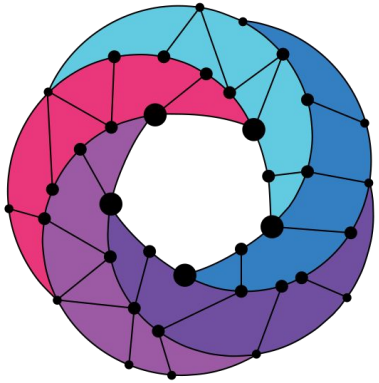


Open Source
MANO

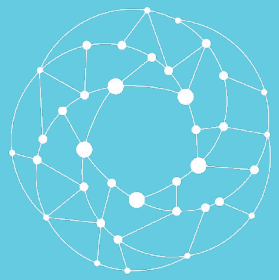
OSM Hackfest – Session 6
Performance & Fault Management

Benjamín Díaz (Whitestack)

ETSI

Introduction

Performance and Fault Management capabilities have made important progress in Release FIVE.
At the time of this Hackfest, Release FIVE has not been launched, so we will use an experimental, preview version.



Open Source
MANO

Installing a preview of Release FIVE

(as of 31-Oct-2018)

ETSI

Installing Release FIVE Preview

Removing Release FOUR installation

docker stack rm osm

→ destroys OSM docker containers

(wait until `docker stack ps osm` shows no containers)

docker system prune --all --volumes

→ removes images, volumes and networks

juju kill-controller osm

→ removes old charms and juju controller

Installing Release FIVE Preview

Open Source MANO

Installing Release FIVE from source code (latest)

git clone https://osm.etsi.org/gerrit/osm/devops

./devops/installers/full_install_osm.sh --test -b master

Installing Release FIVE from latest daily Docker images (faster)

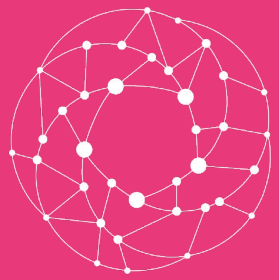
wget https://osm-download.etsi.org/ftp/osm-5.0-five/install_osm.sh

chmod +x install_osm.sh

./install_osm.sh -t releasefive-daily

when initializing LXD at interactive menu, mark default choices and 'no' to IPv6 subnet question

Note: If using a Vagrant image, enable port forwarding for port 3000, 9091 and 5601

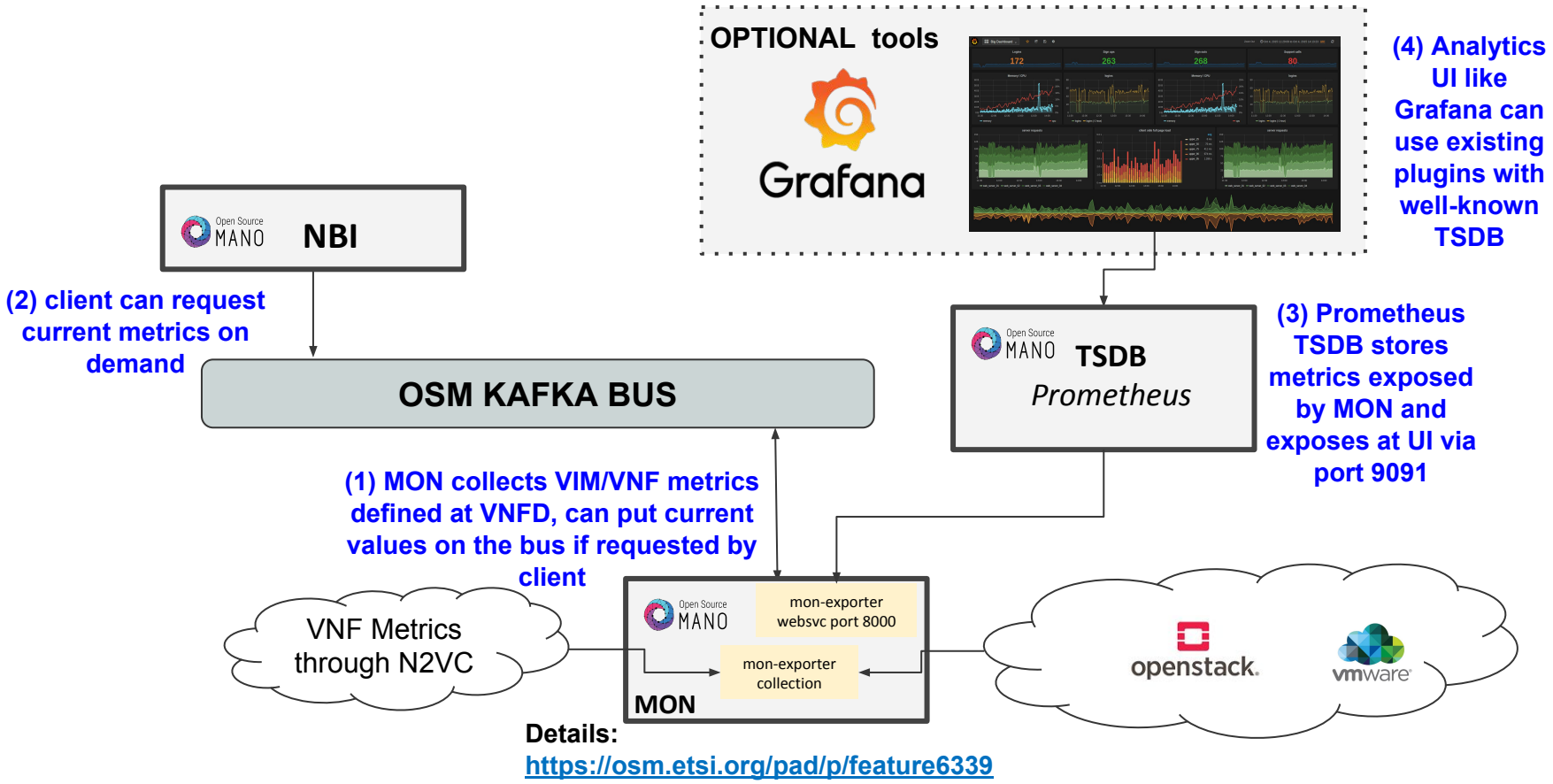


Open Source
MANO

Performance Management

- OSM “MON” Component -

PM – What's available at Release FIVE?



Main features

- Support for VIM metrics (related to VDUs)
  - OpenStack support ready
  - vROps support ready
  - AWS support pending
  - Supported metrics are cpu_utilization, average_memory_utilization, among others.
- Support for VNF-specific metrics.
  - Collection via proxy charms ‘metrics’ layer
  - Commands or API calls are executed from VCA to collect metrics every 5 minutes
- Monitoring happens on a per-VDU basis.

Model review - Sample VNFD

- VDU Metric Collection from VIM

```
vdu:
   id: apache_vdu
   ...
   monitoring-param:
         - id: "apache_cpu_util"
           nfvi-metric: "cpu_utilization"
...
monitoring-param:
-    id: "apache_vnf_cpu_util"
     name: "apache_vnf_cpu_util"
     aggregation-type: AVERAGE
     vdu-monitoring-param:
       vdu-ref: "apache_vdu"
       vdu-monitoring-param-ref: "apache_cpu_util"
```

nfvi-metric corresponds to a established metric name at MON

Model review - Sample VNFD

- VDU Metric Collection through VCA

```
vdu:
 - id: haproxy_vdu
   ...
   interface:
   - external-connection-point-ref: haproxy_mgmt
     mgmt-interface: true
  ...
  vdu-configuration:
    initial-config-primitive:
    ...
    juju:
      charm: testmetrics
    metrics:
      - name: load
...
monitoring-param:
-   id: "haproxy_load"
    name: "haproxy_load"
    aggregation-type: AVERAGE
    vdu-metric:
      vdu-ref: "haproxy_vdu"
      vdu-metric-name-ref: "load"
```

metrics "name" corresponds to a predefined metric name at the proxy charm

Model review - Sample VNFD

- VNF Metric Collection through VCA

```
vnfd:
...
mgmt-interface:
  cp: haproxy_mgmt
vnf-configuration:
  initial-config-primitive:
  ...
  juju:
    charm: testmetrics
  metrics:
    - name: users
...
monitoring-param:
-   id: "haproxy_users"
    name: "haproxy_users"
    aggregation-type: AVERAGE
    vnf-metric:
      vnf-metric-name-ref: "users"
```

metrics "name" corresponds to a predefined metric name at the proxy charm

Proxy Charm metrics layer

- Sample of ‘metrics.yaml’ file (root of charm folder)

```
metrics:
    users:
      type: gauge
      description: "# of users"
      command: who|wc -l
    load:
      type: gauge
      description: "5 minute load average"
      command: cat /proc/loadavg |awk '{print $1}'
```

Metrics collection in action

Walkthrough Example (VIM Metrics)

1. Download and review descriptors from here:

   hackfest_autoscale_vimmetric_nsd

   hackfest_autoscale_vimmetric_vnfd

2. Onboard them!

3. Make sure the 'vim-network-name' points to a "public" network your browser can reach.

4. Make sure you MON container matches the metrics granularity of the underlying VIM

   docker service update --env-add OS_DEFAULT_GRANULARITY=60 osm_mon

4. Launch the NS, you will have a LB (HA Proxy) and a Web server (Apache).

5. Visit the load balancer IP Address with your browser

Metrics collection in action

Walkthrough Example (VIM Metrics)

6. After a couple of minutes, visit the Prometheus TSDB GUI at OSM's IP address, port 9091.

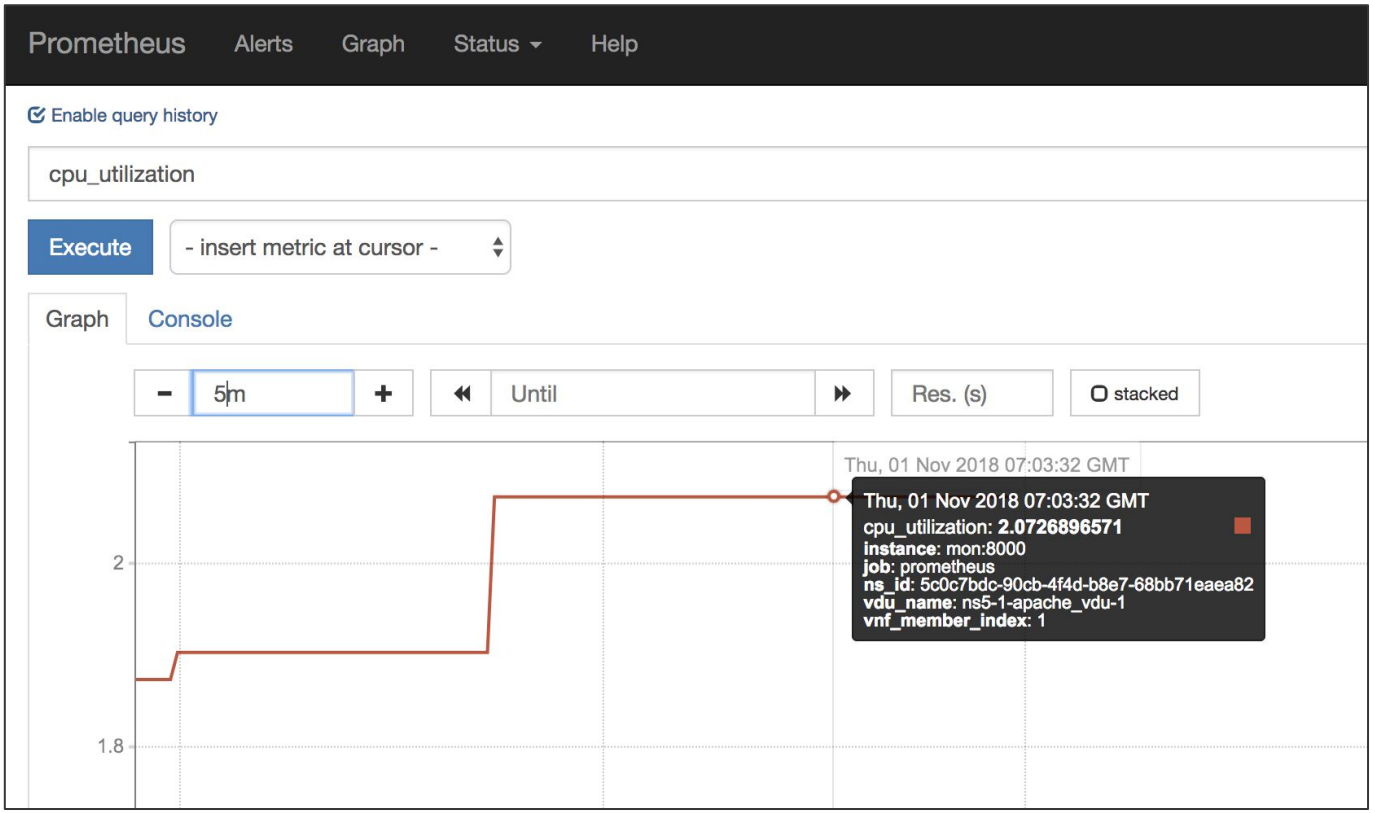
7. Validate that MON exporter "target" is properly connected at Status/Targets

8. Back in "Graph", type "cpu_utilization" or 'average_memory_utilization' and see if metrics are already there.

Metrics collection in action

Walkthrough Example (VIM Metrics)

9. Metrics should appear like this:



Metrics collection in action

Walkthrough Example (VIM Metrics)

10. Now let's add the optional Grafana component to see metrics in a friendlier way

From latest source code

git clone https://osm.etsi.org/gerrit/osm/devops # if not already there

git pull # if devops existed, to ensure latest code

docker stack deploy osm_elk --compose-file ./docker-compose.yml

Metrics collection in action

Walkthrough Example (VIM Metrics)

11. You should be able to visit Grafana at the OSM IP address, port 3000 (admin/admin)

12. There's a default sample dashboard at 'Manage → Dashboards' (to the left), that will show some predefined graphs connected to Prometheus TSDB

Metrics collection in action

Walkthrough Example (VDU Metrics from VCA)

1. Download and review descriptors from here:

   hackfest_autoscale_vnfmet_nsd

   hackfest_autoscale_vnfmet_vnfd

2. Onboard them!

3. Make sure the ‘vim-network-name’ points to a “public” network your browser can reach.

Metrics collection in action

Walkthrough Example (VDU Metrics from VCA)

6. After a couple of minutes, visit the Prometheus TSDB GUI at OSM's IP address, port 9091.

7. Validate that MON exporter "target" is properly connected at Status/Targets

8. Back in "Graph", type "load" or "users" and see if metrics are already there.

Metrics collection in action

Walkthrough Example (VDU Metrics from VCA)

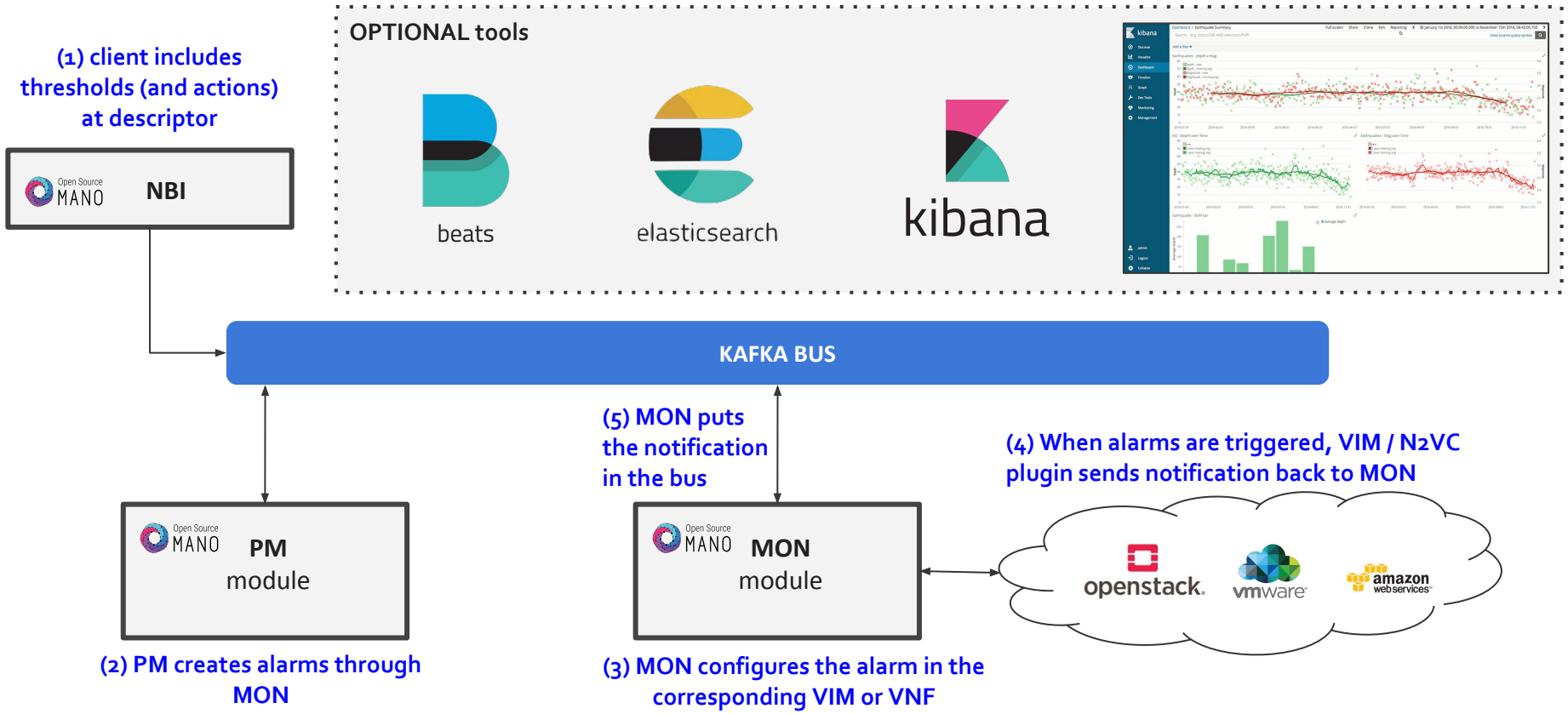
9. Access the VNF and execute ‘yes > /dev/null’. You should see users and load metrics changing in the next collection interval (5mins).

Open Source
MANO

Fault Management

- Docker logging & ‘POL’ Component -

FM – What's available in Release FIVE?



Main Features

- Logging
  - docker containers send their logs to stdout.
  - They can be checked on the fly using:
    - docker logs osm_mon.1…
    - docker logs osm_lcm.1…
  - They can also be found at: /var/lib/containers/[container-id]/[container-id].json.log
  - VCA logs
    - Run ‘juju debug-log’ from the host

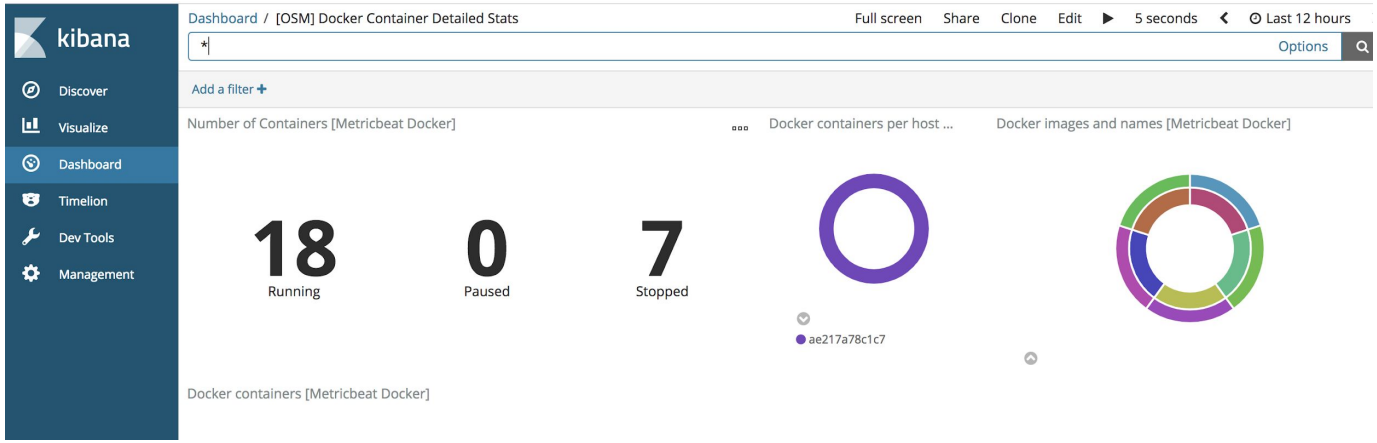
Main Features

- Alarming
  - Today, only alarms associated to metrics thresholds are supported, in the context of an **‘(auto)scaling descriptor’**
  - The supported VIM is OpenStack with Aodh (others on its way!)
  - Alarming happens on a per-VDU basis, and are controlled by the VIM components (Aodh, vROPS, etc.)

    → **this is migrating to a local alarm manager**

FM Experimental Features

- We can enable a “EBK” stack to visualize logs and metrics (Elasticsearch, Beats, Kibana)
  - **Filebeats** collects logs from all docker containers
  - **Metricbeats** collects metrics from the host, containers and applications, through modules.
  - **Elasticsearch** organizes information and provides a way to filter and further process it.
  - **Kibana** provides a way for visualizing information and building dashboards.



FM Experimental Features

- To this date, this stack is not yet part of the installer, so it has to be added directly using a docker compose file.

  cd devops/installers/docker/osm_elk

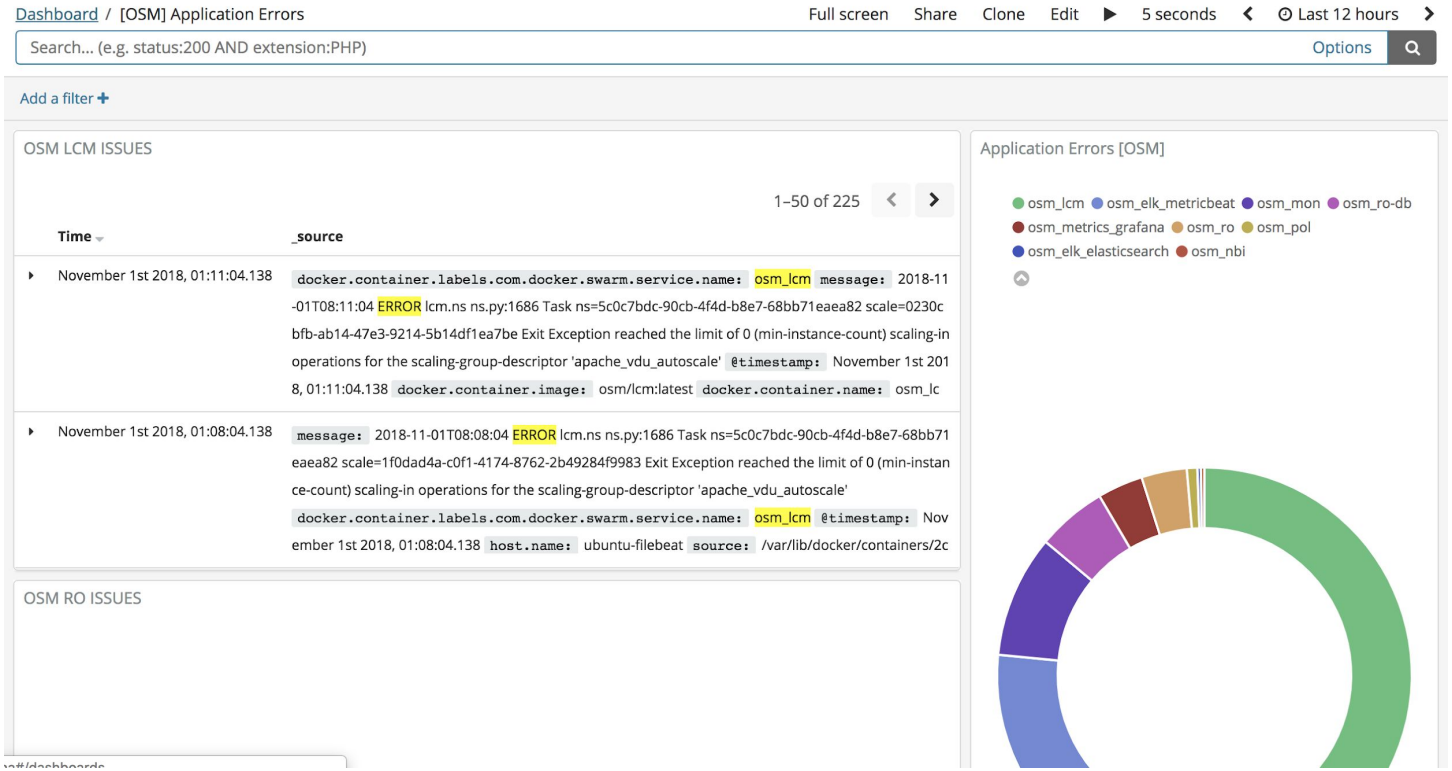
  docker stack deploy osm_elk --compose-file ./docker-compose.yml

  docker stack ps osm_elk # to check container status

- After it's up, visit it with your browser with the OSM IP, port 5601
- Import sample dashboards using this file: https://osm-download.etsi.org/ftp/osm-4.0-four/4th-hackfest/other/osm_elastic_dashboards.json (Management → Saved objects → Import)
- Go to 'Discover' and you will be asked to define one of the 'beats' as default 'index pattern', do so by selecting 'filebeat-*' and clicking

FM Experimental Features

- All metrics and logging activity will appear at Kibana.
- Navigate the sample OSM dashboards and provide feedback!



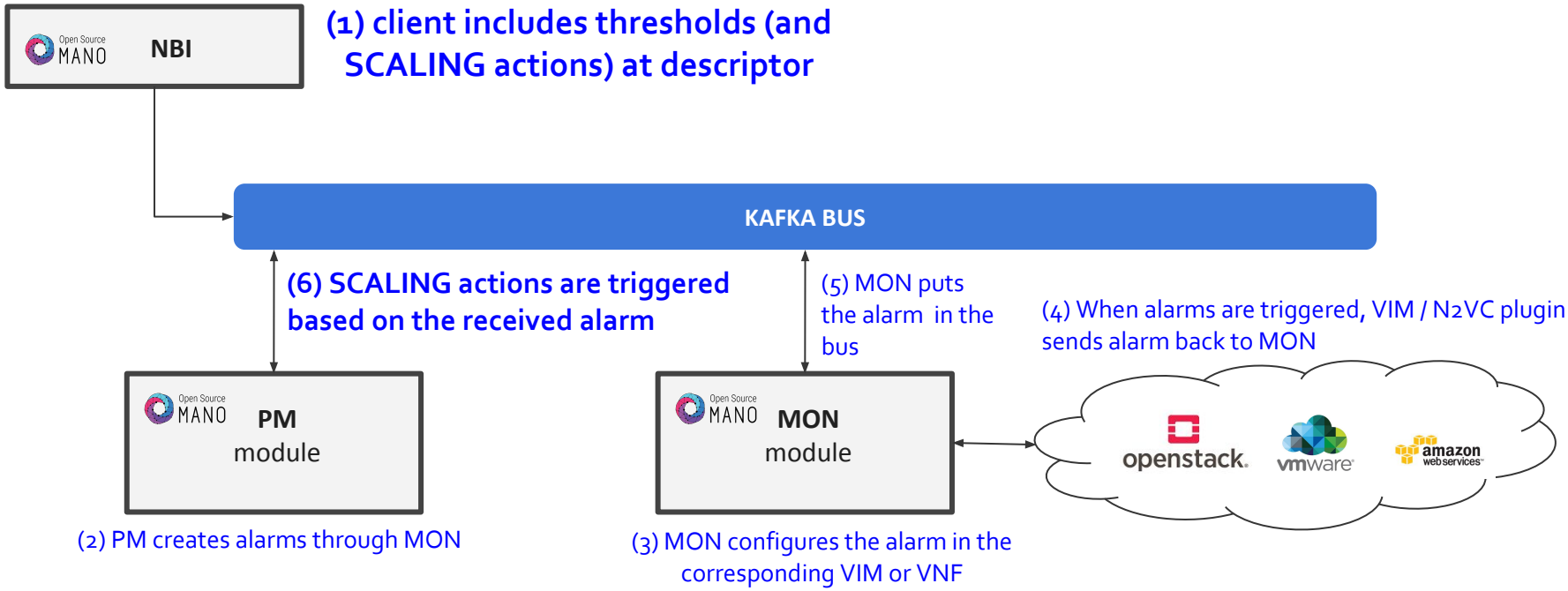
Open Source
MANO

Policy Management

- ‘POL’ Component -

ETSI

PM – What's available in Release FIVE?



Main Features

- Autoscaling
  - Scaling descriptors can be included and be tied to automatic reaction to VIM/VNF metric thresholds.
  - Supported metrics at this point are only VIM metrics, since OSM relies on the VIM's alarming system to detect a threshold has been crossed.
  - For Release FIVE launch, an internal alarm manager is being added, so that VNF metrics can also trigger threshold-violation alarms and scaling actions.

Model review - Sample VNFD

- VNF Scaling Descriptor (automatic, based on metrics)

```
scaling-group-descriptor:
-   name: "apache_vdu_autoscale"
    min-instance-count: 0
    max-instance-count: 10
    scaling-policy:
    -   name: "apache_cpu_util_above_threshold"
        scaling-type: "automatic"
        threshold-time: 10
        cooldown-time: 120
        scaling-criteria:
        -   name: "apache_cpu_util_above_threshold"
            scale-in-threshold: 20
            scale-in-relational-operation: "LT"
            scale-out-threshold: 80
            scale-out-relational-operation: "GT"
            vnf-monitoring-param-ref: "apache_vnf_cpu_util"
```

vnf-monitoring-param-ref corresponds to a predefined ‘monitoring param’

Model review - Sample VNFD

- Please note that scaling actions can also be triggered manually as long as there is a scaling descriptor of type ‘manual’
- The VNFD would look like this:

```
scaling-group-descriptor:
-   name: "apache_vdu_manualscale"
    min-instance-count: 0
    max-instance-count: 10
    scaling-policy:
    -   name: "apache_cpu_util_manual"
        scaling-type: "manual"
        threshold-time: 10
        cooldown-time: 120
```

Model review - Sample VNFD

- The API call for that is:
  - URL: POST to nslcm/v1/ns_instances/{{nsInstanceId}}/scale
  - Body

```
{"scaleType": "SCALE_VNF",
"scaleVnfData":
        {"scaleVnfType": "SCALE_OUT",
                "scaleByStepData": {
                        "scaling-group-descriptor": "apache_vdu_manualscale",
                        "member-vnf-index": "1"
}}}
```

Autoscaling in action

Open Source MANO

Walkthrough Example

1. **Remove previous instances**, and this time launch a web server, with load balancer and a client that can generate stress tests, using the following descriptors:

   hackfest_autoscale_vimmetric_wcli_nsd.tar.gz

   hackfest_autoscale_vimmetric_wcli_vnfd.tar.gz

2. Onboard them!

3. Make sure the 'vim-network-name' points to a "public" network your browser can reach.

4. Launch the NS, you will have a LB (HA Proxy), a Web server (Apache) and a Client (Ubuntu with Apache Bench)

5. Visit the load balancer IP Address with your browser

Autoscaling in action

Walkthrough Example

6. After a couple of minutes, visit the Prometheus TSDB GUI or Grafana GUI and validate that metrics have started being collected.

Autoscaling in action

Walkthrough Example

7. From the client, run a stress test towards your load balancer's IP address:

```
ab -n 5000000 -c 2 http://[HA-Proxy-IP]/test.php
```

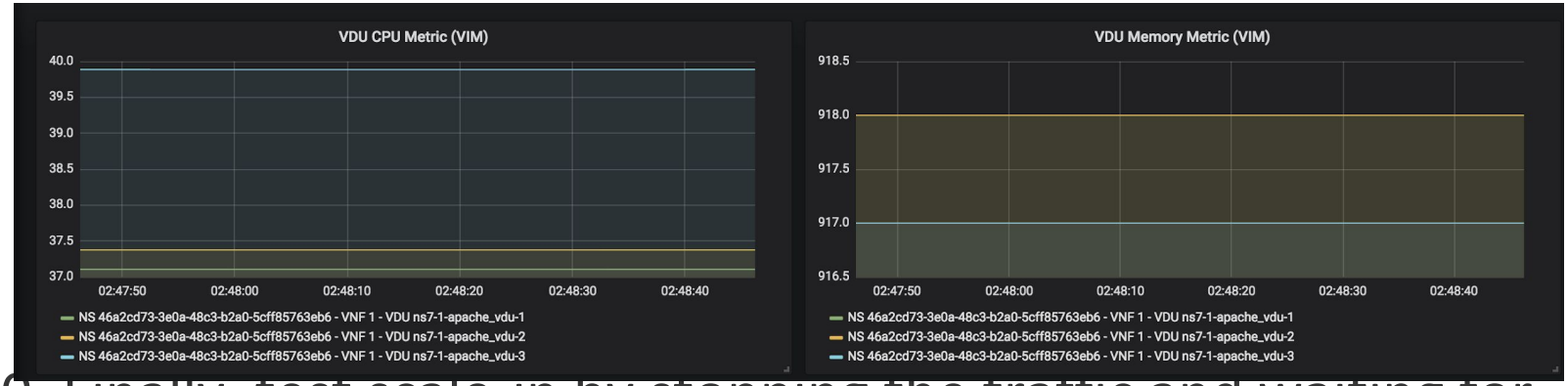
8. Watch the policy manager logs to detect for autoscaling instructions. CPU should start going up in a minute, validate that at the Grafana Dashboard.

11/01/2018 09:43:04 AM - osm_policy_module.common.lcm_client - INFO - Sending scale action message: {"links": {"self": "/osm/nslcm/v1/ns_lcm_op_occs/7734aae7-2114-4f12-bd3d-7ee4ee3ebee4", "nsInstance": "/osm/nslcm/v1/ns_instances/46a2cd73-3e0a-48c3-b2a0-5cff85763eb6"}, "nsInstanceId": "46a2cd73-3e0a-48c3-b2a0-5cff85763eb6", "id": "7734aae7-2114-4f12-bd3d-7ee4ee3ebee4", "operationParams": {"scaleVnfData": {"scaleVnfType": **"SCALE_OUT"**, "scaleByStepData": {"member-vnf-index": "1", "scaling-group-descriptor": "apache_vdu_autoscale"}}, "scaleType": "SCALE_VNF", "scaleTime": "2018-11-01T09:43:04.162733Z"}, "statusEnteredTime": 1541065384.162726, "operationState": "PROCESSING", "isCancelPending": false, "lcmOperationType": "scale", "_id": "7734aae7-2114-4f12-bd3d-7ee4ee3ebee4", "startTime": 1541065384.162726, "isAutomaticInvocation": true}

Autoscaling in action

Walkthrough Example

9. Instances of Apache Web Server should start appearing (up to 2 or 3 before it starts load balancing traffic accordingly), validate this at the OpenStack Network Topology and visiting the HAProxy IP address.



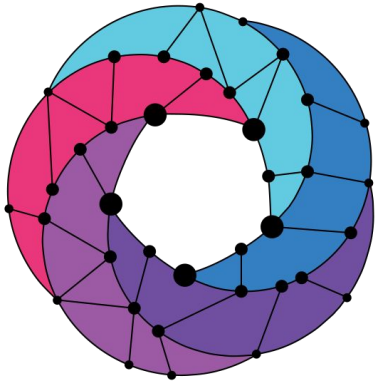
10. Finally, test scale-in by stopping the traffic and waiting for a couple of minutes.

Open Source
MANO

Roadmap & Discussion

Planned features for MON/POL

- Local alarm manager
- Closed-loop for VNF metrics (from N2VC/VCA)
- Extended fault management (VIM, VNFs)
- Additional actions at Policy Manager
- Root cause analysis component
- Dynamic Policy Management
- Real-time metric collection
- Log uniformization



Open Source
MANO

Find us at:

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

ETSI