Red Hat OpenStack Platform 17.0

操作测量

跟踪物理和虚拟资源，并收集指标

Last Updated: 2024-04-05

Red Hat OpenStack Platform 17.0 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

摘要

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

目录

使开源包含更多

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

对红帽文档提供反馈

我们感谢您对文档提供反馈信息。与我们分享您的成功秘诀。

使用直接文档反馈(DDF)功能

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

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

第 1 章 操作测量简介

您可以使用 Red Hat OpenStack Platform (RHOSP)环境中的 Telemetry 服务组件来跟踪物理和虚拟资源，并使用在 Gnocchi 后端中保存聚合的数据收集守护进程收集部署中的 CPU 使用量和资源可用性。

您可以使用可用性和性能监控工具来测量和维护 RHOSP 环境。这些工具执行以下功能：

可用性监控

监控 RHOSP 环境中的所有组件，并确定任何组件当前是否处于中断或无法正常工作。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，并提供使用数据收集守护进程来存储和监控值的机制。此守护进程存储它收集的数据，如操作系统和日志文件。它还可在网络上提供数据。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，并预测将来的系统负载。

1.1. TELEMETRY 架构

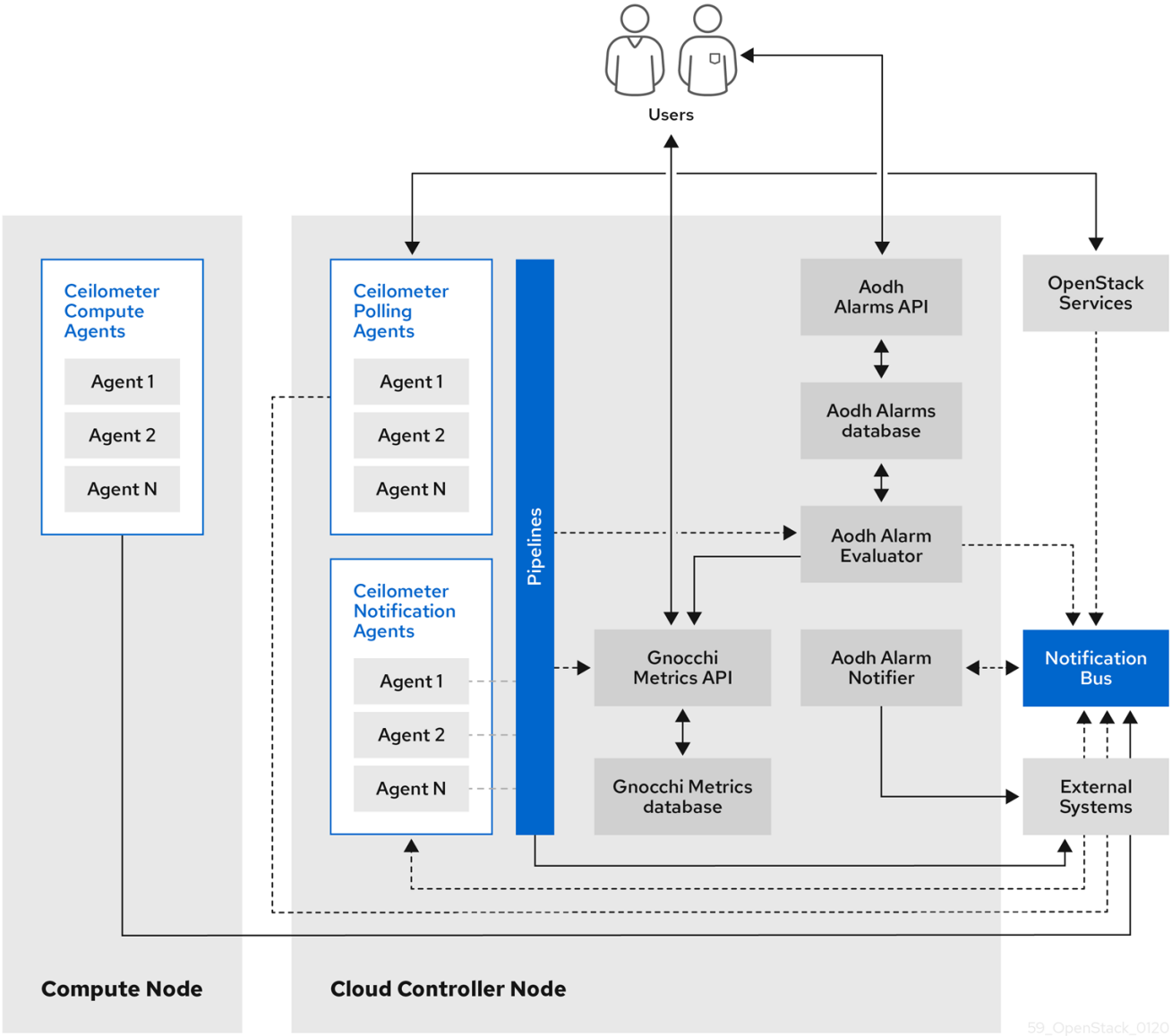
Red Hat OpenStack Platform (RHOSP) Telemetry 为基于 OpenStack 的云提供用户级别的使用数据。您可以使用数据进行客户计费、系统监控或警报。您可以配置 Telemetry 组件，以从现有 RHOSP 组件发送的通知（如计算使用事件）或轮询 RHOSP 基础架构资源（如 libvirt）收集数据。Telemetry 将收集的数据发布到各种目标，包括数据存储和消息队列。

Telemetry 由以下组件组成：

- **数据收集** ：Telemetry 使用 the 来收集指标和事件数据。如需更多信息，请参阅 第 1.2.1 节 “opendoi”。
- **Storage**: Telemetry 将指标数据存储在 Gnocchi 中。如需更多信息，请参阅 第 1.3 节 “使用 Gnocchi 的存储”。
- **警报服务**：Telemetry 使用 Alarming 服务 (Aodh) 根据定义了的规则针对 sVirt 收集的指标或事件数据触发操作。

收集数据后，您可以使用第三方工具来显示和分析指标数据，您可以使用 Alarming 服务为事件配置警报。

图 1.1. Telemetry 架构



1.1.1. 支持监控组件的状态

使用此表查看 Red Hat OpenStack Platform (RHOSP)中监控组件的支持状态。

表 1.1. 支持状态

组件	完全支持自	弃用中的	从后删除	备注
aodh	RHOSP 9	RHOSP 15		支持自动扩展用例。
opendoi	RHOSP 4			支持在自动扩展和服务 Telemetry Framework (STF) 用例中为 RHOSP 收集指标和事件。

组件	完全支持自	弃用中的	从后删除	备注
collectd	RHOSP 11	RHOSP 17.1		支持 STF 的基础架构指标集合。
gnocchi	RHOSP 9	RHOSP 15		支持自动扩展用例的指标存储。
panko	RHOSP 11	RHOSP 12，在 RHOSP 14 起不默认安装	RHOSP 17.0	
QDR	RHOSP 13	RHOSP 17.1		支持将指标和事件数据从 RHOSP 传输到 STF。

1.2. RED HAT OPENSTACK PLATFORM 中的数据收集

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

- OpenStack 组件级监控的 Dan。更多信息请参阅 第 1.2.1 节 “opendoi”。
- collectd 用于基础架构的监控。更多信息请参阅 第 1.2.2 节 “collectd”。

1.2.1. opendoi

Thorntail 是 OpenStack Telemetry 服务的默认数据收集组件，可提供在所有当前 OpenStack 核心组件中规范化和转换数据的功能。Dan 收集与 OpenStack 服务相关的 metering 和事件数据。收集的数据可根据部署配置访问。

Tailoring 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** 相同，使用参数 **--polling namespace-compute**。
- **中央代理(ceilometer-agent-central)**：在中央管理服务器上运行，以轮询未与实例或 Compute 节点关联的资源的资源利用率统计。您可以启动多个代理来水平扩展服务。这与轮询代理 **ceilometer-polling** 相同，它与参数 **--polling namespace-central** 一同运行。
- **通知代理(ceilometer-agent-notification)**：在中央管理服务器上运行，并使用消息队列的消息来构建事件和 metering 数据。数据发布到定义的目标。gnocchi 是默认目标。这些服务使用 RHOSP 通知总线进行通信。

Tailoring 代理使用发布程序将数据发送到对应的端点，如 Gnocchi。您可以在 **pipeline.yaml** 文件中配置此信息。

其他资源

- 有关发布者的详情请参考 第 1.2.1.1 节 “publishers”。

1.2.1.1. publishers

Telemetry 服务提供了多种传输方法，用于将收集的数据传输到外部系统。这个数据的消费者会有所不同，例如监控系统，用于接受数据丢失，以及需要可靠数据传输的计费系统。Telemetry 提供了满足两种系统类型的要求的方法。您可以使用服务的 publisher 组件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。一个链可以包含多个发布者。

默认发布者类型是 Gnocchi。当您启用 Gnocchi publisher 时，测量和资源信息会推送到 Gnocchi 用于时间序列优化存储。确保您在 Identity 服务中注册 Gnocchi，来通过 Identity 服务发现准确的路径。

配置发布者参数
您可以为 Telemetry 服务中的每个数据点配置多发布程序，允许多次向多个目的地发布相同的技术分段或事件，每种传输方法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者参数和默认值，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 overcloud。在 **openstack overcloud deploy** 命令中包含修改后的 YAML 文件。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>
```

其他资源

- 有关参数的更多信息，请参阅 *Overcloud 参数指南中的* 遥测参数 和高级 *Overcloud 自定义指南中的* 参数。

1.2.2. collectd

性能监控定期收集系统信息，并提供使用数据收集代理以各种方式存储和监控值的机制。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。

1.3. 使用 GNOCCHI 的存储

gnocchi 是一个开源时间序列数据库。它以非常大的方式存储指标，并提供对操作器和用户的指标和资源的访问。gnocchi 使用归档策略来定义要计算的聚合以及要保留的聚合数量；以及索引器驱动程序来存储所有资源、归档策略和指标的索引。

1.3.1. 归档策略：在一个时间序列数据库中同时提供简短和长期数据

归档策略定义了要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、最大值、平均、百分比和标准差异。这些聚合在称为粒度的一段时间内计算，并保留在特定时间。

归档策略定义了如何聚合指标及其存储时间。每个归档策略都定义为一个时间增加的点数。

例如，如果您的归档策略定义了 10 点策略，其粒度为 1 秒，则时间序列存档最多会保持 10 秒，代表一个聚合超过 1 秒。这意味着，时间序列最大会在最新的点和旧点之间保留 10 秒的数据。

归档策略还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。

1.3.2. 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。它还负责将资源链接到指标。Red Hat OpenStack Platform director 默认安装 indexer 驱动程序。您需要数据库来索引 Gnocchi 处理的所有资源和指标。支持的驱动程序是 MySQL。

1.3.3. gnocchi Metric-as-a-Service 术语

此表包含用于 Metric-as-a-Service 功能的常用术语的定义。

表 1.2. metric-as-a-Service 术语

术语	定义
聚合方法	用于将多个测量聚合到聚合的功能。例如，min 聚合方法将不同测量的值聚合到时间范围中所有测量的最小值。
聚合	根据归档策略，从多个测量生成的数据点元组。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。归档策略决定了在指标中保留聚合的时长以及聚合方式（聚合方法）。
粒度	指标聚合时间序列中的两个聚合之间的时间。
measure	API 发送到时间序列数据库的传入数据点元组。测量由时间戳和值组成。
指标	存储由 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	代表您将指标与其关联的基础架构中的任何实体。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。

1.4. 显示指标数据

您可以使用以下工具来显示和分析指标数据：

- **Grafana**：一个开源的指标分析和视觉化套件。Grafana 最常用于视觉化用于基础架构和应用程序分析的时间序列数据。
- **Red Hat CloudForms**：IT 公司用来控制用户的自助服务功能的基础架构管理平台，以便在虚拟机和私有云之间配置、管理和确保合规性。

其他资源

- 有关 Grafana 的更多信息，请参阅 第 1.4.1 节 “使用和连接 Grafana 来显示数据”。
- 有关 Red Hat Cloudforms 的更多信息，请参阅产品文档。

1.4.1. 使用和连接 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 计划操作测量

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。

- 有关 IaaS 测量的详情，请参考 第 2.1 节 “法国测量”。
- 有关 collectd 的更多信息，请参阅 第 2.2 节 “collectd 测量”。

2.1. 法国测量

有关 IaaS 措施的完整列表，请参阅 https://docs.openstack.org/ceilometer/train/admin/telemetry-measurements.html

2.2. COLLECTD 测量

以下测量很有用的 collectd 指标：

- disk
- interface
- load
- memory
- tcpconns

2.3. 计划数据存储

gnocchi 存储一组数据点，其中每个数据点都是聚合的。存储格式使用不同的技术进行压缩。因此，要计算时间序列数据库的大小，您必须根据最糟糕的情况估算大小。

警告

使用 Red Hat OpenStack Platform (RHOSP) Object Storage (swift)进行时间序列数据库(Gnocchi)存储只支持小型和非生产环境。

流程

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

   例如，如果要保留一分钟分辨率的数据年，请使用公式：

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

2. 计算时间序列数据库的大小：
   size in bytes = data point X 8 字节

   如果您在示例中应用此公式，则结果为 4.1 MB：

size in bytes = 525600 point X 8 bytes = 4204800 bytes = 4.1 MB

这个值是单个聚合时间序列数据库的预期存储要求。如果您的归档策略使用多个聚合方法(min, max, mean, sum, std, count)，请按照您使用的聚合方法数量乘以这个值。

其他资源

- 第 1.3.1 节 “归档策略：在一个时间序列数据库中同时提供简短和长期数据”
- 第 2.4 节 “规划和管理归档策略”

2.4. 规划和管理归档策略

归档策略定义了如何聚合指标以及将指标存储在时间序列数据库中的时间。归档策略定义为一段时间内的点数。

如果您的归档策略定义了 10 点策略，其粒度为 1 秒，则时间序列存档将最多保留 10 秒，代表一个聚合超过 1 秒。这意味着时间序列在最新点和旧点之间最多保留 10 秒数据。归档策略还定义了要使用的聚合方法。默认设置为参数 **default_aggregation_methods**，其中默认值被设置为 **mean**, **min**, **max**。总、**std**、计数.因此，根据用例，归档策略和粒度可能会有所不同。

要规划归档策略，请确保您熟悉以下概念：

- 指标。更多信息请参阅 第 2.4.1 节 “指标”。
- 测量。更多信息请参阅 第 2.4.2 节 “创建自定义措施”。
- 聚合。更多信息请参阅 第 2.4.4 节 “计算时间序列聚合的大小”。
- 指标 worker。更多信息请参阅 第 2.4.5 节 “测量的 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 2.4.6 节 “创建归档策略”。
2. 管理归档策略。更多信息请参阅 第 2.4.7 节 “管理归档策略”。
3. 创建归档策略规则。更多信息请参阅 第 2.4.8 节 “创建归档策略规则”。

2.4.1. 指标

gnocchi 提供名为 *metric* 的对象类型。指标是您可以测量服务器的 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
- 名称
- 用于存储和聚合措施的归档策略

其他资源

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

2.4.1.1. 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，<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，归档策略属性已被修复且不可更改。您可以通过 **archive_policy** 端点更改归档策略的定义。

2.4.2. 创建自定义措施

测量是 API 发送到 Gnocchi 的传入数据点元组。它由一个时间戳和一个值组成。您可以创建自己的自定义措施。

流程

- 创建自定义方法：

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

2.4.3. 默认归档策略

默认情况下，Gnocchi 有以下归档策略：

- 低
  - 5 分钟的粒度超过 30 天
  - 使用的聚合方法： **default_aggregation_methods**
  - 每个指标的最大估计大小：406 KiB
- 中
  - 7 天中的 1 分钟粒度
  - 1 小时的粒度超过 365 天
  - 使用的聚合方法： **default_aggregation_methods**
  - 每个指标的最大估计大小： 887 KiB
- high
  - 1 秒的粒度(1 小时)
  - 1 分钟的粒度超过 1 周
  - 1 小时的粒度超过 1 年
  - 使用的聚合方法： **default_aggregation_methods**

- 每个指标的最大估计大小： 1 057 KiB

- bool
  - 1 年内的 1 秒粒度
  - 使用的聚合方法：last
  - 每个指标的最大选择大小： 1 539 KiB
  - 每个指标的最大 pessimistic 大小： 277 172 KiB

2.4.4. 计算时间序列聚合的大小

gnocchi 存储一组数据点，其中每个点都是聚合。存储格式使用不同的技术进行压缩。因此，根据最糟的情况，计算时间序列的大小估计，如下例所示。

流程

1. 使用此公式计算点数：
   点数 = 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. 要以字节为单位计算点大小，请使用此公式：
   size in bytes = point X 8 字节数

   size in bytes = 525600 point X 8 bytes = 4204800 bytes = 4.1 MB

   这个值是单个聚合时间序列的预计存储要求。如果您的归档策略使用多个聚合方法 - min, max, mean, sum, std, count - 根据您使用的聚合方法数量乘以这个值。

2.4.5. 测量的 worker

您可以使用指标守护进程来处理测量、创建聚合、存储测量并删除指标。指标守护进程负责 Gnocchi 中大多数 CPU 用量和 I/O 作业。每个指标的归档策略决定了指标守护进程执行的速度。指标检查传入的存储是否有新的措施。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**。

2.4.6. 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，<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名称和值。

2.4.7. 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>
```

2.4.8. 创建归档策略规则

归档策略规则定义指标和归档策略之间的映射。这样，用户可以预定义规则，以便根据匹配的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>
```

2.5. 验证 RED HAT OPENSTACK PLATFORM 部署

您可以使用 **openstack metric** 命令验证部署是否成功。

流程

- 验证部署：

```
(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  |
+-----------------------------------------------------+-------+
```

第 3 章 管理警报

您可以使用 Telemetry Alarming 服务 (Aodh) 根据定义的规则针对 sVirt 收集的指标或事件数据触发操作。

警报可能处于以下状态之一：

ok

指标或事件处于可接受的状态。

触发

指标或事件不在定义的 ok 状态之外。

数据不足

警报状态未知。这可能是出于几个原因，例如，请求的粒度没有数据，检查还没有执行，以此类推。

3.1. 查看现有警报

您可以查看现有的 Telemetry 警报信息，并列出分配给资源的分段，以检查指标的当前状态。

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分段，请指定资源的 UUID。例如：

```
# openstack metric resource show 22592ae1-922a-4f51-b935-20c938f48753

| Field                | Value                                                                     |
+----------------------+---------------------------------------------------------------------------+
| created_by_project_id | 1adaed3aaa7f454c83307688c0825978                                          |
| created_by_user_id    | d8429405a2764c3bb5184d29bd32c46a                                          |
| creator               |
d8429405a2764c3bb5184d29bd32c46a:1adaed3aaa7f454c83307688c0825978 |
| ended_at              | None                                                                      |
| id                    | 22592ae1-922a-4f51-b935-20c938f48753                                      |
| metrics               | cpu: a0375b0e-f799-47ea-b4ba-f494cf562ad8                                 |
|                       | disk.ephemeral.size: cd082824-dfd6-49c3-afdf-6bfc8c12bd2a                 |
|                       | disk.root.size: cd88dc61-ba85-45eb-a7b9-4686a6a0787b                      |
|                       | memory.usage: 7a1e787c-5fa7-4ac3-a2c6-4c3821eaf80a                        |
|                       | memory: ebd38ef7-cdc1-49f1-87c1-0b627d7c189e                              |
|                       | vcpus: cc9353f1-bb24-4d37-ab8f-d3e887ca8856                               |
| original_resource_id  | 22592ae1-922a-4f51-b935-20c938f48753                                      |
```

```
| project_id          | cdda46e0b5be4782bc0480dac280832a                                  |
| revision_end        | None                                                              |
| revision_start      | 2021-09-16T17:00:41.227453+00:00                                  |
| started_at          | 2021-09-16T16:17:08.444032+00:00                                  |
| type                | instance                                                          |
| user_id             | f00de1d74408428cadf483ea7dbb2a83                                  |
+---------------------+-------------------------------------------------------------------+
```

3.2. 创建警报

使用 Telemetry Alarming 服务(aodh)创建在满足特定条件时触发的警报，例如达到阈值时触发。在本例中，当单个实例的平均 CPU 使用率超过 80% 时，警报会激活并添加日志条目。

流程

1. 归档策略在部署过程中预先填充，您很少需要创建新的归档策略。但是，如果没有配置的归档策略，您必须创建一个。要创建一个归档策略，它在 5s * 86400 points (5 天) 内创建指标数据，请使用以下命令：

```
# openstack archive-policy create <name> \
    -d granularity:5s,points:86400 \
    -b 3 -m mean -m rate:mean
```

+ 将 <*name* > 替换为归档策略的名称。

注意

确保将 Telemetry Alarming 服务的评估周期的值设置为大于 60 的整数。sVirt 轮询间隔链接到评估周期。确保将 sVirt 轮询间隔值设置为 60 到 600 之间的数字，并确保该值大于 Telemetry Alarming 服务的评估周期的值。如果 IaaS 轮询间隔太低，则可能会严重影响系统负载。

2. 创建警报并使用查询来隔离实例的特定 ID，以满足监控目的。以下示例中的实例的 ID 是 94619081-abf5-4f1f-81c7-9cedaa872403。

注意

要计算阈值，请使用以下公式： 1,000,000,000 x {granularity} x {percentage_in_decimal}

```
# openstack alarm create \
--type gnocchi_aggregation_by_resources_threshold \
--name cpu_usage_high \
--granularity 5
--metric cpu \
--threshold 48000000000 \
--aggregation-method rate:mean \
--resource-type instance \
--query '{"=": {"id": "94619081-abf5-4f1f-81c7-9cedaa872403"}}' --alarm-action 'log://'
+---------------------------+-------------------------------------------------------+
| Field                     | Value                                                 |
+---------------------------+-------------------------------------------------------+
| aggregation_method        | rate:mean                                             |
```

```
| 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             | 5                                                    |
| insufficient_data_actions | []                                                 |
| metric                  | cpu                                                  |
| 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               | 48000000000.0                                        |
| time_constraints        | []                                                   |
| timestamp               | 2016-12-09T05:18:53.326000                           |
| type                    | gnocchi_aggregation_by_resources_threshold           |
| user_id                 | 32d3f2c9a234423cb52fb69d3741dbbc                     |
+-------------------------+------------------------------------------------------+
```

3.3. 编辑警报

编辑警报时，您可以增加或减少警报的值阈值。

流程

- 要更新阈值，请使用 **openstack alert 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
```

3.4. 禁用警报

您可以禁用并启用警报。

流程

- 禁用警报：

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

3.5. 删除警报

使用 **openstack alert delete** 命令删除警报。

流程

- 要删除警报，请输入以下命令：

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

3.6. 示例：监控实例的磁盘活动

本例演示了如何使用作为 Telemetry Alarming 服务一部分的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，并选择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 创建一个警报，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** 参数进一步等待查询：

```
# 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                     |
+-----------------------+------------------------------------------------------+
```

3.7. 示例：监控 CPU 使用

要监控实例的性能，请检查 Gnocchi 数据库，以识别您可以监控哪些指标，如内存或 CPU 使用量。

流程

1. 要识别您可以监控的指标，请使用实例 UUID 输入 **openstack metric resource show** 命令：

```
$ openstack metric resource show --type instance 22592ae1-922a-4f51-b935-20c938f48753

+----------------------+--------------------------------------------------------------------+
| Field                | Value                                                              |
+----------------------+--------------------------------------------------------------------+
| availability_zone    | nova                                                               |
| created_at           | 2021-09-16T16:16:24+00:00                                          |
| created_by_project_id | 1adaed3aaa7f454c83307688c0825978                                  |
| created_by_user_id   | d8429405a2764c3bb5184d29bd32c46a                                   |
| creator              |
d8429405a2764c3bb5184d29bd32c46a:1adaed3aaa7f454c83307688c0825978 |
| deleted_at           | None                                                               |
| display_name         | foo-2                                                              |
| ended_at             | None                                                               |
| flavor_id            | 0e5bae38-a949-4509-9868-82b353ef7ffb                               |
| flavor_name          | workload_flavor_0                                                  |
| host                 | compute-0.redhat.local                                             |
| id                   | 22592ae1-922a-4f51-b935-20c938f48753                               |
| image_ref            | 3cde20b4-7620-49f3-8622-eeacbdc43d49                               |
| launched_at          | 2021-09-16T16:17:03+00:00                                          |
| metrics              | cpu: a0375b0e-f799-47ea-b4ba-f494cf562ad8                          |
|                      | disk.ephemeral.size: cd082824-dfd6-49c3-afdf-6bfc8c12bd2a          |
|                      | disk.root.size: cd88dc61-ba85-45eb-a7b9-4686a6a0787b               |
|                      | memory.usage: 7a1e787c-5fa7-4ac3-a2c6-4c3821eaf80a                 |
|                      | memory: ebd38ef7-cdc1-49f1-87c1-0b627d7c189e                       |
|                      | vcpus: cc9353f1-bb24-4d37-ab8f-d3e887ca8856                        |
| original_resource_id | 22592ae1-922a-4f51-b935-20c938f48753                               |
| project_id           | cdda46e0b5be4782bc0480dac280832a                                   |
| revision_end         | None                                                               |
| revision_start       | 2021-09-16T17:00:41.227453+00:00                                   |
| server_group         | None                                                               |
| started_at           | 2021-09-16T16:17:08.444032+00:00                                   |
| type                 | instance                                                           |
| user_id              | f00de1d74408428cadf483ea7dbb2a83                                   |
+----------------------+--------------------------------------------------------------------+
```

因此，指标值列出了您可以使用 Alarming 服务监控的组件，如 **cpu**。

2. 要监控 CPU 用量，请使用 **cpu** 指标：

```
$ openstack metric show --resource-id 22592ae1-922a-4f51-b935-20c938f48753 cpu
+-------------------------------+-----------------------------------------------------------------+
| Field                         | Value                                                           |
+-------------------------------+-----------------------------------------------------------------+
| archive_policy/name           | ceilometer-high-rate                                            |
| creator                       |
d8429405a2764c3bb5184d29bd32c46a:1adaed3aaa7f454c83307688c0825978 |
| id                            | a0375b0e-f799-47ea-b4ba-f494cf562ad8                            |
| name                          | cpu                                                             |
| resource/created_by_project_id | 1adaed3aaa7f454c83307688c0825978
|
| resource/created_by_user_id   | d8429405a2764c3bb5184d29bd32c46a
|
| resource/creator              |
d8429405a2764c3bb5184d29bd32c46a:1adaed3aaa7f454c83307688c0825978 |
| resource/ended_at             | None                                                            |
| resource/id                   | 22592ae1-922a-4f51-b935-20c938f48753                            |
| resource/original_resource_id | 22592ae1-922a-4f51-b935-20c938f48753
|
| resource/project_id           | cdda46e0b5be4782bc0480dac280832a                                |
| resource/revision_end         | None                                                            |
| resource/revision_start       | 2021-09-16T17:00:41.227453+00:00                                |
| resource/started_at           | 2021-09-16T16:17:08.444032+00:00                                |
| resource/type                 | instance                                                        |
| resource/user_id              | f00de1d74408428cadf483ea7dbb2a83                                |
| unit                          | ns                                                              |
+-------------------------------+-----------------------------------------------------------------+
```

archive_policy 定义计算 std, count, min, max, sum, mean 值的聚合间隔。

3. 检查 **cpu** 指标当前选择的归档策略：

```
$ openstack metric archive-policy show ceilometer-high-rate

+--------------------+-----------------------------------------------------------------+
| Field              | Value                                                           |
+--------------------+-----------------------------------------------------------------+
| aggregation_methods | rate:mean, mean                                                |
| back_window        | 0                                                               |
| definition         | - timespan: 1:00:00, granularity: 0:00:01, points: 3600         |
|                    | - timespan: 1 day, 0:00:00, granularity: 0:01:00, points: 1440  |
|                    | - timespan: 365 days, 0:00:00, granularity: 1:00:00, points: 8760 |
| name               | ceilometer-high-rate                                            |
+--------------------+-----------------------------------------------------------------+
```

4. 使用 Alarming 服务创建查询 **cpu** 的监控任务。此任务会根据您指定的设置触发事件。例如，要在延长持续时间超过 80% 的实例时引发日志条目，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 \
--threshold 800,000,000.0 \
--comparison-operator ge \
--aggregation-method mean \
--granularity 300 \
--evaluation-periods 1 \
--alarm-action 'log://' \
--ok-action 'log://' \
--resource-type instance \
--resource-id 22592ae1-922a-4f51-b935-20c938f48753

+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| aggregation_method        | rate:mean                            |
| alarm_actions             | ['log:']                             |
| alarm_id                  | c7b326bd-a68c-4247-9d2b-56d9fb18bf38 |
| comparison_operator       | ge                                   |
| description               | High CPU usage                       |
| enabled                   | True                                 |
| evaluation_periods        | 1                                    |
| granularity               | 300                                  |
| insufficient_data_actions | []                                   |
| metric                    | cpu                                  |
| name                      | high-cpu                             |
| ok_actions                | ['log:']                             |
| project_id                | cdda46e0b5be4782bc0480dac280832a     |
| repeat_actions            | False                                |
| resource_id               | 22592ae1-922a-4f51-b935-20c938f48753 |
| resource_type             | instance                             |
| severity                  | low                                  |
| state                     | insufficient data                    |
| state_reason              | Not evaluated yet                    |
| state_timestamp           | 2021-09-21T08:02:57.090592           |
| threshold                 | 800000000.0                          |
| time_constraints          | []                                   |
| timestamp                 | 2021-09-21T08:02:57.090592           |
| type                      | gnocchi_resources_threshold          |
| user_id                   | f00de1d74408428cadf483ea7dbb2a83     |
+---------------------------+--------------------------------------+
```

- compare-operator：ge 运算符定义了在 CPU 使用量大于或等于 80% 时触发警报。
- granularity ：指标关联了一个归档策略，策略可以有不同的粒度。例如，一个月有 5 分钟的聚合时间为 1 小时 + 1 小时聚合。granularity 值必须与归档策略中描述的持续时间匹配。
- assessment-periods：在警报触发前需要传递的粒度周期数。例如，如果将此值设置为 2，则在警报触发前，CPU 用量必须超过 80%。
- [u'log://'] ：当您将 **alerts_actions** 或 **ok_actions** 设置为 **[u'log://']**, events （如事件）被触发或返回到正常状态时，会将警报记录到 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(ADVANCED_actions)，以及当它返回到正常状态(ok_actions)时，如 Webhook URL。

3.8. 查看警报历史记录

要检查是否已触发了特定的警报，您可以查询警报历史记录并查看事件信息。

流程

- 使用 **openstack alert-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 4 章 安装和配置日志服务

Red Hat OpenStack Platform (RHOSP)将信息写入特定的日志文件中 ; 您可以使用这些信息排除和监控系统事件。日志收集代理 Rsyslog 在客户端上收集日志，并将这些日志发送到服务器端运行的 Rsyslog 实例。服务器端 Rsyslog 实例将日志记录重定向到 Elasticsearch 以进行存储。

注意

您不需要手动将各个日志文件附加到支持问题单中。**sosreport** 工具自动收集所需的日志。

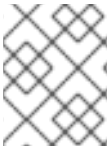
4.1. 集中式日志系统架构和组件

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。Rsyslog 服务提供客户端集中式日志记录(CL)。

所有 RHOSP 服务都会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，将这些日志收集到一个中央位置，简化了调试和管理。

通过集中式日志记录，一个中央位置可在整个 RHOSP 环境中查看日志。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 Identity、Compute 等 OpenStack 服务。集中式日志记录工具链由以下组件组成：

- 日志集合代理(Rsyslog)
- 数据存储(ElasticSearch)
- api/Presentation Layer (Grafana)



注意

Red Hat OpenStack Platform director 不会部署用于集中式日志记录的服务器端组件。红帽不支持服务器端组件，包括 Elasticsearch 数据库和 Grafana。

4.2. 使用 ELASTICSEARCH 启用集中式日志记录

要启用集中式日志记录，您必须指定 **OS::TripleO::Services::Rsyslog** 可组合服务的实现。

注意

Rsyslog 服务只使用 Elasticsearch 作为集中日志记录的数据存储。

前提条件

- Elasticsearch 已在服务器端安装。

流程

- 将日志记录环境文件添加到 **overcloud 部署** 命令中，以及任何与您的环境相关的其他环境文件，如下例所示：

```
openstack overcloud deploy \
<existing_overcloud_environment_files> \
```

```
-e /usr/share/openstack-tripleo-heat-templates/environments/logging-environment-
rsyslog.yaml
```

使用作为现有部署一部分的环境文件列表替换。**<existing_overcloud_environment_files>**

4.3. 配置日志记录功能

要配置日志记录功能，修改 **logging-environment-rsyslog.yaml** 文件中的 **RsyslogElasticsearchSetting** 参数。

流程

1. 将 **tripleo-heat-templates/environments/logging-environment-rsyslog.yaml** 文件复制到您的主目录中。
2. 在 **RsyslogElasticsearchSetting** 参数中创建条目以适合您的环境。以下片段是 **RsyslogElasticsearchSetting** 参数的示例配置：

```
parameter_defaults:
  RsyslogElasticsearchSetting:
    uid: "elastic"
    pwd: "yourownpassword"
    skipverifyhost: "on"
    allowunsignedcerts: "on"
    server: "https://log-store-service-
telemetry.apps.stfcloudops1.lab.upshift.rdu2.redhat.com"
    serverport: 443
```

其他资源

- 有关可配置参数的详情请参考 第 4.3.1 节 “可配置日志记录参数”。

4.3.1. 可配置日志记录参数

此表包含用于在 Red Hat OpenStack Platform (RHOSP)中配置日志记录功能的日志参数的描述。您可以在 **tripleo-heat-templates/deployment/logging/rsyslog-container-puppet.yaml** 文件中找到这些参数。

表 4.1. 可配置日志记录参数

参数	Description
RsyslogElasticsearchSetting	配置 **rsyslog-elasticsearch** 插件。如需更多信息，请参阅 https://www.rsyslog.com/doc/v8-stable/configuration/modules/omelasticsearch.html。
RsyslogElasticsearchTlsCACert	包含发布 Elasticsearch 服务器证书的 CA 的 CA 证书内容。
RsyslogElasticsearchTlsClientCert	包含针对 Elasticsearch 进行客户端证书授权的客户端证书内容。

参数	Description
RsyslogElasticsearchTlsClientKey	包含与 cert **RsyslogElasticsearchTlsClientCert** 对应的私钥内容。

4.4. 管理日志

容器化服务日志文件存储在 **/var/log/containers/<service>** 中，如 **/var/log/containers/cinder**。容器内运行的服务的日志文件存储在本地。可用的日志可能会因启用和禁用的服务而异。

以下示例强制 **logrotate** 任务在达到 **10** MB 时创建新日志文件，并保留 **14** 天的日志文件。

```
parameter_defaults
  LogrotateRotate: '14'
  LogrotatePurgeAfterDays: '14'
  LogrotateMaxsize: '10M'
```

若要自定义日志轮转参数，可在环境模板中包括这些参数 **_defaults**，然后部署 overcloud。

```
openstack overcloud deploy \
--timeout 100 \
--templates /usr/share/openstack-tripleo-heat-templates \
... \
-e /home/stack/templates/rotate.yaml \
--log-file overcloud_deployment_90.log
```

验证：在任何 overcloud 节点上，确保 **logrotate_crond** 已更新：

```
[root@compute0 ~]# podman exec -it logrotate_crond cat /etc/logrotate-crond.conf
/var/log/containers/*/*log /var/log/containers/*/*/*log /var/log/containers/*/*err {
  daily
  rotate 14
  maxage 14
  # minsize 1 is required for GDPR compliance, all files in
  # /var/log/containers not managed with logrotate will be purged!
  minsize 1
  # Do not use size as it's not compatible with time-based rotation rules
  # required for GDPR compliance.
  maxsize 10M
  missingok
  notifempty

  copytruncate


  delaycompress


  compress


}
```

在以下示例中，**nova-compute.log** 文件已被轮转一次。

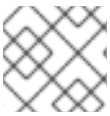
```
[root@compute0 ~]# ls -lah /var/log/containers/nova/
total 48K
drwxr-x---. 2 42436 42436  79 May 12 09:01 .
drwxr-x---. 7 root  root   82 Jan 21  2021 ..
-rw-r--r--. 1 42436 42436 12K May 12 14:00 nova-compute.log
-rw-r--r--. 1 42436 42436 33K May 12 09:01 nova-compute.log.1
-rw-r--r--. 1 42436 42436   0 Jan 21  2021 nova-manage.log
```

日志文件轮转过程发生在 **logrotate_crond** 容器中。**/var/spool/cron/root** 配置文件被读取，其配置被发送到进程。

验证：确保配置存在于任何 Controller 节点上：

```
[root@controller0 ~]# podman exec -it logrotate_crond /bin/sh
()[root@9410925fded9 /]$ cat /var/spool/cron/root
# HEADER: This file was autogenerated at 2021-01-21 16:47:27 +0000 by puppet.
# HEADER: While it can still be managed manually, it is definitely not recommended.
# HEADER: Note particularly that the comments starting with 'Puppet Name' should
# HEADER: not be deleted, as doing so could cause duplicate cron jobs.
# Puppet Name: logrotate-crond
PATH=/bin:/usr/bin:/usr/sbin SHELL=/bin/sh
0 * * * * sleep `expr ${RANDOM} \% 90`; /usr/sbin/logrotate -s /var/lib/logrotate/logrotate-crond.status
/etc/logrotate-crond.conf 2>&1|logger -t logrotate-crond
```

/var/lib/config-data/puppet-generated/crond/etc/logrotate-crond.conf 文件绑定到 **logrotate_crond** 容器中的 **/etc/logrotate-crond.conf**。



注意

由于历史原因，旧的配置文件会被保留，但不会使用它们。

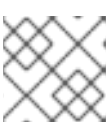
4.5. 覆盖日志文件的默认路径

如果修改默认容器，修改包含服务日志文件的路径，还必须修改默认日志文件路径。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

- 要覆盖 nova-compute 服务的默认路径，请将路径添加到配置文件中的 **NovaComputeLoggingSource** 参数：

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



注意

对于每个服务，定义 **tag** 和 **file**。默认情况下，其他值会被派生。

1. 您可以修改特定服务的格式。这会直接传递给 Rsyslog 配置。**LoggingDefaultFormat** 参数的默认格式是 /(?<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>.*)?$/'
```

4.6. 修改日志记录的格式

您可以修改特定服务的日志记录开始格式。这会直接传递给 Rsyslog 配置。

Red Hat OpenStack Platform (RHOSP) 日志记录的默认格式是 ('^[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}:[0-9]{2}:[0-9]{2}(.[0-9]+ [0-9]+)?（调试|INFO|WARNING|ERROR）')。

流程

- 要添加用于解析日志记录开始的不同正则表达式，请在配置中添加 **startmsg.regex** ：

```
NovaComputeLoggingSource:
      tag: openstack.nova.compute
      file: /some/other/path/nova-compute.log
      startmsg.regex: "^[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}:[0-9]{2}:[0-9]{2}(.[0-9]+ \\+[0-9]+)? [A-Z]+ \\([a-z]+\\)
```

4.7. 测试 RSYSLOG 和 ELASTICSEARCH 之间的连接

在客户端中，您可以验证 Rsyslog 和 Elasticsearch 之间的通信。

流程

- 进入 Rsyslog 容器中 Elasticsearch 连接日志文件 **/var/log/rsyslog/omelasticsearch.log**，或主机上的 **/var/log/containers/rsyslog/omelasticsearch.log**。如果此日志文件不存在，或者日志文件不存在但没有包含日志，则没有连接问题。如果日志文件存在并包含日志，Rsyslog 未能成功连接。

注意

要从服务器端测试连接，请查看 Elasticsearch 日志以了解连接问题。

4.8. 服务器端日志记录

如果 Elasticsearch 集群正在运行，您必须在 **logging-environment-rsyslog.yaml** 文件中配置 **RsyslogElasticsearchSetting** 参数，以连接在 overcloud 节点上运行的 Rsyslog。要配置 **RsyslogElasticsearchSetting** 参数，请参阅 https://www.rsyslog.com/doc/v8-stable/configuration/modules/omelasticsearch.html

4.9. TRACEBACKS

当您遇到问题并开始故障排除时，您可以使用回溯日志诊断问题。日志文件中，回溯通常有几行信息，它们都与同一问题相关。

rsyslog 提供了一个正则表达式，用于定义日志记录如何启动。每个日志记录通常以时间戳开头，回溯的第一行则是包含此信息的唯一行。rsyslog 捆绑了带有第一行的缩进记录，并将其作为一个日志记录发送。

对于这个行为的配置选项，使用 <Service>LoggingSource 中的 **startmsg.regex**。以下正则表达式是 director 中所有 <service>LoggingSource 参数的默认值：

```
startmsg.regex='^[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}:[0-9]{2}:[0-9]{2}(.[0-9]+ [0-9]+)?
(DEBUG|INFO|WARNING|ERROR) '
```

当此默认值与添加或修改的 **LoggingSource** 的日志记录不匹配时，您必须相应地更改 **startmsg.regex**。

4.10. 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。

4.10.1. 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

4.10.2. Block Storage (cinder)日志文件

Service	服务名称	日志路径
块存储 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

Service	服务名称	日志路径
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

4.10.3. 计算(nova)日志文件

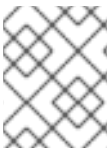
Service	服务名称	日志路径
OpenStack Compute API 服务	openstack-nova-api.service	/var/log/containers/nova/nova-api.log
OpenStack Compute 证书服务器	openstack-nova-cert.service	/var/log/containers/nova/nova-cert.log
OpenStack Compute 服务	openstack-nova-compute.service	/var/log/containers/nova/nova-compute.log
OpenStack 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 Compute 调度程序服务	openstack-nova-scheduler.service	/var/log/containers/nova/nova-scheduler.log

4.10.4. 仪表板(horizon)日志文件

Service	服务名称	日志路径
某些用户交互的日志	仪表板界面	/var/log/containers/horizon/horizon.log

Apache HTTP 服务器为仪表板 Web 界面使用几个额外的日志文件，您可以使用 Web 浏览器或命令行客户端（如 keystone 和 nova）访问这些日志文件。下表的日志文件有助于跟踪仪表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
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**，它将存储在同一主机上运行的其他 Web 服务报告的错误。

4.10.5. Identity Service (keystone)日志文件

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

4.10.6. 镜像服务(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

4.10.7. 网络(neutron)日志文件

Service	服务名称	日志路径
OpenStack Neutron DHCP 代理	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

4.10.8. 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，所有对象存储日志文件都会使用 local0、local1 和 local2 syslog 工具进入 **/var/log/containers/swift/swift.log**。

Object Storage 的日志消息分为两大类：由 REST API 服务以及后台守护进程提供这些类别。API 服务消息包含每行 API 请求，类似于流行的 HTTP 服务器；前端(Proxy)和后端（帐户、容器、对象）服务都发布此类消息。守护进程消息不太结构化，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都会生成消息，源身份始终位于行的开头。

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消息示例：

```
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
```

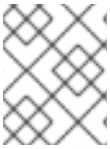
```
Apr 27 17:08:16 rhev-a24c-02 account-replicator: Replication run OVER
Apr 27 17:08:16 rhev-a24c-02 account-replicator: Attempted to replicate 5 dbs in 0.12589 seconds
(39.71876/s)
Apr 27 17:08:16 rhev-a24c-02 account-replicator: Removed 0 dbs
Apr 27 17:08:16 rhev-a24c-02 account-replicator: 10 successes, 0 failures
```

4.10.9. 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
OpenStack Heat Engine 服务	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

4.10.10. 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 共享服务	openstack-manila-share.service	/var/log/containers/manila/share.log



注意

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

4.10.11. Telemetry (ceilometer)日志文件

Service	服务名称	日志路径
OpenStack ceilometer 通知代理	ceilometer_agent_notification	/var/log/containers/ceilometer/agent-notification.log
OpenStack ceilometer Central 代理	ceilometer_agent_central	/var/log/containers/ceilometer/central.log

Service	服务名称	日志路径
OpenStack ceilometer collection	openstack-ceilometer-collector.service	/var/log/containers/ceilometer/collector.log
OpenStack ceilometer 计算代理	ceilometer_agent_compute	/var/log/containers/ceilometer/compute.log

4.10.12. 支持服务的日志文件

以下服务由 OpenStack 核心组件使用，并且具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 （用于简单身份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
虚拟网络交换机(Open vSwitch)	openvswitch-nonetwork.service	/var/log/openvswitch/ovsdb-server.log /var/log/openvswitch/ovs-vswitchd.log

4.10.13. aodh （普通服务）日志文件

Service	容器名称	日志路径
警报 API	aodh_api	/var/log/containers/httpd/aodh-api/aodh_wsgi_access.log
警报评估器日志	aodh_evaluator	/var/log/containers/aodh/aodh-evaluator.log
警报监听程序	aodh_listener	/var/log/containers/aodh/aodh-listener.log
警报通知	aodh_notifier	/var/log/containers/aodh/aodh-notifier.log

4.10.14. gnocchi （指标存储）日志文件

Service	容器名称	日志路径
Gnocchi API	gnocchi_api	/var/log/containers/httpd/gnocchi-api/gnocchi_wsgi_access.log

Service	容器名称	日志路径
gnocchi metricd	gnocchi_metricd	/var/log/containers/gnocchi/gnocchi-metricd.log
gnocchi statsd	gnocchi_statsd	/var/log/containers/gnocchi/gnocchi-statsd.log

第 5 章 COLLECTD 插件

您可以根据 Red Hat OpenStack Platform (RHOSP) 17.0 环境配置多个 collectd 插件。

以下插件列表显示了您可以设置用于覆盖默认值的可用 heat 模板 **ExtraConfig** 参数。每个部分提供 **ExtraConfig** 选项的常规配置名称。例如，如果存在名为 **example_plugin** 的 collectd 插件，则插件标题的格式为 **collectd::plugin::example_plugin**。

引用特定插件的可用参数表，如下例所示：

```
ExtraConfig:
  collectd::plugin::example_plugin::<parameter>: <value>
```

为 Prometheus 或 Grafana 查询引用特定插件的指标表。

5.1. COLLECTD::PLUGIN::AGGREGATION

您可以使用 **聚合** 插件将多个值聚合到其中。使用聚合功能，如 **sum**,**average**,**min**, 和 **max** 来计算指标，如平均和总 CPU 统计信息。

表 5.1. 聚合参数

参数	类型
主机	字符串
plugin	字符串
plugininstance	整数
agg_type	字符串
typeinstance	字符串
sethost	字符串
setplugin	字符串
setplugininstance	整数
settypeinstance	字符串
groupBy	字符串数组
computesum	布尔值
calculatenum	布尔值
计算average	布尔值

参数	类型
calculateminimum	布尔值
calculatemaximum	布尔值
calculatestddev	布尔值

配置示例：

部署三个聚合配置以创建以下文件：

1. **aggregator-calcCpuLoadAvg.conf** ：按主机和状态分组的所有 CPU 内核的平均 CPU 负载
2. **aggregator-calcCpuLoadMinMax.conf**: 最小和最大 CPU 负载组，按主机和状态
3. **aggregator-calcMemoryTotalMaxAvg.conf**: 按类型分组的内存的最大值、平均和总计

聚合配置使用默认的 **cpu 和内存** 插件配置。

```
parameter_defaults:
  CollectdExtraPlugins:
    - aggregation

  ExtraConfig:
    collectd::plugin::aggregation::aggregators:
      calcCpuLoadAvg:
        plugin: "cpu"
        agg_type: "cpu"
        groupby:
          - "Host"
          - "TypeInstance"
        calculateaverage: True
      calcCpuLoadMinMax:
        plugin: "cpu"
        agg_type: "cpu"
        groupby:
          - "Host"
          - "TypeInstance"
        calculatemaximum: True
        calculateminimum: True
      calcMemoryTotalMaxAvg:
        plugin: "memory"
        agg_type: "memory"
        groupby:
          - "TypeInstance"
        calculatemaximum: True
        calculateaverage: True
        calculatesum: True
```

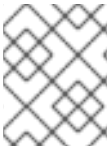
5.2. COLLECTD::PLUGIN::AMQP1

使用 **amqp1** 插件将值写入 amqp1 消息总线，如 AMQ Interconnect。

表 5.2. amqp1 parameters

参数	类型
manage_package	布尔值
传输	字符串
主机	字符串
port	整数
user	字符串
password	字符串
address	字符串
实例	hash
retry_delay	整数
send_queue_limit	整数
interval	整数

使用 **send_queue_limit** 参数来限制传出指标队列的长度。



注意

如果没有 AMQP1 连接，则插件将继续排队消息来发送，这可能会导致未绑定的内存消耗。默认值为 0，它会禁用传出的指标队列。

如果缺少指标，请增加 **send_queue_limit** 参数的值。

配置示例：

```
parameter_defaults:
  CollectdExtraPlugins:
    - amqp1

  ExtraConfig:
    collectd::plugin::amqp1::send_queue_limit: 5000
```

5.3. COLLECTD::PLUGIN::APACHE

使用 **apache** 插件从 Apache Web 服务器提供的 **mod_status** 插件收集 Apache 数据。提供的每个实例都有一个每个间隔 值（以秒为单位）。如果您为实例提供 **timeout** interval 参数，则该值为毫秒。

表 5.3. Apache 参数

参数	类型
实例	hash
interval	整数
manage-package	布尔值
package_install_options	list

表 5.4. apache 实例参数

参数	类型
url	HTTP URL
user	字符串
password	字符串
verifypeer	布尔值
verifyhost	布尔值
cacert	AbsolutePath
sslCiphers	字符串
timeout	整数

配置示例：

在本例中，实例名称是 **localhost**，它连接到位于 http://10.0.0.111/mod_status?auto 的 Apache Web 服务器。您必须将 **?auto** 附加到 URL 的末尾，以防止状态页面返回为与插件不兼容的类型。

```
parameter_defaults:
  CollectdExtraPlugins:
  - apache

  ExtraConfig:
    collectd::plugin::apache::instances:
      localhost:
        url: "http://10.0.0.111/mod_status?auto"
```

其他资源

有关配置 **apache** 插件的更多信息，请参阅 apache。

5.4. COLLECTD::PLUGIN::BATTERY

使用 **电池** 插件报告剩余容量、电源或电池电池。

表 5.5. 电池参数

参数	类型
values_percentage	布尔值
report_degraded	布尔值
query_state_fs	布尔值
interval	整数

其他资源

有关配置 **电池** 插件的更多信息，请参阅 电池。

5.5. COLLECTD::PLUGIN::BIND

使用 **bind** 插件从 DNS 服务器检索查询和响应的编码统计信息，并将这些值提交到 collectd。

表 5.6. 绑定参数

参数	类型
url	HTTP URL
memorystats	布尔值
opcodes	布尔值
parsetime	布尔值
qtypes	布尔值
resolverstats	布尔值
serverstats	布尔值
zonemaintstats	布尔值
视图	数组

参数	类型
interval	整数

表 5.7. 绑定视图参数

参数	类型
name	字符串
qtypes	布尔值
resolverstats	布尔值
cacherrsets	布尔值
zones	字符串列表

配置示例：

```
parameter_defaults:
  CollectdExtraPlugins:
  - bind

  ExtraConfig:
    collectd::plugins::bind:
      url: http://localhost:8053/
      memorystats: true
      opcodes: true
      parsetime: false
      qtypes: true
      resolverstats: true
      serverstats: true
      zonemaintstats: true
      views:
      - name: internal
        qtypes: true
        resolverstats: true
        cacherrsets: true
      - name: external
        qtypes: true
        resolverstats: true
        cacherrsets: true
        zones:
        - "example.com/IN"
```

5.6. COLLECTD::PLUGIN::CEPH

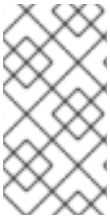
使用 **ceph** 插件从 ceph 守护进程收集数据。

表 5.8. Ceph 参数

参数	类型
守护进程	数组
longrunavglatency	布尔值
convertspecialmetrictypes	布尔值
package_name	字符串

配置示例：

```
parameter_defaults:
  ExtraConfig:
    collectd::plugin::ceph::daemons:
      - ceph-osd.0
      - ceph-osd.1
      - ceph-osd.2
      - ceph-osd.3
      - ceph-osd.4
```



注意

如果 Object Storage Daemon (OSD)没有在每个节点上，您必须列出 OSD。

部署 collectd 时，**ceph** 插件会添加到 Ceph 节点。不要将 Ceph 节点上的 **ceph** 插件添加到 **CollectdExtraPlugins** 中，因为这会导致部署失败。

其他资源

有关配置 **ceph** 插件的更多信息，请参阅 ceph。

5.7. COLLECTD::PLUGINS::CGROUPS

使用 **cgroup** 插件收集 cgroup 中进程的信息。

表 5.9. cgroups 参数

参数	类型
ignore_selected	布尔值
interval	整数
cgroups	list

其他资源

有关配置 **cgroups** 插件的更多信息，请参阅 cgroups。

5.8. COLLECTD::PLUGIN::CONNECTIVITY

使用连接插件来监控网络接口的状态。

注意

如果没有列出接口，则默认监控所有接口。

表 5.10. 连接参数

参数	类型
interfaces	数组

配置示例：

```
parameter_defaults:
  ExtraConfig:
    collectd::plugin::connectivity::interfaces:
    - eth0
    - eth1
```

其他资源

有关配置 **connectivity** 插件的更多信息，请参阅 connectivity。

5.9. COLLECTD::PLUGIN::CONNTRACK

使用 **conntrack** 插件跟踪 Linux 连接跟踪表中的条目数。此插件没有参数。

5.10. COLLECTD::PLUGIN::CONTEXTSWITCH

使用 **ContextSwitch** 插件收集系统处理的上下文切换数。唯一可用的参数是 **interval**，这是以秒为单位定义的轮询间隔。

其他资源

有关配置 **contextswitch** 插件的更多信息，请参阅 contextswitch。

5.11. COLLECTD::PLUGIN::CPU

使用 **cpu** 插件监控 CPU 处于各种状态的时间，例如闲置、执行用户代码、执行系统代码、等待 IO-operations 和其他状态。

cpu 插件收集 *jiffies* 值，而不是百分比值。jiffy 的值取决于您的硬件平台的时钟频率，因此不是绝对的时间间隔单元。

要报告百分比值，请将布尔值参数 **reportbycpu** 和 **reportbystate** 设置为 **true**，然后将布尔值 **参数值百分比** 设置为 true。

此插件默认启用。

表 5.11. CPU 指标

Name	描述	查询
idle	空闲时间量	**collectd_cpu_total{...,type_instance='idle'}**
interrupt	中断的 CPU 阻断	**collectd_cpu_total{...,type_instance='interrupt'}**
法国	运行低优先级进程的时间长度	**collectd_cpu_total{...,type_instance='nice'}**
softirq	为中断请求提供服务的周期量	**collectd_cpu_total{...,type_instance='waitirq'}**
steal	虚拟 CPU 等待实际 CPU 的时间百分比，而虚拟机监控程序为另一个虚拟处理器提供服务	**collectd_cpu_total{...,type_instance='steal'}**
system	系统级别花费的时间（内核）	**collectd_cpu_total{...,type_instance='system'}**
user	用户进程使用的地方	**collectd_cpu_total{...,type_instance='user'}**
wait	等待未完成的 I/O 请求的 CPU	**collectd_cpu_total{...,type_instance='wait'}**

表 5.12. CPU 参数

参数	类型	默认值
reportbystate	布尔值	true
值百分比	布尔值	true
reportbycpu	布尔值	true
reportnumcpu	布尔值	false
reportgueststate	布尔值	false
subtractgueststate	布尔值	true
interval	整数	120

配置示例：

```
parameter_defaults:
```

```
CollectdExtraPlugins:
  - cpu
ExtraConfig:
    collectd::plugin::cpu::reportbystate: true
```

其他资源

有关配置 **cpu** 插件的更多信息，请参阅 cpu。

5.12. COLLECTD::PLUGIN::CPUFREQ

使用 **cpufreq** 插件收集当前的 CPU 频率。此插件没有参数。

5.13. COLLECTD::PLUGIN::CSV

使用 **csv** 插件以 CSV 格式将值写入本地文件。

表 5.13. CSV 参数

参数	类型
dataDir	字符串
storerates	布尔值
interval	整数

5.14. COLLECTD::PLUGIN::DF

使用 **df** 插件为文件系统收集磁盘空间使用情况信息。

此插件默认启用。

表 5.14. df 指标

Name	描述	查询
free	可用磁盘空间量	**collectd_df_df_complex{...,type_instance="free"}**
reserved	保留磁盘空间量	**collectd_df_df_complex{...,type_instance="reserved"}**
使用的	使用的磁盘空间量	**collectd_df_df_complex{...,type_instance="used"}**

表 5.15. df 参数

参数	类型	默认值
devices	数组	**[]**
fstypes	数组	**['xfs']**
忽略选择	布尔值	true
mountpoints	数组	**[]**
reportbydevice	布尔值	true
reportinodes	布尔值	true
reportreserved	布尔值	true
valuesabsolute	布尔值	true
值百分比	布尔值	false

配置示例：

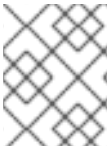
```
parameter_defaults:
  ExtraConfig:
    collectd::plugin::df::fstypes: ['tmpfs','xfs']
```

其他资源

有关配置 **df** 插件的更多信息，请参阅 df。

5.15. COLLECTD::PLUGIN::DISK

使用 **磁盘插件** 来收集硬盘的性能统计信息，并在支持的情况下收集分区。



注意

默认情况下，**disk** 插件会监控所有磁盘。您可以使用 **ignoreselected** 参数来忽略磁盘列表。示例配置会忽略 *sda*、*sdb* 和 *sdc* 磁盘，并监控列表中未包含的所有磁盘。

此插件默认启用。

表 5.16. 磁盘参数

参数	类型	默认值
disks	数组	**[]**
忽略选择	布尔值	false

参数	类型	默认值
udevnameattr	字符串	*<undefined>*

表 5.17. 磁盘指标

Name	描述
合并	可合并的排队操作数量，例如，一个物理磁盘访问提供了两个或者多个逻辑操作。
time	完成 I/O 合作的平均时间。值可能并不准确。
io_time	I/O (ms)花费的时间。您可以使用这个指标作为设备负载百分比。值 1 秒匹配 100% 的负载。
weighted_io_time	测量 I/O 完成时间和可能累积的积压。
pending_operations	显示待处理 I/O 操作的队列大小。

配置示例：

```
parameter_defaults:
  ExtraConfig:
    collectd::plugin::disk::disks: ['sda', 'sdb', 'sdc']
    collectd::plugin::disk::ignoreselected: true
```

其他资源

有关配置 **disk** 插件的更多信息，请参阅 disk。

5.16. COLLECTD::PLUGIN::HUGEPAGES

使用 hugepages 插件收集巨页信息。

> This plugin is enabled by default.

表 5.18. hugepages 参数

参数	类型	默认值
report_per_node_hp	布尔值	true
report_root_hp	布尔值	true
values_pages	布尔值	true

参数	类型	默认值
values_bytes	布尔值	false
values_percentage	布尔值	false

配置示例：

```
parameter_defaults:
  ExtraConfig:
    collectd::plugin::hugepages::values_percentage: true
```

其他资源

- 有关配置 **巨页** 插件的更多信息，请参阅 巨页。

5.17. COLLECTD::PLUGIN::INTERFACE

使用 **接口** 插件来测量 octets 中的接口流量、每秒数据包数和每秒错误率。

> This plugin is enabled by default.

表 5.19. 接口参数

参数	类型	默认
interfaces	数组	[]
忽略选择	布尔值	false
reportinactive	布尔值	true

配置示例：

```
parameter_defaults:
  ExtraConfig:
    collectd::plugin::interface::interfaces:
      - lo
    collectd::plugin::interface::ignoreselected: true
```

其他资源

- 有关配置 **接口** 插件的更多信息，请参阅 接口。

5.18. COLLECTD::PLUGIN::LOAD

使用 负载 插件来收集系统负载和系统使用概述。

> This plugin is enabled by default.

表 5.20. 插件参数

参数	类型	默认
report_relative	布尔值	true

配置示例：

```
parameter_defaults:
  ExtraConfig:
    collectd::plugin::load::report_relative: false
```

其他资源

- 有关配置 负载 插件的更多信息，请参阅 加载。

5.19. COLLECTD::PLUGIN::MCELOG

使用 **mcelog** 插件发送与机器检查异常相关的通知和统计信息。将 **mcelog** 配置为以守护进程模式运行并启用日志记录功能。

表 5.21. mcelog 参数

参数	类型
Mcelogfile	字符串
内存	hash **{ mcelogclientsocket[string], persistentnotification[boolean] }**

配置示例：

```
parameter_defaults:
    CollectdExtraPlugins: mcelog
    CollectdEnableMcelog: true
```

其他资源

- 有关配置 **mcelog** 插件的更多信息，请参阅 mcelog。

5.20. COLLECTD::PLUGIN::MEMCACHED

使用 **memcached** 插件检索有关 memcached 缓存使用、内存和其他相关信息的信息。

表 5.22. Memcached 参数

参数	类型
实例	hash
interval	整数

配置示例：

```
parameter_defaults:
  CollectdExtraPlugins:
  - memcached

  ExtraConfig:
    collectd::plugin::memcached::instances:
      local:
        host: "%{hiera('fqdn_canonical')}"
        port: 11211
```

其他资源

- 有关配置 **memcached** 插件的更多信息，请参阅 memcached。

5.21. COLLECTD::PLUGIN::MEMORY

使用 **内存** 插件检索系统内存信息。

> This plugin is enabled by default.

表 5.23. 内存参数

参数	类型
默认值	valuesabsolute
布尔值	true
值百分比	布尔值

配置示例：

```
parameter_defaults:
  ExtraConfig:
    collectd::plugin::memory::valuesabsolute: true
    collectd::plugin::memory::valuespercentage: false
```

其他资源

- 有关配置 **内存** 插件的更多信息，请参阅 内存。

5.22. COLLECTD::PLUGIN::NTPD

使用 **ntpd** 插件查询配置为允许访问统计信息的本地 NTP 服务器，并检索有关配置的参数和时间同步状态的信息。

表 5.24. ntpd 参数

参数	类型
主机	Hostname
port	端口号（整数）
reverselookups	布尔值
includeunitid	布尔值
interval	整数

配置示例：

```
parameter_defaults:
  CollectdExtraPlugins:
  - ntpd

  ExtraConfig:
    collectd::plugin::ntpd::host: localhost
    collectd::plugin::ntpd::port: 123
    collectd::plugin::ntpd::reverselookups: false
    collectd::plugin::ntpd::includeunitid: false
```

其他资源

- 有关配置 **ntpd** 插件的更多信息，请参阅 ntpd。

5.23. COLLECTD::PLUGIN::OVS_STATS

使用 **ovs_stats** 插件来收集 OVS 连接接口的统计信息。 **ovs_stats** 插件使用 OVSDB 管理协议 (RFC7047)监控机制从 OVSDB 获取统计信息。

表 5.25. ovs_stats parameters

参数	类型
address	字符串
网桥	list
port	整数

参数	类型
socket	字符串

配置示例：

以下示例演示了如何启用 **ovs_stats** 插件。如果您使用 OVS 部署 overcloud，则不需要启用 **ovs_stats** 插件。

```
parameter_defaults:
  CollectdExtraPlugins:
    - ovs_stats
  ExtraConfig:
    collectd::plugin::ovs_stats::socket: '/run/openvswitch/db.sock'
```

其他资源

- 有关配置 **ovs_stats** 插件的更多信息，请参阅 ovs_stats。

5.24. COLLECTD::PLUGIN::PROCESSES

process 插件提供有关系统进程的信息。如果没有指定自定义进程匹配，则该插件只收集按状态和进程分叉率的进程数量。

要收集有关特定进程的更多详细信息，您可以使用 **process** 参数指定进程名称或 **process_match** 选项来指定与正则表达式匹配的进程名称。**process_match** 输出的统计信息按进程名称分组。

表 5.26. 插件参数

参数	类型	默认值
Process	数组	*<undefined>*
process_matches	数组	*<undefined>*
collect_context_switch	布尔值	*<undefined>*
collect_file_descriptor	布尔值	*<undefined>*
collect_memory_maps	布尔值	*<undefined>*

其他资源

- 有关配置 **processes** 插件的更多信息，请参阅 processes。

5.25. COLLECTD::PLUGIN::SMART

使用 **smart** 插件从节点上的物理磁盘收集 SMART （自助监控、分析和报告技术）。您还必须将参数 **CollectdContainerAdditionalCapAdd** 设置为 **CAP_SYS_RAWIO**，以允许 **智能** 插件读取 SMART 遥测。如果您没有设置 **CollectdContainerAdditionalCapAdd** 参数，则会将以下信息写入 collectd 错误日

志中：

Smart 插件：以 root 用户身份运行 collectd，但缺少 CAP_SYS_RAWIO 功能。插件的读取功能可能会失败。您的 init 系统是否丢弃功能？

表 5.27. smart 参数

参数	类型
disks	数组
忽略选择	布尔值
interval	整数

配置示例：

```
parameter_defaults:
  CollectdExtraPlugins:
  - smart

  CollectdContainerAdditionalCapAdd: "CAP_SYS_RAWIO"
```

附加信息

- 有关配置 **智能** 插件的更多信息，请参阅 smart。

5.26. COLLECTD::PLUGIN::SWAP

使用 **swap** 插件收集有关可用和使用 swap 空间的信息。

表 5.28. swap 参数

参数	类型
reportbydevice	布尔值
reportbytes	布尔值
valuesabsolute	布尔值
值百分比	布尔值
reportio	布尔值

配置示例：

```
parameter_defaults:
  CollectdExtraPlugins:
```

```
  - swap

  ExtraConfig:
    collectd::plugin::swap::reportbydevice: false
    collectd::plugin::swap::reportbytes: true
    collectd::plugin::swap::valuesabsolute: true
    collectd::plugin::swap::valuespercentage: false
    collectd::plugin::swap::reportio: true
```

5.27. COLLECTD::PLUGIN::TCPCONNS

使用 **tcpconns** 插件收集有关配置的端口的 TCP 连接入站和出站数量的信息。本地端口配置代表入口连接。远程端口配置代表出口连接。

表 5.29. tcpconns 参数

参数	类型
localports	端口(Array)
远程端口	端口(Array)
侦听	布尔值
allportssummary	布尔值

配置示例：

```
parameter_defaults:
  CollectdExtraPlugins:
  - tcpconns

  ExtraConfig:
    collectd::plugin::tcpconns::listening: false
    collectd::plugin::tcpconns::localports:
    - 22
    collectd::plugin::tcpconns::remoteports:
    - 22
```

5.28. COLLECTD::PLUGIN::THERMAL

使用rmal **插件** 检索 ACPI 的rmal 区信息。

表 5.30. rmal 参数

参数	类型
devices	数组
忽略选择	布尔值

参数	类型
interval	整数

配置示例：

```
parameter_defaults:
  CollectdExtraPlugins:
  - thermal
```

5.29. COLLECTD::PLUGIN::UPTIME

使用 **正常运行时间** 插件收集有关系统运行时间的信息。

This plugin is enabled by default.

表 5.31. 运行时间参数

参数	类型
interval	整数

5.30. COLLECTD::PLUGIN::VIRT

使用 **virt** 插件，通过 **libvirt** API 为主机上的虚拟机收集 CPU、磁盘、网络负载和其他指标。

默认在计算主机上启用此插件。

表 5.32. virt 参数

参数	类型
连接	字符串
refresh_interval	hash
domain	字符串
block_device	字符串
interface_device	字符串
ignore_selected	布尔值
plugin_instance_format	字符串
hostname_format	字符串

参数	类型
interface_format	字符串
extra_stats	字符串

配置示例：

```
ExtraConfig:
  collectd::plugin::virt::hostname_format: "name uuid hostname"
  collectd::plugin::virt::plugin_instance_format: metadata
```

其他资源

有关配置 **virt** 插件的更多信息，请参阅 virt。

5.31. COLLECTD::PLUGIN::VMEM

使用 **vmem** 插件从内核子系统收集有关虚拟内存的信息。

表 5.33. vmem 参数

参数	类型
详细	布尔值
interval	整数

配置示例：

```
parameter_defaults:
  CollectdExtraPlugins:
  - vmem

  ExtraConfig:
    collectd::plugin::vmem::verbose: true
```

5.32. COLLECTD::PLUGIN::WRITE_HTTP

使用 **write_http** 输出插件，通过使用带有 JSON 的 POST 请求和编码指标，或使用 **PUTVAL** 命令将值提交到 HTTP 服务器。

表 5.34. write_http parameters

参数	类型
确保	Enum[*present*,*absent*]

参数	类型
节点	hash[String, Hash[String, Scalar]]
urls	hash[String, Hash[String, Scalar]]
manage_package	布尔值

配置示例：

```
parameter_defaults:
  CollectdExtraPlugins:
    - write_http
  ExtraConfig:
    collectd::plugin::write_http::nodes:
      collectd:
        url: “http://collectd.tld.org/collectd”
        metrics: true
        header: “X-Custom-Header: custom_value"
```

其他资源

- 有关配置 **write_http** 插件的更多信息，请参阅 write_http。

5.33. COLLECTD::PLUGIN::WRITE_KAFKA

使用 **write_kafka** 插件将值发送到 Kafka 主题。使用一个或多个主题块配置 **write_kafka** 插件。对于每个主题块，您必须指定唯一的名称和一个 Kafka producer。您可以在主题块中使用以下每个主题参数：

表 5.35. write_kafka parameters

参数	类型
kafka_hosts	Array[String]
主题	hash
属性	hash
meta	hash

配置示例：

```
parameter_defaults:
  CollectdExtraPlugins:
    - write_kafka
  ExtraConfig:
    collectd::plugin::write_kafka::kafka_hosts:
      - remote.tld:9092
```

```
collectd::plugin::write_kafka::topics:
  mytopic:
    format: JSON
```

其他资源：

有关如何配置 **write_kafka** 插件的更多信息，请参阅 write_kafka。