OSM#9 Hackfest
Hack 3.3 : Adding Auto-Scaling & Alerting to VNFs

Subhankar Pal
(Altran)
Current Auto Scaling & Alarms Feature
OSM Service Assurance
Revisiting Service Assurance MDG

Main components

- **MON**
  - Covers the basic use cases, with a solid architecture to expand them easily.
  - Opportunities to enhance usability.

- **POL**
  - Designed around the autoscaling use case.
  - Starting to cover VNF alarms.
  - Architecture needs a revisit based on expected use cases.

- **Prometheus**
  - OSM’s TSDB for metrics since REL5
  - Opportunities to enhance multi-tenancy to match new RBAC capabilities.

Auxiliary/Optional

- **Grafana**
  - Integrates seamlessly with Prometheus.
  - Great tool for enhancing usability of the system’s Service Assurance

- **ELK**
  - Proved seamless integration with OSM.
  - Main use case remains at log processing where stack is used.

And an upcoming Placement module!
Auto Scaling & Alarms Features

**Auto Scaling**

- Auto scaling allows to automatically scale VNFs with a VDU granularity and based on any available metric.
- Scaling descriptors can be included and be tied to automatic reaction to VIM/VNF metric thresholds.
- Supported metrics are both VIM and VNF metrics.

**Alarms**

- An internal alarm manager has been added to MON through the 'mon-evaluator' module, so that both VIM and VNF metrics can also trigger threshold-violation alarms and scaling actions.
Revisiting MON Architecture

POL Architecture

Formal documentation: https://osm.etsi.org/gitlab/osm-architecture/osm-arch-doc/blob/master/05-pol.md
When configuring alarms associated to scaling actions or just webhook notifications (through the VNFD), the following components interact.

- ‘mon-evaluator’ evaluates thresholds related to metrics
- Policy Manager module (POL) takes actions such as auto-scaling.
- Whenever a threshold is crossed and an alarm is triggered, the notification is generated by MON and put in the Kafka bus so other components, like POL can consume them.
### Scaling Group Descriptor

The scaling descriptor is part of a VNFD. Like the example shows, it mainly specifies:

- An existing metric to be monitored, which should be pre-defined in the monitoring-param list (vnf-monitoring-param-ref).
- The thresholds to monitor (scale-in/out-threshold)
- The minimum and maximum amount of scaled instances to produce.
- The minimum time it should pass between scaling operations (cooldown-time)
- The VDU to be scaled (vdu-id-ref) and the amount of instances to scale per event (count)

<table>
<thead>
<tr>
<th>scaling-group-descriptor:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- <strong>max-instance-count</strong>: 1</td>
</tr>
<tr>
<td>- <strong>cooldown-time</strong>: 120</td>
</tr>
<tr>
<td>- <strong>name</strong>: cpu_util_above_threshold</td>
</tr>
<tr>
<td>- <strong>scale-in-relational-operation</strong>: LT</td>
</tr>
<tr>
<td>- <strong>scale-in-threshold</strong>: 10</td>
</tr>
<tr>
<td>- <strong>scale-out-relational-operation</strong>: GT</td>
</tr>
<tr>
<td>- <strong>scale-out-threshold</strong>: 60</td>
</tr>
<tr>
<td>- <strong>vnf-monitoring-param-ref</strong>: agw_cpu_util</td>
</tr>
<tr>
<td>- <strong>scaling-type</strong>: automatic</td>
</tr>
<tr>
<td>- <strong>threshold-time</strong>: 10</td>
</tr>
<tr>
<td>vdu:</td>
</tr>
<tr>
<td>- <strong>count</strong>: 1</td>
</tr>
<tr>
<td>- <strong>vdu-id-ref</strong>: magma-agw-vdu</td>
</tr>
</tbody>
</table>

**Example:**

- **max-instance-count**: 1
- **cooldown-time**: 120
- **name**: cpu_util_above_threshold
- **scale-in-relational-operation**: LT
- **scale-in-threshold**: 10
- **scale-out-relational-operation**: GT
- **scale-out-threshold**: 60
- **vnf-monitoring-param-ref**: agw_cpu_util
- **scaling-type**: automatic
- **threshold-time**: 10
- **vdu**: count: 1
- **vdu-id-ref**: magma-agw-vdu
Alarms based on metric thresholds can be sent to webhooks. The alarm descriptor is also part of a VNFD. Like the example shows, it mainly specifies:

- An existing metric to be monitored, which should be pre-defined in the monitoring-param list (vnf-monitoring-param-ref).
- The thresholds to monitor (alarm-threshold)
- The webhook to be invoked (url)
New Proposals
OSM Service Assurance
Closed-loop automation powers autonomous networks.

1. Observe
Collect network metrics through different telemetry interfaces.

2. Decide
Processes collected metrics to determines the network status, decides action to be taken based on network policies. This phase is not responsible for executing the action.

3. Act
Acts upon orchestrated object and implements given lifecycle action.
Auto-Scaling & Alarms – New Architecture

VIM Alarms (aodh)

VNF metrics (prometheus)

VNF Alarms (ex. snmp traps → webhook)

OSM VNF Metric Exporters

(1a) Aodh notification

(1b) AlertManager notification

(1c) Custom webhooks

In the future, POL could act upon combined alarms

(2c) if action is to scale: send to bus for LCM to proceed and store action to commonDB

(2d) if action is to notify, send notification to webhook service

webhook service (external)

lcm

pol
Other Features in Roadmap

• Automatically monitor auto scaled VDU
• Setting threshold on correlated metrics (multiple metrics)
• Move away from threshold to ML based anomaly detection
Hands-on!

VNF Monitoring
Let’s play with metrics and (auto)dashboards!

- We will stress the VDU of AGW VNF

This is your IP
Let’s play with metrics and (auto)dashboards!

- Login to AGW VM from your OSM command line
  
  $ ssh magma@172.21.248.106

- Increase CPU load with this command. Not down the process id.
  
  $ yes > /dev/null &


- Observe increase in CPU load and eventually a new VDU is created through auto scaling.
Let’s play with metrics and (auto)dashboards!

- Check webhook invoked at https://webhook.site/ when alarm is generated.
- Now locate the IP of the process and kill it to reduce the extra CPU load
  
  $ kill 3904

- Observe decrease in CPU load and eventually a additional VDU is deleted.