**OSM#10 Hackfest**
**Closed-Loop Operations**
Adding Auto-Scaling & Alerting to VNFs

Subhankar Pal, Altran

# Current Auto Scaling & Alarms Feature

OSM Service Assurance

# Revisiting Service Assurance MDG

**Open Source MANO**

## Main components

## Auxiliary/ Optional

**MON**

**POL**

**PLA**

**Prometheus**

**Grafana**

**ELK**

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

- Designed around the autoscaling use case.
- Starting to cover VNF alarms.

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

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

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

- Proved seamless integration with OSM.
- Main use case remains at log processing where stack is used.
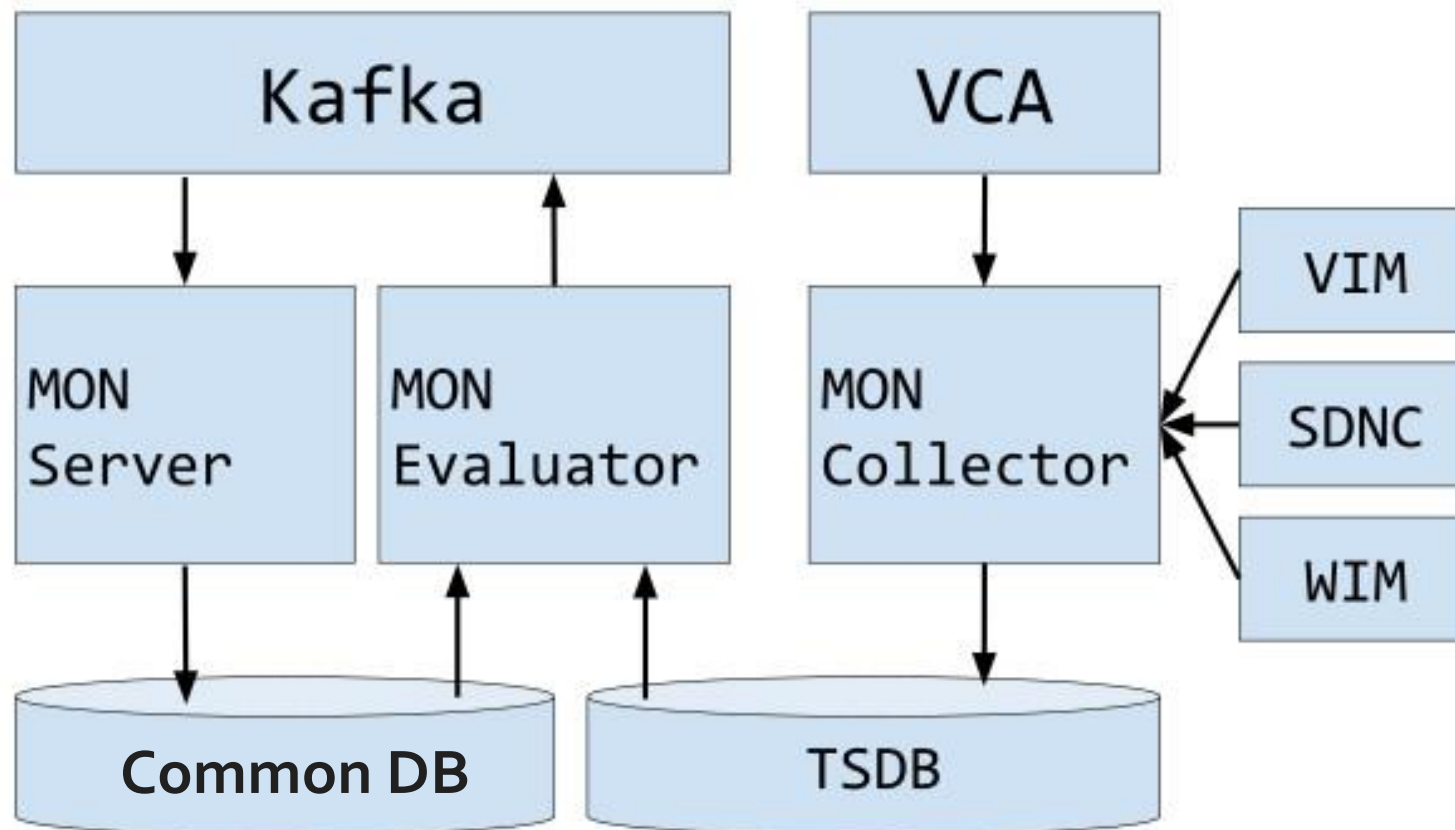
# Auto Scaling & Alarms Features

## 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
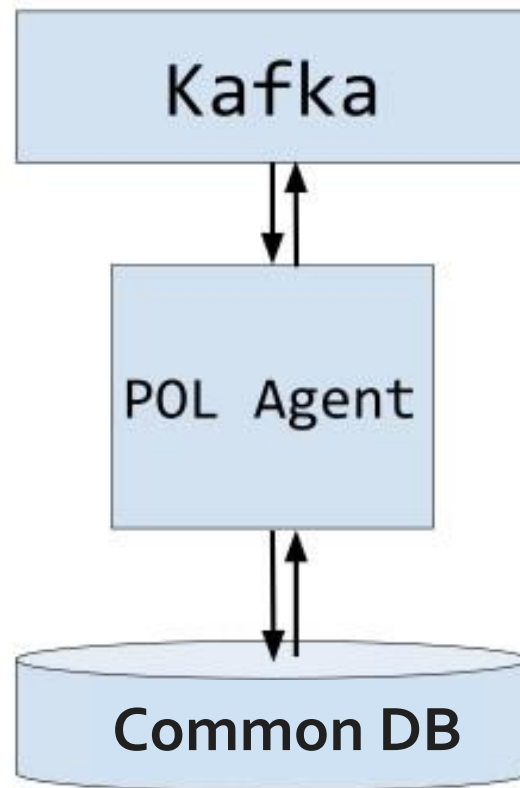
# Revisiting MON Architecture

Formal documentation: https://osm.etsi.org/gitlab/osm-architecture/osm-arch-doc/blob/master/04-mon.md

# POL Architecture

Formal documentation: https://osm.etsi.org/gitlab/osm-architecture/osm-arch-doc/blob/master/05-pol.md

# Auto Scaling & Alarms Architecture

When configuring alarms associated to scaling actions or just webhook notifications (through the VNFD), the following components interact.
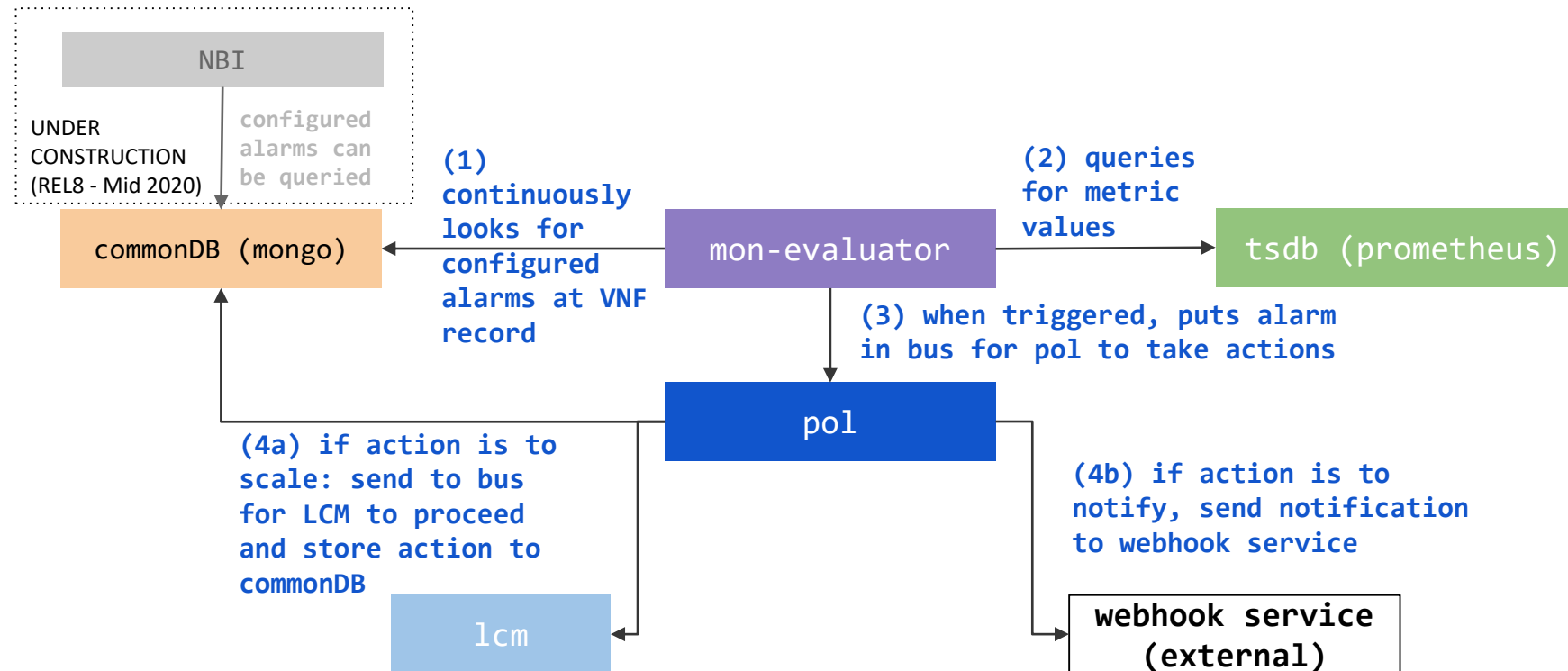
# Scaling Group Descriptor

```
scaling-group-descriptor:
    - max-instance-count: 1
      min-instance-count: 0
      name: vdu_autoscale
      scaling-policy:
    - cooldown-time: 120
      name: cpu_util_above_threshold
      scaling-criteria:
    - 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
```

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)

# Alarm Descriptor

```
-    alarm:
        -    actions:
             alarm:
             -    url: https://webhook.site/5706da10-04a0-4ab0-819b-cb524f71a367
             alarm-id: cpu-above-threshold
             operation: GT
             value: 80
             vnf-monitoring-param-ref: agw_cpu_util
```

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)

# Hands-on!

Auto Scaling & Alerting

# Let's play with metrics and (auto)dashboards!

- We will use slice created previously and stress the VDU of AGW VNF

- Check the AGW VM IP in the VIM http://172.21.247.1/



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 &
```

- Check CPU metrics in Grafana http://172.21.248.xx:3000/

- 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/](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
```

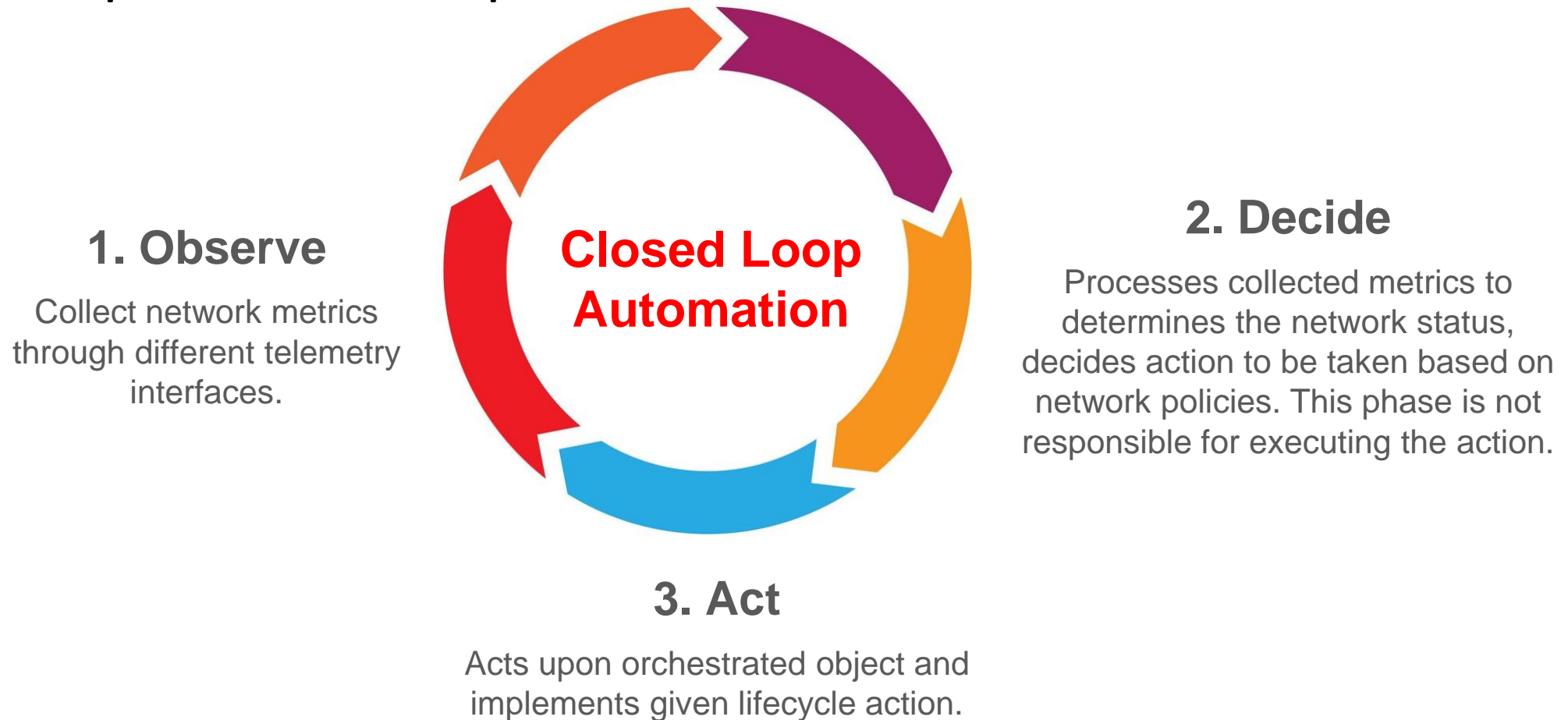- Observe descrease in CPU load and eventually a additional VDU is deleted.

**New Proposals**

OSM Service Assurance

# Future Vision – Proactive Closed Loop

Closed-loop automation powers autonomous networks.



**1. Observe**

Collect network metrics through different telemetry interfaces.

**Closed Loop Automation**

**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

OSM VNF Metric Exporters

| VIM Alarms (aodh) | VNF metrics (prometheus) | VNF Alarms (ex. snmp traps → webhook) |

**(1a)** Aodh notification

**(1b)** AlertManager notification

**(1c)** Custom webhooks

*In the future, POL could act upon combined alarms*

pol

**(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

lcm

webhook service (external)

# Auto-Scaling & Alarms – New Architecture

- Setting threshold on correlated metrics (multiple metrics)

- Move away from threshold to dynamic ML based anomaly detection
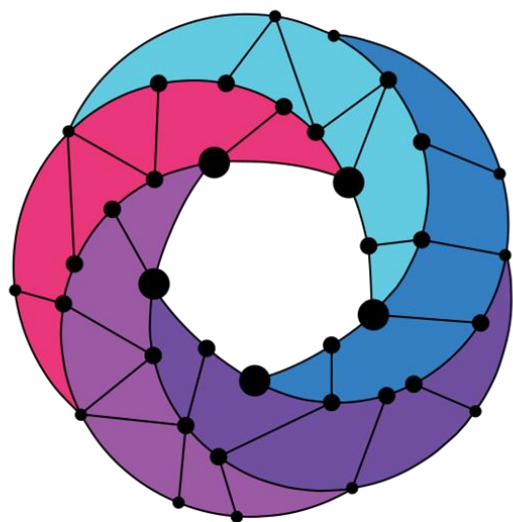
**Thank You !!**

Find us at:
osm.etsi.org
osm.etsi.org/wikipub

© ETSI