Red Hat OpenStack Platform 13

인스턴스의 자동 확장

Red Hat OpenStack Platform에서 자동 확장 구성

Last Updated: 2023-05-25

Red Hat OpenStack Platform 13 인스턴스의 자동 확장

Red Hat OpenStack Platform에서 자동 확장 구성

OpenStack Team
rhos-docs@redhat.com

법적 공지

초록

시스템 사용량에 대응하여 컴퓨팅 인스턴스를 자동으로 확장합니다.

차례

1장. 컴퓨팅 인스턴스의 자동 스케일링 구성

많은 시스템 사용량에 대응하여 컴퓨팅 인스턴스를 자동으로 확장할 수 있습니다. CPU 또는 메모리 사용과 같은 요소를 고려하는 사전 정의된 규칙을 사용하면 필요에 따라 자동으로 추가 인스턴스를 추가하고 제거하도록 오케스트레이션(heat)을 구성할 수 있습니다.

1.1. 자동 스케일링의 아키텍처 개요

1.1.1. 오케스트레이션

자동 확장을 제공하는 핵심 구성 요소는 Orchestration(heat)입니다. 오케스트레이션을 사용하여 사람이 읽을 수 있는 YAML 템플릿을 사용하여 규칙을 정의할 수 있습니다. 이러한 규칙은 Telemetry 데이터를 기반으로 시스템 부하를 평가하여 스택에 더 많은 인스턴스를 추가해야 하는지 확인하도록 적용됩니다. 로드가 삭제되면 오케스트레이션에서 사용되지 않는 인스턴스를 다시 자동으로 제거할 수 있습니다.

1.1.2. telemetry

Telemetry를 사용하여 Red Hat OpenStack Platform 환경의 성능을 모니터링하고, 인스턴스 및 물리적 호스트에 대한 CPU, 스토리지 및 메모리에 대한 데이터를 수집할 수 있습니다. 오케스트레이션 템플릿은 Telemetry 데이터를 검사하여 사전 정의된 작업이 시작되는지 여부를 평가합니다.

1.1.3. 주요 용어

- **스택** - 애플리케이션을 작동하는 데 필요한 모든 리소스를 나타냅니다. 단일 인스턴스 및 해당 리소스만큼 단순하거나 다계층 애플리케이션을 구성하는 모든 리소스 종속 항목이 있는 여러 인스턴스로 복잡할 수 있습니다.
- **템플릿** - heat가 실행할 일련의 작업을 정의하는 YAML 스크립트입니다. 예를 들어 특정 함수에 대해 별도의 템플릿을 사용하는 것이 좋습니다.For example, it is preferred to use separate templates for certain functions:
  - **템플릿 파일** - Telemetry가 응답해야 하는 임계값을 정의하고 자동 확장 그룹을 정의합니다.
  - **환경 파일** - 사용 가능한 플레이버 및 이미지, 가상 네트워크를 구성하는 방법, 설치해야 하는 소프트웨어 등 환경에 대한 빌드 정보를 정의합니다.

1.2. 예: CPU 사용량에 따라 자동 스케일링

이 예제에서 오케스트레이션에서는 원격 분석 데이터를 검사하고 높은 CPU 사용량에 대한 응답으로 인스턴스 수를 자동으로 늘립니다. 필요한 규칙 및 후속 구성을 정의하기 위해 스택 템플릿과 환경 템플릿이 생성됩니다. 이 예제에서는 네트워크와 같은 기존 리소스를 사용하고 자체 환경에서 다를 수 있는 이름을 사용합니다.

1. 인스턴스 플레이버, 네트워킹 구성 및 이미지 유형을 설명하는 환경 템플릿을 생성하여 템플릿 **/home/<user>/stacks/example1/cirros.yaml** 파일에 저장합니다. &lt **;user&gt**; 변수를 실제 사용자 이름으로 바꿉니다.

```
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. **~/stacks/example1/environment.yaml** 에 오케스트레이션 리소스를 등록합니다.

```
resource_registry:

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

3. 조사할 CPU 임계값 및 추가할 인스턴스 수를 설명하는 스택 템플릿을 생성합니다. 또한 이 템플릿에 참여할 수 있는 최소 및 최대 인스턴스 수를 정의하는 인스턴스 그룹도 생성됩니다.

참고

granularity 매개변수는 **gnocchi cpu_util** 지표 단위에 따라 설정해야 합니다. 자세한 내용은 이 솔루션 문서 를 참조하십시오.

~/stacks/example1/template.yaml 에 다음 값을 저장합니다.

```
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_util
      aggregation_method: mean
      granularity: 300
      evaluation_periods: 1
      threshold: 80
      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_util
      aggregation_method: mean
      granularity: 300
      evaluation_periods: 1
      threshold: 5
      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]}
```

4. 다음 OpenStack 명령을 실행하여 환경을 빌드하고 인스턴스를 배포합니다.

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

5. 오케스트레이션에서는 스택을 생성하고 **scaleup_group** 정의의 **min_size** 매개변수에 정의된 대로 정의된 cirros 인스턴스 수를 시작합니다. 인스턴스가 성공적으로 생성되었는지 확인합니다.

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

6. 또한 오케스트레이션에서는 **cpu_alarm_high** 및 **cpu_alarm_low** 에 정의된 대로 확장 또는 축소 이벤트를 트리거하는 데 사용되는 두 개의 cpu 알람을 생성합니다. 트리거가 존재하는지 확인합니다.

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

1.2.1. 자동 확장 인스턴스 테스트

오케스트레이션은 **cpu_alarm_high** 임계값 정의에 따라 자동으로 인스턴스를 확장할 수 있습니다. CPU 사용률이 **임계값** 매개변수에 정의된 값에 도달하면 부하 분산을 위해 다른 인스턴스가 시작됩니다. 위의 **template.yaml** 파일의 **임계값** 이 80%로 설정됩니다.

1. 인스턴스에 로그온하고 여러 **dd** 명령을 실행하여 부하를 생성합니다.

```
$ 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. **dd** 명령을 실행하면 cirros 인스턴스에서 100% CPU 사용률이 있을 것으로 예상할 수 있습니다. 알람이 트리거되었는지 확인합니다.

```
$ 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. 일정 시간(약 60초) 후에 오케스트레이션은 다른 인스턴스를 시작하고 그룹에 추가합니다. **nova list** 명령을 사용하여 이를 확인할 수 있습니다.

```
$ 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. 다른 짧은 기간 후에는 오케스트레이션이 3개의 인스턴스로 다시 자동 확장되었음을 확인할 수 있습니다. 구성은 최대 세 개의 인스턴스로 설정되므로 더 큰 인스턴스(**scaleup_group** 정의: **max_size**)를 확장하지 않습니다. 다시 말하지만, 위에서 언급한 명령을 사용하여 이를 확인할 수 있습니다.

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

1.2.2. 인스턴스 자동 축소

또한 오케스트레이션에서는 **cpu_alarm_low** 임계값을 기반으로 인스턴스를 자동으로 축소할 수 있습니다. 이 예에서는 CPU 사용률이 5% 미만인 경우 인스턴스가 확장됩니다.

1. 실행 중인 **dd** 프로세스를 종료하고 오케스트레이션을 관찰하여 인스턴스를 다시 축소합니다.

```
$ killall dd
```

2. **dd** 프로세스를 중지하면 **cpu_alarm_low 이벤트**가 트리거됩니다. 그 결과 오케스트레이션에서 자동으로 축소를 시작하고 인스턴스를 제거합니다. 해당 알람이 트리거되었는지 확인합니다.

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

5분 후에 오케스트레이션을 통해 인스턴스 수를 **scaleup_group** 정의의 **min_size** 매개변수에 정의된 최소 값으로 지속적으로 줄입니다. 이 시나리오에서는 **min_size** 매개변수가 **1** 로 설정됩니다.

1.2.3. 설정 문제 해결

환경이 제대로 작동하지 않는 경우 로그 파일 및 기록 레코드에서 오류를 찾을 수 있습니다.

1. 상태 전환에 대한 정보를 얻으려면 스택 이벤트 레코드를 나열할 수 있습니다.

```
$ 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. 알람 기록 로그를 읽으려면 다음을 수행합니다.

```
$ 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. 기존 스택에 대해 heat가 수집하는 스케일 아웃 또는 축소 작업의 레코드를 보려면 **awk** 를 사용하여 **heat-engine.log** 를 구문 분석할 수 있습니다.

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

4. aodh 관련 정보 를 보려면 **evaluator.log** 를 확인하십시오.

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