



Department of Computer Science
Computer Networks Research Group

Technology Review



Management of ServiCes Across MultipLE clouds

Authors:

ARKAJIT DHAR
ASHWIN PRASAD SHIVARPATNA VENKATESH
BHARGAVI MOHAN
DEEKSHA MYSORE RAMESH
HARSHITHA PANDITHANAHALLI SOMASHEKARAIH
SANKET KUMAR GUPTA
SUHEEL SHRIRANGAPURA NAZEERSAB
VIVEK JAGANATH

Supervisors:

Prof. Dr. Holger Karl | Sevil Dräxler | Hadi Razzaghi Kouchaksaraei

Paderborn, January 4, 2019

List of Figures

2.1	OSM GUI	v
2.2	Open Stack Dashboard	vi
2.3	Creating a Network in Openstack	vii
2.4	creating a valid tenant/user in openstack	vii
2.5	creating a valid tenant/user in openstack	vii
2.6	Uploading VM image to VIM in openstack	viii
2.7	Adding VIMs to OSM	ix
2.8	Adding VIMs to OSM	ix
2.9	On-boarding of VNFD in OSM	ix
2.10	On-boarding of NS in OSM	x
2.11	Initiating of NS in OSM	x
3.1	List of VNF Managers (VNFM)s	xiii
3.2	Register new PoP	xiii
3.3	Upload NSD	xiv
3.4	List of NSDs	xiv
3.5	List of VNFDs	xv
3.6	Upload NSD	xv
3.7	Upload NSD	xv
4.1	Sonata Dashboard	xviii
4.2	Add WIM	xix
4.3	Add VIM	xix
4.4	Select Router	xx
4.5	Select IDs	xx
4.6	VIM Details	xxi
4.7	Create user	xxii
4.8	Edit project quotas	xxiii
4.9	Create router	xxiv
4.10	REST call for NSD	xxiv
4.11	REST call for VNFD	xxv

Introduction

The aim of technology review is to understand the working of all the tools and technologies relevant for this project.

To achieve this, tasks were assigned to each of the team to review few set of tools as below.

Technology Review		
Team	Members	Reviewed
1	Arkajit Dhar	Open Source Mano
	Suheel Nazeersab	
	Vivek Jaganath	
2	Sanket Kumar	Open Baton
	Harshitha Somashekaraiah	
3	Ashwin Prasad	Sonata
	Bhargavi Mohan	
	Deeksha Ramesh	

MANO frameworks such as Open Source MANO, Sonata and Open Baton have been reviewed in this phase. Different virtual infrastructure managers are tried and worked upon , like Kubernetes and Open Stack.

The MANO frameworks are up and running in virtual machines and the connections are established between MANO and VIM's.

The detailed explanation on installation and steps are given below in the document.

2

Open Source MANO

2.1 Configuration requirements needed to run Open Source MANO (OSM) Release FOUR is a single server or VM:

- MINIMUM: 2 CPUs, 4 GB RAM, 20GB disk and a single interface with internet access
- RECOMMENDED: 2 CPUs, 8 GB RAM, 40GB disk and a single interface with internet access
- Ubuntu16.04 (64-bit variant required) as base image (<http://releases.ubuntu.com/16.04/>)

2.2 Open Source Mano Installation

2.2.1 Steps for Installation:

- Downloading latest version of OSM

```
wget https://osm-download.etsi.org/ftp/osm-4.0-four/install_osm.sh
```

- Installing OSM

```
chmod +x install_osm.sh  
$ ./install_osm.sh 2>&1 | tee osm_install_log.txt
```

2.2.2 verifying installation from the OSM GUI:

- Accessing GUI:

Access <http://1.2.3.4>, replacing 1.2.3.4 with the IP address of your host.
Login using Userid : admin , password : admin

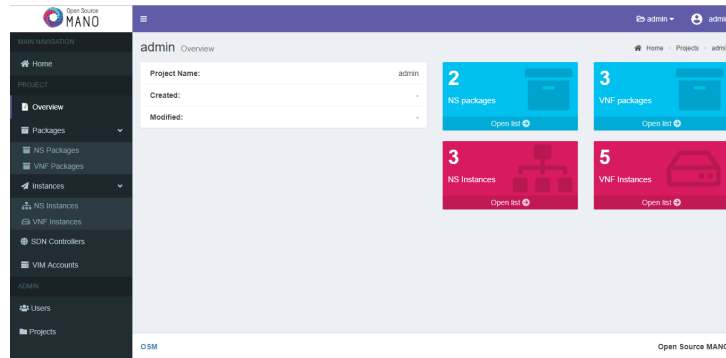


Figure 2.1: OSM GUI

- Verify 10 docker containers were created:

```
docker stack ps osm |grep -i running
docker service ls
```

2.3 VIM Installation

2.3.1 Steps to install openstack using devstack are as follows:

- Create a user “stack”

```
sudo useradd -s /bin/bash -d /opt/stack -m stack
echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack
sudo su -stack
```

- Clone the devstack repository

```
git clone https://git.openstack.org/openstack-dev/devstack
cd devstack
```

- Create and configure the local.conf file

```
[[local|localrc]]
ADMIN_PASSWORD=password
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD
```

- Execute the command

```
./stack.sh
```

- After installation check and verify from openstack horizon GUI:
 - Access <http://1.2.3.4>, replacing 1.2.3.4 with the IP address of your host. Login using Userid : admin , password : admin

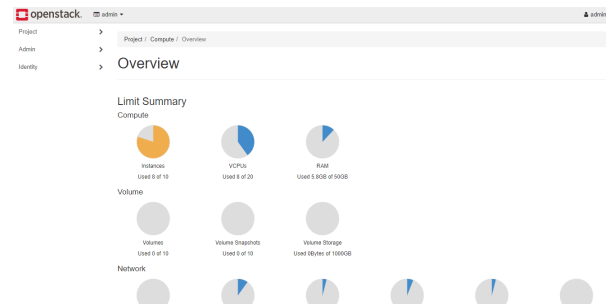


Figure 2.2: Open Stack Dashboard

2.4 Configure openstack for OSM

- Verify that Openstack API endpoints are reachable from OSM (particularly from RO container):
 - Login to openstack API access from the horizon GUI.
 - Click on DOWNLOAD OPENSTACK RC FILE (API version 3).
 - Copy the OS_AUTH_URL variable value.
 - Paste in the browser or do a curl from the VM where OSM is installed to check its reachability.
- Create a management network, with DHCP enabled, reachable from OSM (particularly from VCA container)
 - Login to openstack horizon GUI.
 - Go to admin-> create network.
 - Give the project name as your project (default:admin)
 - Give a network name -> mgmt.
 - Give a network subnet name and network address (10.208.1.0/24).
 - Keep the Network Address source as 'ENTER NETWORK ADDRESS MANUALLY'.
 - Keep Gateway IP blank.
 - In Allocation Pools, give the IPs: start=10.208.0.2,end=10.208.0.254.
 - Leave DNS Name servers and Host Routes blank and click create.
- creating a valid tenant/user
 - Login to openstack horizon gui.
 - Go to identity-> create user.
 - Give the project name as your project (default:admin)
 - Give a user name -> tenant.

Create Network

Network Subnet Subnet Details

Name

Project *
 Select a project

Provider Network Type *
 Local

☒ **Enable Admin State**

☐ **Shared**

☐ **External Network**

☒ **Create Subnet**

Availability Zone Hints
 nova

Cancel Back Next

Figure 2.3: Creating a Network in Openstack

- Give the role also as admin and click create.

Project Identity / Users

Admin Users

Users

Projects

Groups

Rules

Application Credentials

Displaying 10 items

User Name	Description	Email	User ID	Enabled	Domain Name	Actions
nova	-	-	1cc9dc0f8a441028954a295a294f	Yes	Default	Edit
all_demo	-	all_demo@example.com	16221f1cc0e49598194220146db87	Yes	Default	Edit
glider	-	-	3164c1d8c2d4f1ba79f59355a7230	Yes	Default	Edit

Create User Delete User

Figure 2.4: creating a valid tenant/user in openstack

Domain ID
 default

Domain Name
 Default

User Name *

Description

Email

Password *

Confirm Password *

Primary Project
 Select a project

Role
 member

☒ **Enabled**

Cancel Create User

Figure 2.5: creating a valid tenant/user in openstack

- Uploading VM image(s) to the VIM(s)
 - Download the image from the following link: (http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img)
 - Login to openstack horizon gui.
 - Go to admin -> Compute -> Images and click on create image.
 - Give the image name 'cirros034'
 - Upload the downloaded image file in step 1.
 - Choose the image format as QCOW2 : QEMU Emulator
 - Click on create image.

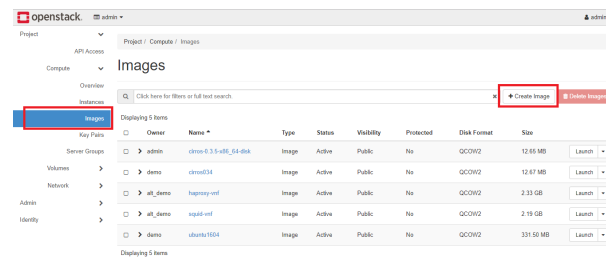


Figure 2.6: Uploading VM image to VIM in openstack

- Adding VIMs to OSM
 - Login to OSM and click on VIM Accounts.
 - Click on new VIM.
 - Give a name to your VIM instance and choose openstack from the type dropdown. Give the VIM URL as the OS_AUTH_URL variable value in openstack's rc file.
 - Enter the VIM userid and password as the login userid and password for openstack horizon gui.
 - Give the tenant name as admin/tenant.
 - Click on create.

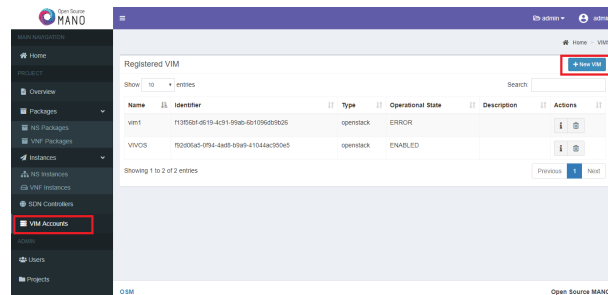


Figure 2.7: Adding VIMs to OSM

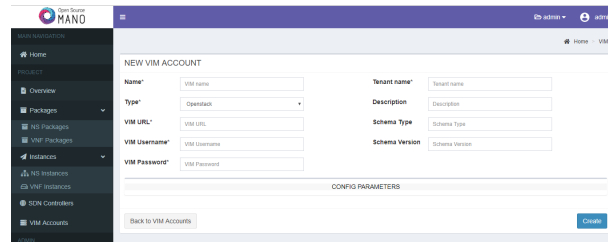


Figure 2.8: Adding VIMs to OSM

2.5 Deploying Network Service

First download the required VNF and NS packages from this URL: (https://osm-download.etsi.org/ftp/osm-3.0-three/examples/cirros_2vnf_ns/)

- On-boarding a VNFD
 - From the UI , Go to Projects → Admin → VNF Packages (Open List)
 - Click on the Onboard VNFD button
 - Drag and drop the VNF package file `cirros_vnf.tar.gz` in the importing area.

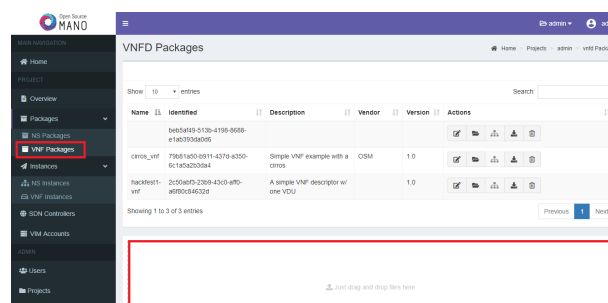


Figure 2.9: On-boarding of VNFD in OSM

- Onboarding a NS
 - From the UI, Go to Projects → Admin → NS Packages (Open List)
 - Click on the Onboard NSD button
 - Drag and drop the NS package file `cirros_2vnf_ns.tar.gz` in the importing area.

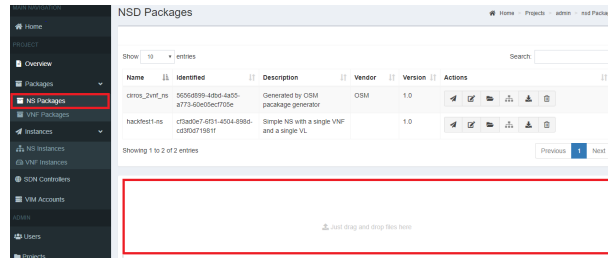


Figure 2.10: On-boarding of NS in OSM

- Instantiating the NS
 - From the UI, Go to Projects → Admin → NS Packages (Open List)
 - Next the NS descriptor to be instantiated, click on Launch
 - Fill the form, adding at least a name and selecting the VIM

New Instance

Name *

Ns name

Description *

Description

Nsd Id *

Select NSD

Vim Account Id *

Select VIM

SSH Key

Paste your key here...

Config

Yaml config

Cancel

Create

Figure 2.11: Initiating of NS in OSM

3.1 Configuration requirements to run Open Baton on a single server or VM

- Operating System: Ubuntu 16.04 as base image (<http://releases.ubuntu.com/16.04/>)
- You will need: Docker (≥ 18.03) and Docker Compose (≥ 1.20)
- Minimal Version: More than 2GB of RAM, and more than 2 CPUs, 10GB of disk space
- Complete Version: More than 8GB of RAM, and more than 8 CPUs, 10GB of disk space

3.2 Open Baton Installation

A minimal version of Open Baton is installed. Please note that Open Baton can only be installed on Ubuntu version 16.04 or older. Open Baton does not provide software package for Ubuntu's xenial (18.04) version yet. The installation guide can also be found at <https://openbaton.github.io/documentation/nfvo-installation-deb/>. A minimal version comprises of following components.

- Network Function Virtualization Orchestrator (NFVO)
- Test Virtual Infrastructure Manager (VIM)
- RabbitMQ as messaging system

Steps of installation:

- Install Curl

```
sudo apt-get install curl
```

- Install components from the Bootstrap repository

```
sh <(curl -s http://get.openbaton.org/bootstrap) release
```

- Know your IP Address

```
curl ifconfig.me
```

- Quick Start

```
curl -o docker-compose.yml  
https://raw.githubusercontent.com/openbaton/bootstrap/master/docker-compose.yml  
| env HOST_IP=$YOUR_LOCAL_IP docker-compose up -d
```

- Replace YOUR_LOCAL_IP with the IP address of your machine. After few seconds check if the OpenBaton dashboard is up and running on <https://localhost:8080>. You can login using following credentials:

- Username: admin
- Password: openbaton

3.3 Deploy a dummy Network Service

Once Open Baton is installed successfully, the following section lists out steps to deploy a dummy NS which needs following components:

- Network Function Virtualization Orchestrator (NFVO)
- Test VIM driver (We do not have to install it as it was installed as a part of bootstrap installation)
- Dummy Virtual Network Function Manager (Installation steps explained below)

Steps to install dummy VNFM Amqp

- Clone the project

```
git clone https://github.com/openbaton/dummy-vnfm-amqp.git
```

- Switch to the directory where dummy-vnfm-amqp is cloned and compile

```
cd dummy-vnfm-amqp; ./dummy-vnfm.sh compile
```

- Start the VNFM

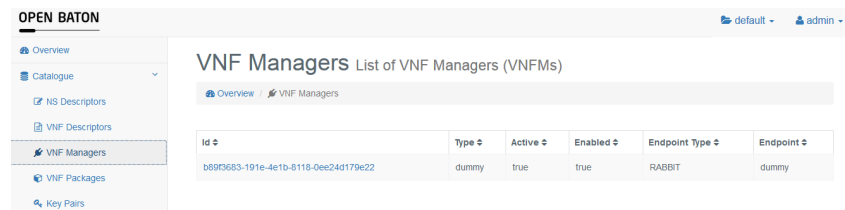
```
./dummy-vnfm.sh start  
java -jar build/libs/dummy-vnfm-amqp-6.0.1-SNAPSHOT.jar
```

- Start the NFVO with test vim driver

- Download the docker-compose file from <https://openbaton.github.io/documentation/compose/dummy-ns.yml>. This file contains dummy VNFM and test VIM driver.

```
docker-compose -p ob -f dummy-ns.yml up -d
```

- Open your browser and navigate to <http://localhost:8080> and login to the dashboard
- Verify the VNFM of type Dummy is listed under Catalog -> VNF Managers



The screenshot shows the Open Baton dashboard with the 'VNF Managers' section selected in the left sidebar. The main area displays a table titled 'VNF Managers List of VNF Managers (VNFM)' with the following data:

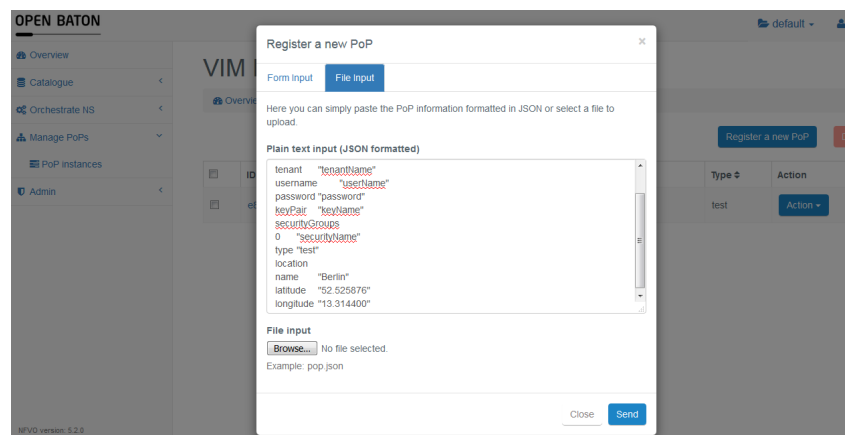
Id	Type	Active	Enabled	Endpoint Type	Endpoint
b69c663-191e-4e1b-8118-0ee24d179e22	dummy	true	true	RABBIT	dummy

Figure 3.1: List of VNF Managers (VNFM)

Deployment of NS using Open Baton dashboard

After installing dummy VNFM the next step is to create a new VIM instance by registering a new Point of Presence using Open Baton dashboard.

- Register a new PoP
 - Open your browser and navigate to <http://localhost:8080> and login to the dashboard
 - Go to Manage PoPs -> PoP Instances -> Register new PoP -> File Input
 - Upload a VIM instance of type test to the NFVO. Copy paste the JSON content of the link <https://openbaton.github.io/documentation/descriptors/vim-instance/test-vim-instance.json>.



The screenshot shows the 'Register a new PoP' dialog box in the Open Baton dashboard. The 'File Input' tab is selected. The dialog contains a text area for 'Plain text input (JSON formatted)' with the following JSON content:

```
{
  "tenant": "tenantName",
  "username": "username",
  "password": "password",
  "keyPair": "keyName",
  "securityGroups": [
    {
      "securityName": "securityName",
      "type": "test",
      "location": {
        "name": "Berlin",
        "latitude": "52.525876",
        "longitude": "13.314400"
      }
    }
  ]
}
```

Below the text area, there is a 'File Input' section with a 'Browse...' button and the text 'No file selected. Example: pop.json'. At the bottom right, there are 'Close' and 'Send' buttons.

Figure 3.2: Register new PoP

- Upload NSD
 - Open your browser and navigate to <http://localhost:8080> and login to the dashboard
 - Go to Catalog -> NS Descriptors -> On board NSD -> Upload JSON
 - Download NSD and upload it. The NSD can be found at <https://openbaton.github.io/documentation/dummy-NSR/tutorial-dummy-NSR.json>.
 - After uploading the NSD, it will be listed under Catalog -> NS Descriptors
 - Also VNF Descriptors can be seen under Catalog -> VNF Descriptors

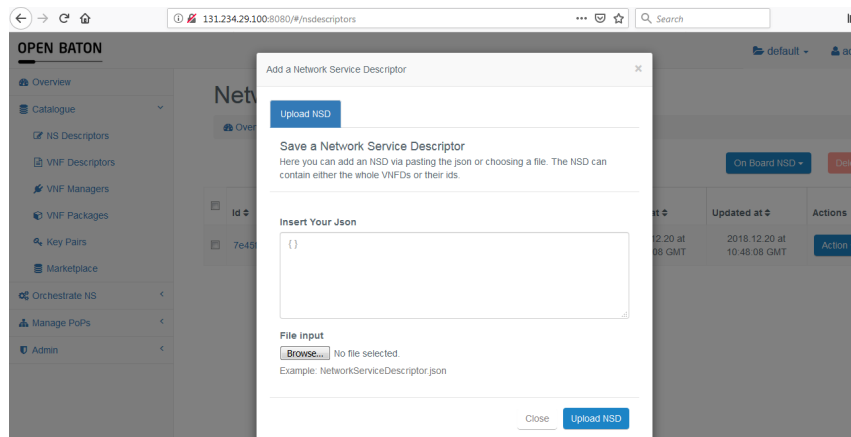


Figure 3.3: Upload NSD

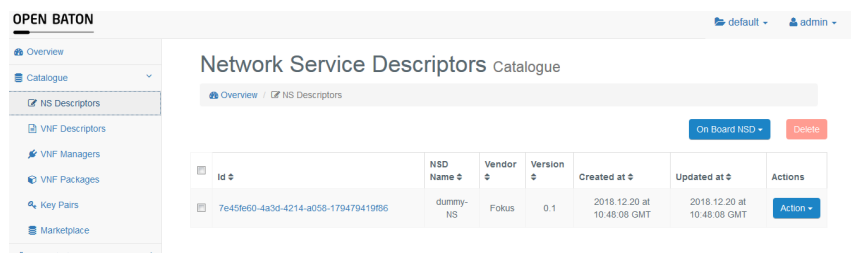


Figure 3.4: List of NSDs

- Deploy NSD
 - Open your browser and navigate to <http://localhost:8080> and login to the dashboard
 - After onboarding the NSD in the NFVO , deploy this NSD by using the dashboard.
 - Goto NS catalogue -> Action -> Launch -> Click Launch again
- Deploy NSD
 - Go to Orchestrate NS -> NS Records
 - You can see the NS with name dummy-NS and status as Active.

OPEN BATON

default

admin

Overview

Catalogue

NS Descriptors

VNF Descriptors

VNF Managers

VNF Packages

Key Pairs

Marketplace

Orchestrate NS

VNF Descriptors

List of VNF Descriptors

Create VNF

Delete

Id	VNFD name	VNFD type	Vendor	Version	Created at	Updated at	Actions
1976559f-e097-40b6-ad2d-a55b220327dd	dummy-server	server	Fokus	0.1	2018.12.20 at 10:48:08 GMT	2018.12.20 at 10:48:08 GMT	Action
6a30683c-32a1-4486-8b51-e400b60a6108	dummy-client	client	Fokus	0.1	2018.12.20 at 10:48:08 GMT	2018.12.20 at 10:48:08 GMT	Action

Figure 3.5: List of VNFDs

OPEN BATON

default

admin

Overview

Catalogue

NS Descriptors

VNF Descriptors

VNF Managers

VNF Packages

Key Pairs

Marketplace

Orchestrate NS

Manage PoPs

Admin

Launch Network Service Descriptor

PoPs

Keys

Configuration Parameters

Monitoring IP

General

dummy-client

dummy-server

Available

Assigned

test-vim-instance

Add to all

No PoPs are chosen for the selected VNFD.

Cancel

Back

Next

Launch

Figure 3.6: Upload NSD

OPEN BATON

default

admin

Overview

Catalogue

Orchestrate NS

NS Records

Events

Manage PoPs

Admin

Network Service Records

List of NSRs

Delete

Id	NSR Name	State	Created at	Updated at	Actions
859ac5f5-39b1-4ed2-a87d-fca7b2c44f96	dummy-NS	ACTIVE	2018.12.20 at 11:25:50 GMT	2018.12.20 at 11:26:32 GMT	Action

Figure 3.7: Upload NSD

SONATA/PISHAHANG

4.1 Configuration requirements to run Pishahang on a single server or VM

- Operating System: Ubuntu 16.04 as base image (<http://releases.ubuntu.com/16.04/>)
- Minimum Requirements: 4GB RAM, 40GB hard disk and a non-root user account

4.2 OpenStack Installation (Ocata)

Set up an OpenStack environment using DevStack, which is installed via a configuration file named local.conf. The installation guide can also be found at <https://docs.openstack.org/devstack/latest/>

- Other references ¹ ²

Steps of installation:

- Create a user “stack”

```
sudo useradd -s /bin/bash -d /opt/stack -m stack
echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack
sudo su - stack
```

- Clone the devstack repository

```
git clone https://git.openstack.org/openstack-dev/devstack -b stable/ocata
cd devstack
```

¹Refer DevStack heat documentation to enable heat service

²Refer DevStack networking-sfc documentation for service chaining

- Create and configure the local.conf file

```
[[local|localrc]]
ADMIN\_PASSWORD=password
DATABASE\_PASSWORD=$ADMIN\_PASSWORD
RABBIT\_PASSWORD=$ADMIN\_PASSWORD
SERVICE\_PASSWORD=$ADMIN\_PASSWORD
```

- Execute the command

```
./stack.sh
```

- After installation check and verify from openstack horizon GUI

Access <http://1.2.3.4>, replace 1.2.3.4 with the IP address of your host Login using user id: admin, password: admin

4.3 Pishahang installation

The Below steps of installation are performed from the non-root user account

- Installing packages

```
sudo apt-get install -y software-properties-common
sudo apt-add-repository -y ppa:ansible/ansible
sudo apt-get update
sudo apt-get install -y git ansible
```

- Clone repository

```
git clone https://github.com/CN-UPB/Pishahang.git cd Pishahang/son-install
echo sonata | tee ~/.ssh/.vault_pass
```

- Start Installation, replace "<your_ip4_address>" with the IP address where SONATA should be available.

```
ansible-playbook utils/deploy/sp.yml -e "target=localhost \
public_ip=<your_ip4_address>" -v
```

- Verify Installation

Open your browser and navigate to http://public_ip. Login using the username sonata and password 1234. If the installation was successful, you should now see the dashboard of the service platform

- Installation of son-cli The SONATA CLI toolset can also be installed via the Python setup script

```
git clone https://github.com/sonata-nfv/son-cli.git
cd son-cli
python3 setup.py install
```

- Test if its working by invoking

```
son-workspace -h
son-package -h
son-publish -h
son-push -h
son-monitor -h
```

Reference Link - <https://github.com/sonata-nfv/son-cli#all-dists-using-setuptools>

4.4 Service Descriptor Packaging and uploading

The son-cli is to be installed and son-examples repository to be cloned in the environment.

- Add WIM
 - Open your browser and navigate to http://public_ip
 - Open the "WIM/VIM Settings" tab

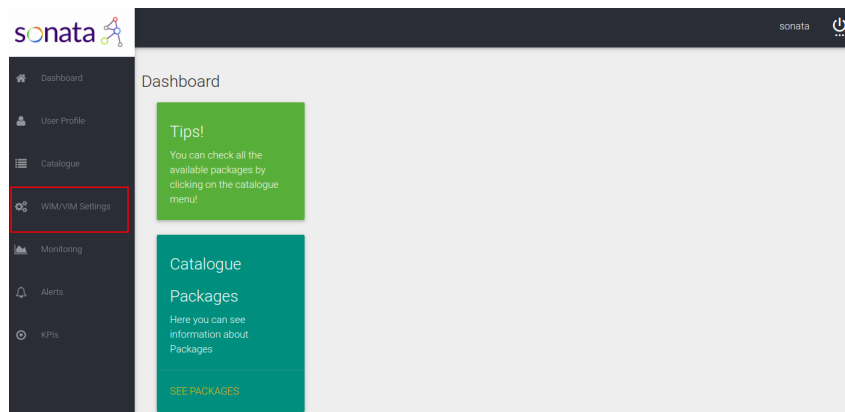


Figure 4.1: Sonata Dashboard

- click on add a WIM
- Select "Mock" WIM vendor
- Enter any WIM name(e.g. Sonata Test), WIM address(e.g. local host), username(e.g., Sonata) and password(e.g. 1234)
- Confirm by clicking "SAVE"

New Wim

Wim Name SonataTest	Wim vendor MOCK
Wim address localhost	Username Sonata
Password	

CANCEL SAVE

Figure 4.2: Add WIM

- Adding OpenStack VIM
 - Click on add a VIM
 - Enter the VIM name(e.g. DevStack) , select the WIM just created, enter the country(e.g. germany) and city(Paderborn)
 - Select "Heat" VIM vendor

New Vim

General Configuration

VIM Name
DevStavk

Select WIM
e1265246-51e7-491e-9134

Country
Germany

City
Paderborn

Compute Configuration

Vim Vendor
Heat

Networking Configuration

Network configuration VIM Type
Select vim vendor

CANCEL SAVE

Figure 4.3: Add VIM

- Tenant ID: DevStack project id (e.g. sonatademo), Tenant External Network ID: DevStack ID of the public network and Tenant External Router ID: DevStack ID of the router created under sonatademo user i.e. sonata-router as shown below

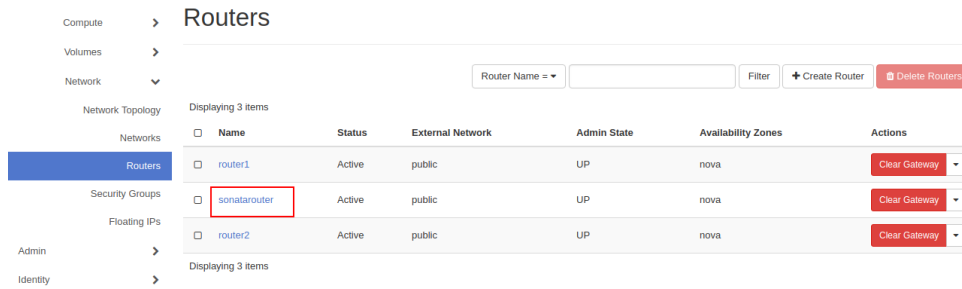


Figure 4.4: Select Router

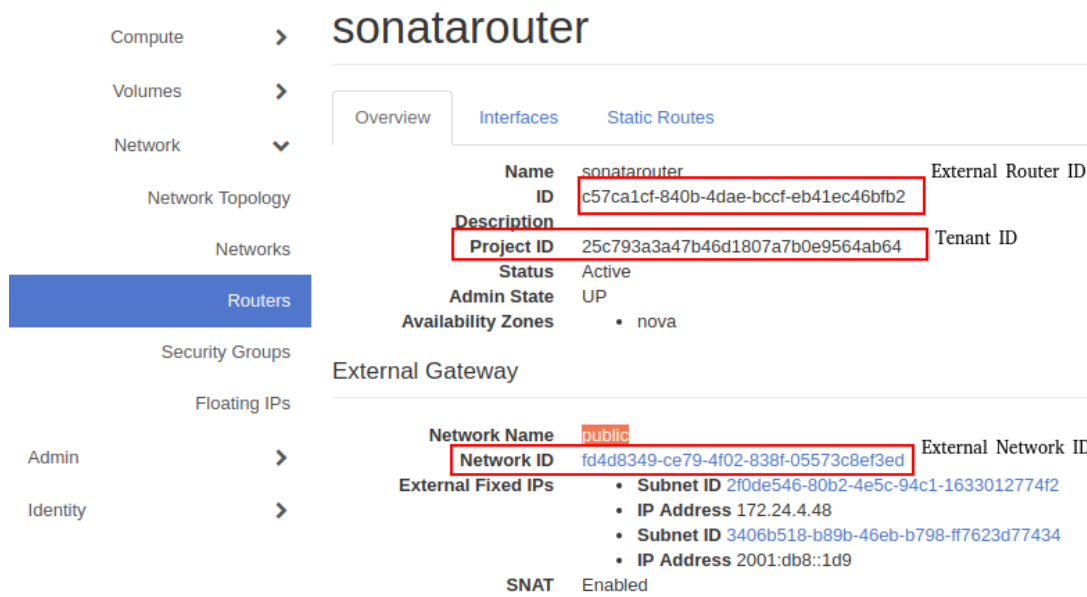


Figure 4.5: Select IDs

- VIM Address: DevStack (131.234.29.34)
- Vim Vendor: "OVS", Username: sonatademo, Password: password of the user sonatademo (e.g. sonata), Domain: Default
- Click on "Save"

Compute Configuration		Networking Configuration	
Vim Vendor		Network configuration VIM Type	
Heat		Select vim vendor	
Tenant ID	Vim address	OVS	
e3cf6987c0ff43c48	131.234.29.112		
Tenant External Network ID	Tenant External Router ID	Vim address	
fd4d8349-ce79-4f02	c57ca1cf-840b-4da1	131.234.29.112	
Username	Password	Username	Password
sonatademo	*****	sonatademo	*****
Domain			
Default			
		CANCEL SAVE	

Figure 4.6: VIM Details

– On-boarding Service Package

```
git clone https://github.com/sonata-nfv/son-examples.git
son-workspace --init
son-validate --project son-examples/service-projects/sonata-demo
son-package --project son-examples/service-projects/sonata-demo -n \
    service_package
son-access config --platform_id ServicePlatform --new --url \
    http://131.234.29.102 --default
son-access auth -u sonata -p 1234
son-access push --upload service_package.son
```

Reference video - <https://www.youtube.com/watch?v=RsXUIt4rzF0>

4.5 Linking VIM to sonata

Login to the DevStack dashboard: <http://131.234.29.34/dashboard>. There are two users created during installation admin and demo. Password for both users is sonata

- Create New User and Project

- Login as admin user in domain Default and create new user (e.g. sonatademo)
- In the menu, go to Identity->User (Create User)
- Give the admin role to the new user

Create User ✕

Domain ID

Domain Name

User Name *

Description

Email

Password *

Primary Project

▼

+

 ▼☒ **Enabled**

Cancel

Create User

Figure 4.7: Create user

- Add a new project with the below details
 - Project name/tenant name: sonatademo
 - Allocate maximum number of resources for that project under Quotas tab

Edit Quotas

Compute *

Volume *

Network *

Instances *

100|

VCPUs *

100

RAM (MB) *

51200

Metadata Items *

128

Key Pairs *

100

Server Groups *

10

Server Group Members *

10

Injected Files *

50

Injected File Content (Bytes) *

52100

Length of Injected File Path *

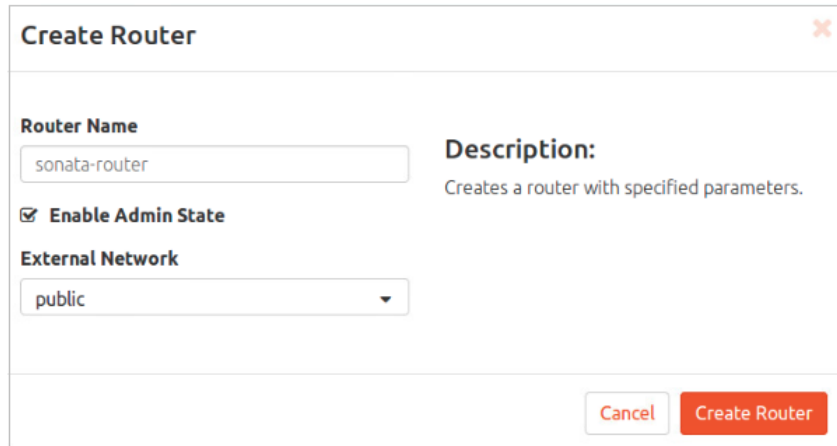
255

Cancel

Save

Figure 4.8: Edit project quotas

- Create Private Network
 - Login as new user(e.g. sonatademo)
 - Create a network(e.g. sonata-priv) and add the subnet as well (e.g. sonata-priv-sub)
 - Add the router
 - Use any private network address, for example 192.168.x.0/24. While creating the router select the External Network as public (Error: Reference source not found). Add the sonata-priv-sub as the interface to the router



Create Router

Router Name
sonata-router

Description:
Creates a router with specified parameters.

☒ **Enable Admin State**

External Network
public

Buttons: Cancel, Create Router

Figure 4.9: Create router

4.6 Onboarding Descriptors

NSD can be pushed to the server by using REST API provided by pishahang.

- For CSDs: http://public_ip:4002/catalogues/api/v2/csds
- For COSDs: http://public_ip:4002/catalogues/api/v2/complex-services
- Dummy NSD that has been uploaded can be seen in the appendix ??

Postman could be used to make the REST calls

1. NSD

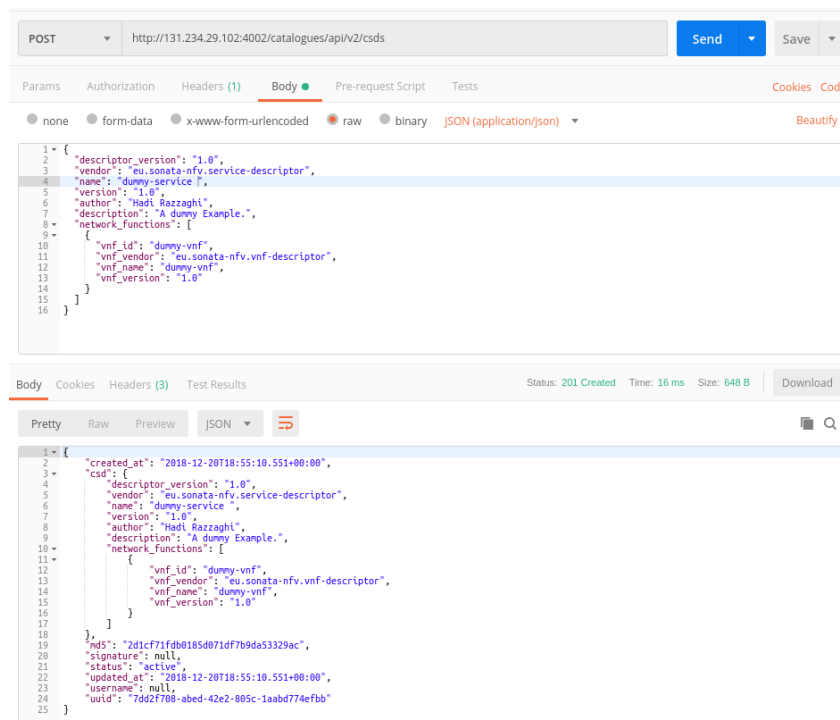


Figure 4.10: REST call for NSD

2. VNFD

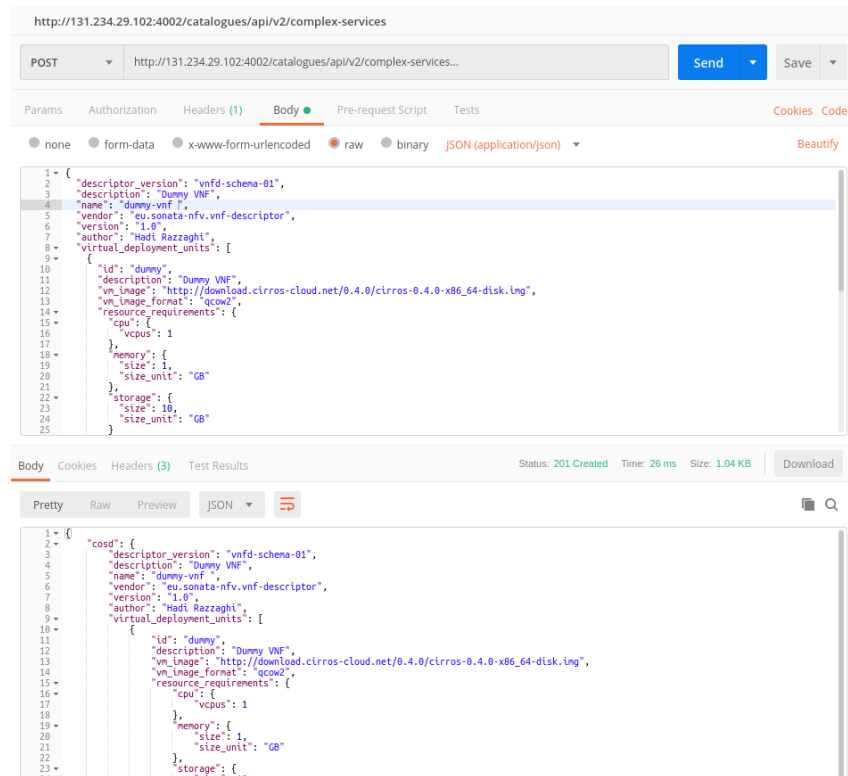


Figure 4.11: REST call for VNFD

4.7 Network Service Instantiation

- Open your browser and navigate to `http://public_ip:25001`
- Open the "Available Complex Services" tab
- Click the "Instantiate" button of the service you want to deploy
- Confirm the instantiate modal (ingress and egress can be empty)

Conclusion

The main goal of this phase is to acquire practical knowledge on MANO frameworks and its workflow. All the steps required to instantiate a network service are completed and verified.

Below defined are the findings from the technology review phase.

- Considering an environment with single domain, all of the network resources and services are managed by a single MANO orchestrator. However considering a multi-domain environment, where network services need to be deployed across multiple and different orchestrators, there is a need for a seamless communication between different orchestrators in order to deploy the end-to-end service successfully. Currently a major hindrance in the communication roots from the fact that each MANO framework uses different descriptor formats for describing the network service and virtual network function. One of the goals of this project is to overcome this shortcoming and implement a translator engine to translate the NSD and VNFD and facilitate communication between different orchestrators in a multi-domain environment.
- When a Network Service is required to be deployed over many parts of the world spanning multiple domains, a splitter is needed which splits a Network Service in smaller Network Services. Then these smaller Network Services are deployed over different domains. In this phase, minimal version of Open Baton orchestrator is installed on our personal machine and on the Virtual Machine provided. Dashboard of Open Baton is familiarised. Using the dashboard we were able to register PoP and upload NSD and then launch it. The deployment process of dummy NS did not actually create any virtual machines and no real network service is deployed.
- Currently, there are no means to add a MANO adopter to a main MANO instance which can communicate with other MANO frameworks to instantiate and monitor services running on them. The ability to do inter framework hierarchical orchestration is missing. Adding such an adopter will enable the MANO instances to scale according to the number of service requests. Therefore, implementing a MANO adaptor to tackle this issue is one of the milestones during the course of this project. MANO adopters for SONATA and OSM will be implemented first and OpenBaton would be considered in the next phase.

A

Appendix

Listing A.1: NSD

```
1 {
2   "descriptor_version": "1.0",
3   "vendor": "eu.sonata-nfv.service-descriptor",
4   "name": "dummy-service",
5   "version": "1.0",
6   "author": "Hadi Razzaghi",
7   "description": "A dummy Example.",
8   "network_functions": [
9     {
10      "vnf_id": "dummy-vnf",
11      "vnf_vendor": "eu.sonata-nfv.vnf-descriptor",
12      "vnf_name": "dummy-vnf",
13      "vnf_version": "1.0"
14    }
15  ]
16 }
```

Listing A.2: VNFD

```
1 {
2   "descriptor_version": "vnfd-schema-01",
3   "description": "Dummy VNF",
4   "name": "dummy-vnf",
5   "vendor": "eu.sonata-nfv.vnf-descriptor",
6   "version": "1.0",
7   "author": "Hadi Razzaghi",
8   "virtual_deployment_units": [
9     {
10      "id": "dummy",
11      "description": "Dummy VNF",
12    }
13  ]
14 }
```

```
13     "vm_image":  
14         "http://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-disk.img",  
15     "vm_image_format": "qcow2",  
16     "resource_requirements": {  
17         "cpu": {  
18             "vcpus": 1  
19         },  
20         "memory": {  
21             "size": 1,  
22             "size_unit": "GB"  
23         },  
24         "storage": {  
25             "size": 10,  
26             "size_unit": "GB"  
27         }  
28     },  
29     "connection_points": [  
30         {  
31             "id": "eth0",  
32             "interface": "ipv4",  
33             "type": "internal"  
34         },  
35         {  
36             "id": "eth1",  
37             "interface": "ipv4",  
38             "type": "internal"  
39         },  
40         {  
41             "id": "eth2",  
42             "interface": "ipv4",  
43             "type": "internal"  
44         }  
45     ],  
46     "user_data": {  
47         "password": "1234",  
48         "chpasswd": {  
49             "expire": false  
50         },  
51         "ssh_pwauth": true  
52     }  
53 ]  
54 }
```