

Open Source  
**MANO**

## OSM is Awesome

A Practical Introduction to OSM  
Gianpietro Lavado - Whitestack  
OSM#5 Louisville - April 2018



- **Introduction: Architecture and OSM R3**
- Hands On! - Installation, configuration and instantiation
- Contributing to the Community



Open Source  
**MANO**

# ETSI NFV architecture and components

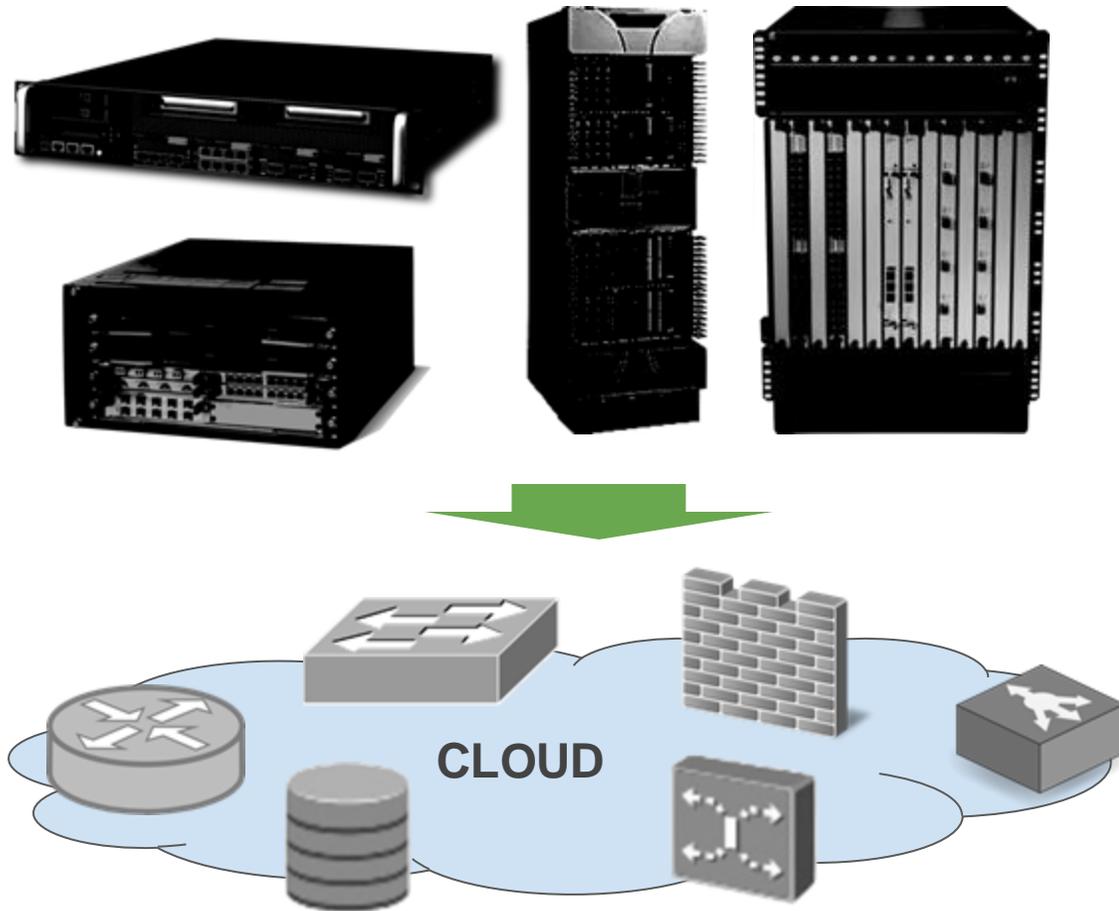
Section: Introduction

ETSI



**Home of NFV**

# What is NFV trying to address?



- Network Function Virtualization (NFV) proposes to virtualize network functions that typically run in dedicated appliances
- The main goal is to support virtualized functions over COTS servers.
- Virtual Network Functions (VNFs) acquire all the advantages of Cloud Applications!

# How was this originated?

- A white paper was written in 2012 by the world's leading telecom network operators.
- This group evolved to the ETSI NFV ISG (Industry Specification Group), formed today by 300+ companies.
- Their main motivation had to do with the increasing TCO of building a network with proprietary hardware appliances.

Network Functions Virtualisation – Introductory White Paper Issue 1

---

## Network Functions Virtualisation

*An Introduction, Benefits, Enablers, Challenges & Call for Action*

**OBJECTIVES**

This is a non-proprietary white paper authored by network operators.

The key objective for this white paper is to outline the benefits, enablers and challenges for Network Functions Virtualisation (as distinct from Cloud/SDN) and the rationale for encouraging an international collaboration to accelerate development and deployment of interoperable solutions based on high volume industry standard servers.

**CONTRIBUTING ORGANISATIONS & AUTHORS**

<b>AT&amp;T:</b>	Margaret Chiosi.
<b>BT:</b>	Don Clarke, Peter Willis, Andy Reid.
<b>CenturyLink:</b>	James Feger, Michael Bugenhagen, Waqar Khan, Michael Fargano.
<b>China Mobile:</b>	Dr. Chunfeng Cui, Dr. Hui Deng.
<b>Colt:</b>	Javier Benitez.
<b>Deutsche Telekom:</b>	Uwe Michel, Herbert Damker.
<b>KDDI:</b>	Kenichi Ogaki, Tetsuro Matsuzaki.
<b>NTT:</b>	Masaki Fukui, Katsuhiro Shimano.
<b>Orange:</b>	Dominique Delisle, Quentin Loudier, Christos Koliass.
<b>Telecom Italia:</b>	Ivano Guardini, Elena Demaria, Roberto Minerva, Antonio Manzalini.
<b>Telefonica:</b>	Diego López, Francisco Javier Ramón Salguero.
<b>Telstra:</b>	Frank Ruhl.
<b>Verizon:</b>	Prodip Sen.

**PUBLICATION DATE**

October 22-24, 2012 at the "SDN and OpenFlow World Congress", Darmstadt-Germany.

# ETSI Publications

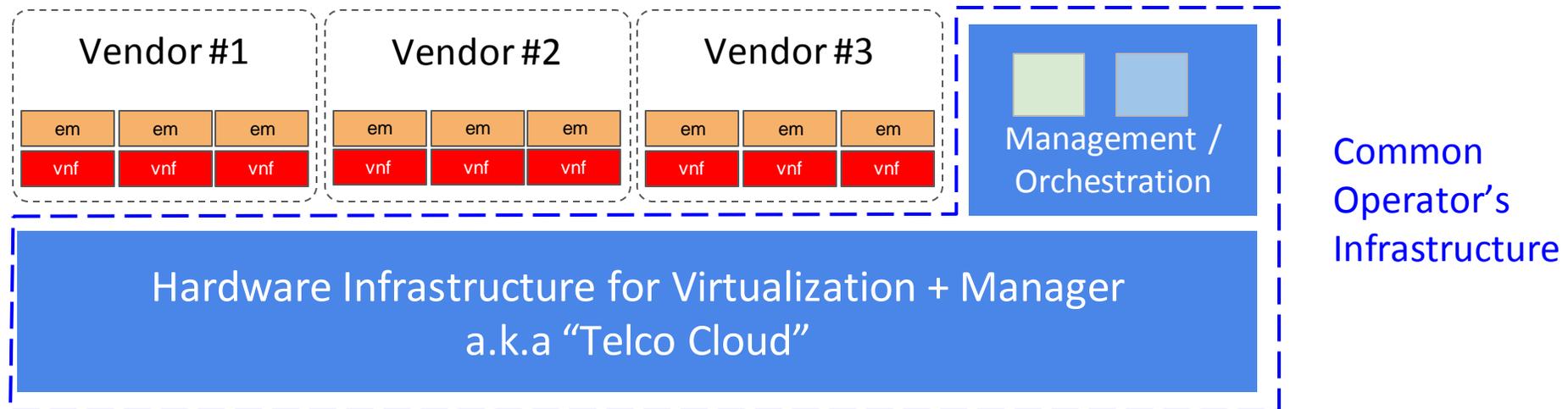
- Based on member's feedback, field experiences and proof of concepts, standard documents have evolved.
- 60+ publications exist today, including the following three main documents:
  - NfV Architectural Framework  
[http://www.etsi.org/deliver/etsi\\_gs/NFV/001\\_099/002/01.02.01\\_60/gs\\_NFV002v010201p.pdf](http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01_60/gs_NFV002v010201p.pdf)
  - NfV Infrastructure Overview  
[http://www.etsi.org/deliver/etsi\\_gs/NFV-INF/001\\_099/001/01.01.01\\_60/gs\\_NFV-INF001v010101p.pdf](http://www.etsi.org/deliver/etsi_gs/NFV-INF/001_099/001/01.01.01_60/gs_NFV-INF001v010101p.pdf)
  - NfV Management and Orchestration  
[http://www.etsi.org/deliver/etsi\\_gs/NFV/001\\_099/002/01.02.01\\_60/gs\\_NFV002v010201p.pdf](http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01_60/gs_NFV002v010201p.pdf)

<http://www.etsi.org/standards-search>



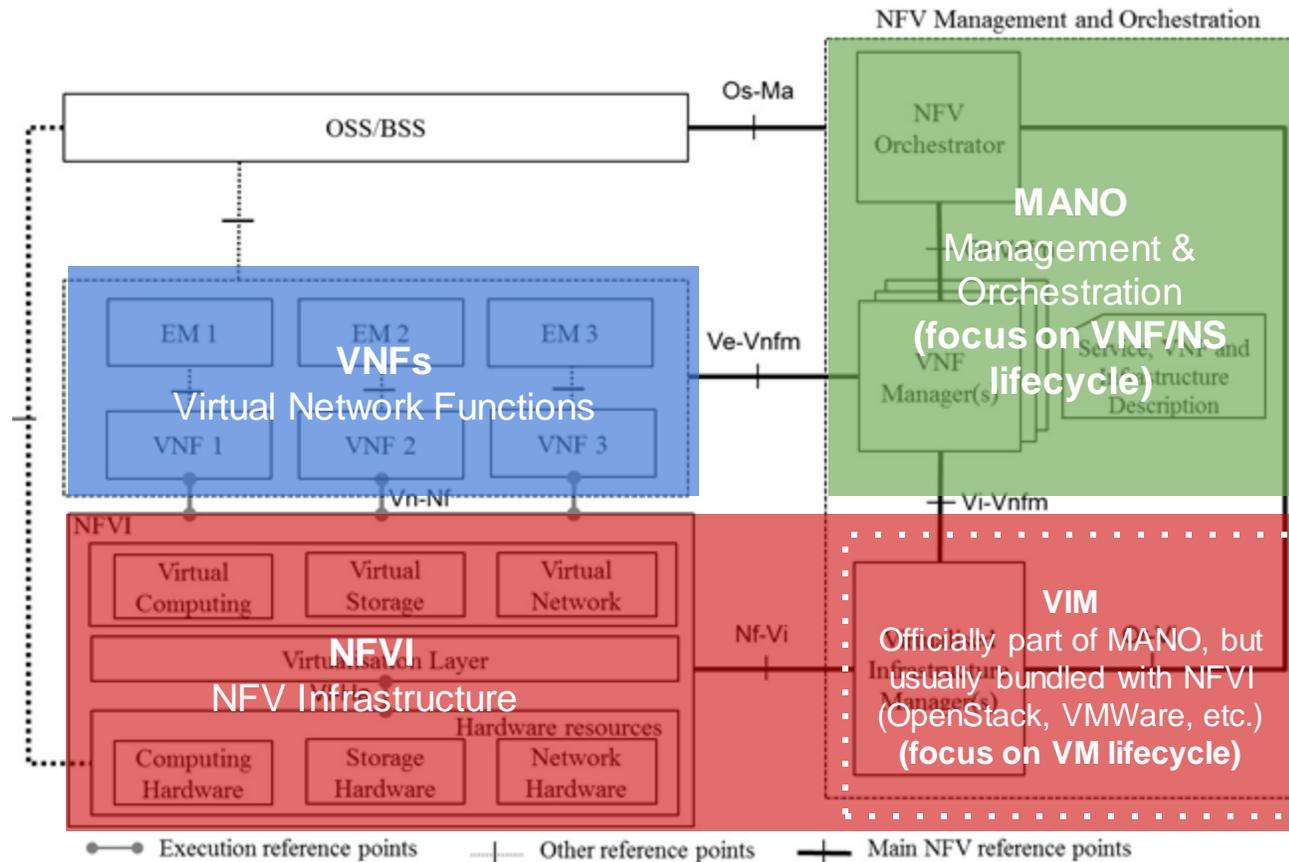
# Benefits of a standard NFV architecture

- We are looking for a **unified and generic virtualization infrastructure**, compatible with any vendor's Virtual Networking Function (VNF), **so standardization is a must.**



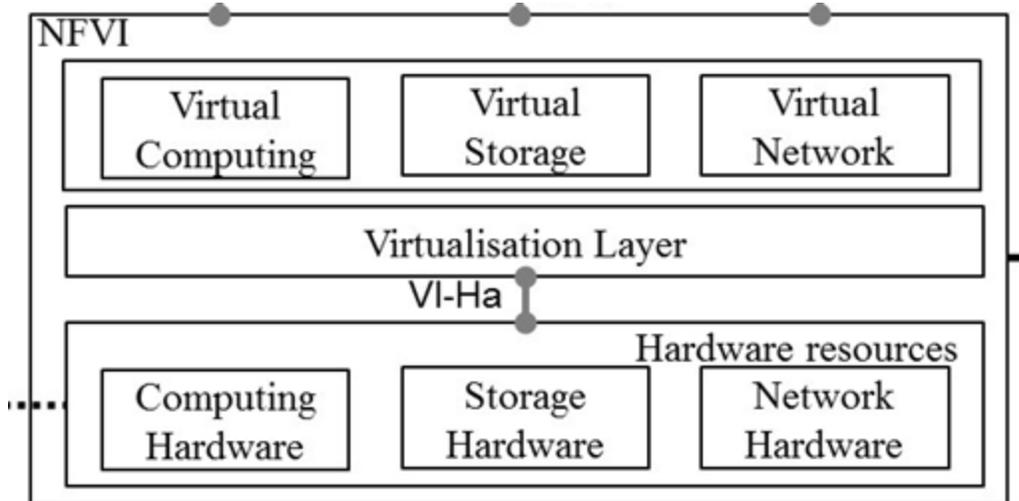
# The ETSI NFV Architecture

- The standard architecture can be better understood in three blocks:



# NFVI: NFV Infrastructure

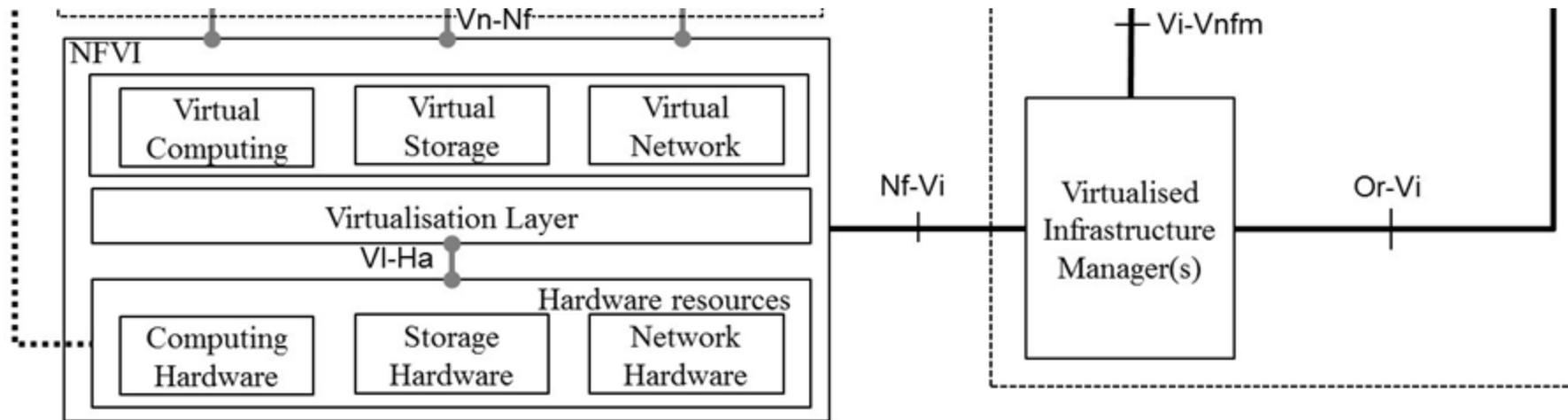
- NFVI goal is to provide a virtualization environment for VNFs, including virtual compute, storage and networking resources.



- But! networking applications may have more strict performance requirements, we will discuss that later.

# MANO: Virtualized Infrastructure Manager

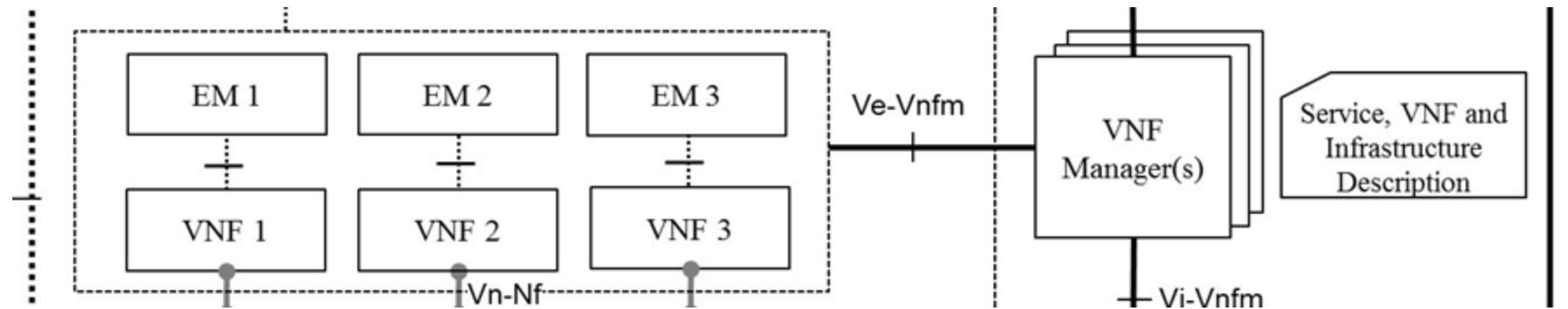
- The Virtualized Infrastructure Manager is part of the 'MANO Stack' and addresses lifecycle management for virtualized resources (VMs, volumes, networking paths and connectivity, etc.)



*Examples: OpenStack distributions, VMWare products, Public Cloud managers, etc.*

# MANO: VNF Manager

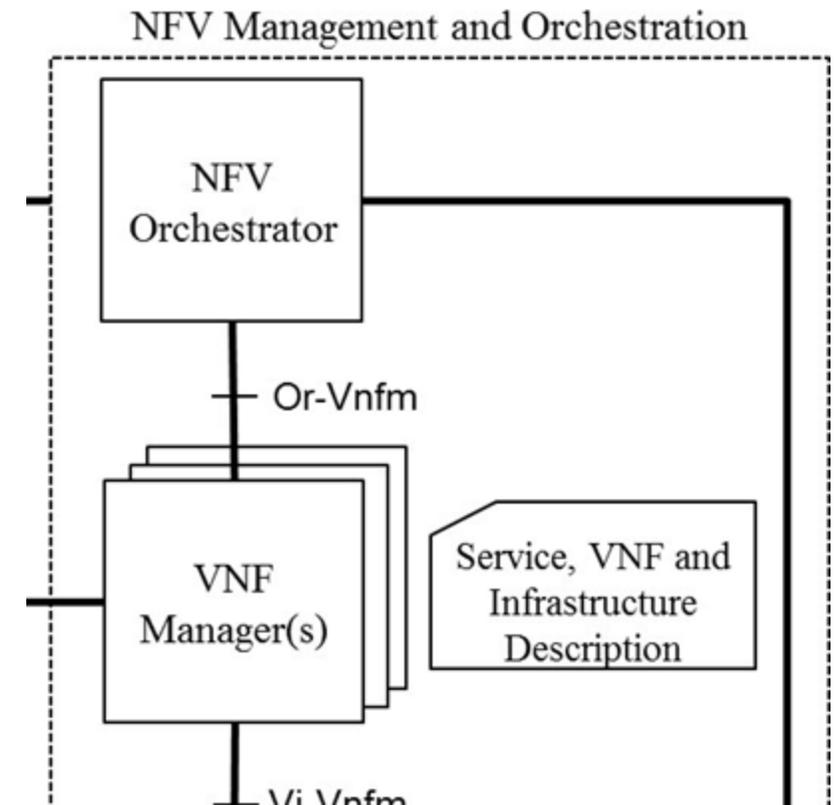
- The VNF Manager, also part of the ‘MANO Stack’, covers lifecycle management for Virtual Network Functions (VNFs), either directly or through their own Element Management System (EMS).



- VNF Managers can be generic (current trend), or vendor-specific ones.

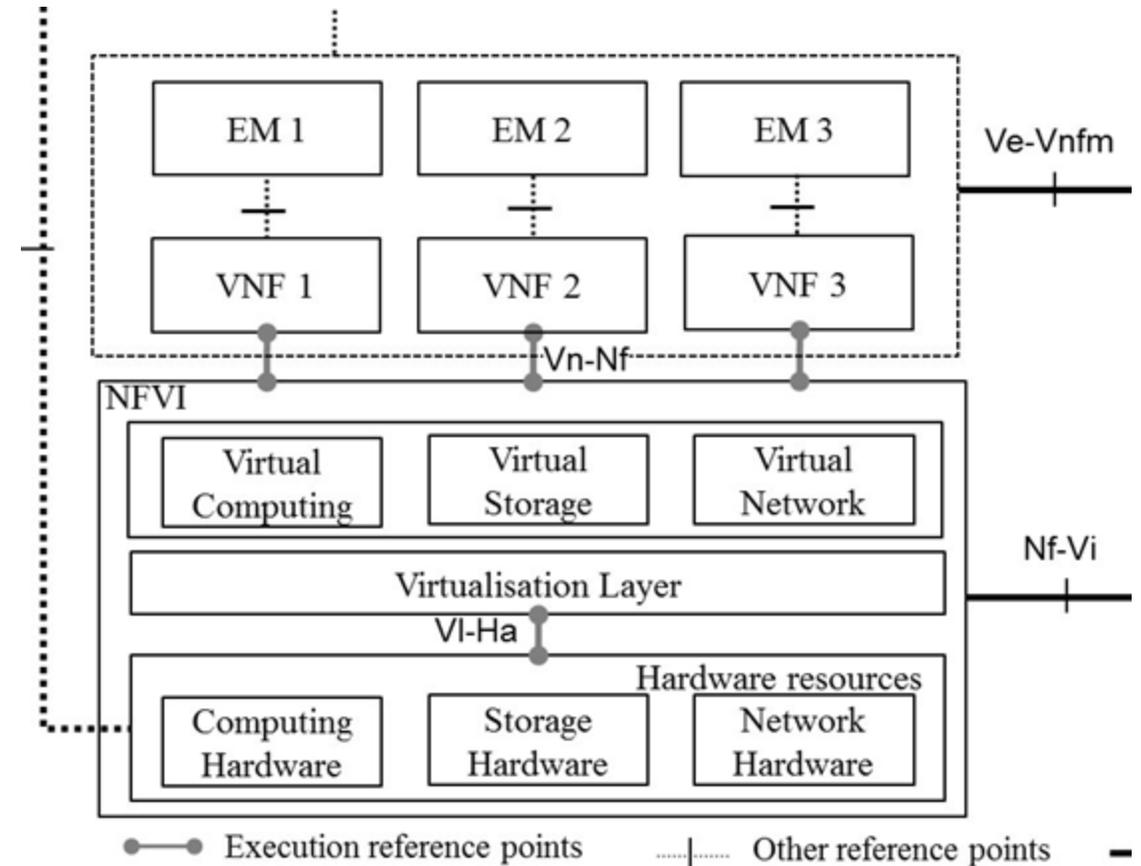
# MANO: NFV Orchestrator

- The NFV Orchestrator, the higher entity in the 'MANO Stack', covers general resource orchestration and services lifecycle, which comprise multiple VNFs and define their roles (traffic paths, scaling decisions, and other service-related requirements)
- It can interact with a generic VNF Manager, or vendor-specific ones.



# Virtual Network Functions

- Finally, the VNFs, which are supported by the underlying NFVI, and managed by their own EM (internal manager) and the VNF Manager (external, 'context-aware' manager)
- They should be able to provide any networking function and interact with other VNFs.



# VNF Descriptor files

One of the most important aspects of achieving a unified VNF catalogue, is having a standard way of describing VNFs.

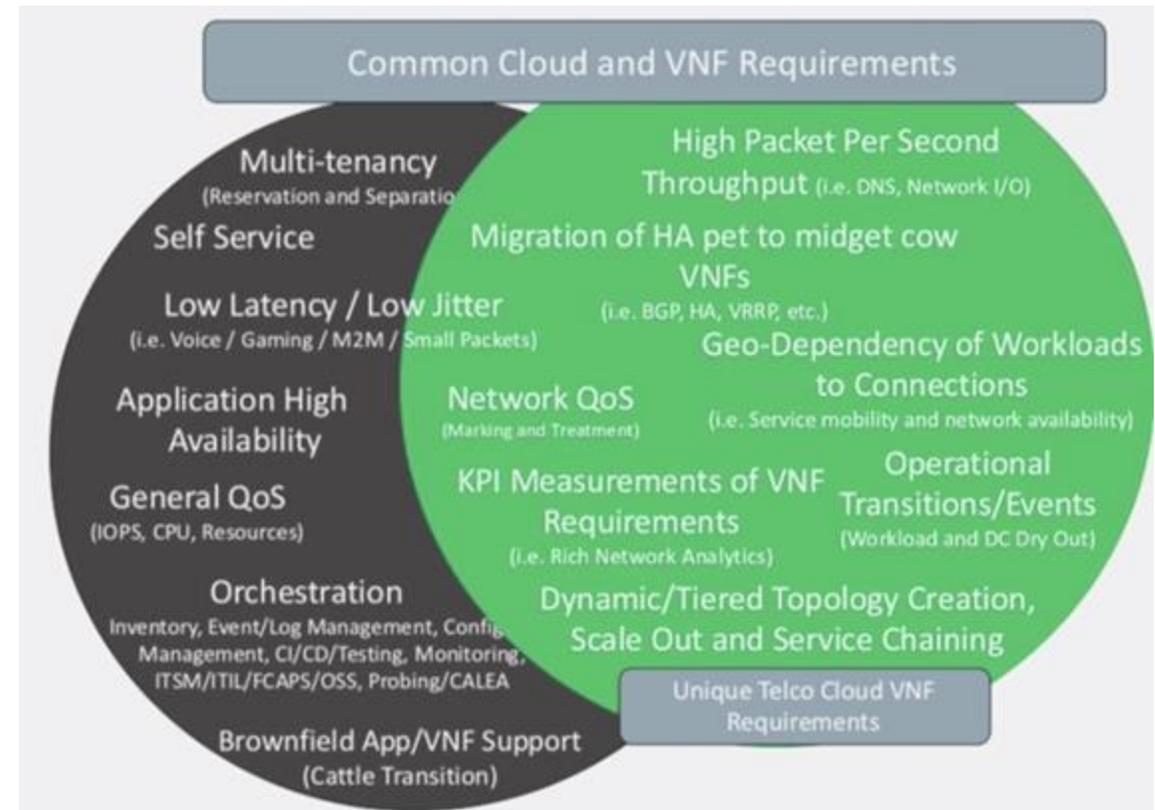
- MANO solutions should give the possibility to describe VNFs through 'descriptor files'
- The industry's goal is a unified and standard descriptor file format across different platforms.
- Both NS (comprised of VNFs) and VNFs should be described in a simple way.

```
vnfd:vnfd-catalog:
  vnfd:vnfd:
  - vnfd:connection-point:
    - vnfd:name: eth0
      vnfd:type: VPORT
    vnfd:description: Generated by OSM pacakage generator
    vnfd:id: ubuntuvnf_vnfd
    vnfd:mgmt-interface:
      vnfd:cp: eth0
    vnfd:name: ubuntuvnf_vnfd
    vnfd:service-function-chain: UNAWARE
    vnfd:short-name: ubuntuvnf_vnfd
    vnfd:vdu:
    - vnfd:cloud-init-file: cloud_init
      vnfd:count: '1'
      vnfd:description: ubuntuvnf_vnfd-VM
      vnfd:guest-epa:
        vnfd:cpu-pinning-policy: ANY
      vnfd:id: ubuntuvnf_vnfd-VM
      vnfd:image: ubuntu_admin
      vnfd:interface:
      - rw-vnfd:floating-ip-needed: 'false'
        vnfd:external-connection-point-ref: eth0
```

# VNF Special Requirements

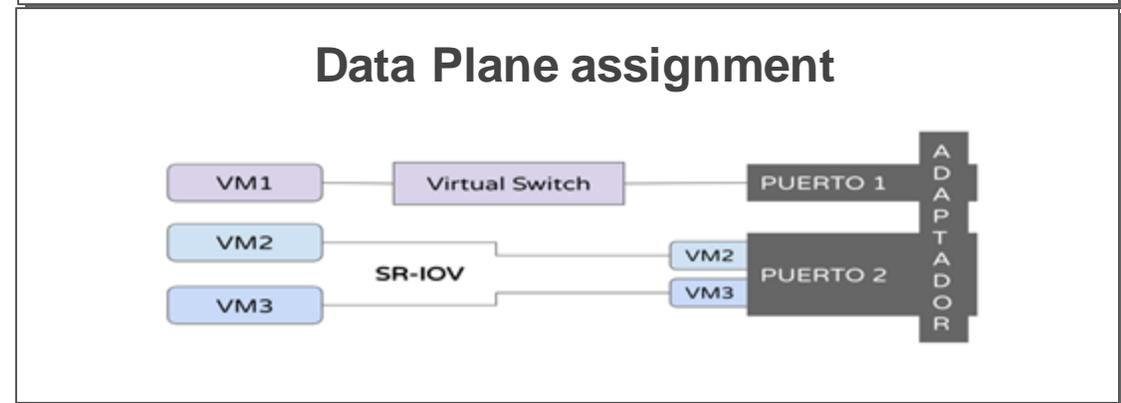
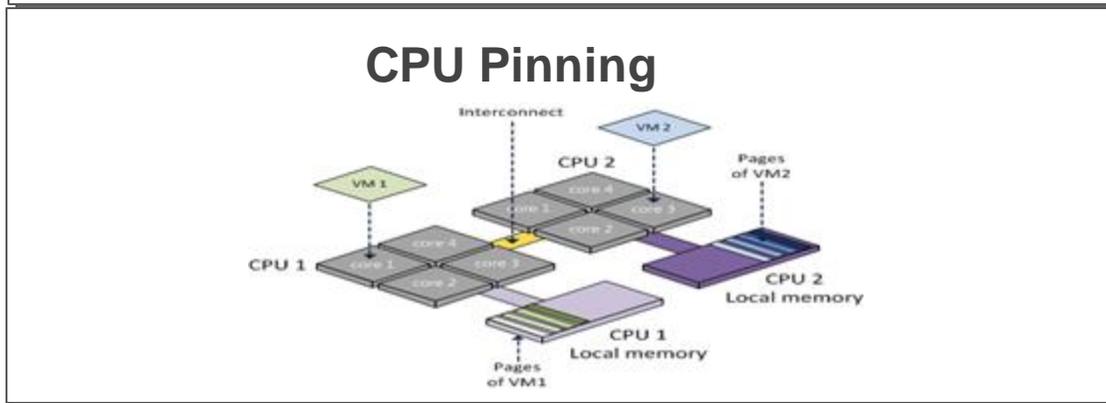
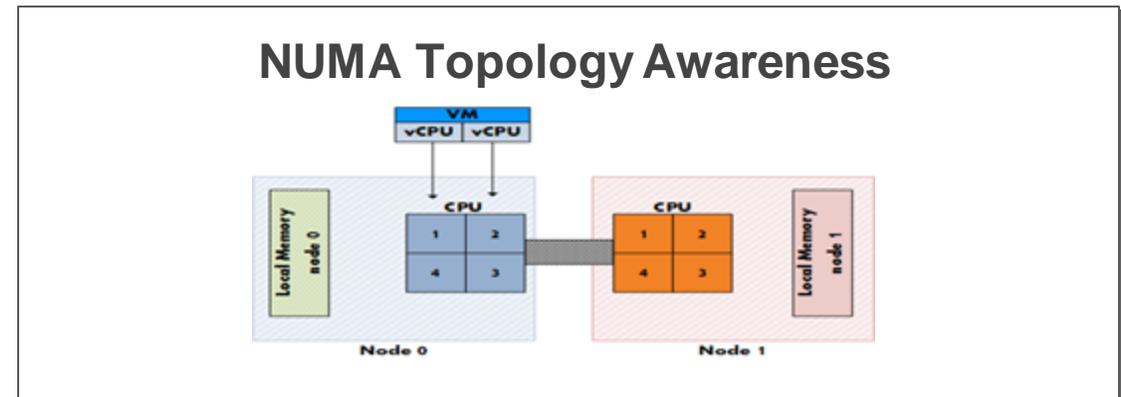
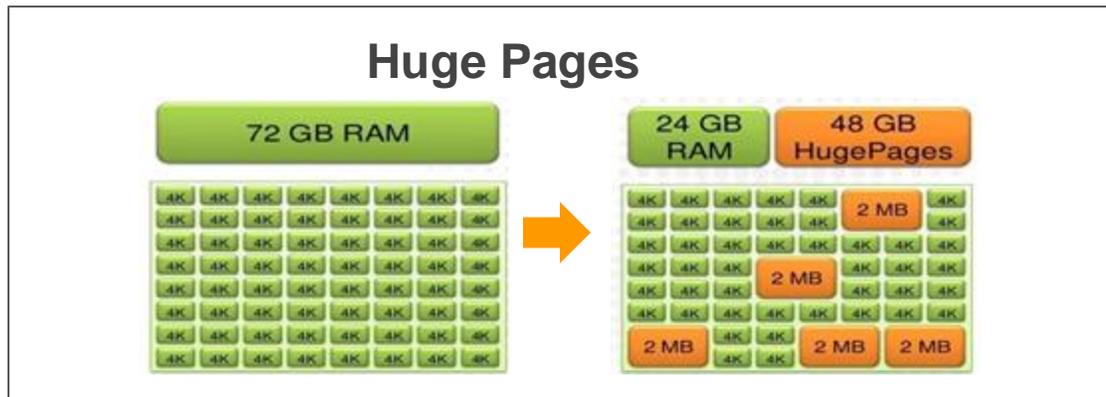
VNFs, especially data-plane ones, usually have additional requirements than common cloud applications, including:

- Minor latency (disk I/O & network)  
→ faster disks, QoS, higher BW
- Geographical distribution  
→ multi-site cloud
- Horizontal auto-scaling  
→ automated operations
- Higher throughput or PPS  
→ EPA: Enhanced Platform Awareness



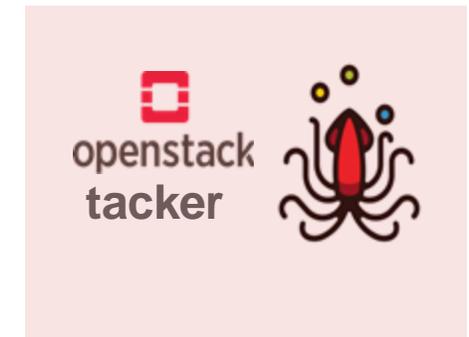
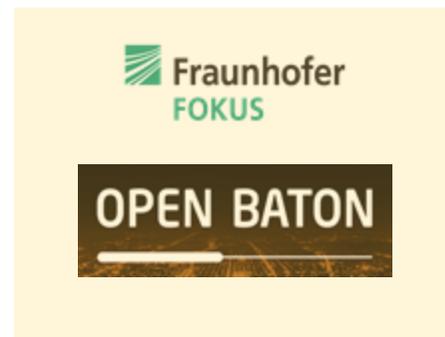
# VNF Special Requirements: EPA!

EPA covers the different approaches that can be taken to increase performance while maintaining a generic (COTS) infrastructure



# The NFV MANO Landscape

- Given that the VIM is already well covered by OpenStack distributions and proprietary solutions, in practice, **the “NFV MANO” part focuses on the VNF Manager and NFV Orchestrator.**
- Among the most popular open source platforms for NFV MANO, we have:





Open Source  
**MANO**

# Introduction to OSM Release Three

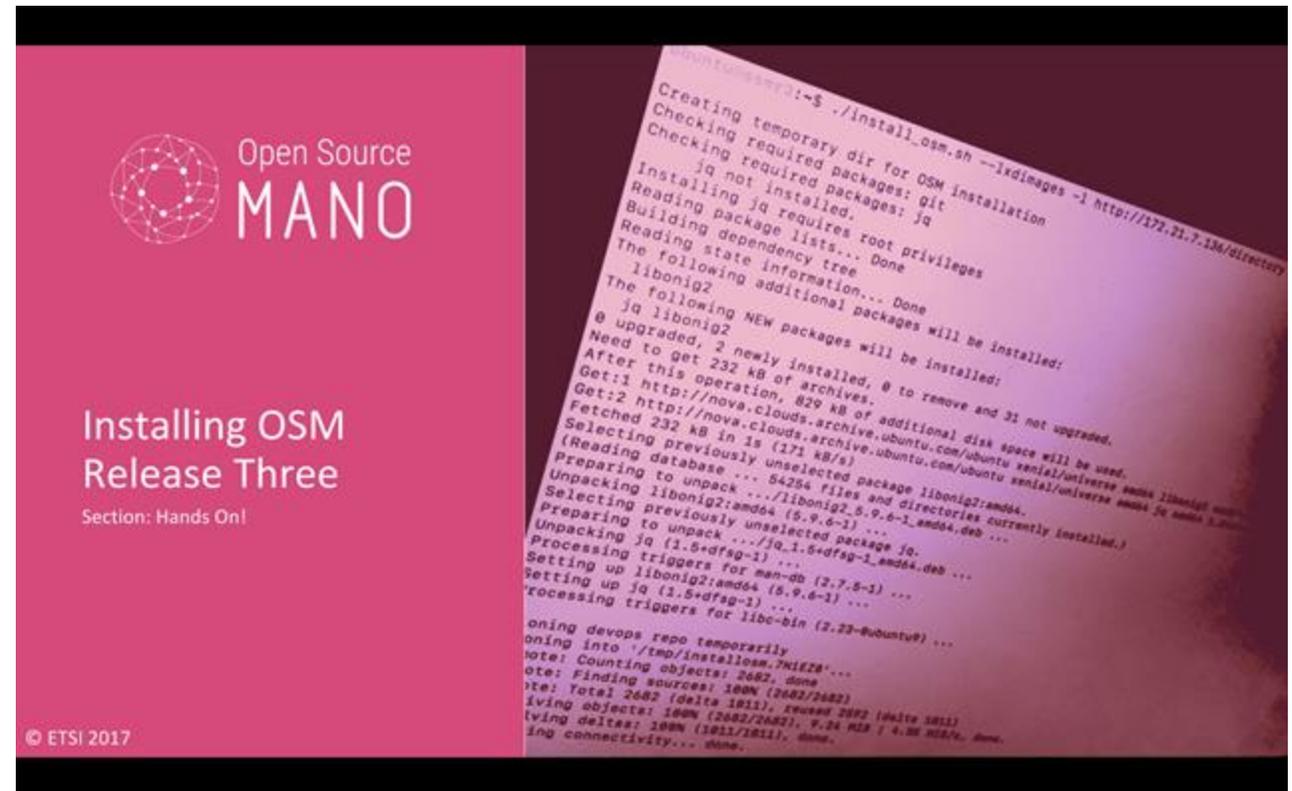
Section: Introduction



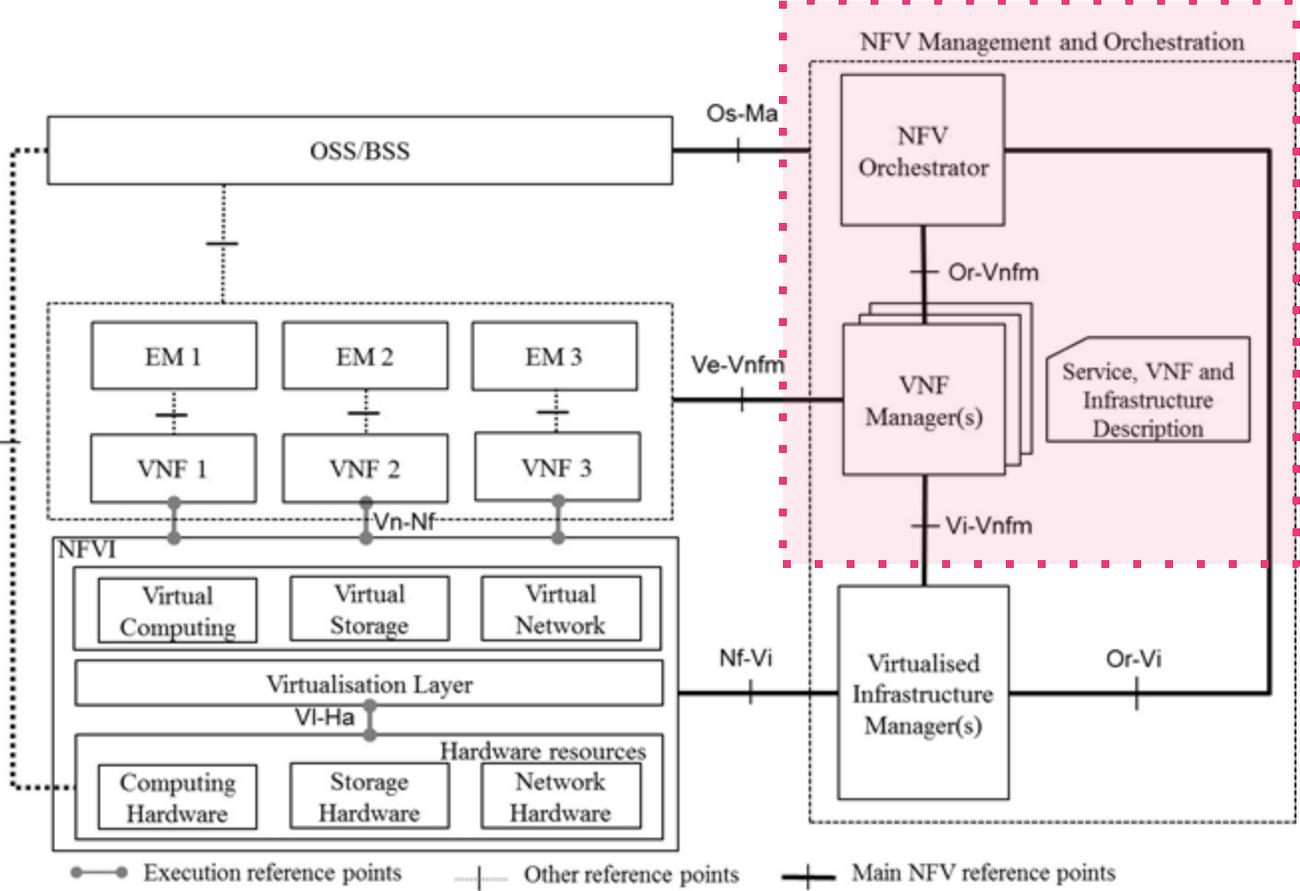
# Let's get started with OSM

Before exploring the highlights of OSM, let's begin the installation to make the best use of time.

- Go to [slide 32](#) and begin OSM installation.



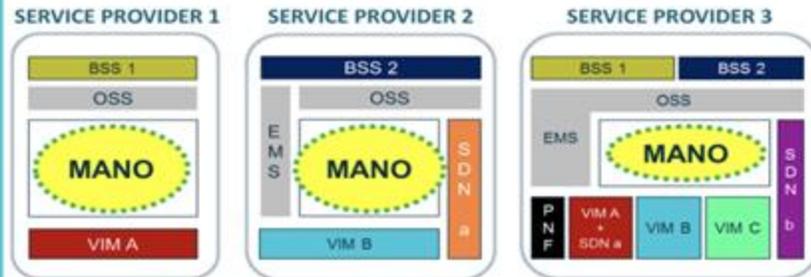
# The Open Source MANO Project



**We are here!**  
 Open Source MANO is an ETSI-hosted project to develop an Open Source NFV Management and Orchestration (MANO) software stack aligned with ETSI NFV.

# OSM Strategy

## FOCUS ON WHAT WE HAVE IN COMMON



Key is **INTEROPERABILITY**, not full architecture

## LEVERAGE ON ETSI NFV WORK



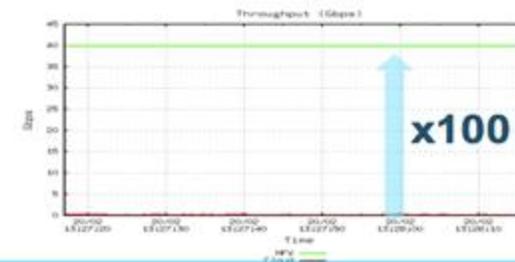
## READY FOR GREENFIELD AND BROWNFIELD



## MULTIPLE VIMs & SDNs ARE HERE TO STAY (public clouds too!)

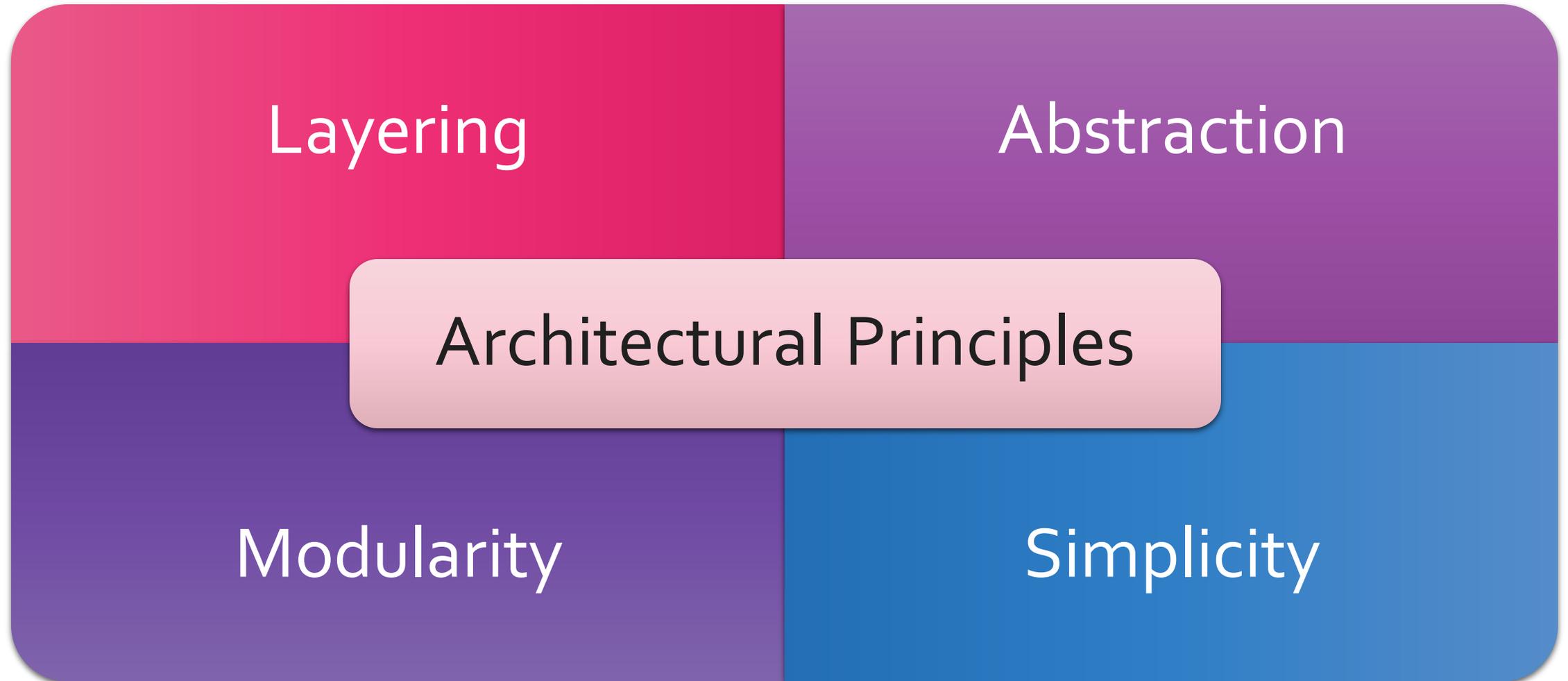


## PERFORMANCE MATTERS FOR THE BUSINESS CASE



## OPEN SOURCE AS TOOL TO FACILITATE CONVERGENCE

# OSM Architectural Principles



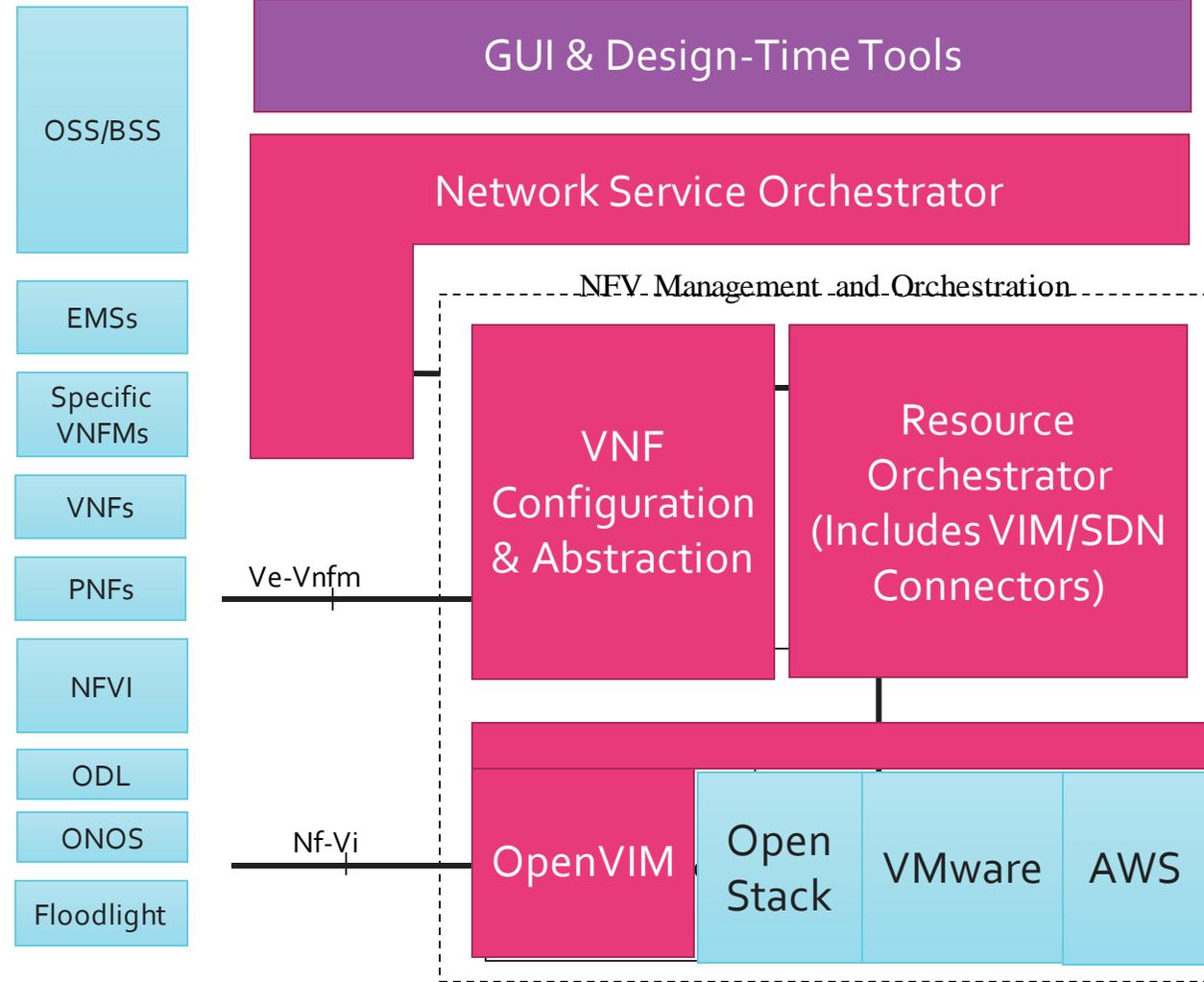
# How OSM maps to ETSI NFV MANO?

## Run-Time Scope

- Automated end-to-end Service Orchestration
- Superset of ETSI NFV MANO
- Plugin model for integrating multiple SDN controllers
- Plugin model for integrating multiple VIMs
- Plugin model for integrating monitoring tools
- Integrated Generic VNFM with support for integrating Specific VNFMs
- Support for Physical Network Function integration
- Greenfield and brownfield deployments

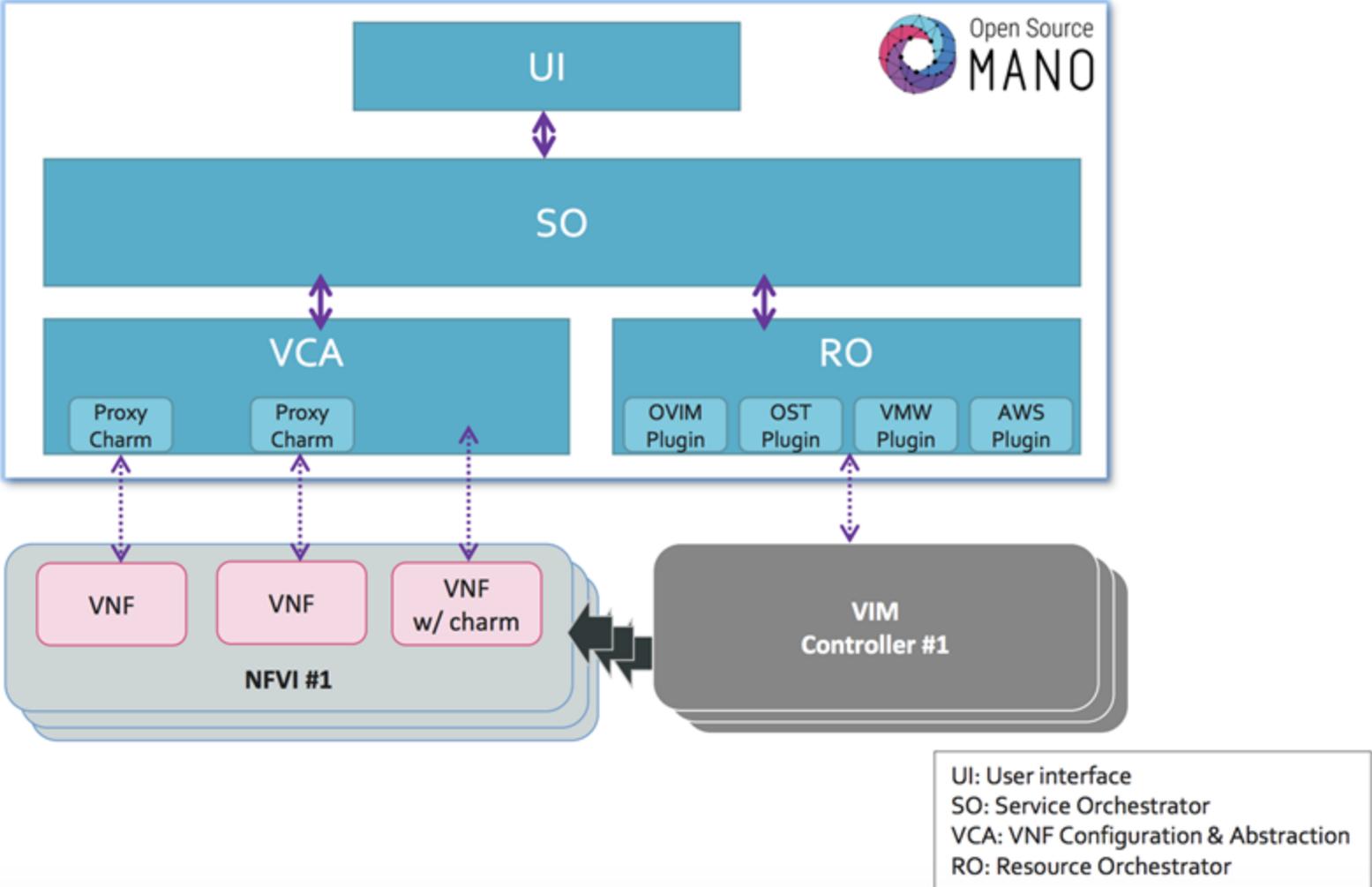
## Design-Time Scope

- Network Service Definition (CRUD operations)
- Model-Driven Environment with Data Models aligned with ETSI NFV
- VNF Package Generation
- GUI

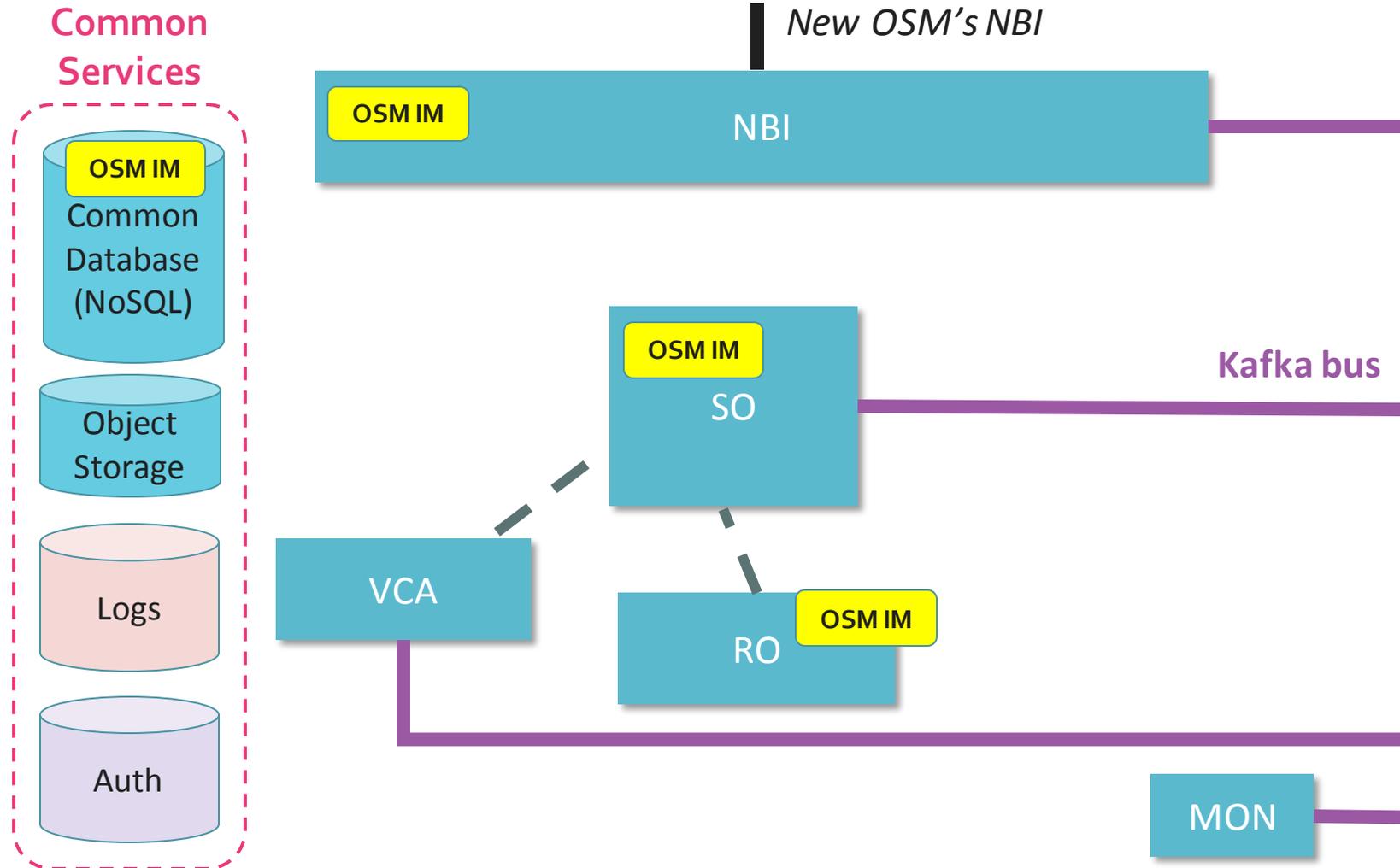


— Main NFV reference points

# This is today's OSM (simplified) Architecture



# ...which is evolving!



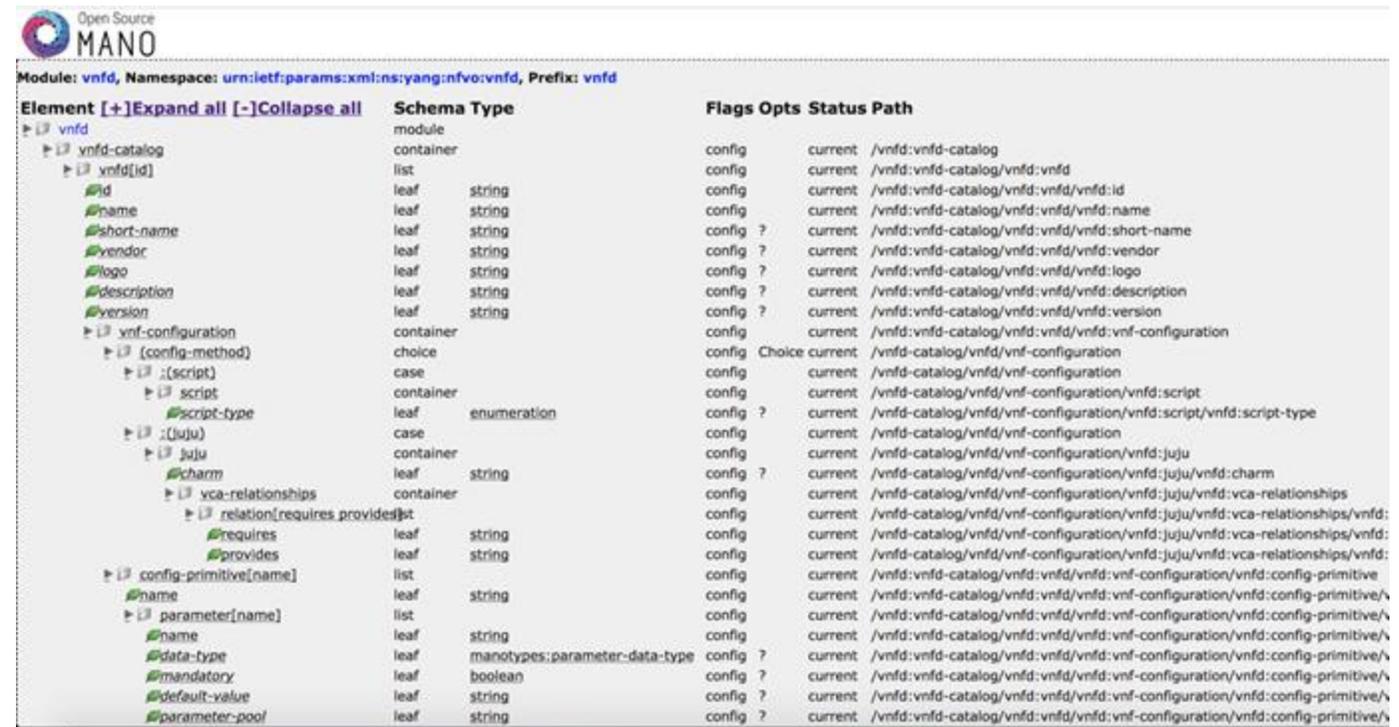
# Why is OSM Awesome?

## It has a rich and open information model

- Agnostic to VIM, SDN platform, VNF and OSS connectors/specifics.
- It allows for a uniform NFV orchestration, abstracted from the environment
- Aligned with ETSI-NFV Information Model

Visit:

[https://osm.etsi.org/wikipub/index.php/OSM\\_Information\\_Model](https://osm.etsi.org/wikipub/index.php/OSM_Information_Model)



Module: vnf, Namespace: urn:ietf:params:xml:ns:yang:nfv:vnf, Prefix: vnf

Element	Schema Type	Flags	Opts	Status	Path
vnf	module				
vnf-catalog	container				/vnf:vnf-catalog
vnf[id]	list				/vnf:vnf-catalog/vnf
id	leaf string				/vnf:vnf-catalog/vnf:vnf[id]:id
name	leaf string				/vnf:vnf-catalog/vnf:vnf[id]:name
short-name	leaf string				/vnf:vnf-catalog/vnf:vnf[id]:short-name
vendor	leaf string	?			/vnf:vnf-catalog/vnf:vnf[id]:vendor
logo	leaf string	?			/vnf:vnf-catalog/vnf:vnf[id]:logo
description	leaf string	?			/vnf:vnf-catalog/vnf:vnf[id]:description
version	leaf string	?			/vnf:vnf-catalog/vnf:vnf[id]:version
vnf-configuration	container				/vnf:vnf-catalog/vnf:vnf-configuration
(config-method)	choice		Choice		/vnf:vnf-catalog/vnf:vnf-configuration
:script	case				/vnf:vnf-catalog/vnf:vnf-configuration
script	container				/vnf:vnf-catalog/vnf:vnf-configuration/vnf:script
script-type	leaf enumeration	?			/vnf:vnf-catalog/vnf:vnf-configuration/vnf:script/vnf:script-type
:juju	case				/vnf:vnf-catalog/vnf:vnf-configuration
juju	container				/vnf:vnf-catalog/vnf:vnf-configuration/vnf:juju
charm	leaf string	?			/vnf:vnf-catalog/vnf:vnf-configuration/vnf:juju/vnf:charm
vca-relationships	container				/vnf:vnf-catalog/vnf:vnf-configuration/vnf:juju/vnf:vca-relationships
requires	leaf string				/vnf:vnf-catalog/vnf:vnf-configuration/vnf:juju/vnf:vca-relationships/vnf:requires
provides	leaf string				/vnf:vnf-catalog/vnf:vnf-configuration/vnf:juju/vnf:vca-relationships/vnf:provides
config-primitive[name]	list				/vnf:vnf-catalog/vnf:vnf-configuration/vnf:config-primitive
name	leaf string				/vnf:vnf-catalog/vnf:vnf-configuration/vnf:config-primitive[name]
parameter[name]	list				/vnf:vnf-catalog/vnf:vnf-configuration/vnf:config-primitive[name]/parameter
name	leaf string				/vnf:vnf-catalog/vnf:vnf-configuration/vnf:config-primitive[name]/parameter[name]
data-type	leaf manotypes:parameter-data-type	?			/vnf:vnf-catalog/vnf:vnf-configuration/vnf:config-primitive[name]/parameter[name]/data-type
mandatory	leaf boolean	?			/vnf:vnf-catalog/vnf:vnf-configuration/vnf:config-primitive[name]/parameter[name]/mandatory
default-value	leaf string	?			/vnf:vnf-catalog/vnf:vnf-configuration/vnf:config-primitive[name]/parameter[name]/default-value
parameter-pool	leaf string	?			/vnf:vnf-catalog/vnf:vnf-configuration/vnf:config-primitive[name]/parameter[name]/parameter-pool

# Why is OSM Awesome?

It has a large and diverse community!  
More than 90 members and growing



Member logos displayed on a world map background include:

- Sprint
- Bell
- BT
- telenor
- amazon web services
- verizon
- PT
- Telefonica
- CableLabs
- SK telecom
- kt

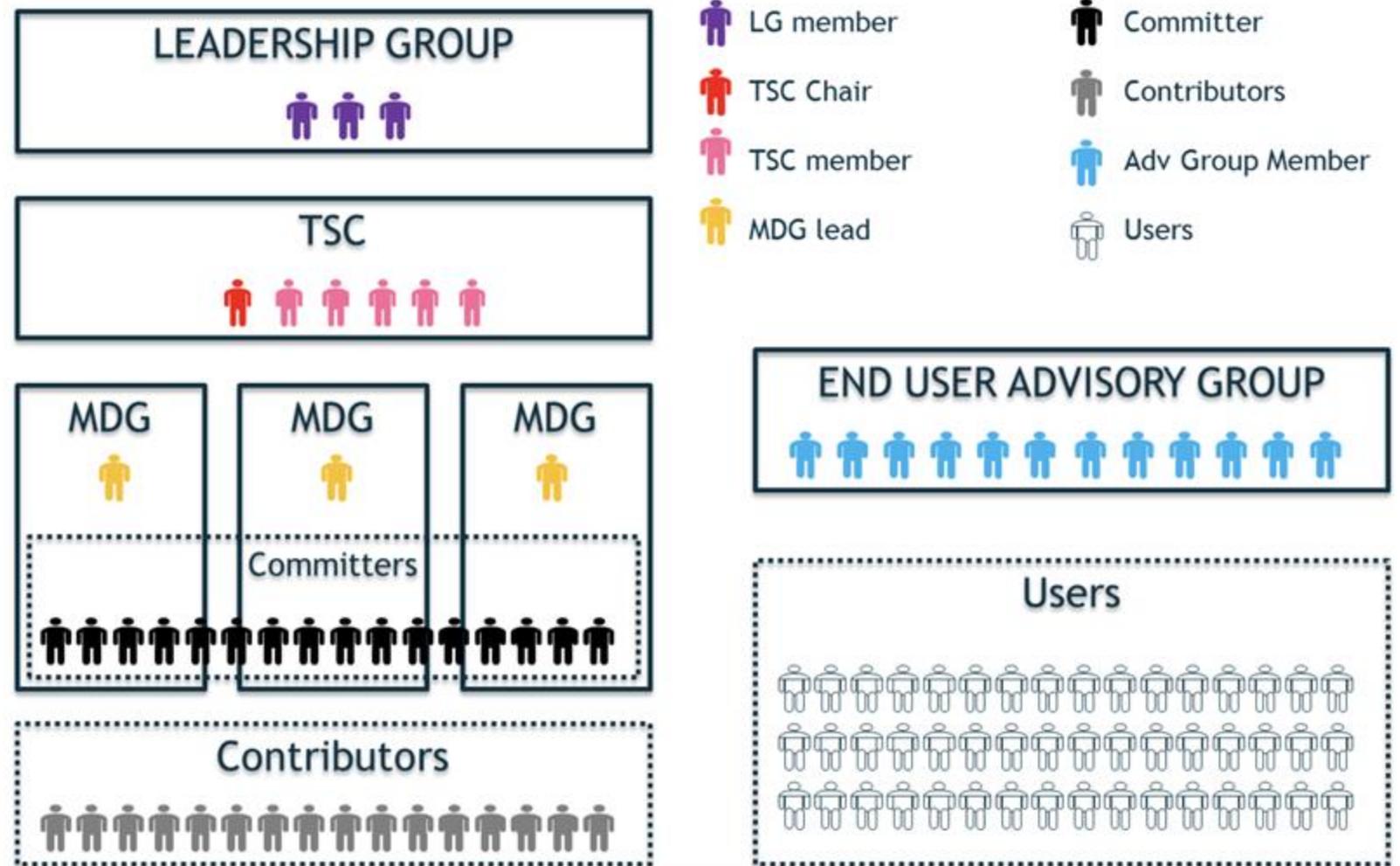
Additional member logos shown in a grid below include:

- Swip, ADVA, ALTRAN, AMPLIPHAE, AOSACOM, ARCTOS LABS, ASTELLIA, AtoS, Armon, big switch, BROCADE, CANONICAL, CISE, cenX
- CITRIX, CNIC, COMARCH, comptel, C-PLANE NETWORKS, TTC, DATAKIT, DELLEMC, Dialogic, EMPIRIX, EURECOM, F5, FONDATION BRUNO KESSLER, Fraunhofer
- i2cat, iconectiv, idea, indra, Infoblox, intel, Iricent, Ixia, keynetic, KING'S COLLEGE LONDON, LAYER123, MANTICA, MAVENIR, MC5G, MeadowCom
- metaswitch, M, mycomOSI, solarwinds n-able, Netcracker, NetNumber, Netrounds, NETSCOUT, NEXXTWORKS, NFWare, PacketFront Software, PADERBORN UNIVERSITY
- PENSA, PROCERA, RADCOM, radware, rediot, RIFT.io, sandvine, SEVEN PRINCIPLES, SIGMA, SIGSCALE, SPIRENT, Tech Mahindra, technicolor, T-Mobile
- TNO, ubiwhere, Universidad Carlos III de Madrid, Universitat de Pisa, VIAVI, virtuosys, vmware, whitestack, WIND, wipro, X-FLOW, V, ZTE

- 11 Global Service Providers
- Leading IT/Cloud players
- VNF providers

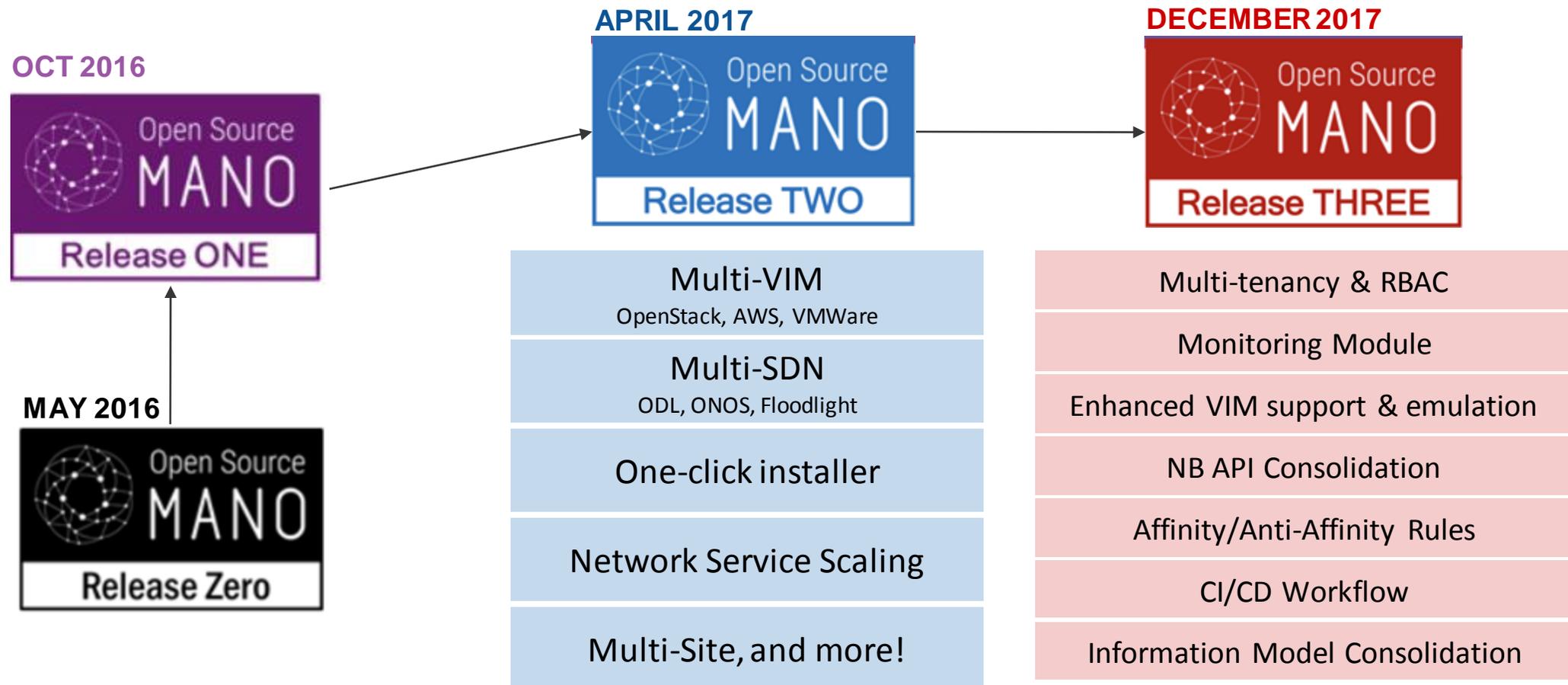
# Why is OSM Awesome?

It is well organized for producing production-ready upstream code



# Why is OSM Awesome?

## It prioritizes features for production readiness



# Why is OSM Awesome?

## ...and will keep expanding its features towards production deployments:

- Alignment of OSM NBI to SOL05 & SOL04
- Tighter integration of Monitoring module
  - Leverage on metrics and alarms to drive further automation (new Policy Manager module)
- Management of VNFs of new generation
  - Docker containers + Kubernetes mgmt
  - Hybrid NFs (Virtual + Physical)
- Support of future 5G deployments
  - Network Slicing likely to require NS Nesting, Management of shared resources
- Improvements in packaging format
- Portable and lightweight deployments

# Agenda

- Introduction: Architecture and OSM R3
- **Hands On! - Installation, configuration and instantiation**
- Contributing to the Community



Open Source  
**MANO**

# Installing OSM Release Three

Section: Hands On!

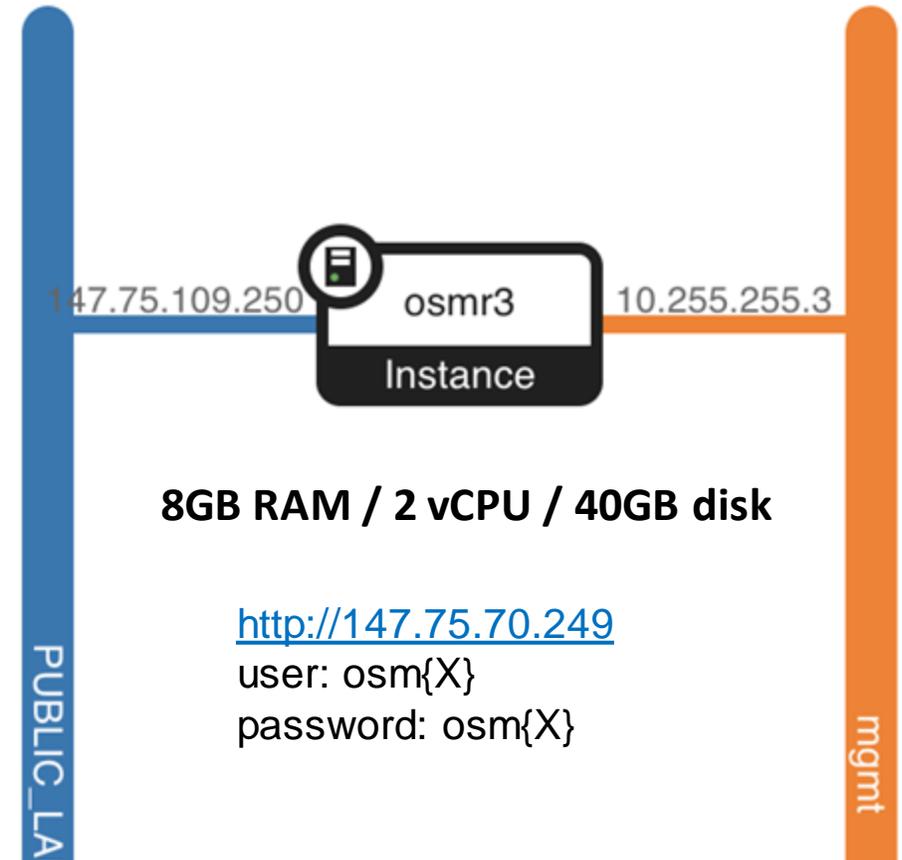
```
ubuntu@osmr3:~$ ./install_osm.sh --lxdimages -l http://172.21.7.136/directory
Creating temporary dir for OSM installation
Checking required packages: git
Checking required packages: jq
jq not installed.
Installing jq requires root privileges
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libonig2
The following NEW packages will be installed:
  jq libonig2
0 upgraded, 2 newly installed, 0 to remove and 31 not upgraded.
Need to get 232 kB of archives.
After this operation, 829 kB of additional disk space will be used.
Get:1 http://nova.clouds.archive.ubuntu.com/ubuntu xenial/universe amd64 libonig2 amd64 5.9.6-1 [171 kB]
Get:2 http://nova.clouds.archive.ubuntu.com/ubuntu xenial/universe amd64 jq amd64 1.5+dfsg-1 [111 kB]
Fetched 232 kB in 1s (171 kB/s)
Selecting previously unselected package libonig2:amd64.
(Reading database ... 54254 files and directories currently installed.)
Preparing to unpack .../libonig2_5.9.6-1_amd64.deb ...
Unpacking libonig2:amd64 (5.9.6-1) ...
Selecting previously unselected package jq.
Preparing to unpack .../jq_1.5+dfsg-1_amd64.deb ...
Unpacking jq (1.5+dfsg-1) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up libonig2:amd64 (5.9.6-1) ...
Setting up jq (1.5+dfsg-1) ...
Processing triggers for libc-bin (2.23-0ubuntu9) ...

Cloning devops repo temporarily
Cloning into '/tmp/installosm.7N1Ez0'...
Note: Counting objects: 2682, done.
Note: Finding sources: 100% (2682/2682)
Note: Total 2682 (delta 1011), reused 2592 (delta 1011)
Note: Receiving objects: 100% (2682/2682), 9.24 MiB | 4.35 MiB/s, done.
Note: Resolving deltas: 100% (1011/1011), done.
Setting up connectivity... done.
```

# The lab environment

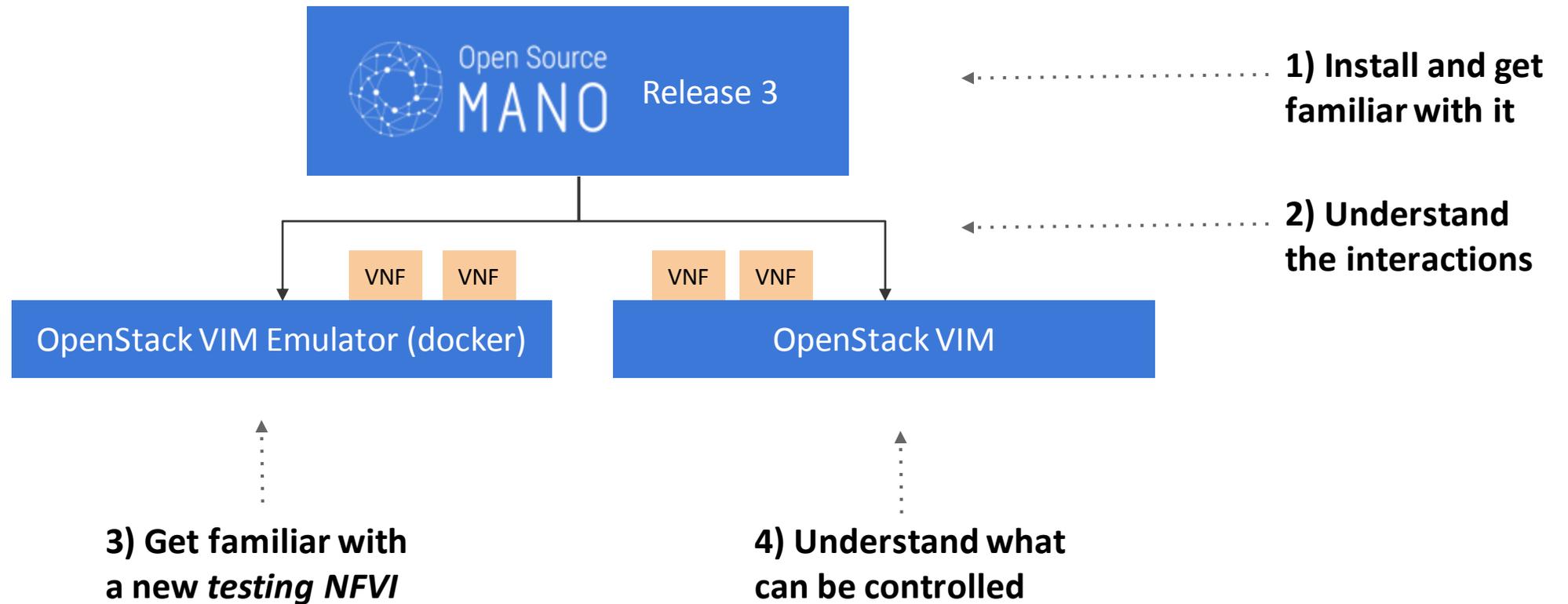
Each POD is an OpenStack tenant containing:

- A VM running Ubuntu, ready to install OSM and the VIM Emulator.
- Connection to 2 networks:
  - “PUBLIC\_LAB”: for external management.
  - “mgmt”: to place VNFs and directly interact with them.



Lab folder: <https://goo.gl/Vuv7xw>

# The lab objectives



# Installing OSM

There are multiple options for installing OSM, including:

- Installing into LXC containers (current official way)
  - From binaries (>60m)
  - From source code (>60m)
  - From prepared LXC images (~7-12m)
  - Adding a the VIM emulator (+ ~10m, ~1m with images)
- Installing into Docker Containers (experimental, ~3m with images)

Official options → [https://osm.etsi.org/wikipub/index.php/OSM\\_Release\\_THREE](https://osm.etsi.org/wikipub/index.php/OSM_Release_THREE)

# Hands-on: Let's install OSM!

In this exercise, we will use prepared LXC containers, with a new option for including an OpenStack VIM Emulator \* (~10min)

1. Access your POD's virtual machine (ubuntu/ubuntu), which already includes LXC/LXD setup, the installer file, and Docker (for the VIM Emulator)
2. Run the following command to initiate automated installation:  

```
./install_osm.sh --lxdimages -l http://147.75.91.107/directory --vimemu
```

  - The procedure will import LXC images from a local repository (if specified without “-l”, it will download them from the official public repository)
  - The --vimemu option is only available with the latest installer file

# Hands-on: Let's install OSM!

At the end of the installation script, some environment variables should be set to your session to run the OSM client and VIM Emulator.

```
# You can copy the following lines or directly put the IPs after the equal signs.
```

```
# OSM client related variables
```

```
export OSM_HOSTNAME=`lxc list | awk '($2=="SO-ub"){print $6}'`
```

```
export OSM_RO_HOSTNAME=`lxc list | awk '($2=="RO"){print $6}'`
```

```
# VIM Emulator related variables
```

```
export VIMEMU_HOSTNAME=172.17.0.2
```

→ They can be set on the fly or persisted at the end of your ~/.bashrc file.

# Hands-on: Let's install OSM!

Finally, you should be able to login with **admin / admin** credentials to the UI, or run some OSM commands from the host

```
# VIM List and Network Service Descriptors list  
# will show empty for now
```

```
osm vim-list
```

```
osm nsd-list
```

```
# VIM Emulator should show a couple of  
# "emulated datacenters"
```

```
docker exec vim-emu vim-emu datacenter list
```

Access the GUI at <https://{YOUR IP}:8443>



## LAUNCHPAD LOGIN



Open Source  
**MANO**

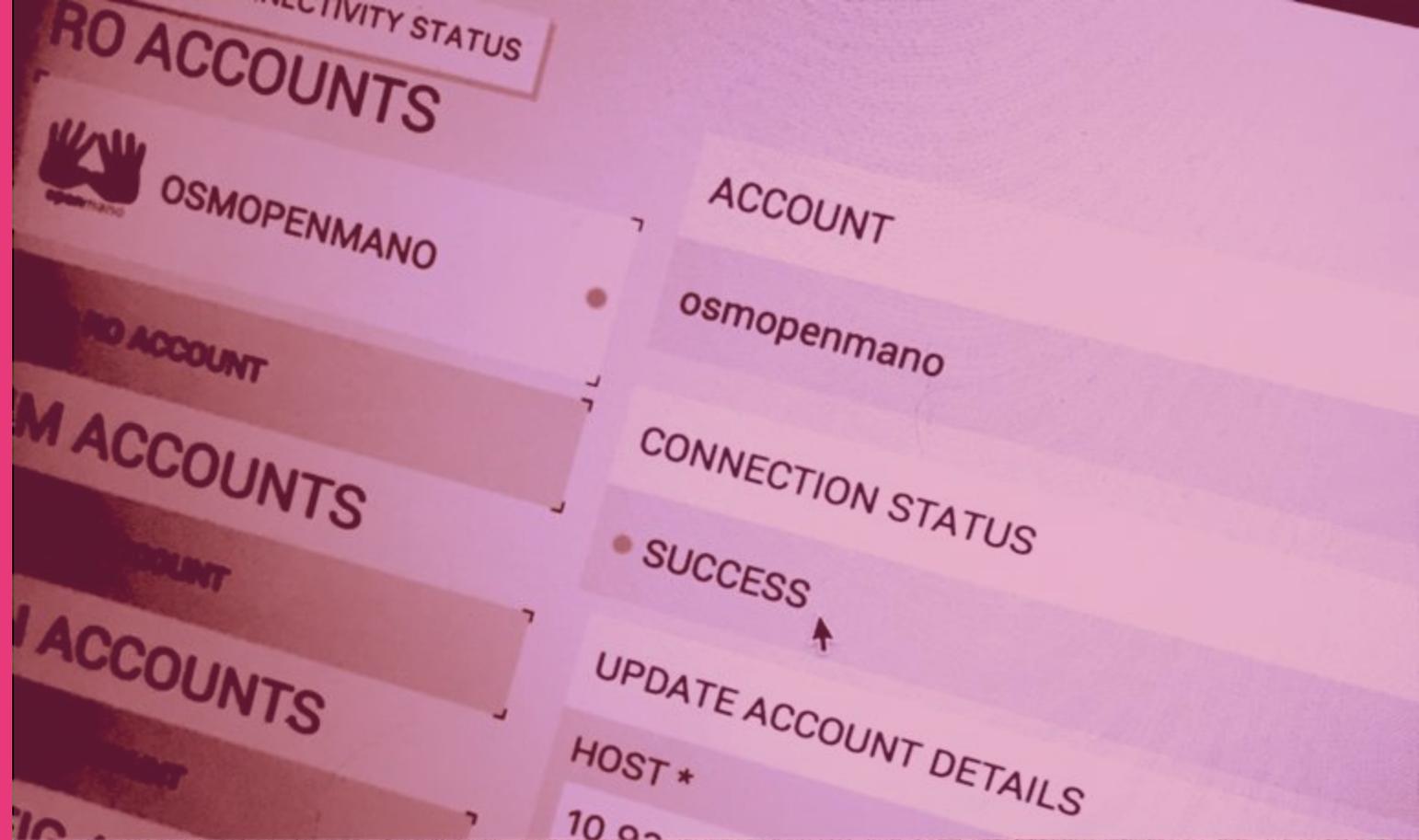
Time for a `<br>`  
let's grab some 



Open Source  
**MANO**

# OSM initial configuration

Section: Hands On!

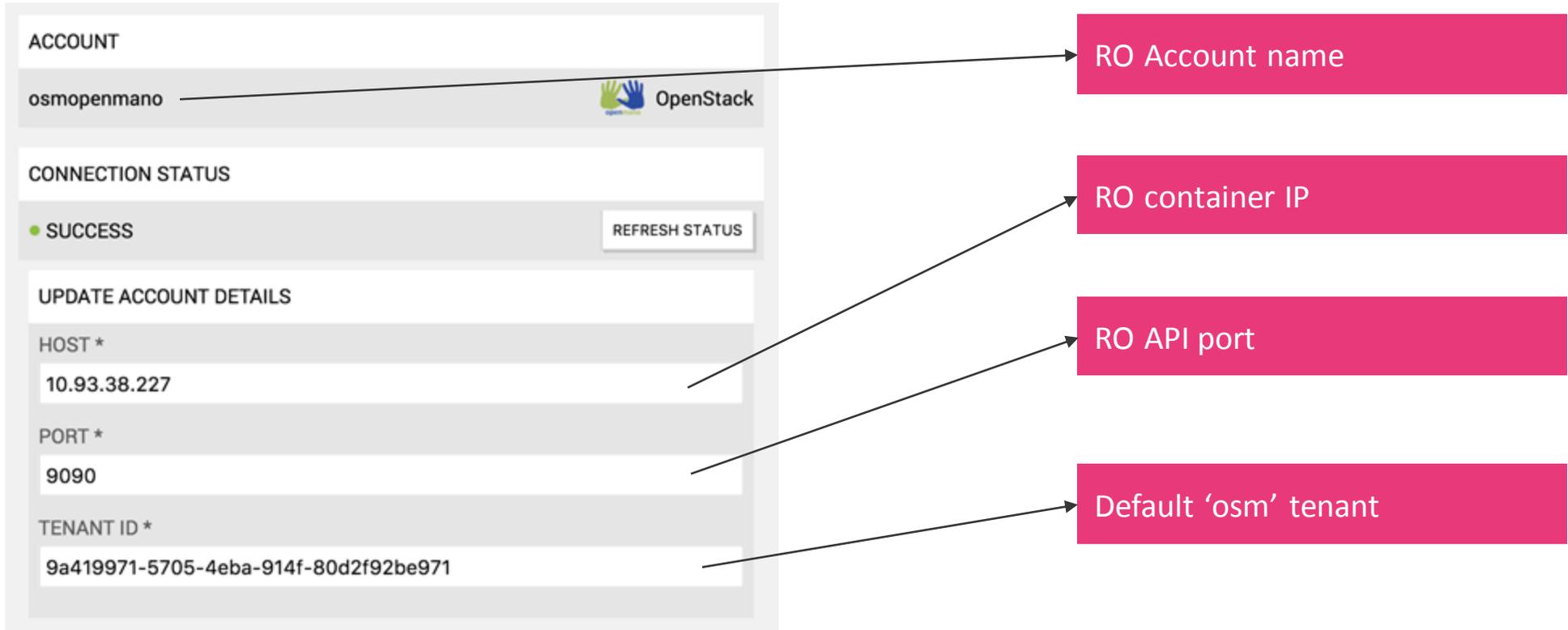


There are a few things to get started:

- Ensure that components have been adequately integrated during the installation process
  - SO - RO
  - SO - VCA Config Agent
- Associate a VIM (or VIMs) to OSM
  - [OpenStack](#) (real, or VIM Emulator)
  - [VMWare VCD](#)
  - [Amazon Web Services](#)

# Hands-on: Let's check the accounts!

In this exercise, we will check that relevant components have been integrated during installation. Go to 'Accounts' tab and check RO:



The screenshot shows the 'ACCOUNT' configuration page in the OpenStack interface. The account name is 'osmopenmano'. The connection status is 'SUCCESS'. The 'UPDATE ACCOUNT DETAILS' section shows the following fields:

- HOST \*: 10.93.38.227
- PORT \*: 9090
- TENANT ID \*: 9a419971-5705-4eba-914f-80d2f92be971

Annotations on the right side of the image point to specific fields:

- RO Account name (points to 'osmopenmano')
- RO container IP (points to '10.93.38.227')
- RO API port (points to '9090')
- Default 'osm' tenant (points to '9a419971-5705-4eba-914f-80d2f92be971')

# Hands-on: Let's check the accounts!

Now, let's check the config-agent account, which is the main Juju controller that resides inside the VCA container:



The screenshot shows the 'ACCOUNT' configuration page in the Open Source MANO interface. The account name is 'osmjuju'. The connection status is 'SUCCESS'. The 'UPDATE ACCOUNT DETAILS' section contains the following fields:

- IP ADDRESS \*: 10.44.127.117
- PORT: 17070
- USERNAME: admin
- SECRET: [Redacted]

Annotations on the right side of the image point to specific fields:

- Account name (points to 'osmjuju')
- Juju controller container IP (nested inside VCA LXC) (points to '10.44.127.117')
- Juju controller API port (points to '17070')
- Controller credentials (points to 'admin' and the redacted 'SECRET' field)

# Hands-on: Our first VIMs

In this exercise, we will connect our first VIMs to be able to instantiate Network Services. Let's start with the VIM Emulator:

1. Using the OSM client, let's register the OpenStack VIM Emulator\*:

```
osm vim-create --name emu-vim1 --user username --password password --auth_url  
http://172.17.0.2:6001/v2.0 --tenant tenantName --account_type openstack
```

1. Check that it gets listed and its details shown:

```
osm vim-list
```

```
osm vim-show emu-vim1
```

\* More references about the VIM Emulator can be found at [https://osm.etsi.org/wikipub/index.php/VIM\\_emulator](https://osm.etsi.org/wikipub/index.php/VIM_emulator)

# Hands-on: Our first VIMs

Now, let's connect an OpenStack VIM that has been prepared for your POD

1. Using the OSM client, let's register the OpenStack VIM:

```
osm vim-create --name openstack-osm{X} --user osm{X} --password osm{X} --  
auth_url http://147.75.70.249:5000/v3 --tenant osm{X} --account_type openstack
```

...where {X} is your POD number (you can copy the command from the POD list)

1. Check that it gets listed and its details shown:

```
osm vim-list
```

```
osm vim-show openstack-osm{X}
```



Open Source  
**MANO**

# Onboarding my first NS/VNF

Section: Hands On!

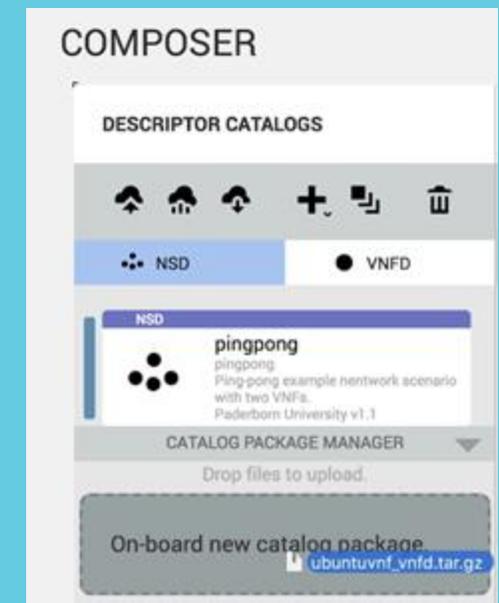


- The NS / VNF package containing the descriptors (at least) should be prepared first, options:
  - Downloading [examples](#) and editing them.
  - Using the Design-time Tools:
    - Build a base one [using the CLI tool](#), then customize it.
    - Build it [using the GUI Composer](#), from scratch.
- Once we have our packages, we need to [onboard them](#), using the OSM client or the GUI Composer. Note that VNFs should be onboarded first, since NS usually contain references to existing VNFs.

# Hands-on: Let's get a NS onboarded!

In this exercise, we will onboard a sample NS/VNF which will be instantiated later over a real OpenStack environment.

1. Download the packages from [here](#) to explore them together first.
2. Onboard the packages by dragging and dropping the tar.gz files from your computer to the 'Catalog' area. Ensure you upload the VNF package first, and then the NS package.
3. Check that the package options can be explored and modified using the composer tool (menus at the right). Double check that they appear using the OSM client as well.



# Hands-on: Let's get a NS onboarded!

Now, we will confirm that the NS/VNFs were correctly onboarded.

1. In 'Catalog' area, check the upload NSDs and VNFDs, you will be able to edit them as well.

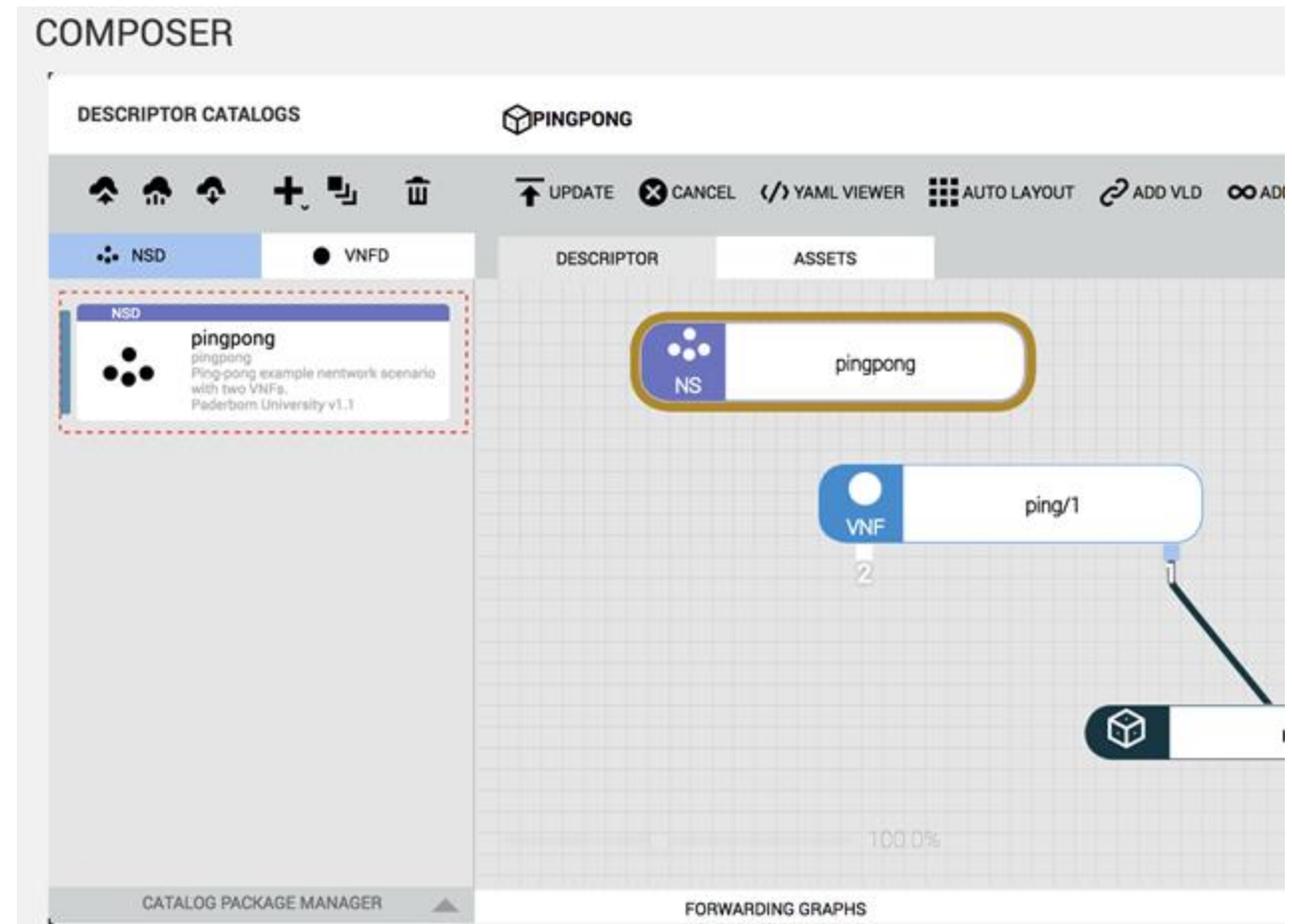
2. From the OSM Client, run:

```
# List VNF Descriptors
```

```
osm vnfd-list
```

```
# List NS Descriptors
```

```
osm nsd-list
```



The screenshot shows the OSM Composer interface. The title bar reads 'COMPOSER'. Below it, there are tabs for 'DESCRIPTOR CATALOGS' and 'PINGPONG'. The 'DESCRIPTOR CATALOGS' tab is active, showing a list of descriptors. A red dashed box highlights the 'pingpong' NS descriptor. The 'pingpong' NS descriptor is shown in a detail view, with a description: 'pingpong example network scenario with two VNFs. Paderborn University v1.1'. The 'pingpong' VNF descriptor is also visible in the 'ASSETS' tab. The interface includes a toolbar with icons for home, back, forward, add, delete, update, cancel, YAML viewer, auto layout, add VLD, and add. The bottom of the interface shows 'CATALOG PACKAGE MANAGER' and 'FORWARDING GRAPHS'.

# Hands-on: Let's onboard another NS!

Next, we will onboard sample NS/VNF packages that come with the VIM Emulator. This time we will use the OSM CLI.

1. Download them from [here](#) to explore them together first.
2. Onboard the packages, already present your host VM

```
# VNFs
```

```
osm upload-package vim-emu/examples/vnfs/ping.tar.gz
```

```
osm upload-package vim-emu/examples/vnfs/pong.tar.gz
```

```
# NS
```

```
osm upload-package vim-emu/examples/services/pingpong_nsd.tar.gz
```

# Hands-on: Let's onboard another NS!

Don't forget to ensure they have been uploaded accordingly.

1. In 'Catalog' area, check the upload NSDs and VNFDs, you will be able to edit them as well.

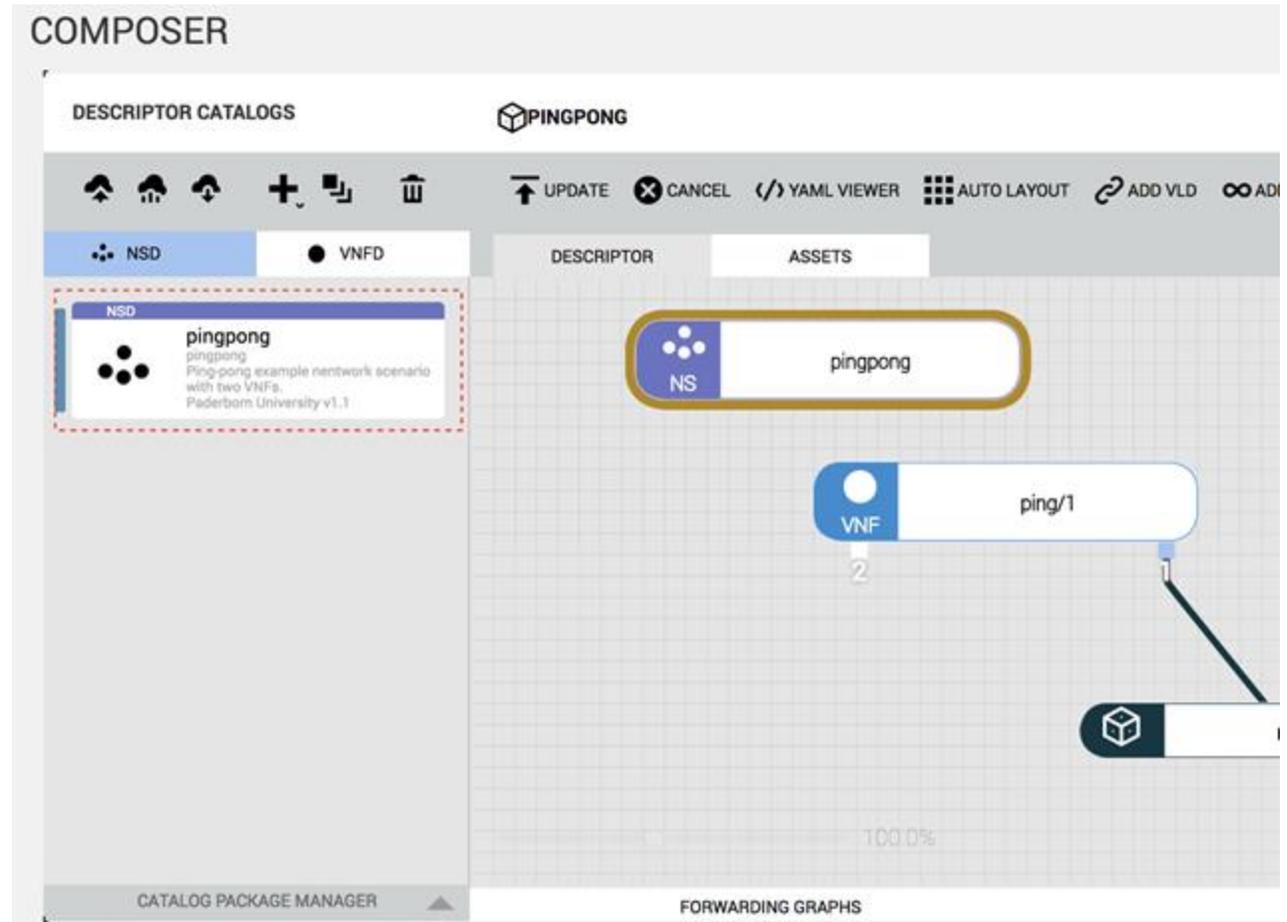
2. From the OSM Client, run:

```
# List VNF Descriptors
```

```
osm vnfd-list
```

```
# List NS Descriptors
```

```
osm nsd-list
```



The screenshot shows the OSM Composer interface. The top bar displays 'COMPOSER' and 'PINGPONG'. Below this, there are tabs for 'DESCRIPTOR CATALOGS' and 'PINGPONG'. The 'DESCRIPTOR CATALOGS' tab is active, showing a list of descriptors. A red dashed box highlights the 'pingpong' NSD entry, which is described as 'pingpong example network scenario with two VNFDs, Paderborn University v1.1'. The 'pingpong' NSD is being added to the 'pingpong' network scenario, which is shown as a graph with two VNFs (ping/1 and ping/2) and a network element (ping/1). The interface includes various icons for navigation and actions, such as 'UPDATE', 'CANCEL', 'YAML VIEWER', 'AUTO LAYOUT', 'ADD VLD', and 'ADD AD'. The bottom of the interface shows 'CATALOG PACKAGE MANAGER' and 'FORWARDING GRAPHS'.



Open Source  
**MANO**

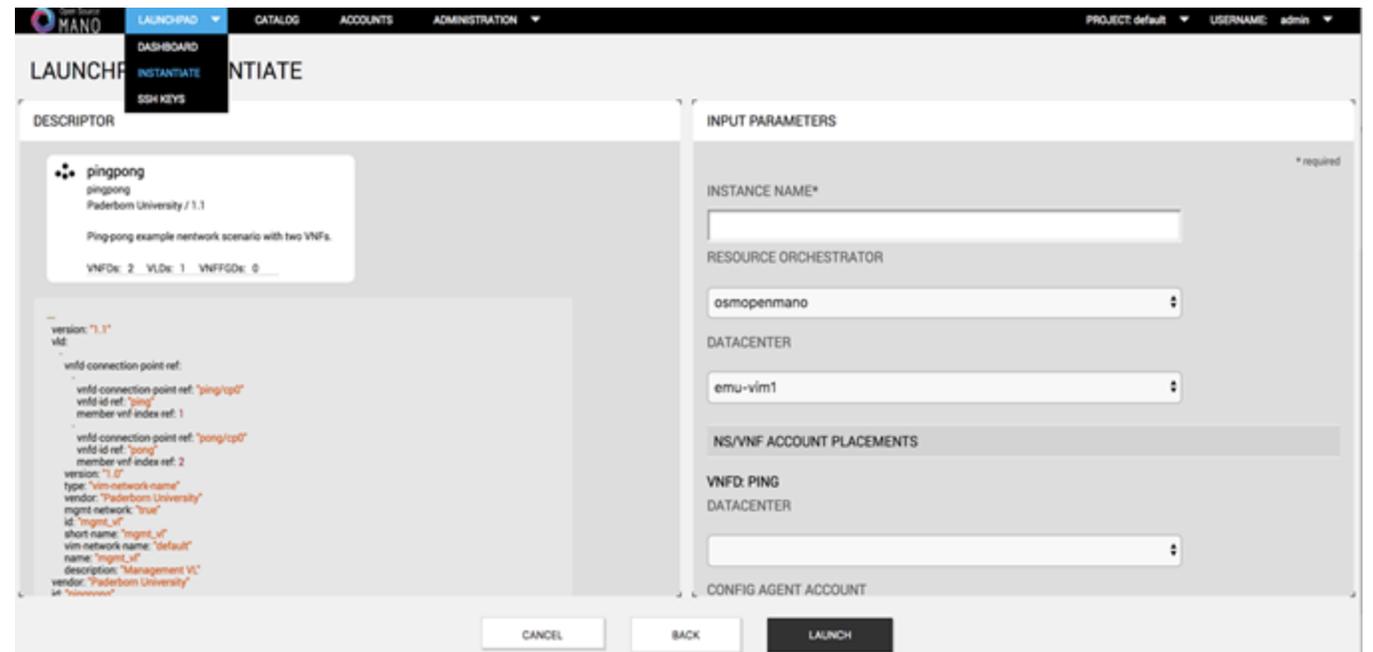
# Launching my first VNF instances

Section: Hands On!



# Launching Network Services

- Launching VNFs consists on instantiating a Network Service that already contains them.
- It can be done through two methods:
  - Using the GUI, 'Launchpad' menu.
  - Using the OSM client.



The screenshot shows the Open Source MANO web interface. At the top, there is a navigation bar with 'LAUNCHPAD', 'CATALOG', 'ACCOUNTS', and 'ADMINISTRATION'. The 'LAUNCHPAD' menu is open, showing 'DASHBOARD', 'INSTANTIATE', and 'SSH KEYS'. The 'INSTANTIATE' page is displayed, showing a 'pingpong' network service descriptor and its input parameters.

**DESCRIPTOR**

pingpong  
pingpong  
Paderborn University / 1.1  
Pingpong example network scenario with two VNFs.  
VNFDs: 2 VLDs: 1 VNFDGs: 0

```
version: "1.1"  
vld:  
  vnfd connection point ref:  
    vnfd connection point ref: "ping/vp1"  
    vnfd id ref: "ping"  
    member vnf index ref: 1  
  vnfd connection point ref: "pong/vp1"  
    vnfd id ref: "pong"  
    member vnf index ref: 2  
version: "1.0"  
type: "vim-network-name"  
vendor: "Paderborn University"  
mngt network: "true"  
id: "mgmt_vf"  
short name: "mgmt_vf"  
vim network name: "default"  
name: "mgmt_vf"  
description: "Management VNF"  
vendor: "Paderborn University"  
of "xxxxxxxxx"
```

**INPUT PARAMETERS**

INSTANCE NAME\*

RESOURCE ORCHESTRATOR  
osmopenmano

DATACENTER  
emu-vim1

NS/VNF ACCOUNT PLACEMENTS

VNFD: PING  
DATACENTER

CONFIG AGENT ACCOUNT

CANCEL BACK LAUNCH

# Hands-on: Let's instantiate our first NS!

In this exercise, we will instantiate the sample Network Service for the VIM Emulator first. We will use the OSM client for this.

1. While viewing the 'Dashboard', and using the OSM client, launch your first Network Service:

```
osm ns-create --nsd_name pingpong --ns_name test --vim_account emu-vim1
```

1. Check the status both in the GUI (Dashboard tab) and using the command line:

```
# From OSM Client
```

```
osm vnf-list / osm ns-list
```

```
# From VIM Emulator command line
```

```
docker exec vim-emu vim-emu compute list
```

# Hands-on: Let's instantiate our first NS!

Now that our first NS is active, let's interact with it. We will see that the VIM emulator launches each instance as a Docker container.

1. Being at the host VM shell, get inside the 'ping VNF' container

```
docker exec -it mn.dc1_test.ping.1.ubuntu /bin/bash
```

1. Once inside the 'ping VNF', check its networking interfaces the other 'pong VNF'

```
ifconfig
```

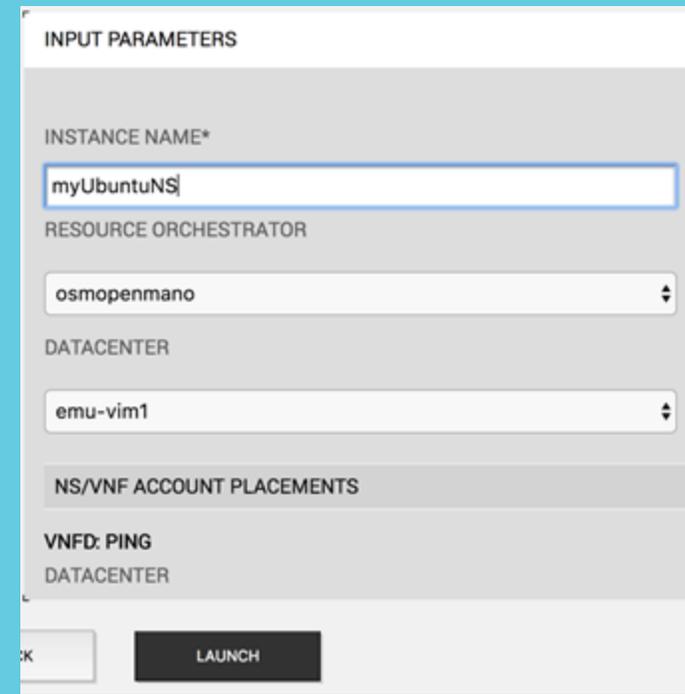
1. Ping the 'pong VNF' to check that the 'default' network has been correctly simulated.

```
ping 192.168.100.4
```

# Hands-on: Let's instantiate a new NS!

In this second exercise, we will instantiate a Network Service in a real OpenStack VIM. We will use the GUI for this.

1. Go to the GUI 'Launchpad' menu and select 'Instantiate'
2. Select the 'ubuntu\_nsd' and clic 'Next'
3. Put a name to your NS instance.
4. Select the OpenStack datacenter corresponding to your POD (example: "openstack-osm1")
5. Clic 'Launch'!



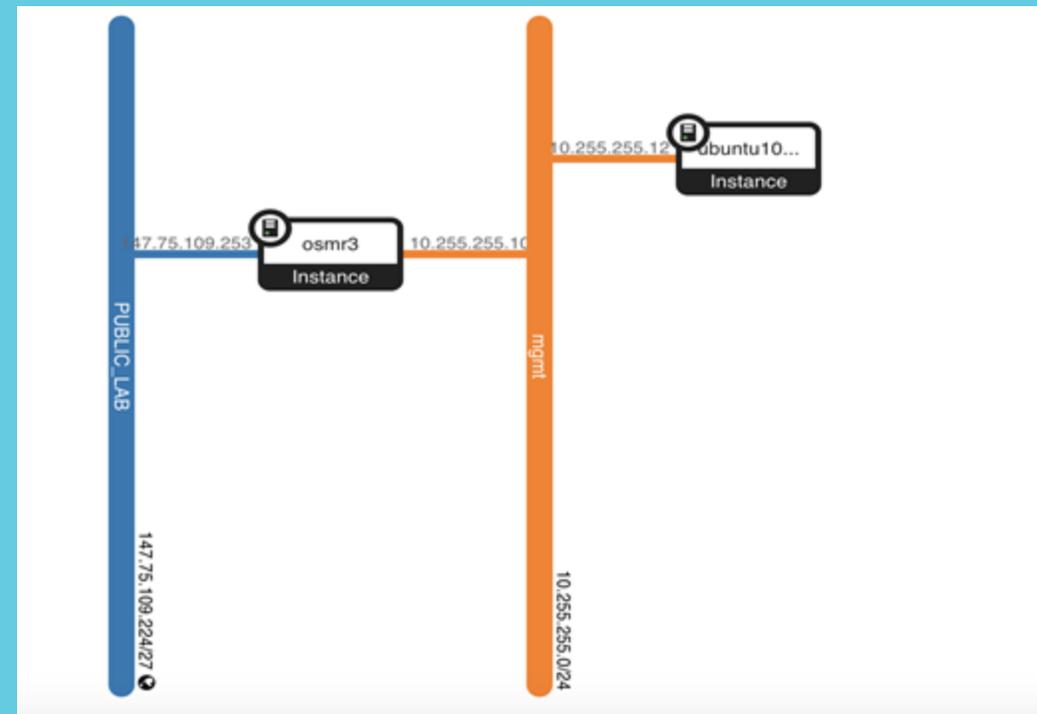
The screenshot shows a web form titled "INPUT PARAMETERS" for instantiating a Network Service. The form contains the following fields and options:

- INSTANCE NAME\***: A text input field containing "myUbuntuNS".
- RESOURCE ORCHESTRATOR**: A dropdown menu with "osmopenmano" selected.
- DATACENTER**: A dropdown menu with "emu-vim1" selected.
- NS/VNF ACCOUNT PLACEMENTS**: A section header.
- VNFD: PING**: A section header.
- DATACENTER**: A dropdown menu (partially visible).
- LAUNCH**: A prominent black button at the bottom right.

# Hands-on: Let's instantiate a new NS!

Now that our second NS is active, let's confirm that it actually appeared in the OpenStack VIM

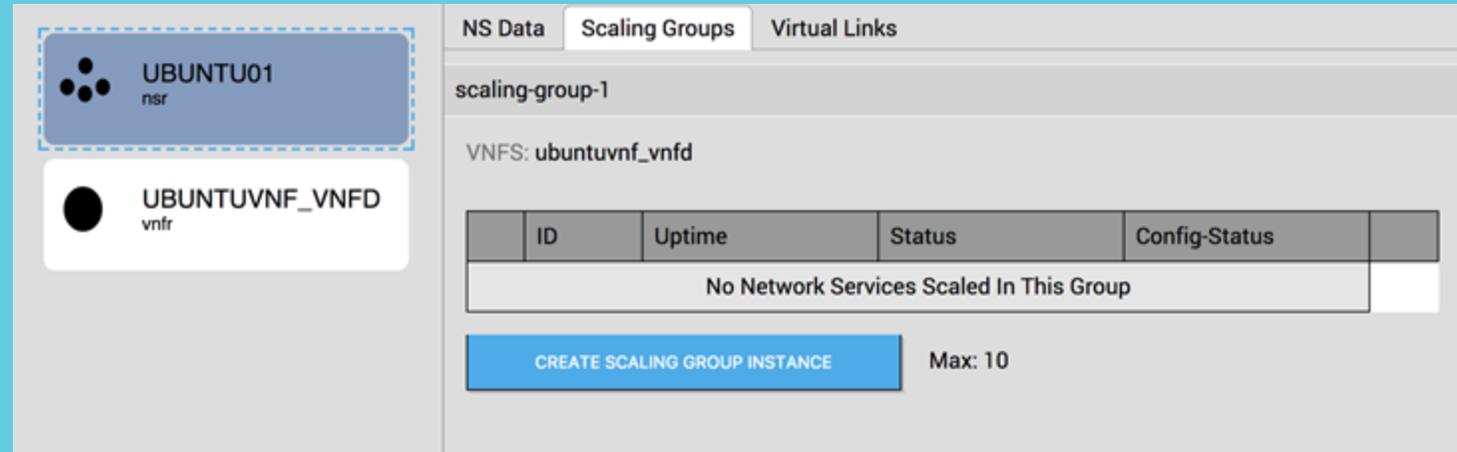
1. Go to the your OpenStack VIM at <http://147.75.70.249>  
(credentials: osm{X}/ osm{X} )
2. Go to Project → Network → Network Topology to see your instantiated VNF.
3. Access the console directly from the OSM GUI Dashboard by clicking the  icon



# Hands-on: Can we scale it?

This VNF's descriptor has been configured to be scaled-out. Let's scale it out manually and then see it appear at the VIM.

1. Click on the  link from the OSM GUI Dashboard, next to the NS name.
2. Click on the 'Scaling Groups' tab and then over the 'Create Scaling Group Instance' button.



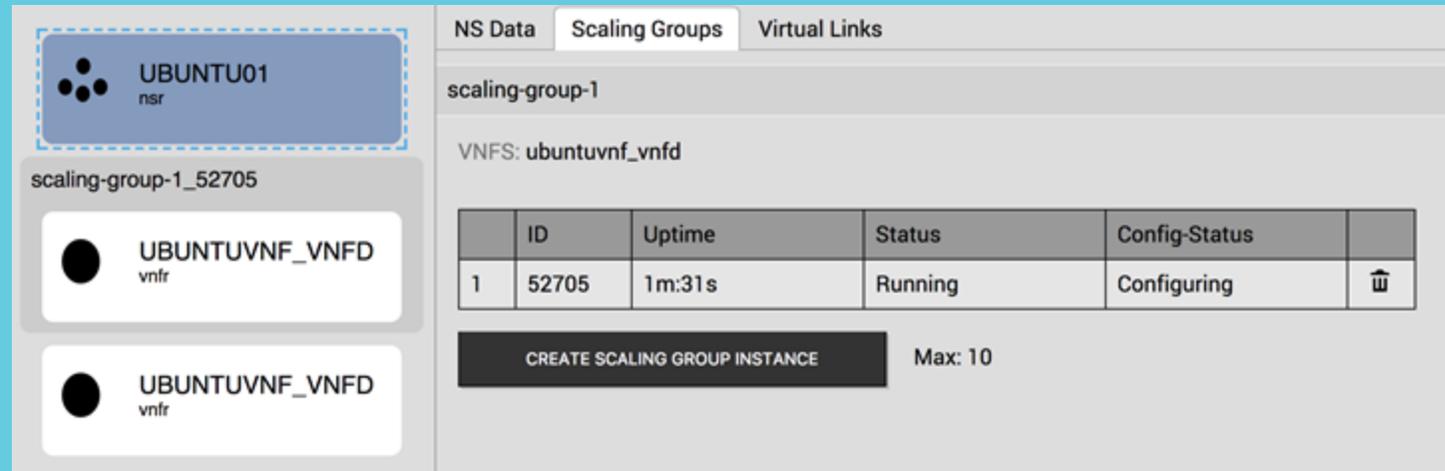
The screenshot shows the OSM GUI interface. On the left, there are two cards: 'UBUNTU01 nsr' (Network Service) and 'UBUNTUVNF\_VNFD vnfr' (Virtual Network Function Descriptor). The 'UBUNTU01 nsr' card is highlighted with a dashed blue border. On the right, the 'Scaling Groups' tab is active, showing a table for 'scaling-group-1' with columns for ID, Uptime, Status, and Config-Status. The table is currently empty, displaying the message 'No Network Services Scaled In This Group'. Below the table is a blue button labeled 'CREATE SCALING GROUP INSTANCE' and a 'Max: 10' indicator.

**Note! Scaling out/in and most other operations are also supported through the OSM client or directly interacting with the REST API**

# Hands-on: Can we scale it?

Autoscaling in or out is not officially supported in Release THREE.  
Let's manually scale the VNF "in" for now.

1. Click on the 'trash can' icon next to the scaling group instance created in the previous task.
2. You will see the VNF disappearing from both the OSM GUI and VIM platforms.



NS Data | **Scaling Groups** | Virtual Links

scaling-group-1

VNFS: ubuntuvnf\_vnfd

	ID	Uptime	Status	Config-Status	
1	52705	1m:31s	Running	Configuring	🗑️

CREATE SCALING GROUP INSTANCE Max: 10

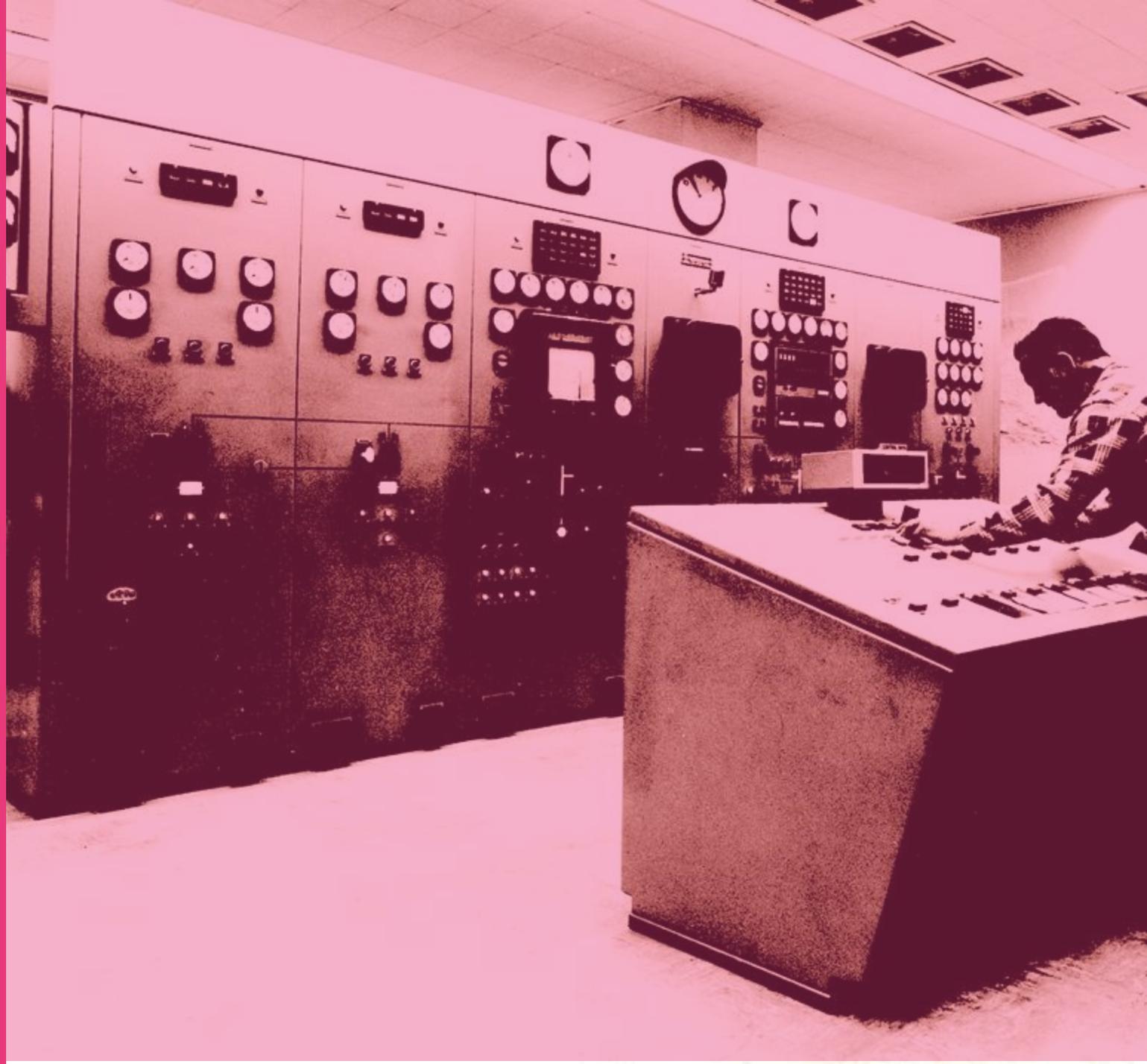
**Note!** Scaling out/in and most other operations are also supported through the OSM client or directly interacting with the REST API



Open Source  
**MANO**

# Managing VNF lifecycle

Section: Hands On!



- The main mechanism to manage VNF lifecycle during runtime is using proxy charms, managed by a Juju Controller at the VCA container.
- Proxy charms implement actions for VNF runtime configuration.
- To build a proxy charm:
  - Follow the guide [here](#) ('Discussion' tab includes further details), it will make use of existing 'layers' already in place that are able to send SSH commands to the VNF.
  - There is a recent contribution that adds support for Ansible, available here: [https://osm.etsi.org/wikipub/index.php/Example\\_VNF\\_Charms](https://osm.etsi.org/wikipub/index.php/Example_VNF_Charms)

# Hands-on: Let's manage this VNF!

The 'ubuntuvnf' descriptor includes a proxy charm. First of all, we'll check the code that implements the main actions for this VNF.

1. Explore the downloaded VNFD package, you will find a 'charms' folder.
2. Open the file located at:  
ubuntuvnf/reactive/ubuntuvnf.py
3. You will find a function called "say\_hello()" which is used for this example.  
Explore the code to see what it's supposed to do.

```
@when('actions.say-hello')
def say_hello():
    err = ''
    #try:
    # Put the code here that you want to execute
    param1 = "Hello " + action_get("name")
    cmd = "sudo wall -n " + param1
    result, err = charms.sshproxy._run(cmd)
    #except:
    #action_fail('command failed:' + err)
    #else:
    #    action_set({'output': result})
    #finally:
    remove_flag('actions.say-hello')
```

# Hands-on: Let's manage this VNF!

Now, let's check that the proxy charm was correctly loaded into the VCA container when the VNF was launched.

1. From the host's shell, go to the VCA container: `lxc exec VCA bash`
2. Run the juju command to confirm if the proxy charm is "active": `juju status`

```
root@VCA:~# juju status
Model  Controller  Cloud/Region  Version  SLA
default  osm          localhost/localhost  2.2.6    unsupported

App
ubuntuab-default-ubuntuab-ubuntuvnf-vnfd-b-b
Version  Status  Scale  Charm  Store  Rev  OS  Notes
active  1  ubuntuvnf  local  14  ubuntu

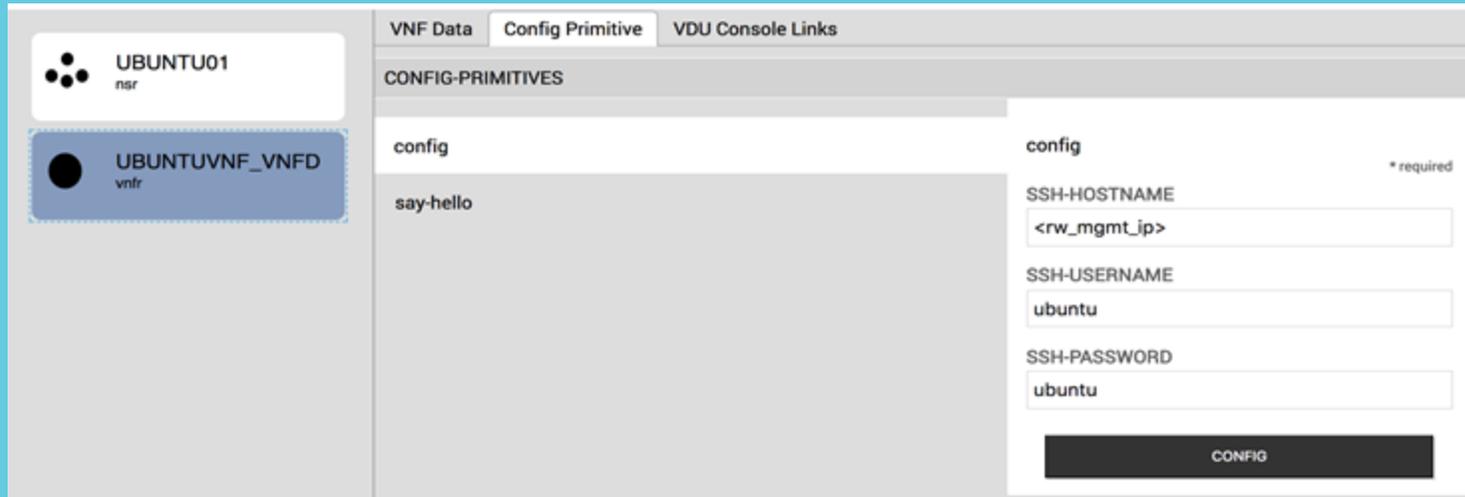
Unit
ubuntuab-default-ubuntuab-ubuntuvnf-vnfd-b-b/0*
5.255.4-ubuntu-ubuntu
Workload  Agent  Machine  Public address  Ports  Message
active  idle  14  10.44.127.136  ready with credentials 10.25

Machine  State  DNS  Inst id  Series  AZ  Message
14  started  10.44.127.136  juju-5f9726-14  xenial  AZ  Running
```

# Hands-on: Let's manage this VNF!

The proxy charm actions reflect into 'config primitives' that can be invoked from the GUI using forms and buttons.

1. Click on the  link in the OSM GUI Dashboard, next to the NS name.
2. Click on the VNF instance, 'Config Primitive' tab, and you will see a list of primitives:

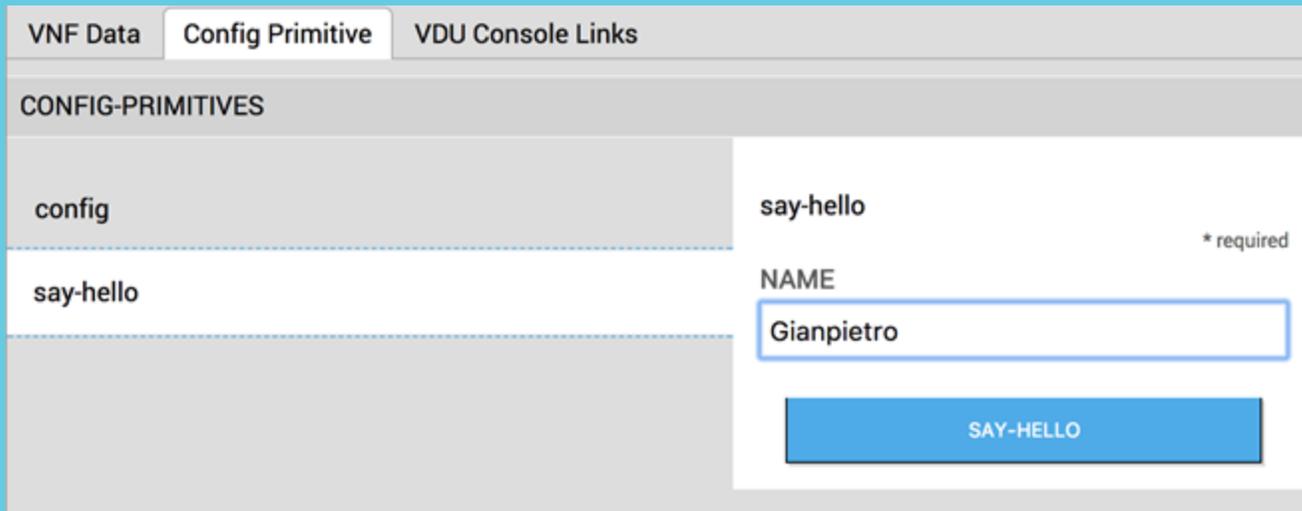


The screenshot shows the OSM GUI interface for managing a VNF instance. On the left, there is a sidebar with two items: 'UBUNTU01 nsr' and 'UBUNTUVNF\_VNFD vnfr'. The 'UBUNTUVNF\_VNFD vnfr' item is selected and highlighted. The main content area is divided into three tabs: 'VNF Data', 'Config Primitive', and 'VDU Console Links'. The 'Config Primitive' tab is active, showing a list of primitives under the heading 'CONFIG-PRIMITIVES'. The list contains two entries: 'config' and 'say-hello'. The 'config' primitive is selected, and its configuration form is displayed on the right. The form includes three input fields: 'SSH-HOSTNAME' with the value '<rw\_mgmt\_ip>', 'SSH-USERNAME' with the value 'ubuntu', and 'SSH-PASSWORD' with the value 'ubuntu'. A '\* required' label is next to the 'SSH-HOSTNAME' field. At the bottom of the form is a 'CONFIG' button.

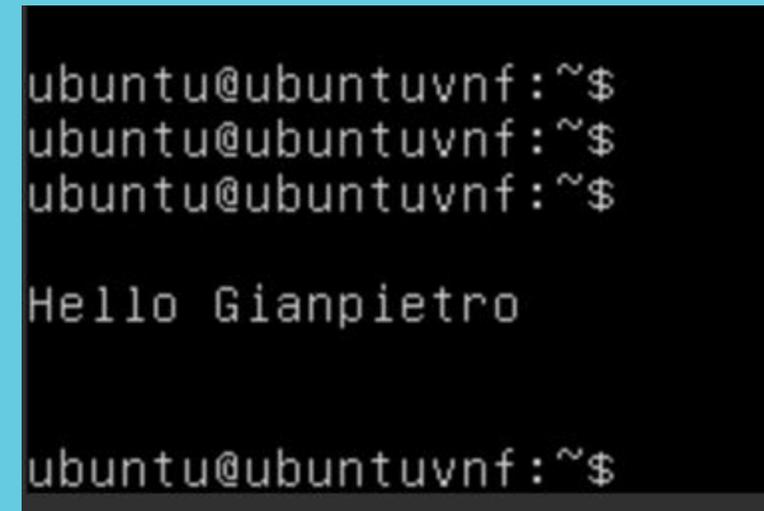
# Hands-on: Let's manage this VNF!

In this example, the “say hello” simple primitive has been implemented to send a message to the VNF’s console.

1. Go to your VNF console by clicking under “VDU Console Link” (ubuntu/ubuntu)
2. Using the ‘say-hello’ primitive, send a message to the VNF terminal



The screenshot shows a web interface with three tabs: "VNF Data", "Config Primitive", and "VDU Console Links". The "Config Primitive" tab is active, displaying a table of configuration primitives under the heading "CONFIG-PRIMITIVES". The table has two columns: "config" and "say-hello". The "say-hello" row is selected, showing a form with a "NAME" field containing "Gianpietro" and a "SAY-HELLO" button. A "\* required" label is visible next to the "say-hello" column header.



```
ubuntu@ubuntuvnf:~$  
ubuntu@ubuntuvnf:~$  
ubuntu@ubuntuvnf:~$  
  
Hello Gianpietro  
  
ubuntu@ubuntuvnf:~$
```

# Agenda

- Introduction: Architecture and OSM R3
- Hands On! - Installation, configuration and instantiation
- **Contributing to the Community**



Open Source  
**MANO**

# Joining OSM

Section: Contributing to the Community



# Joining the OSM Community

- Join [here](#) as a company or individual contributor!

## HOW TO GET INVOLVED IN OSM

There are two paths to get involved in OSM as an organisation: as an ETSI Member, or as an OSM Participant.

Check first if your organization is already involved by consulting the list of [OSM Members and Participants](#).

### Get involved as an ETSI Member

To take part in the development of OSM and participate to the meetings, ETSI Members need to sign the [OSM Membership Agreement and CCLA](#). In doing this, they agree to the OSM operating rules which in some cases are different from those in ETSI's Technical Working Procedures. [Check if your company is an ETSI Member](#).

### Get involved as an OSM Participant

Organizations who are not members of ETSI may also participate in OSM, attend meetings and help to develop OSM by making technical contributions. They are not applicable for leadership (LG) positions and must pay a participation fee to attend OSM meetings. To get involved as a Participant, please sign the [OSM Participant Agreement and the CCLA](#).

### Developers and Users

Individual developers and end users are welcome to contribute code and feedback to OSM, they just need to [create an individual contributor or user account](#).



Open Source  
**MANO**

# Activities

Section: Contributing to the Community



- Weekly Conference Calls
  - Technical, leadership, DevOps, and more!
- Face to Face Meetings
  - Plenaries and Mid-Release meetings (every 2-3 months)
  - Next locations: Oslo (Norway), Palo Alto (US)
- OSM Hackfest
  - [Second edition](#) had place on March 2018 at Spain, with tons of useful information for new comers and advanced OSM users.



Open Source  
**MANO**

# Ways to contribute

Section: Contributing to the Community



# Ways to contribute to OSM

- **Try OSM** and give feedback to the community.
- Join as a developer to **make contributions to the code**.
- Join the community to **contribute to design discussions**.
- **Start building your own distribution** of OSM as an integrator.
- **Host an OSM meeting** to contribute to the community's growth and diversity.

- OSM Release Three main wiki page  
[https://osm.etsi.org/wikipub/index.php/OSM\\_Release\\_THREE](https://osm.etsi.org/wikipub/index.php/OSM_Release_THREE)
- VIM Emulator, by Manuel Peuster  
[https://osm.etsi.org/wikipub/index.php/VIM\\_emulator](https://osm.etsi.org/wikipub/index.php/VIM_emulator)
- VIM Emulator video  
<https://youtu.be/Iji6FFIKL0w>



Open Source  
**MANO**

**Thank you!**

Questions about this presentation?

Contact Gianpietro Lavado at [glavado@whitestack.com](mailto:glavado@whitestack.com)

