LAB3-NFV Open Source MANO

Handout: March 29, 2018

Deadline: April 8 23:00, 2018 (No extension)

Assignment overview

In this assignment, you are required to construct a management and orchestration (MANO) stack for NFV and deploy a simple network service on this platform. After finishing these basic operations, you are encouraged to design and deploy some more complex network services as well as trying to exploit other interesting features of this platform. It may be a little challengeable, but there are enough documents to reference.

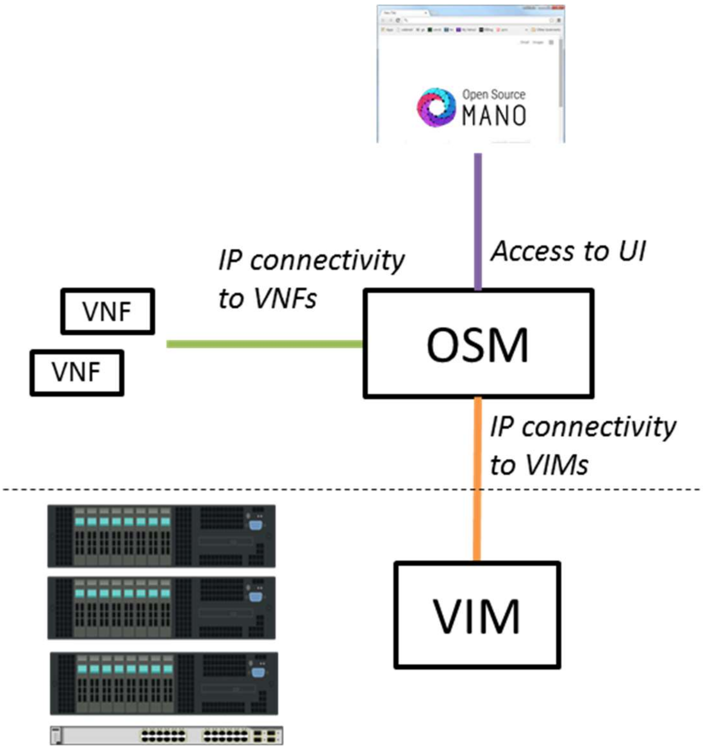
MANO&OSM

As you may have already known, NFV, which is getting more and more popular, consists of three main components: VNF, MANO, NFVI. Among these three components, MANO is the bridge that connect the bottom-layer infrastructure and the top-layer VNF, it plays a critical role in the NFV framework, that's why we design this lab. Recently there have arisen several MANOs developed by either related companies or some open source organizations. And for this lab, we choose OSM as our MANO because it is open-source and stable.

Open Source MANO (OSM) is the open source community that aims to deliver a production-quality MANO stack for NFV, capable of consuming openly published information models, available to everyone, suitable for all VNFs, operationally significant and VIM-independent. OSM is aligned to NFV ISG information models while providing first-hand feedback based on its implementation experience. For detail information about OSM, you can refer OSM Wiki.

Installation Overview

The following figure shows how OSM interacts with VIM and VNFs after the basic operations:



There are three steps to finish the basic operations:

1. Install OSM
2. Add a VIM account
3. Deploy the simple network service

These three steps will be explained in detail in next section.

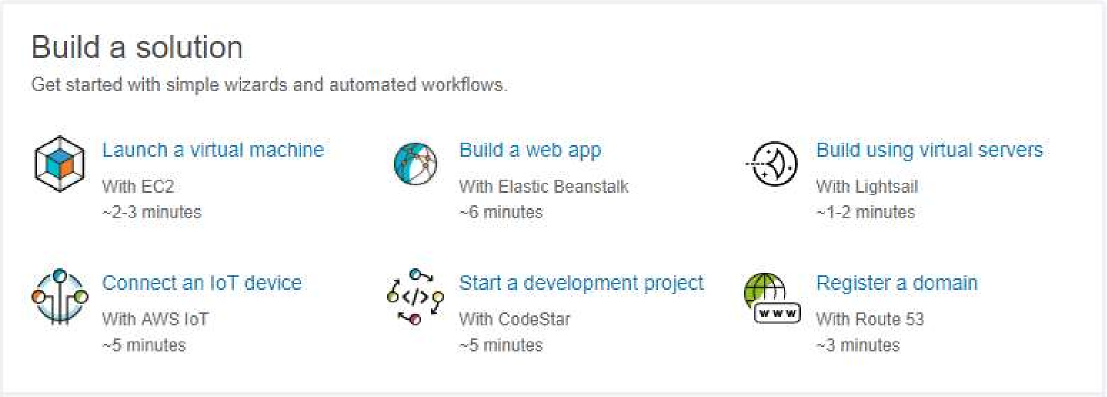
From scratch

Before your work, you are supposed to have an instance running in some cloud server, which can access websites prohibited by GFW without proxy. It is recommended for you to use Amazon EC2 or Google Cloud Server due to their free trials.

Take Amazon EC2 for example.

Firstly a Shadowsocks client is needed on your computer to visit Amazon Web Services (Remember to use global proxy). Register a new account and add the required information, including your debit/visa card information. Then you have 750 free hours to use Amazon EC2.

Sign In to the console, click "Launch a virtual machine", and select "Ubuntu Server 16.04 LTS (HVM), SSD Volume Type - ami-916f59f4".



Build a solution

Get started with simple wizards and automated workflows.

Launch a virtual machine
With EC2
~2-3 minutes

Build a web app
With Elastic Beanstalk
~6 minutes

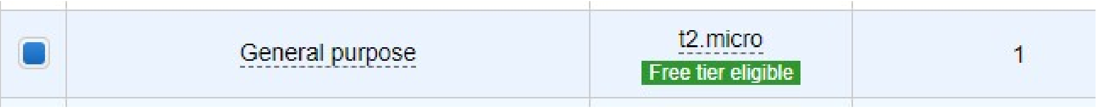
Build using virtual servers
With Lightsail
~1-2 minutes

Connect an IoT device
With AWS IoT
~5 minutes

Start a development project
With CodeStar
~5 minutes

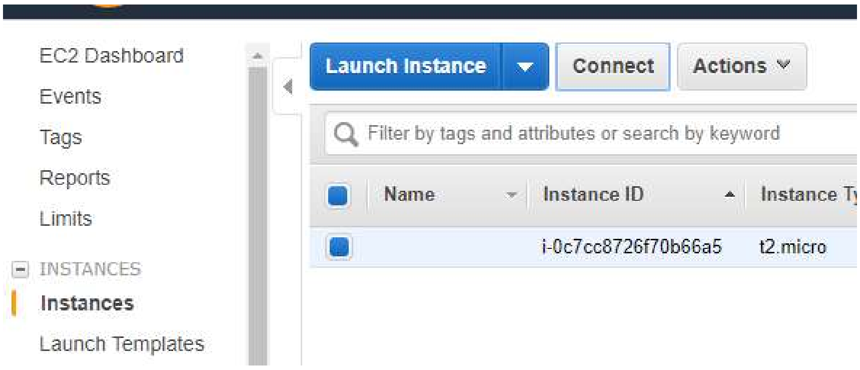
Register a domain
With Route 53
~3 minutes

Select the default and click "Review and Launch". Eventually, launch it. **Note that Amazon might ask you to generate a key pair for secure connection and just do what it indicates. Then you get a .pem file.**

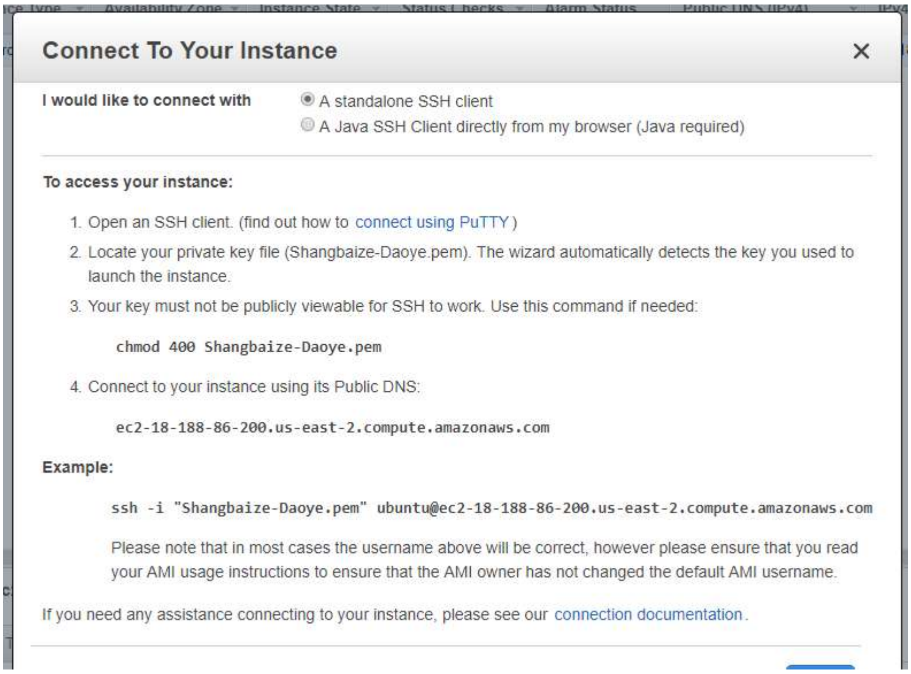


■	General purpose	t2.micro Free tier eligible	1

Now an instance running Ubuntu 16.04 was created and you could press the "Connect" button to look up guides for connecting. You'd better to choose the option of using a standalone SSH client, and do what it says.



i-0c7cc8726f70b66a5 t2.micro



Connect To Your Instance

I would like to connect with
● A standalone SSH client
○ A Java SSH Client directly from my browser (Java required)

To access your instance:

1. Open an SSH client. (find out how to connect using PuTTY)
2. Locate your private key file (Shangbaize-Daoye.pem). The wizard automatically detects the key you used to launch the instance.
3. Your key must not be publicly viewable for SSH to work. Use this command if needed:

```
chmod 400 Shangbaize-Daoye.pem
```

4. Connect to your instance using its Public DNS:

```
ec2-18-188-86-200.us-east-2.compute.amazonaws.com
```

Example:

```
ssh -i "Shangbaize-Daoye.pem" ubuntu@ec2-18-188-86-200.us-east-2.compute.amazonaws.com
```

Please note that in most cases the username above will be correct, however please ensure that you read your AMI usage instructions to ensure that the AMI owner has not changed the default AMI username.

If you need any assistance connecting to your instance, please see our connection documentation.

If connection can be successfully established, congratulations to you! You’ve achieved the target and will be able to go to the next.

Prerequisite

All you need to run OSM Release THREE is a single server or VM with the following requirements:

- MINIMUM: 4 CPUs, 8 GB RAM, 40GB disk and a single interface with Internet access
- RECOMMENDED: 8 CPUs, 16 GB RAM, 80GB disk and a single interface with Internet access
- Ubuntu16.04 (64-bit variant required) as base image, configured to run LXD containers. If you don't have LXD configured, you can follow the instructions below

Note: If you wish to install OSM Release THREE from inside a LXD container, you will need to enable nested containers following instructions here (For advanced users: Nested containers).

Configure LXD

LXD is a pure container hypervisor that runs unmodified Linux guest operating systems with VM-style operations at incredible speed and density. This makes it particularly well-suited for developing complex systems. This can be used to install OSM without tainting your host system with its dependencies. OSM modules will be running in LXD containers, thus not affecting your host system.

Install the LXD package:

```
sudo apt-get update
sudo apt-get install -y lxd
newgrp lxd   # required to log the user in the lxd group if lxd was just installed
```

Configure LXD with a bridge for networking:

```
sudo lxd init
Name of the storage backend to use (dir or zfs) [default=dir]:
Would you like LXD to be available over the network (yes/no) [default=no]?
Do you want to configure the LXD bridge (yes/no) [default=yes]?
  Do you want to setup an IPv4 subnet? Yes
     Default values apply for next questions
  Do you want to setup an IPv6 subnet? No
LXD has been successfully configured.
```

Network Bridge

By default, LXD creates a bridge named lxdbr0.Although further customization is possible, default options for LXD bridge configuration will work.

MTU

Check the MTU of the LXD bridge (lxdbr0) and the MTU of the default interface. If they are different, change the default MTU of the containers. This might be required, for instance, when running OSM in a VM on some OpenStack distributions.

Note: In this example, we will assume that the default interface is ens3 and its MTU is 1446

```
lxc list                                    # This will drive initialization of
lxdbr0
ip address show ens3                       # In case ens3 is the default
interface
ip address show lxdbr0
sudo lxc profile device set default eth0 mtu 1446  # Use the appropriate MTU value
```

Testing LXD

To test that your LXD installation is correct, try to deploy a container and run 'apt-get update' from inside:

```
lxc launch ubuntu:16.04 test  #Create a container based on Ubuntu with name 'test'
lxc exec test bash            # Access the container
root@test:~# apt-get update  # Run command 'apt-get update' from inside the
container
root@test:~# exit             # Exit from the container
lxc stop test                 # Stop the container
lxc delete test               # Delete the container
```

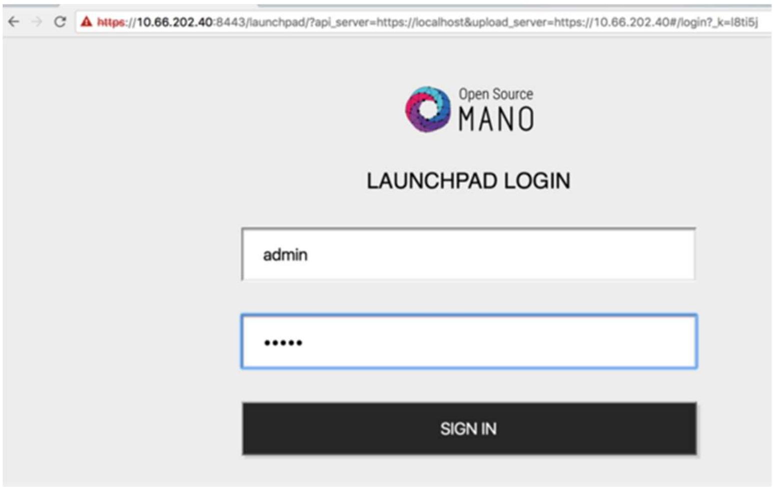
Once you have prepared the host with the previous requirements, all you need to do is:

```
wget https://osm-download.etsi.org/ftp/osm-3.0-three/install_osm.sh
chmod +x install_osm.sh
./install_osm.sh
```

Checking your installation

After some time, you will get a fresh OSM Release THREE installation. You can access to the UI in the following URL (user: admin, password: admin):

You can connect to the service via a web browser (Google Chrome version 50 or later is recommended). Open a browser and connect to https://1.2.3.4:8443 , replacing 1.2.3.4 with the IP address of your host.(Note that it uses https, not http. Google Chrome is recommended.)



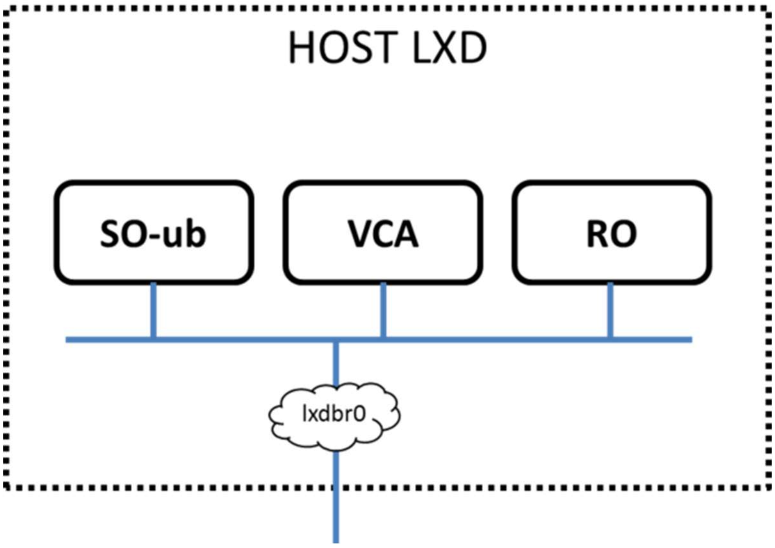
https://10.66.202.40:8443/launchpad/?api_server=https://localhost&upload_server=https://10.66.202.40#/login?_k=l8ti5j

admin

•••••

Make sure that port 8443 is accessible, as well as the following required ports: 8000, 4567, 8008, 80, 9090.

As a result of the installation, three LXD containers are created in the host: RO, VCA, and SO-ub (running the SO and the UI), as shown in the figure below.



Install OSM client

Note: From v3.0.2, the OSM installer installs OSM client by default in your system, so you don't

need to install it. Just type "osm" to see if it is installed in your system.
OSM client is a python-based API and CLI for OSM. To install it:

```
curl http://osm-
download.etsi.org/repository/osm/debian/ReleaseTHREE/OSM%20ETSI%20Release%20Key.gpg
| sudo apt-key add -
sudo add-apt-repository -y "deb [arch=amd64] http://osm-
download.etsi.org/repository/osm/debian/ReleaseTHREE stable osmclient"
sudo apt-get update
sudo apt-get install -y python-osmclient
```

Once installed, type “osm” to see a list of commands. At minimum, you will need to specify the OSM host, either via an environment variable or via the osm command line (see “osm --help” for info):

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

Note: The OSM client can run on a different computer than the one actually running OSM. All you need to configure is the variables OSM_HOSTNAME and OSM_RO_HOSTNAME.
Via the OSM client, you can do the following:

```
config-agent-add
 config-agent-delete
 config-agent-list
 ns-create
 ns-delete
 ns-list
 ns-monitoring-show
 ns-scale
 ns-scaling-show
 ns-show
 nsd-delete
 nsd-list
 ro-dump
 upload-package
 vcs-list
 vim-create
 vim-delete
 vim-list
 vim-show
 vnf-list
 vnf-monitoring-show
 vnf-show
 vnfd-delete
 vnfd-list
```

Adding VIM accounts

Before proceeding, make sure that you have a site with a VIM configured to run with OSM. Different kinds of VIMs are currently supported by OSM:

- OpenVIM. Check the following link to know how to install and use openvim for OSM: OpenVIM installation (Release THREE). OpenVIM must run in 'normal' mode (not test or fake) to have real virtual machines reachable from OSM.
- OpenStack. Check the following link to learn how to configure OpenStack to be used by OSM: OpenStack configuration (Release THREE)
- VMware vCloud Director. Check the following link to learn how to configure VMware VCD to be used by OSM: Configuring VMware vCloud Director for OSM Release THREE
- Amazon Web Services (AWS). Check the following link to learn how to configure AWS (EC2 and Virtual Private Cloud) to be used by OSM: Configuring AWS for OSM Release THREE

OSM can manage external SDN controllers to perform the dataplane underlay network connectivity on behalve of the VIM. See EPA and SDN assist

Here we recommend you to use OpenVIM as your VIM.

OpenVIM site

Execute the following command, using the appropriate parameters (e.g. site name: "openvim-site", IP address: 10.10.10.10, VIM tenant: "osm")

```
osm vim-create --name openvim-site --auth_url http://10.10.10.10:9080/openvim --
account_type openvim --description "Openvim site" --tenant osm --user dummy --
password dummy
```

OpenStack site

Execute the following command, using the appropriate parameters (e.g. site name: "openvim-site", IP address: 10.10.10.10, VIM tenant: "osm")

```
osm vim-create --name openstack-site --user admin --password userpwd --auth_url
http://10.10.10.11:5000/v2.0 --tenant admin --account_type openstack
```

VMware vCloud Director site

Execute the following command, using the appropriate parameters (e.g. site name: "vmware-site", IP address: 10.10.10.12, VIM tenant: "vmware-tenant", user: "osm", password: "osm4u", admin user: "admin", admin password: "adminpwd", organization: "orgVDC")

```
openmano tenant-create Org1-VDC
export OPENMANO_TENANT=Org1-VDC
openmano datacenter-create --type vmware --config '{"admin_password":
"Hive@VMware1!","admin_username":"etsi", "orgname": "Org1", "nsx_user": "etsi",
"nsx_password": "Hive@VMware1!", "nsx_manager":"https://172.21.6.14",
"vcenter_ip":"172.21.6.13",
"vcenter_port":"443","vcenter_user":"etsi@vsphere.local",
"vcenter_password":"Hive@VMware1!"}' Org1 https://172.21.6.26
openmano datacenter-attach Org1 --vim-tenant-name=Org1:Org1-VDC --
user=org1administrator --password='Hive@VMware1!'
```

```
export OPENMANO_DATACENTER=Org1
```

VMware Integrated Openstack (VIO) site

Execute the following command, using the appropriate parameters (e.g. site name: "openstack-site-vio4", IP address: 10.10.10.12, VIM tenant: "admin", user: "admin", password: "passwd")

```
openmano tenant-create admin
export OPENMANO_TENANT=admin
openmano datacenter-create VIO http://172.21.6.31:5000/v3 --type openstack --
description "OpenStack site v3" --config='{insecure: true, vim_type: VIO}'
openmano datacenter-attach VIO --user=etsi --password='Hive@VMware1!' --vim-
tenant-name=admin --config='{APIversion: v3.3,"dataplane_physical_net":dvs-
341,"use_internal_endpoint":true,"dataplane_net_vlan_range":["1-5","7-10"]}'
export OPENMANO_DATACENTER=VIO
```

With respect to Openstack, the additional configuration for VIO is the following:

- **vim_type**: Set to "VIO" to use VMware Integrated openstack as VIM
- **use_internal_endpoint**: When true it allows use of private API endpoints
- **dataplane_physical_net**: The configured network_vlan_ranges at neutron for the SRIOV (binding direct) and passthrough (binding direct-physical) networks, e.g. 'physnet_sriov' in the above configuration. In case of VMware Integrated Openstack (VIO) provide moref ID of distributed virtual switch, e.g 'dvs-46' in above configuration.
- **dataplane_net_vlan_range**: In case of VMware Integrated Openstack (VIO) provide vlan ranges for the SRIOV (binding direct) networks in format ['start_ID - end_ID']

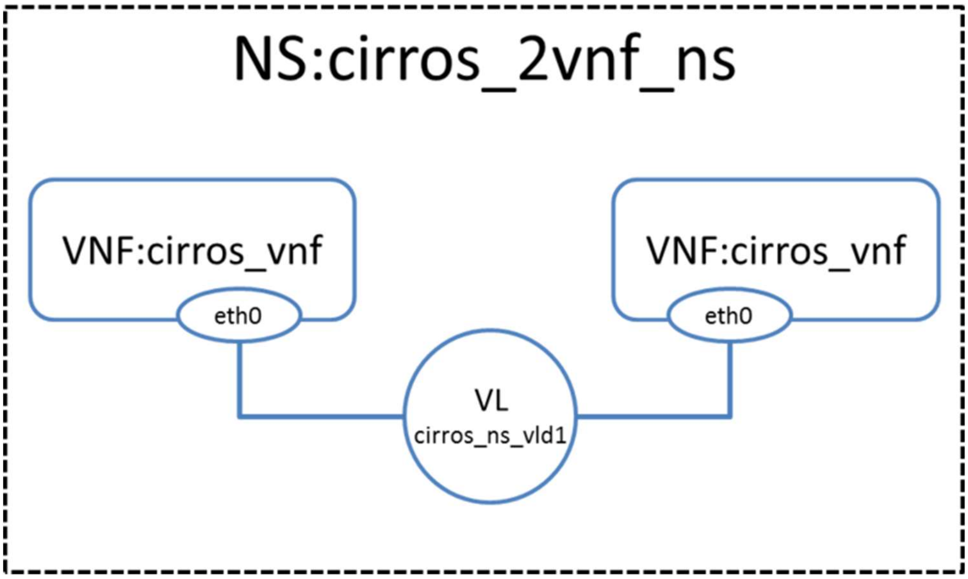
Amazon Web Services (AWS) site

Execute the following command, using the appropriate parameters (e.g. site name: "aws-site", VIM tenant: "admin", user: "XXX", password: "YYY")

```
osm vim-create --name aws-site --user XXX --password YYY --auth_url
https://aws.amazon.com --tenant admin --account_type aws --config '{region_name:
eu-central-1, flavor_info: {t2.nano: {cpus: 1, disk: 100, ram: 512}, t2.micro:
{cpus: 1, disk: 100, ram: 1024}, t2.small: {cpus: 1, disk: 100, ram: 2048},
m1.small: {cpus: 1, disk: 160, ram: 1741}}}'
```

Deploying your first Network Service

In this example we will deploy the following Network Service, consisting of two simple VNFs based on CirrOS connected by a simple VLD.



Before going on, download the required VNF and NS packages from this URL.

Uploading VNF image to the VIM

Get the cirros 0.3.4 image from the following link.Then, onboard the image into the VIM. The instruction differs from one VIM to another

In Openstack:

```
openstack image create --file="./cirros-0.3.4-x86_64-disk.img" --container-format=bare --disk-format=qcow2 cirros034
```

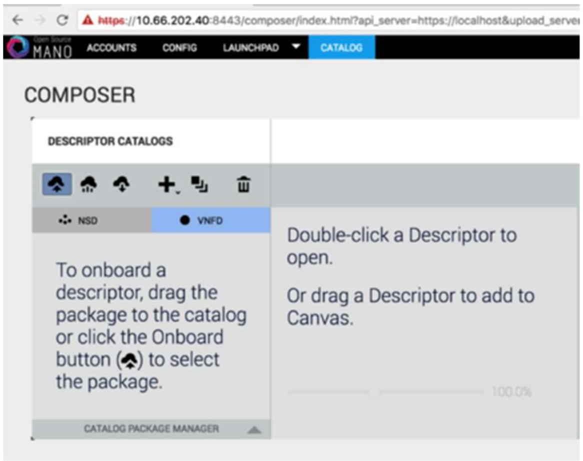
In OpenVIM:

```
#copy your image to the NFS shared folder (e.g. /mnt/openvim-nfs)
cp ./cirros-0.3.4-x86_64-disk.img /mnt/openvim-nfs/
openvim image-create --name cirros034 --path /mnt/openvim-nfs/cirros-0.3.4-x86_64-disk.img
```

Onboarding a VNF

From the UI:

- Go to Catalog
- Click on the import button, then VNFD
- Drag and drop the VNF package file cirros_vnf.tar.gz in the importing area



To onboard a descriptor, drag the package to the catalog or click the Onboard button (⬆) to select the package.

Double-click a Descriptor to open.

Or drag a Descriptor to add to Canvas.

From OSM client:

```
osm upload-package cirros_vnf.tar.gz
osm vnfd-list
```

Onboarding a NS

From the UI:

- Go to Catalog
- Click on the import button, then NSD
- Drag and drop the NS package file cirros_2vnf_ns.tar.gz in the importing area.

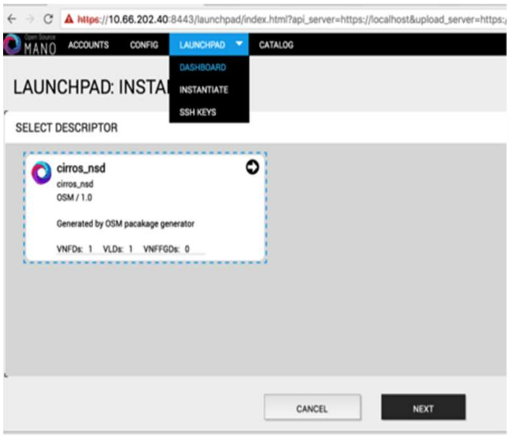
From OSM client:

```
osm upload-package cirros_2vnf_ns.tar.gz
osm nsd-list
```

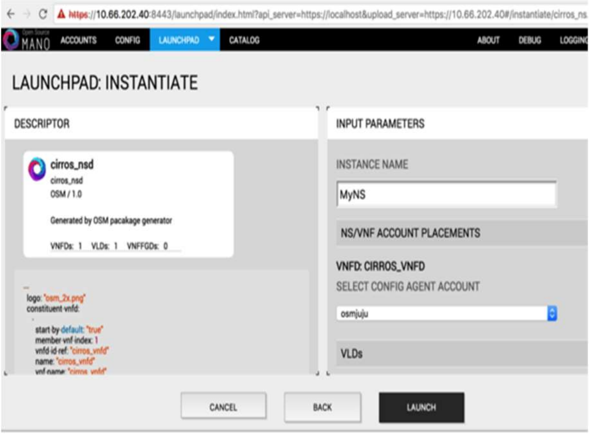
Instantiating the NS

From the UI:

- Go to Launchpad > Instantiate
- Select the NS descriptor to be instantiated, and click on Next
- Add a name to the NS instance, and click on Launch.



cirros_nsd
cirros_nsd
OSM / 1.0
Generated by OSM pacakage generator
VNFDs: 1 VLDs: 1 VNFFGDs: 0


https://10.66.202.40:8443/launchpad/index.html?api_server=https://localhost&upload_server=https://10.66.202.40#/instantiate/cirros_ns
MANO
ACCOUNTS
CONFIG
LAUNCHPAD
CATALOG
ABOUT
DEBUG
LOGGING
LAUNCHPAD: INSTANTIATE
DESCRIPTOR
INPUT PARAMETERS
CANCEL
BACK
LAUNCH

cirros_nsd
cirros_nsd
OSM / 1.0
Generated by OSM pacakage generator
VNFDs: 1 VLDs: 1 VNFFGDs: 0

```
logo: "osm_2x.png"
constituent-vnfd:
  -
    start-by-default: "true"
    member-vnf-index: 1
    vnfd-id-ref: "cirros_vnfd"
    name: "cirros_vnfd"
    vnf-name: "cirros_vnfd"
```

INSTANCE NAME: MyNS

NS/VNF ACCOUNT PLACEMENTS

VNFD: CIRROS_VNFD

SELECT CONFIG AGENT ACCOUNT: osmjuju

VLDs

From OSM client:

```
osm ns-create --nsd_name cirros_2vnf_ns --ns_name <ns-instance-name> --vim_account
<data-center-name>
```

```
osm ns-list
```

Using OSM to view instantiated NS details

OSM client can be used to manage the NSD and VNF catalog, as well as deploying and undeploying NS instances:

```
osm vnfd-list              # show the vnfd catalog
osm nsd-list               # show the nsd catalog
osm ns-list                # show ns instances
osm ns-show <id|name>       # show the details of a running ns instance
osm vnf-list               # show vnf instances
osm vnf-show <id|name>      # show the details of a running vnf instance
```

After finishing above operations, now you can try to deploy some complex service on this platform, the following additional information may give you some help.

Additional information

- Deploy advanced Network Services
- Create your own VNF package
- Reference VNF and NS Descriptors
- Creating your own VNF charm
- Example VNF Charms
- More information about OSM's Information Model
- VIM emulator
- Have you detected any bug? Check this guide to see how to report issues
- Life Cycle Management of VNFs from the RO
- OSM White Paper - Release THREE Technical Overview
- Technical and demonstration videos
- EPA and SDN assist
- OSM Multi-tenancy
- Software upgrade
- OSM MON Module Installation Guide
- OSM MON Usage Guide
- Logs and troubleshooting
- E2E tests
- Videos from workshops and events
- Technical FAQ

Configure the shadowsocks proxy in Ubuntu server

As you know, for some reason, many resources in foreign websites are not accessible. To bypass this problem, you can do this lab in the servers we provide, which are running overseas. You can also apply for a completely free server yourself in both Amazon Cloud or Google Cloud with your credit card.

You can follow the guide below to configure the proxy in your server or personal computer so that these resources will be accessible. We have already built the shadowsocks environment for you to use. You can ask the TAs for detail information.

Install python

```
sudo apt-get install python
```

Install the package helper of python

```
sudo apt-get install python-pip
```

Install shadowsocks through pip

```
sudo apt-get install python-pip
```

Configue the arguments in a file named `shadowsocks.json` to start shadowsocks

```
{
  "server": "{your-server}",
  "server_port":"{your-server-port}",
  "local_port": 1080,
  "password": "{your-password}",
  "timeout": 600,
  "method": "{your-method}"
}
```

Start the shadowsocks service

```
sudo sslocal -c shadowsocks.json -d start
```

Install polipo to configure the proxy

```
sudo apt-get install polipo
```

Modify the settings of polipo in `/etc/polipo/config`

```
logSyslog = true
logFile = /var/log/polipo/polipo.log
proxyAddress = "0.0.0.0"
socksParentProxy = "127.0.0.1:1080"
socksProxyType = socks5
chunkHighMark = 50331648
objectHighMark = 16384
serverMaxSlots = 64
serverSlots = 16
serverSlots1 = 32
```

Restart polipo so that the new settings can work

```
sudo /etc/init.d/polipo restart
```

Configure the proxy for shell by adding the following code into file `.bashrc`

```
export http_proxy=http://127.0.0.1:8123/
export https_proxy="https://127.0.0.1:8123/"
export ftp_proxy="ftp://127.0.0.1:8123/"
```

Types `source ~/.bash` so that the modification works.

Test if you can get access to google.com

```
curl www.google.com
```

TA Contact Information

Hao Yin:	**jsyhpain@163.com**
Zeyu Zhang:	**zeyu.damien.zhang@gmail.com**
QQ Group:	**561405991**