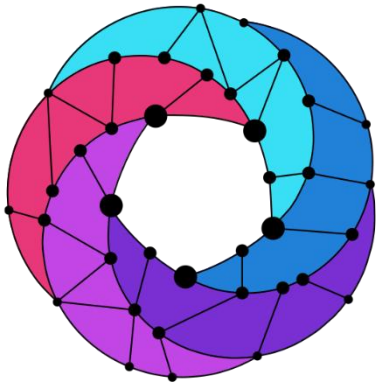


Open Source MANO

MWC DEMO COMPONENTS
TELEFÓNICA OPENMANO

Alfonso Tierno
Gerardo García
Francisco Javier Ramón

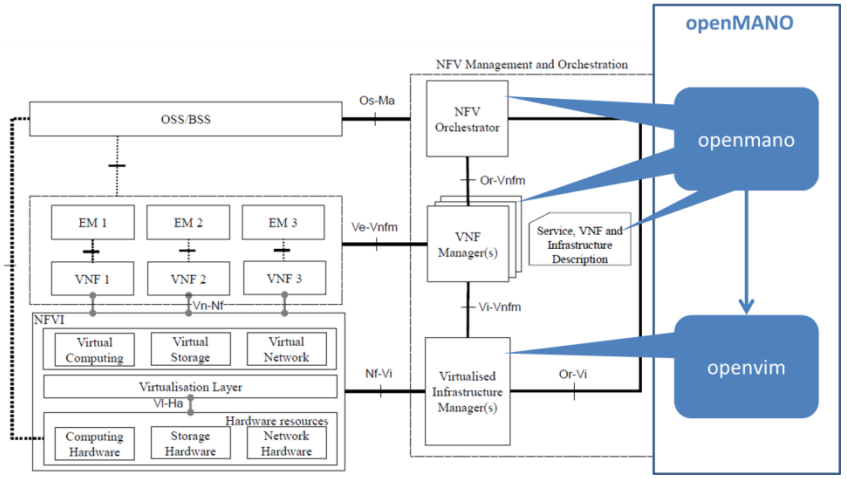
Pablo Montes
Antonio López

ETSI

OPENMANO: A PRACTICAL SOLUTION

Open: open source project released in GitHub under Apache 2 license

MANO: practical implementation of the reference architecture for Management & Orchestration (MANO) under standardization at ETSI's NFV ISG

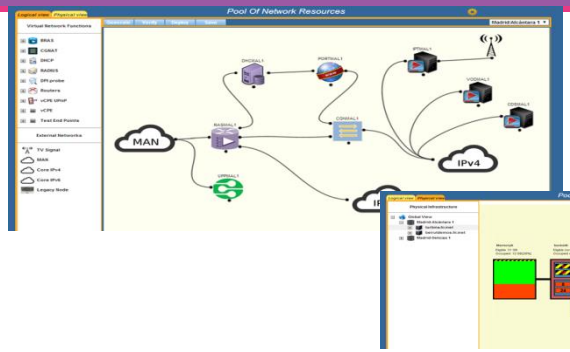


OpenMANO follows an NFVO-centric approach, granting the deterministic allocation of resources and with a simplified VNF instance lifecycle management at the NFVO (VNF instantiation and termination)

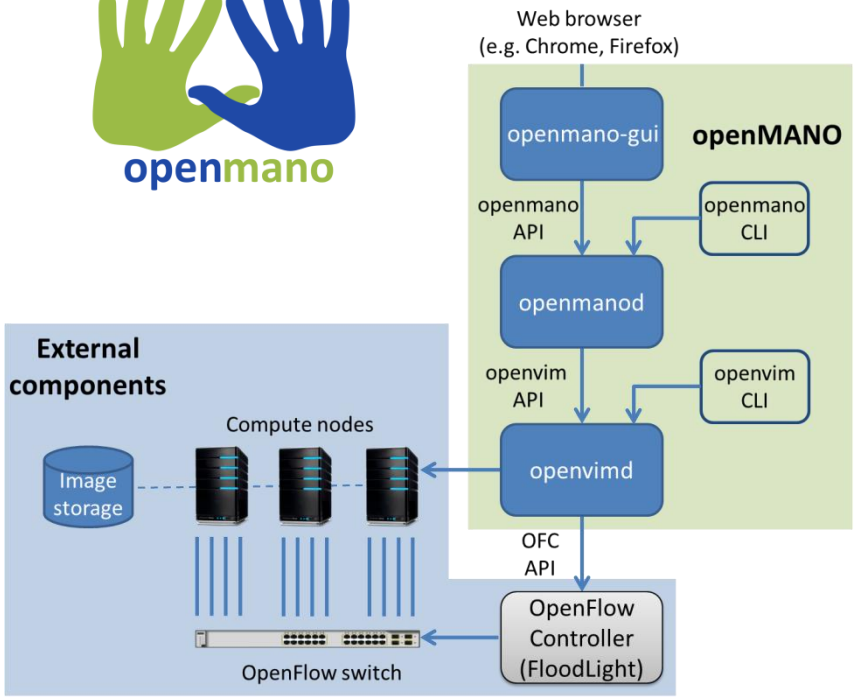
OPENMANO MAIN FEATURES

- **FRIENDLY FOR NETWORK ENGINEERS**
- **NETWORK SCENARIOS**
- Provides **NFVO & VIM** (+ **GUI** and **CLI**)
- Support for **HIGH PERFORMANCE VNFs** (EPA-aware)
- **REST-BASED APIs, OpenStack-friendly**
- MULTIVIM: OpenVim, OpenStack
- **MULTI-VENDOR** by design
  - **No formal integration needed**
  - Assures optimal VNF deployment and IXC
  - >30 VNFs tested

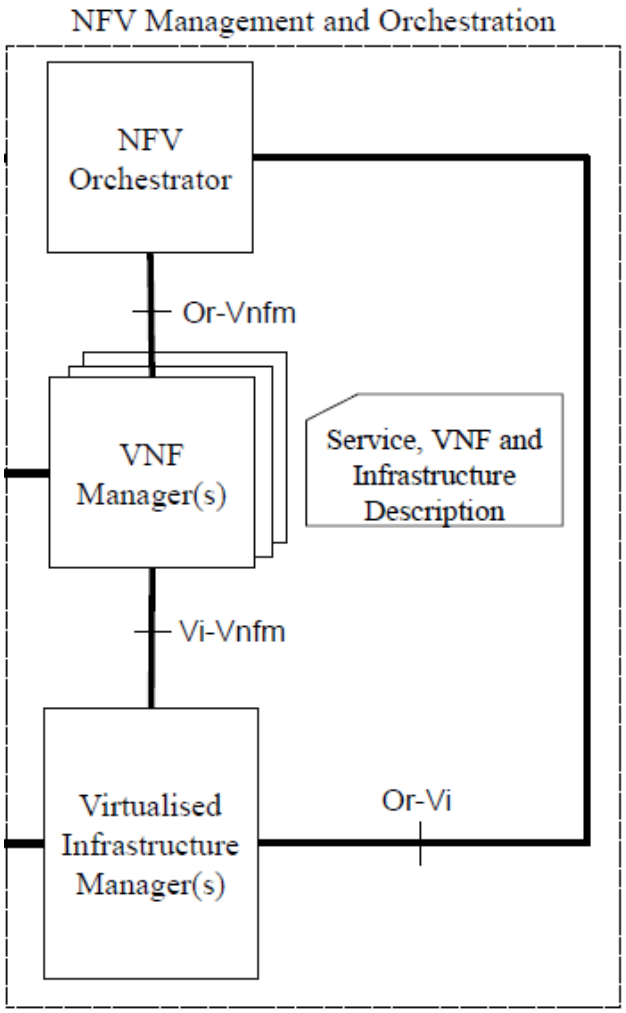
- OpenMANO code @ GitHub:
  - Python-based
  - 45k code lines
  - Released with Apache 2 license
  - 38 forks

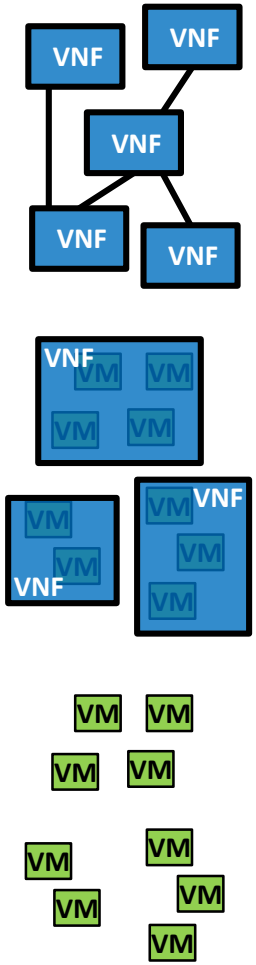


openmano



OPENMANO WORKS WITH **NETWORK SCENARIOS** VIA DESCRIPTORS...





NS
(NETWORK SCENARIO)

VNF
(SW-BASED NODE)

VM
(DEPLOYMENT UNIT)

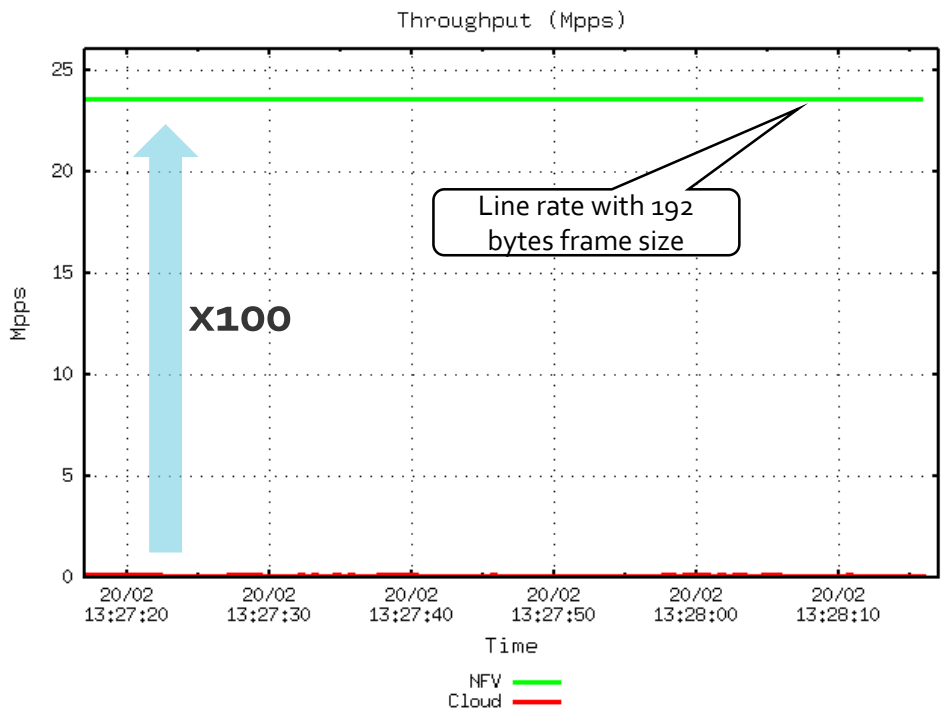
+

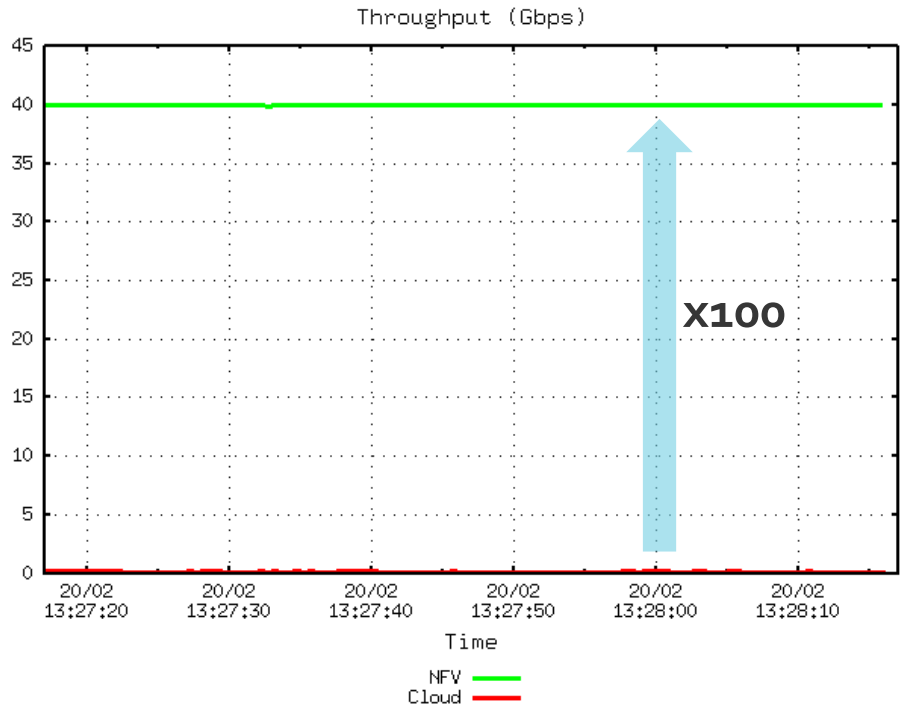
Abstraction

-

… PROVIDING ENHANCED PLATFORM AWARENESS (EPA) NATIVELY…

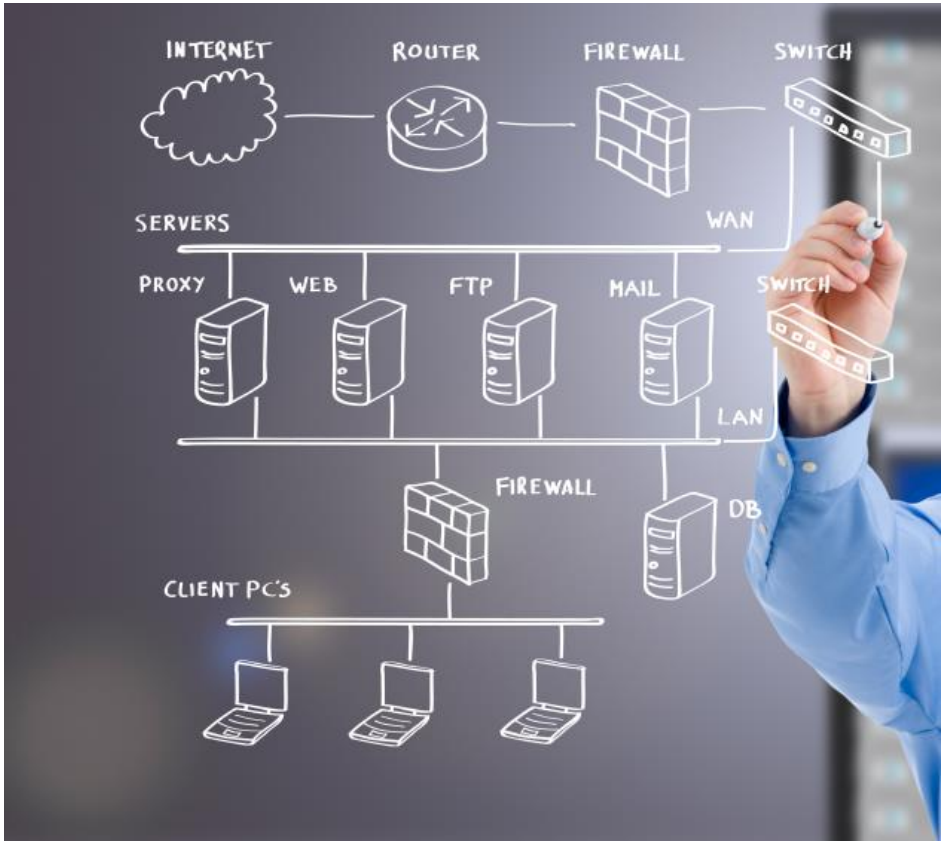
With the right exposure of HW resources to the VNFs, carrier-grade performance can be achieved.





Having x100 times better scalability should be sufficiently appealing! ☺

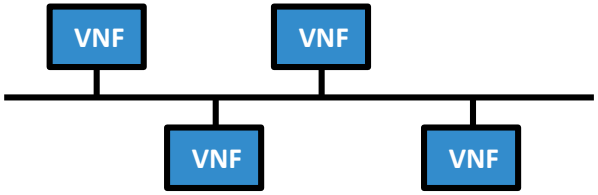
...WHILE A COMPREHENSIVE SET OF **CONNECTIVITIES** IS AVAILABLE



Support of L2 networks with **passthrough, SR-IOV or virtio** interfaces:

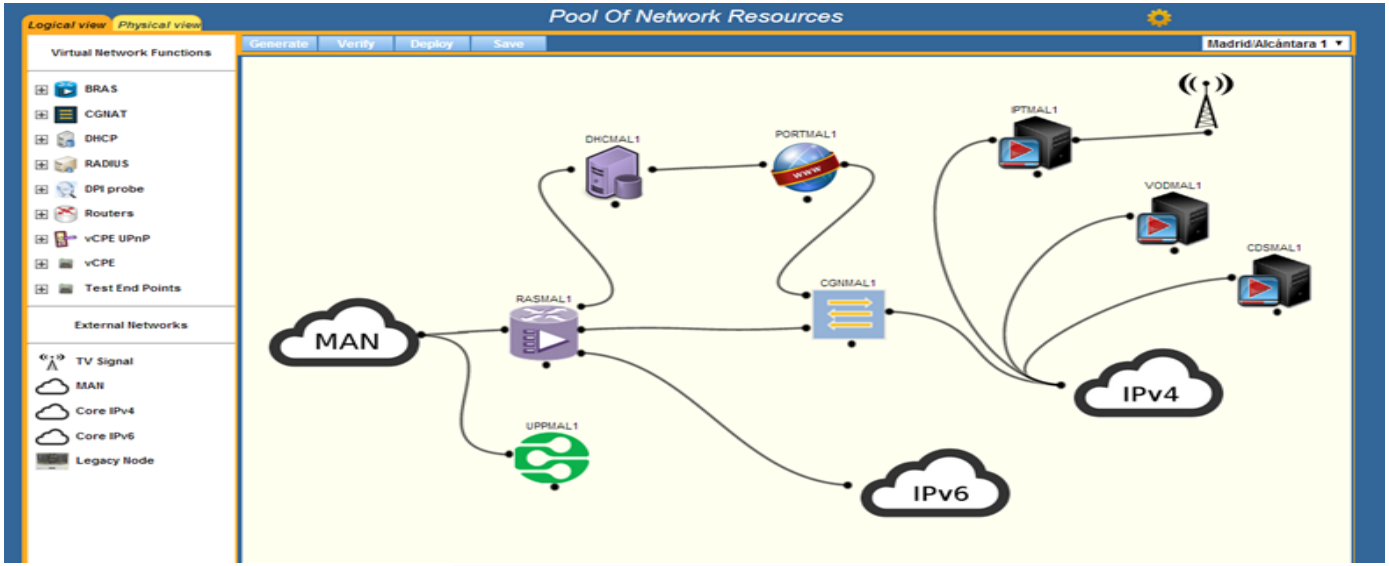
- **E-Line**

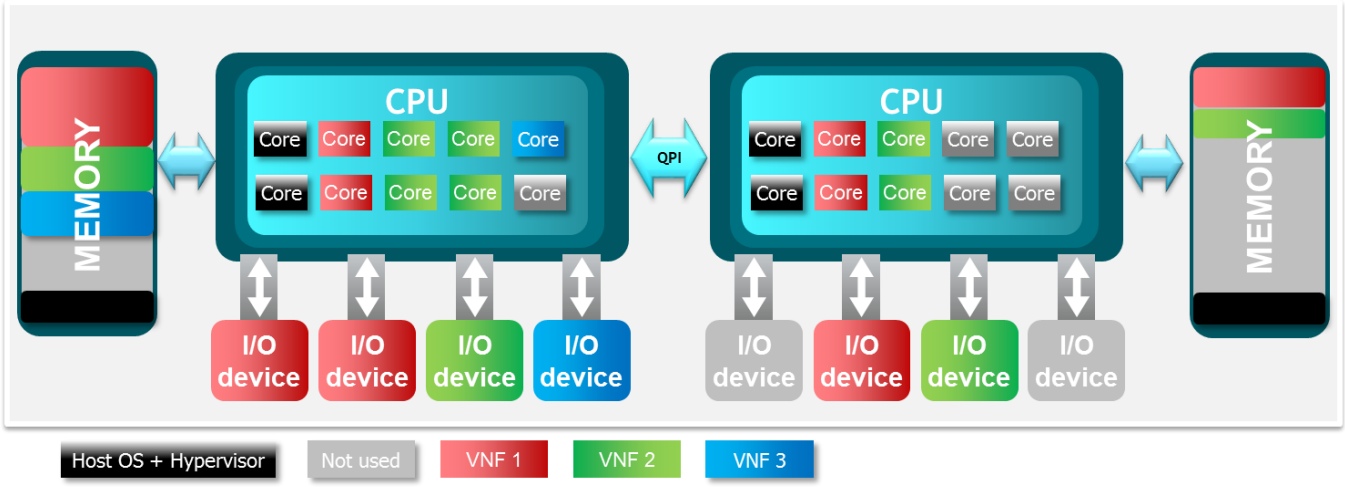
- **E-LAN**



Traditional E-LAN based on virtual bridges/switches is still supported

THIS COMBINATION HIDES LOW-LEVEL COMPLEXITY TO NETWORK ENGINEERS WHILE ASSURES CONSISTENT DEPLOYMENTS

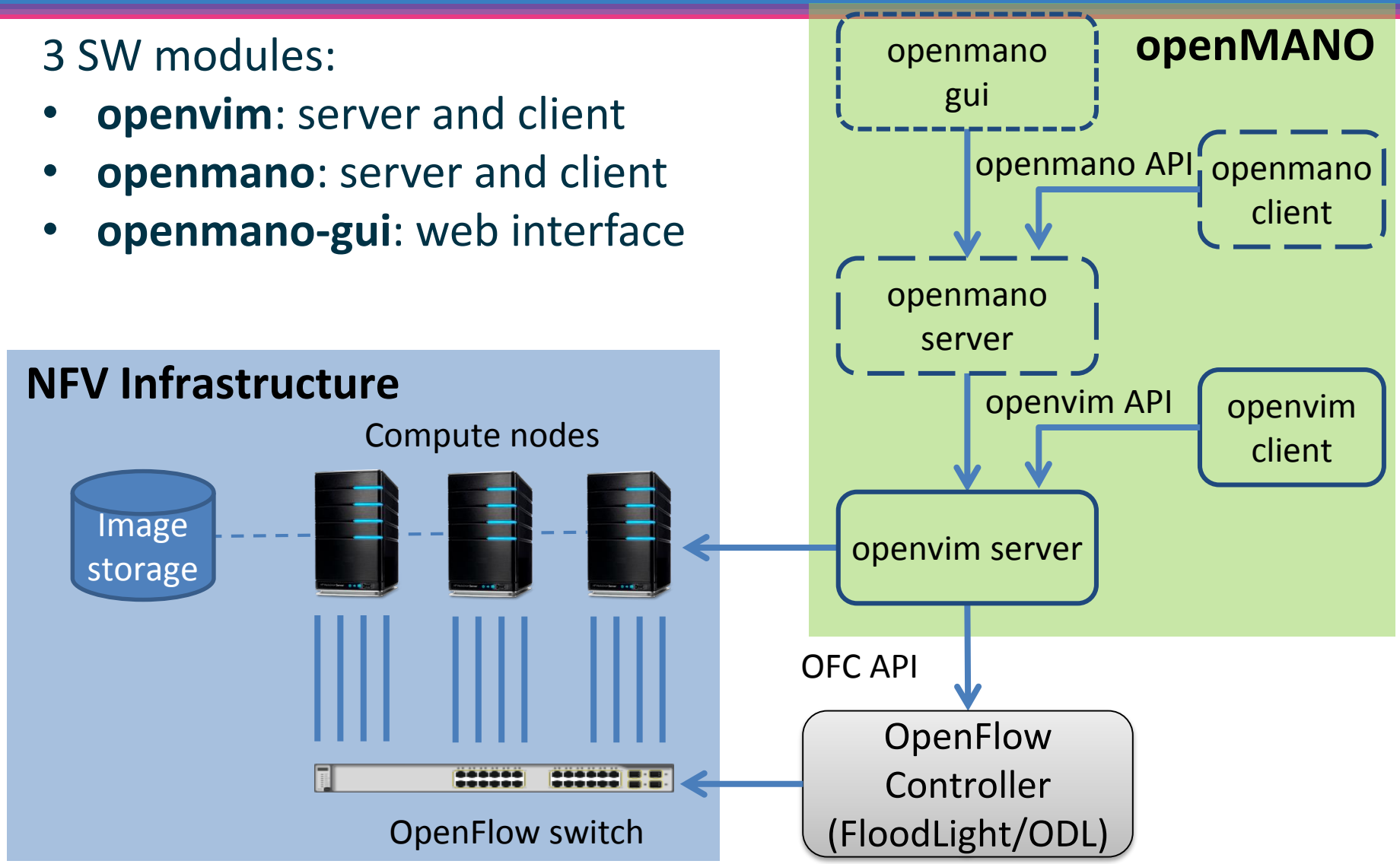




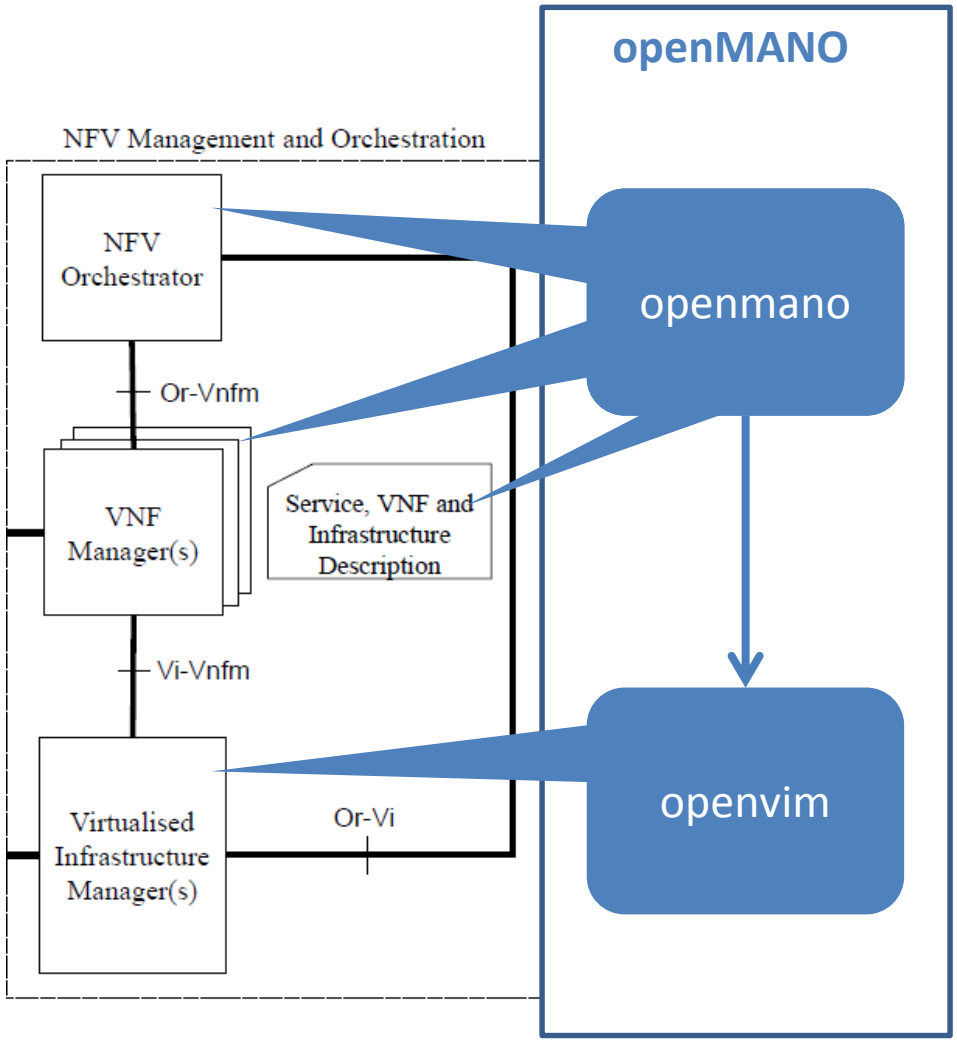
OPENMANO COMPONENTS

3 SW modules:

- **openvim**: server and client
- **openmano**: server and client
- **openmano-gui**: web interface



OPENMANO VS. OPENVIM



openmano

- Tenant and datacenter management
- VNF catalogue management
- Network Scenarios catalogue mgmt
- NS deployment (and VNF deployment)
- Simplified VNF life cycle mgmt

openvim

- Compute node mgmt
- NFVI tenant mgmt
- Image mgmt
- Flavor mgmt
- Network and port mgmt
- VM deployment with EPA support
- Native and bridged layer 2 networks

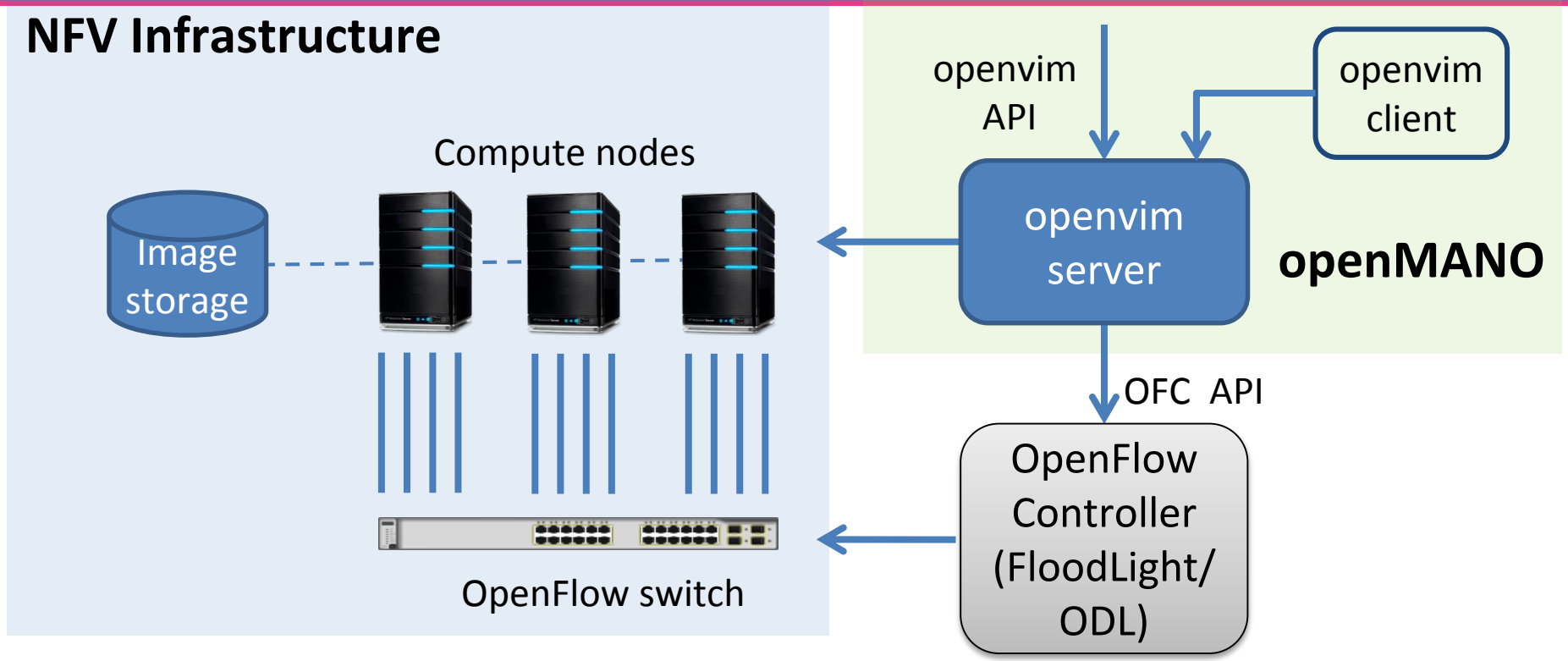
Open Source
MANO

OPENMANO COMPONENTS

Openvim: the VIM module
(not part of OSM)

ETSI

OPENVIM: THE VIM MODULE (I)
RELATION TO NFVI



Openvim + OFC controller (Floodlight/ODL) = NFV VIM

- Interaction with compute nodes through libvirt
- Tested on compute nodes based on Intel Xeon E5 processors, Linux as host OS, KVM as hypervisor
- Openflow switch controlled by proactive rules
- Image storage based on NAS. Image uploading is not managed by openvim

OPENVIM: THE VIM MODULE (II)
MODES OF OPERATION

5 modes to run openvim

MODE	Purpose	Required infrastructure
normal	Regular operation	Compute nodes OpenFlow switch
host only	Deploy without OpenFlow switch and controller	Compute nodes
development	VNF development (deploys without EPA)	"Low performance" compute node
test	Test openMANO installation and API	-
OF only	Test openflow integration	Openflow switch

OPENVIM: THE VIM MODULE (III)
MAIN CHARACTERISTICS

- **Host mgmt**
  - Administrative primitive (managed by an independent thread)
  - Host addition is done manually through a host descriptor file
  - Hosts can be administratively set up or down

ADMINISTRATIVE PRIMITIVE

ORDINARY USE

- **Tenant mgmt**
  - Tenants delimit the property and scope of flavors, images, vms, nets, etc.
  - No identity mgmt: neither users nor roles

ORDINARY USE

- **Networks mgmt**
  - Networks are pure L2 networks:
    - ptp: used to create an E-Line service between two data plane interfaces
    - data: used to create an E-LAN service with data plane interfaces
    - bridge_data: used to create an E-LAN service based on pre-provisioned linux bridges
  - No concept of subnet
  - Public vs private (tenant scope)

ORDINARY USE TO CREATE PUBLIC/SHARED NETWORKS

OPENVIM: THE VIM MODULE (IV)
MAIN CHARACTERISTICS

- **Ports mgmt**
  - Ports are attached to networks (similarly to OpenStack)
  - 2 types of ports:
    - Instance-related ports: VM interfaces created and deleted as part of the VM life cycle
    - External ports: set explicitly by the network administrator in order to define connections to PNF or external networks physically attached to the Openflow switch

ADMINISTRATIVE PRIMITIVE

ORDINARY USE TO CREATE EXTERNAL PORTS

- **Image mgmt**
  - Image uploading to the image repo must be done manually by the end user
  - Support of incremental images

TYPICALLY USED BY OPENMANO MODULE

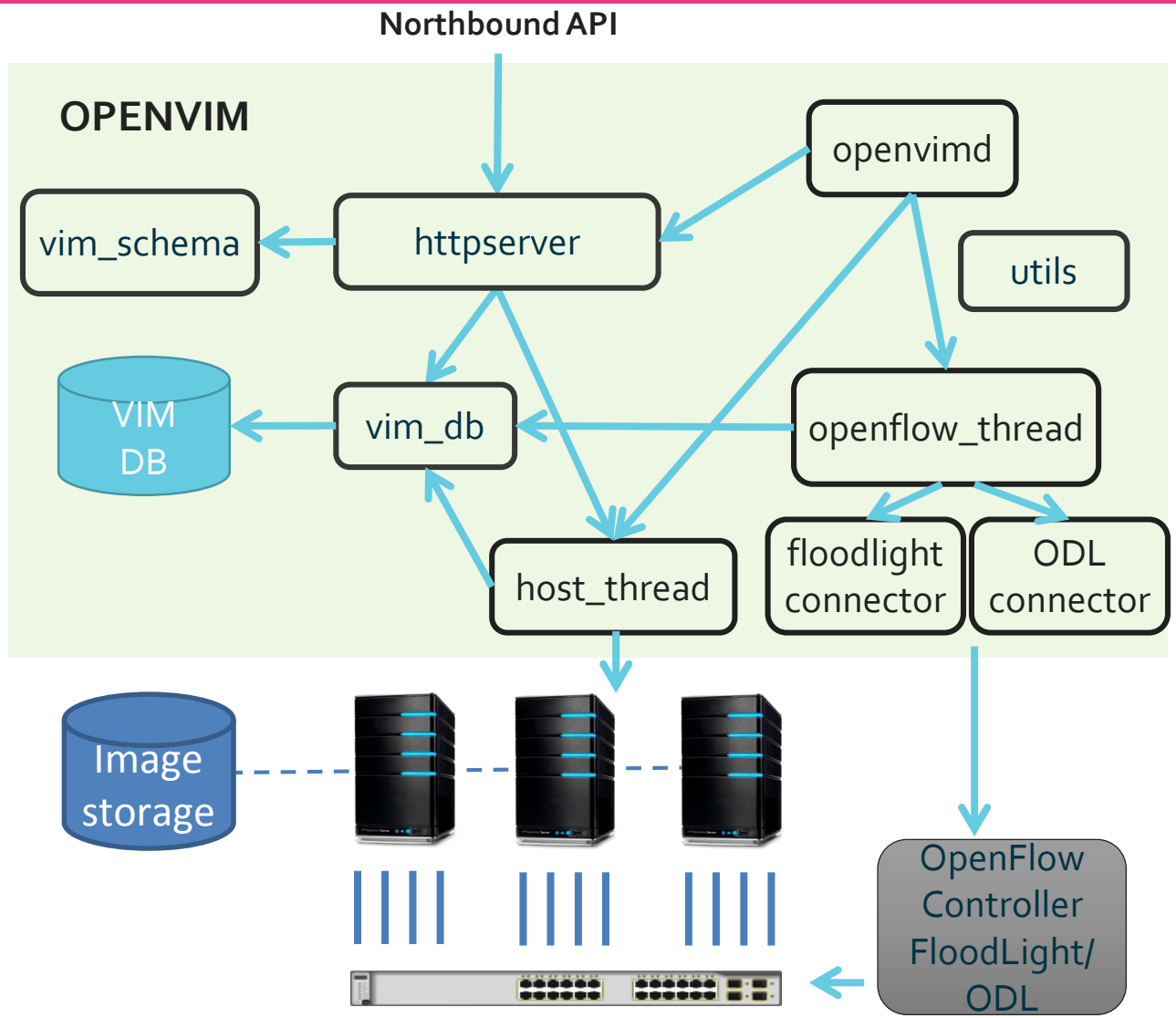
- **Flavor mgmt**
  - Openstack-like, but ***with new fields to indicate EPA requirements***

TYPICALLY USED BY OPENMANO MODULE

- **VM instance mgmt**
  - Besides traditional primitives (create, delete, list), allows actions over VMs (shutdown, start, pause, resume, rebuild, reboot)

ORDINARY USE

OPENVIM: THE VIM MODULE (V) COMPONENTS



openvimd.py
- Main program

host_thread.py
- Thread that interacts with the compute node through libvirt to manage VM instances
- One thread per compute node

httpserver.py
- Thread that manage northbound API requests
- Two threads, for common requests and for administrative ones

vim_db.py
- Module used to interact with the openvim DB
- General table management and transactional-based writings

vim_schema.py
- Dictionary schemas used to validate API request content using jsonschema library

openflow_thread.py
- Interacts with an OpenFlow Floodligth/ODL controller to create dataplane connections

OPENVIM: THE VIM MODULE (VI)
NORTHBOUND API

REST-based API, intentionally similar to Openstack API

Example 1. Removing a flavor

Request: DELETE /openvim/{tenant_id}/flavors/{flavor_id}

Response: 200 OK, 400 Bad Request, …

Example 2. Listing images

Request: GET /openvim/{tenant_id}/images

Response:

200 OK

```
{
  "images": [
    { "id": "70a599e0-31e7-49b7-b260-868f441e862b", "path": /opt/image1.raw", "name": "image1"},
    { "id": "155d900f-4e14-4e4c-a73d-069cbf4541e6", "path": "/opt/image2.qcow2", "name": "image2"}
  ]
}
```

OPENVIM: THE VIM MODULE (VII)
NORTHBOUND API DOCUMENTATION

URL: http://github.com/nfvlabs/openmano/raw/master/docs/openvim-api-0.6.pdf

3.3.3 POST /openvim/{tenant_id}/images

Create image

Params: (Extra parameters are ignored)

- ***id***:(optional): proposed uuid
- ***path***: path where iso/qcow2 image is present.
- ***name***:(Mandatory): user name
- ***description***:(optional): user description

Content-type: application/json

```
{
  "image": {
       "id": "70a599e0-31e7-49b7-b260-868f441e862b",
       "path": "/local/path/where/isoqcow2/is/present",
       "metadata": {
            "architecture": "x86_64",
            "use_incremental": "no",
            "vpci": "0000:07:00.0",
            "os_distro": "ubuntu",
            "os_type": "linux",
            "os_version": "14.04",
       },
       "minDisk": 0,
       "minRam": 0,
       "name": "fakeimage7",
       "description":  "user description"
  }
}
```

OPENVIM: THE VIM MODULE (VIII)
OTHER DETAILS

- openvimd.cfg -> configuration file
  - Mode of operation
  - IP addresses and ports where the HTTP servers is listening to API requests
  - OFC parameters
  - DB parameters
  - Parameters for the new networks: VLAN tags range, compute node bridges

- openvim client, Python-based

```
$ openvim -h
usage: openvim [-h] [--version]
               {config,image-list,image-create,image-delete,image-edit,vm-list,vm-
create,vm-delete,vm-edit,vm-shutdown,vm-start,vm-rebuild,vm-reboot,vm-createImage,port-
list,port-create,port-delete,port-edit,port-attach,port-detach,host-list,host-add,host-
remove,host-edit,host-up,host-down,net-list,net-create,net-delete,net-edit,net-up,net-
down,flavor-list,flavor-create,flavor-delete,flavor-edit,tenant-list,tenant-
create,tenant-delete,tenant-edit,openflow-port-list,openflow-clear-all,openflow-net-
reinstall,openflow-net-list}
               ...

User program to interact with OPENVIM-SERVER (openvimd)
```

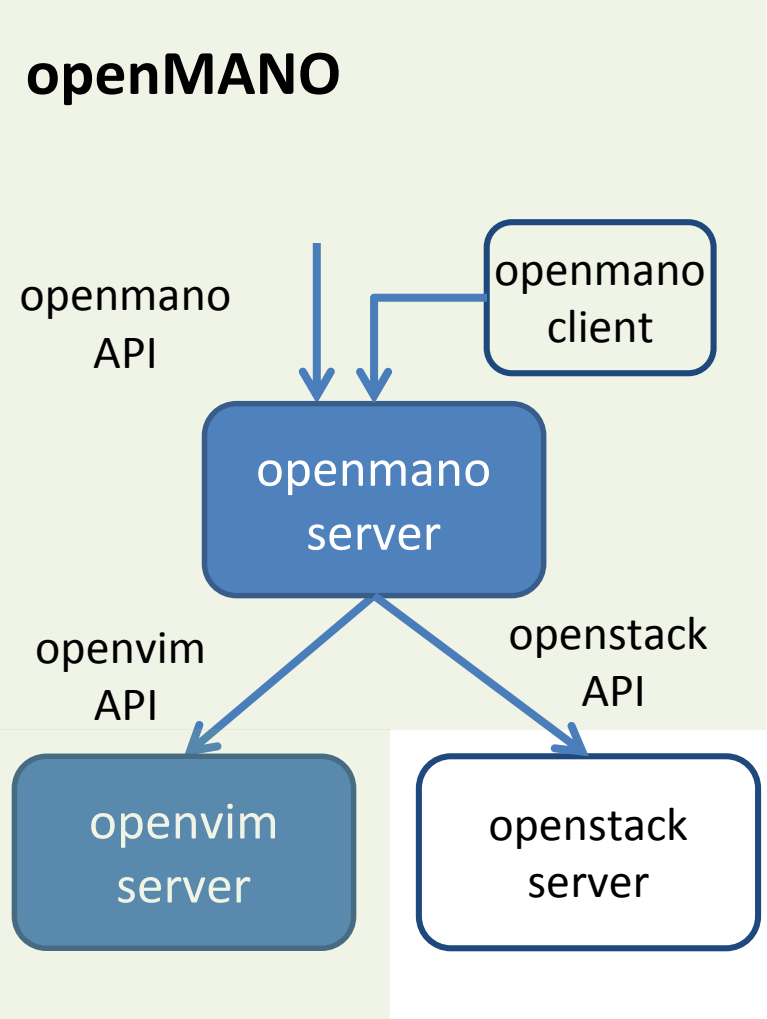
Open Source
MANO

OPENMANO COMPONENTS

Openmano: the NFVO+VNFM module (part of OSM)

ETSI

OPENMANO RELATION TO VIM



CHARACTERISTICS

- **Hides complexity to the network engineer:**
  - No compute nodes
  - No VMs
  - Just nodes and links
- **VNF definitions via descriptors**
- **NS definitions via descriptors**
- **NS instance creation and termination (and associated VNF creation)**

OPENMANO
TENANT AND DATACENTER MANAGEMENT

- **Tenant mgmt**
  - Tenants delimit the property and scope of VNF and NS, and the actions over them (instantiation, termination)
  - Separate from openvim tenant space
    ➔ Different programs with different databases

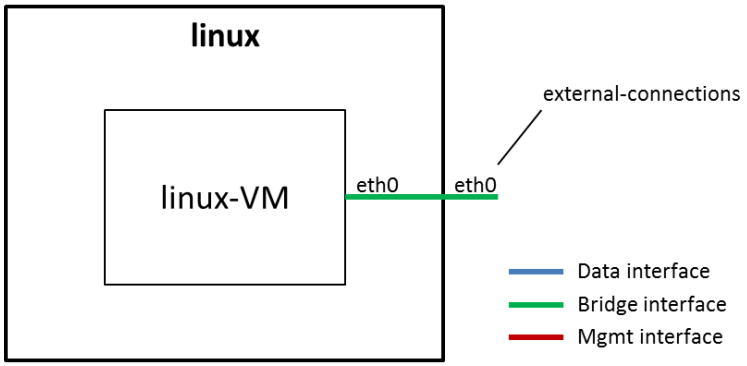
- **Datacenter mgmt**
  - A new datacenter must be added in order to interact with a specific pool of resources. The datacenter is characterized by:
    - Type: openvim (by default) or openstack
    - URL of the VIM that manage that datacenter
    - VIM configuration attributes, e.g. suppress security port
  - Datacenters are not directly available to tenants
  - An openmano tenant must be attached to a datacenter and a VIM tenant
  - Datacenter nets can be inherited as external networks to be used

OPENMANO
VNF CONCEPT AND STRUCTURE

- VNF: SW-based network function that can be deployed on an NFV datacenter
- VNF definition vs VNF instance ⇔ class vs object
- VNF structure: VNFCs/VMs, internal-connections, external-connections

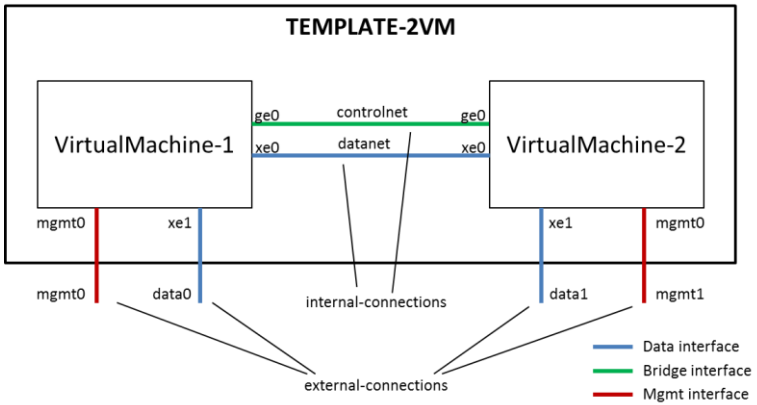
Single-VM VNF

vnfs/examples/linux.yaml



Multi-VM VNF

vnfs/vnf-template-2vm.yaml
(multi-VM VNF)



OPENMANO
VNF DESCRIPTOR

Name: unique name of the VNF
Description
External-connections:

- External interfaces of that VNF that can be connected in an NS to other VNFs or networks
- Properties:
  - name
  - type: mgmt/bridge/data
  - mapping to a VNFC interface

Internal-connections:

- It defines how VNFC/VMs are interconnected. This property is only required in case of VNFs consisting of several VMs
- Properties:
  - name
  - type: mgmt/bridge/data
  - list of interconnected VNFC/VMs (and their interfaces)

VNFC:

- List of components or virtual machines this VNF is composed of.

OPENMANO
VNF DESCRIPTOR (CONTINUATION)

VNFC properties:

- name
- description
- image path:
  - When a new VNF is added to the catalogue, new VM images are created in openvim for each VNFC based on this path
- vcpus: number of virtual CPUs (traditional cloud requirement
- ram: number of virtual CPUs (traditional cloud requirement
- bridge-ifaces: virtio interfaces with no high I/O performance requirements. They will be attached to Linux bridges in the host.

TRADITIONAL REQUIREMENTS

- numas: CPU, memory and interface requirements for high I/O performance
  - CPU type (cores, paired threads, threads), number and pinning
  - Memory (number of 1G hugepages)
  - Interface details:
    - Type: passthrough, VLAN-based SRIOV, MAC-based SRIOV
    - Bandwidth
    - Virtual PCI to assure appropriate identification at the VM

EPA REQUIREMENTS

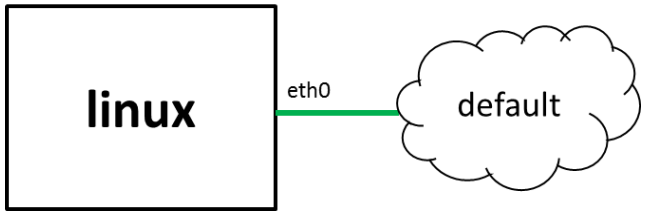
- devices: additional devices can be included (disk, cdrom, etc.) in this section

OPENMANO
NS CONCEPT AND STRUCTURE

- NS: topologies of VNFs and their interconnections
- NS definition vs NS instance ⇔ class vs object
- NS structure: VNFs, networks (external and internal)

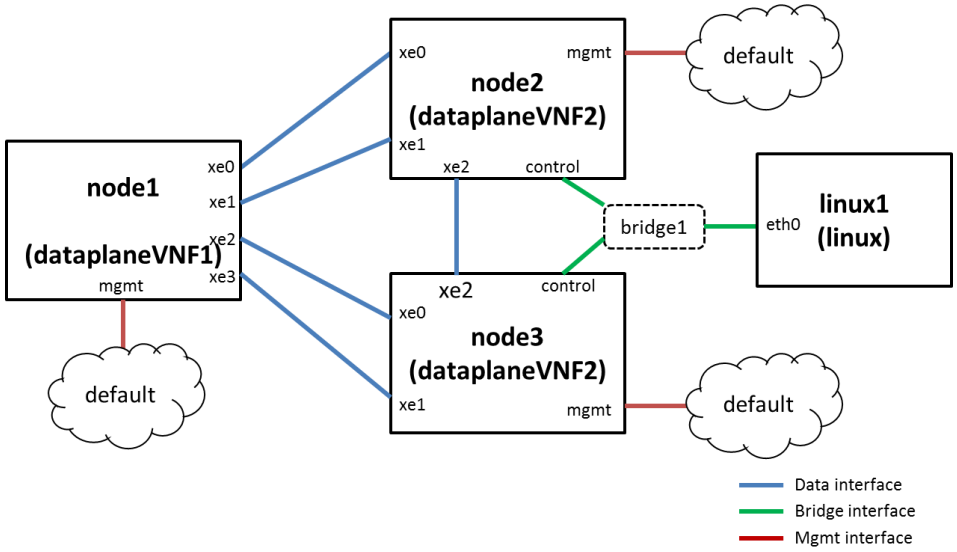
Simple

scenarios/examples/simple.yaml



Complex

scenarios/examples/complex.yaml



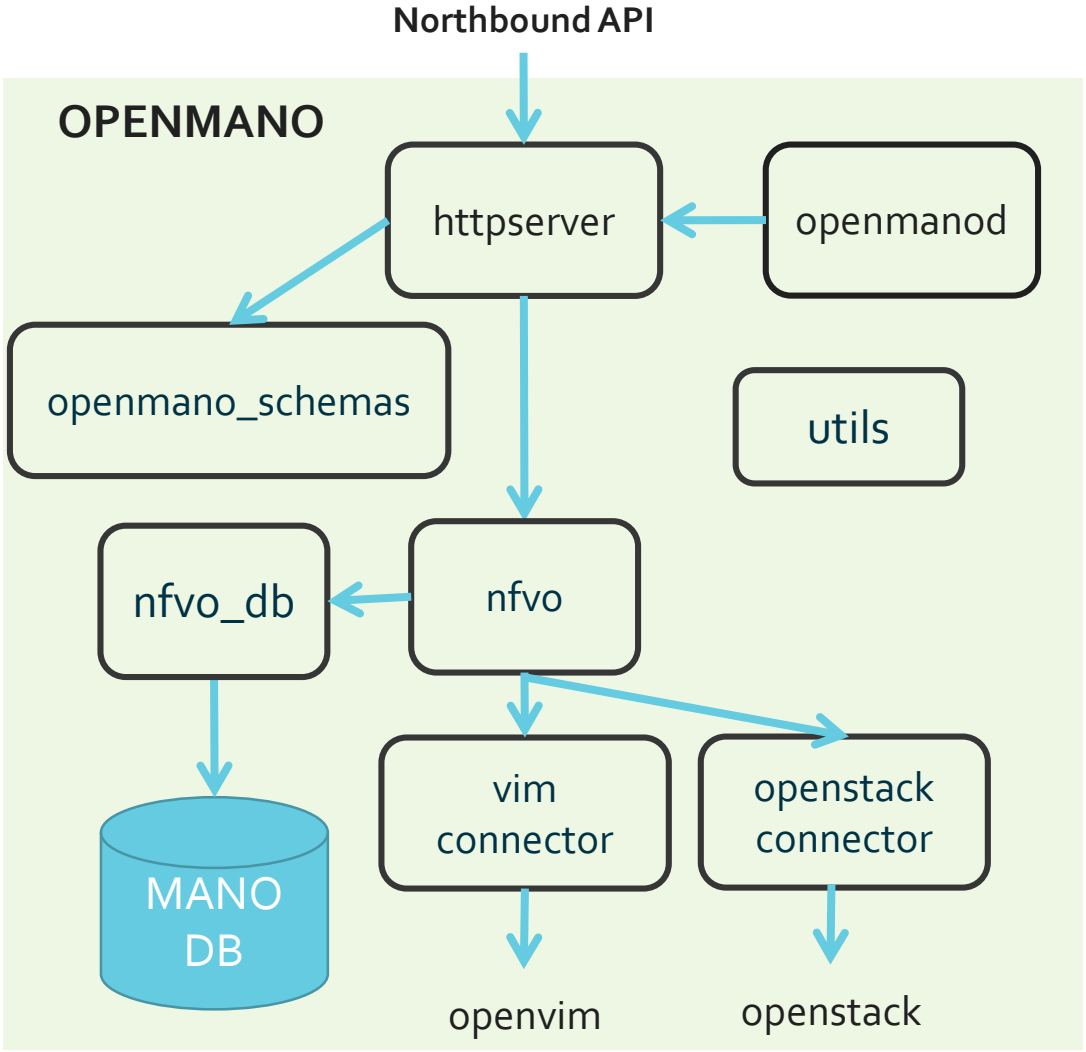
OPENMANO
NS DESCRIPTOR

Name: unique name of the network scenario
Description
Topology: defines the VNFs and the networks interconnecting them

VNFs
- name
- VNF model (id or unique name) : must match a previously created VNF

Networks:
- name
- type:
  - name of the network (in case of external/public datacenter network)
  - bridge (for control plane internal/private networks)
  - dataplane (for data plane internal/private networks)
- list of VNFs and interfaces connected to that network.

OPENMANO COMPONENTS



openmanod.py
- Main program

httpserver.py
- Thread that manage northbound API requests

nfvo.py
- NFVO engine, implementing all the methods for the creation, deletion and management of vnfs, scenarios and instances

nfvo_db.py
- Module used to interact with the openmano DB

openmano_schemas.py
- Dictionary schemas used to validate API request and response content using jsonschema library

vim/openstack connector.py
- Interacts with an openvim-based/openstack VIM through the openvim/openstack API

OPENMANO NORTHBOUND API

- REST-based API
- Not documented

Example 1. Removing a VNF

Request: DELETE /openmano/{tenant_id}/vnfs/{vnf_id}

Response: 200 OK, …

Example 2. Listing network scenarios

Request: GET /openmano/{tenant_id}/scenarios

Response:

200 OK


```
{
  "scenarios": [
    {
      "uuid": "ec1d744e-f56b-11e4-874f-52540032c4fa",
      "created_at": "2015-05-08T12:20:59",
      "description": "Simple network scenario consisting of a single VNF connected to an external network",
      "name": "simple",
      "public": false
    },
  ]
}
```

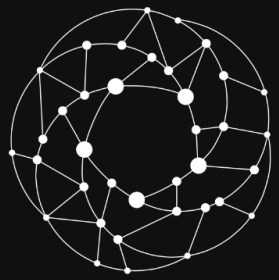
OPENMANO
OTHER DETAILS

- openmanod.cfg -> configuration file
  - IP address and port where the HTTP server is listening to API requests
  - DB parameters
  - VNF repo folder where copies of the VNFD will be stored

- openmano client, Python-based

```
$ openmano -h
usage: openmano [-h] [--version]
                {config,vnf-create,vnf-list,vnf-delete,scenario-create,scenario-
list,scenario-delete,scenario-deploy,scenario-verify,instance-scenario-list,instance-
scenario-delete,tenant-create,tenant-delete,tenant-list,tenant-edit,datacenter-
edit,datacenter-create,datacenter-delete,datacenter-list,datacenter-attach,datacenter-
detach,datacenter-net-edit,datacenter-net-update,datacenter-net-delete,datacenter-net-
list}
                ...

User program to interact with OPENMANO-SERVER (openmanod)
```



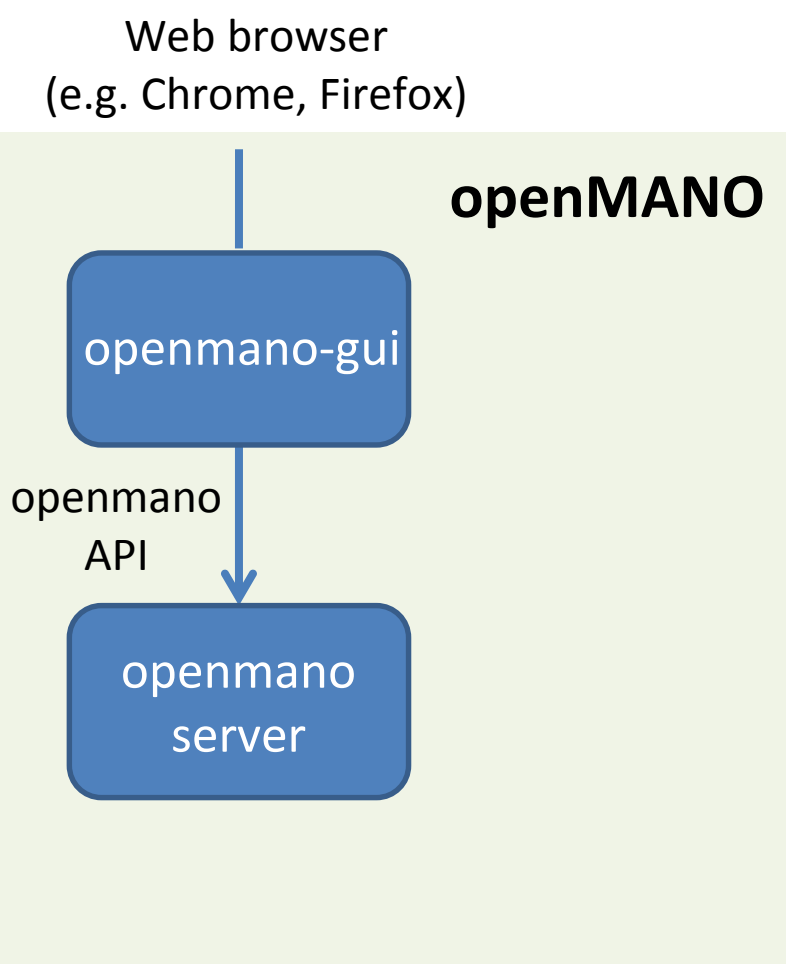
Open Source
MANO

OPENMANO COMPONENTS

Openmano-gui: web-based interface
(not part of OSM)

ETSI

OPENMANO-GUI: THE WEB-BASED INTERFACE RELATION TO OPENMANO



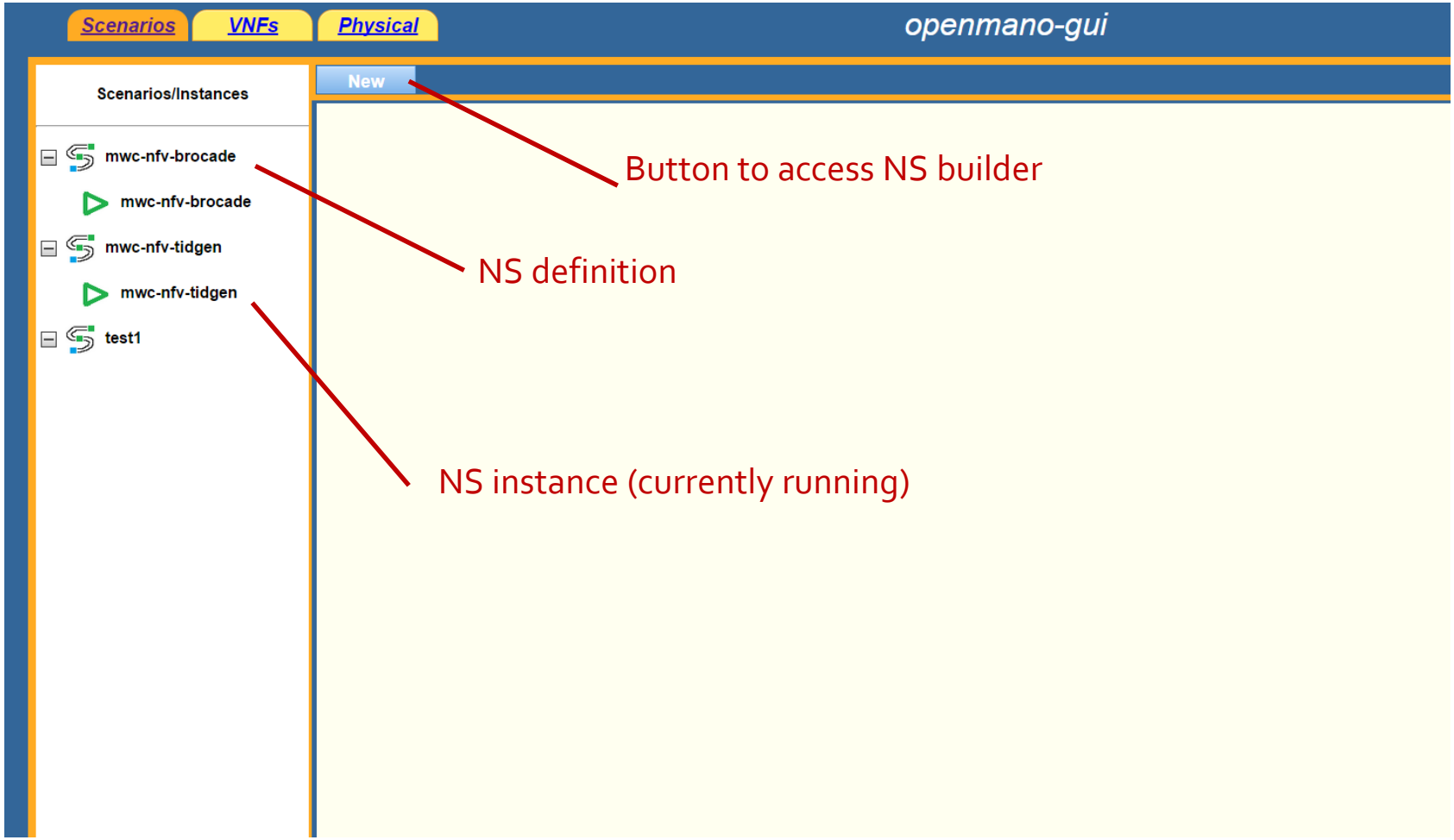
CHARACTERISTICS

- Access to network scenario definitions and instances
- Drag&drop scenario builder with access to the VNF catalogue
- Actions over network scenario instances (stop, shutdown, delete, deploy) and over specific VNF instances inside an NS instance

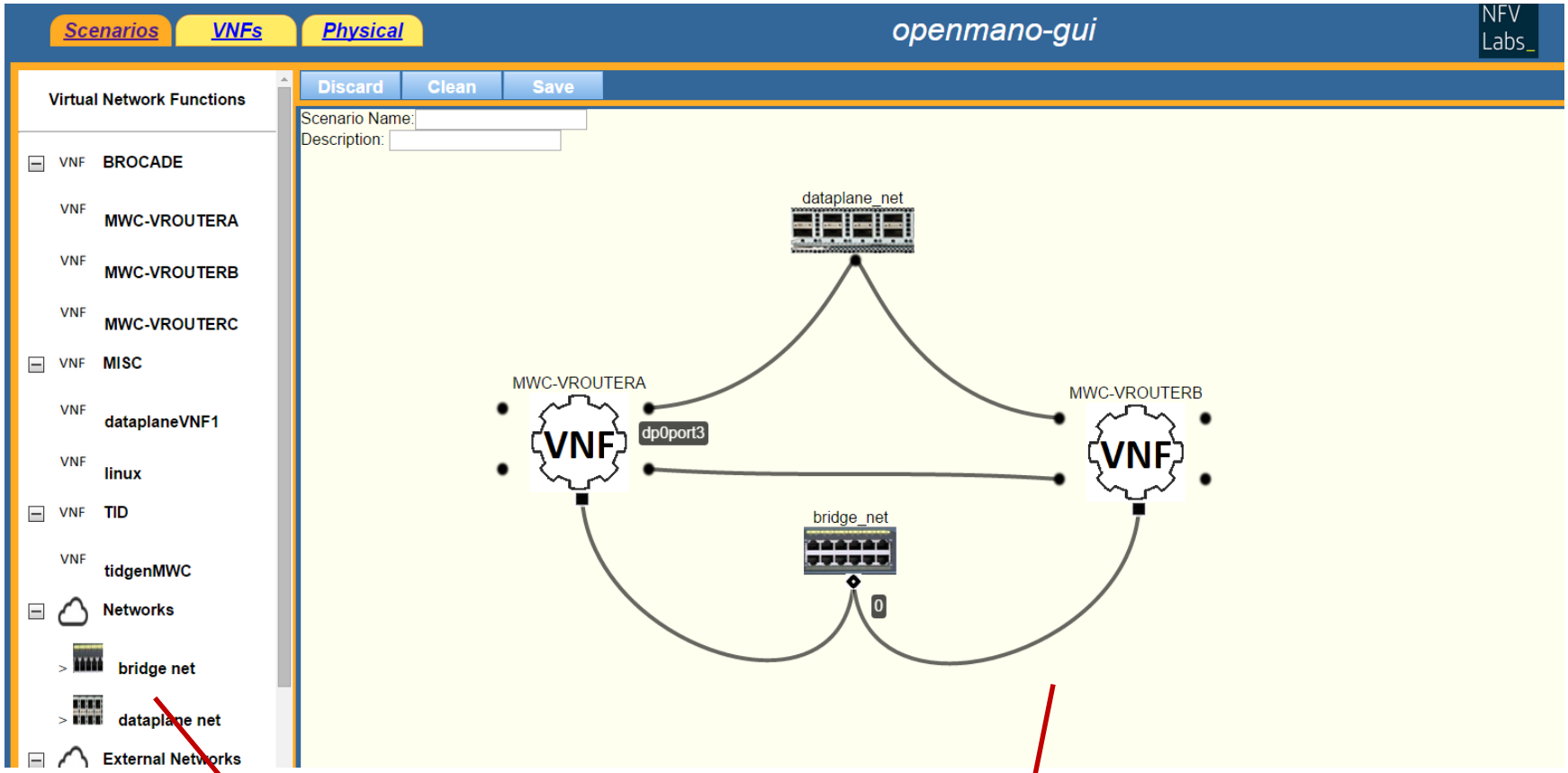
CONFIGURATION

- **config.php**: configuration file

OPENMANO-GUI: THE WEB-BASED INTERFACE (II)
NS DEFINITIONS AND INSTANCES



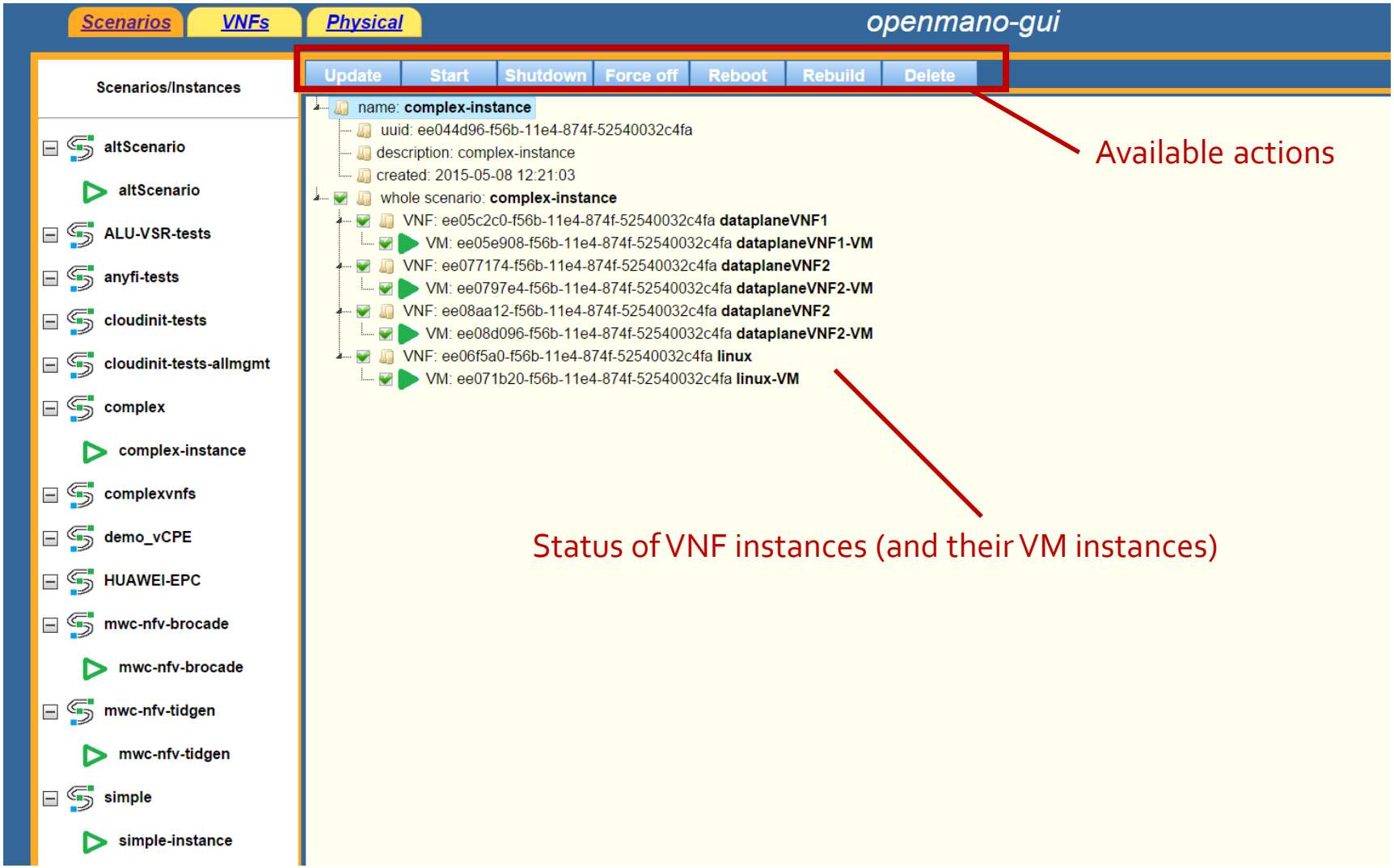
OPENMANO-GUI: THE WEB-BASED INTERFACE (III)
NS BUILDER



Catalogue of VNFs and external/public networks

Drag & drop panel

OPENMANO-GUI: THE WEB-BASED INTERFACE (IV) ACTIONS OVER NS INSTANCES



WANT TO KNOW MORE ABOUT OPENMANO?

All info at:

https://github.com/nfvlabs/openmano

Questions/feedback/suggestions: nfvlabs@tid.es

Open Source MANO

ETSI