

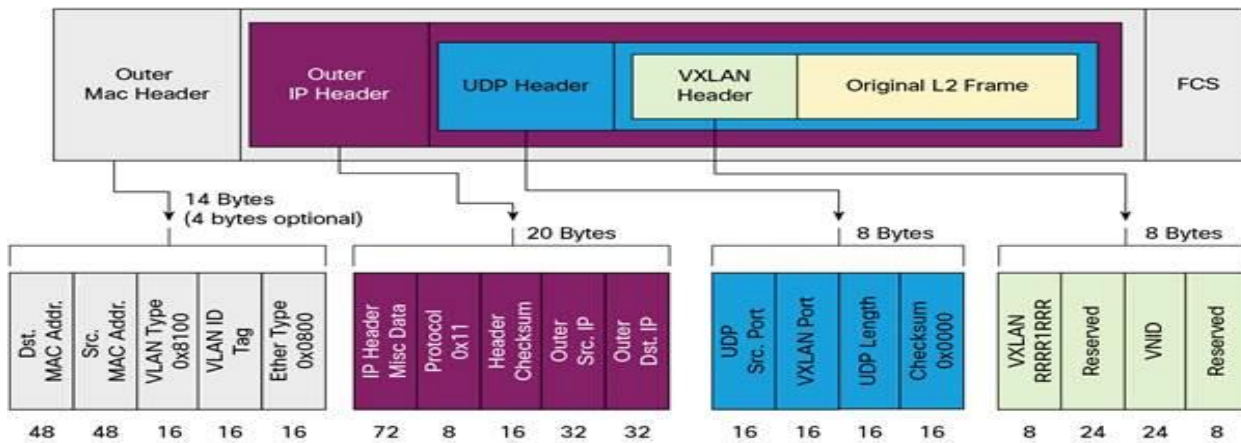
This paper outlines how Pica8’s OS (PicOS) running on a White Box SDN switch is used as a VXLAN Layer 2 Tunnel End Point (VTEP) to connect OpenStack virtual infrastructure to physical infrastructure by leveraging Midokura MidoNet.

Virtual eXtensible LAN (VXLAN) is a standard-based Layer 2 overlay technology, defined in RFC 7348. VXLAN provides the same Ethernet Layer 2 network services as a VLAN, but with greater scalability, extensibility and flexibility. VXLAN provides multi-tenancy across the data centers by extending Layer 2 segments over Layer 3 boundaries. With VXLAN, up to 16M Layer 2 segments are possible in contrast to only 4K with a VLAN. VXLAN is suitable for large-scale deployments when a 4K Layer 2 segment is not enough. VXLAN is also used as an overlay solution to extend Layer 2 segments over Layer 3 segments. One of the common use cases of VXLAN is multi-tenancy in OpenStack networking.

VXLAN

VXLAN is an overlay technology. It uses UDP for transporting Layer 2 MAC frames; it is a MAC-in-UDP encapsulation method. In VXLAN, the original Layer 2 frame is encapsulated inside an IP-UDP packet by adding VXLAN header as illustrated in Figure 1.

Figure 1 VXLAN Packet Format

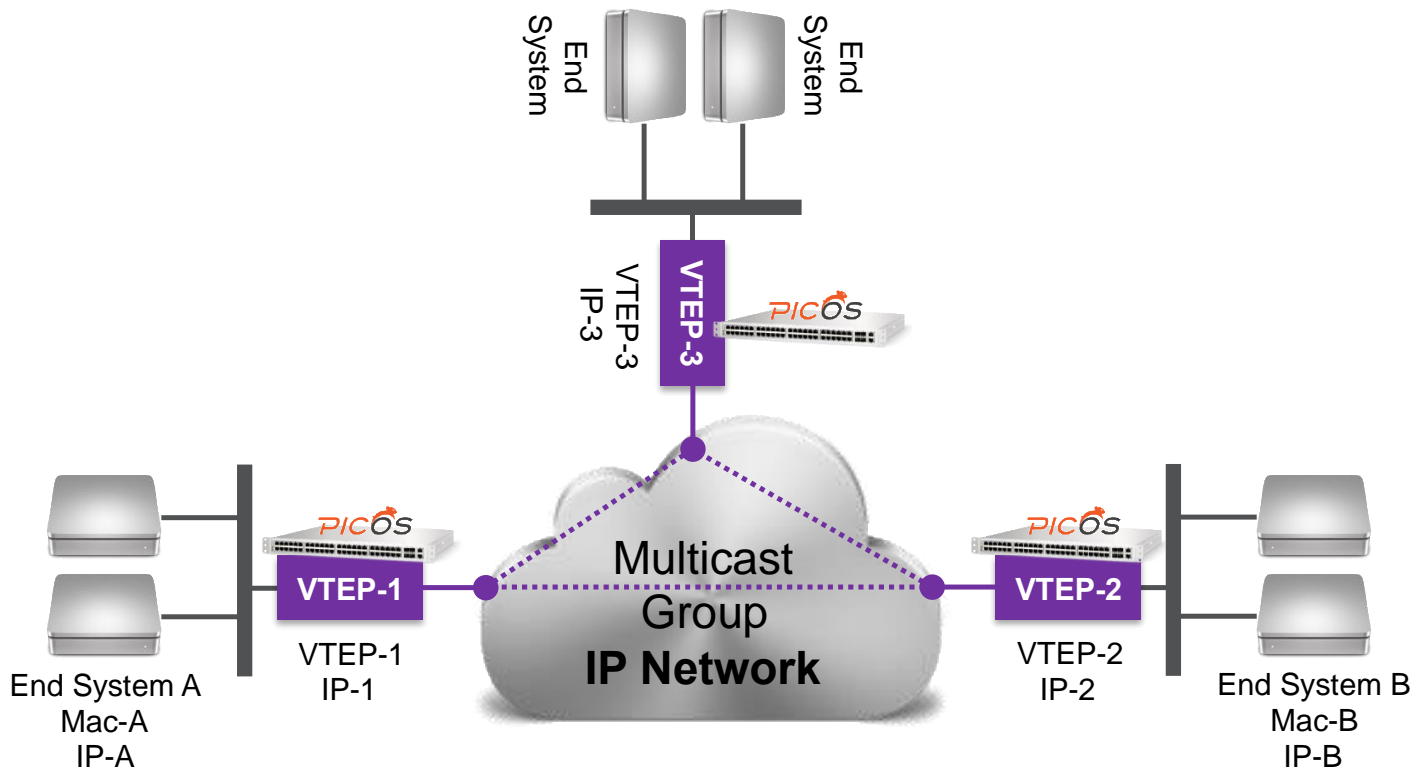


VXLAN header is 8-bytes in length and consists of a 24-bit VXLAN network identifier (VNID) providing up to 16M Layer 2 segments.

VXLAN Tunnel Endpoint (VTEP)

VXLAN uses VTEP to map end devices or tenants to VXLAN and perform both encapsulation and de-encapsulation functions. VTEP consists of two interfaces –one for local LAN and one for IP interfaces to connect to other VTEPs across an IP network. The IP interface identifies the VTEP device with a unique IP address assigned to the interface. VTEP uses IP interface to encapsulate Layer 2 frames and then transports the resulting encapsulated packet over the IP network. Additionally, VTEP discovers the remote VTEPs for relevant VXLAN segments and learns remote MAC Address-to-VTEP binding via the IP interface. The functional components of VTEPs and corresponding logical topology are illustrated in Figure 2

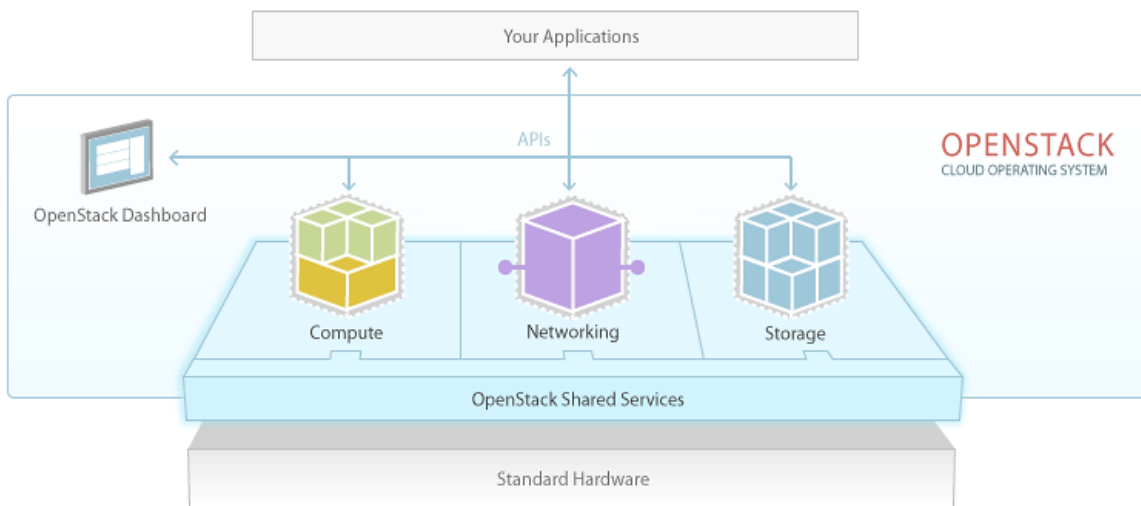
Figure 2 VTEP



OpenStack

OpenStack is an open-source based cloud-computing platform that provides infrastructure as a service (IaaS) solution via a set of related services. In OpenStack, individual services provide APIs to enable integration. Figure 3 depicts an OpenStack solution.

Figure 3 OpenStack



VXLAN in OpenStack

OpenStack supports VXLAN through a set of Neutron plugins. One of the challenges of VXLAN is how to manage MAC-VTEP Broadcast, Unknown unicast and Multicast (BUM) traffic. IP multicast is an easy solution for this problem. However, not all networks support multicast. This problem can also be solved via replication node or via an SDN controller without relying on multicast. VXLAN is the only viable option in an OpenStack deployment when 4K Layer 2 segment or tenant is not enough. VXLAN also provides Layer 2 overlays for OpenStack tenants and connects OpenStack Neutron (virtual) networks to the physical world where servers and services are not virtualized.

Using an SDN Controller to Manage the VXLAN Overlay Network

Native OpenStack with Neutron supports VXLAN overlay, without an SDN Controller, via the ML2 OVS agent. However, the Neutron infrastructure is limited by:

- No hardware VTEP support
- No distributed routing or NAT support

To solve these limitations, a network controller that understands the network topology is needed. For example, set the required tunnels; determine which logical network, represented by a VNI, a physical server should be mapped and to which VTEP a specific MAC address should be forwarded. All these features are included in the SDN controller.

Hardware VTEP?

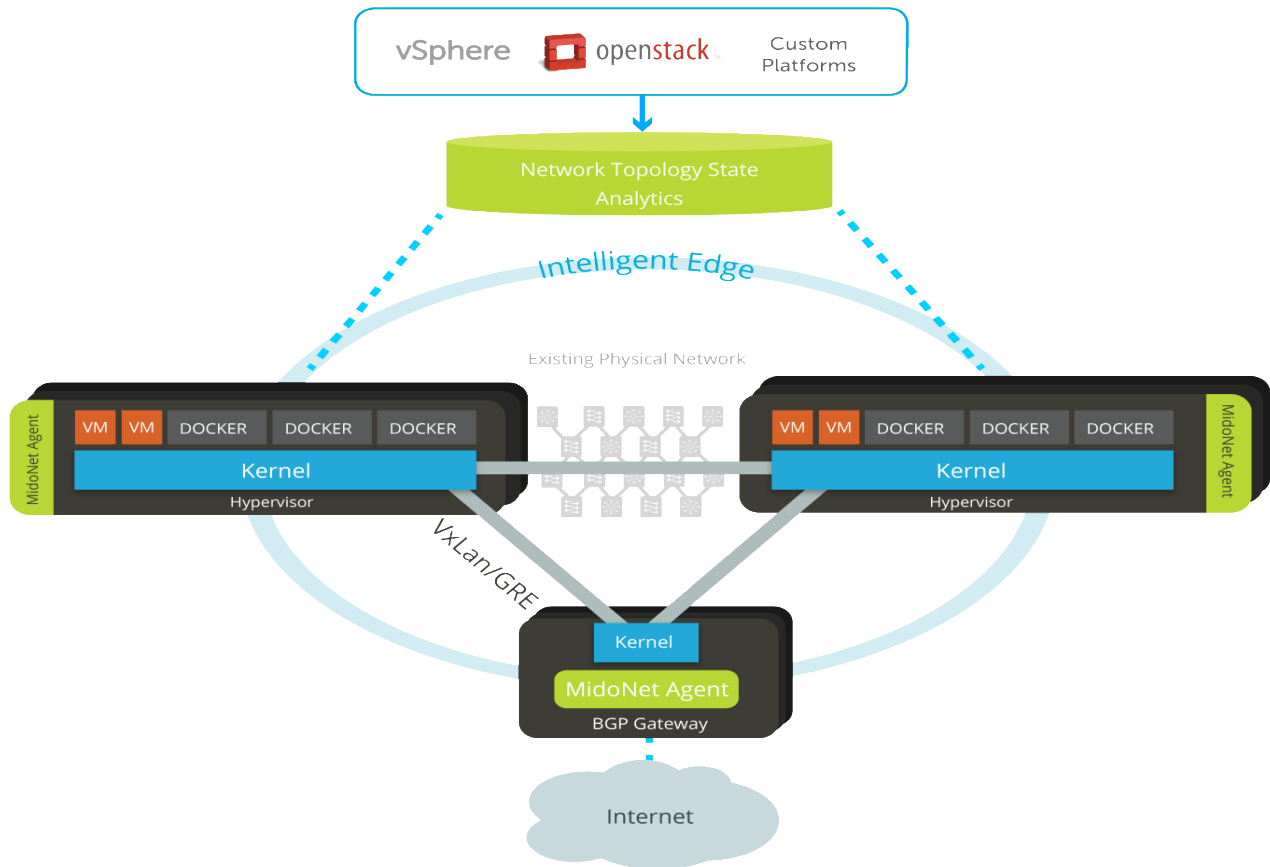
VXLAN is typically used as an overlay in top of the IP fabric infrastructure with the virtual switch/hypervisor on the host being the VXLAN VTEP. In typical data centers, however, not every machine is virtualized and uses a virtual switch. “Bare metal” servers — not virtualized or physical machines — are common in most real data centers.

The solution is to develop Hardware VTEP capacities on physical switching devices that are to be connected to virtual networks. Under the control of the SDN controller, VTEP maps physical ports and VLANs on those ports to logical networks so that any physical device can participate in a given logical network; communicating with the virtual machines that are also connected to that logical network.

Midokura MidoNet SDN Controller

Midokura MidoNet is an open, software-only, highly scalable and resilient, network virtualization system. Distributed architecture of MidoNet enables enterprises and service providers to design, deploy and manage virtual networks with scale, control, security and flexibility. It allows users to build isolated networks in software and overlays the existing network hardware infrastructure. MidoNet fully supports OpenStack Neutron. Midokura Enterprise Manager (MEM) is an advance and commercial version of MidoNet. PicOS VTEP Layer 2 Gateway works with both MidoNet and MEM. MEM is however, outside the scope of this document.

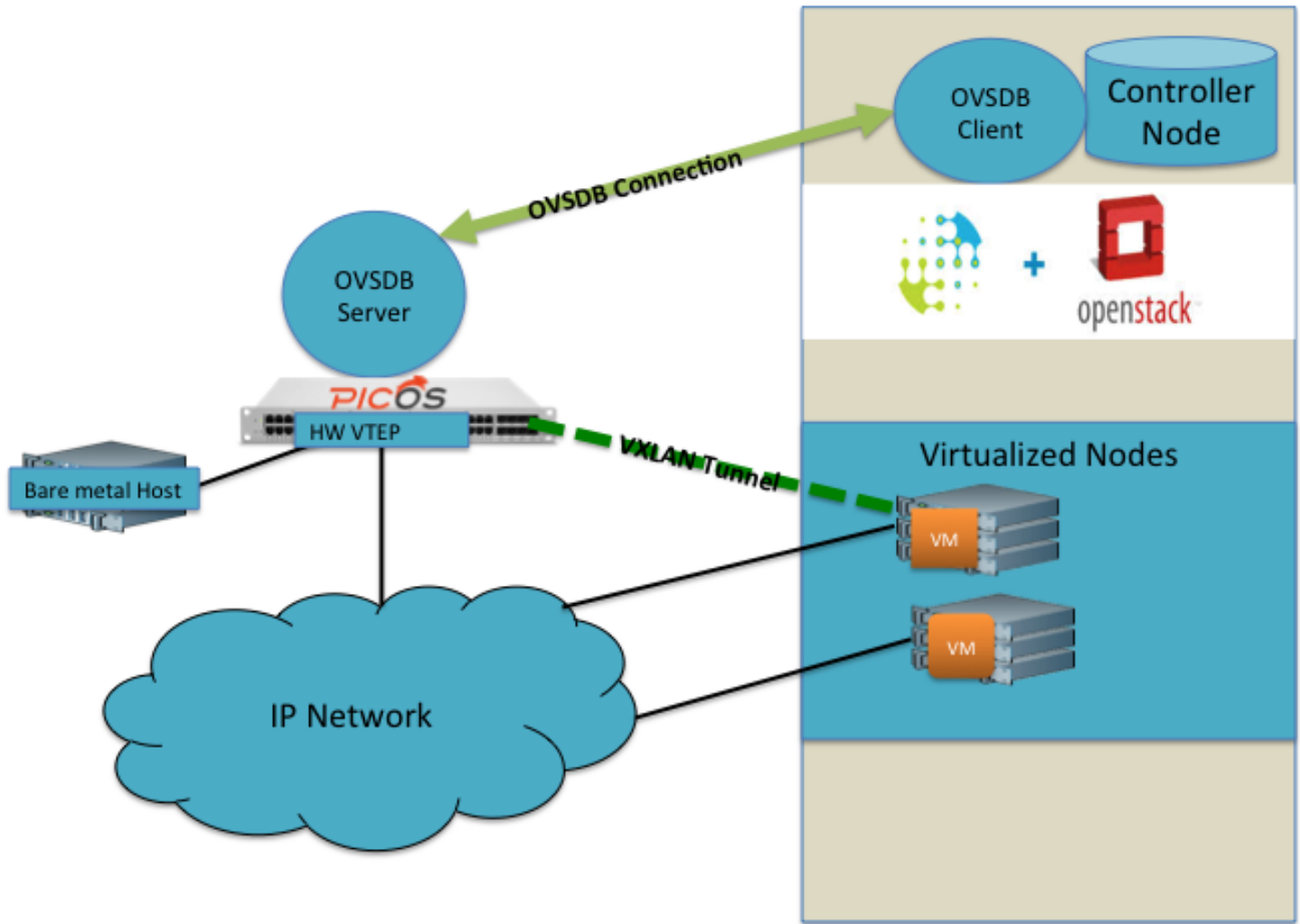
Figure 4 MidoNet



Deploying OpenStack VTEP Gateway with Pica8 and Midokura

Pica8 operating system, PicOS, smoothly integrates with the MidoNet OpenStack infrastructure and provides the VTEP gateway for terminating VXLAN tunnels from the MidoNet VTEP in OpenStack. MidoNet OVSDb client connects to the OVSDb server running on the SDN white box switch running PicOS, and exchanges information about the VTEPs and MAC addresses associated with the OpenStack Neutron networks and provides connectivity between virtual and physical infrastructures.

Figure 5 OpenStack VTEP Gateway with PicOS and MidoNet

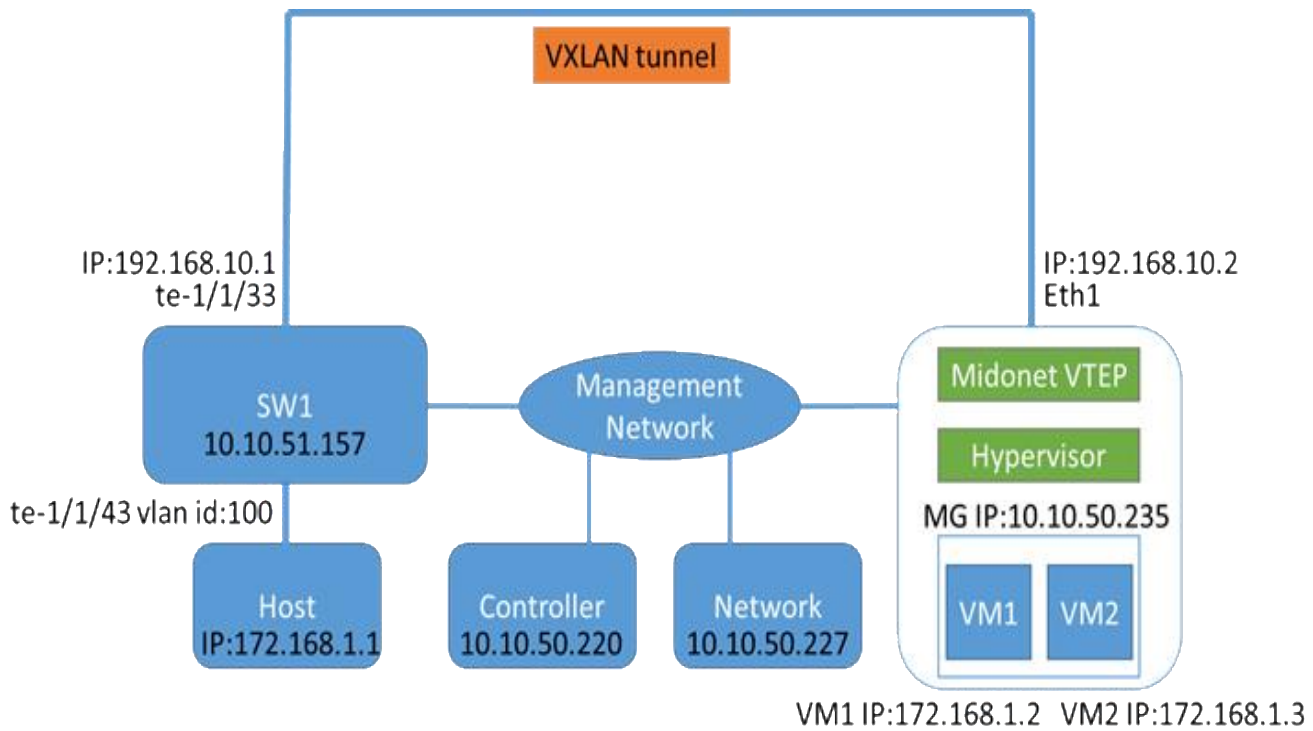


Test Bed Configuration

The test bed is based on OpenStack Juno release of Ubuntu 14.04 with a three-host design; controller, compute and network nodes are running on three virtual machines. Midokura MidoNet integrates into OpenStack and one SDN white box switch running PicOS 2.6 providing network infrastructure, in particular VTEP gateway. MidoNet serves as the OpenStack Neutron plugin. PicOS based SDN white box switch acts as a VTEP gateway and connects OpenStack Neutron network to the physical world.

This switch must support hardware based VXLAN encapsulation at line rate (like the Broadcom TD II ASIC or the Broadcom Tomahawk ASIC) to support VXLAN or VTEP gateway functionality. VM1 and VM2 represent two hosts of a tenant on 172.168.1/24 subnet. Moreover, Host represents a physical node on the 172.168.1.0/24 subnet and connects to VLAN 100 of switch port Te-1/1/43. VXLAN tunnel is established between MidoNet VTEP on hypervisor node and the switch so that VM1 and VM2 have connectivity with physical node (i.e. host). Figure 6 illustrate the setup.

Figure 6 Pica8 OpenStack VTEP Gateway Test Bed



Pica8 VTEP Gateway Configuration

Step	Command	Description
1	edit	enter configuration mode
2	set vlans vlan-id 100	configure VLAN 1000
3	set vlans vlan-id 1000 I3-interface vlan-1000	configure Layer 3 interface for VLAN 1000

Step	Command	Description
4	set interface gigabit-ethernet te-1/1/33 family ethernet-switching native-vlan-id 1000	configure VLAN 1000 as untagged VLAN
5	set interface gigabit-ethernet te-1/1/43 family ethernet-switching vlan members 1000	add VLAN 1000 on Te-1/1/43
6	set interface gigabit-ethernet te-1/1/43 family ethernet-switching port-mode "trunk"	configure Te-1/1/43 as trunk interface
7	set vlan-interface loopback address 10.10.10.1 prefix-length 32	configure ip address for the loopback interface
8	set vlan-interface interface vlan-1000 vif vlan-1000 address 192.168.10.1 prefix-length 24	configure ip address for the vlan-interface 1000
9	set vxlans source-interface vlan-1000 address 192.168.10.1	configure VTEP interface sources ip address
10	set vxlan ovssdb-managed true	enable VXLAN managed by OVSSDB
11	set protocols ovssdb management-ip 10.10.51.157	configure ovssdb management interface ip address
12	set protocols ovssdb controller c1 protocol tcp	configure ovssdb controller protocol
13	set protocols ovssdb controller ovssdb port 6632	configure ovssdb controller port
14	set protocols ovssdb interface te-1/1/43	configure OVSSDB interface

Midonet Configuration

Step	Description
1	Create a tunnel zone of type for VTEP
2	Add a VTEP to MidoNet and assign it to the 'vtep' tunnel zone
3	Create a binding between the VTEP and the Neutron network behind the MidoNet bridge
4	Add the physical host's IP address to the same tunnel zone as the VTEP
5	Create a binding between the VTEP's VLAN 100 and the Neutron network behind the bridge
6	Add the IP address of the host on the VTEP to the security group

Pica8 VTEP Gateway Configuration and Verification

```

set interface qe-interface-mode "SFP"
set interface gigabit-ethernet te-1/1/33 speed "1000"
set interface gigabit-ethernet te-1/1/33 family ethernet-switching native-vlan-id 1000
set interface gigabit-ethernet te-1/1/43 family ethernet-switching port-mode "trunk"
set interface gigabit-ethernet te-1/1/43 family ethernet-switching vlan members 100
set protocols ovssdb management-ip 10.10.51.157
set protocols ovssdb controller c1 protocol "ptcp"
set protocols ovssdb interface te-1/1/43
set vlan-interface interface 1000 vif 1000 address 192.168.10.1 prefix-length 24
set vlans vlan-id 100
set vlans vlan-id 1000 l3-interface "1000"
set vxlans source-interface 1000 address 192.168.10.1
set vxlans ovssdb-managed true
  
```

OpenStack Configurations via Horizon

1. Add a virtualization image on OpenStack Dashboard

Images



The screenshot shows the 'Images' page in the OpenStack Horizon dashboard. It features a table with columns for Image Name, Type, Status, Public, Protected, Format, Size, and Actions. A single image named 'test' is listed with a 'Launch' button. Above the table are filters for Project (1), Shared with Me (0), and Public (0), along with buttons for 'Create Image' and 'Delete Images'.

Image Name	Type	Status	Public	Protected	Format	Size	Actions
test	Image	Active	No	No	QCOW2	12.5 MB	Launch

2. Create Network on OpenStack Dashboard

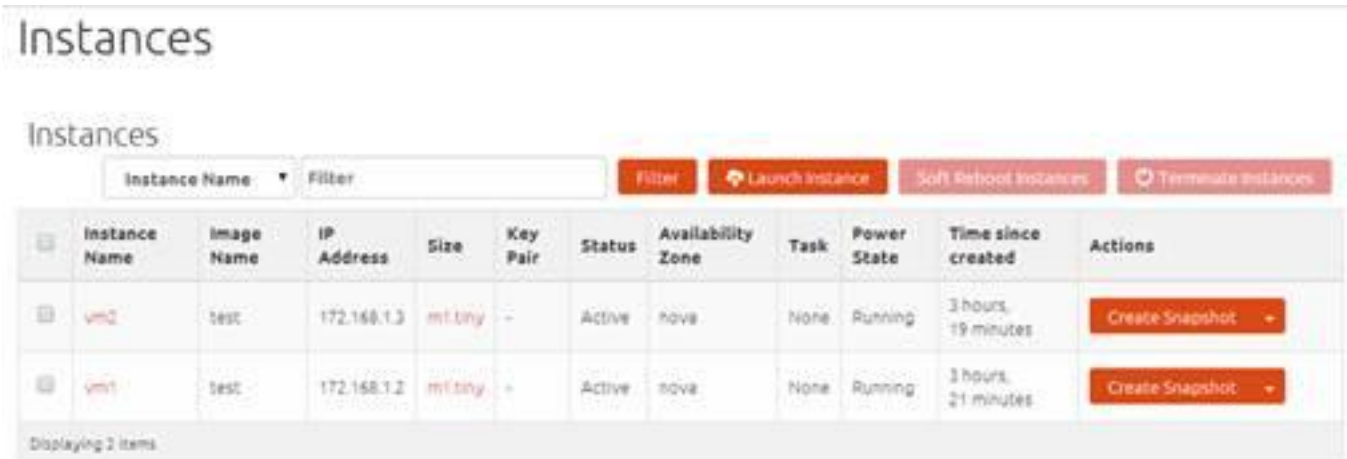
Networks



The screenshot shows the 'Networks' page in the OpenStack Horizon dashboard. It features a table with columns for Project, Network Name, Subnets Associated, DHCP Agents, Shared, Status, Admin State, and Actions. A network named 'vxlan' is listed with an 'Edit Network' button. Above the table are buttons for 'Create Network' and 'Delete Networks'.

Project	Network Name	Subnets Associated	DHCP Agents	Shared	Status	Admin State	Actions
admin	vxlan	m1 172.168.1.0/24	1	No	ACTIVE	UP	Edit Network

3. Create two VMs and assign Network on OpenStack Dashboard



The screenshot shows the OpenStack Instances dashboard. At the top, there are buttons for 'Filter', 'Launch Instance', 'Soft Reboot Instances', and 'Terminate Instances'. Below these is a table with the following columns: Instance Name, Image Name, IP Address, Size, Key Pair, Status, Availability Zone, Task, Power State, Time since created, and Actions. Two instances are listed:

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
vm2	test	172.168.1.3	m1.tiny	-	Active	nova	None	Running	3 hours, 19 minutes	Create Snapshot
vm1	test	172.168.1.2	m1.tiny	-	Active	nova	None	Running	3 hours, 21 minutes	Create Snapshot

Displaying 2 items

Midonet Configuration

1. Create a tunnel zone for VTEP

```
midonet> tunnel-zone create name vtep_zone1 type vtep
tzone0
```

2. Add a VTEP to MidoNet and assign it to the 'VTEP' tunnel zone

```
midonet> vtep add management-ip 10.10.51.157 management-port 6632 tunnel-zone tzone0
name br0 description OVS VTEP Emulator management-ip 10.10.51.157 management-port 6632
tunnel-zone tzone0 connection-state CONNECTED
midonet> list vtep
name br0 description management-ip 10.10.51.157 management-port 6632 tunnel-zone tzone0
connection-state CONNECTED
```

3. Create a binding between the VTEP and Neutron network behind the MidoNet Bridge

```
midonet> bridge list
bridge bridg1 name vxlan state up
midonet> show bridge bridg1 id
85296f07-2235-4963-8160-fb66eca85675
midonet>
```

4. Add Physical host's IP address to the same tunnel zone as the VTEP

```
midonet> tunnel-zone tzone0 add member host host0 address 192.168.10.2
zone tzone0 host host0 address 192.168.10.2
midonet>
```

5. Create a binding between the VTEP's VLAN 100 interface te-1/1/43 and the Neutron network behind the bridge1

```
midonet> vtep management-ip 10.10.51.157 binding add network-id 85296f07-2235-4963-8160-
fb66eca85675 physical-port te-1/1/43 vlan 100
Internal error: The server could not comply with the request since it is either malformed or
otherwise incorrect.
midonet> vtep management-ip 10.10.51.157 binding list
management-ip 10.10.51.157 physical-port te-1/1/43 vlan 100 network-id 85296f07-2235-4963-
8160-fb66eca85675
```

6. Add the IP address of the host on the VTEP to the security group ip-address-group0

```
midonet> ip-address-group ip-address-group0 add ip address 172.168.1.1
address 172.168.1.1
midonet>
```

Verifying the Pica8 Switch

Verifying VXLAN table of SW1:

```
admin@XorPlus# run show vxlan
Egress map:
    egress_id 100009 MAC 0:c:29:23:31:9, port_id 1/1/33, vif_index 8 unicast
L3 tunnel mac map:
    vlan id 1000, ref_count 1
Port vlan map mode map & Termination admin state map:
    port id 1/1/43, ref_count 1
Tunnel Map:
    tunnel id 0X4C000200, dst_vtep 192.168.10.2, nexthops (192.168.10.2 ), ecmp_id
100009, ref_count 1
    tunnel id 0X4C000001, dst_vtep 224.0.0.1, nexthops (), ecmp_id 0, ref_count 1
Access ports:
    id 0X80000002, vpn_id 0X7000, port_id 1/1/43, vlan_id 100, egress_id 100010
Network ports:
    id 0X80000003, vpn_id 0X7000, port_id 1/1/33, egress_id 100009, tunnel_id
0X4C000200, unicast
    id 0X80000004, vpn_id 0X7000, port_id 1/1/33, egress_id 100011, tunnel_id
0X4C000200, multicast
    id 0X80000001, vpn_id 0XFFFFFFF, port_id 1/1/0, egress_id 100006, tunnel_id
0X4C000001, multicast
BFD sessions:
admin@XorPlus#
```

Note: The **show vxlan** command displays information about VXLAN endpoint configuration and next-hops of the corresponding remote VTEP.

Verify VXLAN MAC Table of SW1

```

admin@XorPlus# run show vxlan address-table
VNID          MAC address          Type          Interface          VTEP
-----          -
10001         00:1e:c9:bb:bb:ce    Dynamic       te-1/1/43
10001         fa:16:3e:00:0c:f3    Static
10001         fa:16:3e:28:aa:cd    Static
admin@XorPlus#

```

Dump the OVSDB Hardware VTEP Table of the Pica8 Switch

```

root@XorPlus$ovsdb-client dump hardware_vtep
Arp_Sources_Local table
_uuid locator src_mac
-----
Arp_Sources_Remote table
_uuid locator src_mac
-----

Global table
_uuid          managers          switches
-----
4146166b-ad2e-4d05-857f-8ba4b3f0ac0d [bd6ac790-b304-4ed7-a77b-8ab7063b8132] [cfdcc9fa-0295-44b0-81c3-c975b3d463cb]

Logical_Binding_Stats table
_uuid bytes_from_local bytes_to_local packets_from_local packets_to_local
-----

Logical_Router table
_uuid description name static_routes switch_binding
-----

Logical_Switch table
_uuid          description name          options
tunnel_key
-----
e1e37b4a-37fe-43f2-a9f7-3a9925b6e92e ""          "mn-85296f07-2235-4963-8160-fb66eca85675" {}
10001

Manager table
_uuid          inactivity_probe is_connected max_backoff other_config status
target
-----
bd6ac790-b304-4ed7-a77b-8ab7063b8132 30000          true          3000          {}
{bound_port="6632", sec_since_connect="13921", state=ACTIVE} "ptcp:6632"

Mcast_Macs_Local table
MAC _uuid ipaddr locator_set logical_switch
-----

```

```

Mcast_Macs_Remote table
MAC          _uuid          ipaddr locator_set
logical_switch
-----
unknown-dst 8c6b4993-7be8-4d85-811b-3255e15d2f92 ""      6d275247-2c1b-4c79-8f08-b17d93bd1e32
e1e37b4a-37fe-43f2-a9f7-3a9925b6e92e

Physical_Locator table
_uuid          dst_ip          encapsulation_type
-----
d983943f-c791-4431-89a2-ec6a531a4d15 "192.168.10.1" "vxlan_over_ipv4"
09c0f3c2-d42a-406b-8644-3bffc472a247 "192.168.10.2" "vxlan_over_ipv4"

Physical_Locator_Set table
_uuid          locators
-----
6d275247-2c1b-4c79-8f08-b17d93bd1e32 [09c0f3c2-d42a-406b-8644-3bffc472a247]

Physical_Port table
_uuid          description name          port_fault_status vlan_bindings
vlan_stats
-----
35f008a2-e248-4330-a1e5-85f3f843bc68 ""          "te-1/1/43" []          {100=e1e37b4a-37fe-
43f2-a9f7-3a9925b6e92e} {}

Physical_Switch table
_uuid          description management_ips  name  ports
switch_fault_status tunnel_ips          tunnels
-----
cfdcc9fa-0295-44b0-81c3-c975b3d463cb ""          ["10.10.51.157"] "br0" [35f008a2-e248-4330-a1e5-
85f3f843bc68, 3d5eae61-46bc-4e3c-84f3-06aed7961ff5, c30d1ef4-b54a-4946-bd8c-460af234875e] []
["192.168.10.1"] []

SSL table
_uuid bootstrap_ca_cert ca_cert certificate external_ids private_key
-----

Tunnel table
_uuid bfd_config_local bfd_config_remote bfd_params bfd_status local remote
-----

Ucast_Macs_Local table
MAC          _uuid          ipaddr locator
logical_switch
-----
"00:1e:c9:bb:bb:ce" f431e446-6c1c-4842-ad3a-19cd04a54952 ""      d983943f-c791-4431-89a2-
ec6a531a4d15 e1e37b4a-37fe-43f2-a9f7-3a9925b6e92e

```

```
Ucast_Macs_Remote table
MAC          _uuid          ipaddr locator
logical_switch
-----
"fa:16:3e:00:0c:f3" 0c6732d3-9e72-4444-9d8f-07abde993aa7 "" 09c0f3c2-d42a-406b-8644-3bffc472a247 e1e37b4a-37fe-43f2-a9f7-3a9925b6e92e
"fa:16:3e:28:aa:cd" b1160e14-8e35-4293-a987-59ddc29f7304 "" 09c0f3c2-d42a-406b-8644-3bffc472a247 e1e37b4a-37fe-43f2-a9f7-3a9925b6e92e
```

Verify Connectivity between Virtual Machine and Physical Host

Ping VM1 and VM2 from the Host:

```
root@Dev-45:~# ping 172.168.1.2 -c 5
PING 172.168.1.2 (172.168.1.2) 56(84) bytes of data.
64 bytes from 172.168.1.2: icmp_req=1 ttl=64 time=3.92 ms
64 bytes from 172.168.1.2: icmp_req=2 ttl=64 time=1.51 ms
64 bytes from 172.168.1.2: icmp_req=3 ttl=64 time=1.47 ms
64 bytes from 172.168.1.2: icmp_req=4 ttl=64 time=1.59 ms
64 bytes from 172.168.1.2: icmp_req=5 ttl=64 time=1.57 ms

--- 172.168.1.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 1.476/2.015/3.920/0.954 ms
root@Dev-45:~# ping 172.168.1.3 -c 5
PING 172.168.1.3 (172.168.1.3) 56(84) bytes of data.
64 bytes from 172.168.1.3: icmp_req=1 ttl=64 time=10.1 ms
64 bytes from 172.168.1.3: icmp_req=2 ttl=64 time=1.70 ms
64 bytes from 172.168.1.3: icmp_req=3 ttl=64 time=1.64 ms
64 bytes from 172.168.1.3: icmp_req=4 ttl=64 time=1.62 ms
64 bytes from 172.168.1.3: icmp_req=5 ttl=64 time=1.67 ms

--- 172.168.1.3 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 1.623/3.365/10.189/3.412 ms
root@Dev-45:~# arp -n
Address          HWtype  HWaddress          Flags Mask          Iface
172.168.1.3      ether    fa:16:3e:28:aa:cd  C                   eth1.100
```