Red Hat OpenStack Platform 17.1

인스턴스의 자동 확장

Red Hat OpenStack Platform에서 자동 확장 구성

Last Updated: 2024-09-11

Red Hat OpenStack Platform 17.1 인스턴스의 자동 확장

Red Hat OpenStack Platform에서 자동 확장 구성

OpenStack Team
rhos-docs@redhat.com

법적 공지

초록

Red Hat OpenStack Platform Telemetry 구성 요소 및 heat 템플릿을 사용하여 워크로드에 대한 인스턴스를 자동으로 시작합니다.

차례

보다 포괄적 수용을 위한 오픈 소스 용어 교체

Red Hat은 코드, 문서, 웹 속성에서 문제가 있는 용어를 교체하기 위해 최선을 다하고 있습니다. 먼저 마스터(master), 슬레이브(slave), 블랙리스트(blacklist), 화이트리스트(whitelist) 등 네 가지 용어를 교체하고 있습니다. 이러한 변경 작업은 작업 범위가 크므로 향후 여러 릴리스에 걸쳐 점차 구현할 예정입니다. 자세한 내용은 CTO Chris Wright의 메시지 를 참조하십시오.

RED HAT 문서에 관한 피드백 제공

문서 개선을 위한 의견을 보내 주십시오. Red Hat이 어떻게 더 나은지 알려주십시오.

Jira에서 문서 피드백 제공

Create Issue 양식을 사용하여 OpenShift (RHOSO) 또는 이전 Red Hat OpenStack Platform (RHOSP)의 Red Hat OpenStack Services 문서에 대한 피드백을 제공합니다. RHOSO 또는 RHOSP 문서에 대한 문제를 생성할 때 RHOSO Jira 프로젝트에 문제가 기록되어 피드백의 진행 상황을 추적할 수 있습니다.

문제 생성 양식을 완료하려면 Jira에 로그인해야 합니다. Red Hat Jira 계정이 없는 경우 https://issues.redhat.com 에서 계정을 생성할 수 있습니다.

1. 다음 링크를 클릭하여 **문제 생성** 페이지를 엽니다.
   https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12336920&summary=Documentation%20feedback:%20%3CAdd%20summary%20here%3E&i
   <Include+the+documentation+URL,+the%20chapter+or+section+number,+and+a+detailed+descrip

2. **요약** 및 **설명** 필드를 작성합니다. **설명** 필드에 문서 URL, 장 또는 섹션 번호, 문제에 대한 자세한 설명을 포함합니다. 양식의 다른 필드를 수정하지 마십시오.

3. **생성**을 클릭합니다.

1장. 자동 스케일링 구성 요소 소개

Telemetry 구성 요소를 사용하여 CPU, 스토리지 및 메모리 사용량과 같은 RHOSP(Red Hat OpenStack Platform) 환경에 대한 데이터를 수집합니다. 워크로드 수요 및 리소스 가용성에 따라 인스턴스를 시작하고 확장할 수 있습니다. 오케스트레이션 서비스(heat) 템플릿의 인스턴스 스케일링을 제어하는 원격 분석 데이터의 상한과 하한을 정의할 수 있습니다.

다음 Telemetry 구성 요소를 사용하여 자동 인스턴스 스케일링을 제어합니다.

- **데이터 수집**: Telemetry는 데이터 수집 서비스(Ceilometer)를 사용하여 메트릭 및 이벤트 데이터를 수집합니다.
- **Storage**: Telemetry는 시계열 데이터베이스 서비스(gnocchi)에 지표 데이터를 저장합니다.
- **Alarm** ing 서비스(aodh)를 사용하여 Ceilometer에서 수집한 메트릭 또는 이벤트 데이터에 대한 규칙에 따라 작업을 트리거합니다.

1.1. 자동 스케일링을 위한 데이터 수집 서비스(CEILOMETER)

Ceilometer를 사용하여 RHOSP(Red Hat OpenStack Platform) 구성 요소의 미터링 및 이벤트 정보에 대한 데이터를 수집할 수 있습니다.

Ceilometer 서비스는 세 개의 에이전트를 사용하여 RHOSP 구성 요소에서 데이터를 수집합니다.

- **컴퓨팅 에이전트(ceilometer-agent-compute):** 각 컴퓨팅 노드에서 실행되며 리소스 사용량 통계를 폴링합니다.
- **중앙 에이전트(ceilometer-agent-central):** 컨트롤러 노드에서 을 실행하여 컴퓨팅 노드에서 제공하지 않는 리소스에 대한 리소스 사용 통계를 폴링합니다.
- **알림 에이전트(ceilometer-agent-notification):** 컨트롤러 노드에서 실행되며 메시지 큐의 메시지를 사용하여 이벤트 및 미터링 데이터를 빌드합니다.

Ceilometer 에이전트는 게시자를 사용하여 데이터를 해당 엔드포인트(예: 시계열 데이터베이스 서비스(gnocchi))로 전송합니다.

추가 리소스

- *오버클라우드 관찰 기능 관리* 가이드의 Ceilometer.

1.1.1. 게시자

RHOSP(Red Hat OpenStack Platform)에서는 여러 전송 방법을 사용하여 수집된 데이터를 STF(Service Telemetry Framework)와 같은 스토리지 또는 외부 시스템으로 전송할 수 있습니다.

gnocchi 게시자를 활성화하면 측정 및 리소스 정보가 시계열 데이터로 저장됩니다.

1.2. 자동 스케일링을 위한 시계열 데이터베이스 서비스(GNOCCHI)

Gnocchi는 SQL에 지표를 저장하는 데 사용할 수 있는 시계열 데이터베이스입니다. 알람 서비스(aodh) 및 Orchestration 서비스(heat)는 자동 스케일링을 위해 gnocchi에 저장된 데이터를 사용합니다.

추가 리소스

- Gnocchi가 있는 스토리지입니다.

1.3. 알람 서비스(AODH)

Ceilometer에서 수집하고 gnocchi에 저장된 메트릭 데이터에 대한 규칙을 기반으로 작업을 트리거하도록 알람 서비스(aodh)를 구성할 수 있습니다. 경고는 다음 상태 중 하나일 수 있습니다.

- **OK**: 메트릭 또는 이벤트가 허용 가능한 상태입니다.
- **firing**: 메트릭 또는 이벤트가 정의된 **Ok** 상태 외부에 있습니다.
- **충분하지 않은 데이터**: 예를 들어 요청된 단위에 대한 데이터가 없거나 검사가 실행되지 않은 경우와 같이 경보 상태를 알 수 없습니다.

1.4. 자동 스케일링을 위한 오케스트레이션 서비스(HEAT)

director는 오케스트레이션 서비스(heat) 템플릿을 오버클라우드 배포의 템플릿 형식으로 사용합니다. 일반적으로 Heat 템플릿은 YAML 형식으로 표시됩니다. 템플릿의 목적은 heat가 생성하는 리소스 컬렉션 및 리소스 구성인 스택을 정의하고 생성하는 것입니다. 리소스는 RHOSP(Red Hat OpenStack Platform)의 개체이며 컴퓨팅 리소스, 네트워크 구성, 보안 그룹, 확장 규칙 및 사용자 지정 리소스를 포함할 수 있습니다.

추가 리소스

- heat 템플릿 이해

2장. 자동 스케일링을 위해 오버클라우드 구성 및 배포

자동 스케일링을 활성화하는 오버클라우드의 서비스에 대한 템플릿을 구성해야 합니다.

프로세스

1. 자동 스케일링을 위해 오버클라우드를 배포하기 전에 자동 스케일링 서비스를 위한 환경 템플릿과 리소스 레지스트리를 생성합니다. 자세한 내용은 다음을 참조하십시오. 2.1절. "자동 스케일링을 위해 오버클라우드 설정"

2. 오버클라우드를 배포합니다. 자세한 내용은 다음을 참조하십시오. 2.2절. "자동 스케일링을 위해 오버클라우드 배포"

2.1. 자동 스케일링을 위해 오버클라우드 설정

자동 스케일링을 제공하는 서비스를 배포하는 데 필요한 환경 템플릿 및 리소스 레지스트리를 생성합니다.

프로세스

1. 오버클라우드 관리자 인증 정보를 사용하여 언더클라우드 호스트에 로그인합니다(예: **overcloudrc**).

2. 자동 스케일링 구성 파일을 위한 디렉터리를 만듭니다.

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

3. 서비스를 자동 스케일링하는 데 필요한 정의에 대한 리소스 레지스트리 파일을 생성합니다.

```
$ 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-pacemaker-puppet.yaml
EOF
```

4. 환경 템플릿을 생성하여 자동 확장에 필요한 서비스를 구성합니다.

```
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
```

Red Hat Ceph Storage를 시계열 데이터베이스 서비스의 데이터 스토리지 백엔드로 사용하는 경우 **매개변수-autoscaling.yaml 파일에 다음 매개변수를 추가합니다.**

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

메트릭을 저장하려면 먼저 정의된 아카이브 정책 **일반** 을 생성해야 합니다. 배포 후 이 아카이브 정책을 정의합니다. 자세한 내용은 3.1절. "자동 스케일링을 위한 일반 아카이브 정책 생성" 의 내용을 참조하십시오.

5. poll **_interval** 매개변수를 설정합니다(예: 60초). poll **_interval 매개변수** 의 값은 아카이브 정책을 생성할 때 정의한 gnocchi 세분화 값과 일치해야 합니다. 자세한 내용은 3.1절. "자동 스케일링을 위한 일반 아카이브 정책 생성"의 내용을 참조하십시오.

6. 오버클라우드를 배포합니다. 자세한 내용은 다음을 참조하십시오. 2.2절. "자동 스케일링을 위해 오버클라우드 배포"

2.2. 자동 스케일링을 위해 오버클라우드 배포

director를 사용하여 자동 스케일링할 오버클라우드를 배포할 수 있습니다.

사전 요구 사항

- 자동 스케일링 기능을 제공하는 서비스를 배포하기 위한 환경 템플릿을 생성했습니다. 자세한 내용은 2.1절. "자동 스케일링을 위해 오버클라우드 설정" 의 내용을 참조하십시오.

프로세스

- 2.2.1절. "director를 사용하여 자동 스케일링을 위해 오버클라우드 배포"

2.2.1. director를 사용하여 자동 스케일링을 위해 오버클라우드 배포

director를 사용하여 오버클라우드를 배포합니다.

사전 요구 사항

- 배포된 언더클라우드. 자세한 내용은 언더클라우드에 director 설치를 참조하십시오.

프로세스

1. **stack** 사용자로 언더클라우드에 로그인합니다.
2. **stackrc** 언더클라우드 인증 정보 파일을 소싱합니다.

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

3. 다른 환경 파일과 함께 스택에 자동 스케일링 환경 파일을 추가하고 오버클라우드를 배포합니다.

```
(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. 자동 스케일링을 위한 오버클라우드 배포 확인

자동 스케일링 서비스가 배포되고 활성화되어 있는지 확인합니다. 확인 출력 예제는 사용 사례와 다를 수 있습니다.

사전 요구 사항

- director를 사용하여 기존 오버클라우드에 자동 스케일링 서비스를 배포했습니다. 자세한 내용은 2.2절. "자동 스케일링을 위해 오버클라우드 배포"의 내용을 참조하십시오.

프로세스

1. **stack** 사용자로 환경에 로그인합니다.
2. director 환경의 경우 **stackrc** 언더클라우드 인증 정보 파일을 소싱합니다.

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

검증

1. 배포가 성공했는지 확인하고 자동 스케일링을 위해 서비스 API 끝점을 사용할 수 있는지 확인합니다.

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

오버클라우드 인증 정보를 사용하여 heat 서비스를 확인합니다.

```
(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. 서비스가 오버클라우드에서 실행 중인지 확인합니다.

```
$ 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. 시계열 데이터베이스 서비스를 사용할 수 있는지 확인합니다.

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

3장. 자동 스케일링에 HEAT 서비스 사용

오버클라우드에서 자동 스케일링을 제공하는 데 필요한 서비스를 배포한 후 오케스트레이션 서비스(heat)에서 자동 스케일링을 위해 인스턴스를 관리할 수 있도록 오버클라우드 환경을 구성해야 합니다.

사전 요구 사항

- 배포된 오버클라우드입니다. 자세한 내용은 2.2절. "자동 스케일링을 위해 오버클라우드 배포"의 내용을 참조하십시오.

프로세스

- 3.1절. "자동 스케일링을 위한 일반 아카이브 정책 생성"
- 3.2절. "자동으로 인스턴스 확장을 위한 heat 템플릿 구성"
- 3.3절. "자동 스케일링을 위한 스택 배포 생성"

3.1. 자동 스케일링을 위한 일반 아카이브 정책 생성

오버클라우드에서 자동 스케일링을 위해 서비스를 배포한 후 오케스트레이션 서비스(heat)에서 자동 스케일링을 위해 인스턴스를 관리할 수 있도록 오버클라우드 환경을 구성해야 합니다.

사전 요구 사항

- 자동 확장 서비스가 있는 오버클라우드를 배포했습니다. 자세한 내용은 2.1절. "자동 스케일링을 위해 오버클라우드 설정"의 내용을 참조하십시오.

프로세스

1. **stack** 사용자로 환경에 로그인합니다.
2. director 환경의 경우 **stackrc** 파일을 소싱합니다.

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

3. **$HOME/templates/autoscaling/parameters-autoscaling.yaml** 에 정의된 아카이브 정책을 생성합니다.

```
$ 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'
```

검증

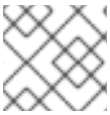
- 보관 정책이 생성되었는지 확인합니다.

```
$ 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. 자동으로 인스턴스 확장을 위한 HEAT 템플릿 구성

인스턴스를 생성하고 트리거할 때 인스턴스를 생성하고 확장하는 알람을 구성하도록 오케스트레이션 서비스(heat) 템플릿을 구성할 수 있습니다.



참고

이 절차에서는 환경에 맞게 변경해야 하는 예제 값을 사용합니다.

사전 요구 사항

- 자동 스케일링 서비스를 사용하여 오버클라우드를 배포했습니다. 자세한 내용은 2.2절. "자동 스케일링을 위해 오버클라우드 배포"의 내용을 참조하십시오.
- 원격 분석 스토리지를 자동 스케일링하기 위한 아카이브 정책으로 오버클라우드를 구성했습니다. 자세한 내용은 3.1절. "자동 스케일링을 위한 일반 아카이브 정책 생성" 의 내용을 참조하십시오.

프로세스

1. **stack** 사용자로 환경에 로그인합니다.

```
$ source ~/stackrc
```

2. 자동 스케일링 그룹에 대한 인스턴스 구성을 저장할 디렉터리를 만듭니다.

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

3. 인스턴스 구성 템플릿을 생성합니다(예: **$HOME/templates/autoscaling/vnf/instance.yaml**).
4. **instance.yaml** 파일에 다음 구성을 추가합니다.

```
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** 매개변수는 이 새 리소스에 대한 사용자 지정 매개 변수를 정의합니다.
- **resources** 매개변수의 **vnf** 하위 매개변수는 **OS::Heat::AutoScalingGroup** 에서 참조하는 사용자 지정 하위 리소스의 이름을 정의합니다(예: **OS::Nova::Server::VNF**).

5. heat 템플릿에서 참조할 리소스를 생성합니다.

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

6. heat에 대한 배포 템플릿을 생성하여 인스턴스 스케일링을 제어합니다.

```
$ 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
```

참고

스택의 출력은 정보이며 ScalingPolicy 또는 AutoScalingGroup에서 참조되지 않습니다. 출력을 보려면 **openstack stack show <stack_name> 명령을** 사용합니다.

3.3. 자동 스케일링을 위한 스택 배포 생성

작동된 VNF 자동 스케일링 예에 대한 스택 배포를 생성합니다.

사전 요구 사항

- 3.2절. "자동으로 인스턴스 확장을 위한 heat 템플릿 구성"

프로세스

1. 오버클라우드 관리자 인증 정보를 사용하여 언더클라우드 호스트에 로그인합니다. 예를 들면 **overcloudrc**:

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

2. 스택을 생성합니다.

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

검증

1. 스택이 생성되었는지 확인합니다.

```
$ openstack stack show vnf -c id -c stack_status
+--------------+--------------------------------------+
| Field        | Value                                |
+--------------+--------------------------------------+
| id           | cb082cbd-535e-4779-84b0-98925e103f5e |
| stack_status | CREATE_COMPLETE                      |
+--------------+--------------------------------------+
```

2. 알람, 확장 정책, 자동 스케일링 그룹을 포함하여 스택 리소스가 생성되었는지 확인합니다.

```
$ 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. 스택 생성을 통해 인스턴스가 시작되었는지 확인합니다.

```
$ 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. 스택에 대한 알람이 생성되었는지 확인합니다.

   a. 알람 ID를 나열합니다. 알람 상태는 일정 기간 동안 **충분하지 않은 데이터** 상태에 있을 수 있습니다. 최소 시간은 데이터 수집 및 데이터 스토리지 세분성 설정의 폴링 간격입니다.

```
$ 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. 스택의 리소스를 나열하고 **cpu_alarm_high** 및 **cpu_alarm_low** 리소스의 **physical_resource_id** 값을 기록해 둡니다.

```
$ 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 값은 **openstack alarm list** 명령의 출력에 있는 **alarm_id** 와 일치해야 합니다.

5. 스택에 대한 지표 리소스가 있는지 확인합니다. **server_group** 쿼리의 값을 스택 ID로 설정합니다.

```
$ 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. 스택을 통해 생성된 인스턴스 리소스에 대한 측정이 있는지 확인합니다.

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

4장. 자동 스케일링 테스트 및 문제 해결

오케스트레이션 서비스(heat)를 사용하여 임계값 정의에 따라 인스턴스를 자동으로 확장 및 축소합니다. 환경의 문제를 해결하려면 로그 파일 및 기록 레코드에서 오류를 찾을 수 있습니다.

4.1. 인스턴스 자동 확장 테스트

Orchestration 서비스(heat)를 사용하여 **cpu_alarm_high** 임계값 정의에 따라 자동으로 인스턴스를 확장할 수 있습니다. CPU 사용량이 **threshold** 매개변수에 정의된 값에 도달하면 로드의 균형을 조정하기 위해 다른 인스턴스가 시작됩니다. **template.yaml** 파일의 **임계값** 은 80%로 설정됩니다.

프로세스

1. **stack** 사용자로 호스트 환경에 로그인합니다.
2. director 환경의 경우 **stackrc** 파일을 소싱합니다.

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

3. 인스턴스에 로그인합니다.

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

4. 여러 **dd** 명령을 실행하여 부하를 생성합니다.

```
[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. 실행 중인 인스턴스를 종료하고 호스트로 돌아갑니다.
6. **dd** 명령을 실행한 후 인스턴스에서 100% CPU 사용을 기대할 수 있습니다. 경고가 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    |
+--------------------------------------+----------------------------------------+-------------------------
----------+-------+----------+---------+
```

7. 약 60초 후에 오케스트레이션에서 다른 인스턴스를 시작하여 그룹에 추가합니다. 인스턴스가 생성되었는지 확인하려면 다음 명령을 입력합니다.

```
$ 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. 또 다른 짧은 기간 후에 오케스트레이션 서비스가 세 개의 인스턴스로 자동 스케일링되었는지 확인합니다. 구성은 최대 3개의 인스턴스로 설정됩니다. 세 개의 인스턴스가 있는지 확인합니다.

```
$ 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. 인스턴스 자동 확장 테스트

Orchestration 서비스(heat)를 사용하여 **cpu_alarm_low** 임계값에 따라 인스턴스를 자동으로 축소할 수 있습니다. 이 예에서는 CPU 사용량이 5% 미만이면 인스턴스가 축소됩니다.

프로세스

1. 워크로드 인스턴스 내에서 실행 중인 **dd** 프로세스를 종료하고 오케스트레이션이 인스턴스를 다시 축소하기 시작하는 것을 관찰합니다.

```
$ killall dd
```

2. **stack** 사용자로 호스트 환경에 로그인합니다.

3. director 환경의 경우 **stackrc** 파일을 소싱합니다.

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

4. **dd** 프로세스를 중지하면 **cpu_alarm_low 이벤트** 경고가 트리거됩니다. 결과적으로 오케스트레이션은 자동으로 축소되고 인스턴스를 제거하기 시작합니다. 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    |
+--------------------------------------+----------------------------------------+-----------------------
----------+-------+----------+---------+
```

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

4.3. 자동 스케일링 문제 해결

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

프로세스

1. **stack** 사용자로 호스트 환경에 로그인합니다.
2. director 환경의 경우 **stackrc** 파일을 소싱합니다.

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

3. 상태 전환에 대한 정보를 검색하려면 스택 이벤트 레코드를 나열합니다.

```
$ 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. 알람 기록 로그를 읽습니다.

```
$ 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. heat가 기존 스택에 대해 수집하는 스케일 아웃 또는 스케일 다운 작업의 레코드를 보려면 **awk** 명령을 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
```

6. aodh 관련 정보를 보려면 **aodh-evaluator.log** 를 검사합니다.

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

4.4. RATE:MEAN AGGREGRATION을 사용할 때 자동 스케일링 임계값에 CPU TELEMETRY 값 사용

OS::Heat::Autoscaling heat 오케스트레이션 템플릿(HOT)을 사용하고 CPU에 대한 임계값을 설정하는 경우 값은 인스턴스 워크로드에 할당된 가상 CPU 수에 따라 동적 값인 CPU 시간 나노초로 표시됩니다. 이 참조 가이드에서는 Gnocchi **rate:mean** aggregration 방법을 사용할 때 CPU 나노초 값을 백분율로 계산하고 표시하는 방법을 살펴보겠습니다.

4.4.1. CPU Telemetry 값을 백분율로 계산

CPU Telemetry는 Gnocchi(OpenStack 시계열 데이터 저장소)에 나노초 단위의 CPU 사용률으로 저장됩니다. CPU Telemetry를 사용하여 자동 스케일링 임계값을 정의할 때 임계값을 정의할 때 더 자연스럽기 때문에 값을 CPU 사용률의 백분율로 표시하는 것이 유용합니다. 자동 스케일링 그룹의 일부로 사용되는 확장 정책을 정의할 때 백분율로 정의된 원하는 임계값을 사용하고 정책 정의에 사용되는 나노초 단위로 필요한 임계값을 계산할 수 있습니다.

값(ns)	세분성(s)	백분율
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. 인스턴스 워크로드 vCPU를 백분율로 표시

openstack metric aggregates 명령을 사용하여 gnocchi-stored CPU Telemetry 데이터를 인스턴스의 나노초 값이 아닌 백분율로 표시할 수 있습니다.

사전 요구 사항

- 인스턴스 워크로드를 생성하는 자동 스케일링 그룹 리소스를 사용하여 heat 스택을 생성합니다.

프로세스

1. OpenStack 환경에 클라우드 관리자로 로그인합니다.
2. 자동 스케일링 그룹 heat 스택의 ID를 검색합니다.

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

3. 스택 ID의 값을 환경 변수로 설정합니다.

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

4. 백분율로 계산된 값을 사용하여 리소스 유형 인스턴스(서버 ID)에 따른 집계로 메트릭을 반환합니다. 집계는 CPU 시간 나노초 값으로 반환됩니다. 이 수를 1000000000으로 나누어 값을 초 단위로 나눕니다. 그런 다음 값을 세분성으로 나눕니다. 이 예에서는 60초입니다. 그런 다음 이 값은 100을 곱하여 백분율로 변환됩니다. 마지막으로, 총 값을 인스턴스에 할당된 플레이버에서 제공하는 vCPU 수로 나눕니다. 이 예제에서는 2 vCPU 값을 제공하여 CPU 시간의 백분율로 표시됩니다.

```
$ 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. 인스턴스 워크로드에 사용 가능한 Telemetry 검색

인스턴스 워크로드에 사용 가능한 Telemetry를 검색하고 vCPU 사용률을 백분율로 표현합니다.

사전 요구 사항

- 인스턴스 워크로드를 생성하는 자동 스케일링 그룹 리소스를 사용하여 heat 스택을 생성합니다.

프로세스

1. OpenStack 환경에 클라우드 관리자로 로그인합니다.

2. 자동 스케일링 그룹 heat 스택의 ID를 검색합니다.

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

3. 스택 ID의 값을 환경 변수로 설정합니다.

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

4. 데이터를 반환할 워크로드 인스턴스의 ID를 검색합니다. 서버 목록을 긴 양식으로 사용하고 자동 스케일링 그룹의 일부인 인스턴스를 필터링합니다.

```
$ 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. 반환된 인스턴스 워크로드 이름 중 하나에 대한 인스턴스 ID를 설정합니다.

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

6. 인스턴스 리소스 ID에 대한 지표가 저장되었는지 확인합니다. 사용 가능한 메트릭이 없는 경우 인스턴스가 생성된 이후 시간이 경과할 수 없습니다. 충분한 시간이 경과한 경우 **/var/log/containers/ceilometer/** 에서 데이터 수집 서비스의 로그를 확인하고 **/var/log/containers/gnocchi/**:에서 시계열 데이터베이스 서비스 gnocchi에 대한 로그를 확인할 수 있습니다.

```
$ 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. **openstack metric aggregates** 명령을 실행할 때 리소스 메트릭에 사용할 수 있는 측정값이 있는지 확인하고 세분 값을 기록해 둡니다.

```
$ 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. 인스턴스 워크로드에 대해 구성된 플레이버를 검토하여 워크로드 인스턴스에 적용되는 vCPU 코어 수를 검색합니다.

```
$ 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. 백분율로 계산된 값을 사용하여 리소스 유형 인스턴스(서버 ID)에 따른 집계로 메트릭을 반환합니다. 집계는 CPU 시간 나노초 값으로 반환됩니다. 이 수를 1000000000으로 나누어 값을 초 단위로 나눕니다. 그런 다음 값을 단위로 나눕니다. 이 예에서는 이전에 **openstack metric measures show** 명령을 사용하여 검색한 대로 60초입니다. 그런 다음 이 값은 100을 곱하여 백분율로 변환됩니다. 마지막으로, 총 값을 인스턴스에 할당된 플레이버에서 제공하는 vCPU 수로 나눕니다. 이 예제에서는 2 vCPU 값을 제공하여 CPU 시간의 백분율로 표시됩니다.

```
$ 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 |
...
```