OSM IN PRACTICE
Gerardo García (Telefónica), Technical Steering Committee
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
Architectural decisions

2) Decouple NBI from LCM
Architectural decisions
3) Message bus, common DB and object storage

- Common Database (NoSQL)
- Object Storage
- NBI
- Any other module
- Kafka bus
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
Microservice architecture to enable extensibility (OSM Release FOUR)

**OSM stack**
- NBI
- Kafka
- MON
- MONGO
- Zookeeper
- POL
- Light-UI
- LCM
- RO
- RO-DB
- VCA
  (juju controller)

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

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

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

-netOSM docker network

© ETSI 2019
Microservice architecture in OSM Release FIVE

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

EBK stack
- Elasticsearch
- Kibana
- Filebeat
- Metricbeat

Perf. Mon. stack
- Grafana

Add here your stack
- docker X
- docker y

LXD
- VCA (juju controller)

netOSM docker network
OSM Release FIVE 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
- Auth
- TSDB

NBI

- osmclient
- light-ui

New OSM’s NBI

Kafka bus

- OSM IM
- LCM
- N2VC

VCA

- 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

© ETSI 2019
Multiple distribution formats

- Installer script (dockers or source code)
- Dockers: https://hub.docker.com/u/opensourcemanano/
  - Tag ‘latest’ for latest stable docker images
  - Tag ‘releasefive-daily’ for daily docker images
- Vagrant: https://app.vagrantup.com/osm
  - Latest Vagrant image: osm/releasefive
- Debian packages:
  - Stable: http://osm-download.etsi.org/repository/osm/debian/ReleaseFIVE
- Source code:
  - Gitweb: https://osm.etsi.org/gitweb/
  - Gerrit: https://osm.etsi.org/gerrit/#/admin/projects/
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

NS package

NSD
NS artifacts
Additional metadata?

Resource description aspects
NS topology

Management procedures
NS primitives
- Day-1
- Day-2
Charms

Additional info
Icon, README, etc.

© ETSI 2019
Design

What is a charm?

- A charm is a programmatic wrapping of the primitives
  - VNF charm -> VNF primitives
  - NS charm -> NS primitives

- Types of charms:
  - Proxy charms: run in LXD containers in the OSM host
  - Native charms: run in the VNF

- Proxy charms provide an abstraction layer that hides the complexity of the implementation behind
Demystifying charms
Mapping charms to descriptors

Descriptor
```
vnfd:
  ...
  vnf-configuration:
    juju:
      charm: vendor-epc
  config-primitive:
    - name: new-pcrf
      parameter: ...
    - name: new-hss
      parameter: ...
    - name: new-ippool
      parameter: ...
    - name: new-apn
      parameter: ...
    - name: full-config
      parameter: ...
```

```
actions.yaml
```
```
new-pcrf:
  description: ...
  params: ...
new-hss:
  description: ...
  params: ...
new-ippool:
  description: ...
  params: ...
new-apn:
  description: ...
  params: ...
full-config:
  description: ...
  params: ...
```

Charm
```
reactive/charm.py
```
```
@when('actions.new-hss')
def newHSS():
    ...
    hss_ip = action_get('hss_ip')
    ...
    cmd = "add_hss.sh " + hss_ip
    result, err = charms.sshproxy._run(cmd)
    ...

@when('actions.new-apn')
def newAPN():
    ...
    apn_name = action_get('apn_name')
    ...
    jsondata = {'name': apn_name}
    url = "https://" + epc_mgmt_ip + "api/new-apn"
    r = requests.post(url, jsondata)
    ...

@when('actions.full-config')
def fullConfig():
    ...
    #createFullConfigFileFromParams
    #copyFileToHTTPserver
    cmd = "apply_full_config " + url
    result, err = charms.sshproxy._run(cmd)
    ...
Onboarding

CRUD operations over VNF and NS packages go directly to the Common DB and Object storage, without travelling through the Kafka bus.
Read operations

Listing the available VNF and NS packages
NS instantiation
Deployment and day-0 configuration

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)
NS instantiation
Day-1 configuration

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)
Service operation (day-2)

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.)
Demo 1: onboarding and instantiation
OSM PoC framework

• The Open Source Mano community encourages members to design and deploy Proofs of Concept (PoCs) that are:
  • Relevant to contribute to further development of OSM
  • Showcase OSM in real-world useful scenarios and use cases

• Multi-vendor and/or component PoCs are welcome
How to submit an official OSM PoC

• Have at least one member that is an active OSM member or participant

• Send a formal PoC proposal to the OSM_TECH mailing list with the purpose to
  • Inform the OSM community
  • Get feedback from the OSM community

• OSM TSC will acknowledge receipt of the proposal and communicate approval to the OSM_TECH mailing list.

• Send relevant information to OSM MARCOM for publishing on OSM website.

• Present PoC results in TECH meetings with the purpose to:
  • Regularly provide updates to OSM community and get feedback
  • Complete the PoC - videos and supporting documentation

For more details, see the OSM wiki at: https://osm.etsi.org/wikipub/index.php/OSM_PoC_Framework
List of OSM PoCs

- OSM PoC#1 - DevOps in Service Chains & 5G Network Slices
  - PoC Team Members: Netrounds, Telenor, Intel, RIFT.io, Arctos Labs

- OSM PoC#2 - OSM with VMware Integrated Openstack in MEC architectures
  - PoC Team Members: VMware, RIFT.io, Fortinet, DataArt

- OSM PoC#3 - Heimdall Hybrid Web: Orchestrating enterprise web 5G vertical scenario
  - PoC Team Members: IT Aveiro, Universidade de Aveiro, Universidade Federal do Rio Grande, Wavecom

- OSM PoC#4 - Charmed Open5GCore Deployment with OSM Rel FIVE
  - PoC Team Members: Canonical, TNO

- OSM PoC#5 - Placement of Workloads in Distributed Cloud Networks
  - PoC Team Members: Arctos Labs, Netrounds, WindRiver, Telenor

- OSM PoC#6 - 5G Network Slice Orchestration with OSM
  - PoC Team Members: Telefonica, Telenor, Intel, Mavenir, Oracle, Ixia, Indra, Altran, Tech Mahindra, RedHat

List of OSM PoCs: https://osm.etsi.org/wikipub/index.php/OSM_PoCs
Objective

Deploy 2 Network Slices with some input parameters and operate them through Day-2 operations at Network Slice level

- **Deployment**: Each slice is modeled as a set of Network Services connected by networks or VLDs
  - Simple input parameters determine the type of slice to be created on demand
  - The 2 slices share some Network Services (shared NS Subnets)
    - If the shared NS was already deployed, it won’t be deployed again
    - It will be reused, but initial configuration for the second Network Slice can still be done in the shared NS to let it know that new elements are present.

- **Operation**: Running Day-2 primitives at Network Slice level (handled as a single object)
  - OSM, behind the scenes, maps them to a sequence of calls to NS primitives, which, in turn, are derived in calls to VNF primitives
New construct for composition of NS

NS1
NS2
NS3
NS3
NS5
NS6

Shared/non-shared NS

NST1

NS2
NS3
NS3
NS5
NS6

NST2

Actually running
PoC #6: 5G Network Slice Orchestration with OSM

1st Network Slice: eMBB

Slice #1: eMBB

Core (shared) Subnet

Radio (shared) Subnet

Mavenir Core (EPC + HSS)

S1

S1u

S5u

S5c

MGMT

SGi

Gx

SGi (INET)

Oracle PCRF

PCRF (shared) Subnet
PoC #6: 5G Network Slice Orchestration with OSM
2nd Network Slice: URLLC

Slice#2: URLLC

Core (shared) Subnet

Radio (shared) Subnet

PGW2 (dedicated) Subnet

© ETSI 2019
Seeing the full picture

Slice #1: eMBB

- Mavenir Core (EPC + HSS)
- Oracle PCRF
- RAN
- S1
- S5u
- S5c
- S1mm
- S1u
- SGi
- Gx
- PGW2
- Core (shared) Subnet
- Radio (shared) Subnet
- PGW2 (dedicated) Subnet
- PCRF (shared) Subnet

Slice #2: URLLC

- Mavenir PGW2
- MGMT
- SGi (INET)
- Gx
- S5u
- S5c
- Subnet
Demo 2: 5G Network Slices (PoC#6)
Key takeaways

- OSM is constantly incorporating best SW practices:
  - Microservice architecture
  - Proper decoupling of components depending on the NB operations
  - Multiple distribution options

- Deployment and management procedures of VNF, NS and Slices are encapsulated into packages (descriptor + artifacts)

- The main goal of PoC framework is to showcase OSM in real-world useful scenarios and use cases

- A PoC on 5G Network slices with OSM was recently showcased in MWC
Thank you!

Questions?

gerardo.garciadeblas@telefonica.com