OSM MR Hackfest – Hack 1
OSM Architecture & Installation

Gianpietro Lavado (Whitestack)
Cyndi Alarcón (Whitestack)
Francisco Rodriguez (Indra)
OSM Architecture Review
OSM Architecture overview

Common Services

- OSM IM
- Object Storage
- TSDB (Metrics)
- Auth

- OSM IM
- Common Database (NoSQL)

- Kafka bus
- MON
- PLA
- POL
- MON
- light-ui

- osmclient

- LCM
- N2VC

End to end orchestrator: LifeCycle Management (LCM) component

Integrated components for placement, policy, fault and performance management

Complete control through CLI and stand-alone UI

Unified Northbound Interface

unified message bus for async communications

common DB, storage, authentication and TSDB systems
When dealing with the creation, modification or deletion of users, projects and roles, the interacting components vary according to the selected backend.
When reading, uploading, modifying and deleting a Network Slice Template, Network Service Package or VNF Package, the following components interact.

CLI Example: `osm vnfpkg-create myvnfpackage.tar.gz`
Adding VIM/SDNC Sessions

When registering new sessions with VIMs or SDN Controllers, the following components interact.

CLI Example: osm vim-create --name myVIM --user myuser --password myprecious --auth_url http://172.21.7.5:5000/v3 --tenant mytenant --account_type openstack
Adding a K8 Cluster

When registering new sessions with Kubernetes clusters, the following components interact.

CLI Example: `osm k8scluster-add --creds myCredentials.yaml --version '1.15' --vim myVIM --description "My K8s cluster" --k8s-nets '{"net1": "myVIMnet"}' myK8Cluster`
When launching a new instance of a Network Service or Slice Instance (n x VNFs), the following components interact.

**CLI Example:** `osm ns-create --ns_name myNS --nsd_name myNSD --vim_account myVIM`
KNF Instantiation

When launching a new instance of a Network Service or Slice Instance (n x VNFs), the following components interact.

**CLI Example:** osm ns-create --ns_name myNS --nsd_name myNSD --vim_account myVIM

![Diagram showing interaction between components](image-url)
Instantiating with Primitives

When launching a new instance of a Network Service or Slice Instance (n x VNFs), with Day-1/2 automation, direct interaction with the NF is needed, so the following components interact.

CLI Example of Day-2 primitive: `osm ns-action myNS --vnf_name 1 --action_name myAction`
Instantiating with Placement

When launching a new instance of a Network Service or Slice Instance (n x VNFs), with placement support, the following components interact.

CLI Example: `osm ns-create --ns_name myNS --nsd_name myNSD --vim_account myVIM --config '{placement-engine: PLA, placement-constraints: {...}}'`
When launching a new instance of a Network Service or Slice Instance (n x VNFs) which is described with the collection of VNF Metrics that come from infrastructure (NFVI), the following components interact.

1. The commonDB (mongo) continuously looks for active VNFs with metrics.
2. The VIM (external) makes API calls to collect metrics.
3. The mon-collector exporter looks for active metrics.
4. The grafana tsdb (prometheus) reads and presents the vim_status metrics.
When launching a new instance of a Network Service or Slice Instance (n x VNFs) which is described with the collection of VNF Metrics that come from the VNF itself, the following components interact.

1. **lcm**
   - Automatically after instantiation
   - Initiates metrics-collection primitive

2. **vca**
   - Juju-metrics primitive
   - Collects active VNFs with juju metrics

3. **tsdb (prometheus)**
   - Continuously looks for active VNFs

4. **commonDB (mongo)**
   - API calls to VIM, to collect metrics

5. **mon-collector**
   - Exporter
   - Looks for active metrics

6. **grafana**
   - Reads & presents vim_status metrics

© ETSI 2020
When configuring alarms associated to scaling actions or just webhook notifications (through the VNFD), the following components interact.

1. NBI continuously looks for configured alarms at VNF record.
2. Queries for metric values.
3. When triggered, puts alarm in bus for pol to take actions.
4a. If action is to scale: send to bus for LCM to proceed and store action to commonDB.
4b. If action is to notify, send notification to webhook service.
When creating Projects or Network Services, Grafana dashboards are created automatically and the following elements interact.

- **commonDB (mongo)** continuously looks for new projects or NS with metrics.
- **mon-dashboarder** creates Project dashboard if a project is present. Creates NS dashboard (with sample graphs) if a NS is present. Continuously deletes obsolete dashboards.
- **grafana**
A general approach for OSM Troubleshooting is to first look for error messages in “show” commands, as in:

```bash
osm ns-show [ns]
osm vim-show [vim]
```

Besides that, knowing which components interact for each operation, you can troubleshoot by looking at the logs of each component. All troubleshooting tips are being documented in the user guide, here:

https://osm.etsi.org/docs/user-guide/09-troubleshooting.html
OSM Installation methods
1. OSM can be installed in a single server or VM with the following requirements:

<table>
<thead>
<tr>
<th></th>
<th>CPU</th>
<th>RAM</th>
<th>DISK</th>
<th>NIC</th>
<th>Internet</th>
<th>SO</th>
</tr>
</thead>
<tbody>
<tr>
<td>MINIMUM</td>
<td>2</td>
<td>4GB</td>
<td>20GB</td>
<td>1</td>
<td>Yes</td>
<td>Ubuntu18.04 (64-bit variant required)</td>
</tr>
<tr>
<td>RECOMMENDED</td>
<td>2</td>
<td>8GB</td>
<td>80GB</td>
<td>1</td>
<td>Yes</td>
<td>Ubuntu18.04 (64-bit variant required)</td>
</tr>
</tbody>
</table>

2. Once you have prepared the host with the previous requirements, all you need to do is:

```bash
chmod +x install_osm.sh
```
OSM Installation methods

OSM R7 can be installed using these main options:

- Docker Swarm: 
  
  ```
  ./install_osm.sh
  ```

- Kubernetes: 
  
  ```
  ./install_osm.sh -c k8s
  ./install_osm.sh -c charmed
  ```

For more information go to https://osm.etsi.org/docs/user-guide/01-quickstart.html#installing-osm
Hands-on: OSM Installation over Kubernetes
Hands-on: OSM Installation over Kubernetes

Scenario

K8S Single Cluster

- **kube-system**
- **monitoring**
- **osm**

Where:
- **Namespace**
- **Pod**
- **Container**

OSM GUI

2. Check the IP of your VM at [http://172.21.247.1/project/instances/](http://172.21.247.1/project/instances/), then access it through SSH
   user: ubuntu
   Password: osm4u

3. Now, let’s follow the user-guide at: [https://osm.etsi.org/docs/user-guide/01-quickstart.html#installing-osm](https://osm.etsi.org/docs/user-guide/01-quickstart.html#installing-osm)

   ```bash
   ```

4. Make the installer executable

   ```bash
   chmod +x install_osm.sh
   ```
5. Run the installer with -c k8s flag

./install_osm.sh -c k8s

6. You will be asked to confirm the installation of the following components:

The installation will do the following
1. Install and configure LXD
2. Install juju
3. Install docker CE
4. Disable swap space
5. Install and initialize Kubernetes as pre-requirements.

Do you want to proceed (Y/n)? Y
7. When installation is finished, execute the following commands to check k8s installation:

```bash
kubectl get nodes
kubectl get namespaces
kubectl get pods --all-namespaces
kubectl get all -n kube-system
kubectl get all -n osm
kubectl describe pod light-ui-xyz -n osm
```

8. Test the OSM client:

```bash
osm --help
osm user-list
```
9. Go to OSM GUI at http://<VM-IP> and access with admin/admin
Hands-on: OSM System Monitoring
Do not do this yet

The usual way to go

```bash
./install_osm.sh -c k8s --k8s_monitor
Access dashboard: http://<osm-host>:3000
```

- Kubernetes health
- OSM component status
- OSM component resource consumption
Getting the latest release candidate

• Note1: We are assuming you did not include the switch “–k8s_monitor” in the previous installation. Otherwise please do now installers/uninstall-k8s-monitoring.sh after step 2

• Note 2: We are assuming you used the switch “-c k8s”

STEPS

1. git clone "https://osm.etsi.org/gerrit/osm/devops"
2. cd ~/devops/
3. git pull "https://osm.etsi.org/gerrit/osm/devops" refs/changes/72/8372/10
4. cd ~/devops/installers/
5. ./full_install_osm.sh -o k8s_monitor -D $HOME/devops
What is installed

Resources monitored

- Kubernetes core
- "osm" namespace
- OSM pods
- Host

Monitoring components

- "monitoring" namespace
- Prometheus Operator & exporters
- Dashboards

Monitoring OSM
• Available in the k8s deployment of OSM.
• There is a similar feature for the docker swarm (classic) deployment of OSM (not to be discussed here)
• Aimed at monitoring OSM infrastructure, NOT the VNF/NS deployed
• Implementation based on Prometheus operator (Helm chart), plus some Prometheus exporters (node, Kafka, mysql, mongodb), in “monitoring” namespace
More implementation details

Resources monitored

Adapters to Prometheus

Monitoring pods

Configuration

- Kubernetes pods
- OSM pods
- Host OS
- Prometheus operator
- Prometheus CR
- ServiceMonitor CR
- Dashboards
- ConfigMaps

**Resources monitored**
- Kubernetes pods
- OSM pods
- Host OS

**Adapters to Prometheus**
- Mongodb adapter
- Mysql adapter
- Kafka adapter
- Node adapter

**Monitoring pods**
- Prometheus pod
- Prometheus CR

**Configuration**
- "monitoring" namespace

"Change here to customize the dashboards"
http://ip-address>:3001 (admin:prom-operator)

- **Kubernetes cluster**
  upstream dashboards in Prometheus operator helm chart

- **Open Source MANO**
  Specific dashboards for OSM
  - OSM Status summary
  - Hosts
  - Kafka, mongodb, mysql
OSM Status summary

- Failed pods / Failed nodes (if any)
- K8s resources requested
- OSM components status (up/down)
- CPU/Memory per OSM component
Hosts status

Summary (uptime, used memory, CPU, disk)
- CPU usage
- Disk usage
- Memory usage
- Network usage
Mongo, mysql and Kafka dashboards

Kafka
- Messages produced/consumed
- Lag by consumer group
- Partitions per topic

Mongodb
- Connections
- Document operation stats
- Network operations

Mysql
- Connections
- Disk occupation (indexes, tables)
- Network operations
Inspecting the “monitoring” namespace

• See all the objects deployed in the monitoring namespace
  • kubectl --namespace monitoring get all

• In particular, the dashboards are stored as configmaps
  • kubectl --namespace monitoring get configmap

• Servicemonitors specify what is to be scrapped by Prometheus
  • kubectl --namespace monitoring get servicemonitor
Let’s play a little

Force no pods running nbi

```shell
kubectl scale --namespace osm --replicas=0 deployment/nbi
```
We are going to improve the dashboard

Go to Edit -> Visualization

Coloring: Activate “value”

Gauge: Deactivate “show”

Value Mappings: Set value mappings

null -> error

0 -> error

1 -> ok
And make the change persistent

• Get the summary dashboard configmap definition to your computer
  
  scp  
  ubuntu@<ip-addr>:/home/ubuntu/devops/installers/k8s/summary-dashboard.yaml.

• In grafana, “export” ▶ “json”, and copy in the data contents of the .yaml file defining the configmap

• Upload the modified file
  • scp summary-dashboard.yaml  
    ubuntu@<ip-addr>:/home/ubuntu/devops/installers/k8s

• Update the dashboard
  • kubectl –n monitoring apply –f summary-dashboard.yaml
OSM Packages overview
What is a package in NFV?

Packages contain the information that orchestrators need to launch a network service. The are basically two types of packages.

The VNF Package

- It contains the characteristics of the VNF, for example:
  - The software image(s) it needs.
  - Compute resources.
  - Network connections between its components (Internal Virtual Links)
  - Performance requirements.
  - Automation scripts.
- Its main element is the VNF Descriptor (VNFD)
- It is built and provided by the VNF vendor.
- This applies in a similar way to new conceptual kinds of Network Functions (NFs), like a Physical NF (PNF), a Containerized NF (CNF), a Kubernetes-based NF (KNF), and Hybrid Network Package (HNF), etc.
What is a package in NFV?

Packages contain the information that orchestrators need to launch a network service. There are basically two types of packages.

**The Network Service Package**

- **NS (Network Service)**
  - **VNF Package**
    - **VDU** (Virtualisation Deployment Unit)
      - **VM Image**
    - **Metadata**
    - **VNFD (VNF Descriptor)**
      - Connection points
      - Lifecycle Events
      - Virtual Links
    - **Scripts**
  - **NSD - Network Service Descriptor**
    - VNF Descriptors
    - Virtual Link Descriptors
    - Dependencies

- **It contains the characteristics of the Network Service, for example:**
  - The VNF(s) it needs.
  - Network connections between VNFs (external Virtual Links)
- **Its main element is the NS Descriptor (NSD)**
- **It is built by the operator from VNFs that conform the Network Service that needs to be provided.**
Packages in OSM

Package descriptors in OSM are modeled in an increasing alignment to ETSI NFV standards (SOL006). Everything that can be put in a descriptor to model a VNF or NS, is present at OSM’s Information Model, maybe the richest model of the NFV MANO industry.

Visit this link to navigate the model: https://osm.etsi.org/docs/user-guide/11-osm-im.html
The NS Package is the one actually being launched in OSM. It requires constituent VNF Packages to be present in the system.

```
nsd:nsd-catalog:
  nsd:
    - id: hackfest_basic-ns
      name: hackTest_basic-ns
      short-name: hackfest_basic-ns
      description: Simple NS with a single VNF and a single VL
      version: '1.0'
      logo: osm.png
      constituent-vnfd:
        - vnfd-id-ref: hackfest_basic-vnf
          member-vnf-index: '1'
      vld:
        - id: mgmtnet
          name: mgmtnet
          short-name: mgmtnet
          type: ELAN
          mgmt-network: 'true'
          vnfd-connection-point-ref:
            - vnfd-id-ref: hackfest_basic-vnf
              member-vnf-index-ref: '1'
              vnfd-connection-point-ref: vnf-cp0
```

Network Service “hackfest_basic-ns”

It needs VNF “hackfest_basic-vnf” to be present

It will put the VNF in a new network called ‘mgmtnet’
The VNF Package is the one describing a given Network Function. It requires VIM/NFVIs to support whatever characteristic is being required through its descriptor.

```
vnf:vnfd-catalog:
    vnfd:
        - id: hackfest_basic-vnf
          name: hackfest_basic-vnf
          short-name: hackfest_basic-vnf
          version: '1.0'
          description: A basic VNF descriptor w/ one VDU
          logo: osm.png
          connection-point:
              - name: vnf-cp0
                type: VPORT
              id: hackfest_basic-VM
              name: hackfest_basic-VM
              image: ubuntu1604
              alternative-images:
                  - vim-type: aws
                    image: ubuntu/images/hvm-ssd/ubuntu-focal-17.10-amd64-server-20180509
                    count: '1'
                    vm-flavor:
                        vcpu-count: '1'
                        memory-mb: '1024'
                        storage-gb: '10'
              interface:
                  - name: vdu-eth0
                    type: EXTERNAL
                    virtual-interface:
                        type: PARAVIRT
                        external-connection-point-ref: vnf-cp0
              mgmt-interface:
                  cp: vnf-cp0
```

VNF “hackfest_basic-vnf”

It has one VDU (VM) that requires an image called ‘ubuntu1604’, and a flavor with 1 vCPU, 1GB RAM and 10GB of storage.

It has one interface, exposed to the Network Service as external Connection Point “vnf-cp0”
Once NS Packages and their constituent VNF Packages are present in the system, and at least a VIM is registered, a Network Service can be launched.
Hands-on: Integrating a VIM & Instantiating a basic Network Service
1. Create a VIM in OSM via CLI

```
osm vim-create --name openstack-site-hackfest-x --user osm_hackfest_x --password <Pass> --auth_url http://<VIM-IP>:5000/v3 --tenant osm_hackfest_x --account_type openstack --config='{security_groups: default}'
```

2. Validate the VIM creation. The status should be ENABLED

```
osm vim-list
osm vim-show openstack-site-hackfest-x
```
Hands-on: Integrating a VIM

1. Create a VIM in OSM via GUI
2. Go to VIM accounts -> add new VIM
   ○ Name: openstack-site-hackfest-
   ○ Type: Openstack
   ○ VIM URL: http://<VIM-IP>:5000/v3
   ○ VIM Username: osm_hackfest_
   ○ VIM Password: ********
   ○ Tenant name: osm_hackfest_
3. Click in Create button
4. Validate the VIM creation.
   The status should be ENABLED
Hands-on: Launching your first NS

VNFD Descriptor and Diagram

VNF name: hackfest_basic-vnf

VNF:
- Name: hackfest_basic-vnf
- logo: osm.png
- mgmt-interface:
  - cp: vnf-cp0
  - name: hackfest_basic-vnf
  - short-name: hackfest_basic-vnf
- vdu:
  - alternative-images:
    - image: ubuntu/images/hvm-ssd/ubuntu-artful-17.10-amd64-server-20180509
      vim-type: aws
    - id: hackfest_basic-VM
      image: ubuntu1604
      interface:
        - external-connection-point-ref: vnf-cp0
          name: vdu-eth0
          type: EXTERNAL
          virtual-interface:
            type: PARAVIRT
            name: hackfest_basic-VM
            vm-flavor:
              memory-mb: '1024'
              storage-gb: '10'
              vcpu-count: '1'
              version: '1.0'

vnfd:vnfd-catalog:
- vnfd:
  type: VPORT
  - connection-point:
    name: vnf-cp0
    type: VPORT
  description: A basic VNF descriptor w/ one VDU
  id: hackfest_basic-vnf
  logo: osm.png
  mgmt-interface:
  - cp: vnf-cp0
  name: hackfest_basic-vnf
  short-name: hackfest_basic-vnf
  vdu:
  - alternative-images:
    - image: ubuntu/images/hvm-ssd/ubuntu-artful-17.10-amd64-server-20180509
      vim-type: aws
    - id: hackfest_basic-VM
      image: ubuntu1604
      interface:
      - external-connection-point-ref: vnf-cp0
        name: vdu-eth0
        type: EXTERNAL
        virtual-interface:
          type: PARAVIRT
          name: hackfest_basic-VM
          vm-flavor:
            memory-mb: '1024'
            storage-gb: '10'
            vcpu-count: '1'
            version: '1.0'

VNF name: hackfest_basic-vnf

VDU
- Name: hackfest_basic-VM
- Image: ubuntu1604
- Flavor:
  - 1 CPU
  - 1GB RAM
  - 10 GB Disk

© ETSI 2020
Hands-on: Launching your first NS

NSD Descriptor and Diagram

nsd:nsd-catalog:
  nsd:
    - constituent-vnfd:
      - member-vnf-index: '1'
        vnfd-id-ref: hackfest_basic-vnf
      description: Simple NS with a single VNF and a single VL
      id: hackfest_basic-ns
      logo: osm.png
      name: hackfest_basic-ns
      short-name: hackfest_basic-ns
      version: '1.0'
    vld:
      - id: mgmtnet
        mgmt-network: 'true'
        name: mgmtnet
        short-name: mgmtnet
        type: ELAN
        vnfd-connection-point-ref:
          - member-vnf-index-ref: '1'
            vnfd-connection-point-ref: vnf-cp0
            vnfd-id-ref: hackfest_basic-vnf

VDU
- Name: hackfest_basic-VM
- Image: ubuntu1604
- Flavor:
  - 1 CPU
  - 1GB RAM
  - 10 GB Disk
Hands-on: Launching your first NS

1. Download the nsd and vnfd packages

   wget http://osm-download.etsi.org/ftp/osm-5.0-five/6th-hackfest/packages/hackfest_basic_vnf.tar.gz
   wget http://osm-download.etsi.org/ftp/osm-5.0-five/6th-hackfest/packages/hackfest_basic_ns.tar.gz

2. Create the NSD and VNFD in OSM

   osm vnfd-create hackfest_basic_vnf.tar.gz
   osm nsd-create hackfest_basic_ns.tar.gz

3. Create an SSH key

   ssh-keygen

4. Create the Network Service in OSM

   osm ns-create --ns_name hackfest1 --nsd_name hackfest_basic-ns --vim_account openstack-site-hackfest-x
   --ssh_keys .ssh/id_rsa.pub --config '{vld: [ {name: mgmtnet, vim-network-name: osm-ext} ] }'
Hands-on: Launching your first NS

5. Validate NS creation in OSM via CLI
   - `osm ns-list`
   - `osm ns-show hackfest1`

6. Validate NS creation in OSM via GUI
   - Go to Instances -> NS Instances

7. Access to the VM created in Openstack VIM
   - `ssh -i .ssh/id_rsa ubuntu@<MGMT_IP>`

8. Delete NS, NSD and VNFD
   - `osm ns-delete hackfest1`
   - `osm vnfd-delete hackfest_basic_vnf`
   - `osm nsd-delete hackfest_basic_ns`
Bonus Hands-on: Creating VNF & NS Descriptors

1. Create the VNF Descriptor
   
   ```
   osm package-create vnf hackfest-basic
   ```

2. Create the NS Descriptor
   
   ```
   osm package-create ns hackfest-basic
   ```

3. Build the packages
   
   ```
   osm package-build hackfest-basic_vnf
   osm package-build hackfest-basic_ns
   ```

4. Upload NFD and NDS to OSM
   
   ```
   osm vnfd-create hackfest-basic_vnf.tar.gz
   osm nsd-create hackfest-basic_ns.tar.gz
   ```
5. Create the Network Service

```
osm ns-create --ns_name hf-basic --nsd_name hackfest-basic_nsd --vim_account openstack-site-hackfest-x --ssh_keys ~/.ssh/id_rsa.pub --config '{vld: [ {name: mgmt, vim-network-name: osm-ext} ]}'
```

6. Validate NS creation in OSM via GUI
   ○ Go to Instances -> NS Instances

7. Compare the VNFD of this example with the previous Hands-On, find the difference and fix it