Red Hat OpenStack Platform 16.1

实例的自动扩展

在 Red Hat OpenStack Platform 中配置自动扩展

Last Updated: 2023-12-05

Red Hat OpenStack Platform 16.1 实例的自动扩展

在 Red Hat OpenStack Platform 中配置自动扩展

OpenStack Team
rhos-docs@redhat.com

法律通告

摘要

自动扩展您的计算实例以响应系统使用情况。

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 为 COMPUTE 实例配置自动扩展

了解如何根据大量系统使用量自动扩展计算实例。通过使用预定义的规则来考虑 CPU 或内存用量等因素，您可以配置编排(heat)以在需要时自动添加和删除其他实例。

1.1. 自动扩展的架构概述

提供自动扩展的核心组件是 Orchestration (heat)。您可以使用 Orchestration 使用人类可读的 YAML 模板来定义规则。这些规则应用于根据 Telemetry 数据评估系统负载，以了解是否需要在堆栈中添加更多实例。当负载下降时，Orchestration 可以再次自动删除未使用的实例。

使用 Telemetry 监控 Red Hat OpenStack Platform (RHOSP)环境的性能，收集实例和物理主机的 CPU、存储和内存使用率的数据。编配模板检查遥测数据，以评估是否应启动任何预定义的操作。

1.1.1. 主要自动扩展术语

- **stack** - 堆栈代表操作应用所需的所有资源。它可以像单一实例及其资源一样简单，或者像多个实例一样复杂，包含多层应用的所有资源依赖项。
- **模板** - 为要执行的 Heat 定义一系列任务的 YAML 脚本。例如，最好将单独的模板用于某些功能：
  - **模板文件** - 这是您定义 Telemetry 响应并定义自动扩展组的阈值。
  - **环境文件** - 定义环境的构建信息：要使用的类别和镜像、如何配置虚拟网络以及要安装的软件。

1.2. 示例：根据 CPU 使用情况自动扩展

在本例中，编配会检查 Telemetry 数据，并自动增加实例数量，以响应高 CPU 使用。创建堆栈模板和环境模板，以定义规则和后续配置。本例使用现有资源，如网络，并使用与您自己的环境中可能不同的名称。

注意

cpu_util 指标已弃用，并从 Red Hat OpenStack Platform 中删除。

流程

1. 创建环境模板，描述实例类别、网络配置和镜像类型。将模板保存到 **/home/<user>/stacks/example1/cirros.yaml** 文件中。将 & **lt;user** > 变量替换为真实用户名。

```
heat_template_version: 2016-10-14
description: Template to spawn an cirros instance.

parameters:
  metadata:
    type: json
  image:
    type: string
    description: image used to create instance
    default: cirros
  flavor:
    type: string
```

```
    description: instance flavor to be used
    default: m1.tiny
  key_name:
    type: string
    description: keypair to be used
    default: mykeypair
  network:
    type: string
    description: project network to attach instance to
    default: internal1
  external_network:
    type: string
    description: network used for floating IPs
    default: external_network

resources:
  server:
    type: OS::Nova::Server
    properties:
      block_device_mapping:
        - device_name: vda
          delete_on_termination: true
          volume_id: { get_resource: volume }
      flavor: {get_param: flavor}
      key_name: {get_param: key_name}
      metadata: {get_param: metadata}
      networks:
        - port: { get_resource: port }

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

  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 }

  volume:
    type: OS::Cinder::Volume
    properties:
      image: {get_param: image}
      size: 1
```

2. 在 **~/stacks/example1/environment.yaml** 中注册 Orchestration 资源：

```
resource_registry:

    "OS::Nova::Server::Cirros": ~/stacks/example1/cirros.yaml
```

3. 创建堆栈模板。描述要监视的 CPU 阈值以及要添加的实例数量。也会创建一个实例组来定义可参与此模板的最小和最大实例数量。

注意

cpu_util 指标已弃用，并从 Red Hat OpenStack Platform 中删除。要获得等同的功能，请使用累积的 **cpu** 指标和一个包含 **rate:mean** 聚合方法的归档策略。例如，**ceilometer-high-rate** 和 **ceilometer-low-rate**。您必须将阈值从%转换为 ns，以使用 CPU utilisation 警报的 **cpu** 指标。公式为：time_ns = 1,000,000,000 x {granularity} x {percentage_in_decimal}。例如，如果阈值为 80%，粒度为 1s，阈值为 1,000,000,000 x 1 x 0.8 = 800,000,000.0

4. 在 **~/stacks/example1/template.yaml** 中保存以下值：

```
heat_template_version: 2016-10-14
  description: Example auto scale group, policy and alarm
  resources:
    scaleup_group:
      type: OS::Heat::AutoScalingGroup
      properties:
        cooldown: 300
        desired_capacity: 1
        max_size: 3
        min_size: 1
        resource:
          type: OS::Nova::Server::Cirros
          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: 300
        scaling_adjustment: 1

    scaledown_policy:
      type: OS::Heat::ScalingPolicy
      properties:
        adjustment_type: change_in_capacity
        auto_scaling_group_id: { get_resource: scaleup_group }
        cooldown: 300
        scaling_adjustment: -1

    cpu_alarm_high:
      type: OS::Aodh::GnocchiAggregationByResourcesAlarm
      properties:
        description: Scale up if CPU > 80%
        metric: cpu
        aggregation_method: rate:mean
```

```
      granularity: 1
      evaluation_periods: 3
      threshold: 800000000.0
      resource_type: instance
      comparison_operator: gt
      alarm_actions:
        - str_replace:
            template: trust+url
            params:
              url: {get_attr: [scaleup_policy, signal_url]}
      query:
        str_replace:
          template: '{"=": {"server_group": "stack_id"}}'
          params:
            stack_id: {get_param: "OS::stack_id"}

  cpu_alarm_low:
    type: OS::Aodh::GnocchiAggregationByResourcesAlarm
    properties:
      metric: cpu
      aggregation_method: rate:mean
      granularity: 600
      evaluation_periods: 3
      threshold: 200000000.0
      resource_type: instance
      comparison_operator: lt
      alarm_actions:
        - str_replace:
            template: trust+url
            params:
              url: {get_attr: [scaledown_policy, signal_url]}
      query:
        str_replace:
          template: '{"=": {"server_group": "stack_id"}}'
          params:
            stack_id: {get_param: "OS::stack_id"}

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

  scaledown_policy_signal_url:
    value: {get_attr: [scaledown_policy, signal_url]}
```

5. 构建镜像并部署实例：

```
$ openstack stack create  -t template.yaml -e environment.yaml example
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| id                  | 248a98bb-f56e-4934-a281-fffde62d78d8 |
| stack_name          | example                              |
| description         | Example auto scale group, policy and alarm |
| creation_time       | 2017-03-06T15:00:29Z                 |
| updated_time        | None                                 |
```

```
| stack_status        | CREATE_IN_PROGRESS                     |
| stack_status_reason | Stack CREATE started                   |
+---------------------+----------------------------------------+
```

6. 编排创建堆栈，并启动定义的最少 cirros 实例数量，如 **scaleup_group** 定义的 **min_size** 参数中定义。验证实例是否已成功创建：

```
$ openstack server list
+--------------------------------------+-------------------------------------------------------+--------+----
--------+------------+----------------------------------+
| ID                                   | Name                                                  | Status | Task State | Power
State | Networks                         |
+--------------------------------------+-------------------------------------------------------+--------+----
--------+------------+----------------------------------+
| e1524f65-5be6-49e4-8501-e5e5d812c612 | ex-3gax-5f3a4og5cwn2-png47w3u2vjd-
server-vaajhuv4mj3j | ACTIVE | -          | Running     | internal1=10.10.10.9, 192.168.122.8 |
+--------------------------------------+-------------------------------------------------------+--------+----
--------+------------+----------------------------------+
```

7. 编排还创建两个 cpu 警报，它们触发 scale-up 或 scale-down 事件，如 **cpu_ alarm_high** 和 **cpu_ alarm_low** 中定义的。验证触发器是否存在：

```
$ 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 | insufficient data | low      | True    |
| 46ed2c50-e05a-44d8-b6f6-f1ebd83af913 | gnocchi_aggregation_by_resources_threshold |
example-cpu_alarm_low-m37jvnm56x2t  | insufficient data | low      | True    |
+--------------------------------------+----------------------------------------+------------------------
------------+------------------+----------+---------+
```

1.3. 测试自动扩展实例

编排可以基于 **cpu_ alarm_high** 阈值定义自动扩展实例。当 CPU 使用率达到 **threshold** 参数中定义的值时，另一个实例开始平衡负载。**template.yaml** 文件中的 **阈值** 设置为 80%。

流程

1. 登录到实例并运行几个 **dd** 命令来生成负载：

```
$ ssh -i ~/mykey.pem cirros@192.168.122.8
$ sudo dd if=/dev/zero of=/dev/null &
$ sudo dd if=/dev/zero of=/dev/null &
$ sudo dd if=/dev/zero of=/dev/null &
```

2. 运行 **dd** 命令后，您可以预期在 cirros 实例中有 100% CPU 使用。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    |
+--------------------------------------+-------------------------------------------+-------------------------
-----------+-------+----------+---------+
```

3. 大约 60 秒后，编配会启动另一个实例并将它添加到组中。要验证这一点，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   |
+--------------------------------------+-------------------------------------------------------+--------+------
------+-------------+--------------------------------------+
```

4. 在另一个短时间内，观察编排已再次自动缩放到三个实例。该配置设置为最大三个实例，因此无法扩展任何更高的实例(**scaleup_group** 定义： **max_size**)。要验证是否有三个实例，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   |
| 6c88179e-c368-453d-a01a-555eae8cd77a | ex-3gax-fvxz3tr63j4o-36fhftuja3bw-server-
rhl4sqkjuy5p | ACTIVE | -          | Running     | internal1=10.10.10.5, 192.168.122.5   |
+--------------------------------------+-------------------------------------------------------+--------+------
------+-------------+--------------------------------------+
```

1.4. 自动缩减实例

您可以使用 Orchestration 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. 终止正在运行的 **dd** 进程，并观察编排开始缩减实例。

```
$ killall dd
```

2. 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**。

1.5. 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. 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)
```

2. 要读取警报历史记录日志，请输入以下命令：

```
$ 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"}                          |                          |
+-------------------------+-----------------+-------------------------------------------------------------
--------------------------------------+----------------------------------
```

3. 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
```

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

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