Red Hat OpenStack Platform 17.0

インスタンスの自動スケーリング

Red Hat OpenStack Platform での自動スケーリングの設定

Last Updated: 2023-09-20

Red Hat OpenStack Platform 17.0 インスタンスの自動スケーリング

Red Hat OpenStack Platform での自動スケーリングの設定

OpenStack Team
rhos-docs@redhat.com

法律上の通知

概要

Red Hat OpenStack Platform のテレメトリーコンポーネントとヒートテンプレートを使用して、ワークロードのインスタンスを自動的に起動します。

目次

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、Red Hat CTO である Chris Wright のメッセージ を参照してください。

RED HAT ドキュメントへのフィードバック (英語のみ)

Red Hat ドキュメントに対するご意見をお聞かせください。ドキュメントの改善点があればお知らせください。

ドキュメントへのダイレクトフィードバック (DDF) 機能の使用 (英語版のみ)

特定の文章、段落、またはコードブロックに対して直接コメントを送付するには、DDF の **Add Feedback** 機能を使用してください。なお、この機能は英語版のドキュメントでのみご利用いただけます。

1. **Multi-page HTML** 形式でドキュメントを表示します。
2. ドキュメントの右上隅に **Feedback** ボタンが表示されていることを確認してください。
3. コメントするテキスト部分をハイライト表示します。
4. **Add Feedback** をクリックします。
5. **Add Feedback** フィールドにコメントを入力します。
6. オプション: ドキュメントチームが問題の詳細を確認する際に使用できるメールアドレスを記入してください。
7. **Submit** をクリックします。

第1章 自動スケーリングコンポーネントの概要

テレメトリーコンポーネントを使用して、CPU、ストレージ、メモリー使用量など、Red Hat OpenStack Platform (RHOSP) 環境に関するデータを収集します。ワークロードの需要とリソースの可用性に応じて、インスタンスを起動およびスケーリングできます。オーケストレーションサービス (ヒート) テンプレートでインスタンスのスケーリングを制御するテレメトリーデータの上限と下限を定義できます。

次のテレメトリーコンポーネントを使用して、インスタンスの自動スケーリングを制御します。

- **データ収集**: テレメトリーは、データ収集サービス (Ceilometer) を使用して、メトリクスとイベントデータを収集します。
- **ストレージ**: テレメトリーは、メトリクスデータを時系列データベースサービス (gnocchi) に保存します。
- **アラーム**: テレメトリーは、Alarming サービス (aodh) を使用して、Ceilometer によって収集されたメトリクスまたはイベントデータに対するルールに基づき、アクションをトリガーします。

1.1. 自動スケーリング用のデータ収集サービス (CEILOMETER)

Ceilometer を使用して、Red Hat OpenStack Platform (RHOSP) コンポーネントの測定およびイベント情報に関するデータを収集できます。

Ceilometer サービスは、3 つのエージェントを使用して、RHOSP コンポーネントからデータを収集します。

- **コンピュートエージェント (ceilometer-agent-compute)**: 各コンピュートノードで実行され、リソース使用統計をポーリングします。
- **中央エージェント (ceilometer-agent-central)**: コントローラーノードで実行され、コンピュートノードによって提供されないリソースのリソース使用統計をポーリングします。
- **通知エージェント (ceilometer-agent-notification)**: コントローラーノードで実行され、メッセージキューからのメッセージを使用して、イベントおよび測定データを構築します。

Ceilometer エージェントは、パブリッシャーを使用して、時系列データベースサービス (gnocchi) などの対応エンドポイントにデータを送信します。

関連情報

- **運用測定** ガイドの Ceilometer。

1.1.1. パブリッシャー

Red Hat OpenStack Platform (RHOSP) では、複数の転送方法を使用して、収集したデータをストレージまたは Service Telemetry Framework (STF) などの外部システムに転送できます。

gnocchi パブリッシャーを有効にすると、測定およびリソース情報が時系列データとして保存されます。

1.2. 自動スケーリング用の時系列データベースサービス (GNOCCHI)

Gnocchi は、SQL でメトリクスを保存するために使用できる時系列データベースです。Alarming サービス (aodh) と Orchestration サービス (heat) は、自動スケーリングのために、gnocchi に保存されたデータを使用します。

関連情報

- gnocchi を使用したストレージ。

1.3. ALARMING サービス (AODH)

Ceilometer によって収集され、gnocchi に保存されたメトリクスデータに対するルールに基づいて、アクションをトリガーするように、Alarming サービス (aodh) を設定できます。アラームは以下の状態のいずれかになります。

- **Ok**: メトリクスまたはイベントは、許容可能な状態です。
- **Firing**: メトリクスまたはイベントは、定義された **Ok** 状態から外れています。
- **insufficient data**: アラームの状態は不明です。たとえば、要求された粒度のデータがない場合や、チェックがまだ実行されていない場合などです。

1.4. 自動スケーリング用のオーケストレーションサービス (HEAT)

Director は、オーケストレーションサービス (heat) テンプレートをオーバークラウドデプロイメントのテンプレート形式として使用します。通常、heat テンプレートは YAML 形式で表現されます。テンプレートの目的は、heat が作成するリソースのコレクションであるスタックを定義および作成し、リソースを設定することです。リソースとは、コンピュートリソース、ネットワーク設定、セキュリティーグループ、スケーリングルール、カスタムリソースなどの Red Hat OpenStack Platform (RHOSP) のオブジェクトを指します。

関連情報

- heat テンプレートの概要

第2章 自動スケーリングのためのオーバークラウドの設定とデプロイ

自動スケーリングを有効にするオーバークラウド上のサービスのテンプレートを設定する必要があります。

手順

1. 自動スケーリング用にオーバークラウドをデプロイする前に、自動スケーリングサービス用の環境テンプレートとリソースレジストリーを作成します。詳細については、「自動スケーリングのためのオーバークラウドの設定」 を参照してください。
2. オーバークラウドをデプロイする。詳細は、「自動スケーリングのためのオーバークラウドのデプロイ」 を参照してください。

2.1. 自動スケーリングのためのオーバークラウドの設定

自動スケーリングを提供するサービスをデプロイするために必要な環境テンプレートとリソースレジストリーを作成します。

手順

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. 自動スケーリング設定ファイル用のディレクトリーを作成します。

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

3. 自動スケーリングのためにサービスが必要とする定義のリソースレジストリーファイルを作成します。

```
$ 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::HeatApiCfn: /usr/share/openstack-tripleo-heat-
templates/deployment/heat/heat-api-cfn-container-puppet.yaml
  OS::TripleO::Services::HeatApiCloudwatch: /usr/share/openstack-tripleo-heat-
templates/deployment/heat/heat-api-cloudwatch-disabled-puppet.yaml
  OS::TripleO::Services::HeatEngine: /usr/share/openstack-tripleo-heat-
templates/deployment/heat/heat-engine-container-puppet.yaml
  OS::TripleO::Services::Redis: /usr/share/openstack-tripleo-heat-
templates/deployment/database/redis-container-puppet.yaml
EOF
```

4. 自動スケーリングに必要なサービスを設定する環境テンプレートを作成します。

```
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 Storage を時系列データベースサービスのデータストレージバックエンドとして使用する場合は、以下のパラメーターを **parameters-autoscaling.yaml** ファイルに追加します。

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

メトリクスを保存する前に、定義済みのアーカイブポリシー **generic** を作成する必要があります。このアーカイブポリシーは、デプロイ後に定義します。詳細は、「自動スケーリング用の汎用アーカイブポリシーの作成」 を参照してください。

5. **polling_interval** パラメーターを、たとえば 60 秒に設定します。**polling_interval** パラメーターの値は、アーカイブポリシーの作成時に定義した gnocchi 粒度の値に一致する必要があります。詳細は、「自動スケーリング用の汎用アーカイブポリシーの作成」 を参照してください。

6. オーバークラウドをデプロイする。詳細は、「自動スケーリングのためのオーバークラウドのデプロイ」 を参照してください。

2.2. 自動スケーリングのためのオーバークラウドのデプロイ

Director を使用するか、スタンドアロン環境を使用して、自動スケーリング用にオーバークラウドをデプロイできます。

前提条件

- 自動スケーリング機能を提供するサービスをデプロイするための環境テンプレートを作成している。詳細は、「自動スケーリングのためのオーバークラウドの設定」 を参照してください。

手順

- 「Director を使用して自動スケーリング用のオーバークラウドをデプロイする」
- 「スタンドアロン環境で自動スケーリング用のオーバークラウドをデプロイする」

2.2.1. Director を使用して自動スケーリング用のオーバークラウドをデプロイする

Director を使用して、オーバークラウドをデプロイします。スタンドアロン環境を使用している場合は、「スタンドアロン環境で自動スケーリング用のオーバークラウドをデプロイする」 を参照してください。

前提条件

- デプロイされたアンダークラウド。詳細は、アンダークラウドに director をインストールする を参照してください。

手順

1. アンダークラウドに **stack** ユーザーとしてログインします。
2. **stackrc** アンダークラウド認証情報ファイルを入手します。

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

3. 自動スケーリング環境ファイルを他の環境ファイルとともにスタックに追加し、オーバークラウドをデプロイします。

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

2.2.2. スタンドアロン環境で自動スケーリング用のオーバークラウドをデプロイする

実稼働前環境で環境ファイルをテストするには、スタンドアロンデプロイメントを使用して、自動スケーリングに必要なサービスを含むオーバークラウドをデプロイします。

注記

この手順では、実稼働環境に合わせて変更する必要がある値とコマンドの例を使用します。

Director を使用して、自動スケーリング用のオーバークラウドをデプロイする場合は、「Director を使用して自動スケーリング用のオーバークラウドをデプロイする」 を参照してください。

前提条件

- オールインワンの RHOSP 環境が python3-tripleoclient でステージングされている。詳細は、オールインワンの Red Hat OpenStack Platform 環境のインストール を参照してください。
- オールインワンの RHOSP 環境が、基本設定でステージングされている。詳細は、オールインワンの Red Hat OpenStack Platform 環境の設定 を参照してください。

手順

1. オーバークラウドのデプロイメントを管理するユーザー (例: **stack** ユーザー) に変更します。

```
[root@standalone ~]# su - stack
```

2. オーバークラウドデプロイメントの環境変数 **$IP**、**$NETMASK**、および **$VIP** を置換または設定します。

```
$ export IP=192.168.25.2
$ export VIP=192.168.25.3
$ export NETMASK=24
```

3. オーバークラウドをデプロイして、リソースとパラメーターファイルをテストおよび検証します。

```
$ sudo openstack tripleo deploy \
  --templates \
  --local-ip=$IP/$NETMASK \
  --control-virtual-ip=$VIP \
  -e /usr/share/openstack-tripleo-heat-templates/environments/standalone/standalone-tripleo.yaml \
  -r /usr/share/openstack-tripleo-heat-templates/roles/Standalone.yaml \
  -e $HOME/containers-prepare-parameters.yaml \
  -e $HOME/standalone_parameters.yaml \
  -e $HOME/templates/autoscaling/resources-autoscaling.yaml \
  -e $HOME/templates/autoscaling/parameters-autoscaling.yaml \
  --output-dir $HOME \
  --standalone
```

4. **OS_CLOUD** 環境変数をエクスポートします。

```
$ export OS_CLOUD=standalone
```

関連情報

- Director のインストールと使用 ガイド。
- スタンドアロン導入 ガイド。

2.3. 自動スケーリングのためのオーバークラウドデプロイメントの検証

自動スケーリングサービスがデプロイされ、有効になっていることを確認します。検証の出力はスタンドアロン環境からのものですが、Director ベースの環境でも同様の出力が得られます。

前提条件

- スタンドアロンまたは Director を使用して、既存のオーバークラウドに自動スケーリングサービスをデプロイしている。詳細は、「自動スケーリングのためのオーバークラウドのデプロイ」 を参照してください。

手順

1. **stack** ユーザーとして環境にログインします。
2. スタンドアロン環境の場合は、**OS_CLOUD** 環境変数を設定します。

```
[stack@standalone ~]$ export OS_CLOUD=standalone
```

3. Director 環境の場合は、**stackrc** アンダークラウド認証情報ファイルを入手します。

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

検証

1. デプロイが成功したことを確認し、自動スケーリング用のサービス API エンドポイントが利用可能であることを確認します。

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

```
$ 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. サービスがオーバークラウドで実行されていることを確認します。

```
$ 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:17.0              kolla_start
27 minutes ago  Up 27 minutes ago (healthy)             aodh_api
77acf3487736  registry.redhat.io/rhosp-rhel9/openstack-aodh-listener:17.0          kolla_start
27 minutes ago  Up 27 minutes ago (healthy)             aodh_listener
29ec47b69799  registry.redhat.io/rhosp-rhel9/openstack-aodh-evaluator:17.0
kolla_start  27 minutes ago  Up 27 minutes ago (healthy)             aodh_evaluator
43efaa86c769  registry.redhat.io/rhosp-rhel9/openstack-aodh-notifier:17.0           kolla_start
27 minutes ago  Up 27 minutes ago (healthy)             aodh_notifier
0ac8cb2c7470  registry.redhat.io/rhosp-rhel9/openstack-aodh-api:17.0              kolla_start
27 minutes ago  Up 27 minutes ago (healthy)             aodh_api_cron
31b55e091f57  registry.redhat.io/rhosp-rhel9/openstack-ceilometer-central:17.0
kolla_start  27 minutes ago  Up 27 minutes ago (healthy)             ceilometer_agent_central
5f61331a17d8  registry.redhat.io/rhosp-rhel9/openstack-ceilometer-compute:17.0
kolla_start  27 minutes ago  Up 27 minutes ago (healthy)             ceilometer_agent_compute
7c5ef75d8f1b  registry.redhat.io/rhosp-rhel9/openstack-ceilometer-notification:17.0
kolla_start  27 minutes ago  Up 27 minutes ago (healthy)
ceilometer_agent_notification
88fa57cc1235  registry.redhat.io/rhosp-rhel9/openstack-gnocchi-api:17.0            kolla_start
23 minutes ago  Up 23 minutes ago (healthy)             gnocchi_api
0f05a58197d5  registry.redhat.io/rhosp-rhel9/openstack-gnocchi-metricd:17.0
kolla_start  23 minutes ago  Up 23 minutes ago (healthy)             gnocchi_metricd
6d806c285500  registry.redhat.io/rhosp-rhel9/openstack-gnocchi-statsd:17.0
kolla_start  23 minutes ago  Up 23 minutes ago (healthy)             gnocchi_statsd
7c02cac34c69  registry.redhat.io/rhosp-rhel9/openstack-heat-api:17.0              kolla_start
27 minutes ago  Up 27 minutes ago (healthy)             heat_api_cron
d3903df545ce  registry.redhat.io/rhosp-rhel9/openstack-heat-api:17.0              kolla_start
27 minutes ago  Up 27 minutes ago (healthy)             heat_api
db1d33506e3d  registry.redhat.io/rhosp-rhel9/openstack-heat-api-cfn:17.0          kolla_start
27 minutes ago  Up 27 minutes ago (healthy)             heat_api_cfn
051446294c70  registry.redhat.io/rhosp-rhel9/openstack-heat-engine:17.0           kolla_start
27 minutes ago  Up 27 minutes ago (healthy)             heat_engine
```

3. 時系列データベースサービスが利用可能であることを確認します。

```
$ openstack metric status --fit-width
```

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

第3章 HEAT サービスを使用した自動スケーリング

オーバークラウドで自動スケーリングを提供するために必要なサービスをデプロイしたら、オーケストレーションサービス (heat) が自動スケーリング用のインスタンスを管理できるように、オーバークラウド環境を設定する必要があります。

前提条件

- デプロイされたオーバークラウド。詳細は、「自動スケーリングのためのオーバークラウドのデプロイ」 を参照してください。

手順

- 「自動スケーリング用の汎用アーカイブポリシーの作成」
- 「インスタンスを自動的にスケーリングするための heat テンプレートの設定」
- 「自動スケーリングのためのスタンドアロンデプロイの準備」
- 「自動スケーリング用のスタックデプロイメントの作成」

3.1. 自動スケーリング用の汎用アーカイブポリシーの作成

オーバークラウドに自動スケーリング用のサービスをデプロイしたら、オーケストレーションサービス (heat) が自動スケーリング用のインスタンスを管理できるように、オーバークラウド環境を設定する必要があります。

前提条件

- 自動スケーリングサービスを備えたオーバークラウドをデプロイしている。詳細は、「自動スケーリングのためのオーバークラウドの設定」 を参照してください。

手順

1. **stack** ユーザーとして環境にログインします。
2. スタンドアロン環境の場合は、**OS_CLOUD** 環境変数を設定します。

```
[stack@standalone ~]$ export OS_CLOUD=standalone
```

3. Director 環境の場合は、**stackrc** ファイルを入手します。

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

4. **$HOME/templates/autoscaling/parameters-autoscaling.yaml** で定義されたアーカイブポリシーを作成します。

```
$ 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'
```

検証

検証

- アーカイブポリシーが作成されたことを確認します。

```
$ 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. インスタンスを自動的にスケーリングするための HEAT テンプレートの設定

オーケストレーションサービス (heat) テンプレートを設定して、インスタンスを作成し、トリガー時に、インスタンスを作成およびスケーリングするアラームを設定できます。

注記

この手順では、環境に合わせて変更する必要がある値の例を使用します。

前提条件

- 自動スケーリングサービスを使用して、オーバークラウドをデプロイしている。詳細は、「自動スケーリングのためのオーバークラウドのデプロイ」 を参照してください。
- テレメトリーストレージを自動スケーリングするためのアーカイブポリシーを使用して、オーバークラウドを設定している。詳細は、「自動スケーリング用の汎用アーカイブポリシーの作成」 を参照してください。

手順

1. **stack** ユーザーとして環境にログインします。

```
$ source ~/stackrc
```

2. 自動スケーリンググループのインスタンス設定を保持するディレクトリーを作成します。

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

3. **$HOME/templates/autoscaling/vnf/instance.yaml** などのインスタンス設定テンプレートを作成します。

4. 次の設定を **instance.yaml** ファイルに追加します。

```
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** パラメーターは、この新しいリソースのカスタムパラメーターを定義します。
- **resources** パラメーターの **vnf** サブパラメーターは、**OS::Heat::AutoScalingGroup** で参照されるカスタムサブリソースの名前を定義します (例: **OS::Nova::Server::VNF**)。

5. heat テンプレートで参照するリソースを作成します。

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

6. インスタンスのスケーリングを制御するための heat のデプロイメントテンプレートを作成します。

```
$ cat <<EOF > $HOME/templates/autoscaling/vnf/template.yaml
heat_template_version: wallaby
description:  Example auto scale group, policy and alarm
resources:
  scaleup_group:
    type: OS::Heat::AutoScalingGroup
    properties:
      max_size: 3
      min_size: 1
      #desired_capacity: 1
      resource:
        type: OS::Nova::Server::VNF
        properties:
          metadata: {"metering.server_group": {get_param: "OS::stack_id"}}

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

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

  cpu_alarm_high:
    type: OS::Aodh::GnocchiAggregationByResourcesAlarm
    properties:
      description: Scale up instance if CPU > 50%
      metric: cpu
      aggregation_method: rate:mean
      granularity: 60
      evaluation_periods: 3
      threshold: 60000000000.0
      resource_type: instance
      comparison_operator: gt
      alarm_actions:
        - str_replace:
            template: trust+url
            params:
```

```
            url: {get_attr: [scaleup_policy, signal_url]}
      query:
        list_join:
          - ''
          - - {'=': {server_group: {get_param: "OS::stack_id"}}}

  cpu_alarm_low:
    type: OS::Aodh::GnocchiAggregationByResourcesAlarm
    properties:
      description: Scale down instance if CPU < 20%
      metric: cpu
      aggregation_method: rate:mean
      granularity: 60
      evaluation_periods: 3
      threshold: 24000000000.0
      resource_type: instance
      comparison_operator: lt
      alarm_actions:
        - str_replace:
            template: trust+url
            params:
              url: {get_attr: [scaledown_policy, signal_url]}
      query:
        list_join:
          - ''
          - - {'=': {server_group: {get_param: "OS::stack_id"}}}

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

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

注記

スタックの出力は、情報提供用であり、ScalingPolicy または AutoScalingGroup では参照されません。出力を表示するには、**openstack stack show <stack_name>** コマンドを使用します。

3.3. 自動スケーリングのためのスタンドアロンデプロイの準備

実稼働前環境で自動スケーリングされたインスタンスのスタックのデプロイメントをテストするには、スタンドアロンデプロイメントを使用して、スタックをデプロイします。この手順を使用して、スタンドアロン環境でデプロイメントをテストできます。実稼働環境では、デプロイメントコマンドが異なります。

手順

1. **stack** ユーザーとして環境にログインします。
2. **OS_CLOUD** 環境変数を設定します。

```
[stack@standalone ~]$ export OS_CLOUD=standalone
```

3. プライベートおよびパブリックネットワークインターフェイスが接続された Fedora 36 クラウドイメージを使用する、シミュレーションされた VNF ワークロードのデプロイメントを許可するようにクラウドを設定します。この例は、スタンドアロンデプロイメントを使用する作業設定です。

```
$ export GATEWAY=192.168.25.1
$ export STANDALONE_HOST=192.168.25.2
$ export PUBLIC_NETWORK_CIDR=192.168.25.0/24
$ export PRIVATE_NETWORK_CIDR=192.168.100.0/24
$ export PUBLIC_NET_START=192.168.25.3
$ export PUBLIC_NET_END=192.168.25.254
$ export DNS_SERVER=1.1.1.1
```

4. フレーバーを作成します。

```
$ openstack flavor create --ram 2048 --disk 10 --vcpu 2 --public m1.small
```

5. Fedora 36 x86_64 クラウドイメージをダウンロードおよびインポートします。

```
$ curl -L
'https://download.fedoraproject.org/pub/fedora/linux/releases/36/Cloud/x86_64/images/Fedora-Cloud-Base-36-1.5.x86_64.qcow2' -o $HOME/fedora36.qcow2
```

```
$ openstack image create fedora36 --container-format bare --disk-format qcow2 --public --file $HOME/fedora36.qcow2
```

6. 公開鍵を生成およびインポートします。

```
$ ssh-keygen -f $HOME/.ssh/id_rsa -q -N "" -t rsa -b 2048
```

```
$ openstack keypair create --public-key $HOME/.ssh/id_rsa.pub default
```

7. SSH、ICMP、および DNS プロトコルを許可する **basic** セキュリティーグループを作成します。

```
$ openstack security group create basic
```

```
$ openstack security group rule create basic --protocol tcp --dst-port 22:22 --remote-ip 0.0.0.0/0
```

```
$ openstack security group rule create --protocol icmp basic
```

```
$ openstack security group rule create --protocol udp --dst-port 53:53 basic
```

8. 外部ネットワーク (パブリック) を作成します。

```
$ openstack network create --external --provider-physical-network datacentre --provider-network-type flat public
```

9. プライベートネットワークを作成します。

```
$ openstack network create --internal private
```

```
openstack subnet create public-net \
  --subnet-range $PUBLIC_NETWORK_CIDR \
  --no-dhcp \
  --gateway $GATEWAY \
  --allocation-pool start=$PUBLIC_NET_START,end=$PUBLIC_NET_END \
  --network public
```

```
$ openstack subnet create private-net \
  --subnet-range $PRIVATE_NETWORK_CIDR \
  --network private
```

10. ルーターを作成します。

```
$ openstack router create vrouter
```

```
$ openstack router set vrouter --external-gateway public
```

```
$ openstack router add subnet vrouter private-net
```

関連情報

- スタンドアロン導入 ガイド。

3.4. 自動スケーリング用のスタックデプロイメントの作成

例として操作した VNF 自動スケーリングのスタックデプロイメントを作成します。

前提条件

- 「インスタンスを自動的にスケーリングするための heat テンプレートの設定」

手順

1. スタックを作成します。

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

検証

1. スタックが正常に作成されたことを確認します。

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

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

2. アラーム、スケーリングポリシー、自動スケーリンググループを含むスタックリソースが作成されたことを確認します。

```
$ 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. スタックの作成によってインスタンスが起動されたことを確認します。

```
$ 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 | standalone-80.localdomain | metering.server_group='cb082cbd-535e-4779-84b0-98925e103f5e' |
```

4. スタックのアラームが作成されたことを確認します。

   a. アラーム ID を一覧表示します。アラームの状態は、一定期間、**insufficient data** 状態になることがあります。最小期間は、データ収集とデータストレージの粒度設定のポーリング間隔です。

```
$ 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. スタックのリソースを一覧表示し、**cpu_alarm_high** および **cpu_alarm_low** リソースの **physical_resource_id** 値をメモします。

```
$ 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 の値は、**openstack alarm list** コマンドの出力の **alarm_id** に一致する必要があります。

5. スタックのメトリクスリソースが存在することを確認します。**server_group** クエリーの値をスタック ID に設定します。

```
$ 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. スタックを介して作成されたインスタンスリソースの測定値が存在することを確認します。

```
$ openstack metric aggregates --resource-type instance --sort-column timestamp '(metric
cpu rate:mean)' server_group="$STACK_ID"
+-------------------------------------------------------+--------------------------+-------------+---------------
+
| name                                                  | timestamp                | granularity |         value |
+-------------------------------------------------------+--------------------------+-------------+---------------
+
| 62e1b27c-8d9d-44a5-a0f0-80e7e6d437c7/cpu/rate:mean | 2022-10-06T23:11:00+00:00 |
60.0 | 69470000000.0 |
| 62e1b27c-8d9d-44a5-a0f0-80e7e6d437c7/cpu/rate:mean | 2022-10-06T23:12:00+00:00 |
60.0 | 81060000000.0 |
| 62e1b27c-8d9d-44a5-a0f0-80e7e6d437c7/cpu/rate:mean | 2022-10-06T23:13:00+00:00 |
60.0 | 82840000000.0 |
| 62e1b27c-8d9d-44a5-a0f0-80e7e6d437c7/cpu/rate:mean | 2022-10-06T23:14:00+00:00 |
60.0 | 66660000000.0 |
| 62e1b27c-8d9d-44a5-a0f0-80e7e6d437c7/cpu/rate:mean | 2022-10-06T23:15:00+00:00 |
60.0 |  7360000000.0 |
| 62e1b27c-8d9d-44a5-a0f0-80e7e6d437c7/cpu/rate:mean | 2022-10-06T23:16:00+00:00 |
60.0 |  3150000000.0 |
| 62e1b27c-8d9d-44a5-a0f0-80e7e6d437c7/cpu/rate:mean | 2022-10-06T23:17:00+00:00 |
60.0 |  2760000000.0 |
| 62e1b27c-8d9d-44a5-a0f0-80e7e6d437c7/cpu/rate:mean | 2022-10-06T23:18:00+00:00 |
60.0 |  3470000000.0 |
| 62e1b27c-8d9d-44a5-a0f0-80e7e6d437c7/cpu/rate:mean | 2022-10-06T23:19:00+00:00 |
60.0 |  2770000000.0 |
| 62e1b27c-8d9d-44a5-a0f0-80e7e6d437c7/cpu/rate:mean | 2022-10-06T23:20:00+00:00 |
60.0 |  2700000000.0 |
+-------------------------------------------------------+--------------------------+-------------+---------------
+
```

第4章 自動スケーリングのテストとトラブルシューティング

オーケストレーションサービス (heat) を使用して、しきい値の定義に基づいて、インスタンスを自動的にスケールアップおよびスケールダウンします。環境のトラブルシューティングを行うには、ログファイルと履歴レコードでエラーを探します。

4.1. インスタンスの自動スケールアップのテスト

オーケストレーションサービス (heat) を使用して、**cpu_alarm_high** しきい値の定義に基づいて、インスタンスを自動的にスケーリングできます。CPU 使用率が **threshold** パラメーターで定義された値に達すると、負荷を分散するために、別のインスタンスが起動します。**template.yaml** ファイルの **threshold** 値は、80% に設定されます。

手順

1. **stack** ユーザーとしてホスト環境にログインします。
2. スタンドアロン環境の場合は、**OS_CLOUD** 環境変数を設定します。

```
[stack@standalone ~]$ export OS_CLOUD=standalone
```

3. Director 環境の場合は、**stackrc** ファイルを入手します。

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

4. インスタンスにログインします。

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

5. 複数の **dd** コマンドを実行して、負荷を生成します。

```
[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 &
```

6. 実行中のインスタンスを終了し、ホストに戻ります。
7. **dd** コマンドを実行すると、インスタンスで 100% の CPU 使用率を期待できます。アラームがトリガーされていることを確認します。

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

8. 約 60 秒後、オーケストレーションは、別のインスタンスを開始し、グループに追加します。インスタンスが作成されたことを確認するには、次のコマンドを入力します。

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

9. しばらくしてから、オーケストレーションサービスが 3 つのインスタンスに自動スケーリングされたことを確認します。設定は、最大 3 つのインスタンスに設定されます。3 つのインスタンスがあることを確認します。

```
$ 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. インスタンスの自動スケールダウンのテスト

オーケストレーションサービス (heat) を使用して、**cpu_alarm_low** しきい値に基づいて、インスタンスを自動的にスケールダウンできます。この例では、CPU の使用率が 5% 未満の場合に、インスタンスはスケールダウンされます。

手順

1. ワークロードインスタンス内から、実行中の **dd** プロセスを終了し、オーケストレーションがインスタンスのスケールダウンを開始することを確認します。

```
$ killall dd
```

2. **stack** ユーザーとしてホスト環境にログインします。

3. スタンドアロン環境の場合は、**OS_CLOUD** 環境変数を設定します。

```
[stack@standalone ~]$ export OS_CLOUD=standalone
```

4. Director 環境の場合は、**stackrc** ファイルを入手します。

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

5. **dd** プロセスを停止すると、**cpu_alarm_low event** アラームがトリガーされます。これにより、オーケストレーションは自動的にスケールダウンを開始し、インスタンスを削除します。対応するアラームがトリガーされていることを確認します。

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

数分後に、オーケストレーションは、**scaleup_group** 定義の **min_size** パラメーターで定義される最小値までインスタンスの数を継続的に削減します。このシナリオでは、**min_size** パラメーターは **1** に設定されています。

4.3. 自動スケーリングのトラブルシューティング

環境が適切に機能していない場合は、ログファイルと履歴レコードでエラーを確認できます。

手順

1. **stack** ユーザーとしてホスト環境にログインします。
2. スタンドアロン環境の場合は、**OS_CLOUD** 環境変数を設定します。

```
[stack@standalone ~]$ export OS_CLOUD=standalone
```

3. Director 環境の場合は、**stackrc** ファイルを入手します。

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

4. 状態遷移についての情報を取得するには、スタックイベントレコードをリスト表示します。

```
$ 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)
```

5. アラーム履歴ログを読み取ります。

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

6. 既存のスタック用に heat が収集するスケールアウトまたはスケールダウン操作のレコードを表示するには、**awk** コマンドを使用して **heat-engine.log** を解析することができます。

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

7. aodh 関連の情報を表示するには、**evaluator.log** を調べます。

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

4.4. RATE:MEAN 集計を使用する場合の自動スケーリングのしきい値に CPU テレメトリー値を使用する

OS::Heat::Autoscaling ヒートオーケストレーションテンプレート (HOT) を使用し、CPU のしきい値を設定すると、値は CPU 時間のナノ秒単位で表されます。これは、インスタンスのワークロードに割り当てられた仮想 CPU の数に基づく動的な値です。このリファレンスガイドでは、gnocchi の **rate:mean** 集計メソッドを使用する場合に、CPU ナノ秒値をパーセンテージで計算および表現する方法について説明します。

4.4.1. CPU テレメトリー値をパーセンテージとして計算する

CPU テレメトリーは、ナノ秒単位の CPU 使用率として gnocchi (OpenStack 時系列データストア) に保存されます。CPU テレメトリーを使用して、自動スケーリングのしきい値を定義する場合は、CPU 使用率のパーセンテージで値を表すと便利です。しきい値を定義する場合は、その方が自然です。自動スケーリンググループの一部として使用されるスケーリングポリシーを定義する場合は、パーセンテージとして定義された目的のしきい値を取得し、ポリシー定義で使用される必要なしきい値をナノ秒単位で計算できます。

値 (ナノ秒)	粒度 (秒)	パーセンテージ
60000000000	60	100
54000000000	60	90

値 (ナノ秒)	粒度 (秒)	パーセンテージ
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. インスタンスワークロードの vCPU をパーセンテージで表示する

openstack metric aggregates コマンドを使用して、gnocchi に保存された CPU テレメトリーデータをインスタンスのナノ秒値ではなく、パーセンテージで表示できます。

前提条件

- 自動スケーリンググループリソースを使用して、heat スタックを作成し、インスタンスワークロードを生成している。

手順

1. クラウド管理者として OpenStack 環境にログインします。
2. 自動スケーリンググループの heat スタックの ID を取得します。

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

3. スタック ID の値を環境変数に設定します。

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

4. リソースタイプインスタンス (サーバー ID) ごとの集計としてメトリクスを返します。値はパーセンテージとして計算されます。集計は、CPU 時間のナノ秒単位の値として返されます。その数値を 1000000000 で割って、秒単位の値を取得します。次に、値を粒度 (この例では 60 秒)

で割ります。その値は、100 を掛けて、パーセンテージに変換されます。最後に、インスタンスに割り当てられたフレーバーによって提供される vCPU の数で合計値を割ります。この例では、2 vCPU の値であり、CPU 時間のパーセンテージとして表される値を提供します。

```
$ 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. インスタンスワークロードの使用可能なテレメトリーを取得する

インスタンスワークロードの使用可能なテレメトリーを取得し、vCPU 使用率をパーセンテージで表します。

前提条件

- 自動スケーリンググループリソースを使用して、heat スタックを作成し、インスタンスワークロードを生成している。

手順

1. クラウド管理者として OpenStack 環境にログインします。
2. 自動スケーリンググループの heat スタックの ID を取得します。

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

3. スタック ID の値を環境変数に設定します。

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

4. データを返すワークロードインスタンスの ID を取得します。サーバーリストの長い形式を使用し、自動スケーリンググループの一部であるインスタンスをフィルタリングします。

```
$ 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              | standalone.localdomain |
metering.server_group='e0a15cee-34d1-418a-ac79-74ad07585730' |
```

5. 返されたインスタンスワークロード名のいずれかのインスタンス ID を設定します。

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

6. インスタンスリソース ID のメトリクスが保存されていることを確認します。メトリクスが利用できない場合は、インスタンスが作成されてから、十分な時間が経過していない可能性があります。十分な時間が経過したら、データ収集サービスのログを **/var/log/containers/ceilometer/** で、時系列データベースサービス gnocchi のログを **/var/log/containers/gnocchi/** で確認できます。

```
$ 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. リソースメトリクスに使用可能な測定値があることを確認し、**openstack metric aggregates** コマンドを実行する際に使用する粒度の値をメモします。

```
$ 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. インスタンスワークロード用に設定されたフレーバーを確認して、ワークロードインスタンスに適用される vCPU コアの数を取得します。

```
$ 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. リソースタイプインスタンス (サーバー ID) ごとの集計としてメトリクスを返します。値はパーセンテージとして計算されます。集計は、CPU 時間のナノ秒単位の値として返されます。その数値を 1000000000 で割って、秒単位の値を取得します。次に、値を粒度で割ります。この例では、60 秒です (以前に **openstack metric measure show** コマンドで取得)。その値は、100 を掛けて、パーセンテージに変換されます。最後に、インスタンスに割り当てられたフレーバーによって提供される vCPU の数で合計値を割ります。この例では、2 vCPU の値であり、CPU 時間のパーセンテージとして表される値を提供します。

```
$ 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 |
...
```