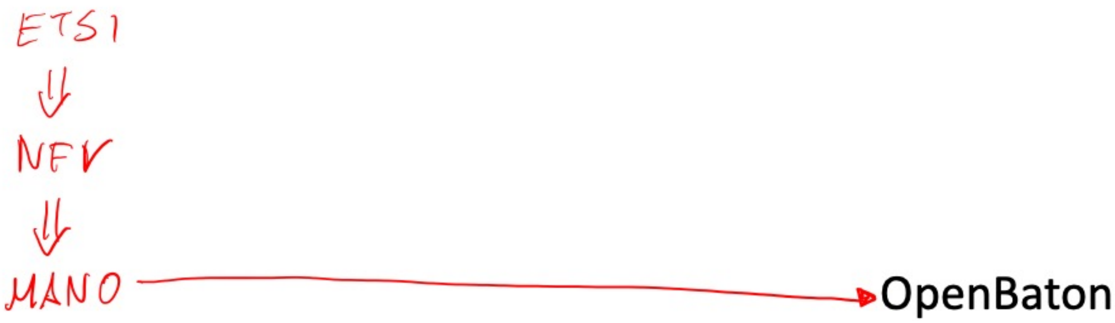
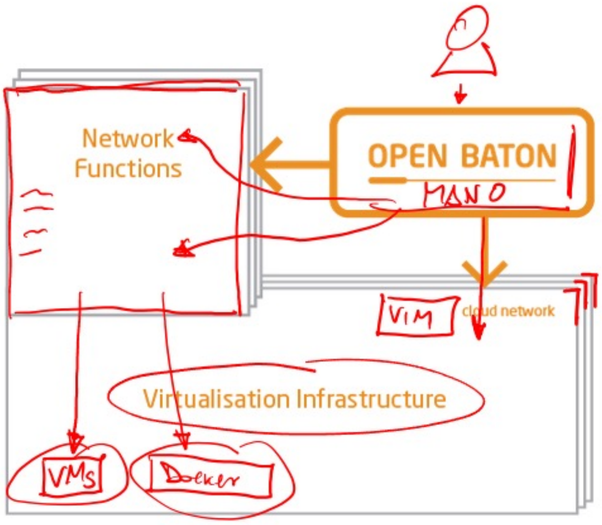
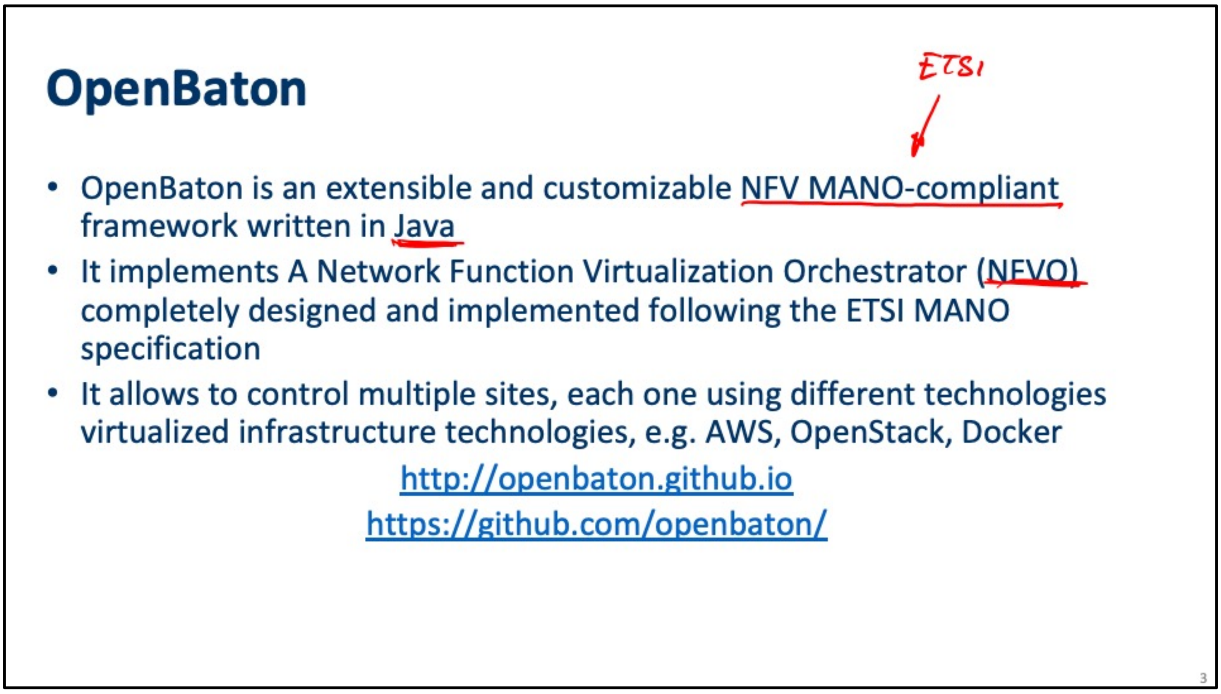


OpenBaton

Antonio Virdis
Assistant Professor@ University of Pisa
antonio.virdis@unipi.it

OPENNESS AND EXTENSIBILITY





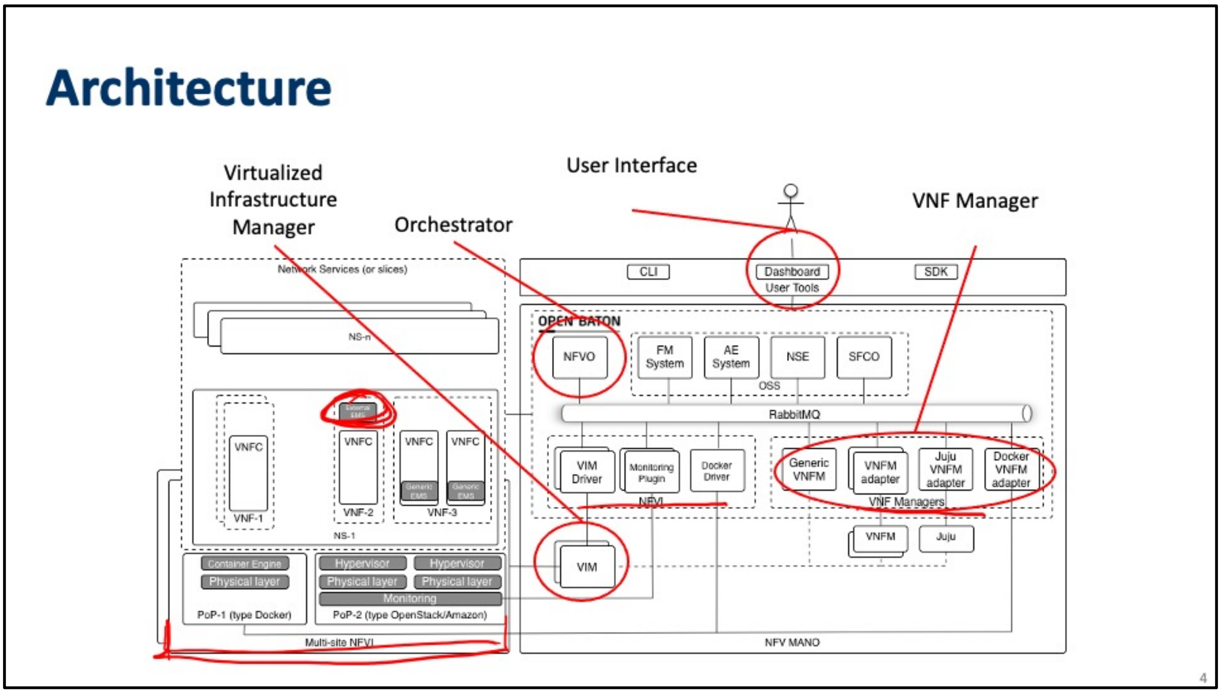
OpenBaton

ETSI

- OpenBaton is an extensible and customizable NFV MANO-compliant framework written in Java
- It implements A Network Function Virtualization Orchestrator (NFVO) completely designed and implemented following the ETSI MANO specification
- It allows to control multiple sites, each one using different technologies virtualized infrastructure technologies, e.g. AWS, OpenStack, Docker

http://openbaton.github.io

https://github.com/openbaton/



Architecture

A generic Virtual Network Function Manager (**VNFM**) and Generic Element Management System (**EMS**) able to manage the lifecycle of VNFs based on their descriptors.

The Generic VNF Manager is an implementation following the ETSI MANO specifications. It works as intermediate component between the NFVO and the VNFs, particularly the Virtual Machines on top of which the VNF software is installed. In order to complete the lifecycle of a VNF, it interoperates with the Open Baton Element Management System (EMS) which acts as an agent inside the VMs and executing scripts contained in a VNF package

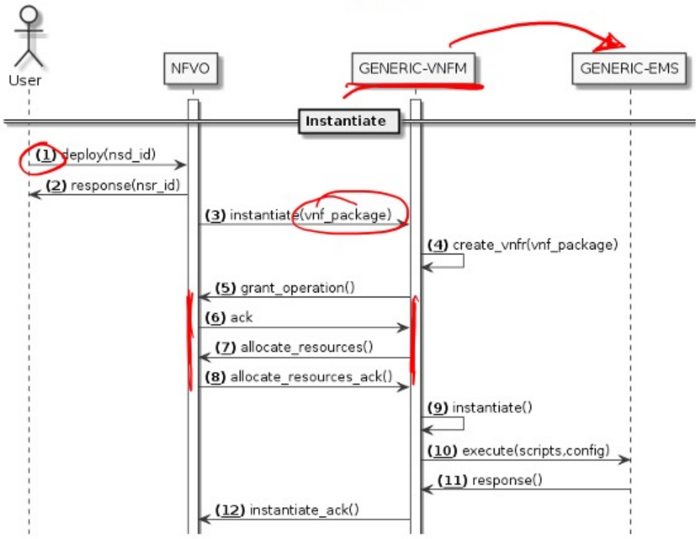
A Docker VNFM and VIM driver for instantiating containers on top of Docker Engine / Docker Swarm

Example of NS deployment

NS
VNF
VNF
VNF

- **GRANT_OPERATION**:
  check if the resources are available on the selected PoP.

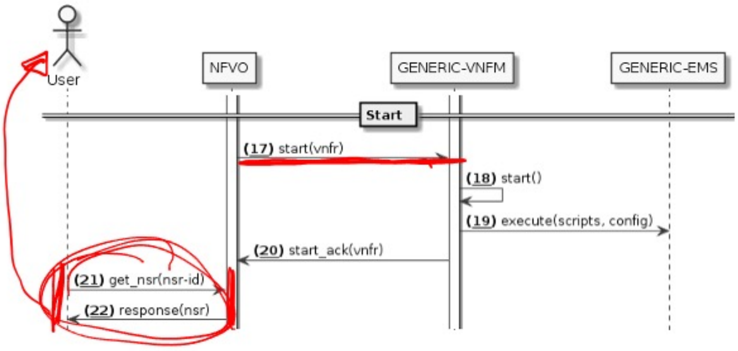
- **ALLOCATE_RESOURCE**:
  This message ask the NFVO to create all the resources

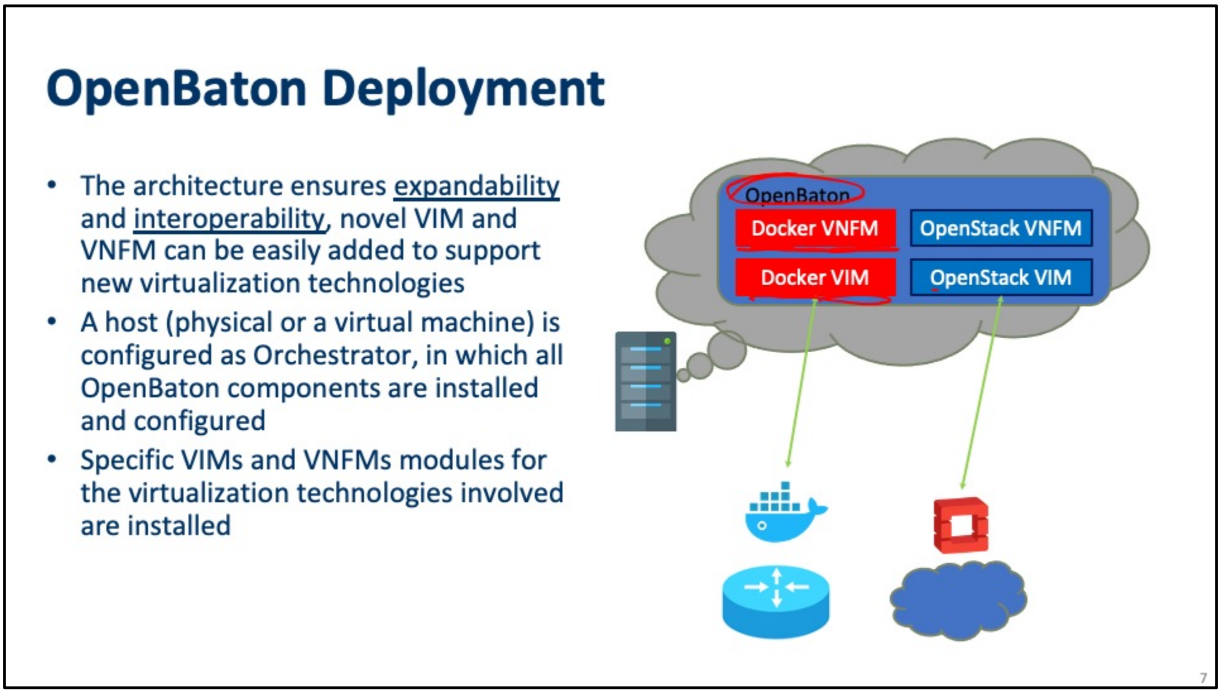


User
NFVO
GENERIC-VNFM
GENERIC-EMS
Instantiate

(1) deploy(nsd_id)
(2) response(nsr_id)
(3) instantiate(vnf_package)
(4) create_vnfr(vnf_package)
(5) grant_operation()
(6) ack
(7) allocate_resources()
(8) allocate_resources_ack()
(9) instantiate()
(10) execute(scripts,config)
(11) response()
(12) instantiate_ack()

Example of NS deployment (2)





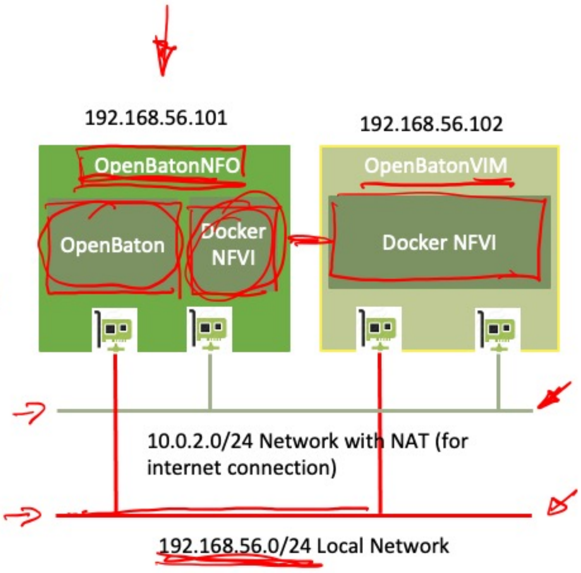
OpenBaton Deployment

- The architecture ensures expandability and interoperability, novel VIM and VNFM can be easily added to support new virtualization technologies
- A host (physical or a virtual machine) is configured as Orchestrator, in which all OpenBaton components are installed and configured
- Specific VIMs and VNFMs modules for the virtualization technologies involved are installed

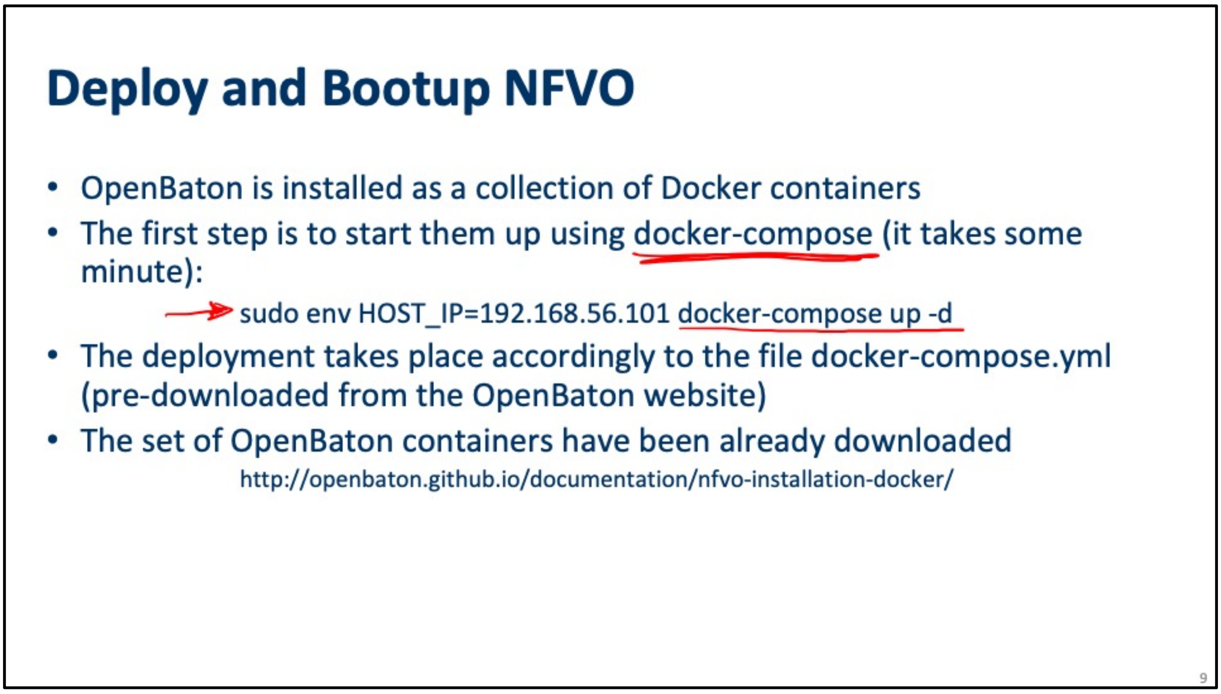
It manages a multi-site NFVI supporting heterogeneous virtualization and cloud technologies.

Our Deployment

- Docker is exploited as NFV Infrastructure
- Two Virtual Machines are provided:
  - OpenBatonNFO with the orchestrator installed and a VIM and a NFVM for Docker installed. The VM has also the docker daemon installed and configured, i.e. it can run containers
  - OpenBatonVIM with docker installed and configured


192.168.56.101
192.168.56.102
OpenBatonNFO
OpenBatonVIM
OpenBaton
Docker NFVI
Docker NFVI
10.0.2.0/24 Network with NAT (for internet connection)
192.168.56.0/24 Local Network

User: osboxes / Password: osboxes.org



Deploy and Bootup NFVO

- OpenBaton is installed as a collection of Docker containers
- The first step is to start them up using docker-compose (it takes some minute):

```
sudo env HOST_IP=192.168.56.101 docker-compose up -d
```

- The deployment takes place accordingly to the file docker-compose.yml (pre-downloaded from the OpenBaton website)
- The set of OpenBaton containers have been already downloaded

http://openbaton.github.io/documentation/nfvo-installation-docker/

```
sudo dpkg-reconfigure keyboard-configuration
```

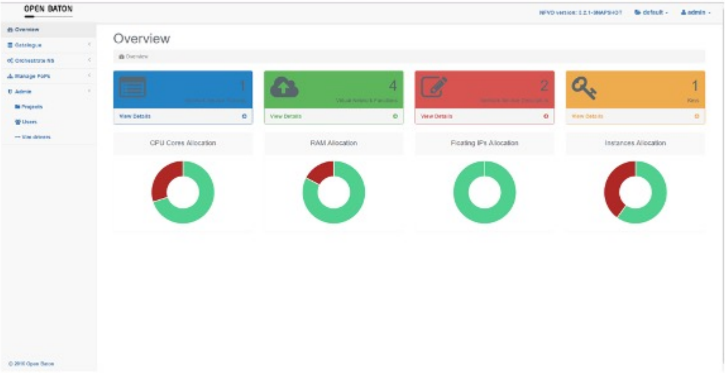
to change keyboard layout

Access the Web Dashboard

- Open a browser and go to:

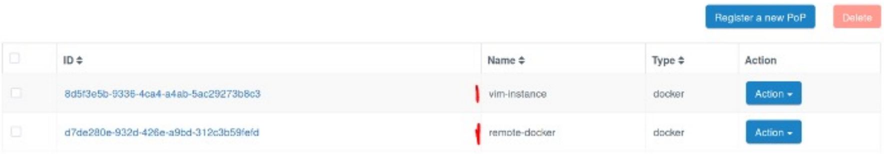
  http://192.168.56.101:8080

- Login, User: admin / Password: openbaton



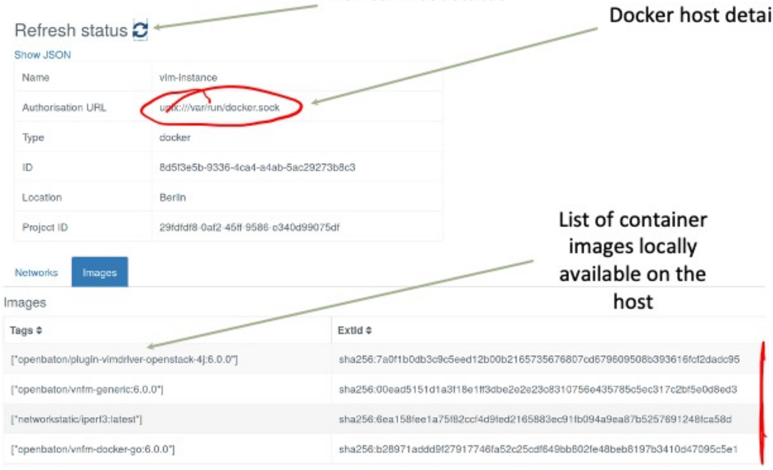
NFVI Integration

- The two hosts has been already integrated into the NFVO as part of the NFV Infrastructure
- New Docker hosts (or using other virtualization technologies) can be integrated via:
  - Manage POPs -> POP Instances -> Register a new POP
- Check the correct integration of the two hosts



□	ID ⇕	Name ⇕	Type ⇕	Action
□	8d5f3e5b-9336-4ca4-a4ab-5ac29273b8c3	vim-instance	docker	Action ▾
□	d7de280e-932d-426e-a9bd-312c3b59fefd	remote-docker	docker	Action ▾

NFVI Integration



Refresh status

Show JSON

Name	vim-instance
Authorisation URL	unix:///var/run/docker.sock
Type	docker
ID	8d5f3e5b-9336-4ca4-a4ab-5ac29273b8c3
Location	Berlin
Project ID	29fdfdf8-0af2-45ff-9586-c340d99075df

Images

Tags	ExtId
["openbaton/plugin-vimdriver-openstack-4j:6.0.0"]	sha256:7a0f1b0db3c9c5eed12b00b2165735676807cd679609508b393616fcf2dadc95
["openbaton/vnfm-generic:6.0.0"]	sha256:00ead5151d1a3f18e1ff3dbe2e2e23c8310756e435785c5ec317c2bf5e0d8ed3
["networkstatic/iperf3:latest"]	sha256:6ea158fee1a75f82ccf4d9fed2165883ec91fb094a9ea87b5257691248fca58d
["openbaton/vnfm-docker-go:6.0.0"]	sha256:b28971addd9f27917746fa52c25cdf649bb802fe48beb8197b3410d47095c5e1

Refresh host status

Docker host details

List of container images locally available on the host

VNF Creation – Create the Container

- Install the container on the host on which the VNF might be deployed (e.g. a container with a telnet server)
  - Create a Dockerfile

```
FROM rohan/ascii-telnet-server

EXPOSE 23
```

  - Build the container image

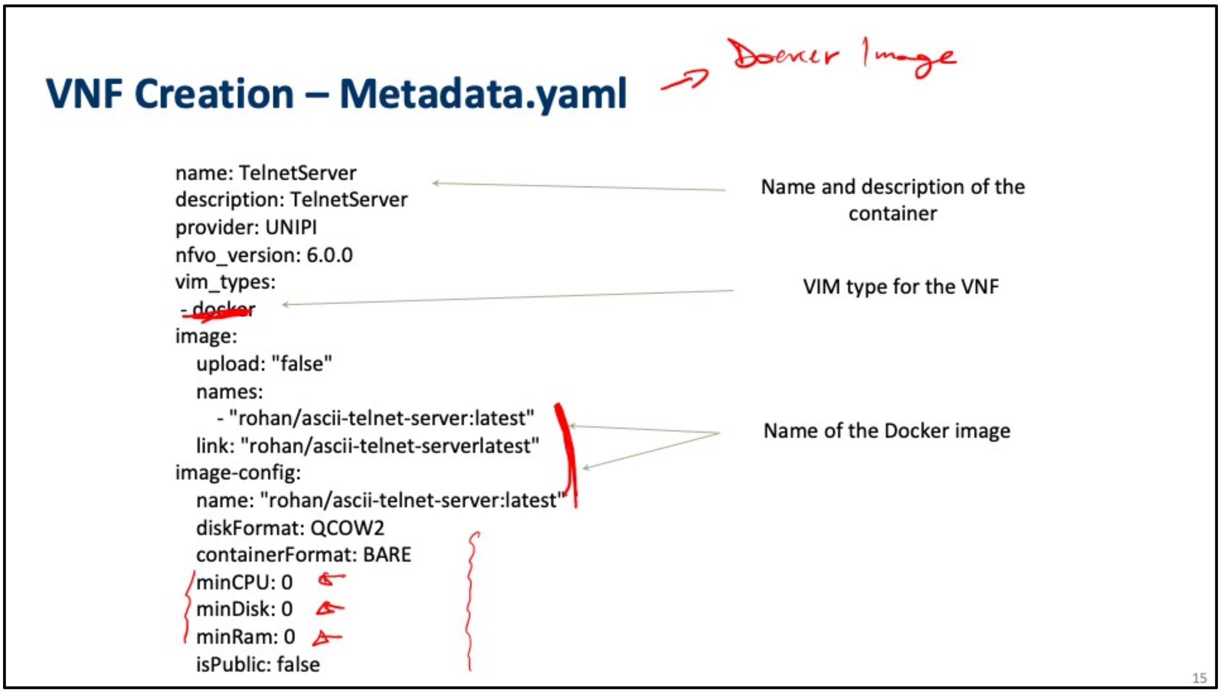
/folder

```
sudo docker build -t telnet_custom .
```

- Check that the container is in the list of images by refreshing the Image List in the POP page from the OpenBaton dashboard

VNF Creation – Setup VNF Package

- A VNF package is a package describing the VNF
- The VNF is described by two files:
  - **Metadata.yaml**, which describes the container that implements the VNF
  - **vnfd.json**, which describes how OpenBaton has to instantiate the container and the VNF
- Both the files have to be included in a *tar package* and uploaded into the system to create the VNF
- To create a tar package on windows look for a specific tool (e.g. http://www.peazip.org/tar-windows.html)



VNF Creation – Metadata.yaml

→ Docker Image

```
name: TelnetServer
description: TelnetServer
provider: UNIPI
nfvo_version: 6.0.0
vim_types:
 - docker
image:
   upload: "false"
   names:
      - "rohan/ascii-telnet-server:latest"
   link: "rohan/ascii-telnet-serverlatest"
image-config:
   name: "rohan/ascii-telnet-server:latest"
   diskFormat: QCOW2
   containerFormat: BARE
   minCPU: 0
   minDisk: 0
   minRam: 0
   isPublic: false
```

Name and description of the container

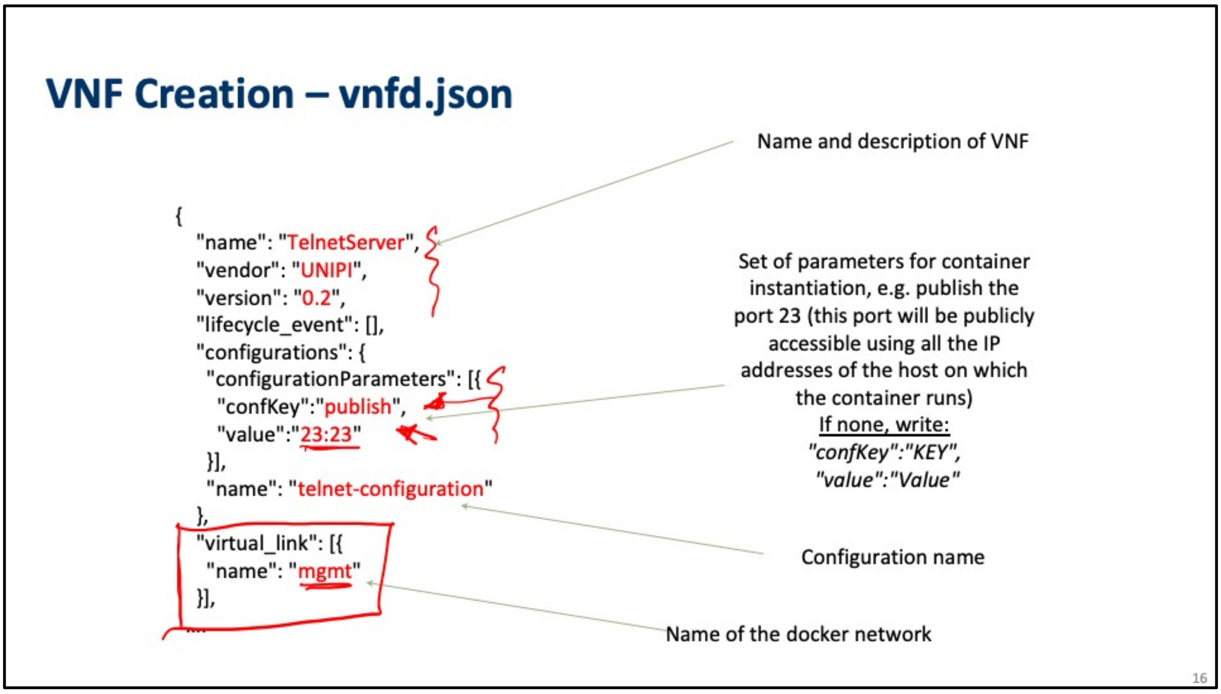
VIM type for the VNF

Name of the Docker image

15

vim_types: must be *docker* (pointing to the Docker VIM Driver)

image upload has to be *false*, please make sure that the images listed in **image names** are present in the VIM instances that you want to use



VNF Creation – vnfd.json

```
{
  "name": "TelnetServer",
  "vendor": "UNIPI",
  "version": "0.2",
  "lifecycle_event": [],
  "configurations": {
   "configurationParameters": [{
    "confKey":"publish",
    "value":"23:23"
   }],
   "name": "telnet-configuration"
  },
  "virtual_link": [{
   "name": "mgmt"
  }],
 ...
```

Name and description of VNF

Set of parameters for container instantiation, e.g. publish the port 23 (this port will be publicly accessible using all the IP addresses of the host on which the container runs)

<u>If none, write:</u>

"confKey":"KEY",
"value":"Value"

Configuration name

Name of the docker network

VNF Creation – vnfd.json

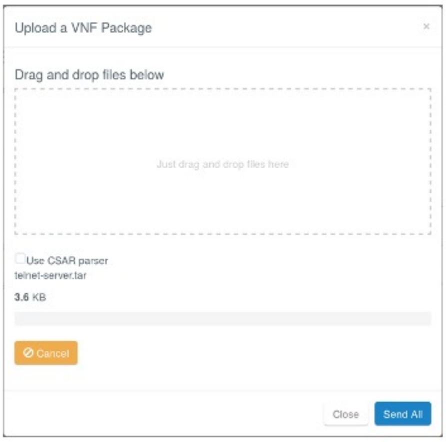
```
...

"vdu": [{
    "vm_image": [
    ],
    "scale_in_out": 2,
    "vnfc": [{
      "connection_point": [{
        "virtual_link_reference": "mgmt"
      }]
    }]
  }],
  "deployment_flavour": [{
    "flavour_key": "m1.small"
  }],
  "type": "telnet",
  "endpoint": "docker"
 }
```

Type of the VIM, docker in this case

VNF Creation – Upload VNF Package

- Create a TAR package with the two files
- Go to the page:
  - Catalogue -> VNF Package -> Upload VNF Package
- Select the package and click on “Send All”
- A new VNF package is created



Upload a VNF Package

Drag and drop files below

Just drag and drop files here

□ Use CSAR parser

telnet-server.tar

3.6 KB

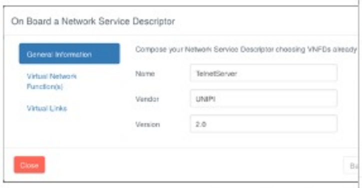
compress using

```
tar -cf telnet-pack.tar Metadata.yaml vnfd.json
```

VNF Creation – Create a Network Service Descriptor

- Before being able to deploy the VNF, a new Network Service (NS) Descriptor has to be created
- The NS is a collection of VNFs (it can be one or more)
- Go to:
  - NS Descriptors -> On Board NSD -> Compose NSD

Add the VNF as part of the NS



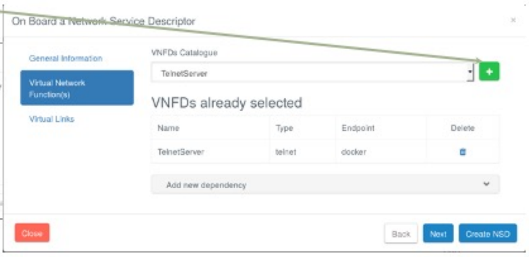
On Board a Network Service Descriptor

Compose your Network Service Descriptor choosing VNFDs already

Name: TelnetServer

Vendor: UNIPI

Version: 2.0



On Board a Network Service Descriptor

VNFDs Catalogue: TelnetServer

VNFDs already selected

Name	Type	Endpoint	Delete
TelnetServer	telnet	docker	🗑

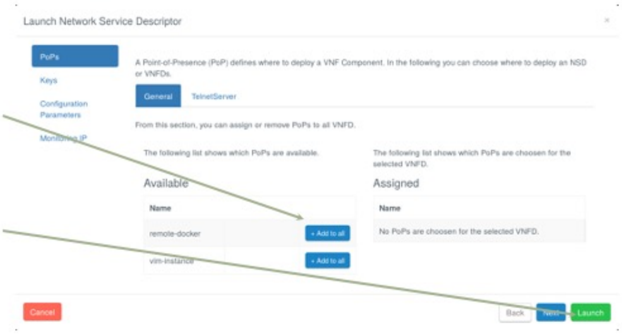
Add new dependency

Lunch the NS with all its VNFs

- Before being able to deploy the VNF, a new Network Service Descriptor has to be created
- Go to:
  - NS Descriptors -> Action (on one NS) -> Launch

Select the hosts on which the NS has to be deployed

Deploy



Launch Network Service Descriptor

A Point-of-Presence (PoP) defines where to deploy a VNF Component. In the following you can choose where to deploy an NSD or VNFDs.

From this section, you can assign or remove PoPs to all VNFD.

The following list shows which PoPs are available.

Available

Name	
remote-docker	+ Add to all
vim-instance	+ Add to all

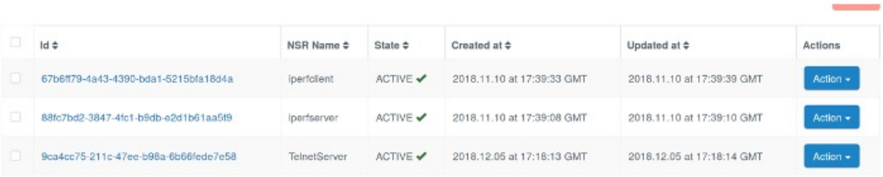
The following list shows which PoPs are choosen for the selected VNFD.

Assigned

Name
No PoPs are choosen for the selected VNFD.

Check NS status

- To retrieve the list of NSs currently deployed go to:
  - Orchestrate NS -> NS Records

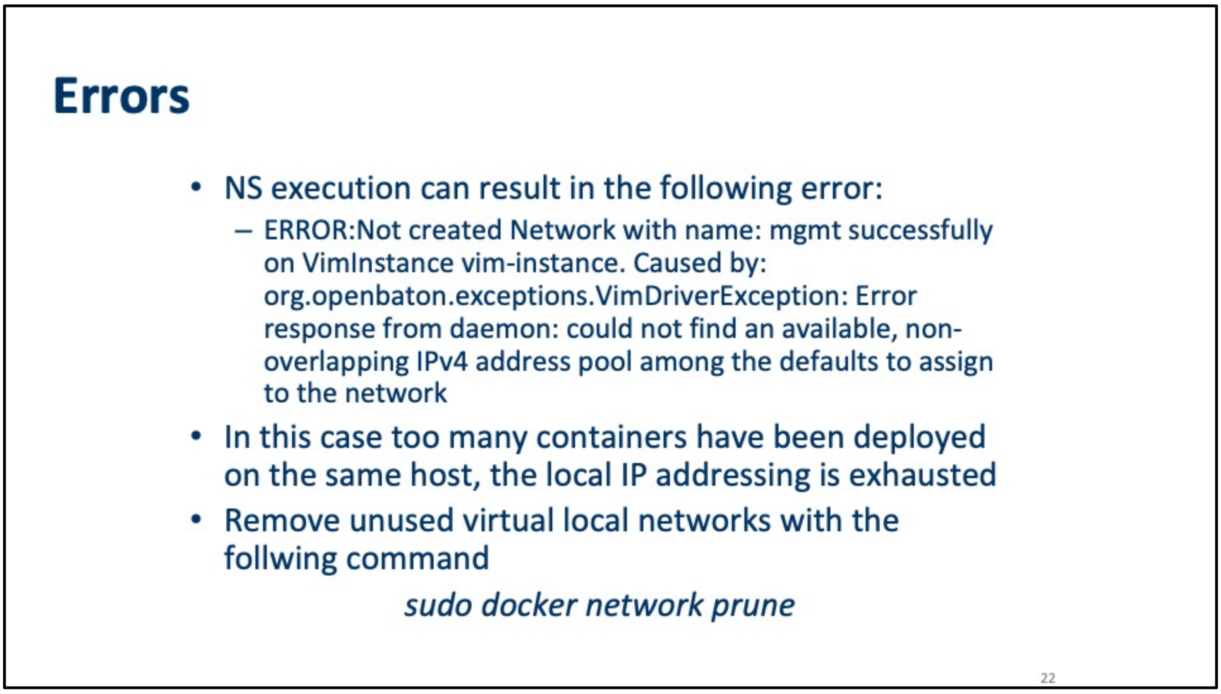


□	Id ⇕	NSR Name ⇕	State ⇕	Created at ⇕	Updated at ⇕	Actions
□	67b6ff79-4a43-4390-bda1-5215bfa18d4a	iperfclient	ACTIVE ✓	2018.11.10 at 17:39:33 GMT	2018.11.10 at 17:39:39 GMT	Action ▾
□	88fc7bd2-3847-4fc1-b9db-e2d1b61aa5f9	iperfserver	ACTIVE ✓	2018.11.10 at 17:39:08 GMT	2018.11.10 at 17:39:10 GMT	Action ▾
□	9ca4cc75-211c-47ee-b98a-6b66fede7e58	TelnetServer	ACTIVE ✓	2018.12.05 at 17:18:13 GMT	2018.12.05 at 17:18:14 GMT	Action ▾

If you connect to one of the two hosts you can check that the telnet server is actually running :

sudo docker ps

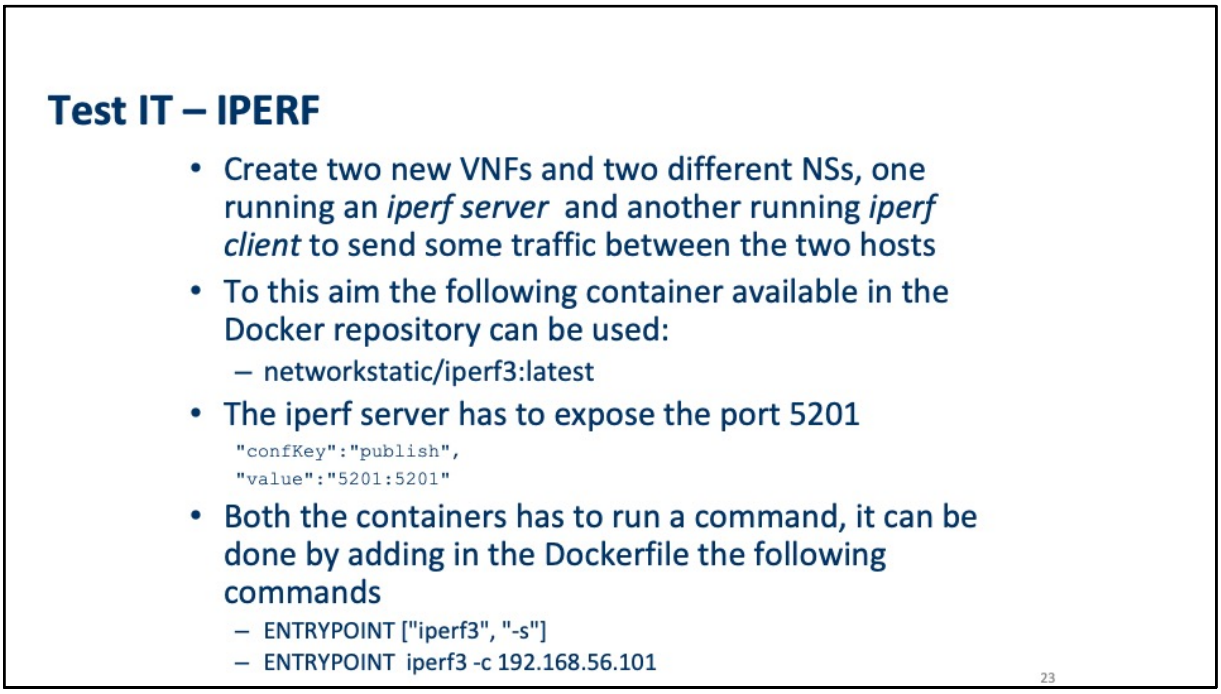
telnet localhost 23



Errors

- NS execution can result in the following error:
  - ERROR:Not created Network with name: mgmt successfully on VimInstance vim-instance. Caused by: org.openbaton.exceptions.VimDriverException: Error response from daemon: could not find an available, non-overlapping IPv4 address pool among the defaults to assign to the network
- In this case too many containers have been deployed on the same host, the local IP addressing is exhausted
- Remove unused virtual local networks with the follwing command

```
sudo docker network prune
```



Test IT – IPERF

- Create two new VNFs and two different NSs, one running an *iperf server* and another running *iperf client* to send some traffic between the two hosts
- To this aim the following container available in the Docker repository can be used:
  - networkstatic/iperf3:latest
- The iperf server has to expose the port 5201

```
"confKey":"publish",
"value":"5201:5201"
```

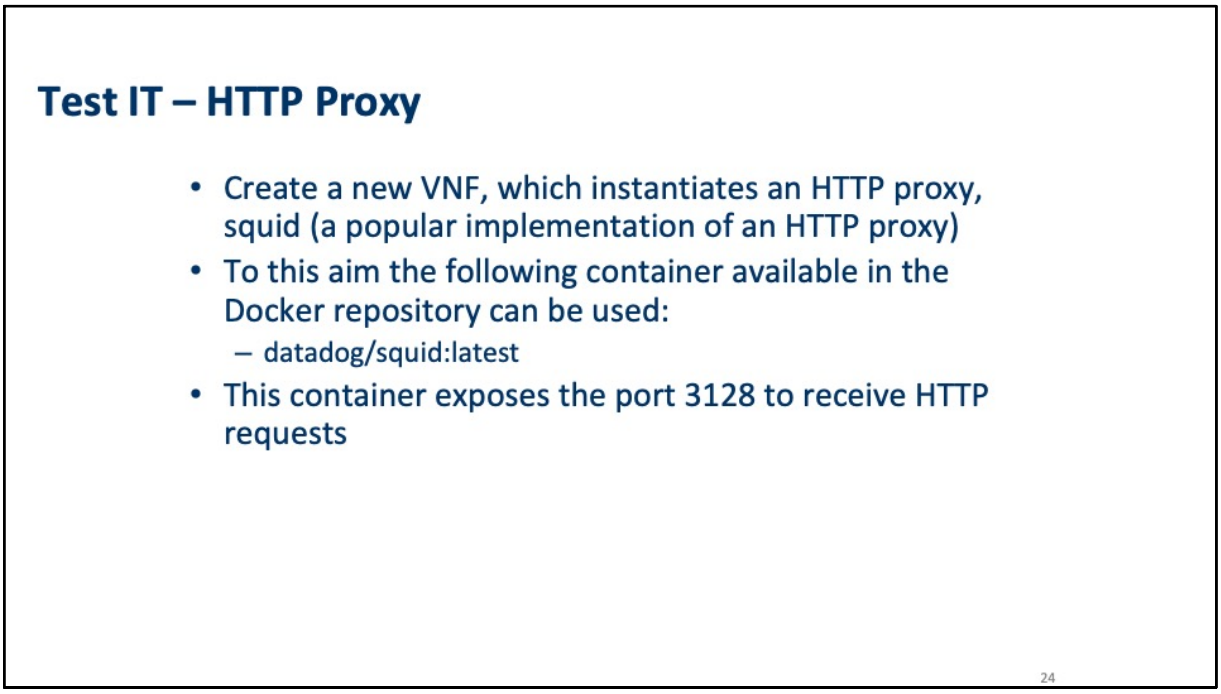
- Both the containers has to run a command, it can be done by adding in the Dockerfile the following commands
  - ENTRYPOINT ["iperf3", "-s"]
  - ENTRYPOINT iperf3 -c 192.168.56.101

“vim-instance” is on the orchestrator (NFO)

“remote-docker” is on the other one (VIM)

to test:

- ifconfig enp0s8
- download and use bmon



Test IT – HTTP Proxy

- Create a new VNF, which instantiates an HTTP proxy, squid (a popular implementation of an HTTP proxy)
- To this aim the following container available in the Docker repository can be used:
  - datadog/squid:latest
- This container exposes the port 3128 to receive HTTP requests