VXLAN VTEP Gateway with Pica8 and VMware NSX

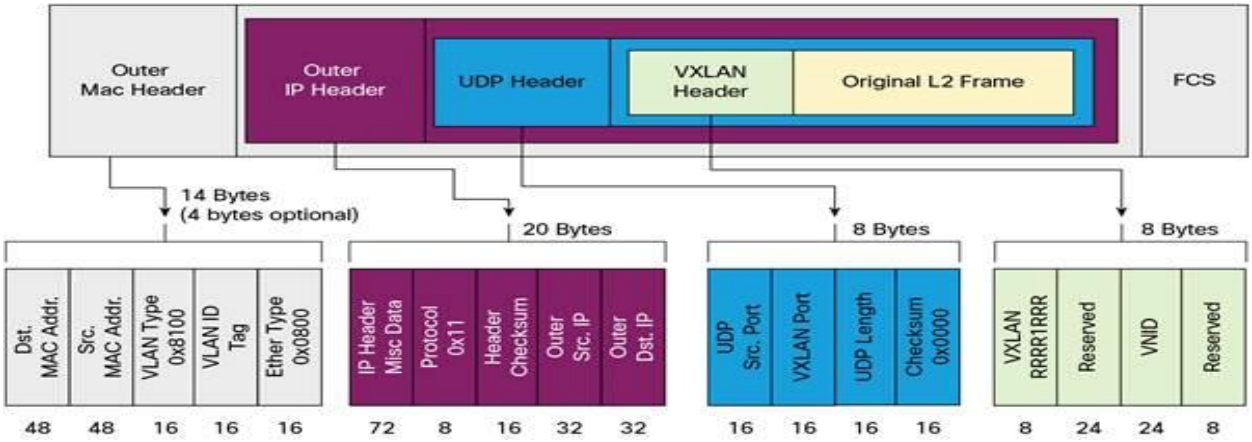
This paper outlines how Pica8's OS (PicOS) running on a White Box SDN switch is used as a VXLAN Layer 2 Tunnel End Point (VTEP) to interconnect physical and virtual networks by leveraging VMware NSX platform in both OpenStack and non-OpenStack deployment.

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. One of the common use cases of VXLAN are VM mobility across Layer 3, integration with OpenStack, and gateway functionality between physical and logical networks.

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 For*mat*

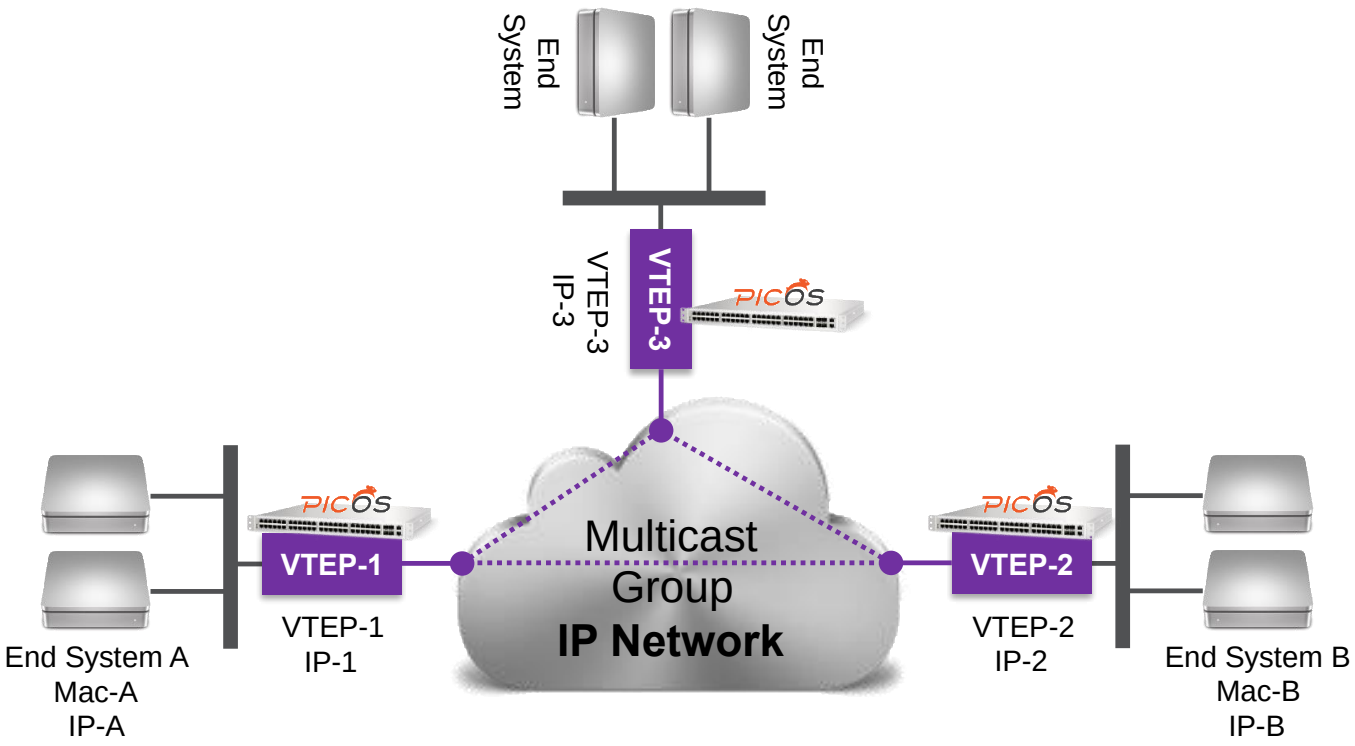


VXLAN header is 8-bytes in length and consists of a 24-bit VXLAN network identifier (VNID) providing up to 16M Layer 2 segments.

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.

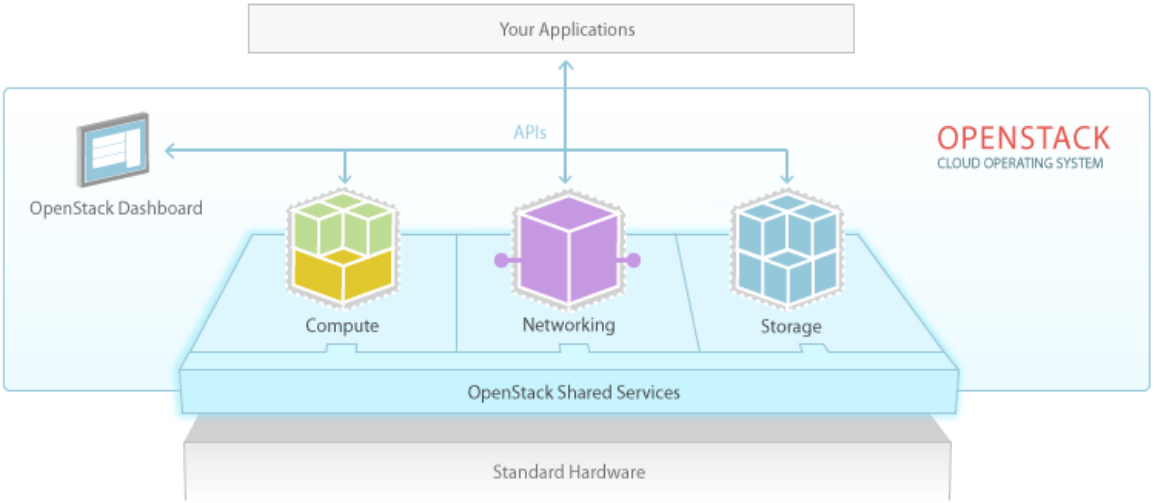
Figure 2 VTEP



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 segments or tenant is not adequate. 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

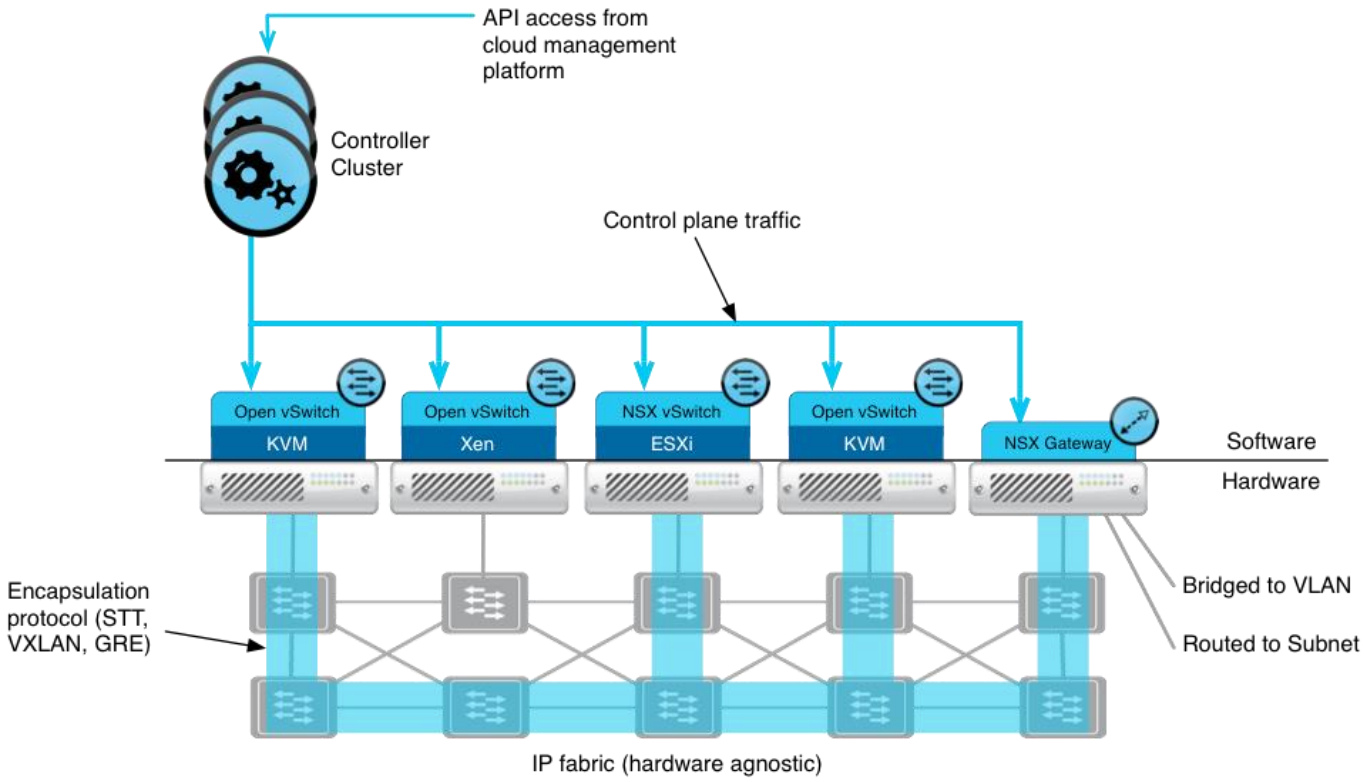
VXLAN is typically used as an overlay on top of the IP fabric infrastructure with the virtual switch/hypervisor on the host being the VXLAN VTEP. In typical data centers, however, not every machine is virtualized and uses a virtual switch. “Bare metal” servers — not virtualized or physical machines —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

VMware NSX SDN Controller

VMware NSX is a network virtualization platform providing network virtualization. In NSX, virtual networks are programmatically provisioned and managed independent of underlying networking hardware similar to VM in computing. NSX reproduces the entire software network environment and enables diverse network topology creation and provisioning within seconds.

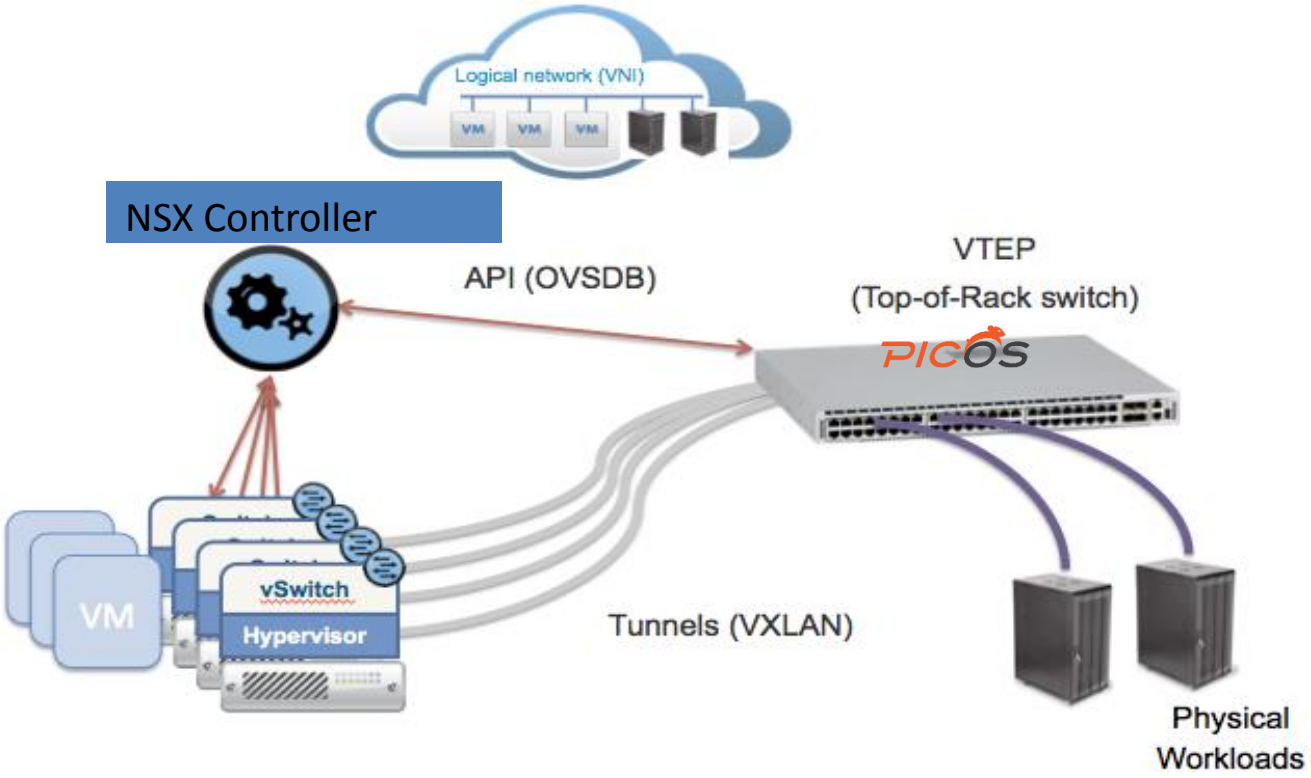
Figure 4 VMware NSX



Deploying VTEP Gateway with Pica8 and VMware NSX

Pica8 operating system, PicOS, smoothly integrates with the NSX infrastructure and provides the VTEP gateway for terminating VXLAN tunnels from the NSX controller. PicOS running on the SDN white box switch acts as a VTEP gateway and connects to NSX controller via OVSDB. As a result, VMware NSX controller becomes the central point of management and control functions and provides seamless connectivity between virtual and physical worlds.

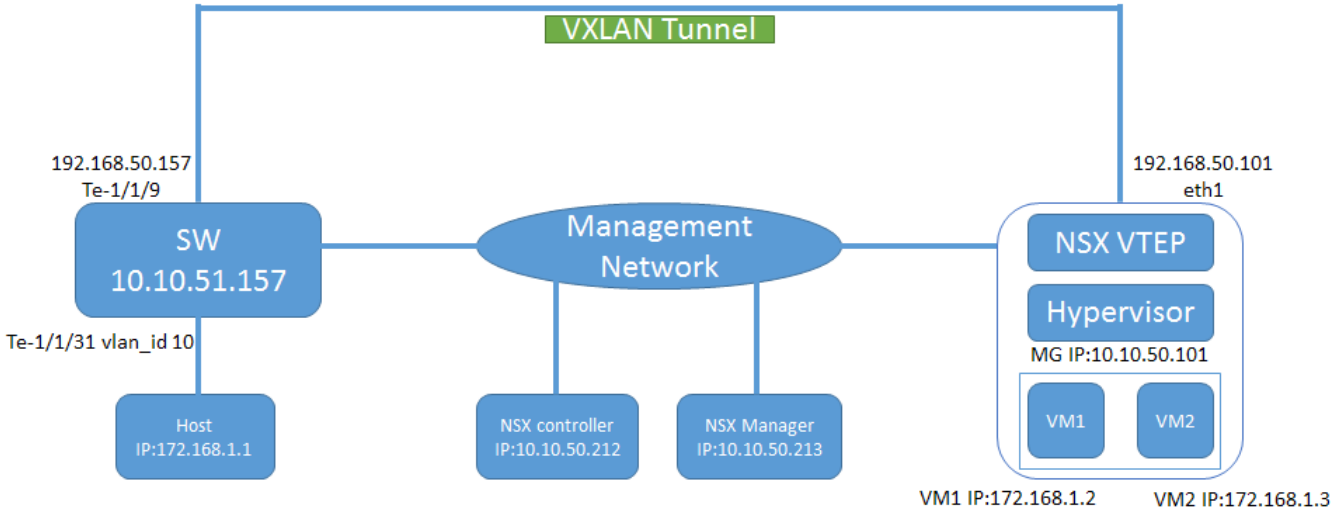
Figure 5 PicOS VTEP Gateway Integration with VMware NSX



Test Bed Configuration

The test bed consists of VMware NSX platforms and one SDN white box running PicOS as the VTEP gateway. PicOS VTEP gateway integrates into the NSX Controller via OVSDB. 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 hosts of a tenant on 172.168.1/024 subnet. Moreover, Host represents a physical node on the 172.168.1.0/24 subnet and connects to VLAN 100 of switch port Te-1/1/41. VXLAN tunnel is established between NSX VTEP on hypervisor node and the Pica8 switch (i.e. SW) so that VM1 and VM2 have connectivity with physical node (i.e. host). Figure 6 illustrates the setup.

Figure 6 Pica8 NSX VTEP Gateway Test Bed



Pica8 VTEP Gateway Configuration

Step	Command	Description
1	edit	enter configuration mode
2	set vlans vlan-id 1000	configure VLAN 1000
3	set vlans vlan-id 1000 l3-interface vlan-1000	configure Layer 3 interface for VLAN 1000
4	set interface gigabit-ethernet te-1/1/9 family ethernet-switching native-vlan-id 1000	configure VLAN 1000 as untagged VLAN
5	set interface gigabit-ethernet te-1/1/31 family ethernet-switching vlan members 100	add VLAN 1000 on Te-1/1/31
6	set interface gigabit-ethernet te-1/1/31 family ethernet-switching port-mode "trunk"	configure Te-1/1/31 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 192.168.50.157 prefix-length 24	configure IP address for the vlan-interface 1000
9	set vxlans source-interface vlan-1000 address 192.168.50.157	configure VTEP interface source IP address

Step	Command	Description
10	set vxlan ovsdb-managed true	enable VXLAN managed by OVSDB
11	set protocols ovsdb management-ip 10.10.51.157	configure OVSDB management interface IP address
12	set protocols ovsdb controller address 10.10.50.212	configure OVSDB controller IP address
13	set protocols ovsdb controller c1 protocol ssl	configure OVSDB controller protocol
14	set protocols ovsdb controller ovsdb port 6632	configure OVSDB controller port
15	set protocols ovsdb interface te-1/1/31	configure OVSDB interface

VMware NSX Configuration

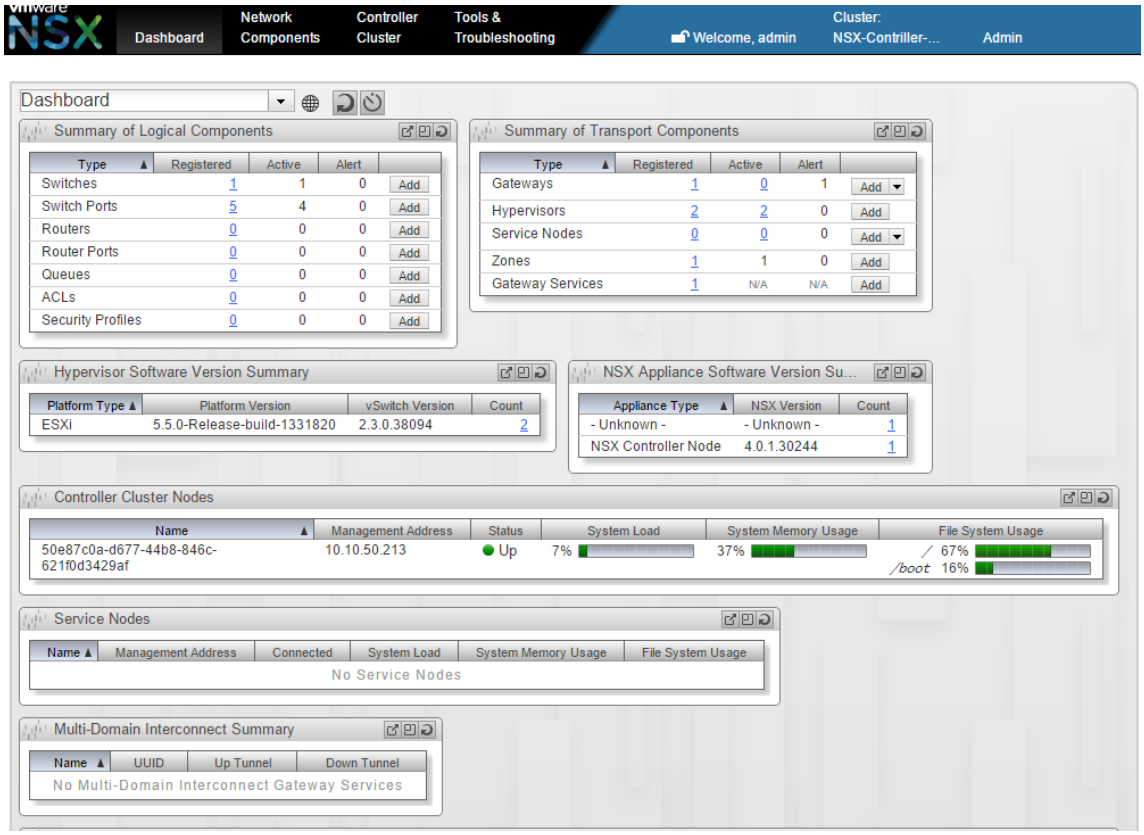
Step	Description
1	Configure OVSDB on NSX Manager for SW1
2	Add Components in NSX Manager
3	Add ESXi in NSX Manager
4	Add Hardware Gateway
5	Create Transport Zone
6	Create Logical Switch
7	Create Logical Port
8	Add Gateway VTEP Port

Pica8 VTEP Gateway Configuration and Verification

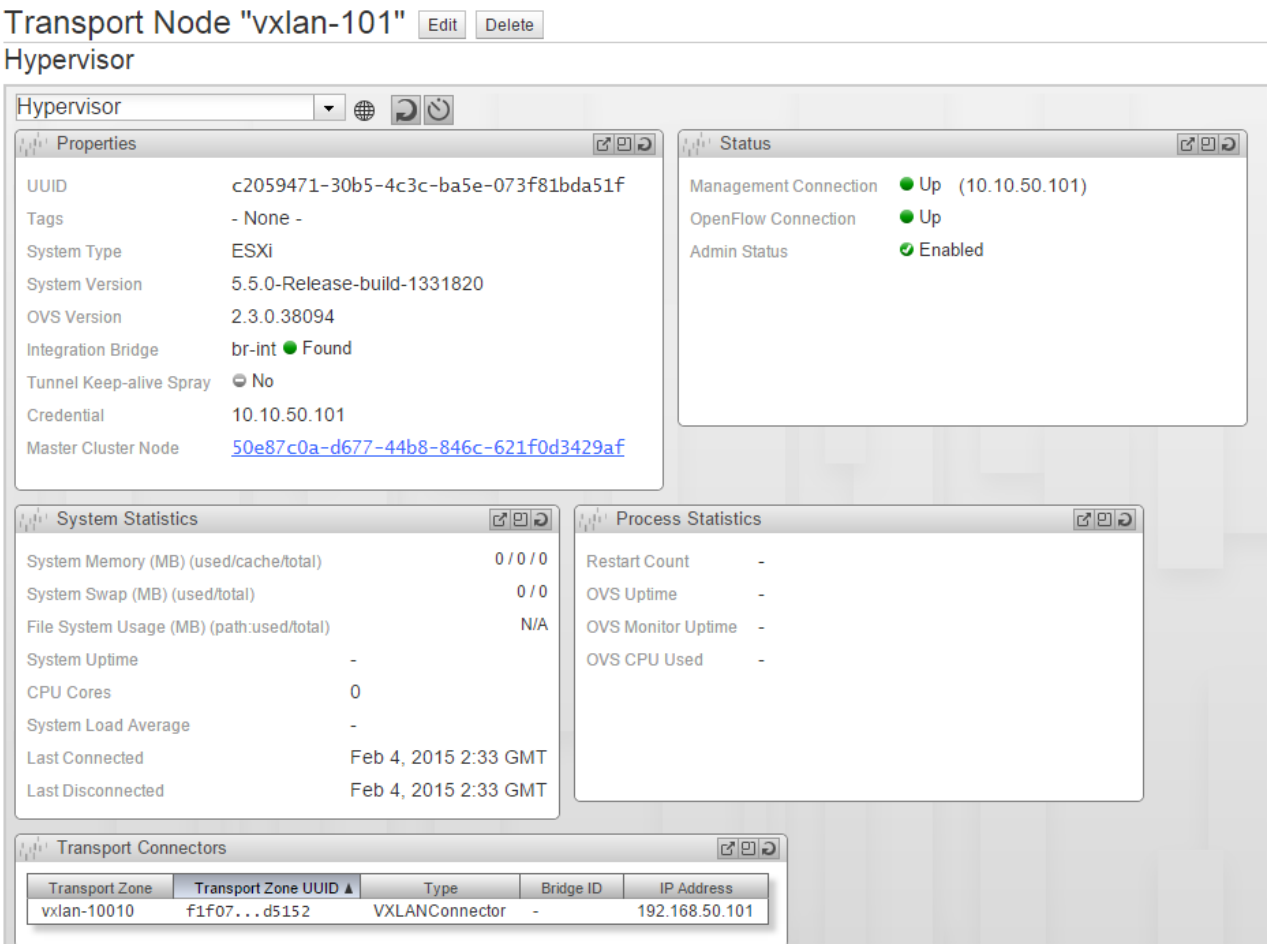
```
set interface gigabit-ethernet te-1/1/9 family ethernet-switching native-vlan-id 1000
set interface gigabit-ethernet te-1/1/31 family ethernet-switching port-mode "trunk"
set interface gigabit-ethernet te-1/1/31 family ethernet-switching vlan members 100
set protocols ovsdb management-ip 10.10.51.157
set protocols ovsdb controller ovsdb protocol ssl
set protocols ovsdb controller ovsdb address 10.10.50.212
set protocols ovsdb controller ovsdb port 6632
set protocols ovsdb interface te-1/1/31
set vlan-interface interface 1000 vif 1000 address 192.168.50.157 prefix-length 24
set vlans vlan-id 100
set vlans vlan-id 1000 l3-interface "1000"
set vxlans source-interface 1000 address 192.168.50.157
set vxlans ovsdb-managed true
```

VMware Configuration via NSX Dashboard

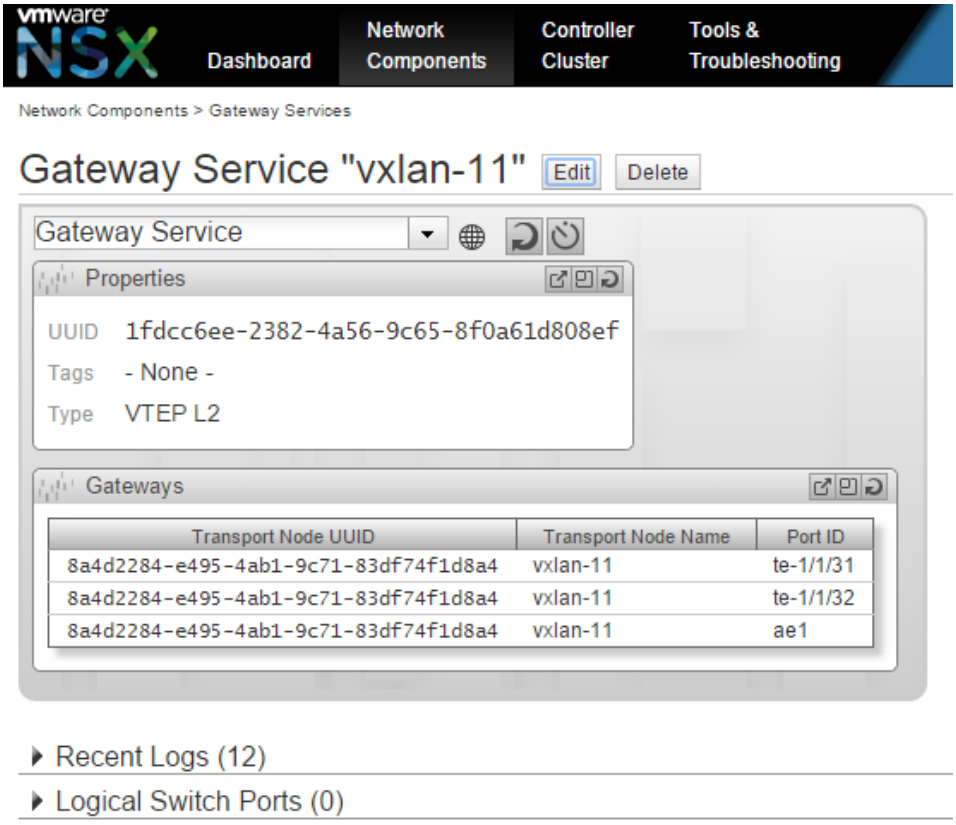
1. Configure OVSDB Manager for SW1 in NSX Manager
2. Add Components in NSX Manager



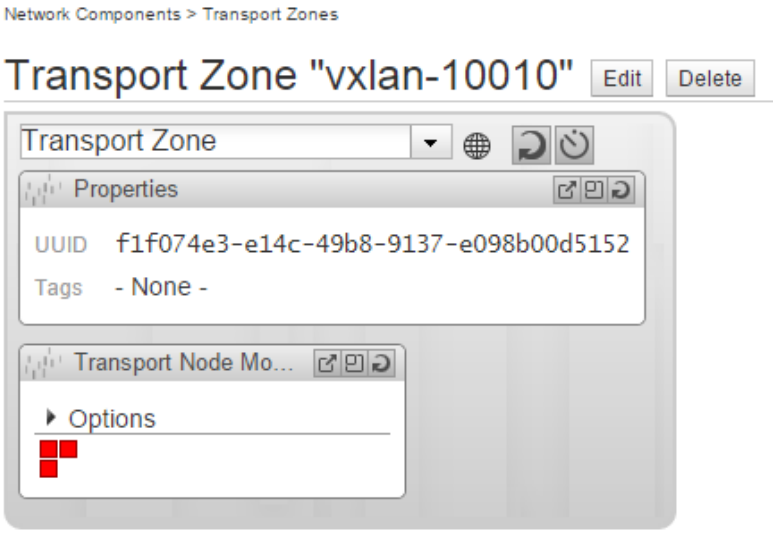
3. Add ESXi in NSX Manager



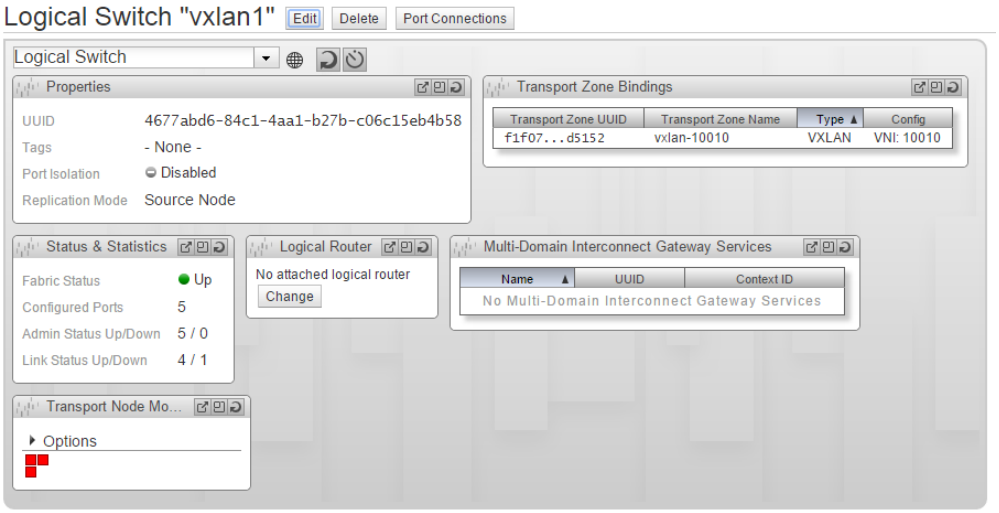
4. Add Hardware Gateway



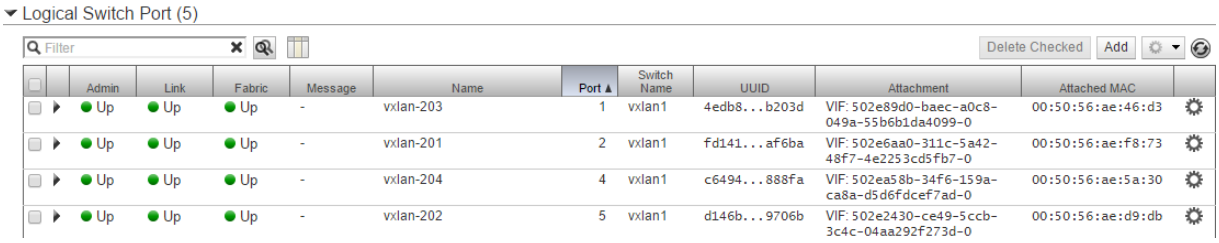
5. Create Transport Zone



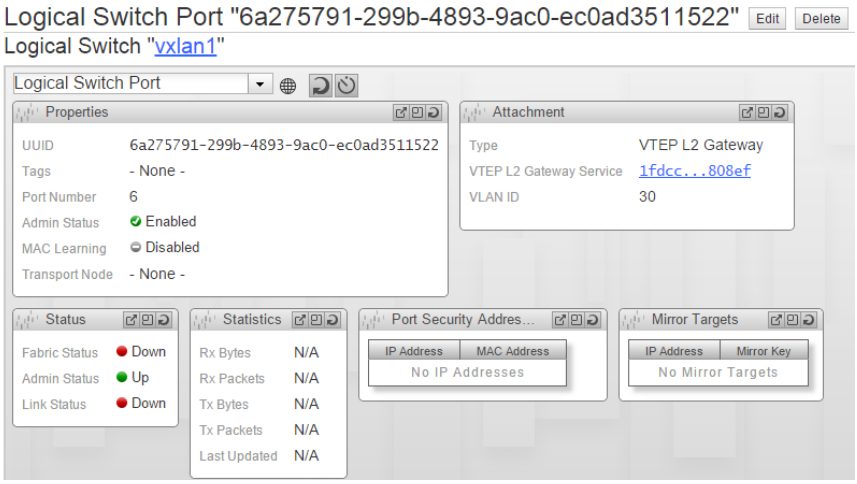
6. Create Logical Switch



7. Create Logical Port



8. Add Gateway VTEP Port



Verifying the Pica8 Switch

Verifying VXLAN Table of SW1

```
admin@XorPlus# run show vxlan tunnel
Total number of tunnels: 1
VNI 2, Encap:service-vlan-delete, Decap:service-vlan-add
src addr:192.168.50.157, dst addr:192.168.50.101, state:UP
traffic type:all
nexthops:192.168.50.101
output ports:te-1/1/9
```

Verifying VXLAN MAC Table of SW1

```
admin@XorPlus# run show vxlan address-table
VNID MAC address Type Interface
VTEP
----------- ----------------- ------- ----------------
---------------
10010 70:72:cf:9d:6f:fb Dynamic te-1/1/31
10010 00:50:56:ae:46:d3 Static
192.168.50.101
10010 00:50:56:ae:5a:30 Static
192.168.50.101
admin@XorPlus#
```

Dump the OVSDB Hardware VTEP Table SW1

```
root@XorPlus$ovsdb-client dump hardware_vtepArp_Sources_Local table
_uuid locator src_mac
----- ------- -------
Arp_Sources_Remote table
_uuid locator src_mac
----- ------- -------
Global table
_uuid managers switches
------------------------------------
3542b066-e49a-4df6-91e5-731ce43868c7
[5cc533b6-3591-4d5f-9848-52a9d8b38cab]
[7eea2f69-e31e-4a71-a997-3a370bd7f468]
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 tunnel_key
------------------------------------ -----------
be9b786a-d5ab-4f78-8cf7-e15e2c336994 ""
"4677abd6-84c1-4aa1-b27b-c06c15eb4b58" 10010
2e70752c-a135-4a94-ab1b-057d51eded0d "" _nvp_internal []
```

```
Manager table
_uuid inactivity_probe is_connected max_backoff other_config
status target
------------------------------------ ---------------- ------------
5cc533b6-3591-4d5f-9848-52a9d8b38cab 30000 false 3000 {} {}
"ssl:10.10.50.212:6632"
Mcast_Macs_Local table
MAC _uuid ipaddr locator_set logical_switch
--- ----- ------ ----------- --------------
Mcast_Macs_Remote table
MAC _uuid ipaddr locator_set logical_switch
--- ----- ------ ----------- --------------
------------------
e4e82b72-bfb8-4534-8e67-990d92b2e104 "192.168.50.101"
"vxlan_over_ipv4"
d69f550e-c90d-4327-a294-465d734a595c "192.168.50.157"
"vxlan_over_ipv4"
acb5aea7-db90-436e-a180-93561249c74c "192.168.50.243"
"vxlan_over_ipv4"
Physical_Locator_Set table
_uuid locators
----- --------
Physical_Port table
_uuid description name port_fault_status vlan_bindings vlan_stats
------------------------------------ ----------- -----------
c012b9a0-d840-4de2-9599-499ac27929a0 "" "te-1/1/31" []
{100=be9b786a-d5ab-4f78-8cf7-e15e2c336994} {}
Physical_Switch table
_uuid description management_ips name ports switch_fault_status
tunnel_ips tunnels
------------------------------------ ----------- --------------
7eea2f69-e31e-4a71-a997-3a370bd7f468 "" [] "br0"
[95d3ce46-6ddd-4897-acae-9417938fc463,
c012b9a0-d840-4de2-9599-499ac27929a0] [] ["192.168.50.101"] []
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
------------------- ------------------------------------ ------
"70:72:cf:9d:6f:fb" 50f22ba9-3055-4f68-803b-43bdcc722ae6 ""
d69f550e-c90d-4327-a294-465d734a595c
be9b786a-d5ab-4f78-8cf7-e15e2c336994
Ucast_Macs_Remote table
MAC _uuid ipaddr locator logical_switch
------------------- ------------------------------------ ------
"00:50:56:ae:46:d3" 4e8261ea-5b2a-49a3-af28-f5e302fde888 ""
e4e82b72-bfb8-4534-8e67-990d92b2e104
be9b786a-d5ab-4f78-8cf7-e15e2c336994
"00:50:56:ae:5a:30" 6178ea7c-3f90-47fb-bf3f-a7a2f32b0696 ""
e4e82b72-bfb8-4534-8e67-990d92b2e104
be9b786a-d5ab-4f78-8cf7-e15e2c336994
```