OSM#10 Hackfest
Monitoring Network Functions

Atul Agarwal
(Altran)
## Service Assurance MDG

### Main components

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MON</strong></td>
<td>Covers the basic use cases, with a solid architecture to expand them easily.</td>
</tr>
<tr>
<td><strong>POL</strong></td>
<td>Provides computation of optimal placement of NFs over VIMs. Considers cost of compute/network.</td>
</tr>
<tr>
<td><strong>PLA</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Prometheus</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Grafana</strong></td>
<td></td>
</tr>
<tr>
<td><strong>ELK</strong></td>
<td></td>
</tr>
</tbody>
</table>

### Auxiliary/Optional

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Grafana</strong></td>
<td>Integrates seamlessly with Prometheus. Great tool for enhancing usability of the system’s Service Assurance.</td>
</tr>
</tbody>
</table>

© ETSI 2020
MON Architecture

Formal documentation: https://osm.etsi.org/gitlab/osm-architecture/osm-arch-doc/blob/master/04-mon.md
When launching a new instance of a Network Service or Slice Instance (n x VNFs) which is described with the collection of VNF Metrics that come from infrastructure (NFVI), the following components interact.

1. **metrics collection primitive**
   - **lcm**
   - **VNFs**
   - **VIM (external)**
   - **VCA**
   - **commonDB (mongo)**
2. **metrics primitive**
   - **Juju**
3. **continuously looks for active VNFs with metrics**
   - **commonDB (mongo)**
4. **API calls to VIM/VCA, to collect metrics**
   - **lcm**
   - **mon**
   - **exporter**
5. **looks for active metrics**
   - **tsdb (prometheus)**
6. **reads & presents metrics**
   - **grafana**

The components interact as follows:

- The **lcm** initiates the metrics collection and sets the metrics collection primitive.
- **VNFs** are connected to the **VIM (external)** as a component of the infrastructure.
- The **commonDB (mongo)** continuously looks for active VNFs with metrics.
- **Mon** makes API calls to VIM/VCA to collect metrics.
- The **tsdb (prometheus)** continuously looks for active metrics.
- The **grafana** application reads and presents the metrics.
When creating Projects or Network Services, Grafana dashboards are created automatically and the following elements interact.

1. continuously looks for new projects or NS with metrics
2a. if project: create Project dashboard
2b. if NS: create NS dashboard (with sample graphs)
(*) continuously delete obsolete dashboards
Metrics Collection

- VDU Metric Collection from VIM

```yaml
monitoring-param:
  - aggregation-type: AVERAGE
    id: agw_cpu_util
    name: agw_cpu_util
    vdu-monitoring-param:
      vdu-monitoring-param-ref: agw_cpu_util
      vdu-ref: magma-agw-vdu
  - aggregation-type: AVERAGE
    id: agw_memory_util
    name: agw_memory_util
    vdu-monitoring-param:
      vdu-monitoring-param-ref: agw_memory_util
      vdu-ref: magma-agw-vdu
  - aggregation-type: AVERAGE
    id: agw_packets_received
    name: agw_packets_received
    vdu-monitoring-param:
      vdu-monitoring-param-ref: agw_packets_received
      vdu-ref: magma-agw-vdu
  - aggregation-type: AVERAGE
    id: agw_packets_sent
    name: agw_packets_sent
    vdu-monitoring-param:
      vdu-monitoring-param-ref: agw_packets_sent
      vdu-ref: magma-agw-vdu
```
## Prometheus collects the following metrics from “MON Exporter”

<table>
<thead>
<tr>
<th>Metric</th>
<th>Collection type</th>
<th>Behavior</th>
<th>KPI</th>
<th>Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIM Status</td>
<td>Infrastructure</td>
<td>By default</td>
<td>status (up/down)</td>
<td>vim_id</td>
</tr>
<tr>
<td>SDNC Status</td>
<td></td>
<td></td>
<td>status (up/down)</td>
<td>sdnc_id</td>
</tr>
<tr>
<td>VM Status</td>
<td></td>
<td></td>
<td>status (up/down)</td>
<td></td>
</tr>
<tr>
<td>VDU CPU Utilization</td>
<td>VNF</td>
<td>Enabled by descriptor</td>
<td>utilization, rate, etc.</td>
<td>nsr_id, vnf_member_index, vdu_name</td>
</tr>
<tr>
<td>VDU Memory Utilization</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VDU Packet forwarding</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VNF Metrics through Juju</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(to be deprecated)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
VNF Indicator Collection

VNF PACKAGE

1. Instantiation
   - NBI

2. HELM Deployment
   - Helm chart: MetricsEE
     (SNMP Exporter + Config API)

3. (Re) Configure
   - LCM
   - Config API

4. SNMP Exporter
   - generator.yaml & MIB files

5. MIB files + generator.yaml

6. Prometheus
   - Get me all metrics at "mySnmpMetrics" module, from this target VNF

7. VNF
   - Helm-based Exec Environment

**NEW JOB**
- mySnmpMetrics:
  - VDU Mgmt IP
  - SNMP Exporter IP
  - NS/VNF/VDU ID/Names (labels)
  - Metrics prefix

GET & Export

vendor provided
(generator template provided by OSM)

base chart provided by OSM
Hands-on!

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

• In the VNF Package editor add the following lined in YAML after line #8

```
monitoring-param:
  - aggregation-type: AVERAGE
    id: agw_cpu_util
    name: agw_cpu_util
    vdu-monitoring-param:
      vdu-monitoring-param-ref: agw_cpu_util
      vdu-ref: magma-agw-vdu
  - aggregation-type: AVERAGE
    id: agw_memory_util
    name: agw_memory_util
    vdu-monitoring-param:
      vdu-monitoring-param-ref: agw_memory_util
      vdu-ref: magma-agw-vdu
  - aggregation-type: AVERAGE
    id: agw_packets_received
    name: agw_packets_received
    vdu-monitoring-param:
      vdu-monitoring-param-ref: agw_packets_received
      vdu-ref: magma-agw-vdu
  - aggregation-type: AVERAGE
    id: agw_packets_sent
    name: agw_packets_sent
    vdu-monitoring-param:
      vdu-monitoring-param-ref: agw_packets_sent
      vdu-ref: magma-agw-vdu
```
Let’s play with metrics and (auto)dashboards!

Metrics collection is starts (5 to 10 minutes due to current collection period