b51f09d-ded1-4807-b6bb-65fdc87669e4 |
+----------------------------+-----------------+-------------------------------------------------------------------------------
-------------------------------------------------------------------------------+------------------------
------------+
```

第 6 章 日志

Red Hat OpenStack Platform (RHOSP)将信息消息写入特定的日志文件中；您可以使用这些消息来故障排除和监控系统事件。

注意

您不需要手动将单个日志文件附加到您的支持问题单中。sosreport 工具会自动收集所需日志。

6.1. OPENSTACK 服务的日志文件位置

每个 OpenStack 组件都有一个单独的日志记录目录，其中包含特定于正在运行的服务的文件。

6.1.1. 裸机置备(ironic)日志文件

Service	服务名称	日志路径
OpenStack Ironic API	openstack-ironic-api.service	/var/log/containers/ironic/ironic-api.log
OpenStack Ironic Conductor	openstack-ironic-conductor.service	/var/log/containers/ironic/ironic-conductor.log

6.1.2. Block Storage (cinder)日志文件

Service	服务名称	日志路径
Block Storage API	openstack-cinder-api.service	/var/log/containers/cinder-api.log
块存储备份	openstack-cinder-backup.service	/var/log/containers/cinder/backup.log
信息性信息	cinder-manage 命令	/var/log/containers/cinder/cinder-manage.log
块存储调度程序	openstack-cinder-scheduler.service	/var/log/containers/cinder/scheduler.log
块存储卷	openstack-cinder-volume.service	/var/log/containers/cinder/volume.log

6.1.3. compute (nova)日志文件

Service	服务名称	日志路径
OpenStack Compute API 服务	openstack-nova-api.service	/var/log/containers/nova/nova-api.log
OpenStack 计算证书服务器	openstack-nova-cert.service	/var/log/containers/nova/nova-cert.log
OpenStack 计算服务	openstack-nova-compute.service	/var/log/containers/nova/nova-compute.log
OpenStack Compute Conductor 服务	openstack-nova-conductor.service	/var/log/containers/nova/nova-conductor.log
OpenStack Compute VNC 控制台身份验证服务器	openstack-nova-consoleauth.service	/var/log/containers/nova/nova-consoleauth.log
信息性信息	nova-manage 命令	/var/log/containers/nova/nova-manage.log
OpenStack Compute NoVNC 代理服务	openstack-nova-novncproxy.service	/var/log/containers/nova/nova-novncproxy.log
OpenStack 计算调度程序服务	openstack-nova-scheduler.service	/var/log/containers/nova/nova-scheduler.log

6.1.4. 仪表板(horizon)日志文件

Service	服务名称	日志路径
特定用户交互的日志	仪表板接口	/var/log/containers/horizon/horizon.log

Apache HTTP 服务器将多个额外的日志文件用于 Dashboard Web 界面，您可以使用网页浏览器或命令行客户端（如 keystone 和 nova）进行访问。以下日志文件有助于跟踪控制面板的使用并诊断错误：

用途	日志路径
所有已处理的 HTTP 请求	/var/log/containers/httpd/horizon_access.log
HTTP 错误	/var/log/containers/httpd/horizon_error.log

用途	日志路径
admin-role API 请求	/var/log/containers/httpd/keystone_wsgi_admin_access.log
admin-role API 错误	/var/log/containers/httpd/keystone_wsgi_admin_error.log
member-role API 请求	/var/log/containers/httpd/keystone_wsgi_main_access.log
member-role API 错误	/var/log/containers/httpd/keystone_wsgi_main_error.log

注意

另外，还有 /var/log/containers/httpd/default_error.log，它存储同一主机上运行的其他 Web 服务报告的错误。

6.1.5. 数据处理(sahara)日志文件

Service	服务名称	日志路径
Sahara API Server	openstack-sahara-all.service openstack-sahara-api.service	/var/log/containers/sahara/sahara-all.log /var/log/containers/messages
Sahara Engine Server	openstack-sahara-engine.service	/var/log/containers/messages

6.1.6. 数据库即服务(trove)日志文件

Service	服务名称	日志路径
OpenStack Trove API 服务	openstack-trove-api.service	/var/log/containers/trove/trove-api.log
OpenStack Trove Conductor 服务	openstack-trove-conductor.service	/var/log/containers/trove/trove-conductor.log
OpenStack Trove 客户机代理服务	openstack-trove-guestagent.service	/var/log/containers/trove/logfile.txt

Service	服务名称	日志路径
OpenStack Trove taskmanager Service	openstack-trove-taskmanager.service	/var/log/containers/trove/trove-taskmanager.log

6.1.7. Identity Service (keystone)日志文件

Service	服务名称	日志路径
OpenStack 身份服务	openstack-keystone.service	/var/log/containers/keystone/keystone.log

6.1.8. Image Service (glance)日志文件

Service	服务名称	日志路径
OpenStack Image Service API 服务器	openstack-glance-api.service	/var/log/containers/glance/api.log
OpenStack Image Service Registry 服务器	openstack-glance-registry.service	/var/log/containers/glance/registry.log

6.1.9. networking (neutron)日志文件

Service	服务名称	日志路径
OpenStack Neutron DHCP Agent	neutron-dhcp-agent.service	/var/log/containers/neutron/dhcp-agent.log
OpenStack 网络层 3 代理	neutron-l3-agent.service	/var/log/containers/neutron/l3-agent.log
元数据代理服务	neutron-metadata-agent.service	/var/log/containers/neutron/metadata-agent.log
元数据命名空间代理	不适用	/var/log/containers/neutron/neutron-ns-metadata-proxy-*UUID*.log
Open vSwitch 代理	neutron-openvswitch-agent.service	/var/log/containers/neutron/openvswitch-agent.log
OpenStack 网络服务	neutron-server.service	/var/log/containers/neutron/server.log

6.1.10. Object Storage (swift)日志文件

OpenStack Object Storage 仅将日志发送到系统日志功能。

注意

默认情况下，所有 Object Storage 日志文件都会使用 local0、local1 和 local2 syslog 工具进入 /var/log/containers/swift.log。

对象存储的日志消息分为两大类：由 REST API 服务以及后台守护进程使用它们。API 服务消息包含每个 API 请求的一个行，其方式与常见的 HTTP 服务器类似；前端(Proxy)和后端(Account、Container、Object)服务也会发布此类消息。守护进程消息的结构较少，通常包含有关执行定期任务的守护进程的人类可读信息。但是，无论对象存储的哪个部分生成消息，源身份始终位于行首。

以下是代理消息的示例：

```
Apr 20 15:20:34 rhev-a24c-01 proxy-server: 127.0.0.1 127.0.0.1 20/Apr/2015/19/20/34 GET
/v1/AUTH_zaitcev%3Fformat%3Djson%26marker%3Dtestcont HTTP/1.0 200 - python-swiftclient-
2.1.0 AUTH_tk737d6... - 2 - txc454fa8ea4844d909820a-0055355182 - 0.0162 - -
1429557634.806570053 1429557634.822791100
```

以下是来自后台守护进程的临时信息示例：

```
Apr 27 17:08:15 rhev-a24c-02 object-auditor: Object audit (ZBF). Since Mon Apr 27 21:08:15 2015:
Locally: 1 passed, 0 quarantined, 0 errors files/sec: 4.34 , bytes/sec: 0.00, Total time: 0.23, Auditing
time: 0.00, Rate: 0.00
Apr 27 17:08:16 rhev-a24c-02 object-auditor: Object audit (ZBF) "forever" mode completed: 0.56s.
Total quarantined: 0, Total errors: 0, Total files/sec: 14.31, Total bytes/sec: 0.00, Auditing time: 0.02,
Rate: 0.04
Apr 27 17:08:16 rhev-a24c-02 account-replicator: Beginning replication run
Apr 27 17:08:16 rhev-a24c-02 account-replicator: Replication run OVER
Apr 27 17:08:16 rhev-a24c-02 account-replicator: Attempted to replicate 5 dbs in 0.12589 seconds
(39.71876/s)
Apr 27 17:08:16 rhev-a24c-02 account-replicator: Removed 0 dbs
Apr 27 17:08:16 rhev-a24c-02 account-replicator: 10 successes, 0 failures
```

6.1.11. 编配(heat)日志文件

Service	服务名称	日志路径
OpenStack Heat API 服务	openstack-heat-api.service	/var/log/containers/heat/heat-api.log

Service	服务名称	日志路径
OpenStack Heat Engine Service	openstack-heat-engine.service	/var/log/containers/heat/heat-engine.log
编排服务事件	不适用	/var/log/containers/heat/heat-manage.log

6.1.12. 共享文件系统服务(manila)日志文件

Service	服务名称	日志路径
OpenStack Manila API Server	openstack-manila-api.service	/var/log/containers/manila/api.log
OpenStack Manila 调度程序	openstack-manila-scheduler.service	/var/log/containers/manila/scheduler.log
OpenStack Manila Share Service	openstack-manila-share.service	/var/log/containers/manila/share.log

注意

Manila Python 库的一些信息也可以记录在 /var/log/containers/manila/manila-manage.log 中。

6.1.13. 遥测(ceilometer)日志文件

Service	服务名称	日志路径
OpenStack ceilometer 通知代理	openstack-ceilometer-notification.service	/var/log/containers/ceilometer/agent-notification.log
OpenStack ceilometer 警报评估	openstack-ceilometer-alarm-evaluator.service	/var/log/containers/ceilometer/alarm-evaluator.log
OpenStack ceilometer 警报通知	openstack-ceilometer-alarm-notifier.service	/var/log/containers/ceilometer/alarm-notifier.log
OpenStack ceilometer API	httpd.service	/var/log/containers/ceilometer/api.log
信息性信息	MongoDB 集成	/var/log/containers/ceilometer/ceilometer-dbsync.log

Service	服务名称	日志路径
OpenStack ceilometer 中央代理	openstack-ceilometer-central.service	/var/log/containers/ceilometer/central.log
OpenStack ceilometer 集合	openstack-ceilometer-collector.service	/var/log/containers/ceilometer/collector.log
OpenStack ceilometer 计算代理	openstack-ceilometer-compute.service	/var/log/containers/ceilometer/compute.log

6.1.14. 支持服务的日志文件

下列服务由核心 OpenStack 组件使用，并拥有自己的日志目录和文件。

Service	服务名称	日志路径
消息代理(RabbitMQ)	rabbitmq-server.service	/var/log/rabbitmq/rabbit@*short_hostname*.log /var/log/rabbitmq/rabbit@*short_hostname*-sasl.log （用于简单验证和安全层相关的日志消息）
数据库服务器(MariaDB)	mariadb.service	/var/log/mariadb/mariadb.log
面向文档的数据库(MongoDB)	mongod.service	/var/log/mongodb/mongodb.log
虚拟网络交换机(Open vSwitch)	openvswitch-nonetwork.service	/var/log/openvswitch/ovsdb-server.log /var/log/openvswitch/ovs-vswitchd.log

6.2. 集中式日志系统架构和组件

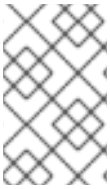
监控工具使用客户端-服务器模型以及部署到 Red Hat OpenStack Platform (RHOSP) overcloud 节点上的客户端。Fluentd 服务提供客户端集中式日志记录(CL)。所有 RHOSP 服务都生成和更新日志文件。这些日志文件记录操作、错误、警告和其他事件。在 OpenStack 等分布式环境中，将这些日志收集中央位置简化了调试和管理。集中化日志记录允许您有一个中央位置查看整个 OpenStack 环境的日志。这些日志来自操作系统，如 syslog 和 audit 日志文件、基础架构组件（如 RabbitMQ 和 MariaDB）以及 OpenStack 服务，如 Identity、Compute 等等。集中式日志记录工具链包括以下组件：

- log Collection Agent (Fluentd)

- log Relay/Transformer (Fluentd)
- 数据存储(ElasticSearch)
- API/Presentation Layer (Kibana)

注意

Red Hat OpenStack Platform director 不会为集中式日志记录部署服务器端组件。红帽不支持服务器端组件，包括 ElasticSearch 数据库、Kibana 和 Fluentd，带有作为日志聚合器运行的插件。下图中描述了集中式日志记录组件及其交互。



注意

蓝色中显示的项目表示红帽支持的组件。

图 6.1. Red Hat OpenStack Platform 单一 HA 部署

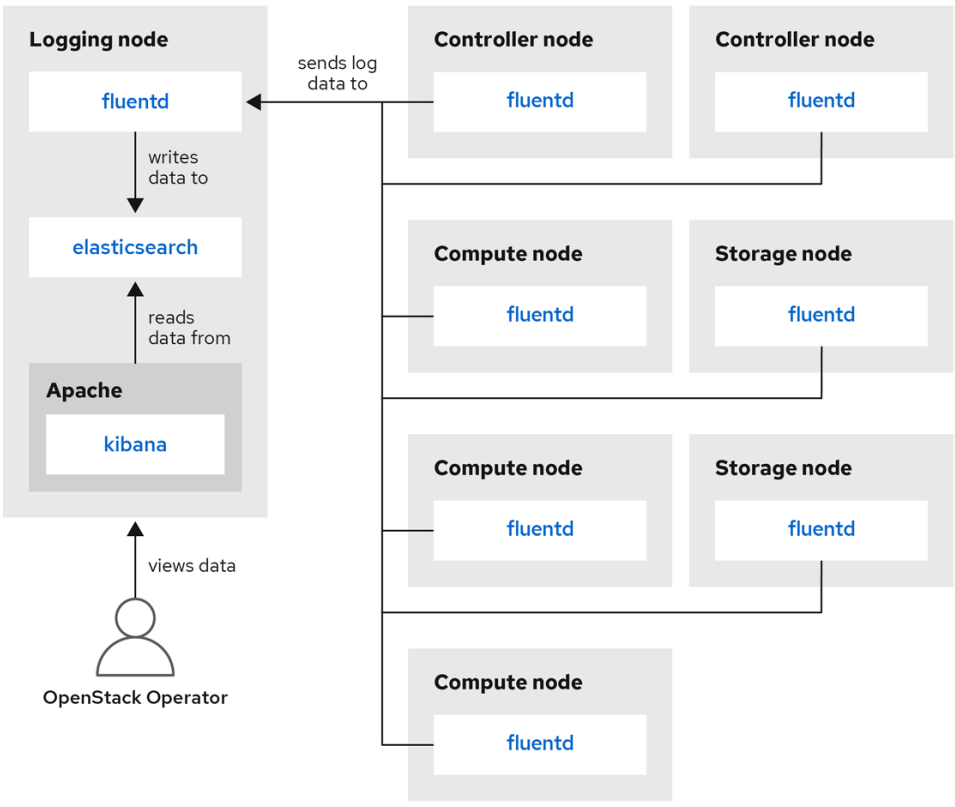
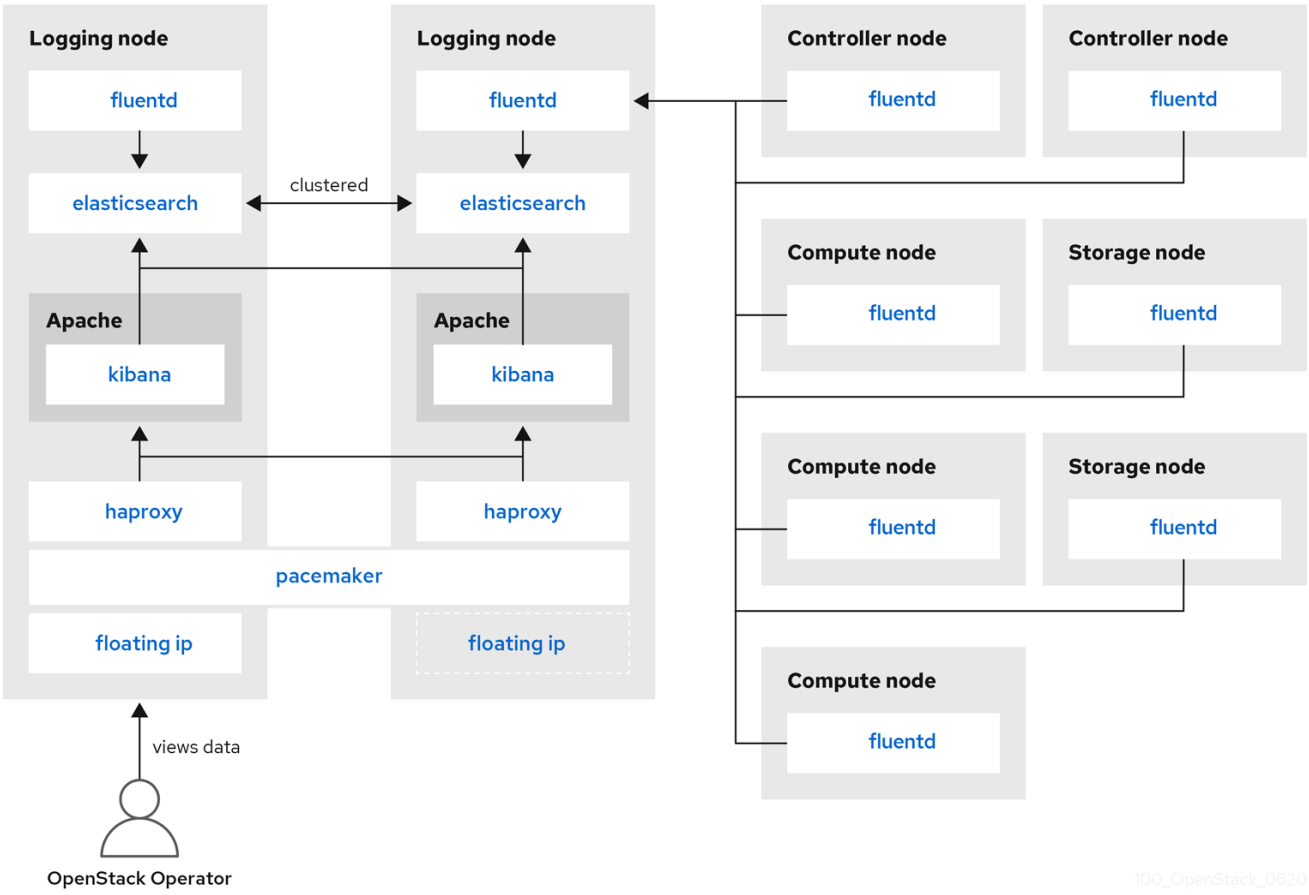


图 6.2. Red Hat OpenStack Platform 的 HA 部署



6.3. 日志服务安装概述

日志收集代理 Fluentd 收集客户端的日志，并将这些日志发送到在服务器端运行的 Fluentd 实例。此 Fluentd 实例将日志记录重定向到 Elasticsearch 以进行存储。

6.4. 在所有机器上部署 FLUENTD

Fluentd 是一个日志收集代理，是集中式日志记录工具链的一部分。要在所有机器上部署 Fluentd，您必须修改 logging-environment.yaml 文件中的 LoggingServers 参数：

前提条件

- 确保在服务器端安装 Elasticsearch 和 Fluentd 转发。如需更多信息，请参阅 opstools-ansible 项目中 与客户端集成兼容的示例部署。

流程

1. 将 tripleo-heat-templates/environments/logging-environment.yaml 文件复制到您的主目

录。

2. **在复制的文件中，在 LoggingServers 参数中创建条目以适应您的环境。以下片段是 LoggingServers 参数配置示例：**

```
parameter_defaults:

Simple configuration

LoggingServers:
 - host: log0.example.com
   port: 24224
 - host: log1.example.com
   port: 24224
```

3. **在 openstack overcloud deploy 命令中包括修改后的环境文件，以及与您环境和部署相关的任何其他环境文件。使用 <existing_overcloud_environment_files> 属于现有部署一部分的环境文件列表替换：**

```
$ openstack overcloud deploy \
<existing_overcloud_environment_files> \
-e /home/templates/environments/logging-environment.yaml \
...
```

S .Additional 资源

- **更多信息请参阅 第 6.5 节 “可配置的日志记录参数”。**

6.5. 可配置的日志记录参数

这个表格包含您可以配置的日志记录参数的描述。您可以在 tripleo-heat-templates/puppet/services/logging/fluentd-config.yaml 文件中找到这些参数。

参数	Description
LoggingDefaultFormat	用于解析日志文件中消息的默认格式。
LoggingPosFilePath	放置 Fluentd **pos_file** 文件的目录。它用于跟踪 **尾部** 输入类型的文件位置。
LoggingDefaultGroups	将 Fluentd 用户添加到这些组。如果要修改组的默认列表，请覆盖此参数。使用 **LoggingExtraGroups** 参数，将 Fluentd 用户添加到其他组中。

参数	Description
LoggingExtraGroups	除了 **LoggingDefaultGroups** 参数外，还要将 Fluentd 用户添加到这些组，以及单个可组合服务提供的组。
LoggingDefaultFilters	Fluentd 默认过滤器列表。此列表传递给 **fluentd::config** 资源的 **过滤器键**。如果您不希望默认过滤器集，则覆盖此项。如果要添加额外的服务器，请使用 **LoggingExtraFilters** 参数。
LoggingExtraFilters	附加 Fluentd 过滤器列表。此列表传递给 **fluentd::config** 资源的 **过滤器键**。
LoggingUsesSSL	布尔值表示是否使用 **secure_forward** 插件转发日志消息。
LoggingSSLKey	用于 Fluentd CA 证书的 PEM 编码密钥。**in_secure_forward** 参数使用 **LoggingSSLKey** 参数的值。
LoggingSSLCertificate	用于 Fluentd 的 PEM 编码 SSL CA 证书。
LoggingSSLKeyPassphrase	**LoggingSSLKey** 参数的密语。**in_secure_forward** 参数使用 **LoggingSSLKeyPassphrase** 参数的值。
LoggingSharedKey	Fluentd **secure-forward** 插件的共享 secret。
LoggingDefaultSources	Fluentd 的默认日志源列表。覆盖此参数以禁用默认的日志记录源。使用 **LoggingExtraSources** 参数来定义额外的源配置。
LoggingExtraSources	此列表与 **LoggingDefaultSources** 参数以及可组合服务定义的任何日志记录源合并。

6.6. 覆盖日志文件的默认路径

如果您修改默认容器和修改包含服务日志文件的路径，还必须修改默认的日志文件路径。每个可组合服务都有一个 < service_name>LoggingSource 参数。例如，对于 nova-compute 服务，参数是 NovaComputeLoggingSource。

流程

1. **要覆盖 nova-compute 服务的默认路径，请在配置文件中添加 NovaComputeLoggingSource 参数的路径。**

```
NovaComputeLoggingSource:
    tag: openstack.nova.compute
    path: /some/other/path/nova-compute.log
```

tag 和 path 属性是 < service_name>LoggingSource 参数的必要 元素。在每个服务上，定义标签和路径，默认派生了其余值。

2. **您可以修改特定服务的格式。这会直接传递给 Fluentd 配置。LoggingDefaultFormat 参数的默认格式为 /(?<time>\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}.\d+) (<pid>\d+) (<priority>\S+) (<message>.*)$/使用以下语法：**

```
<service_name>LoggingSource:
   tag: <service_name>.tag
   path: <service_name>.path
   format: <service_name>.format
```

以下片段是一个更复杂的转换示例：

```
ServiceLoggingSource:
   tag: openstack.Service
   path: /var/log/containers/service/service.log
   format: multiline
   format_firstline: '/^\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}.\d{3} \d+ \S+ \S+ \[(req-\S+ \S+ \S+ \S+ \S+ \S+|-)\]/'
   format1: '/^(?<Timestamp>\S+ \S+) (?<Pid>\d+) (?<log_level>\S+) (?<python_module>\S+) (\[(req-(?<request_id>\S+) (?<user_id>\S+) (?<tenant_id>\S+) (?<domain_id>\S+) (?<user_domain>\S+) (?<project_domain>\S+)|-)\])? (?<Payload>.*)?$/'
```

6.7. 验证部署是否成功

要验证集中式日志记录是否已成功部署，请查看日志来了解输出是否与预期相符。您可以使用第三方视觉化软件，如 Kibana。