OPNFV Configuration Guide

Release arno.2015.1.0 (71fa5c6)

OPNFV

February 01, 2016

CONTENTS

CHAPTER

ONE

ABSTRACT

This document provides guidance and instructions for the configuration of the Brahmaputra release of OPNFV.

The release includes four installer tools leveraging different technologies; Apex, Compass4nfv, Fuel and JOID, which deploy components of the platform.

This document provides a guide for the selection of tools and components including guidelines for how to deploy and configure the platform to an operational state.

CHAPTER

TWO

CONFIGURATION OPTIONS

OPNFV provides a variety of virtual infrastructure deployments called scenarios designed to host virtualised network functions (VNF's). Each scenario provide specific capabilities and/or components aimed to solve specific problems for the deployment of VNF's. A scenario may include components such as OpenStack, OpenDaylight, OVS, KVM etc. where each scenario will include different source components or configurations.

2.1 OPNFV Scenario's

Each OPNFV scenario provides unique features and capabilities, it is important to understand your target platform capabilities before installing and configuring your target scenario. This configuration guide outlines how to install and configure components in order to enable the features you require.

Scenarios are implemented as deployable compositions through integration with an installation tool. OPNFV supports multiple installation tools and for any given release not all tools will support all scenarios. While our target is to establish parity across the installation tools to ensure they can provide all scenarios, the practical challenge of achieving that goal for any given feature and release results in some disparity.

2.1.1 Scenario Naming

In OPNFV scenarios are identified by short scenario names, these names follow a scheme that identifies the key components and behaviours of the scenario. The rules for scenario naming are as follows:

os-[controller]-[feature]-[mode]-[option]

Details of the fields are

- os: mandatory
  - Refers to the platform type used
  - possible value: os (OpenStack)
- [controller]: mandatory
  - Refers to the SDN controller integrated in the platform
  - example values: nosdn, ocl, odl, onos
  - [feature]: mandatory
    - Refers to the feature projects supported by the scenario
    - example values: nofeature, kvm, ovs
  - [mode]: mandatory

        - Refers to the deployment type, which may include for instance high availability
        - possible values: ha, noha
    - [option]: optional
        - Used for the scenarios those do not fit into naming scheme.
        - The optional field in the short scenario name should not be included if there is no optional scenario.

Some examples of supported scenario names are:

- os-nosdn-kvm-noha
    - This is an OpenStack based deployment using neutron including the OPNFV enhanced KVM hypervisor
- os-odl_l2-nofeature-ha
    - This is an OpenStack deployment in high availability mode including OpenDaylight layer2 networking
- os-onos-kvm_ovs-noha
    - This is an OpenStack deployment using ONOS including OPNFV enhanced KVM and OVS versions

2.1.2 Installing your scenario

There are two main methods of deploying your target scenario, one method is to follow this guide which will walk you through the process of deploying to your hardware using scripts or ISO images, the other method is to set up a Jenkins slave and connect your infrastructure to the OPNFV Jenkins master.

For the purposes of evaluation and development a number of Brahmaputra scenarios are able to be deployed virtually to mitigate the requirements on physical infrastructure. Details and instructions on performing virtual deployments can be found in the installer specific installation instructions.

To set up a Jenkins slave for automated deployment to your lab, refer to the Jenkins slave connect guide.

2.1.3 Brahmaputra scenario overeview

The following table provides an overview of the installation tools and available scenario's in the Brahmaputra release of OPNFV.

Features ↓	Apex	Compass	Fuel	Joid
BGPVPN	?		✗	
Doctor	✗			
KVM4NFV			✗	
ONOS	✗	✗	✗	✗
OpenContrail	?	?	?	?
ODL Layer2	✗	✗	✗	✗
ODL Layer3	✗		✗	
OpenStack	✗	✗	✗	✗
OVS4NFV			✗	
SFC	✗		?	

This document will describe how to install and configure your target OPNFV scenarios. Remember to check the associated validation procedures section following your installation for details of the use cases and tests that have been run.

CHAPTER

THREE

INSTALLER CONFIGURATION

The following sections describe the per installer configuration options. Further details for each installer are captured in the referred project documentation.

3.1 <Project> configuration

Add a brief introduction to configure OPNFV with this specific installer

3.1.1 Pre-configuration activities

Describe specific pre-configuration activities. Refer to Installations guide and release notes

3.1.2 Hardware configuration

Describe the hardware configuration needed for this specific installer

3.1.3 Jumphost configuration

Describe intial Jumphost configuration (network and software)needed in order to deploy the installer

3.1.4 Platform components configuration

Describe the configuration of each component in the installer

CHAPTER

FOUR

DEPLOY JOID IN YOUR LAB

minimum 2 networks

```
1. First for Admin network with gateway to access external network
2. Second for public network to consume by tenants for floating ips
```

NOTE: JOID support multiple isolated networks for data as well as storage. Based on your network options for Openstack.

```
Minimum H/W Spec needed
CPU cores: 16
Memory: 32 GB
Hard Disk: 1(250 GB)
NIC: eth0(Admin, Management), eth1 (external network)
```

```
Minimum H/W Spec
CPU cores: 16
Memory: 32 GB
Hard Disk: 1(500 GB)
NIC: eth0(Admin, Management), eth1 (external network)
```

```
Minimum H/W Spec
CPU cores: 16
Memory: 32 GB
Hard Disk: 1(1 TB) this includes the space for ceph as well
NIC: eth0(Admin, Management), eth1 (external network)
```

NOTE: Above configuration is minimum and for better performance and usage of the Openstack please consider higher spec for each nodes.

Make sure all servers are connected to top of rack switch and configured accordingly. No DHCP server should be up and configured. Only gateway at eth0 and eth1 network should be configure to access the network outside your lab.

1. Install Ubuntu 14.04 LTS server version of OS on the nodes. 2. Install the git and bridge-utils packages on the server and configure minimum two bridges on jump host:

brAdm and brPublic cat /etc/network/interfaces

```
# The loopback network interface
auto lo
iface lo inet loopback
iface eth0 inet manual
auto brAdm
iface brAdm inet static
    address 10.4.1.1
    netmask 255.255.248.0
    network 10.4.0.0
    broadcast 10.4.7.255
    gateway 10.4.0.1
    # dns-* options are implemented by the resolvconf package, if installed
    dns-nameservers 10.4.0.2
    bridge_ports eth0
auto brPublic
iface brPublic inet static
    address 10.2.66.2
    netmask 255.255.255.0
    bridge_ports eth2
```

NOTE: If you choose to use the separate network for management, data and storage then you need to create bridge for each interface. In case of VLAN tags use the appropriate network on jump-host depend upon VLAN ID on the interface.

Get the joid code from gerritt

git clone https://gerrit.opnfv.org/gerrit/p/joid.git

cd joid/ci

Enable MAAS

- Create a directory in maas/<company name>/<pod number>/ for example

mkdir maas/intel/pod7/

- Copy files from pod5 to pod7

cp maas/intel/pod5/ maas/intel/pod7/*

4 files will get copied: deployment.yaml environments.yaml interfaces.host lxc-add-more-interfaces

This file has been used to configure your maas and bootstrap node in a VM. Comments in the file are self explanatory and we expect fill up the information according to match lab infrastructure information. Sample deployment.yaml can be found at https://gerrit.opnfv.org/gerrit/gitweb?p=joid.git;a=blob;f=ci/maas/intel/pod5/deployment.yaml

under section case $3 add the intelpod7 section and make sure you have information provided correctly. Before example consider your network has 192.168.1.0/24 your default network. and eth1 is on public network which will be used to assign the floating ip.

```
    'intelpod7' )

# As per your lab vip address list be deafult uses 10.4.1.11 - 10.4.1.20
        sed -i -- 's/10.4.1.1/192.168.1.2/g' ./bundles.yaml
       # Choose the external port to go out from gateway to use.

sed -i -- 's/#        "ext-port": "eth1"/        "ext-port": "eth1"/g' ./bundles.yaml
       ;;
```

NOTE: If you are using seprate data network then add this line below also along with other changes. which represents network 10.4.9.0/24 will be used for data network for openstack

```
        sed -i -- 's/#os-data-network: 10.4.8.0\/21/os-data-network: 10.4.9.0\/24/g' ./bun
```

under section case $1 add the intelpod7 section and make sure you have information provided correctly.

```
     'intelpod7' )
       cp maas/intel/pod7/deployment.yaml ./deployment.yaml
       ;;
```

NOTE: If you are using VLAN tags or more network for data and storage then make sure you modify the case $1 section under Enable vlan interface with maas appropriately. In the example below eth2 has been used as separate data network for tenants in openstack with network 10.4.9.0/24 on compute and control nodes.

```
   'intelpod7' )
       maas refresh
       enableautomodebyname eth2 AUTO "10.4.9.0/24" compute || true
       enableautomodebyname eth2 AUTO "10.4.9.0/24" control || true
       ;;
```

Once you have done the change in above section then run the following commands to do the automatic deployments.

After integrating the changes as mentioned above run the MAAS install. Suppose you name the integration lab as intelpod7 then run the below commands to start the MAAS deployment.

```
   ./02-maasdeploy.sh intelpod7
```

```
   ./deploy.sh -o liberty -s odl -t ha -l intelpod7 -f none
```

NOTE: Possible options are as follows:

```
   choose which sdn controller to use.
     [-s ]
     nosdn: openvswitch only and no other SDN.
     odl: OpenDayLight Lithium version.
```

```
opencontrail: OpenContrail SDN can be installed with Juno Openstack today.
     onos: ONOS framework as SDN.

     [-t ]
     nonha: NO HA mode of Openstack
     ha: HA mode of openstack.
     [-o ]
     juno: Juno Openstack
     liberty: Liberty version of openstack.
     [-l ] etc...

default: For virtual deployment where installation will be done on KVM created using ./02-n
     intelpod5: Install on bare metal OPNFV pod5 of Intel lab.
     intelpod6
     orangepod2
     ..
     ..
     : if you make changes as per your pod above then please use that.
     [-f ]
     none: no special feature will be enabled.
     ipv6: ipv6 will be enabled for tenant in openstack.
```

By default debug is enabled in script and error messages will be printed on ssh terminal where you are running the scripts.

To access of any control or compute nodes. juju ssh for example to login into openstack-dashboard container.

```
juju ssh openstack-dashboard/0
juju ssh nova-compute/0
juju ssh neutron-gateway/0
```

By default juju will add the Ubuntu user keys for authentication into the deployed server and only ssh access will be available.

CHAPTER

FIVE

FEATURE CONFIGURATION

The following sections describe the configuration options for specific platform features provided in Brahmaputra. Further details for each feature are captured in the referred project documentation.

5.1 Configuring SDNVPN features

Stuff about configuring SDNVPN features...

CHAPTER

SIX

POST CONFIGURATION ACTIVITIES

This section describes the post configuration activitites that will allow you to validate the success of your configuration. Further details may be found in the referred project specific documentation.

6.1 Functional testing Installation

Pull the Functest Docker image from the Docker hub:

```
$ docker pull opnfv/functest:brahmaputra.1.0
```

Check that the image is available:

```
$ docker images
```

Run the docker container giving the environment variables:

```
- INSTALLER_TYPE. Possible values are "apex", "compass", "fuel" or "joid".
- INSTALLER_IP. each installer has its installation strategy.
```

Functest may need to know the IP of the installer to retrieve the credentials (e.g. usually “10.20.0.2” for fuel, not neede for joid...).

The minimum command to create the Functest docker file can be described as follows:

```
docker run -it -e "INSTALLER_IP=10.20.0.2" -e "INSTALLER_TYPE=fuel" opnfv/functest:brahmaputra.1.0 /k
```

Optionally, it is possible to precise the container name through the option –name:

```
docker run --name "CONTAINER_NAME" -it -e "INSTALLER_IP=10.20.0.2" -e "INSTALLER_TYPE=fuel" opnfv/fun
```

It is also possible to to indicate the path of the OpenStack creds using -v:

```
docker run  -it -e "INSTALLER_IP=10.20.0.2" -e "INSTALLER_TYPE=fuel" -v <path_to_your_local_creds_fil
```

The local file will be mounted in the container under /home/opnfv/functest/conf/openstack.creds

After the run command the prompt appears which means that we are inside the container and ready to run Functest.

Inside the container, the following directory structure should be in place:

```
`-- home
    `-- opnfv
      |-- functest
      |   |-- conf
      |   |-- data
      |   `-- results
```

```
    `-- repos
        |-- bgpvpn
        |-- functest
        |-- odl_integration
        |-- rally
        |-- releng
        `-- vims-test
```

Basically the container includes:

- Functest directory to store the configuration (the OpenStack creds are paste in /home/opngb/functest/conf), the data (images neede for test for offline testing), results (some temporary artifacts may be stored here)
- Repositories: the functest repository will be used to prepare the environment, run the tests. Other repositories are used for the installation of the tooling (e.g. rally) and/or the retrieval of feature projects scenarios (e.g. bgpvpn)

The arborescence under the functest repo can be described as follow:

```
.
|-- INFO
|-- LICENSE
|-- commons
|   |-- ims
|   |-- mobile
|   `-- traffic-profile-guidelines.rst
|-- docker
|   |-- Dockerfile
|   |-- common.sh
|   |-- prepare_env.sh
|   |-- requirements.pip
|   `-- run_tests.sh
|-- docs
|   |-- configguide
|   |-- functest.rst
|   |-- images
|   `-- userguide
`-- testcases
    |-- Controllers
    |-- VIM
    |-- __init__.py
    |-- config_functest.py
    |-- config_functest.yaml
    |-- functest_utils.py
    |-- functest_utils.pyc
    |-- vIMS
    `-- vPing
```

We may distinguish 4 different folders:

- commons: it is a folder dedicated to store traffic profile or any test inputs that could be reused by any test project
- docker: this folder includes the scripts that will be used to setup the environment and run the tests
- docs: this folder includes the user and installation/configuration guide
- testcases: this folder includes the scripts required by Functest internal test cases

Firstly run the script to install functest environment:

```
$ ${repos_dir}/functest/docker/prepare_env.sh
```

NOTE: ${repos_dir} is a default environment variable inside the docker container, which points to /home/opnfv/repos

Run the script to start the tests:

```
$ ${repos_dir}/functest/docker/run_tests.sh
```

6.2 <Project> post installation procedures

Add a brief introduction to the methods of validating the installation according to this specific installer or feature.

6.2.1 Automated post installation activities

Describe specific post installation activities performed by the OPNFV deployment pipeline including testing activities and reports. Refer to the relevant testing guides, results, and release notes.

note: this section should be singular and derived from the test projects once we have one test suite to run for all deploy tools. This is not the case yet so each deploy tool will need to provide (hopefully very simillar) documentation of this.

6.2.2 <Project> post configuration procedures

Describe any deploy tool or feature specific scripts, tests or procedures that should be carried out on the deployment post install and configuration in this section.

6.2.3 Platform components validation

Describe any component specific validation procedures necessary for your deployment tool in this section.