OSM Hackfest – Session 6
Fault & Performance Management
Gianpietro Lavado (Whitestack)
Performance and Fault Management capabilities are in an evolving phase in OSM. Some of the features are in experimental phase and may present significant changes in the following release cycle.
PM – What’s available in Release 4.0.1?

**OPTIONAL tools**

1. **NBI**
   - Client enables instant or continuous monitoring for selected VDU/metrics

2. **MON**
   - MON module returns metrics and puts them in “metric_response” topic

3. **Auxiliary “OSM Kafka To Prometheus Exporter” module**
   - Reads metrics from ‘metric_response’, translates them to prometheus-friendly format and exposes them as a web service in port 12340 to act as a Prometheus ‘target’

4. **Prometheus**
   - Reads metrics from web service and stores them

5. **Grafana**
   - Presents metrics in a friendlier GUI

**OSM KAFKA BUS**

- Contains the workflow of data from NBI to MON through OSM Kafka to Prometheus Exporter aux module, eventually visualized in Grafana.

© ETSI 2017
Basic Performance Management

• Main documentation is here:


• As of 4.0.1, the supported VIM is OpenStack with Gnocchi (others on its way!)

• Monitoring happens on a per-VDU basis.

• Supported metrics are:

  cpu_utilization, average_memory_utilization, disk_read_ops, disk_write_ops,
  disk_read_bytes, disk_write_bytes, packets_dropped_<nic number>,
  packets_received, packets_sent
Basic Performance Management

• Exporting a metric to the bus
  • Information to the bus:

    osm ns-metric-export --ns [NS NAME] --vnf [VNF MEMBER INDEX] --vdu [VDU ID] --metric [METRIC NAME]

  • Information exported to the bus can be seen in the MON docker logs:

    docker logs [MON container id] [--tail [XX lines]]

  • Basic ‘continuous monitoring’ can be activated with an ‘--interval’ option:


  • Linux functionality can be used to send those metrics to the background (just don’t forget about them 😊 )
PM Experimental Features (1)

- We can enable a Prometheus + Grafana stack to visualize metrics.
  - Install the PM stack to your environment
    
    # download the installer, make it executable and...
    
    ./install_osm.sh -o pm_stack
    
    ...if running Vagrant, you should forward ports 12340 and 3000
  
  - There is an adaptation layer provided by a ‘kafka_exporter’ container, which translates the metrics from the Kafka bus to a Prometheus ‘target’
    
    # once it’s there, go inside and make sure it successfully connected to Kafka
    
    docker exec [kafka_exporter_id] tail kafka-topic-exporter.log
    
  - You can then see active metrics being exported to Prometheus at http://[OSM_IP]:12340 (first time it takes some minutes)
PM Experimental Features (2)

- Once you see metrics being exported for Prometheus, Grafana will be able to build graphs out of them.
- If you are monitoring CPU or Memory, you will find the associated graphs already configured at the “OSM Sample Dashboard”
FM – What’s available in Release 4.0.1?

1. Client configures alarm associated to VDU metric threshold

2. PM creates alarms through MON

3. MON configures the alarm in the corresponding VIM or VNF

4. When alarms are triggered, VIM / N2VC plugin sends notification back to MON

5. MON puts the notification in the bus

6. PM sends a “scaling_action” message to the bus, and associated “log” to TCP port 5000 (if available)

OPTIONAL tools:
- Collect & Transform
- Search & Analyze
- Visualize & Manage

(Optional tools: logstash, elasticsearch, kibana)
Basic Fault Management

• Main documentation is here:


• Regarding logs, docker containers store their own logs for now.

• Regarding alarms, as of 4.0.1:
  • the supported VIM is OpenStack with Aodh (others on its way!)
  • we can configure alarms at the VIM level, that are not associated to a specific actions (scaling, etc)

• Alarming happens on a per-VDU basis, and are controlled by the VIM components.
Basic Fault Management: Logs

• To access logs, as of 4.0.1, it depends on the container.

• LCM logs
  • Path /app/log/lcm.log

• RO logs
  • Path /var/log/osm/openmano.log

• VCA logs
  • Run ‘juju debug-logs’ from the host

• MON/PM logs
  • docker logs [MON/PM id]
Basic Fault Management: Alarms

• Creating an alarm
  • Creating an alarm through the client:
    
  
    • This will configure an alarm at OpenStack, which can be seen with the following command:
      
      openstack alarm list
      openstack alarm show [ID]
    
  • Once the alarm triggers, it will notify MON about it, and a log will be shown in that container.
• We can enable a ELK stack stack to visualize logs and kafka events.
  • Install the ELK stack to your environment
    
    # download the installer, make it executable and...
    ./install_osm.sh –o elk_stack
    ...
    
    ...if running Vagrant, you should forward port 5601 (Kibana)
  
  • Make sure a default pattern was created, otherwiser you would have to create one through Kibana or by using the following script:

    curl -f -XPOST -H "Content-Type: application/json" -H "kbn-xrf: anything"
      "http://localhost:5601/api/saved_objects/index-pattern/logstash-*"
      -d "{"attributes":{"title":"logstash-*","timeFieldName":"@timestamp"}}"

    curl -XPOST -H "Content-Type: application/json" -H "kbn-xrf: anything"
      "http://localhost:5601/api/kibana/settings/defaultIndex"
      -d "{"value":"logstash-*"}"
FM Experimental Features (2)

• By default, Logstash will gather:
  • Any log sent to TCP port 5000
    • You can enable Policy Manager to send logs with:
      docker service update --env-add LOGSTASH_URI=logstash:5000 osm_pm
  • Kafka topic: ‘alarm_response’

• Most alarm-related activity will appear at Kibana
The near future...

• PM
  • OSM to have its own TSDB to store metrics selectively, in a more VNF-aware fashion and without the need of adaptation layers.
  • This also introduces historical metrics while minimizing impact to the bus.

• FM
  • Centralize logs and notifications
  • Every log to be exported to ELK (or other compatible systems)
Find us at:

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