Red Hat OpenStack Platform 17.1

管理 overcloud 可观察性

跟踪物理和虚拟资源，以及收集指标

Last Updated: 2024-05-23

Red Hat OpenStack Platform 17.1 管理 overcloud 可观察性

跟踪物理和虚拟资源，以及收集指标

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。

在 JIRA 中提供文档反馈

使用 Create Issue 表单对文档提供反馈。JIRA 问题将在 Red Hat OpenStack Platform Jira 项目中创建，您可以在其中跟踪您的反馈进度。

1. 确保您已登录到 JIRA。如果您没有 JIRA 帐户，请创建一个帐户来提交反馈。
2. 点击以下链接打开 **Create Issue** 页面： Create Issue
3. 完成 **Summary** 和 **Description** 字段。在 **Description** 字段中，包含文档 URL、章节或章节号以及问题的详细描述。不要修改表单中的任何其他字段。
4. 点 **Create**。

第 1 章 操作测量简介

您可以使用 ceilometer、collectd 和日志记录服务等可观察性组件从 Red Hat OpenStack Platform (RHOSP)环境中收集数据。您可以将 Gnocchi 中收集的数据存储用于自动扩展用例，也可以使用 **metrics_qdr** 将数据转发到 Service Telemetry Framework (STF)。

如需有关自动扩展的更多信息，请参阅实例自动扩展

有关 STF 的更多信息，请参阅 Service Telemetry Framework 1.5

1.1. OBSERVABILITY 架构

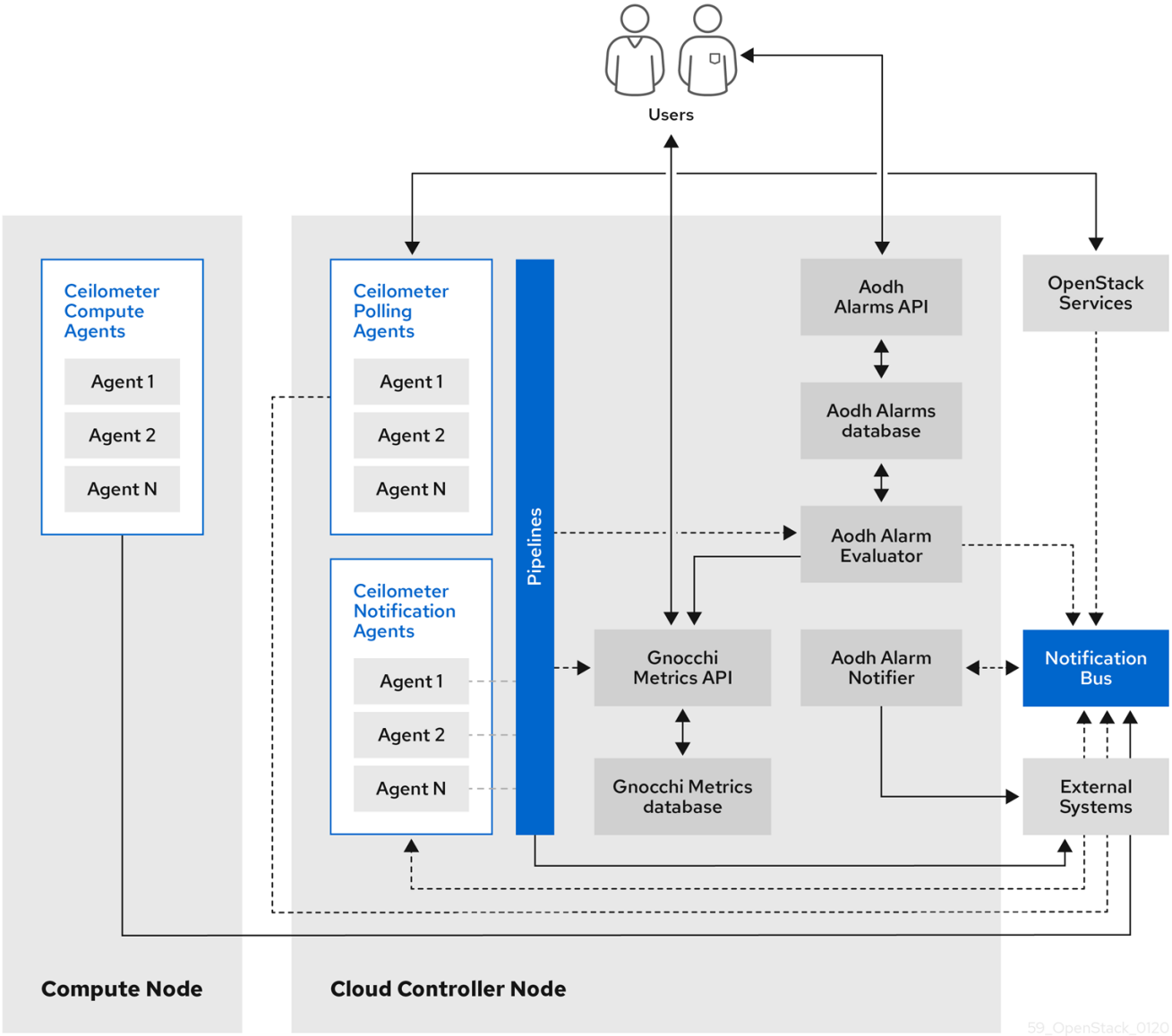
Red Hat OpenStack Platform (RHOSP) Observability 为基于 OpenStack 的云提供用户级使用情况数据。您可以配置可观察性组件，从现有 RHOSP 组件发送的通知收集数据，如计算使用事件，或轮询 RHOSP 基础架构资源，如 libvirt。Ceilometer 将收集的数据发布到各种目标，如数据存储和消息队列，包括服务遥测框架(STF)。

Observability 由以下组件组成：

- **数据收集** ： Observability 使用 Ceilometer 收集指标和事件数据。如需更多信息，请参阅 第 1.2.1 节 “ilo”。
- **Storage**: Observability 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 存储”。
- **警报服务** ： Observability 使用 Alarming 服务(Aodh)根据定义的规则针对 Ceilometer 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. Observability 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	删除自	备注
aodh	RHOSP 9	RHOSP 15		支持自动扩展用例。
ilo	RHOSP 4			支持自动扩展和服务遥测框架(STF)用例中 RHOSP 的指标和事件集合。
collectd	RHOSP 11	RHOSP 17.1		支持 STF 的基础架构指标集合。

组件	完全支持自	弃用	删除自	备注
Gnocchi	RHOSP 9	RHOSP 15		支持存储自动扩展用例的指标。
panko	RHOSP 11	RHOSP 12，不默认安装，因为 RHOSP 14	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：

- 用于 RHOSP 组件级监控的 Ceilometer。如需更多信息，请参阅 第 1.2.1 节 “ilo”。
- 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. ilo

Ceilometer 是 Red Hat OpenStack Platform (RHOSP)的默认数据收集组件，它可以在所有当前 RHOSP 核心组件中对数据进行规范化和转换。Ceilometer 收集与 RHOSP 服务相关的计量和事件数据。

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** running with parameter **--polling namespace-compute** 相同。
- **中央代理(ceilometer-agent-central)**：在中央管理服务器上运行，轮询未绑定到实例或 Compute 节点的资源利用率统计。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，该代理使用参数 **--polling namespace-central** 进行操作。
- **通知代理(ceilometer-agent-notification)**：在中央管理服务器上运行，并且使用来自消息队列的消息来构建事件和计量数据。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。

Ceilometer 代理使用发布者将数据发送到对应的端点，如 Gnocchi 或 AMQP 版本 1 (QDR)。

1.2.2. collectd

collectd 是另一个数据收集代理，可用于提供基础架构指标。它从配置的源中重复拉取数据。您可以将指标转发到 Service Telemetry Framework (STF)，以存储和视觉化数据。

1.3. 使用 GNOCCHI 存储

Gnocchi 是一个开源时间序列数据库。您可以使用 gnocchi 来存储并提供对 operator 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。

在 Red Hat OpenStack Platform (RHOSP)中使用 Gnocchi 支持自动扩展用例。如需有关自动扩展的更多信息，请参阅实例自动扩展

1.3.1. 归档策略：在时间序列数据库中同时存储短期和长期数据

归档策略定义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，如最小值、最大值、平均百分比和标准偏差。这些聚合在名为 granularity 的一段时间内计算，并保留在特定时间span 中。

归档策略定义了指标如何聚合以及如何存储它们。每个归档策略被定义为时间span 上的点数。

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 1 秒的聚合。这意味着，时间序列是最大的，在最新点和旧点之间保留 10 秒的数据。

归档策略还定义使用哪种聚合方法。默认值为参数 **default_aggregation_methods**，其值默认设置为 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. indexer 驱动程序

索引器负责存储所有资源的索引、归档策略和指标及其定义、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。

1.3.3. Gnocchi 术语

此表包含 Gnocchi 功能常用术语的定义。

表 1.2. Gnocchi 术语

术语	定义
聚合方法	此函数用于将多个测量结果聚合到一个聚合中。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。归档策略决定了指标中聚合的时长以及聚合如何聚合（聚合方法）。
granularity	指标聚合系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。测量结果由时间戳和值组成。
指标	存储由 UUID 标识的聚合的实体。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。

术语	定义
资源	代表您与指标关联的基础架构中任何实体。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。

第 2 章 准备操作测量

您可以使用 Ceilometer 或 collectd 收集自动扩展或 Service Telemetry Framework (STF)的遥测数据。

2.1. COLLECTD 测量

以下是默认的 collectd 测量：

- cpu
- 磁盘可用
- 磁盘用量
- hugepages
- interface
- load
- 内存
- unixsock
- uptime

2.2. 数据存储规划

Gnocchi 存储数据点集合，其中每个数据点都是聚合。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来估算大小。

警告

将 Red Hat OpenStack Platform (RHOSP) Object Storage (swift)用于时间序列数据库(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，如果要使用一分钟分辨率保留一年的数据，请使用公式：

   数据点数 = (365 天 X 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：
   size in bytes = data points X 8 字节

   如果您将这个公式应用到示例，则结果为 4.1 MB：

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

这个值是单个聚合的时间序列数据库的存储要求。如果您的归档策略使用多个聚合方法(min, max, mean, sum, std, count)，请将这个值乘以您使用的聚合方法数量。

其他资源

- 第 1.3.1 节 "归档策略：在时间序列数据库中同时存储短期和长期数据"
- 第 2.3 节 "规划和管理归档策略"

2.3. 规划和管理归档策略

您可以使用归档策略来配置指标的聚合方式，以及将指标存储在时间序列数据库中的时间。归档策略定义为时间span 上的点数。

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。这意味着时间序列会在最新点和旧点之间保留最多 10 秒的数据。归档策略还定义要使用的聚合方法。默认设置为参数 **default_aggregation_methods**，其中默认值被设置为 **mean**, **min**, **max**。**sum**、**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，请确保您熟悉以下概念：

- 指标.更多信息请参阅 第 2.3.1 节 "指标"。
- 测量.如需更多信息，请参阅 第 2.3.2 节 "创建自定义测量结果"。

2.3.1. 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、房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。

2.3.2. 创建自定义测量结果

测量结果是 API 发送到 Gnocchi 的传入元组。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

- 创建自定义测量结果：

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

2.3.3. 验证指标状态

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

如果没有错误消息，则代表您的部署成功。

2.3.4. 创建归档策略

您可以创建一个归档策略来定义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。

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。

2.3.5. 查看归档策略

使用以下步骤检查您的归档策略。

流程

1. 列出归档策略。

```
$ openstack metric archive policy list
```

2. 查看归档策略的详情：

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

2.3.6. 删除归档策略

如果要删除归档策略，请使用以下步骤。

流程

- 删除归档策略。将 <archive-policy-name> 替换为您要删除的策略名称。

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

验证

- 检查您删除的归档策略是否没有归档策略。

```
$ openstack metric archive policy list
```

2.3.7. 创建归档策略规则

您可以使用归档策略规则来配置指标和归档策略之间的映射。

流程

- 创建归档策略规则。将 <rule-name> 替换为规则的名称，将 <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 安装和配置日志服务

您可以使用日志消息进行故障排除和监控系统事件。日志代理 Rsyslog 收集客户端的日志，并将这些日志记录发送到与支持的 Red Hat OpenStack Platform (RHOSP)环境独立的远程 Elasticsearch 存储系统，例如。

3.1. 日志系统架构和组件

监控工具使用客户端-服务器模型，客户端部署到 Red Hat OpenStack Platform (RHOSP) overcloud 节点上。Rsyslog 服务提供客户端日志。

RHOSP 中的日志示例包括：

- 操作系统日志，如 syslog 和 audit 日志文件。
- 来自基础架构组件的日志，如 RabbitMQ 和 MariaDB。
- 来自 RHOSP 服务的日志，如 Identity (keystone)和计算(nova)。

日志文件记录操作、错误、警告和其他事件。在分布式环境中，在一个位置收集各种日志可帮助您进行调试和管理。

注意

RHOSP director 不部署用于日志的服务器端组件。

3.2. 启用 ELASTICSEARCH 的日志记录

Elasticsearch 是一个服务器端数据库，可用于存储日志。要为 Elasticsearch 启用日志服务，您必须为 Elasticsearch 验证日志服务。

注意

Rsyslog 服务只使用 Elasticsearch 作为数据存储进行日志记录。

先决条件

- 您已部署了 Elasticsearch。
- 您有服务器的用户名、密码和 URL。

流程

1. 在自定义模板目录中创建一个文件，如 **$HOME/custom_templates/logging-connector.yaml**，您可以编辑来为您的环境配置 **RsyslogElasticsearchSetting** 参数，如下例所示：

```
parameter_defaults:
  RsyslogElasticsearchSetting:
    uid: "elastic"
    pwd: "yourownpassword"
    skipverifyhost: "on"
    allowunsignedcerts: "on"
    server: "https://openstack-log-storage.elasticsearch.tld"
    serverport: 443
```

2. 将 **logging-environment-rsyslog.yaml** 和 **logging-connector.yaml** 环境文件的文件路径添加到 **overcloud 部署** 命令中：

```
$ openstack overcloud deploy \
<overcloud_environment_files> \
-e <filepath>/logging-environment-rsyslog.yaml
-e $HOME/custom_templates/logging-connector.yaml
```

- 将 **<overcloud_environment_files** > 替换为现有部署中的环境文件列表。
- 将 **<filepath** > 替换为 **logging-environment-rsyslog.yaml** 文件的文件路径，如 **/usr/share/openstack-tripleo-heat-templates/environments/**。

3.3. 可配置日志记录参数

此表包含用来在 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 **/usr/share/openstack-tripleo-heat-templates/deployment/logging/rsyslog-container-puppet.yaml** 文件中找到这些参数。

表 3.1. 可配置日志记录参数

参数	Description
RsyslogElasticsearchSetting	配置 **rsyslog-elasticsearch** 插件。
RsyslogElasticsearchTlsCACert	包含签发 Elasticsearch 服务器证书的 CA 证书的内容。
RsyslogElasticsearchTlsClientCert	包含用于对 Elasticsearch 进行客户端证书授权的客户端证书内容。
RsyslogElasticsearchTlsClientKey	包含与证书 **RsyslogElasticsearchTlsClientCert** 对应的私钥的内容。
RsyslogReopenOnTruncate	当磁盘上的文件大小小于内存中的当前偏移时，提示 rsyslog 重新打开输入文件。
RsyslogMaxMessageSize	为日志消息的大小设置限制。

3.4. 覆盖日志文件的默认路径

如果修改默认容器使其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

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
    file: <filepath>/nova-compute.log
```

- 将 **<filepath** > 替换为 **nova-compute.log** 文件的文件路径
- 确保为该服务定义 **tag** 和 **file** 参数的值。您可以将默认值用于其他参数。

2. 您可以修改特定服务的格式。格式传递给 Rsyslog 配置。以下示例显示了基本语法：

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

以下示例显示了更复杂的转换：

```
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>.*)?$/'
```

3. 如果启用了集中式日志记录，您可以在自定义模板中使用以下定义来转发额外的日志文件，例如 /var/log/messages ：

```
parameter_defaults:
  ExtraConfig:
    tripleo_logging_sources_messages:
      - tag: openstack.host.messages
        file: /var/log/host/messages
        startmsg.regex: "^[a-zA-Z]{3} [ 1-9][0-9] [:0-9]{8}"
```

3.5. 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的格式。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]+)?(DEBUG|INFO|WARNING|ERROR)'）。

流程

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

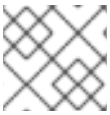
```
NovaComputeLoggingSource:
    tag: openstack.nova.compute
    file: /some/other/path/nova-compute.log
    startmsg.regex: `^[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}:[0-9]{2}:[0-9]{2}(.[0-9]+ \\+[0-9]+)? [A-
Z]+ \\([a-z]+\\)`
```

3.6. 验证 RSYSLOG 和 ELASTICSEARCH 之间的连接

在客户端中，您可以测试并验证 Rsyslog 和 Elasticsearch 之间的通信。

流程

- 进入到 Elasticsearch 连接日志文件，它是 Rsyslog 容器中的 **/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，或者日志文件存在但不包含日志，则不会出现连接问题。如果日志文件存在并包含日志，Rsyslog 尚未成功连接。



注意

要从服务器端测试连接，请查看 Elasticsearch 日志中的连接问题。

3.7. 回溯

如果要对问题进行故障排除，您可以使用回溯日志诊断问题。在日志文件中，回溯通常具有多个信息行，它们都与同一问题有关。

rsyslog 提供了一个正则表达式来定义日志记录的启动方式。每个日志记录通常以时间戳开头，并且回溯的第一行是包含此信息的唯一行。rsyslog 使用第一行来绑定缩进记录，并将它们作为一条日志记录发送。

对于这种行为配置选项，使用了 <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**。

3.8. RED HAT OPENSTACK PLATFORM 服务的日志文件位置

每个 Red Hat OpenStack Platform (RHOSP)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。

3.8.1. 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

3.8.2. 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
信息性消息	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

3.8.3. Compute (nova)日志文件

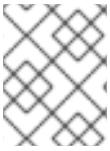
Service	服务名称	日志路径
OpenStack Compute API 服务	openstack-nova-api.service	/var/log/containers/nova/nova-api.log
OpenStack Compute 证书服务器	openstack-nova-cert.service	/var/log/containers/nova/nova-cert.log
OpenStack Compute 服务	openstack-nova-compute.service	/var/log/containers/nova/nova-compute.log
OpenStack 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
信息性消息	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 Scheduler 服务	openstack-nova-scheduler.service	/var/log/containers/nova/nova-scheduler.log

3.8.4. dashboard (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 服务器将额外的日志文件用于 Dashboard Web 界面，您可以使用 Web 浏览器或命令行客户端（如 keystone 和 nova ）访问该界面。您可以使用以下日志文件跟踪仪表板使用情况并诊断错误：

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

还有一个 **/var/log/containers/httpd/default_error.log**，它存储了在同一主机上运行的其他 Web 服务报告的错误。

3.8.5. 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

3.8.6. 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

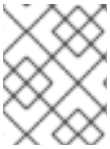
Service	服务名称	日志路径
OpenStack Image Service Registry 服务器	openstack-glance-registry.service	/var/log/containers/glance/registry.log

3.8.7. networking (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

3.8.8. 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 Object Storage 日志文件都进入 **/var/log/containers/swift/swift.log**，使用 local0、local1 和 local2 syslog 工具。

Object Storage 日志消息可以从 REST API 服务或后台守护进程中。

- API 服务消息为每个 API 请求包含一个行。前端和后端服务 post 消息。
- 守护进程消息包含有关守护进程任务的人类可读信息。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
```

以下是守护进程消息的示例：

```
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
```

3.8.9. 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
编配服务事件	不适用	/var/log/containers/heat/heat-manage.log

3.8.10. 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

您还可以在 **/var/log/containers/manila/manila-manage.log** 中的 Manila Python 库日志信息。

3.8.11. 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 中央代理	ceilometer_agent_central	/var/log/containers/ceilometer/central.log
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

3.8.12. 支持服务的日志文件

以下服务供 RHOSP 核心组件使用，并有自己的日志目录和文件。

服务	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

3.8.13. aodh (alarming 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 evaluator 日志	aodh_evaluator	/var/log/containers/aodh/aodh-evaluator.log
警报侦听器	aodh_listener	/var/log/containers/aodh/aodh-listener.log
警报通知	aodh_notifier	/var/log/containers/aodh/aodh-notifier.log

3.8.14. 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
Gnocchi 指标	gnocchi_metricd	/var/log/containers/gnocchi/gnocchi-metricd.log
Gnocchi statsd	gnocchi_statsd	/var/log/containers/gnocchi/gnocchi-statsd.log

第 4 章 COLLECTD 插件

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

以下插件列表显示可用的 heat 模板 **ExtraConfig** 参数，您可以将其设置为覆盖默认值。每个部分都提供 **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**。

请参考特定插件的可用参数表，如下例所示：

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

引用 Prometheus 或 Grafana 查询的特定插件的指标表。

4.1. COLLECTD::PLUGIN::AGGREGATION

您可以使用聚合插件将多个值 **聚合** 为一个值。使用 **sum**、**average**、**min** 和 **max** 等聚合功能来计算指标，如平均和总 CPU 统计。

表 4.1. 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
calculatesum	布尔值
calculatenum	布尔值
calculateaverage	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**: 按主机和状态划分的最小和最大 CPU 负载组
3. **aggregator-calcMemoryTotalMaxAvg.conf**: maximum, average, 和 total 用于按类型分组的内存

聚合配置使用默认的 **cpu** 和 **memory** 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
```

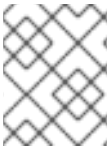
4.2. COLLECTD::PLUGIN::AMQP1

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

表 4.2. 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，插件将继续排队要发送的消息，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
```

4.3. COLLECTD::PLUGIN::APACHE

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

表 4.3. Apache 参数

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

表 4.4. 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，以防止返回状态页面作为与插件不兼容的类型返回。

```
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。

4.4. COLLECTD::PLUGIN::BATTERY

使用 **电池** 插件报告剩余的容量、电源或笔记本电脑电池。

表 4.5. 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。

4.5. COLLECTD::PLUGIN::BIND

使用 **bind** 插件检索有关 DNS 服务器的查询和响应的编码统计信息，并将这些值提交到 collectd。

表 4.6. bind 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 4.7. 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"
```

4.6. COLLECTD::PLUGIN::CEPH

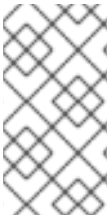
使用 **ceph** 插件从 ceph 守护进程收集数据。

表 4.8. 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)不在每个节点中，您必须列出 OSD。

部署 collectd 时，**ceph** 插件将添加到 Ceph 节点。不要在 Ceph 节点上将 **ceph** 插件添加到 **CollectdExtraPlugins**，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。

4.7. COLLECTD::PLUGINS::CGROUPS

使用 **cgroups** 插件收集 cgroup 中进程的信息。

表 4.9. 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。

4.8. COLLECTD::PLUGIN::CONNECTIVITY

使用 connectivity 插件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 4.10. 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。

4.9. 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。

4.10. COLLECTD::PLUGIN::CONTEXTSWITCH

使用 **ContextSwitch** 插件收集系统处理的上下文切换数量。唯一可用的参数是 **interval**，它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。

4.11. COLLECTD::PLUGIN::CPU

使用 **cpu** 插件监控 CPU 在各种状态下花费的时间，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，因此不是绝对时间间隔单位。

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

此插件默认为启用。

表 4.11. CPU 指标

Name	描述	查询
idle	闲置时间量	**collectd_cpu_total{...,type_instance='idle'}**
interrupt	中断阻止的 CPU	**collectd_cpu_total{...,type_instance='interrupt'}**
nice	运行低优先级进程的时间	**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	在系统级别（内核）上花费的时间长度（内核）	**collectd_cpu_total{...,type_instance='system'}**
user	用户进程使用的 MAPPING	**collectd_cpu_total{...,type_instance='user'}**
wait	CPU 等待未完成的 I/O 请求	**collectd_cpu_total{...,type_instance='wait'}**

表 4.12. CPU 参数

参数	类型	默认值
reportbystate	布尔值	true
valuespercentage	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。

4.12. COLLECTD::PLUGIN::CPUFREQ

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

4.13. COLLECTD::PLUGIN::CSV

使用 **csv** 插件将值写入 CSV 格式的本地文件。

表 4.13. CSV 参数

参数	类型
datadir	字符串
storerates	布尔值
interval	整数

4.14. COLLECTD::PLUGIN::DF

使用 **df** 插件收集文件系统的磁盘空间使用信息。

此插件默认为启用。

表 4.14. df 指标

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

表 4.15. df 参数

参数	类型	默认值
devices	数组	**[]**
fstypes	数组	**['xfs']**
ignoreselected	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
valuespercentage	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。

4.15. COLLECTD::PLUGIN::DISK

使用 **disk** 插件收集硬盘的性能统计信息，如果受支持，则可使用分区。

注意

disk 插件默认监控所有磁盘。您可以使用 **ignoreselected** 参数忽略磁盘列表。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。

此插件默认为启用。

表 4.16. 磁盘参数

参数	类型	默认值
disks	数组	**[]**
ignoreselected	布尔值	false

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

表 4.17. 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多个逻辑操作。
time	完成 I/O 工作所需的平均时间。值可能不准确。
io_time	进行 I/O (ms)花费的时间。您可以使用此指标作为设备负载百分比。值 1 秒与 100% 的负载匹配。
weighted_io_time	测量 I/O 完成时间和可能会积累的积压。
pending_operations	显示待处理的 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。

4.16. COLLECTD::PLUGIN::HUGEPAGES

使用 hugepages 插件收集大页信息。

> This plugin is enabled by default.

表 4.18. 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 **hugepages** 插件的更多信息，请参阅 巨页。

4.17. COLLECTD::PLUGIN::INTERFACE

使用 **interface** 插件测量 octets 中的接口流量、每秒数据包和错误率。

This plugin is enabled by default.

表 4.19. 接口参数

参数	类型	默认
interfaces	数组	[]
ignoreselected	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 **interfaces** 插件的更多信息，请参阅 接口。

4.18. COLLECTD::PLUGIN::LOAD

使用 **load** 插件收集系统负载和系统使用的概览。

> This plugin is enabled by default.

表 4.20. plugin parameters

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 **load** 插件的更多信息，请参阅 load。

4.19. COLLECTD::PLUGIN::MCELOG

使用 **mcelog** 插件发送通知和统计数据，这些通知与 Machine Check Exceptions 发生时相关。将 **mcelog** 配置为以守护进程模式运行并启用日志功能。

表 4.21. 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。

4.20. COLLECTD::PLUGIN::MEMCACHED

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

表 4.22. 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。

4.21. COLLECTD::PLUGIN::MEMORY

使用 **memory** 插件检索有关系统内存的信息。

> This plugin is enabled by default.

表 4.23. 内存参数

参数	类型
默认值	valuesabsolute
布尔值	true
valuespercentage	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 **memory** 插件的更多信息，请参阅内存。

4.22. 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 4.24. ntpd 参数

参数	类型
主机	Hostname
port	端口号(Integer)
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** 插件的详情，请参考 ntpd。

4.23. COLLECTD::PLUGIN::OVS_STATS

使用 **ovs_stats** 插件收集 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 4.25. ovs_stats parameters

参数	类型
address	字符串
bridge	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。

4.24. COLLECTD::PLUGIN::PROCESSES

processes 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，插件只根据状态和进程分叉率收集进程数量。

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 4.26. plugin parameters

参数	类型	默认值
进程	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。

4.25. COLLECTD::PLUGIN::SMART

使用 **smart** 插件从节点上的物理磁盘中收集 SMART （自助监控、分析和报告技术）信息。您还必须将 **CollectdContainerAdditionalCapAdd** 参数设置为 **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：

智能插件：以 root 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 系统丢弃功能吗？。

表 4.27. 智能参数

参数	类型
disks	数组
ignoreselected	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 **smart** 插件的更多信息，请参阅 smart。

4.26. COLLECTD::PLUGIN::SWAP

使用 **swap** 插件收集有关可用和已用 swap 空间的信息。

表 4.28. swap 参数

参数	类型
reportbydevice	布尔值
reportbytes	布尔值
valuesabsolute	布尔值
valuespercentage	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
```

4.27. COLLECTD::PLUGIN::TCPCONNS

使用 **tcpconns** 插件收集有关从配置的端口入站或出站的 TCP 连接数的信息。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 4.29. tcpconns 参数

参数	类型
localports	port (Array)
remoteports	port (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
```

4.28. COLLECTD::PLUGIN::THERMAL

使用rmal **插件** 检索 ACPI rmal 区域信息。

表 4.30. Thermal parameters

参数	类型
devices	数组
ignoreselected	布尔值

参数	类型
interval	整数

配置示例：

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

4.29. COLLECTD::PLUGIN::UPTIME

使用 **uptime** 插件收集有关系统正常运行时间的信息。

This plugin is enabled by default.

表 4.31. uptime 参数

参数	类型
interval	整数

4.30. COLLECTD::PLUGIN::VIRT

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

此插件在计算主机上默认启用。

表 4.32. 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。

4.31. 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 4.33. 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
```

4.32. COLLECTD::PLUGIN::WRITE_HTTP

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

表 4.34. write_http 参数

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。

4.33. 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，您必须指定唯一名称和一个 Kafka producer。您可以在主题块中使用以下 per-topic 参数：

表 4.35. write_kafka parameters

参数	类型
kafka_hosts	Array[String]
topics	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。