Open Source MANO

OSM-MR#10 Hackfest
Network Automation (Scaling)
Subhankar Pal (Altran)
The big picture !!

Cloud Environment

- OpenStack
- K8s
- Wiki
- Virtual Desktop
- Web Cache
- Performance Monitor
- LDAP

Auto scale on demand

OSM

Employee

Firewall

Private Network

© ETSI
Hands-on!
Auto Scaling of Wiki Service
Let’s play with wiki service!

Build Package

- Login to Mgmt. VM (172.21.1.3)

$ ssh osm_hackfest_xx@172.21.1.3

- Copy the scripts from location ~/Hackfest/HD2.2-Scaling/.

$ cp ~/Hackfest/HD2.2-Scaling/wiki-*.sh .

- Build and upload wiki VNF and NS package.
  Warning ‘osm-packages’ folder will be deleted. So if you need it later, backup this folder, before running this script,

$ ./wiki-build.sh
Let’s play with wiki service!
Verify package are uploaded in OSM

• Verify the packages are successfully deployed in OSM

```bash
$ osm nspkg-list
```

```
nsd name                  id                                      
wiki_webserver_autoscale_ns_osm_instructor_9  1cdc9351-cfd7-42c2-9416-33ff85216a9e 
```

```bash
$ osm vnfpkg-list
```

```
nfpkg name              id                          desc type          
wiki_webserver_autoscale_vnf_osm_instructor_9 c70fe4c5-9401-4e19-b1af-d70978114280 sol006 
```
Let's play with wiki service!

Instantiate Wiki Network Service

- Install wiki network service

  $ ./wiki-launch.sh

- Verify the network service is deployed properly.

  $ osm ns-list

<table>
<thead>
<tr>
<th>ns instance name</th>
<th>id</th>
<th>date</th>
<th>ns state</th>
<th>current operation</th>
<th>error details</th>
</tr>
</thead>
<tbody>
<tr>
<td>wiki</td>
<td>706f5a5a-74c5-4cba-9145-5d1a7eb22d5c</td>
<td>2021-03-08T16:07:08</td>
<td>READY</td>
<td>IDLE (None)</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Let’s play with wiki service!
Verify wiki service is accessible

• List VMs deployed in the Openstack VIM. There should be

$$\texttt{openstack server list}$$

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Status</th>
<th>Networks</th>
</tr>
</thead>
<tbody>
<tr>
<td>e089084e-ba89-46c0-0c6a-6a692c4da932</td>
<td>wiki-1-apache_vdu-0</td>
<td>ACTIVE</td>
<td>wiki-internal_service=192.168.28.9</td>
</tr>
<tr>
<td>dcac250-0db9-47de-55fe-2bced175a4e6</td>
<td>wiki-1-haproxy_vdu-0</td>
<td>ACTIVE</td>
<td>osm-ext=172.21.10.211; wiki-internal_service=192.168.28.2</td>
</tr>
<tr>
<td>94a6c0d9-c961-4252-a51d-905d1b79c2be</td>
<td>vyos-pnl-router</td>
<td>ACTIVE</td>
<td>osm-ext=172.21.19.99; private=192.168.299.250</td>
</tr>
</tbody>
</table>

• Check if haproxy is working by trying to access the URL

http://<HA Proxy External IP>/
Let’s play with wiki service!
Troubleshoot if necessary

• You should see the following page.

• **Troubleshooting Tips** If HA proxy is not started start automatically play the following commands from haproxy VM

  `$ service haproxy status`
  `$ sudo service haproxy restart`
Let’s play with wiki service!

Service Monitoring

- Check monitoring is happening by logging into Grafana

http://<OSM IP Address>:3000/
Username / Password = osm_hackfest_xx / osm_hackfest_xx
Let's play with wiki service!

Increase web traffic

• Login to haproxy VM

$ ssh ubuntu@172.21.18.211
Password = osm2021

• Increase http traffic to the apache web server.

$ ab -n 10000000 -c 1000 http://<Private IP of Apache webserver>:8080/
Let’s play with wiki service!
Hurray!! Wiki scaled out.

- Verify CPU utilization increase in Grafana

- In Mgmt. VM (172.21.1.3) check additional webserver VMs are instantiated

```bash
$ openstack server list
```

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Status</th>
<th>Networks</th>
<th>Image</th>
<th>Flavor</th>
</tr>
</thead>
<tbody>
<tr>
<td>6cf3f5a8-7dd0</td>
<td>wiki-1-apache_vdu-1</td>
<td>ACTIVE</td>
<td>wiki-internal_service=192.168.2.8</td>
<td>apacheUbuntu</td>
<td>1C-180G</td>
</tr>
<tr>
<td>e0b1d4-8c83-4d0b</td>
<td>wiki-1-apache_vdu-1</td>
<td>ACTIVE</td>
<td>wiki-internal_service=192.168.2.8</td>
<td>apacheUbuntu</td>
<td>1C-180G</td>
</tr>
<tr>
<td>4ac255a9-0db7-477d</td>
<td>wiki-1-haproxy_vdu-0</td>
<td>ACTIVE</td>
<td>osm-ext=172.21.16.211; wiki-internal_service=192.168.2.8</td>
<td>haproxyUbuntu</td>
<td>4C-480G</td>
</tr>
<tr>
<td>9a6c8d9-72ae-4525</td>
<td>vmos-pf_router</td>
<td>ACTIVE</td>
<td>osm-ext=172.21.15.59, private=192.168.235.250</td>
<td>vmos-1.1.7-cloudinit</td>
<td>m1.small</td>
</tr>
</tbody>
</table>
Let’s play with wiki service!
Stop additional traffic to trigger scale in.

- Also, HA Proxy is connecting to these additional webserver VMs

Thank you for trying OSM!
Active web server VDUs in this VNF: 2
192.168.28.8
192.168.28.9

- Stop the traffic generator to see the scale in triggered.
Auto Scaling & Alert Architecture

OSM Service Assurance
Revisiting Service Assurance MDG

Main components

- **MON**
  - Covers the basic uses 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.

- **PLA**
  - Provides computation of optimal placement of NFs over VIMs.
  - Considers cost of compute/network.

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

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

Auxiliary/Optional

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

Alerts

• 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

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.

1. NBI
2. commonDB (mongo)
3. mon-evaluator
4a. lcm
4b. pol
5. tsdb (prometheus)
6. Policy DB (MySQL)
7. webhook service (external)

(1) continuously looks for configured alarms at VNF record
(2) queries for metric values
(3) when triggered, puts alarm in bus for pol to take actions
(4a) if action is to scale: send to bus for LCM to proceed and store action to commonDB
(4b) if action is to notify, send notification to webhook service
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 (monitoring-parameter).
- The thresholds to monitor (scale-in/out-threshold)
- The minimum amount of scaled instances to produce (max-scale-level).
- The minimum time it should pass between scaling operations (cooldown-time)
- The VDU to be scaled and the amount of instances to scale per event
What’s New in Release 9?
For OSM Service Assurance
Role-Based Access Control (RBAC) in OSM provides different users and projects a controlled access to different resources. For achieving this, different backends are available.

Option 1 (Internal)

Option 2 (Keystone)

Option 3 (LDAP)
Multi Tenant Grafana

Grafana multi-tenancy extends the RBAC feature to OSM’s Grafana and provides OSM users with controlled access to OSM dashboards. With multi-tenancy, users can now login to Grafana with their OSM credentials instead of a common username as was the case in previous releases.

Feature Description: https://osm.etsi.org/gerrit/#/c/osm/Features/+/9177/
How it was before Release 9?

User 1

Project 1

User 2

Project 2

Open Source MANO

Grafana

admin (default)
How it is in Release 9?
OSM RBAC & Grafana Mapping

- User
- Project
- Role
- NFV/VNF

- User
- Team
- Team Permissions
- Dashboards

Grafana

OSM RBAC
Thank You !!
Find us at:

osm.etsi.org
osm.etsi.org/wikipub