# SDN VPN user guide

***Release draft (fd6f067)***

**OPNFV**

August 23, 2016

CONTENTS


**1 Introduction** **1**

**2 SDN VPN feature description** **3**

**3 Hardware requirements** **5**
3.1 Bare metal deployment on Pharos Lab . . . . . 5
3.2 Virtual deployment hardware requirements . . . . . 5

**4 Preparing the host to install Fuel by script** **7**
4.1 Installation of required packages . . . . . 7
4.2 Download the source code and artifact . . . . . 7

**5 Simplified scenario deployment procedure using Fuel** **9**
5.1 Scenario Preparation . . . . . 9
5.2 Installation procedures . . . . . 9

**6 Feature and API usage guidelines and example** **11**
6.1 Example . . . . . 11

**7 Troubleshooting** **13**

CHAPTER

ONE

# INTRODUCTION

This document will provide an overview of how to work with the SDN VPN features in OPNFV.

CHAPTER

TWO

# SDN VPN FEATURE DESCRIPTION

A high-level description of the scenarios is provided in this section. For details of the scenarios and their provided capabilities refer to the scenario description document: http://artifacts.opnfv.org/colorado/sdnpvn/scenarios/os-odl_l2-bgpvpn/index.html

The BGPVPN feature enables creation of BGP VPNs on the Neutron API according to the OpenStack BGPVPN blueprint at https://blueprints.launchpad.net/neutron/+spec/neutron-bgp-vpn. In a nutshell, the blueprint defines a BGPVPN object and a number of ways how to associate it with the existing Neutron object model, as well as a unique definition of the related semantics. The BGPVPN framework supports a backend driver model with currently available drivers for Bagpipe, OpenContrail, Nuage and OpenDaylight. The OPNFV scenario makes use of the OpenDaylight driver and backend implementation through the ODL VPNService project.

CHAPTER

THREE

# HARDWARE REQUIREMENTS

The SDNVPN scenarios can be deployed as a bare-metal or a virtual environment on a single host.

## 3.1 Bare metal deployment on Pharos Lab

Hardware requirements for bare-metal deployments of the OPNFV infrastructure are specified by the Pharos project. The Pharos project provides an OPNFV hardware specification for configuring your hardware at: http://artifacts.opnfv.org/pharos/docs/pharos-spec.html.

## 3.2 Virtual deployment hardware requirements

To perform a virtual deployment of an OPNFV scenario on a single host, that host has to meet the hardware requirements outlined in the <missing spec>.

When ODL is used as an SDN Controller in an OPNFV virtual deployment, ODL is running on the OpenStack Controller VMs. It is therefore recommended to increase the amount of resources for these VMs.

Our recommendation is to have 2 additional virtual cores and 8GB additional virtual memory on top of the normally recommended configuration.

Together with the commonly used recommendation this sums up to:

```
4 virtual cores
16 GB virtual memory
```

See in Installation section below how to configure this.

CHAPTER

FOUR

# PREPARING THE HOST TO INSTALL FUEL BY SCRIPT

Before starting the installation of the <scenario> scenario some preparation of the machine that will host the Fuel VM must be done.

## 4.1 Installation of required packages

To be able to run the installation of the basic OPNFV fuel installation the Jumphost (or the host which serves the VMs for the virtual deployment) needs to install the following packages:

```
sudo apt-get install -y git make curl libvirt-bin libpq-dev qemu-kvm \
                        qemu-system tightvncserver virt-manager sshpass \
                        fuseiso genisoimage blackbox xterm python-pip \
                        python-git python-dev python-oslo.config \
                        python-pip python-dev libffi-dev libxml2-dev \
                       libxslt1-dev libffi-dev libxml2-dev libxslt1-dev \
                       expect curl python-netaddr p7zip-full

sudo pip install GitPython pyyaml netaddr paramiko lxml scp \
                 python-novaclient python-neutronclient python-glanceclient \
                 python-keystoneclient debtcollector netifaces enum
```

## 4.2 Download the source code and artifact

To be able to install the scenario os-odl_l2-bgpvpn one can follow the way CI is deploying the scenario. First of all the opnfv-fuel repository needs to be cloned:

```
git clone ssh://<user>@gerrit.opnfv.org:29418/fuel
```

This command downloads the whole repository fuel. We need now to switch it to the stable Colorado branch:

```
cd fuel
git checkout stable/colorado
```

Now download the appropriate OPNFV Fuel ISO into an appropriate folder:

```
wget http://artifacts.opnfv.org/fuel/colorado/opnfv-colorado.1.0.iso
```

The exact name of the ISO image may change. Check https://www.opnfv.org/opnfv-colorado-fuel-users to get the latest ISO.

CHAPTER

FIVE

# SIMPLIFIED SCENARIO DEPLOYMENT PROCEDURE USING FUEL

This section describes the installation of the os-odl_l2-bgpvpn-ha or os-odl_l2-bgpvpn-noha OPNFV reference platform stack across a server cluster or a single host as a virtual deployment.

## 5.1 Scenario Preparation

dea.yaml and dha.yaml need to be copied and changed according to the lab-name/host where you deploy. Copy the full lab config from:

```
cp <path-to-opnfv-fuel-repo>/deploy/config/labs/devel-pipeline/elx \
   <path-to-opnfv-fuel-repo>/deploy/config/labs/devel-pipeline/<your-lab-name>
```

Add at the bottom of dha.yaml

```
disks:
  fuel: 100G
  controller: 100G
  compute: 100G

define_vms:
  controller:
    vcpu:
      value: 4
    memory:
      attribute_equlas:
        unit: KiB
      value: 16388608
    currentMemory:
      attribute_equlas:
        unit: KiB
      value: 16388608
```

Check if the default settings in dea.yaml are in line with your intentions and make changes as required.

## 5.2 Installation procedures

We describe several alternative procedures in the following. First, we describe several methods that are based on the deploy.sh script, which is also used by the OPNFV CI system. It can be found in the Fuel repository.

In addition, the SDNVPN feature can also be configured manually in the Fuel GUI. This is described in the last subsection.

Before starting any of the following procedures, go to

```
cd <opnfv-fuel-repo>/ci
```

### 5.2.1 Full automatic virtual deployment High Availablity Mode

The following command will deploy the high-availability flavor of SDNVPN scenario os-odl_l2-bgpvpn-ha in a fully automatic way, i.e. all installation steps (Fuel server installation, configuration, node discovery and platform deployment) will take place without any further prompt for user input.

```
sudo bash ./deploy.sh -b file://<path-to-opnfv-fuel-repo>/config/ -l devel-pipeline -p <your-lab-name
```

### 5.2.2 Full automatic virtual deployment NO High Availability Mode

The following command will deploy the SDNVPN scenario in its non-high-availability flavor (note the different scenario name for the -s switch). Otherwise it does the same as described above.

```
sudo bash ./deploy.sh -b file://<path-to-opnfv-fuel-repo>/config/ -l devel-pipeline -p <your-lab-name
```

### 5.2.3 Automatic Fuel installation and manual scenario deployment

A useful alternative to the full automatic procedure is to only autodeploy the Fuel host and to run host selection, role assignment and SDNVPN scenario configuration manually.

```
sudo bash ./deploy.sh -b file://<path-to-opnfv-fuel-repo>/config/ -l devel-pipeline -p <your-lab-name
```

With -e option the installer does not launch environment deployment, so a user can do some modification before the scenario is really deployed. Another interesting option is the -f option which deploys the scenario using an existing Fuel host.

The result of this installation is a fuel sever with the right config for BGPVPN. Now the deploy button on fuel dashboard can be used to deploy the environment. It is as well possible to do the configuration manuell.

### 5.2.4 Feature configuration on existing Fuel

If a Fuel server is already provided but the fuel plugins for Opendaylight, Openvswitch and BGPVPN are not provided install them by:

```
cd /opt/opnfv/
fuel plugins --install fuel-plugin-ovs-*.noarch.rpm
fuel plugins --install opendaylight-*.noarch.rpm
fuel plugins --install bgpvpn-*.noarch.rpm
```

If plugins are installed and you want to update them use –force flag.

Now the feature can be configured. Create a new environment with “Neutron with ML2 plugin” and in there “Neutron with tunneling segmentation”. Then go to settings/other and check “OpenDaylight plugin”, “Install Openvswitch with NSH/DPDK” and “BGPVPN plugin”. Then you should be able to check “BGPVPN extensions” in OpenDaylight plugin section.

Now the deploy button on fuel dashboard can be used to deploy the environment.

CHAPTER

SIX

# FEATURE AND API USAGE GUIDELINES AND EXAMPLE

For the details of using OpenStack BGPVPN API, please refer to the documentation at http://docs.openstack.org/developer/networking-bgpvpn/.

## 6.1 Example

In the example we will show a BGPVPN associated to 2 neutron networks. The BGPVPN will have the import and export routes in the way that it imports its own Route. The outcome will be that vms sitting on these two networks will be able to have a full L3 connectivity.

Some defines:

```
net_1="Network1"
net_2="Network2"
subnet_net1="10.10.10.0/24"
subnet_net2="10.10.11.0/24"
```

Create neutron networks and save network IDs:

```
neutron net-create --provider:network_type=local $net_1
export net_1_id=`echo "$rv" | grep " id " |awk '{print $4}'`
neutron net-create --provider:network_type=local $net_2
export net_2_id=`echo "$rv" | grep " id " |awk '{print $4}'`
```

Create neutron subnets:

```
neutron subnet-create $net_1 --disable-dhcp $subnet_net1
neutron subnet-create $net_2 --disable-dhcp $subnet_net2
```

Create BGPVPN:

```
neutron bgpvpn-create --route-distinguishers 100:100 --route-targets 100:2530 --name L3_VPN
```

Start VMs on both networks:

```
nova boot --flavor 1 --image <some-image> --nic net-id=$net_1_id vm1
nova boot --flavor 1 --image <some-image> --nic net-id=$net_2_id vm2
```

The VMs should not be able to see each other.

Associate to Neutron networks:

```
neutron bgpvpn-net-assoc-create L3_VPN --network $net_1_id
neutron bgpvpn-net-assoc-create L3_VPN --network $net_2_id
```

Now the VMs should be able to ping each other

CHAPTER

SEVEN

# TROUBLESHOOTING

Check neutron logs on the controller:

```
tail -f /var/log/neutron/server.log |grep -E "ERROR|TRACE"
```

Check Opendaylight logs:

```
tail -f /opt/opendaylight/data/logs/karaf.log
```

Restart Opendaylight:

```
service opendaylight restart
```