Red Hat OpenStack Platform 17.1

自动扩展实例

在 Red Hat OpenStack Platform 中配置自动扩展

Last Updated: 2024-05-23

Red Hat OpenStack Platform 17.1 自动扩展实例

在 Red Hat OpenStack Platform 中配置自动扩展

OpenStack Team
rhos-docs@redhat.com

法律通告

摘要

使用 Red Hat OpenStack Platform 遥测组件和 heat 模板为工作负载自动启动实例。

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 自动扩展组件简介

使用遥测组件收集有关 Red Hat OpenStack Platform (RHOSP)环境的数据，如 CPU、存储和内存用量。您可以启动和扩展实例，以响应工作负载需求和资源可用性。您可以定义遥测数据的上限和下限，以控制编排服务(heat)模板中的实例扩展。

使用以下遥测组件控制自动实例扩展：

- **数据收集** ：Telemetry 使用数据收集服务(Ceilometer)来收集指标和事件数据。
- **存储** ：Telemetry 将指标数据存储在时间序列数据库服务(gnocchi)中。
- **警报** ：Telemetry 使用 Alarming 服务(aodh)根据规则针对 Ceilometer 收集的指标或事件数据触发操作。

1.1. 用于自动扩展的数据收集服务(CEILOMETER)

您可以使用 Ceilometer 收集 Red Hat OpenStack Platform (RHOSP)组件的计量和事件信息的数据。

Ceilometer 服务使用三个代理从 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使用统计数据。
- **中央代理(ceilometer-agent-central)** ：在 Controller 节点上运行轮询，以轮询资源使用不是由 Compute 节点提供的资源的统计数据。
- **通知代理(ceilometer-agent-notification)** ：在 Controller 节点上运行，并使用来自消息队列的消息来构建事件和计量数据。

Ceilometer 代理使用发布者将数据发送到对应的端点，如时间序列数据库服务(gnocchi)。

其他资源

- *overcloud 可观察性*指南中的 Ceilometer.

1.1.1. Publishers

在 Red Hat OpenStack Platform (RHOSP)中，您可以使用多种传输方法将收集的数据传送到存储或外部系统中，如 Service Telemetry Framework (STF)。

当您启用 gnocchi publisher 时，测量和资源信息会存储为时间序列数据。

1.2. 用于自动扩展的时间序列数据库服务(GNOCCHI)

Gnocchi 是一个时间序列数据库，可用于将指标存储在 SQL 中。警报服务(aodh)和编排服务(heat)使用存储在 gnocchi 中的数据进行自动扩展。

其他资源

- 带有 gnocchi 的存储。

1.3. 警报服务(AODH)

您可以配置 Alarming 服务(aodh)，以基于针对 Ceilometer 收集的指标数据的规则触发操作，并存储在 gnocchi 中。警报可以处于以下状态之一：

- **OK** ：指标或事件处于可接受的状态。
- **trigger** ：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，例如，没有请求粒度的数据，或者检查尚未执行，以此类推。

1.4. 用于自动扩展的编排服务(HEAT)

director 使用编排服务(heat)模板作为 overcloud 部署的模板格式。Heat 模板通常以 YAML 格式表示。模板的目的是定义和创建堆栈，这是 heat 创建的资源集合，以及资源的配置。资源是 Red Hat OpenStack Platform (RHOSP)中的对象，可以包含计算资源、网络配置、安全组、扩展规则和自定义资源。

其他资源

- 了解 heat 模板.

第 2 章 配置和部署 OVERCLOUD 进行自动扩展

您必须为 overcloud 上的服务配置模板来启用自动扩展。

流程

1. 在部署 overcloud 进行自动扩展前，为自动扩展服务创建环境模板和资源 registry。如需更多信息，请参阅 第 2.1 节 “配置 overcloud 进行自动扩展”。
2. 部署 overcloud。如需更多信息，请参阅 第 2.2 节 “部署 overcloud 进行自动扩展”。

2.1. 配置 OVERCLOUD 进行自动扩展

创建部署提供自动扩展的服务所需的环境模板和资源 registry。

流程

1. 使用您的 overcloud 管理员凭据登录 undercloud 主机，如 **overcloudrc**。
2. 为自动扩展配置文件创建目录：

```
$ mkdir -p $HOME/templates/autoscaling/
```

3. 为服务自动扩展所需的定义创建资源 registry 文件：

```
$ cat <<EOF > $HOME/templates/autoscaling/resources-autoscaling.yaml
resource_registry:
  OS::TripleO::Services::AodhApi: /usr/share/openstack-tripleo-heat-templates/deployment/aodh/aodh-api-container-puppet.yaml
  OS::TripleO::Services::AodhEvaluator: /usr/share/openstack-tripleo-heat-templates/deployment/aodh/aodh-evaluator-container-puppet.yaml
  OS::TripleO::Services::AodhListener: /usr/share/openstack-tripleo-heat-templates/deployment/aodh/aodh-listener-container-puppet.yaml
  OS::TripleO::Services::AodhNotifier: /usr/share/openstack-tripleo-heat-templates/deployment/aodh/aodh-notifier-container-puppet.yaml
  OS::TripleO::Services::CeilometerAgentCentral: /usr/share/openstack-tripleo-heat-templates/deployment/ceilometer/ceilometer-agent-central-container-puppet.yaml
  OS::TripleO::Services::CeilometerAgentNotification: /usr/share/openstack-tripleo-heat-templates/deployment/ceilometer/ceilometer-agent-notification-container-puppet.yaml
  OS::TripleO::Services::ComputeCeilometerAgent: /usr/share/openstack-tripleo-heat-templates/deployment/ceilometer/ceilometer-agent-compute-container-puppet.yaml
  OS::TripleO::Services::GnocchiApi: /usr/share/openstack-tripleo-heat-templates/deployment/gnocchi/gnocchi-api-container-puppet.yaml
  OS::TripleO::Services::GnocchiMetricd: /usr/share/openstack-tripleo-heat-templates/deployment/gnocchi/gnocchi-metricd-container-puppet.yaml
  OS::TripleO::Services::GnocchiStatsd: /usr/share/openstack-tripleo-heat-templates/deployment/gnocchi/gnocchi-statsd-container-puppet.yaml
  OS::TripleO::Services::HeatApi: /usr/share/openstack-tripleo-heat-templates/deployment/heat/heat-api-container-puppet.yaml
  OS::TripleO::Services::HeatEngine: /usr/share/openstack-tripleo-heat-templates/deployment/heat/heat-engine-container-puppet.yaml
  OS::TripleO::Services::Redis: /usr/share/openstack-tripleo-heat-templates/deployment/database/redis-container-puppet.yaml
EOF
```

4. 创建环境模板来配置自动扩展所需的服务：

```
cat <<EOF > $HOME/templates/autoscaling/parameters-autoscaling.yaml
parameter_defaults:
  NotificationDriver: 'messagingv2'
  GnocchiDebug: false
  CeilometerEnableGnocchi: true
  ManagePipeline: true
  ManageEventPipeline: true

  EventPipelinePublishers:
    - gnocchi://?archive_policy=generic
  PipelinePublishers:
    - gnocchi://?archive_policy=generic

  ManagePolling: true
  ExtraConfig:
    ceilometer::agent::polling::polling_interval: 60
EOF
```

如果您使用 Red Hat Ceph Storage 作为时间序列数据库服务的数据存储后端，请在 **parameters-autoscaling.yaml** 文件中添加以下参数：

```
parameter_defaults:
  GnocchiRbdPoolName: 'metrics'
  GnocchiBackend: 'rbd'
```

您必须先创建定义的归档策略 **通用**，然后才能存储指标。您可以在部署后定义此归档策略。如需更多信息，请参阅 第 3.1 节 “创建用于自动扩展的通用归档策略”。

5. 设置 **polling_interval** 参数，例如 60 秒。**polling_interval** 参数的值必须与您在创建归档策略时定义的 gnocchi granularity 值匹配。如需更多信息，请参阅 第 3.1 节 “创建用于自动扩展的通用归档策略”。

6. 部署 overcloud。如需更多信息，请参阅 第 2.2 节 “部署 overcloud 进行自动扩展”。

2.2. 部署 OVERCLOUD 进行自动扩展

您可以使用 director 部署 overcloud 进行自动扩展。

先决条件

- 您已创建了环境模板，以部署提供自动扩展功能的服务。如需更多信息，请参阅 第 2.1 节 “配置 overcloud 进行自动扩展”。

流程

- 第 2.2.1 节 “使用 director 部署 overcloud 进行自动扩展”

2.2.1. 使用 director 部署 overcloud 进行自动扩展

使用 director 部署 overcloud。

先决条件

- 已部署 undercloud。有关更多信息，请参阅在 undercloud 上安装 director。

流程

1. 以 **stack** 用户身份登录 undercloud。
2. 查找 **stackrc** undercloud 凭证文件：

```
[stack@director ~]$ source ~/stackrc
```

3. 使用其他环境文件将自动扩展环境文件添加到堆栈中，并部署 overcloud：

```
(undercloud)$ openstack overcloud deploy --templates \
  -e [your environment files] \
  -e $HOME/templates/autoscaling/parameters-autoscaling.yaml \
  -e $HOME/templates/autoscaling/resources-autoscaling.yaml
```

2.3. 验证 OVERCLOUD 部署以进行自动扩展

验证自动扩展服务是否已部署并已启用。验证输出示例可能与您的用例不同。

先决条件

- 您已使用 director 在现有 overcloud 中部署了自动扩展服务。如需更多信息，请参阅 第 2.2 节 “部署 overcloud 进行自动扩展”。

流程

1. 以 **stack** 用户身份登录您的环境。
2. 对于 director 环境，提供 **stackrc** undercloud 凭证文件：

```
[stack@undercloud ~]$ source ~/stackrc
```

验证

1. 验证部署是否成功，并确保自动扩展的服务 API 端点可用：

```
$ openstack endpoint list --service metric
+----------------------------------+-----------+-------------+-------------+---------+-----------+---------
----------------+
| ID                               | Region    | Service Name | Service Type | Enabled | Interface | URL
|
+----------------------------------+-----------+-------------+-------------+---------+-----------+---------
----------------+
| 2956a12327b744b29abd4577837b2e6f | regionOne | gnocchi      | metric       | True    |
internal  | http://192.168.25.3:8041 |
| 583453c58b064f69af3de3479675051a | regionOne | gnocchi      | metric       | True    |
admin     | http://192.168.25.3:8041 |
| fa029da0e2c047fc9d9c50eb6b4876c6 | regionOne | gnocchi      | metric       | True    | public
| http://192.168.25.3:8041 |
+----------------------------------+-----------+-------------+-------------+---------+-----------+---------
----------------+
```

```
$ openstack endpoint list --service alarming
+----------------------------------+-----------+-------------+--------------+---------+-----------+---------
----------------+
| ID                               | Region    | Service Name | Service Type | Enabled | Interface | URL
|
+----------------------------------+-----------+-------------+--------------+---------+-----------+---------
----------------+
| 08c70ec137b44ed68590f4d5c31162bb | regionOne | aodh         | alarming     | True    |
internal  | http://192.168.25.3:8042 |
| 194042887f3d4eb4b638192a0fe60996 | regionOne | aodh         | alarming     | True    |
admin     | http://192.168.25.3:8042 |
| 2604b693740245ed8960b31dfea1f963 | regionOne | aodh         | alarming     | True    |
public    | http://192.168.25.3:8042 |
+----------------------------------+-----------+-------------+--------------+---------+-----------+---------
----------------+
```

使用 overcloud 凭证检查 heat 服务：

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

$ openstack endpoint list --service orchestration
+----------------------------------+-----------+-------------+---------------+---------+-----------+--------
-----------------------------------+
| ID                               | Region    | Service Name | Service Type  | Enabled | Interface | URL
|
+----------------------------------+-----------+-------------+---------------+---------+-----------+--------
-----------------------------------+
| 00966a24dd4141349e12680307c11848 | regionOne | heat         | orchestration | True    |
admin     | http://192.168.25.3:8004/v1/%(tenant_id)s |
| 831e411bb6d44f6db9f5103d659f901e | regionOne | heat         | orchestration | True    |
public    | http://192.168.25.3:8004/v1/%(tenant_id)s |
| d5be22349add43ae95be4284a42a4a60 | regionOne | heat         | orchestration | True    |
internal  | http://192.168.25.3:8004/v1/%(tenant_id)s |
+----------------------------------+-----------+-------------+---------------+---------+-----------+--------
-----------------------------------+
```

2. 验证服务是否在 overcloud 上运行：

```
$ sudo podman ps --filter=name='heat|gnocchi|ceilometer|aodh'
CONTAINER ID  IMAGE                                                COMMAND      CREATED
STATUS                    PORTS       NAMES
31e75d62367f  registry.redhat.io/rhosp-rhel9/openstack-aodh-api:17.1              kolla_start
27 minutes ago  Up 27 minutes ago (healthy)             aodh_api
77acf3487736  registry.redhat.io/rhosp-rhel9/openstack-aodh-listener:17.1          kolla_start
27 minutes ago  Up 27 minutes ago (healthy)             aodh_listener
29ec47b69799  registry.redhat.io/rhosp-rhel9/openstack-aodh-evaluator:17.1
kolla_start  27 minutes ago  Up 27 minutes ago (healthy)             aodh_evaluator
43efaa86c769  registry.redhat.io/rhosp-rhel9/openstack-aodh-notifier:17.1           kolla_start
27 minutes ago  Up 27 minutes ago (healthy)             aodh_notifier
0ac8cb2c7470  registry.redhat.io/rhosp-rhel9/openstack-aodh-api:17.1              kolla_start
27 minutes ago  Up 27 minutes ago (healthy)             aodh_api_cron
31b55e091f57  registry.redhat.io/rhosp-rhel9/openstack-ceilometer-central:17.1
kolla_start  27 minutes ago  Up 27 minutes ago (healthy)             ceilometer_agent_central
5f61331a17d8  registry.redhat.io/rhosp-rhel9/openstack-ceilometer-compute:17.1
kolla_start  27 minutes ago  Up 27 minutes ago (healthy)             ceilometer_agent_compute
```

```
7c5ef75d8f1b  registry.redhat.io/rhosp-rhel9/openstack-ceilometer-notification:17.1
kolla_start  27 minutes ago  Up 27 minutes ago (healthy)
ceilometer_agent_notification
88fa57cc1235  registry.redhat.io/rhosp-rhel9/openstack-gnocchi-api:17.1              kolla_start
23 minutes ago  Up 23 minutes ago (healthy)            gnocchi_api
0f05a58197d5  registry.redhat.io/rhosp-rhel9/openstack-gnocchi-metricd:17.1
kolla_start  23 minutes ago  Up 23 minutes ago (healthy)            gnocchi_metricd
6d806c285500  registry.redhat.io/rhosp-rhel9/openstack-gnocchi-statsd:17.1
kolla_start  23 minutes ago  Up 23 minutes ago (healthy)            gnocchi_statsd
7c02cac34c69  registry.redhat.io/rhosp-rhel9/openstack-heat-api:17.1                kolla_start
27 minutes ago  Up 27 minutes ago (healthy)            heat_api_cron
d3903df545ce  registry.redhat.io/rhosp-rhel9/openstack-heat-api:17.1                kolla_start
27 minutes ago  Up 27 minutes ago (healthy)            heat_api
db1d33506e3d  registry.redhat.io/rhosp-rhel9/openstack-heat-api-cfn:17.1             kolla_start
27 minutes ago  Up 27 minutes ago (healthy)            heat_api_cfn
051446294c70  registry.redhat.io/rhosp-rhel9/openstack-heat-engine:17.1              kolla_start
27 minutes ago  Up 27 minutes ago (healthy)            heat_engine
```

3. 验证时间序列数据库服务是否可用：

```
$ openstack metric status --fit-width
+-----------------------------------------------------+-------------------------------------------------------
-------------------------------------------------------+
| Field                                       | Value
|
+-----------------------------------------------------+-------------------------------------------------------
-------------------------------------------------------+
| metricd/processors                                  | ['host-80.general.local.0.a94fbf77-1ac0-49ed-bfe2-
a89f014fde01',                                        |
|                                             | 'host-80.general.local.3.28ca78d7-a80e-4515-8060-
233360b410eb',                                        |
|                                             | 'host-80.general.local.1.7e8b5a5b-2ca1-49be-bc22-
25f51d67c00a',                                        |
|                                             | 'host-80.general.local.2.3c4fe59e-23cd-4742-833d-
42ff0a4cb692']                                        |
| storage/number of metric having measures to process | 0
|
| storage/total number of measures to process         | 0
|
+-----------------------------------------------------+-------------------------------------------------------
-------------------------------------------------------+
```

第 3 章 使用 HEAT 服务进行自动扩展

在部署 overcloud 中提供自动扩展所需的服务后，您必须配置 overcloud 环境，以便编排服务(heat)可以管理实例以进行自动扩展。

先决条件

- 已部署 overcloud。如需更多信息，请参阅 第 2.2 节 “部署 overcloud 进行自动扩展”。

流程

- 第 3.1 节 “创建用于自动扩展的通用归档策略”
- 第 3.2 节 “为自动扩展实例配置 heat 模板”
- 第 3.3 节 “创建用于自动扩展的堆栈部署”

3.1. 创建用于自动扩展的通用归档策略

在 overcloud 中部署用于自动扩展的服务后，您必须配置 overcloud 环境，以便编排服务(heat)能够管理用于自动扩展的实例。

先决条件

- 您已部署了具有自动扩展服务的 overcloud。如需更多信息，请参阅 第 2.1 节 “配置 overcloud 进行自动扩展”。

流程

1. 以 **stack** 用户身份登录您的环境。
2. 对于 director 环境，提供 **stackrc** 文件：

```
[stack@undercloud ~]$ source ~/stackrc
```

3. 创建 **$HOME/templates/autoscaling/parameters-autoscaling.yaml** 中定义的归档策略：

```
$ openstack metric archive-policy create generic \
  --back-window 0 \
  --definition timespan:'4:00:00',granularity:'0:01:00',points:240 \
  --aggregation-method 'rate:mean' \
  --aggregation-method 'mean'
```

验证

- 验证是否已创建归档策略：

```
$ openstack metric archive-policy show generic
+--------------------+-----------------------------------------------------+
| Field              | Value                                               |
+--------------------+-----------------------------------------------------+
| aggregation_methods | mean, rate:mean                                    |
| back_window        | 0                                                   |
```

```
| definition          | - timespan: 4:00:00, granularity: 0:01:00, points: 240 |
| name                | generic                                                |
+---------------------+--------------------------------------------------------+
```

3.2. 为自动扩展实例配置 HEAT 模板

您可以配置编排服务(heat)模板来创建实例，并配置在触发时创建和扩展实例的警报。

注意

此流程使用示例值，您必须更改为适合您的环境。

先决条件

- 您已使用自动扩展服务部署了 overcloud。如需更多信息，请参阅 第 2.2 节 “部署 overcloud 进行自动扩展”。
- 您已为 overcloud 配置了一个用于自动扩展遥测存储的归档策略。如需更多信息，请参阅 第 3.1 节 “创建用于自动扩展的通用归档策略”。

流程

1. 以 **stack** 用户身份登录您的环境。

```
$ source ~/stackrc
```

2. 创建存放自动扩展组实例配置的目录：

```
$ mkdir -p $HOME/templates/autoscaling/vnf/
```

3. 创建实例配置模板，如 **$HOME/templates/autoscaling/vnf/instance.yaml**。
4. 在 **instance.yaml** 文件中添加以下配置：

```
cat <<EOF > $HOME/templates/autoscaling/vnf/instance.yaml
heat_template_version: wallaby
description: Template to control scaling of VNF instance

parameters:
  metadata:
    type: json
  image:
    type: string
    description: image used to create instance
    default: fedora36
  flavor:
    type: string
    description: instance flavor to be used
    default: m1.small
  key_name:
    type: string
    description: keypair to be used
    default: default
  network:
```

```
    type: string
    description: project network to attach instance to
    default: private
  external_network:
    type: string
    description: network used for floating IPs
    default: public

resources:
  vnf:
    type: OS::Nova::Server
    properties:
      flavor: {get_param: flavor}
      key_name: {get_param: key_name}
      image: { get_param: image }
      metadata: { get_param: metadata }
      networks:
        - port: { get_resource: port }

  port:
    type: OS::Neutron::Port
    properties:
      network: {get_param: network}
      security_groups:
        - basic

  floating_ip:
    type: OS::Neutron::FloatingIP
    properties:
      floating_network: {get_param: external_network }

  floating_ip_assoc:
    type: OS::Neutron::FloatingIPAssociation
    properties:
      floatingip_id: { get_resource: floating_ip }
      port_id: { get_resource: port }
EOF
```

- **parameters** 参数为此新资源定义自定义参数。
- **resources** 参数的 **vnf** 子参数定义了在 **OS::Heat::AutoScalingGroup** 中引用的自定义子资源的名称，例如 **OS::Nova::Server::VNF**。

5. 创建要在 heat 模板中引用的资源：

```
$ cat <<EOF > $HOME/templates/autoscaling/vnf/resources.yaml
resource_registry:
  "OS::Nova::Server::VNF": $HOME/templates/autoscaling/vnf/instance.yaml
EOF
```

6. 为 heat 创建部署模板以控制实例扩展：

```
$ cat <<EOF > $HOME/templates/autoscaling/vnf/template.yaml
heat_template_version: wallaby
description:  Example auto scale group, policy and alarm
resources:
```

```
scaleup_group:
  type: OS::Heat::AutoScalingGroup
  properties:
    max_size: 3
    min_size: 1
    #desired_capacity: 1
    resource:
      type: OS::Nova::Server::VNF
      properties:
        metadata: {"metering.server_group": {get_param: "OS::stack_id"}}

scaleup_policy:
  type: OS::Heat::ScalingPolicy
  properties:
    adjustment_type: change_in_capacity
    auto_scaling_group_id: { get_resource: scaleup_group }
    cooldown: 60
    scaling_adjustment: 1

scaledown_policy:
  type: OS::Heat::ScalingPolicy
  properties:
    adjustment_type: change_in_capacity
    auto_scaling_group_id: { get_resource: scaleup_group }
    cooldown: 60
    scaling_adjustment: -1

cpu_alarm_high:
  type: OS::Aodh::GnocchiAggregationByResourcesAlarm
  properties:
    description: Scale up instance if CPU > 50%
    metric: cpu
    aggregation_method: rate:mean
    granularity: 60
    evaluation_periods: 3
    threshold: 60000000000.0
    resource_type: instance
    comparison_operator: gt
    alarm_actions:
      - str_replace:
          template: trust+url
          params:
            url: {get_attr: [scaleup_policy, signal_url]}
    query:
      list_join:
        - ''
        - - {'=': {server_group: {get_param: "OS::stack_id"}}}

cpu_alarm_low:
  type: OS::Aodh::GnocchiAggregationByResourcesAlarm
  properties:
    description: Scale down instance if CPU < 20%
    metric: cpu
    aggregation_method: rate:mean
    granularity: 60
    evaluation_periods: 3
```

```
      threshold: 24000000000.0
      resource_type: instance
      comparison_operator: lt
      alarm_actions:
        - str_replace:
            template: trust+url
            params:
              url: {get_attr: [scaledown_policy, signal_url]}
      query:
        list_join:
          - ''
          - - {'=': {server_group: {get_param: "OS::stack_id"}}}

outputs:
  scaleup_policy_signal_url:
    value: {get_attr: [scaleup_policy, alarm_url]}

  scaledown_policy_signal_url:
    value: {get_attr: [scaledown_policy, alarm_url]}
EOF
```

注意

堆栈上的输出是信息性的，在 ScalingPolicy 或 AutoScalingGroup 中不引用。要查看输出，请使用 **openstack stack show <stack_name>** 命令。

3.3. 创建用于自动扩展的堆栈部署

为工作的 VNF 自动扩展示例创建堆栈部署。

先决条件

- 第 3.2 节 “为自动扩展实例配置 heat 模板”

流程

1. 使用您的 overcloud 管理员凭证登录 undercloud 主机，如 **overcloudrc** ：

```
(undercloud)$ source ~/overcloudrc
```

2. 创建堆栈：

```
$ openstack stack create \
  -t $HOME/templates/autoscaling/vnf/template.yaml \
  -e $HOME/templates/autoscaling/vnf/resources.yaml \
  vnf
```

验证

1. 验证堆栈是否已成功创建：

```
$ openstack stack show vnf -c id -c stack_status
+--------------+--------------------------------------+
```

```
| Field        | Value                                |
+--------------+--------------------------------------+
| id           | cb082cbd-535e-4779-84b0-98925e103f5e |
| stack_status | CREATE_COMPLETE                      |
+--------------+--------------------------------------+
```

2. 验证堆栈资源是否已创建，包括警报、扩展策略和自动扩展组：

```
$ export STACK_ID=$(openstack stack show vnf -c id -f value)
```

```
$ openstack stack resource list $STACK_ID
+------------------+--------------------------------------+----------------------------------------+-----
-------------+---------------------+
| resource_name    | physical_resource_id                 | resource_type                          |
resource_status | updated_time        |
+------------------+--------------------------------------+----------------------------------------+-----
-------------+---------------------+
| cpu_alarm_high   | d72d2e0d-1888-4f89-b888-02174c48e463 |
OS::Aodh::GnocchiAggregationByResourcesAlarm | CREATE_COMPLETE | 2022-10-
06T23:08:37Z |
| scaleup_policy   | 1c4446b7242e479090bef4b8075df9d4     | OS::Heat::ScalingPolicy
| CREATE_COMPLETE | 2022-10-06T23:08:37Z |
| cpu_alarm_low    | b9c04ef4-8b57-4730-af03-1a71c3885914 |
OS::Aodh::GnocchiAggregationByResourcesAlarm | CREATE_COMPLETE | 2022-10-
06T23:08:37Z |
| scaledown_policy | a5af7faf5a1344849c3425cb2c5f18db     | OS::Heat::ScalingPolicy
| CREATE_COMPLETE | 2022-10-06T23:08:37Z |
| scaleup_group    | 9609f208-6d50-4b8f-836e-b0222dc1e0b1 | OS::Heat::AutoScalingGroup
| CREATE_COMPLETE | 2022-10-06T23:08:37Z |
+------------------+--------------------------------------+----------------------------------------+-----
-------------+---------------------+
```

3. 通过堆栈创建验证实例是否已启动：

```
$ openstack server list --long | grep $STACK_ID

| 62e1b27c-8d9d-44a5-a0f0-80e7e6d437c7 | vn-dvaxcqb-6bqh2qd2fpif-hicmkm5dzjug-vnf-
ywrydc5wqjjc | ACTIVE | None       | Running     | private=192.168.100.61, 192.168.25.99 |
fedora36   | a6aa7b11-1b99-4c62-a43b-d0b7c77f4b72 | m1.small    | 5cd46fec-50c2-43d5-
89e8-ed3fa7660852 | nova             | host-80.localdomain |
metering.server_group='cb082cbd-535e-4779-84b0-98925e103f5e' |
```

4. 验证是否为堆栈创建了警报：

   a. 列出警报 ID。在一段时间内，警报的状态可能位于 **数据不足** 的状态。最短的时间是数据收集和数据存储粒度设置的轮询间隔：

```
$ openstack alarm list
+--------------------------------------+----------------------------------------+--------------------
------------+-------+----------+---------+
| alarm_id                             | type                                   | name                       | state |
severity | enabled |
+--------------------------------------+----------------------------------------+--------------------
------------+-------+----------+---------+
| b9c04ef4-8b57-4730-af03-1a71c3885914 |
```

```
gnocchi_aggregation_by_resources_threshold | vnf-cpu_alarm_low-pve5eal6ykst  |
alarm | low      | True    |
| d72d2e0d-1888-4f89-b888-02174c48e463 |
gnocchi_aggregation_by_resources_threshold | vnf-cpu_alarm_high-5xx7qvfsurxe | ok
| low      | True    |
+--------------------------------------+----------------------------------------+--------------------
------------+-------+----------+---------+
```

b. 列出堆栈的资源，并记录下 **cpu_alarm_high** 和 **cpu_alarm_low** 资源的 **physical_resource_id** 值。

```
$ openstack stack resource list $STACK_ID
+-----------------+--------------------------------------+----------------------------------------
+----------------+---------------------+
| resource_name    | physical_resource_id                 | resource_type                          |
resource_status | updated_time         |
+-----------------+--------------------------------------+----------------------------------------
+----------------+---------------------+
| cpu_alarm_high   | d72d2e0d-1888-4f89-b888-02174c48e463 |
OS::Aodh::GnocchiAggregationByResourcesAlarm | CREATE_COMPLETE | 2022-10-
06T23:08:37Z |
| scaleup_policy   | 1c4446b7242e479090bef4b8075df9d4     | OS::Heat::ScalingPolicy
| CREATE_COMPLETE | 2022-10-06T23:08:37Z |
| cpu_alarm_low    | b9c04ef4-8b57-4730-af03-1a71c3885914 |
OS::Aodh::GnocchiAggregationByResourcesAlarm | CREATE_COMPLETE | 2022-10-
06T23:08:37Z |
| scaledown_policy | a5af7faf5a1344849c3425cb2c5f18db     | OS::Heat::ScalingPolicy
| CREATE_COMPLETE | 2022-10-06T23:08:37Z |
| scaleup_group    | 9609f208-6d50-4b8f-836e-b0222dc1e0b1 |
OS::Heat::AutoScalingGroup                | CREATE_COMPLETE | 2022-10-
06T23:08:37Z |
+-----------------+--------------------------------------+----------------------------------------
+----------------+---------------------+
```

physical_resource_id 的值必须与 **openstack alarm list** 命令的输出中的 **alarm_id** 匹配。

5. 验证堆栈是否存在指标资源。将 **server_group** 查询的值设置为堆栈 ID：

```
$ openstack metric resource search --sort-column launched_at -c id -c display_name -c
launched_at -c deleted_at --type instance server_group="$STACK_ID"
+--------------------------------------+--------------------------------------------------------+--------------
------------------+------------+
| id                                   | display_name                                           | launched_at
| deleted_at |
+--------------------------------------+--------------------------------------------------------+--------------
------------------+------------+
| 62e1b27c-8d9d-44a5-a0f0-80e7e6d437c7 | vn-dvaxcqb-6bqh2qd2fpif-hicmkm5dzjug-vnf-
ywrydc5wqjjc | 2022-10-06T23:09:28.496566+00:00 | None       |
+--------------------------------------+--------------------------------------------------------+--------------
------------------+------------+
```

6. 验证通过堆栈创建的实例资源是否存在测量：

```
$ openstack metric aggregates --resource-type instance --sort-column timestamp '(metric
cpu rate:mean)' server_group="$STACK_ID"
```

```
+--------------------------------------------------+-------------------------+-------------+---------------
+
| name                                             | timestamp               | granularity |         value |
+--------------------------------------------------+-------------------------+-------------+---------------
+
| 62e1b27c-8d9d-44a5-a0f0-80e7e6d437c7/cpu/rate:mean | 2022-10-06T23:11:00+00:00 |
60.0 | 69470000000.0 |
| 62e1b27c-8d9d-44a5-a0f0-80e7e6d437c7/cpu/rate:mean | 2022-10-06T23:12:00+00:00 |
60.0 | 81060000000.0 |
| 62e1b27c-8d9d-44a5-a0f0-80e7e6d437c7/cpu/rate:mean | 2022-10-06T23:13:00+00:00 |
60.0 | 82840000000.0 |
| 62e1b27c-8d9d-44a5-a0f0-80e7e6d437c7/cpu/rate:mean | 2022-10-06T23:14:00+00:00 |
60.0 | 66660000000.0 |
| 62e1b27c-8d9d-44a5-a0f0-80e7e6d437c7/cpu/rate:mean | 2022-10-06T23:15:00+00:00 |
60.0 |  7360000000.0 |
| 62e1b27c-8d9d-44a5-a0f0-80e7e6d437c7/cpu/rate:mean | 2022-10-06T23:16:00+00:00 |
60.0 |  3150000000.0 |
| 62e1b27c-8d9d-44a5-a0f0-80e7e6d437c7/cpu/rate:mean | 2022-10-06T23:17:00+00:00 |
60.0 |  2760000000.0 |
| 62e1b27c-8d9d-44a5-a0f0-80e7e6d437c7/cpu/rate:mean | 2022-10-06T23:18:00+00:00 |
60.0 |  3470000000.0 |
| 62e1b27c-8d9d-44a5-a0f0-80e7e6d437c7/cpu/rate:mean | 2022-10-06T23:19:00+00:00 |
60.0 |  2770000000.0 |
| 62e1b27c-8d9d-44a5-a0f0-80e7e6d437c7/cpu/rate:mean | 2022-10-06T23:20:00+00:00 |
60.0 |  2700000000.0 |
+--------------------------------------------------+-------------------------+-------------+---------------
+
```

第 4 章 测试和故障排除自动扩展

使用编排服务(heat)根据阈值定义自动扩展实例。要排除您的环境，您可以在日志文件和历史记录记录中查找错误。

4.1. 测试自动扩展实例

您可以使用编排服务(heat)根据 **cpu_ alarm_high** 阈值定义自动扩展实例。当 CPU 使用达到 **threshold** 参数中定义的值时，另一个实例会启动以平衡负载。**template.yaml** 文件中的 **阈值** 设置为 80%。

流程

1. 以 **stack** 用户身份登录主机环境。
2. 对于 director 环境，提供 **stackrc** 文件：

```
[stack@undercloud ~]$ source ~/stackrc
```

3. 登录到实例：

```
$ ssh -i ~/mykey.pem cirros@192.168.122.8
```

4. 运行多个 **dd** 命令来生成负载：

```
[instance ~]$ sudo dd if=/dev/zero of=/dev/null &
[instance ~]$ sudo dd if=/dev/zero of=/dev/null &
[instance ~]$ sudo dd if=/dev/zero of=/dev/null &
```

5. 从正在运行的实例退出，再返回到主机。
6. 运行 **dd** 命令后，您可以预期在实例中使用 100% CPU。验证警报是否已触发：

```
$ openstack alarm list
+--------------------------------------+----------------------------------------+-------------------------
----------+-------+----------+---------+
| alarm_id                             | type                                   | name                          | state |
severity | enabled |
+--------------------------------------+----------------------------------------+-------------------------
----------+-------+----------+---------+
| 022f707d-46cc-4d39-a0b2-afd2fc7ab86a | gnocchi_aggregation_by_resources_threshold |
example-cpu_alarm_high-odj77qpbld7j | alarm | low      | True    |
| 46ed2c50-e05a-44d8-b6f6-f1ebd83af913 | gnocchi_aggregation_by_resources_threshold |
example-cpu_alarm_low-m37jvnm56x2t  | ok    | low      | True    |
+--------------------------------------+----------------------------------------+-------------------------
----------+-------+----------+---------+
```

7. 大约 60 秒后，编排将启动另一个实例并将它添加到组中。要验证实例是否已创建，请输入以下命令：

```
$ openstack server list
+--------------------------------------+-------------------------------------------------------+--------+------
------+------------+-----------------------------------+
| ID                                   | Name                                                  | Status | Task State | Power
```

```
State | Networks                          |
+--------------------------------------+-------------------------------------------------------+--------+------
------+------------+--------------------------------------+
| 477ee1af-096c-477c-9a3f-b95b0e2d4ab5 | ex-3gax-4urpikl5koff-yrxk3zxzfmpf-server-
2hde4tp4trnk | ACTIVE | -          | Running     | internal1=10.10.10.13, 192.168.122.17 |
| e1524f65-5be6-49e4-8501-e5e5d812c612 | ex-3gax-5f3a4og5cwn2-png47w3u2vjd-server-
vaajhuv4mj3j | ACTIVE | -          | Running     | internal1=10.10.10.9, 192.168.122.8   |
+--------------------------------------+-------------------------------------------------------+--------+------
------+------------+--------------------------------------+
```

8. 在另一个短时间内，观察编排服务已自动缩放到三个实例。该配置设置为最多三个实例。验证有三个实例：

```
$ openstack server list
+--------------------------------------+-------------------------------------------------------+--------+------
------+------------+--------------------------------------+
| ID                                   | Name                                                  | Status | Task State | Power
State | Networks                          |
+--------------------------------------+-------------------------------------------------------+--------+------
------+------------+--------------------------------------+
| 477ee1af-096c-477c-9a3f-b95b0e2d4ab5 | ex-3gax-4urpikl5koff-yrxk3zxzfmpf-server-
2hde4tp4trnk | ACTIVE | -          | Running     | internal1=10.10.10.13, 192.168.122.17 |
| e1524f65-5be6-49e4-8501-e5e5d812c612 | ex-3gax-5f3a4og5cwn2-png47w3u2vjd-server-
vaajhuv4mj3j | ACTIVE | -          | Running     | internal1=10.10.10.9, 192.168.122.8   |
| 6c88179e-c368-453d-a01a-555eae8cd77a | ex-3gax-fvxz3tr63j4o-36fhftuja3bw-server-
rhl4sqkjuy5p | ACTIVE | -          | Running     | internal1=10.10.10.5, 192.168.122.5   |
+--------------------------------------+-------------------------------------------------------+--------+------
------+------------+--------------------------------------+
```

4.2. 测试自动缩减实例

您可以使用编排服务(heat)根据 **cpu_ alarm_low** 阈值自动扩展实例。在本例中，当 CPU 使用率低于 5% 时，实例将缩减。

流程

1. 从工作负载实例中，终止正在运行的 **dd** 进程，并观察编排开始缩减实例。

```
$ killall dd
```

2. 以 **stack** 用户身份登录主机环境。

3. 对于 director 环境，提供 **stackrc** 文件：

```
[stack@undercloud ~]$ source ~/stackrc
```

4. 当您停止 **dd** 进程时，这将触发 **cpu_alarm_low 事件** 警报。因此，编排开始缩减和删除实例。验证对应的警报是否已触发：

```
$ openstack alarm list
+--------------------------------------+--------------------------------------------+-------------------------
-----------+-------+----------+---------+
| alarm_id                             | type                                       | name                     | state |
severity | enabled |
```

```
+--------------------------------------+----------------------------------------+-------------------------
----------+-------+----------+---------+
| 022f707d-46cc-4d39-a0b2-afd2fc7ab86a | gnocchi_aggregation_by_resources_threshold |
example-cpu_alarm_high-odj77qpbld7j | ok    | low      | True    |
| 46ed2c50-e05a-44d8-b6f6-f1ebd83af913 | gnocchi_aggregation_by_resources_threshold |
example-cpu_alarm_low-m37jvnm56x2t  | alarm | low      | True    |
+--------------------------------------+----------------------------------------+-------------------------
----------+-------+----------+---------+
```

几分钟后，编排持续将实例数量减少到 **scaleup_group** 定义的 **min_size** 参数中定义的最小值。在这种情况下，**min_size** 参数设置为 **1**。

4.3. 自动扩展故障排除

如果您的环境无法正常工作，您可以在日志文件和历史记录记录中查找错误。

流程

1. 以 **stack** 用户身份登录主机环境。
2. 对于 director 环境，提供 **stackrc** 文件：

```
[stack@undercloud ~]$ source ~/stackrc
```

3. 要检索有关状态转换的信息，请列出堆栈事件记录：

```
$ openstack stack event list example
2017-03-06 11:12:43Z [example]: CREATE_IN_PROGRESS  Stack CREATE started
2017-03-06 11:12:43Z [example.scaleup_group]: CREATE_IN_PROGRESS  state changed
2017-03-06 11:13:04Z [example.scaleup_group]: CREATE_COMPLETE  state changed
2017-03-06 11:13:04Z [example.scaledown_policy]: CREATE_IN_PROGRESS  state
changed
2017-03-06 11:13:05Z [example.scaleup_policy]: CREATE_IN_PROGRESS  state changed
2017-03-06 11:13:05Z [example.scaledown_policy]: CREATE_COMPLETE  state changed
2017-03-06 11:13:05Z [example.scaleup_policy]: CREATE_COMPLETE  state changed
2017-03-06 11:13:05Z [example.cpu_alarm_low]: CREATE_IN_PROGRESS  state changed
2017-03-06 11:13:05Z [example.cpu_alarm_high]: CREATE_IN_PROGRESS  state changed
2017-03-06 11:13:06Z [example.cpu_alarm_low]: CREATE_COMPLETE  state changed
2017-03-06 11:13:07Z [example.cpu_alarm_high]: CREATE_COMPLETE  state changed
2017-03-06 11:13:07Z [example]: CREATE_COMPLETE  Stack CREATE completed
successfully
2017-03-06 11:19:34Z [example.scaleup_policy]: SIGNAL_COMPLETE  alarm state
changed from alarm to alarm (Remaining as alarm due to 1 samples outside threshold, most
recent: 95.4080102993)
2017-03-06 11:25:43Z [example.scaleup_policy]: SIGNAL_COMPLETE  alarm state
changed from alarm to alarm (Remaining as alarm due to 1 samples outside threshold, most
recent: 95.8869217299)
2017-03-06 11:33:25Z [example.scaledown_policy]: SIGNAL_COMPLETE  alarm state
changed from ok to alarm (Transition to alarm due to 1 samples outside threshold, most
recent: 2.73931707966)
2017-03-06 11:39:15Z [example.scaledown_policy]: SIGNAL_COMPLETE  alarm state
changed from alarm to alarm (Remaining as alarm due to 1 samples outside threshold, most
recent: 2.78110858552)
```

4. 读取警报历史记录日志：

```
$ openstack alarm-history show 022f707d-46cc-4d39-a0b2-afd2fc7ab86a
+---------------------------+-----------------+-------------------------------------------------------------------
-----------------------------------+----------------------------------+
| timestamp                 | type            | detail
| event_id                         |
+---------------------------+-----------------+-------------------------------------------------------------------
-----------------------------------+----------------------------------+
| 2017-03-06T11:32:35.510000 | state transition | {"transition_reason": "Transition to ok due
to 1 samples inside threshold, most recent:             | 25e0e70b-3eda-466e-abac-
42d9cf67e704 |
|                           |                 | 2.73931707966", "state": "ok"}
|                                  |
| 2017-03-06T11:17:35.403000 | state transition | {"transition_reason": "Transition to alarm
due to 1 samples outside threshold, most recent:         | 8322f62c-0d0a-4dc0-9279-
435510f81039 |
|                           |                 | 95.0964497325", "state": "alarm"}
|                                  |
| 2017-03-06T11:15:35.723000 | state transition | {"transition_reason": "Transition to ok due
to 1 samples inside threshold, most recent:             | 1503bd81-7eba-474e-b74e-
ded8a7b630a1 |
|                           |                 | 3.59330523447", "state": "ok"}
|                                  |
| 2017-03-06T11:13:06.413000 | creation        | {"alarm_actions":
["trust+http://fca6e27e3d524ed68abdc0fd576aa848:delete@192.168.122.126:8004/v1/fd |
224f15c0-b6f1-4690-9a22-0c1d236e65f6 |
|                           |                 |
1c345135be4ee587fef424c241719d/stacks/example/d9ef59ed-b8f8-4e90-bd9b-
|                                  |
|                           |                 | ae87e73ef6e2/resources/scaleup_policy/signal"], "user_id":
"a85f83b7f7784025b6acdc06ef0a8fd8",      |                                  |
|                           |                 | "name": "example-cpu_alarm_high-odj77qpbld7j", "state":
"insufficient data", "timestamp":          |                                  |
|                           |                 | "2017-03-06T11:13:06.413455", "description": "Scale up if
CPU > 80%", "enabled": true,              |                                  |
|                           |                 | "state_timestamp": "2017-03-06T11:13:06.413455", "rule":
{"evaluation_periods": 1, "metric":       |                                  |
|                           |                 | "cpu_util", "aggregation_method": "mean", "granularity": 300,
"threshold": 80.0, "query": "{\"=\":   |                                  |
|                           |                 | {\"server_group\": \"d9ef59ed-b8f8-4e90-bd9b-
ae87e73ef6e2\"}}", "comparison_operator": "gt",       |                                  |
|                           |                 | "resource_type": "instance"}, "alarm_id": "022f707d-46cc-
4d39-a0b2-afd2fc7ab86a",                  |                                  |
|                           |                 | "time_constraints": [], "insufficient_data_actions": null,
"repeat_actions": true, "ok_actions":    |                                  |
|                           |                 | null, "project_id": "fd1c345135be4ee587fef424c241719d",
"type":                                  |                                  |
|                           |                 | "gnocchi_aggregation_by_resources_threshold", "severity":
"low"}                                |                                  |
+---------------------------+-----------------+-------------------------------------------------------------------
-----------------------------------+----------------------------------
```

5. 要查看 heat 为现有堆栈收集的横向扩展或缩减操作的记录，您可以使用 **awk** 命令解析 **heat-engine.log** ：

```
$ awk '/Stack UPDATE started/,/Stack CREATE completed successfully/ {print $0}'
/var/log/containers/heat/heat-engine.log
```

6. 要查看 aodh 相关信息，请检查 **aodh-evaluator.log** ：

```
$ sudo grep -i alarm /var/log/containers/aodh/aodh-evaluator.log | grep -i transition
```

4.4. 使用 RATE:MEAN 聚合时，使用 CPU 遥测值进行自动扩展阈值

当使用 **OS::Heat::Autoscaling** heat 编配模板 (HOT) 并为 CPU 设置阈值时，其代表以纳秒为单位的 CPU 时间，它是一个基于分配给实例工作负载虚拟 CPU 数量的动态值。在本参考指南中，我们将探索如何在使用 Gnocchi **rate:mean** aggregration 方法时，将如何计算和表达 CPU 纳秒值作为百分比。

4.4.1. 计算 CPU 遥测值的百分比

CPU 遥测存储在 Gnocchi (OpenStack 时间序列数据存储中)，以纳秒为单位的 CPU 使用率。当使用 CPU 遥测来定义自动扩展阈值时，将值表示为 CPU 使用率百分比，因为定义阈值时更自然。当定义用作自动扩展组的扩展策略时，我们可以取所需的阈值定义为百分比，并在策略定义中计算所需的阈值（以纳秒为单位）。

value (ns)	粒度(s)	百分比
60000000000	60	100
54000000000	60	90
48000000000	60	80
42000000000	60	70
36000000000	60	60
30000000000	60	50
24000000000	60	40
18000000000	60	30
12000000000	60	20
6000000000	60	10

4.4.2. 以百分比的形式显示实例工作负载 vCPU

您可以使用 **openstack metric aggregates** 命令显示 gnocchi-stored CPU 遥测数据作为百分比而不是实例的 nanosecond 值。

先决条件

- 使用自动扩展组资源创建 heat 堆栈，导致实例工作负载。

流程

1. 以云管理员身份登录到您的 OpenStack 环境。
2. 检索自动扩展组 heat 堆栈的 ID：

```
$ openstack stack show vnf -c id -c stack_status
+--------------+--------------------------------------+
| Field        | Value                                |
+--------------+--------------------------------------+
| id           | e0a15cee-34d1-418a-ac79-74ad07585730 |
| stack_status | CREATE_COMPLETE                      |
+--------------+--------------------------------------+
```

3. 将堆栈 ID 的值设置为环境变量：

```
$ export STACK_ID=$(openstack stack show vnf -c id -f value)
```

4. 按资源类型实例（服务器 ID）将指标返回为聚合，其值计算为百分比。聚合返回为 CPU 时间的纳秒值。我们将该数量除以 1000000000，以获得以秒为单位的值。然后，我们按我们的粒度来划分值，本例中为 60 秒。然后，该值将通过 100 乘以一个百分比。最后，我们将总值除以分配给实例的类别提供的 vCPU 数量，在本示例中，值为 2 vCPU，从而以 CPU 时间百分比表示的值：

```
$ openstack metric aggregates --resource-type instance --sort-column timestamp --sort-
descending '(/ (* (/ (/ (metric cpu rate:mean) 1000000000) 60) 100) 2)'
server_group="$STACK_ID"
+------------------------------------------------+--------------------------+-------------+---------------
-----+
| name                                           | timestamp                | granularity |             value |
+------------------------------------------------+--------------------------+-------------+---------------
-----+
| 61bfb555-9efb-46f1-8559-08dec90f94ed/cpu/rate:mean | 2022-11-07T21:03:00+00:00 |
60.0 |  3.158333333333333 |
| 61bfb555-9efb-46f1-8559-08dec90f94ed/cpu/rate:mean | 2022-11-07T21:02:00+00:00 |
60.0 | 2.6333333333333333 |
| 199b0cb9-6ed6-4410-9073-0fb2e7842b65/cpu/rate:mean | 2022-11-07T21:02:00+00:00 |
60.0 |  2.533333333333333 |
| 61bfb555-9efb-46f1-8559-08dec90f94ed/cpu/rate:mean | 2022-11-07T21:01:00+00:00 |
60.0 |  2.833333333333333 |
| 199b0cb9-6ed6-4410-9073-0fb2e7842b65/cpu/rate:mean | 2022-11-07T21:01:00+00:00 |
60.0 | 3.0833333333333335 |
| 61bfb555-9efb-46f1-8559-08dec90f94ed/cpu/rate:mean | 2022-11-07T21:00:00+00:00 |
60.0 | 13.450000000000001 |
| a95ab818-fbe8-4acd-9f7b-58e24ade6393/cpu/rate:mean | 2022-11-07T21:00:00+00:00 |
60.0 |              2.45 |
| 199b0cb9-6ed6-4410-9073-0fb2e7842b65/cpu/rate:mean | 2022-11-07T21:00:00+00:00 |
60.0 | 2.6166666666666667 |
| 61bfb555-9efb-46f1-8559-08dec90f94ed/cpu/rate:mean | 2022-11-07T20:59:00+00:00 |
60.0 | 60.583333333333336 |
| a95ab818-fbe8-4acd-9f7b-58e24ade6393/cpu/rate:mean | 2022-11-07T20:59:00+00:00 |
60.0 |              2.35 |
| 199b0cb9-6ed6-4410-9073-0fb2e7842b65/cpu/rate:mean | 2022-11-07T20:59:00+00:00 |
```

```
60.0 |              2.525 |
| 61bfb555-9efb-46f1-8559-08dec90f94ed/cpu/rate:mean | 2022-11-07T20:58:00+00:00 |
60.0 |  71.35833333333333 |
| a95ab818-fbe8-4acd-9f7b-58e24ade6393/cpu/rate:mean | 2022-11-07T20:58:00+00:00 |
60.0 |              3.025 |
| 199b0cb9-6ed6-4410-9073-0fb2e7842b65/cpu/rate:mean | 2022-11-07T20:58:00+00:00 |
60.0 |               9.3 |
| 61bfb555-9efb-46f1-8559-08dec90f94ed/cpu/rate:mean | 2022-11-07T20:57:00+00:00 |
60.0 |  66.19166666666668 |
| a95ab818-fbe8-4acd-9f7b-58e24ade6393/cpu/rate:mean | 2022-11-07T20:57:00+00:00 |
60.0 |              2.275 |
| 199b0cb9-6ed6-4410-9073-0fb2e7842b65/cpu/rate:mean | 2022-11-07T20:57:00+00:00 |
60.0 |  56.31666666666667 |
| 61bfb555-9efb-46f1-8559-08dec90f94ed/cpu/rate:mean | 2022-11-07T20:56:00+00:00 |
60.0 |  59.50833333333333 |
| a95ab818-fbe8-4acd-9f7b-58e24ade6393/cpu/rate:mean | 2022-11-07T20:56:00+00:00 |
60.0 |              2.375 |
| 199b0cb9-6ed6-4410-9073-0fb2e7842b65/cpu/rate:mean | 2022-11-07T20:56:00+00:00 |
60.0 | 63.949999999999996 |
| a95ab818-fbe8-4acd-9f7b-58e24ade6393/cpu/rate:mean | 2022-11-07T20:55:00+00:00 |
60.0 | 15.558333333333335 |
| 199b0cb9-6ed6-4410-9073-0fb2e7842b65/cpu/rate:mean | 2022-11-07T20:55:00+00:00 |
60.0 |              93.85 |
| a95ab818-fbe8-4acd-9f7b-58e24ade6393/cpu/rate:mean | 2022-11-07T20:54:00+00:00 |
60.0 |  59.54999999999999 |
| 199b0cb9-6ed6-4410-9073-0fb2e7842b65/cpu/rate:mean | 2022-11-07T20:54:00+00:00 |
60.0 |  61.23333333333334 |
| a95ab818-fbe8-4acd-9f7b-58e24ade6393/cpu/rate:mean | 2022-11-07T20:53:00+00:00 |
60.0 |  74.73333333333333 |
| a95ab818-fbe8-4acd-9f7b-58e24ade6393/cpu/rate:mean | 2022-11-07T20:52:00+00:00 |
60.0 |  57.86666666666667 |
| a95ab818-fbe8-4acd-9f7b-58e24ade6393/cpu/rate:mean | 2022-11-07T20:51:00+00:00 |
60.0 | 60.416666666666664 |
+----------------------------------------------------+--------------------------+------------+--------------
-----+
```

4.4.3. 为实例工作负载检索可用的遥测

检索实例工作负载的可用遥测，并以百分比表示 vCPU 使用率。

先决条件

- 使用自动扩展组资源创建 heat 堆栈，导致实例工作负载。

流程

1. 以云管理员身份登录到您的 OpenStack 环境。
2. 检索自动扩展组 heat 堆栈的 ID：

```
$ openstack stack show vnf -c id -c stack_status
+--------------+--------------------------------------+
| Field        | Value                                |
+--------------+--------------------------------------+
```

```
| id           | e0a15cee-34d1-418a-ac79-74ad07585730 |
| stack_status | CREATE_COMPLETE                      |
+--------------+--------------------------------------+
```

3. 将堆栈 ID 的值设置为环境变量：

```
$ export STACK_ID=$(openstack stack show vnf -c id -f value)
```

4. 检索您要返回数据的工作负载实例的 ID。我们使用服务器列表长表单和过滤属于自动扩展组的实例：

```
$ openstack server list --long --fit-width | grep "metering.server_group='$STACK_ID'"
| bc1811de-48ed-44c1-ae22-c01f36d6cb02 | vn-xlfb4jb-yhbq6fkk2kec-qsu2lr47zigs-vnf-
y27wuo25ce4e | ACTIVE | None       | Running     | private=192.168.100.139, 192.168.25.179
| fedora36   | d21f1aaa-0077-4313-8a46-266c39b705c1 | m1.small    | 692533fe-0912-417e-
b706-5d085449db53 | nova              | host.localdomain | metering.server_group='e0a15cee-
34d1-418a-ac79-74ad07585730' |
```

5. 为返回的实例工作负载名称设置实例 ID：

```
$ INSTANCE_NAME='vn-xlfb4jb-yhbq6fkk2kec-qsu2lr47zigs-vnf-y27wuo25ce4e' ; export
INSTANCE_ID=$(openstack server list --name $INSTANCE_NAME -c ID -f value)
```

6. 验证已经为实例资源 ID 存储了指标。如果没有可用的指标，则自实例创建后，可能没有足够的时间。如果有足够的时间，您可以检查 **/var/log/containers/ceilometer/** 中数据收集服务的日志，并在 **/var/log/containers/gnocchi/** 中检查时间序列数据库服务 gnocchi 的日志：

```
$ openstack metric resource show --column metrics $INSTANCE_ID
+---------+-------------------------------------------------------------------+
| Field   | Value                                                             |
+---------+-------------------------------------------------------------------+
| metrics | compute.instance.booting.time: 57ca241d-764b-4c58-aa32-35760d720b08 |
|         | cpu: d7767d7f-b10c-4124-8893-679b2e5d2ccd                         |
|         | disk.ephemeral.size: 038b11db-0598-4cfd-9f8d-4ba6b725375b         |
|         | disk.root.size: 843f8998-e644-41f6-8635-e7c99e28859e              |
|         | memory.usage: 1e554370-05ac-4107-98d8-9330265db750                |
|         | memory: fbd50c0e-90fa-4ad9-b0df-f7361ceb4e38                      |
|         | vcpus: 0629743e-6baa-4e22-ae93-512dc16bac85                       |
+---------+-------------------------------------------------------------------+
```

7. 验证资源指标是否有可用的测量结果，并记录粒度值，因为我们在运行 **openstack metric aggregates** 命令时使用它：

```
$ openstack metric measures show --resource-id $INSTANCE_ID --aggregation rate:mean
cpu
+---------------------------+-------------+---------------+
| timestamp                 | granularity |         value |
+---------------------------+-------------+---------------+
| 2022-11-08T14:12:00+00:00 |        60.0 | 71920000000.0 |
| 2022-11-08T14:13:00+00:00 |        60.0 | 88920000000.0 |
| 2022-11-08T14:14:00+00:00 |        60.0 | 76130000000.0 |
| 2022-11-08T14:15:00+00:00 |        60.0 | 17640000000.0 |
```

```
| 2022-11-08T14:16:00+00:00 |      60.0 |  3330000000.0 |
| 2022-11-08T14:17:00+00:00 |      60.0 |  2450000000.0 |
...
```

8. 通过查看实例工作负载的配置类别来检索应用到工作负载实例的 vCPU 内核数：

```
$ openstack server show $INSTANCE_ID -cflavor -f value
m1.small (692533fe-0912-417e-b706-5d085449db53)

$ openstack flavor show 692533fe-0912-417e-b706-5d085449db53 -c vcpus -f value
2
```

9. 按资源类型实例（服务器 ID）将指标返回为聚合，其值计算为百分比。聚合返回为 CPU 时间的纳秒值。我们将该数量除以 1000000000，以获得以秒为单位的值。然后，我们按我们的粒度划分值，本例中为 60 秒（之前使用 **openstack metric measures show** 命令检索）。然后，该值将通过 100 乘以一个百分比。最后，我们将总值除以分配给实例的类别提供的 vCPU 数量，在本示例中，值为 2 vCPU，从而以 CPU 时间百分比表示的值：

```
$ openstack metric aggregates --resource-type instance --sort-column timestamp --sort-
descending '(/ (* (/ (/ (metric cpu rate:mean) 1000000000) 60) 100) 2)' id=$INSTANCE_ID
+-------------------------------------------------+---------------------------+-------------+---------------
-----+
| name                                            | timestamp                 | granularity |             value |
+-------------------------------------------------+---------------------------+-------------+---------------
-----+
| bc1811de-48ed-44c1-ae22-c01f36d6cb02/cpu/rate:mean | 2022-11-08T14:26:00+00:00 |
60.0 |              2.45 |
| bc1811de-48ed-44c1-ae22-c01f36d6cb02/cpu/rate:mean | 2022-11-08T14:25:00+00:00 |
60.0 |            11.075 |
| bc1811de-48ed-44c1-ae22-c01f36d6cb02/cpu/rate:mean | 2022-11-08T14:24:00+00:00 |
60.0 |              61.3 |
| bc1811de-48ed-44c1-ae22-c01f36d6cb02/cpu/rate:mean | 2022-11-08T14:23:00+00:00 |
60.0 |  74.78333333333332 |
| bc1811de-48ed-44c1-ae22-c01f36d6cb02/cpu/rate:mean | 2022-11-08T14:22:00+00:00 |
60.0 | 55.383333333333326 |
...
```