Highly functional, but with some opportunities for enhancement:

- NBI tightly coupled to SO
- Difficult to add new modules to extend functionality
Architectural decisions
1) Keep layered communication to VIMs and VNFs

© ETSI 2018
Architectural decisions
2) Decouple NBI from NS and VNF LCM
Architectural decisions
3) Message bus, common DB and object storage
OSM Release FOUR Architecture

1. Unified Northbound Interface (SOL005-based), decoupled from LCM
2. Lightweight Life Cycle Manager (LCM)
3. Message bus for async communications
4. Common DB and object storage
5. Integrated components for policy, fault and performance management
6. Complete control through CLI and stand-alone new UI

Common Services

OSM IM Common Database (NoSQL)
Object Storage

Kafka bus

osmclient light-ui
New OSM’s NBI

OSM IM
NBI

LCM
OSM IM

VCA
OSM IM

RO
OSM IM

POL
MON
Extending the plugin model to other modules

**RO**
- OVIM plugin
- OST plugin
- VMW plugin
- AWS plugin
- ODL plugin
- ONOS plugin
- Floodl. plugin
- ODL WIM plugin
- TAPI plugin

**Message bus**
- FS plugin
- Kafka plugin

**Object Storage**
- FS plugin
- Ceph plugin

**MON**
- OST plugin
- VMW plugin
- AWS plugin
- VNF metrics plugin

**Common DB**
- FS plugin
- MONGO plugin

**NBI RBAC**
- Keystone plugin
Microservice architecture to enable extensibility

**OSM stack**
- NBI
- Kafka
- MON
- MONGO
- Zookeeper
- POL
- Light-UI
- LCM
- RO
- RO-DB

**LXD**
- VCA (juju controller)

**ELK stack**
- Elasticsearch
- Logstash
- Kibana

**Perf. Mon. stack**
- Prometheus
- Grafana

**Add here your stack**
- docker X
- docker y

© ETSI 2018
From design to operation
Design
Descriptors and packages, result of the design phase

- **VNF package**
  - VNFD
  - VNF artifacts
  - Additional metadata?

- **Resource description aspects**
  - VNF resource orchestration info (EPA resources and internal connectivity)

- **Management procedures**
  - VNF primitives
    - Day-1
    - Day-2
  - Charms

- **Additional info**
  - Icon, README, etc.
Design

Descriptors and packages, result of the design phase
Design

What is a charm?

• A charm is a set of actions and hooks
  • Actions are programs
  • Hooks are events/signals

• For commodity and reusability, those actions and hooks are grouped in layers

• A charm will always have one layer:
  • That layer has some actions and hooks
  • In addition, that layer can import other layers

• The resulting charm has all the actions and hooks from all the layers joined together, plus additional default actions and hooks (e.g. ‘config’ action)
Design
Implementing actions

@when('actions.touch')
def touch():
    err = ''
    try:
        filename = action_get('filename')
        cmd = ['touch {}'.format(filename)]
        result, err = charms.sshproxy._run(cmd)
    except:
        action_fail('command failed:' + err)
    else:
        action_set({'output': result})
    finally:
        clear_flag('actions.touch')
Primitives are mapped to charm actions in the descriptor

```json
vnfd:
  ...
  vnf-configuration:
    juju:
      charm: aaa
    initial-config-primitive:
    - seq: '1'
      name: getConfigurationFromEMS
      parameter:
      - name: ems-ip-address
        value: '/home/ubuntu/first-touch'
    config-primitive:
    - name: touch
      parameter:
      - name: filename
        data-type: STRING
    - name: addUser
      parameter:
      - name: userName
        data-type: STRING
```
Onboarding

Common Services
- Common Database (NoSQL)
- Object Storage

OSM IM

CRUD operations over VNF and NS packages go directly to the Common DB and Object storage, without travelling through the Kafka bus.
1. NBI creates record in common DB
2. NBI publishes message in Kafka: instantiate NS_id
3. LCM consumes the message and deploys the NS
4. Deployment and day-0 configuration
5. Day-1 configuration (parametrized)
1. NBI publishes message in Kafka
2. Consumer module depends on the specific primitive (day-2 operation primitives are consumed by LCM, on-demand metric export and alarm configuration are consumed by MON.)
Release FIVE preview on new monitoring capabilities
Instantiation process summary

Unified NBI lets users manage the NS lifecycle

OSM CLI

- **Module**: LCM
- **Function**: Takes descriptors and sends instructions to the bus

OSM KAFKA BUS

- **Module**: MON
- **Function**: Starts monitoring VNF according to descriptor!

- **Module**: POL
- **Function**: Creates alarms through MON based on thresholds

- **Module**: OSM Light UI

- **Module**: NBI
- **Function**: Takes care of VIM resources

- **Module**: OSM CLI
- **Function**: Unified NBI lets users manage the NS lifecycle
Demo 1: Metrics collection and visualization

(1) MON collects VIM/VNF metrics and stores in TSDB
(2) Prometheus reads and stores selected VNF metrics through N2VC-VCA modules
(3) Grafana presents selected VNF metrics

Optional tools

OSM TSDB

VNF metrics

MON module

openstack
vmware
amazon web services
Demo 2: Auto-scaling action!

(1) MON notifies POL about an alarm being triggered

(2) POL submits an action to the bus based on descriptor logic (i.e. SCALE)

(3) LCM instructs the system to generate the scaled instances

An optional ELK (or similar) logging analytics stack can inform about anything going on through the bus or logs
Auto-scaling description

VNF Descriptor indicates based on which metric and how to scale VDUs

```json
vdu:
...

  monitoring-param:
    - id: datavdu_cpu_util
      nfvi-metric: cpu_utilization

scaling-group-descriptor:
  - max-instance-count: 10
  - min-instance-count: 0
  name: scale_datavdu
  scaling-policy:
    - cooldown-time: 60
      name: auto_cpu_util_above_threshold
  scaling-criteria:
    - name: group1_cpu_util_above_threshold
      scale-in-relational-operation: LT
      scale-in-threshold: 20
      scale-out-relational-operation: GT
      scale-out-threshold: 80
      vnf-monitoring-param-ref: datavdu_vnf_cpu_util
  scaling-type: automatic
  threshold-time: 10
vdu:
  - count: 1
    vdu-id-ref: datavdu
```
Conclusions

• Regarding monitoring, OSM Release FIVE will:
  • Monitor VNF metrics
  • Support auto-scaling actions
  • Include its own TSDB
  • Enhance integration with external logging analysis stacks like ELK
Thank you!

Questions?

gerardo.garciadeblas@telefonica.com
glavado@whitestack.com