



Installation Runbook for Palo Alto Networks virtual Firewall with Nuage Networks VSP as SDN

Application Type	Virtual Firewall
Application Version	PAN-OS 7.x based image
Mirantis OpenStack Version	7.0
OpenStack Version	2015.1 Kilo
SDN solution	Nuage Networks VSP
SDN version	3.2.R4

Table of Contents

Document History	3
1 Introduction.....	4
1.1 Target Audience	4
2 Solution overview	5
2.1 Nuage Networks VSP Overview	5
2.1.1 Virtualized Services Directory (VSD).....	5
2.1.2 Virtualized Services Controller (VSC)	6
2.1.3 Virtual Routing and Switching (VRS).....	6
2.1.4 Virtual Routing Services Gateway (VRS-G)	6
2.1.5 Nuage Networks OpenStack Neutron Plugin (Kilo)	6
2.1.6 Service Insertion in Nuage Networks VSP	7
3 Reference Architecture	9
4 Physical & Logical Network Topology.....	10
5 Installation & Configuration.....	11
5.1 Environment Preparation.....	11
5.2 MOS installation	12
5.2.1 Health Check Results	14
5.3 Nuage Networks VSP Installation	16
5.3.1 Pre-installation Setup	16
5.3.2 VSD Installation.....	17
5.3.3 VSC Installation.....	23
5.3.4 Neutron Plugin installation.....	28
5.3.5 VRS Installation.....	35
5.3.6 VRS-G Installation.....	39
5.3.7 Installation & Setup Validation.....	46
5.4 PAN vFW Setup & Configuration	47
5.4.1 Deploying the PAN vFW	48
5.4.2 Nuage Networks VSP insertion of PAN vFW	51
5.4.3 Establishing Nuage Networks VSP «sync» with PANW vFW	56
5.4.4 PAN vFW setup.....	57
5.5 Testing.....	61
5.5.1 Test Tools.....	61
5.5.2 Target use case(s)	61
5.5.3 Test cases	62
5.5.4 Test Results	63
Appendix A: OpenStack Network Information (Horizon)	68
Appendix B: Nuage Networks VSD VM xml.....	70
Appendix C: Nuage Networks VSC VM xml.....	72
Appendix D: Nuage Networks VRS-G VM xml.....	74
Appendix E: Additional PAN vFW Initial Setup Commands	76
References:	77

Document History

Version	Revision Date	Description
0.1	14-12-2015	Initial Version
2.1	13-02-2016	Published

1 Introduction

This document is to serve as a detailed Deployment Guide for Palo-Alto Networks Virtual Firewall deployed within Mirantis OpenStack with Nuage Networks VSP. This document describes the reference architecture, installation steps for certified MOS+Nuage VSP+PAN vFW, limitations and testing procedures.

1.1 Target Audience

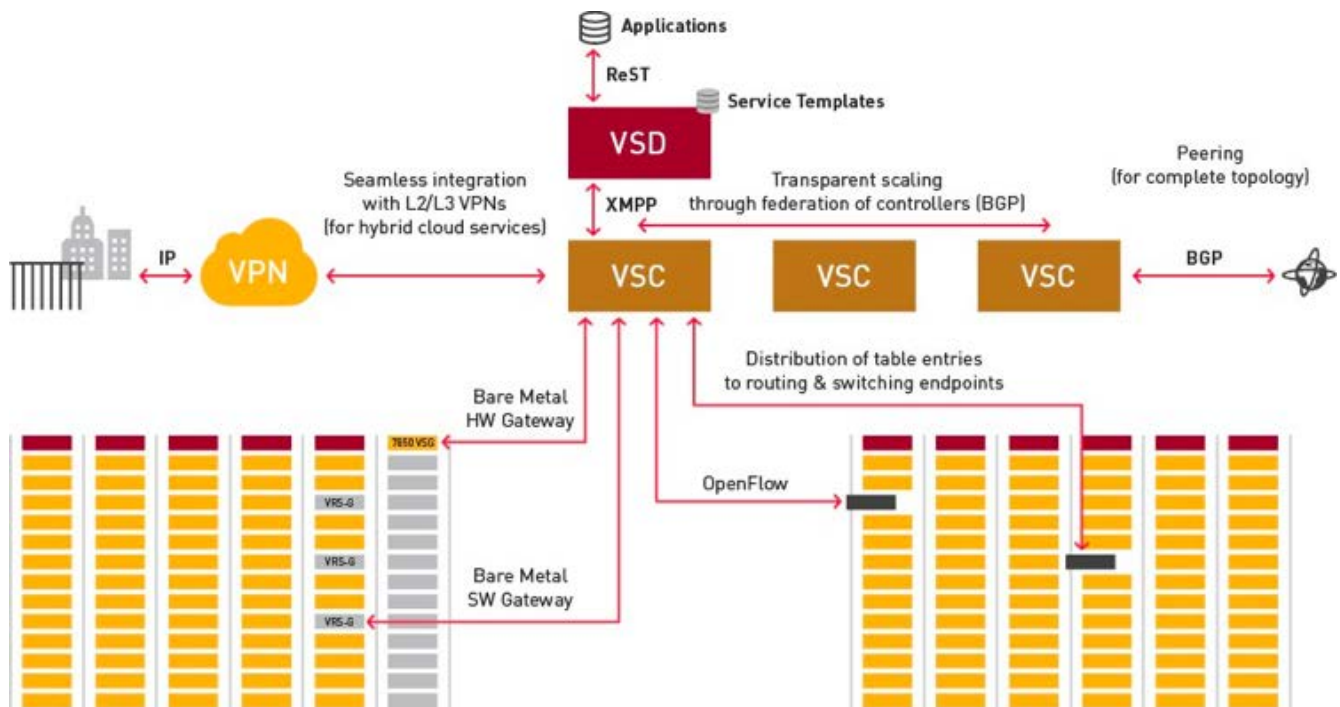
This guide is designed for OpenStack Administrators who are deploying Mirantis OpenStack with Nuage Networks VSP as the SDN solution and Palo Alto Networks Virtual NGFW for security.

2 Solution overview

2.1 Nuage Networks VSP Overview

Nuage Networks Virtualized Services Platform (VSP) is a Software-Defined Networking (SDN) solution that provides data center (DC) network virtualization and automatically establishes connectivity between compute resources upon their creation. Leveraging programmable business logic and a powerful policy engine, the Nuage VSP provides an open and highly responsive solution that scales to meet the stringent needs of massive multi-tenant DCs. The Nuage VSP is a software solution that can be deployed over an existing DC IP network fabric.

The main components in the Nuage VSP solution are: Virtualized Services Directory (VSD) and Virtualized Services Directory Architect (VSD-A), Virtualized Services Controller (VSC), Virtual Routing and Switching (VRS) and Virtual Routing and Switching Gateway (VRS-G). For OpenStack deployments, the solution has an OpenStack Neutron-plugin.



2.1.1 Virtualized Services Directory (VSD)

The Nuage VSD is a programmable policy and analytics engine. It provides a flexible and hierarchical network policy framework that enables IT administrators to define and enforce resource policies in a user- friendly manner. The VSD contains a multi- tenant service directory which supports role- based administration of users, compute, and network resources. It also manages network resource assignments such as IP addresses and ACLs.

For the purpose of service assurance, the VSD allows the definition of sophisticated statistics rules such as collection frequencies, rolling averages and samples, as well as Threshold Crossing Alerts (TCA). When a TCA occurs it will trigger an event that can be exported to external systems through a generic messaging bus. Statistics are aggregated over hours, days and months and stored in a Hadoop® analytics cluster to facilitate data mining and performance reporting. The VSD runs as a number of processes in a virtual machine (VM) environment.

2.1.2 Virtualized Services Controller (VSC)

The Nuage VSC is the industry's most powerful SDN controller. It functions as the robust network control plane for DCs, maintaining a full view of per-tenant network and service topologies. Through the VSC, virtual routing and switching constructs are established to program the network forwarding plane, the Nuage VRS, using the OpenFlow™ protocol.

The VSC communicates with the VSD policy engine using Extensible Messaging and Presence Protocol (XMPP). An ejabberd XMPP server/cluster is used to distribute messages between the VSD and VSC entities. Multiple VSC instances can be federated within and across DCs by leveraging MP-BGP.

2.1.3 Virtual Routing and Switching (VRS)

The Nuage VRS component is an enhanced Open vSwitch (OVS) implementation that constitutes the network forwarding plane. It encapsulates and de-encapsulates user traffic, enforcing L2-L4 traffic policies as defined by the VSD. The VRS tracks VM creation, migration and deletion events in order to dynamically adjust network connectivity.

2.1.4 Virtual Routing Services Gateway (VRS-G)

The software-based VRS-G can be run in a dedicated bare-metal server or as a virtual machine, allowing the incorporation of bare-metal assets as virtualized extensions of the data center, as well as providing an exit gateway for the DVRS' overlay, floating-IP and shared-services traffic.

2.1.5 Nuage Networks OpenStack Neutron Plugin (Kilo)

The Nuage Neutron Plugin allows OpenStack to take advantage of this scale and flexibility. Unlike many other OpenStack Networking plugins, the Nuage Neutron Plugin provides fully distributed L2 and L3 networking, including L2 and L3 network isolation, without requiring centralized routing instances such as the Neutron L3 Agent. The Nuage Neutron Plugin also allows connectivity between OpenStack defined networks and other cloud networks, permitting users to deploy flexible network configurations, including routers and subnets, which are shared between OpenStack and other Cloud Management Systems. This allows Cloud administrators unparalleled flexibility in deploying cloud applications and migrating workloads and applications from one CMS to another.

2.1.5.1 [OpenStack Heat Support for Nuage Extensions](#)

Nuage VSP version 3.2R2 and higher support OpenStack Heat (Kilo) with Neutron supported APIs, and Nuage extensions including: VSD-managed subnet, Gateway and Application Designer.

2.1.5.2 OpenStack Horizon Support for Nuage Extensions

Nuage VSP version 3.2R2 provides Nuage extensions for OpenStack Horizon (Kilo). Supported extensions include: Net Partition, VSD-Managed Subnet, Gateway and Application Designer.

2.1.6 Service Insertion in Nuage Networks VSP

The Nuage Networks VSP provides several deployment options of services that depend on the supported service appliances used by the customer. There is a significant amount of overlap in the integration steps among the different models, and the difference is mostly based on:

- Service type (Firewall or Load Balancer)
- Appliance form factor (Virtual or Physical)
- Appliance deployment model (Centralized or Distributed)
- Multi-tenant support

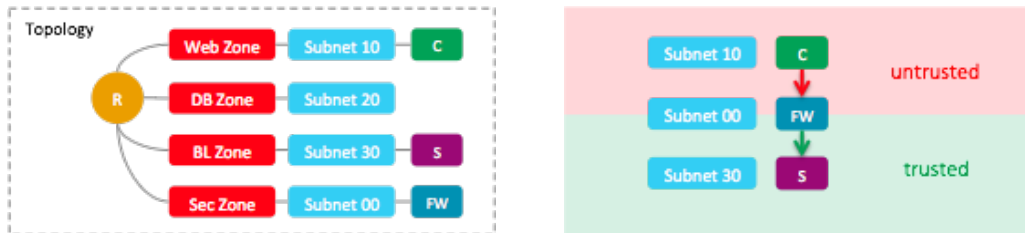
The following diagram provides a high-level view of the supported integration models. In this case, we will use OpenStack as the cloud management system (CMS) to deploy/connect the service appliance as well as any guest VMs. This runbook will focus on the insertion of the PAN vFW, but is not limited to that deployment model.



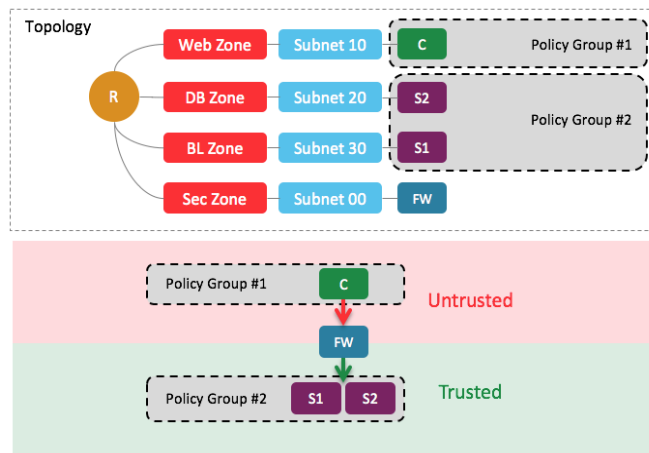
Security services can be deployed within a Nuage Networks VSP overlay network in a centralized or distributed mode. A centralized security appliance is deployed in a dedicated part of the network and traffic will be steered towards a security instance in that location, where it will get processed and if allowed, will then be transported to its final destination. In the distributed security appliance mode, the appliance will be deployed on every hypervisor and traffic will be redirected to the local security appliance (i.e. appliance deployed that particular hypervisor). Nuage supports insertion of virtual and physical service appliances. This document will focus on the virtual appliance, but the physical appliance case will not be significantly different from the virtual one (from a workflow perspective)

To illustrate using a common use-case, the network consists of a single router (L3 domain) and four zones, which are logical groups of subnets, and each security zone will have one subnet. One of the

zones will be a dedicated security zone, which will contain the virtual firewall. The following diagram is an illustration of this (left diagram is the view from VSD and the right view is a logical diagram):

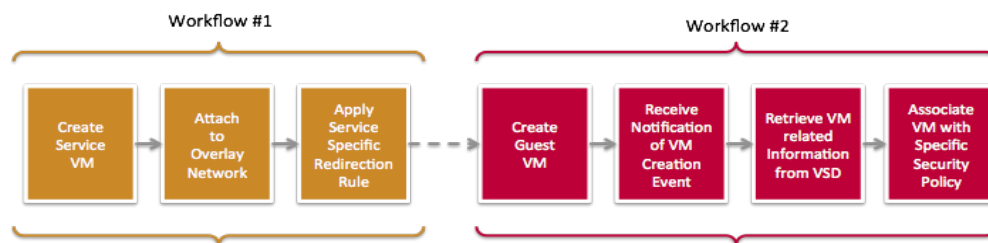


In this example, “ALL” traffic from the “Web Zone” will get redirected to the “Security Zone”, where it will get processed by the virtual firewall. Once processed, traffic will then get sent back to the “Business Logic Zone”. Another way to view this, is having two areas, a “trusted” area and an “untrusted” with a firewall between them, where the traffic from the “untrusted” to the “trusted” area is forwarded to the firewall appliance. The rules that are used to allow/deny traffic can be applied on a policy group “PG” as illustrated in the following diagrams



This model is not limited to a single appliance or the topology shown above. The virtual appliance can be on any subnet, zone or policy group, and there can be multiple instances of the virtual appliance, each appliance serving a particular tenant.

The insertion process consists of three parts. First, the service VM creation and setup. Then, adding a guest VM to the topology. Finally, setting the security policy.



3 Reference Architecture

OpenStack is an extensible, versatile, and flexible cloud management platform. It is a portfolio of cloud infrastructure services that are exposed through ReST APIs. It enables a wide range of control over these services, both from the perspective of an Integrated Infrastructure as a Service (IaaS) controlled by applications and as a set of tools that enable automated manipulation of the infrastructure itself. Mirantis OpenStack is a productized snapshot of the open source technologies with Fuel, a graphical web tool to quickly deploy your cloud environment.

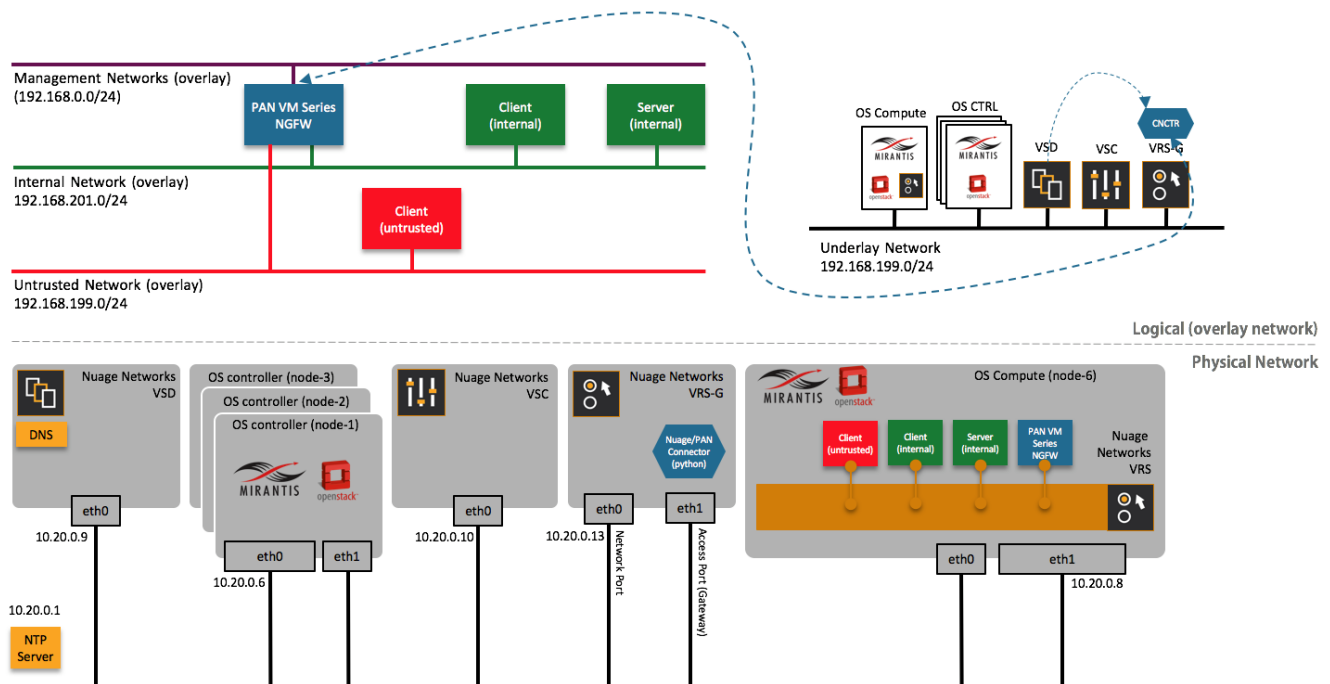
This runbook will utilize Mirantis OpenStack for OpenStack control using three OS controllers, and compute with a single OS compute node. Nuage Networks VSP will be the Software Defined Networking solution deployed, and the integration will be done through a Nuage Networks OpenStack neutron plugin. Palo Alto Networks VM-series firewall will provide the L4-L7 security in the overlay network through service insertion by the Nuage Networks VSP.

Using this architecture, the user can easily deploy their applications using this overlay solution and segment their overlay network to ensure application security with full visibility and control over north-south traffic as well as east-west traffic. Through the Nuage Network VSP, logical groupings of virtual ports can be created independent of VLANs and security policies/rules can be automatically applied/enforced to manage traffic between segments. Whether the traffic needs to be denied, allowed without inspection or redirected to the Palo Alto Networks vFW, the user has complete control of how they would like to manage traffic in a fully programmable and automated way. Furthermore, the joint solution enables them to not only enforce policy and monitor traffic, it can also be easily reprogrammed to dynamically respond to specific anomalies that are observed by the system.

4 Physical & Logical Network Topology

The setup and testing in this runbook will be based on the logical and physical topology outlined in the following diagram. The «underlay» network will be used for the Mirantis OpenStack and Nuage Networks VSP elements that will jointly deliver the overlay networking solution, where the Palo Alto Networks VM-series virtual firewall will be inserted into the data path of traffic that flows between the various micro-segments created by the user. If more virtual firewalls are needed, it is recommended that a Palo Alto Networks Panorama, but the workflow describe in this runbook will not change.

In addition to the overlay networks for user traffic, there will be a separate management network to manage devices in the overlay, and that management network will also have access to underlay network if needed using security policies configured on the Nuage Networks VSP.



The above topology is one example that demonstrates the joint solution, but the solution is not limited to this deployment model. To learn more about the flexibility offered by the Nuage Networks VSP, please refer to the User Guide. In the guide will can learn more about mixing physical devices with VMs or containers as well as connecting multiple data-centers.

5 Installation & Configuration

5.1 Environment Preparation

The lab consists of two physical servers with the following configuration:

- 2x Xeon E5-2620v2, 2.1GHz (24 cores total with HT)
- RAM: 64GB DDR3
- DISK: 2x 400GB SSD

Physical network schema is the following:

- tpi84:eth0 -> to Internet
- tpi84:eth1 -> directly to tpi92:eth1
- tpi92:eth0 -> not connected
- tpi92:eth1 -> directly to tpi84:eth1

Tpi84 has Ubuntu 14.04 installed and libvirt/KVM configured. It's used to carry virtual machines with Fuel Master node, 3 Controller nodes, Nuage VSD, and Nuage VSC nodes. Tpi92 is the only Compute node here.

The Mirantis Software can be downloaded from <https://software.mirantis.com>. The following tool is used for fast preparing virtual networks and VMs on KVM <https://github.com/skolekonov/fuel-kvm/>

```
root@tpi84:~# git clone https://github.com/skolekonov/fuel-kvm/
root@tpi84:~# ./deploy_fuel.sh ../MirantisOpenStack-7.0.iso 5
```

The tool creates virtual bridges, define VMs, and installs Fuel to one of the VMs. After it finished we've got installed Fuel Master node and 5 Target nodes (3 for Controllers and 2 spare nodes). Fuel Master node has 1vCPU and 1GB RAM while target nodes have 1vCPU and 2GB RAM.

Then, at **tpi84**, **eth1** should be put into **fuel-pxe** bridge to make **tpi92** available from Fuel. Finally we have the following configuration:

```
root@tpi84:~# brctl show
```

bridge name	bridge id	STP enabled	interfaces
docker0	8000.56847afe9799	no	
fuel-adm-public	8000.525400853abb	yes	fuel-admlic-nic vnet1
fuel-public	8000.5254007b816a	yes	fuel-public-nic vnet11 vnet3 vnet5 vnet7 vnet9
fuel-pxe	8000.002590c2d009	yes	eth1 fuel-pxe-nic vnet0 vnet10 vnet2

vnet4
vnet6
vnet8

```
root@tpi84:~# ip r
default via 10.10.218.1 dev eth0
10.20.0.0/24 dev fuel-pxe proto kernel scope link src 10.20.0.1
172.16.0.0/24 dev fuel-public proto kernel scope link src 172.16.0.1
172.16.1.0/24 dev fuel-adm-public proto kernel scope link src 172.16.1.1
```

Reboot tpi92 with PXE as the first boot device to get it discovered by Fuel.

5.2 MOS installation

Using Fuel Web UI the following cluster was created:

- OS – Ubuntu 14.04
- Hypervisor – KVM
- Networking – Neutron + VLAN
- Storage – Ceph for all
- Additional services – Murano only

3 Controllers and 1 Compute were added to the cluster:

Controller (3) ☐ Select All

<input type="checkbox"/>	KVM	Untitled (93:2d) CONTROLLER		READY	CPU: 1 (1) HDD: 75.0 GB RAM: 2.0 GB
<input type="checkbox"/>	KVM	Untitled (33:98) CONTROLLER		READY	CPU: 1 (1) HDD: 75.0 GB RAM: 2.0 GB
<input type="checkbox"/>	KVM	Untitled (34:e1) CONTROLLER		READY	CPU: 1 (1) HDD: 75.0 GB RAM: 2.0 GB

Compute (1) ☐ Select All

<input type="checkbox"/>	SUPER MICRO	Untitled (02:99) COMPUTE		READY	CPU: 2 (24) HDD: 0.9 TB RAM: 64.0 GB
--------------------------	----------------	-----------------------------	--	-------	--------------------------------------

A nodes' disk configuration wasn't changed.

Network interfaces configuration of all Controllers is the following:

Nuage + PAN vFW (4 nodes)

Dashboard Nodes Networks Settings Logs Health Check

Interfaces configuration of 3 nodes

Interface	Speed	Admin (PXE)	Storage (VLAN ID)	Management (VLAN ID)	Private (VLAN IDs)
eth0	1.0 Gbps		VLAN ID:102	VLAN ID:101	VLAN IDs:1000-1030
Offloading Modes: Default					
eth1	1.0 Gbps	Public			
Offloading Modes: Default					

Network interfaces configuration of the Compute node is the following:

Nuage + PAN vFW (4 nodes)

Dashboard Nodes Networks Settings Logs Health Check

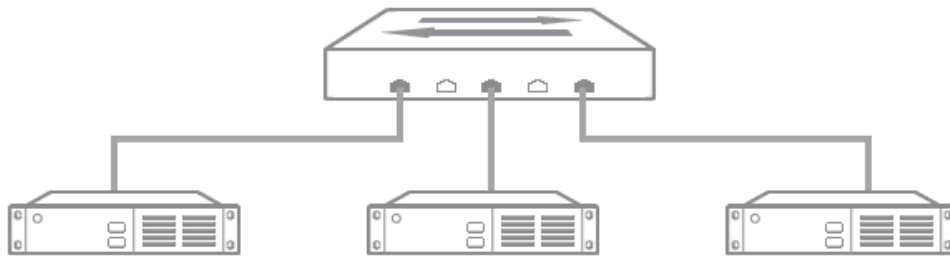
Interfaces configuration of Untitled (02:99)

Interface	MAC	Speed	Admin (PXE)	Storage (VLAN ID)	Management (VLAN ID)	Private (VLAN IDs)
eth0	00:25:90:f5:02:98	0.1 Gbps				
Offloading Modes: Default						
eth1	00:25:90:f5:02:99	1.0 Gbps		VLAN ID:102	VLAN ID:101	VLAN IDs:1000-1030
Offloading Modes: Default						

All knobs and buttons on the Settings tab remained untouched. The same for the Networks tab. Network Verification completed successfully.

Neutron L3 Configuration

Internal network CIDR	192.168.111.0/24
Internal network gateway	192.168.111.1
Guest OS DNS Servers	8.8.4.4
	8.8.8.8



Verification succeeded. Your network is configured correctly.

Then the cluster was deployed successfully.

5.2.1 Health Check Results

The cluster's operability was verified by Health Check using CLI fuel client. Here are the results:

```
[root@fuel ~]# for i in sanity smoke ha cloudvalidation; do echo Running "\"$i\""
```

```
test group; fuel --env 1 health --check $i; done
```

```
Running "sanity" test group
```

```
[ 1 of 16] [success] 'Request flavor list' (4.16 s)
```

```
[ 2 of 16] [success] 'Request image list using Nova' (3.081 s)
```

```
[ 3 of 16] [success] 'Request instance list' (0.3757 s)
```

```
[ 4 of 16] [success] 'Request absolute limits list' (0.1736 s)
```

```
[ 5 of 16] [success] 'Request snapshot list' (2.028 s)
```

```
[ 6 of 16] [success] 'Request volume list' (2.34 s)
```

```
[ 7 of 16] [success] 'Request image list using Glance v1' (0.02689 s)
```

```
[ 8 of 16] [success] 'Request image list using Glance v2' (0.01297 s)
```

```
[ 9 of 16] [success] 'Request stack list' (0.01677 s)
```

```
[10 of 16] [success] 'Request active services list' (2.222 s)
```

```
[11 of 16] [success] 'Request user list' (0.4923 s)
```

```
[12 of 16] [success] 'Check that required services are running' (9.751 s)
```

```
[13 of 16] [success] 'Create and delete Murano environment' (6.302 s)
```

```
[14 of 16] [success] 'Get list of Murano applications categories' (0.1568 s)
```

```
[15 of 16] [success] 'Get list of Murano applications packages' (0.6833 s)
[16 of 16] [success] 'Request list of networks' (2.329 s)
```

Running "smoke" test group

```
[ 1 of 13] [success] 'Create instance flavor' (2.581 s)
[ 2 of 13] [success] 'Check create, update and delete image actions using Glance v1'
(30.32 s)
[ 3 of 13] [success] 'Check create, update and delete image actions using Glance v2'
(19.2 s)
[ 4 of 13] [skipped] 'Create volume and boot instance from it' (0.009922 s) There
are no cinder nodes or ceph storage for volume
[ 5 of 13] [skipped] 'Create volume and attach it to instance' (0.004635 s) There
are no cinder nodes or ceph storage for volume
[ 6 of 13] [success] 'Check network connectivity from instance via floating IP'
(276.7 s)
[ 7 of 13] [success] 'Create keypair' (4.167 s)
[ 8 of 13] [success] 'Create security group' (9.843 s)
[ 9 of 13] [success] 'Check network parameters' (1.134 s)
[10 of 13] [success] 'Launch instance' (65.69 s)
[11 of 13] [success] 'Launch instance with file injection' (128.9 s)
[12 of 13] [success] 'Launch instance, create snapshot, launch instance from
snapshot' (270.0 s)
[13 of 13] [success] 'Create user and authenticate with it to Horizon' (1.827 s)
```

Running "ha" test group

```
[ 1 of 6] [success] 'Check data replication over mysql' (9.432 s)
[ 2 of 6] [success] 'Check if amount of tables in databases is the same on each
node' (10.58 s)
[ 3 of 6] [success] 'Check galera environment state' (3.827 s)
[ 4 of 6] [success] 'Check pacemaker status' (5.901 s)
[ 5 of 6] [success] 'RabbitMQ availability' (35.86 s)
[ 6 of 6] [success] 'RabbitMQ replication' (54.32 s)
```

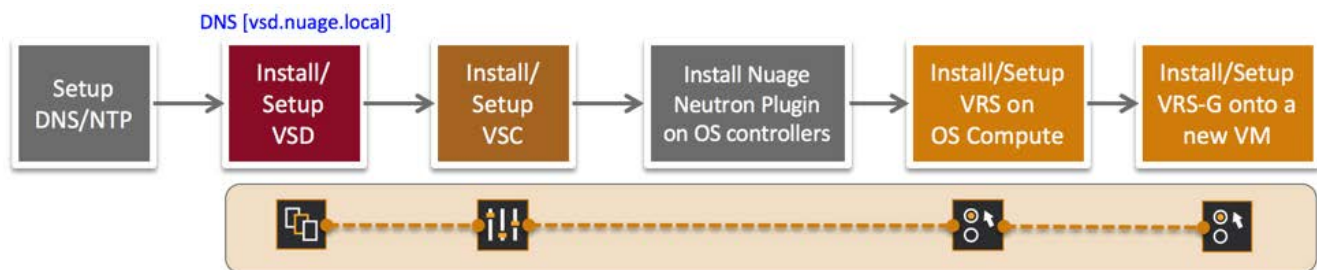
Running "cloudvalidation" test group

```
[ 1 of 2] [success] 'Check disk space outage on controller and compute nodes' (3.578
s)
[ 2 of 2] [success] 'Check log rotation configuration on all nodes' (3.211 s)
```

5.3 Nuage Networks VSP Installation

Automated deployment of the Nuage Networks VSP using Ansible scripts is available (Nuage Networks Installation Runbook MOSv 6 – [download here](#)). However, as of the timing of this test, the MOS 7 automation scripts have not been published yet. Therefore, for this testing, we will outline the detailed steps to install the Nuage Networks VSP in a Mirantis OS environment (MOS 7).

In order to install all of the Nuage Networks VSP components listed in the previous section in a Mirantis OpenStack environment, it is recommended that the installation follow the sequence outlined below



NOTE

The following steps are based on the installation of the Nuage Networks VSP Release 3.2.R4 and the OpenStack Kilo Neutron Plugin.

5.3.1 Pre-installation Setup

The Nuage Networks VSP requires all of its components to be synchronized to an NTP server, therefore it is important to have an NTP server pre-setup and accessible by all of the Nuage Networks VSP components. For this installation, the NTP server was configured as 10.20.0.1

Furthermore, the Nuage VSD requires an FQDN prior to the installation, this entry will be used by the VSC to communicate with the VSD. During this deployment, vsd.nuage.local will be the FQDN for the VSD server.

You will also need a valid product license as well as the following files (please contact your Nuage Networks Account Team for more details):

- Nuage-VSD-3.2.4-134-ISO.tar.gz **OR** Nuage-VSD-3.2.4-134-QCOW.tar.gz
- Nuage-VSC-3.2.4-133.tar.gz
- Nuage-VRS-3.2.4-133-ubuntu.12.04.5.tar.gz **OR** Nuage-VRS-3.2.4-133-ubuntu.14.04.tar.gz
- Nuage-openstack-3.2.4.tar.gz

5.3.2 VSD Installation

Due to the nature of the Nuage Networks VSP and what it provides, all of its components will be installed in the “underlay” (not within OpenStack). Prior to installing the VSD software, the following are the minimum server requirements:

- A physical or virtual machine with:
 - The hardware requirements specified in the current release notes.
 - A freshly installed 64 bit CentOS 6.5 minimal server.
 - Sufficient RAM, where each of the nodes in an HA cluster has to be the same (or at least in the same memory tier). In this example, we will only deploy a single VSD instance (i.e. Standalone VSD deployment)
- VSD Networking Requirements
 - An IP address and an FQDN (i.e. vsd.nuage.local)
 - SSH root access
 - Time synchronization to an NTP server (10.20.0.1)
 - A configured DNS server (setup on 10.20.0.9, the VSD VM using local host configuration and dnsmasq)

It is recommended that an ISO is used for the installation in production networks. However, for PoCs and lab deployments, the qcow2 image is sufficient. In this example, we will use the qcow2 image [file: VSD-3.2.4_134.qcow2]

5.3.2.1 VSD VM Setup

In this section, you should be ready to start a new virtual machine to install the VSD server software on. In this example, an 8GB of memory was used, and that is reflected in the virt-install parameters (i.e. -r 8192). If your server has more memory, then you can launch with a different memory parameter.

1. Copy qcow2 image to the libvirt images directory

- `mv VSD-3.2.4_134.qcow2 /var/lib/libvirt/images/VSD-3.2.4_134.qcow2`

2. Set a disk path variable and a VM name:

- `vsd_disk=/var/lib/libvirt/images/VSD-3.2.4_134.qcow2`
- `vsd_name=vsd_3.2R4`

3. Launch VSD VM:

- `virt-install --connect qemu:///system -n $vsd_name -r 8192 --os-type=linux --os-variant=rhel6 --disk path=$vsd_disk,device=disk,bus=virtio,format=qcow2 --vcpus=6 --graphics vnc,listen=0.0.0.0 --noautoconsole --import --network network=fuel-pxe`

Sample Output

```
# vsd_disk=/var/lib/libvirt/images/VSD-3.2.4_134.qcow2
# vsd_name=vsd_3.2R4
# virt-install --connect qemu:///system -n $vsd_name -r 8192 --os-type=linux --os-variant=rhel6 --disk
path=$vsd_disk,device=disk,bus=virtio,format=qcow2 --vcpus=6 --graphics vnc,listen=0.0.0.0 --
noautoconsole -import --network network=fuel-pxe

Starting install...
Creating
domain...
|      0 B      00:00
Domain creation completed. You can restart your domain by running:
virsh --connect qemu:///system start vsd_3.2R4
```

4. After the VM has been created, the network settings should be edited to ensure proper connectivity. Use the following commands to edit the VM configuration:
 - Display VMs:
 - `virsh list --all`
 - Edit target VM's configuration:
 - `virsh edit vsd_3.2R4`
5. [Optional – for reference only] While in edit mode, search for the “Interface type='network'” section and modify it so that the source is ‘fuel-pxe’ (the value can be changed according to your own setup needs, this is an example of a way to change the interface, you can skip if step #3 was sufficient)

Original Configuration	Modified Configuration
<pre><interface type='network'> <mac address='52:54:00:3e:55:56' /> <source network='default' /> <target dev='vnet12' /> <model type='virtio' /> <alias name='net0' /> <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0' /> </interface></pre>	<pre><interface type='network'> <mac address='52:54:00:3e:55:56' /> <source network='fuel-pxe' /> <target dev='vnet12' /> <model type='virtio' /> <alias name='net0' /> <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0' /> </interface></pre>

6. Restart the VM for the change to take effect using the following commands:
 - `virsh shutdown vsd_3.2R4`
 - `virsh start vsd_3.2R4`
7. Access the console of the newly started VM and when prompted for a username and password, use root/Alcateldc:
 - `virsh console vsd_3.2R4`

Sample Output

```
# virsh console 3171
Connected to domain vsd_3.2R4
Escape character is ^]

CentOS release 6.5 (Final)
Kernel 2.6.32-431.17.1.el6.x86_64 on an x86_64

login: root
Password:
Last login: Mon Jan 18 18:17:45 on ttyS0
Welcome to VSD. (3.2.4_134)
```

8. Modify the settings of the network interface “eth0” so that it gets an IP address on the private management interface (in this example, 10.20.0.9/24 was chosen as the IP address for the VSD).

Original Configuration	Modified Configuration
<pre>/etc/sysconfig/network-scripts/ifcfg-eth0 DEVICE="eth0" IPV6INIT="yes" NM_CONTROLLED="yes" ONBOOT="yes" TYPE="Ethernet" BOOTPROTO="dhcp"</pre>	<pre>/etc/sysconfig/network-scripts/ifcfg-eth0 DEVICE="eth0" IPV6INIT="yes" NM_CONTROLLED="yes" ONBOOT="yes" TYPE="Ethernet" BOOTPROTO="static" NETWORK=10.20.0.0 NETMASK=255.255.255.0 IPADDR=10.20.0.9</pre>

9. [optional] Add the following static route on the host “192.168.202.0/24 via 10.20.0.254”; create “route-eth0”. The next-hop address will vary based on your choice of address, it just needs to be the gateway for the private management subnet (10.20.0.254 was the one configured for our setup). This entry is used in the setup so that you can connect to the PAN vFW GUI using SSH tunnels. This is not needed if you can access the IP address locally

Original Configuration	Modified Configuration
<none>	<pre>/etc/sysconfig/network-scripts/route-eth0 192.168.202.0/24 via 10.20.0.254</pre>

10. Restart network services

Sample Output

```
#/etc/init.d/network restart
```

11. Check NTP Server configuration (NTP server in our example is 10.20.0.1)

Sample Output

```
#cat /etc/ntp.conf | grep sevrer

# Use public servers from the pool.ntp.org project.
server 10.20.0.1 iburst
#server 0.centos.pool.ntp.org iburst
#server 1.centos.pool.ntp.org iburst
#server 2.centos.pool.ntp.org iburst
#server 3.centos.pool.ntp.org iburst
#broadcast 192.168.1.255 autokey # broadcast server
#broadcast 224.0.1.1 autokey      # multicast server
#manycastserver 239.255.254.254    # manycast server

# ntpq -p

      remote           refid      st t when poll reach  delay  offset jitter
=====
*10.20.0.1          95.213.132.254    3 u  896 1024  377   1.712    0.431   0.618

# ntpstat

synchronised to NTP server (10.20.0.1) at stratum 4
time correct to within 267 ms
polling server every 1024 s
```

5.3.2.2 VSD Software Installation

For detailed installation steps please refer to the Nuage Networks VSP installation guide. This section will describe one particular installation path that can be used as a basic guide to setup a demo environment and/or POC.

The two main components in the earlier setup process were the NTP and DNS server settings. The installation checks for the hostname as well as a synchronized status for the server and will abort if either check fails. In order to go through the installation, execute the installation script “/opt/vsd/install.sh” and answer the following questions:

- VSD Configuration? **standalone**
- VSD Node FQDN? **vsd.nuage.local**
- Continue? **Yes**

Once the installation, which usually takes 15-20 minutes, please use “service vsd status” to confirm that all service is operational. The following is a sample output from an installation that was completed for this setup to be used as a reference:

Sample Output

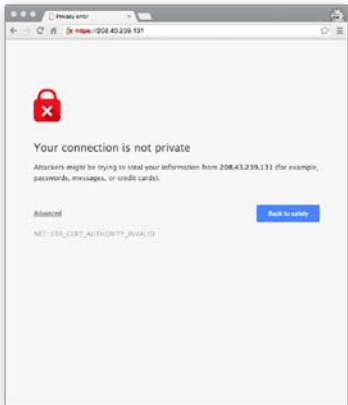
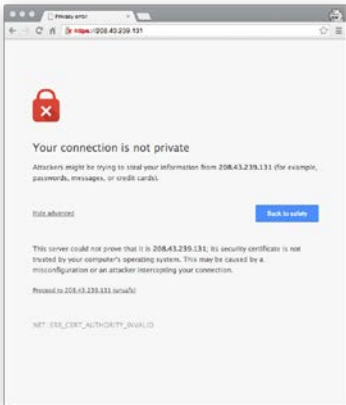
```
/opt/vsd/install/checkNTP.sh
[root@vsd ~]
[root@vsd ~]# ./install.sh
-----
V I R T U A L I Z E D   S E R V I C E S   D I R E C T O R Y
version 3.2.R4
(c) 2014 Nuage Networks
-----
VSD supports two configurations:
1) HA, consisting of 3 redundant installs of VSD with a cluster name node server.
2) Standalone, where all services are installed on a single machine.
Is this a redundant (r) or standalone (s) installation [r|s]? (default=s): s
Please enter the fully qualified domain name (fqdn) for this node: vsd.nuage.local
Install VSD on single host vsd.nuage.local...
VSD node: vsd.nuage.local
Continue [y|n]? (default=y): y
Starting VSD installation. This may take as long as 20 minutes in some situations ...
A self-signed certificate has been generated to get you started using VSD.
You may import one from a certificate authority later.
VSD installed and the services have started.
[root@vsd ~]# service vsd status
***** VSD Local Summary *****
** Host Name: vsd.nuage.local
** Host IP: 10.20.0.9
** Node Type Standalone Node CDHS
** VSD Version: 3.2.4_134
** VSD Cluster Mode: Standalone
** Ejabberd TLS Mode: Clear
** NTP Status: PASS
** Percona Status: PASS
** Ejabberd Status: PASS
** JBOSS Status: PASS
** Mediator Status: PASS
** STATS Status: PASS
** Tca Status: PASS
** Stats Coll. Status: PASS
*****
[root@vsd ~]#
```

5.3.2.3 Remote Connectivity Test & Activation

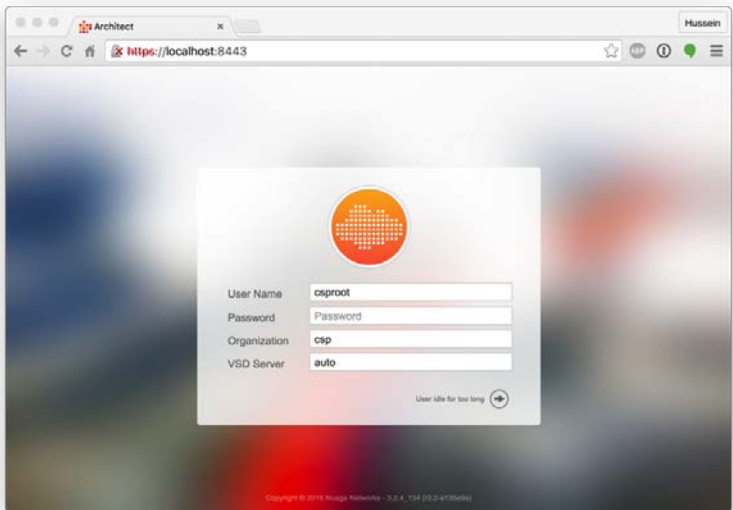
After the VSD has been started and the configuration required to remotely access it has been completed, it should now be possible to remotely access the Nuage Networks VSD Architect. Use the following URL format in a new browser (the VSD IP address can be found during the installation process as the “Host IP” , or during the VM setup #8 section 5.3.2.1)

URL: <https://10.20.0.9:8443>

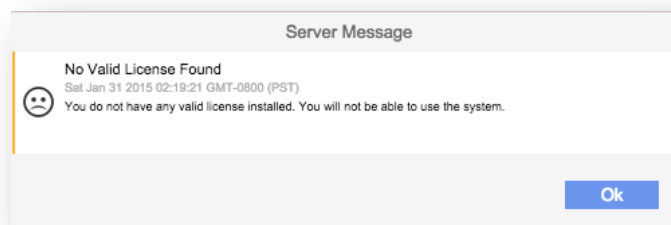
You will be warned that the connection is not a private connection. Please acknowledge the error and proceed to connect to the VSP architect.

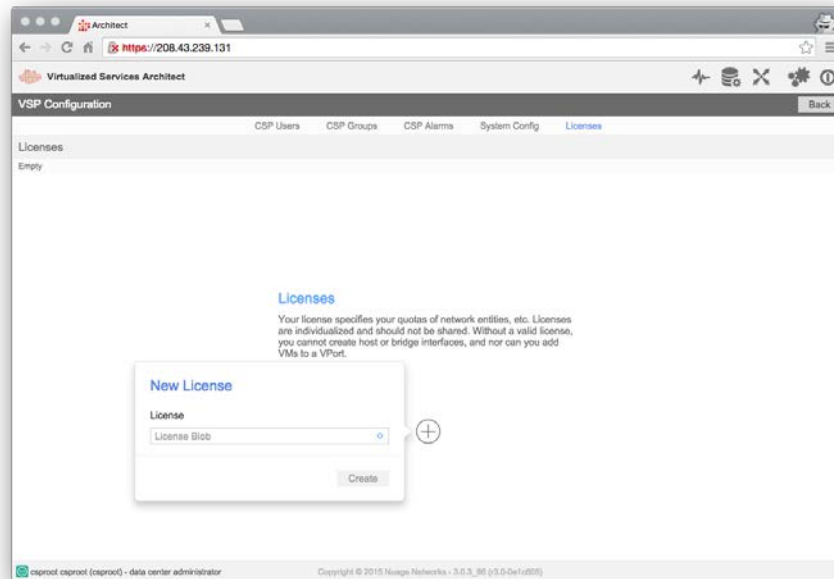
Initial browser security warning	“Proceed” link shown in advanced section
 <p>A screenshot of a browser's security warning dialog. It features a red padlock icon with a white 'X'. The text reads: "Your connection is not private. Attackers might be trying to steal your information from 208.43.239.131 (for example, passwords, messages, or credit cards)." Below this, it says "Advanced" and "NET::ERR_CERT_AUTHORITY_INVALID". A blue "Back to safety" button is visible on the right.</p>	 <p>A screenshot of the same browser security warning dialog, but with the "Advanced" section expanded. It shows additional text: "This server could not prove that it is 208.43.239.131; its security certificate is not trusted by your computer's operating system. This may be caused by a misconfiguration or an attacker intercepting your connection. Proceed to 208.43.239.131 (unsafe)". The "Back to safety" button remains on the right.</p>

At the login window, please enter the following default login credentials:

<p>User Name: csproot Password: csproot Organization: csp VSD server: auto</p>	 <p>A screenshot of a web browser window titled "Architect" showing a login form. The form has fields for "User Name" (filled with "csproot"), "Password" (filled with "csproot"), "Organization" (filled with "csp"), and "VSD Server" (filled with "auto"). Above the fields is a circular logo with a grid pattern. Below the fields is a "User idle for too long" message with a refresh icon. The browser's address bar shows "https://localhost:8443".</p>
---	---

The system will display a “No Valid License Found” warning. You can acknowledge the message and enter your license key in the VSD Configuration section, under “License”



**NOTE**

For additional details on using the VSD Architect, please refer the Nuage Networks User Guide. Also, please contact your Nuage Networks sales representative to obtain a product license

5.3.3 VSC Installation

The VSC will also be installed in the “underlay” (not in the OpenStack overlay networks) and is also qcow2 based image with the VSC software pre-installed. Listed below, are the minimum server requirements:

- A physical or virtual machine with:
 - The hardware requirements specified in the current release notes.
 - Sufficient RAM
- VSC Networking Requirements
 - The Linux server is a clean installation with a minimum of configuration and applications.
 - An IP address is already assigned for the management network
 - Either one or three NTP servers for the VSC to sync with
 - The user has root access and has a means of copying the VSC software files to the server.
 - Two independent network interfaces for management and data traffic, connected to two Linux Bridge interfaces (In this deployment, we will use a single network interface)

5.3.3.1 VSC VM Setup

In this section, you should be ready to start a new virtual machine with VSC software loaded on it using a qcow2 file.

- Transfer VSC qcow2 image “vsc_singledisk.qcow2” to the target server
- Once the transfer is complete, copy the file to the libvirt/images directory:
 - `cp /tmp/nuage_images/vsc_singledisk.qcow2 /var/lib/libvirt/images/`
- Use the “vsc.xml” file that was provided with the Nuage software release to define a new VM. If you are going to deploy the “single_disk” version, then use the “vsc.xml” from that directory
 - `cp /single_disk/vsc.xml ./vsc_3.2R4.xml`
- Edit the xml file to modify the “interface” to have a “fuel-pxe” as the source network and correct disk source file

Original vsc.xml (single_disk)	Modified file “vsc_3.0R3.xml”
	<pre><domain type='kvm'> <name>vsc-3.2R4</name> ... <type arch='x86_64' machine='pc-i440fx-trusty'>hvm</type> <devices> <emulator>/usr/bin/kvm-spice</emulator> <disk type='file' device='disk' snapshot='no'> <driver name='qemu' type='qcow2' cache='writethrough'/> <source file='/var/lib/libvirt/images/vsc_singledisk.qcow2'/> <target dev='hda' bus='ide'/> <boot order='1'/> <address type='drive' controller='0' bus='0' target='0' unit='0'/> </disk> ... <controller type='pci' index='0' model='pci-root'/> <interface type='network'> <mac address='52:54:00:8f:e6:85'/> <source network='fuel-pxe'/> <model type='virtio'/> <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/> </interface></pre>

- Define and Start the VSC VM:
 - `virsh define vsc_3.2R4.xml`
 - `virsh start vsc-3.2R4`

5.3.3.2 VSC Configuration

- Check the status and access the console of the newly started VM and when prompted for a username and password, use admin/admin:
 - `virsh list`
 - `virsh console vsc_3.2R4`

Sample Output

```
[root@nuage-vsc ~]# virsh console vsc_3.2R4
```

```
Login: admin
```

```
Password: <admin password>
```

- Next, we will configure the VSC itself. Please refer to the VSP installation guide for more details on the commands being used here:
 - Configure the system and management interface:
 - Set the location of the configuration file:
 - `bof primary-config cf1:\config.cfg`
 - Configure the management interface address:
 - `bof address 10.20.0.10/24`
 - Configure the DNS server & static routes for this guides testing:
 - `bof primary-dns 10.20.0.9`
 - `static-route 172.0.0.0/8 next-hop 10.20.0.2` (optional – used for SSH tunnel access only / please ignore)
 - `static-route 192.0.0.0/8 next-hop 10.20.0.2` (optional – used for SSH tunnel access only / please ignore)
 - Save the management interface and boot information, then display configuration:
 - `bof save`
 - `show bof`

Sample Output

```
*A:vsc3.2R4# bof save
```

```
Writing BOF to cf1:/bof.cfg ... OK  
Completed.
```

```
*A:vsc3.2R4# show bof
```

```
=====
```

```
BOF (Memory)
```

```
=====
```

primary-image	cf1:\timos\cpm.tim
primary-config	cf1:\config.cfg
address	10.20.0.10/24 active
primary-dns	10.20.0.9
static-route	172.0.0.0/8 next-hop 10.20.0.2
static-route	192.0.0.0/8 next-hop 10.20.0.2
autonegotiate	
duplex	full
speed	100
wait	3
persist	off
no li-local-save	
no li-separate	
console-speed	115200

```
=====
```

```
*A:vsc3.2R4#
```

- Configure the system name:
 - `configure system name vsc3.2R4`
- Configure the system's "control" interface:
 - `configure router interface "control" address 10.20.0.10/24`
 - check status using `"show router interface"`

Sample Output

```
*A:vsc# show router interface
=====
Interface Table (Router: Base)
=====
Interface-Name      Adm      Opr(v4/v6)  Mode      Port/SapId
IP-Address          PfxState
-----
control             Up        Up/--       Network   A/2
10.20.0.10/24       n/a
system              Up        Down/--     Network   system
-                   -
-----
Interfaces : 2
=====
```

- Configure NTP on the VSC:
 - `configure system time ntp server 10.20.0.9`
 - `configure system time ntp no shutdown`
 - check status using “`show system ntp all`”

Sample Output

```
*A:vsc3.2R4# show system ntp all
=====
NTP Status
=====
Configured       : Yes          Stratum         : 5
Admin Status     : up            Oper Status     : up
Server Enabled   : No           Server Authenticate : No
Clock Source     : 10.20.0.9   Auth Check      : Yes
Current Date & Time: 2016/01/25 07:23:10 UTC
=====

=====
NTP Active Associations
=====
State  Remote      Reference ID  St Type  A  Poll Reach  Offset(ms)
-----
chosen 10.20.0.9    10.20.0.1    4  srvr  -  256 YYYYYYYY -0.877
=====

=====
NTP Clients
=====
*A:vsc3.2R4#
```

- Configure connectivity to the VSD server using the hostname used during the installation of the VSD server. For this to function, you will need to have DNS also working, so we will include steps to check that as well:
 - `ping router “management” 10.20.0.9`
 - `ping router “management” vsd.nuage.local`
 - `configure vswitch-controller xmpp-server “NSC-vPE-2:password@vsd.nuage.local”`
 - check status (make sure that the state is “Functional”)

Sample Output

```
*A:vsc3.2R4# show vswitch-controller xmp-server detail

=====
XMPP Server Table
=====
XMPP FQDN       : vsd.nuage.local
XMPP User Name  : vsc32R4
Last changed since : 9d 10:02:01
State           : Functional
IQ Tx.          : 7496
IQ Error        : 0
IQ Min. Rtt     : 60
IQ Ack Rcvd.    : 7486
Nuage Updates Rcvd.: 61
Nuage Msg Tx.   : 6581
Nuage Msg Ack. Rx. : 6571
Nuage Msg Min. Rtt : 20
Nuage Sub Tx.   : 4
Nuage Msg Timed Out: 10
Encryption Type : none
IQ Rx.          : 7496
IQ Timed Out    : 10
IQ Max. Rtt     : 10380
VSD Updates Rcvd. : 28015
Nuage Msg Rx.   : 6581
Nuage Msg Error  : 0
Nuage Msg Max. Rtt : 10380
Nuage UnSub Tx. : 0

=====
```

- Save the VSC configuration:
 - `admin save`

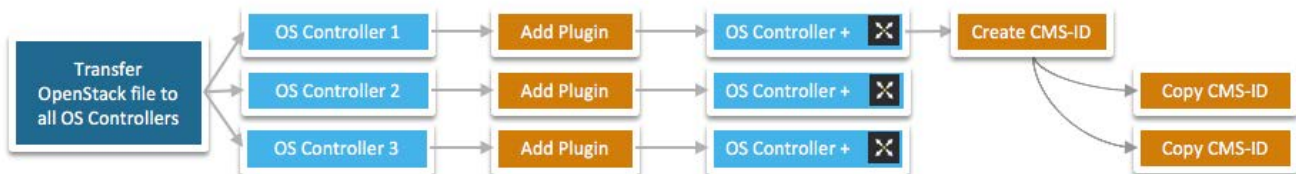
Sample Output

```
*A:vsc3.2R4# admin save
Writing configuration to cf1:\config.cfg
Saving configuration ... OK
Completed.
```

The VSC setup has now been complete, and next we will deploy the Nuage Networks Neutron Plugin on the OS controller(s)

5.3.4 Neutron Plugin installation

The Mirantis setup has three Openstack “Controllers” (node-1, node-2 and node-3 at 10.20.0.6, 10.20.0.5, 10.20.0.3 respectively). All required Nuage Networks VSP OpenStack related components can be found in the file “Nuage-openstack-3.2.4.tar.gz” (we will use the Kilo / Ubuntu 14.04 plugin) and the VRS file “Nuage-VRS-3.2.4-133-ubuntu.14.04.tar.gz”. The installation process will follow the outline below.



5.3.4.1 Install the Nuage Networks Neutron Plugin on Controller 1

- Copy and untar the Nuage-openstack-3.2.4.tar.gz file on the OS Controller (Node-1)
 - `tar xvfz Nuage-openstack-3.2.4.tar.gz`
- Install the following 5 packages:
 1. `cd kilo/ubuntu-1404`
 2. `dpkg -i nuage-openstack-neutron_2015.1.1725_all.deb`
 3. `dpkg -i nuagenetlib_2015.1.3.2.4.133_all.deb`
 4. `dpkg -i nuage-openstack-neutronclient_2015.1.1725_all.deb`
 5. `dpkg -i nuage-openstack-heat_2015.1.1725_all.deb` (optional)
 6. `dpkg -i nuage-openstack-horizon_2015.1.1725_all.deb` (optional)

NOTE

Nuage Neutron client extensions require the Python-Neutron client to be version 2.4 and above because it uses the extensions framework delivered in OpenStack Kilo with version 2.4.0. Ubuntu Cloud Kilo repo comes with Python-Neutronclient version 2.3.11. Until this is upgraded to 2.4.0, some of the features will not work. See the Nuage Networks VSP Release Notes for details.

- Update the dashboard `local_settings.py` to enable Nuage Extension

Sample Configuration

```
#vi /usr/share/openstack-dashboard/openstack_dashboard/local/local_settings.py

HORIZON_CONFIG = {
    'dashboards': ('project', 'admin', 'settings',),
    'default_dashboard': 'project',
    'user_home': 'openstack_dashboard.views.get_user_home',
    'ajax_queue_limit': 10,
    'auto_fade_alerts': {
        'delay': 3000,
        'fade_duration': 1500,
        'types': ['alert-success', 'alert-info']
    },
    'help_url': "http://docs.openstack.org",
    'exceptions': {'recoverable': exceptions.RECOVERABLE,
                  'not_found': exceptions.NOT_FOUND,
                  'unauthorized': exceptions.UNAUTHORIZED},
    'customization_module': 'nuage_horizon.customization',
    'modal_backdrop': 'static',
    'angular_modules': [],
    'js_files': [],
    'js_spec_files': [],
}
```

- Update Web Server config file to use Nuage Extension

NOTE

Add the following Alias before Alias for `/horizon/static`:

```
Alias /horizon/static/nuage /usr/lib/python2.7/dist-packages/nuage_horizon/static/
```

Sample Configuration

```
#vi /etc/apache2/conf-available/openstack-dashboard.conf

<Directory /usr/lib/python2.7/dist-packages/nuage_horizon>
    Options FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
Alias /horizon/static/nuage /usr/lib/python2.7/dist-packages/nuage_horizon/static/
```

- Restart the Web Server and Neutron Server
 - `service apache2 restart`
 - `service neutron-server restart`

- Configure the OS controller and plugin
 - Modify Neutron Configuration

Sample Configuration

```
# vi /etc/neutron/neutron.conf

[DEFAULT]
allow_overlapping_ips = True
core_plugin = neutron.plugins.nuage.plugin.NuagePlugin # L3 service plugin must be
disabled # service_plugins =neutron.services.l3_router.l3_router_plugin.L3RouterPlugin
```

- Modify Nova Configuration

Sample Configuration

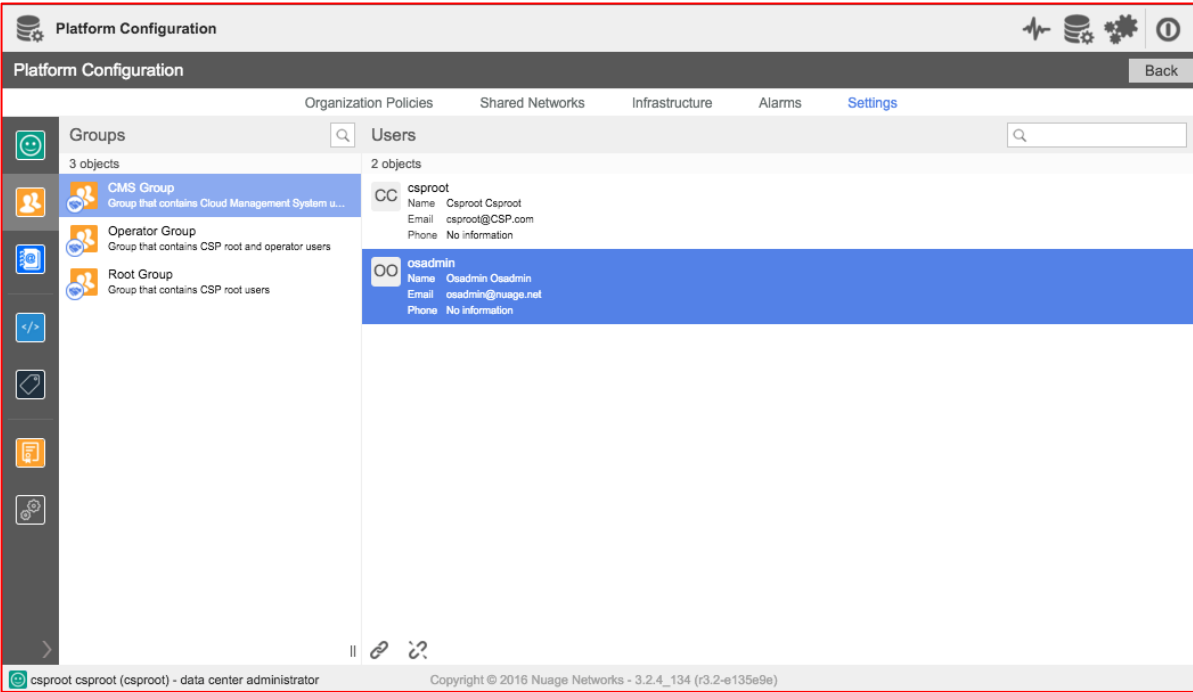
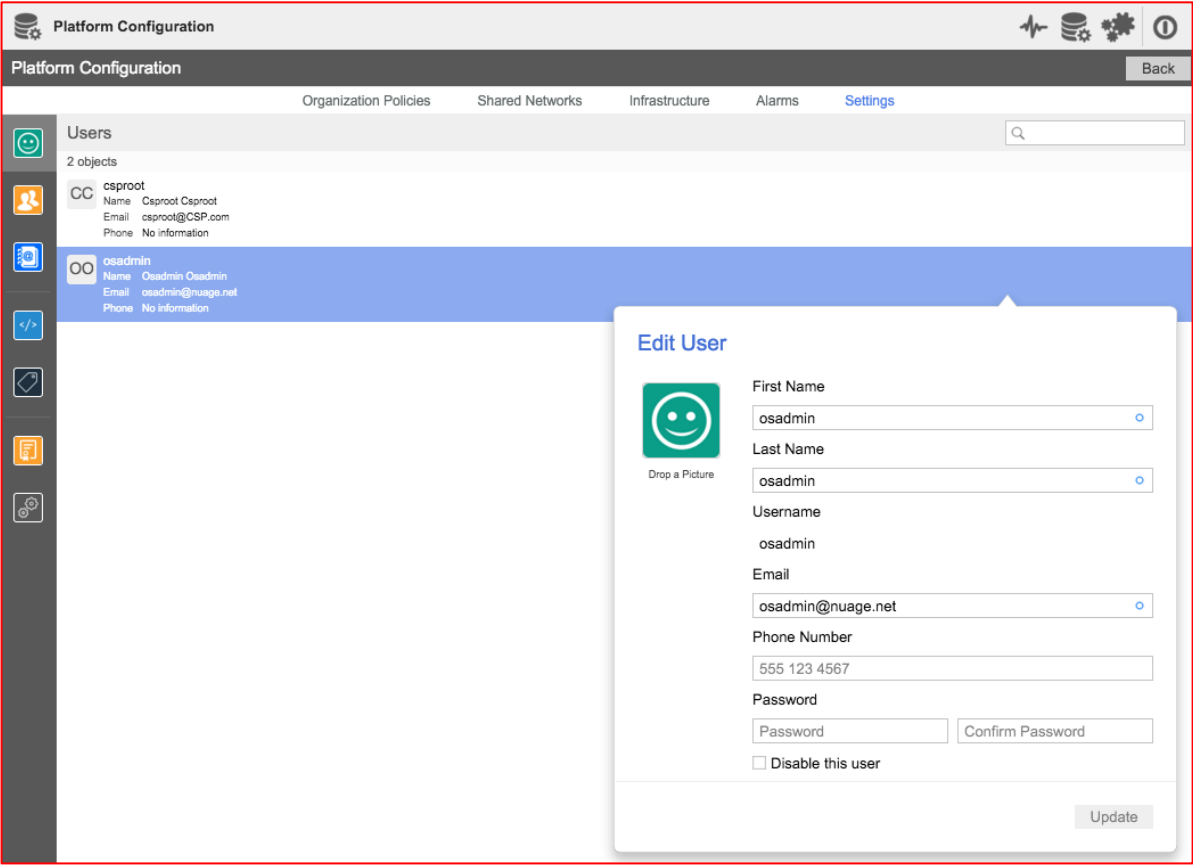
```
# vi /etc/nova/nova.conf

[DEFAULT]
network_api_class = nova.network.neutronv2.api.API
security_group_api = neutron
firewall_driver=nova.virt.firewall.NoopFirewallDriver
[neutron]
ovs_bridge = alubr0
[libvirt]
vif_driver = nova.virt.libvirt.vif.LibvirtGenericVIFDriver
```

- On the Nuage Networks VSD, create a new CMS user (i.e. osadmin , password: osadmin) and assign the user to the CMS user group (same permissions as csproot)

GUI Flow





- Back on the OS controller, create a nuage plugin file in the new «[nuage](#)» directory and the following content, paying attention to the server IP address (VSD IP), the user (VSD newly created user) and the «default_net_partition_name»

Sample Configuration

```
#cd /etc/neutron/plugins/
#mkdir nuage
#vi /etc/neutron/plugins/nuage/nuage_plugin.ini

# [DATABASE]
# Database connection if not specified in neutron.conf
# connection =
mysql://nuage_neutron:$DB_PASS@$OPENSTACK_HOSTNAME/nuage_neutron?charset=utf8
# [KEYSTONE]
# Keystone authentication if not specified in neutron.conf
# keystone_service_endpoint = http://$OPENSTACK_HOSTNAME:35357/v2.0/
# keystone_admin_token = $ADMIN_TOKEN
[RESTPROXY]
# Desired Name of VSD Organization/Enterprise to use when net-partition
# is not specified
default_net_partition_name = OpenStack_mirantis
# Hostname or IP address and port for connection to VSD server
server = 10.20.0.9:8443
# VSD Username and password for OpenStack plugin connection
# User must belong to CSP Root group and CSP CMS group
serverauth = osadmin:osadmin
### Do not change the below options for standard installs
organization = csp
auth_resource = /me
serverssl = True
base_uri = /nuage/api/v3_2
```

- On the OS Controller, add the following line to the /etc/default/neutron-server (path to the nuage neutron plugin file)
 - NEUTRON_PLUGIN_CONFIG="/etc/neutron/plugins/nuage/nuage_plugin.ini"

Sample Configuration

```
#vi /etc/default/neutron-server

NEUTRON_PLUGIN_CONFIG="/etc/neutron/plugins/nuage/nuage_plugin.ini"
```

- Execute the database migration script and enable the CMS ID for the Nuage Networks plugin by using the scripts in nuage-openstack-upgrade-*.tar.gz
 1. mkdir -p upgrade
 2. tar xvfz nuage-openstack-upgrade-1725.tar.gz -C upgrade
 3. cd upgrade/

4. `python nuage-os-upgrade/set_and_audit_cms.py --neutron-config-file /etc/neutron/neutron.conf --plugin-config-file /etc/neutron/plugins/nuage/nuage_plugin.ini`
5. verify that the CMS ID was added to the `/etc/neutron/plugins/nuage/nuage_plugin.ini` file, then copy it (to be used on OS controller 2 and 3 – you do not need to run this script on the other two controllers)

Sample Configuration

```
#cd nuage/
#cd kilo/
#cd ubuntu-1404/
#mkdir -p upgrade
#tar xvfz nuage-openstack-upgrade-1725.tar.gz -C upgrade
#cd upgrade/

#python nuage-os-upgrade/set_and_audit_cms.py --neutron-config-file
/etc/neutron/neutron.conf --plugin-config-file
/etc/neutron/plugins/nuage/nuage_plugin.ini

#vi /etc/neutron/plugins/nuage/nuage_plugin.ini

# [DATABASE]
# Database connection if not specified in neutron.conf
# connection =
mysql://nuage_neutron:$DB_PASS@$OPENSTACK_HOSTNAME/nuage_neutron?charset=utf8
# [KEYSTONE]
# Keystone authentication if not specified in neutron.conf
# keystone_service_endpoint = http://$OPENSTACK_HOSTNAME:35357/v2.0/
# keystone_admin_token = $ADMIN_TOKEN
[RESTPROXY]
# Desired Name of VSD Organization/Enterprise to use when net-partition
# is not specified
default_net_partition_name = OpenStack_mirantis
# Hostname or IP address and port for connection to VSD server
server = 10.20.0.9:8443
# VSD Username and password for OpenStack plugin connection
# User must belong to CSP Root group and CSP CMS group
serverauth = osadmin:osadmin
### Do not change the below options for standard installs
organization = csp
auth_resource = /me
serverssl = True
base_uri = /nuage/api/v3_2
cms_id = b39014f5-99fa-4370-82af-edd6f5651ed9
```

7. The following Neutron services must be disabled because their functions are provided by Nuage VSP:
 - a. L3 agent
 - b. DHCP agent
 - c. Metadata agent
 - d. Open vSwitch agent

Sample Output

```
apt-get remove neutron-dhcp-agent neutron-l3-agent neutron-metadata-agent  
neutron-plugin-openvswitch
```

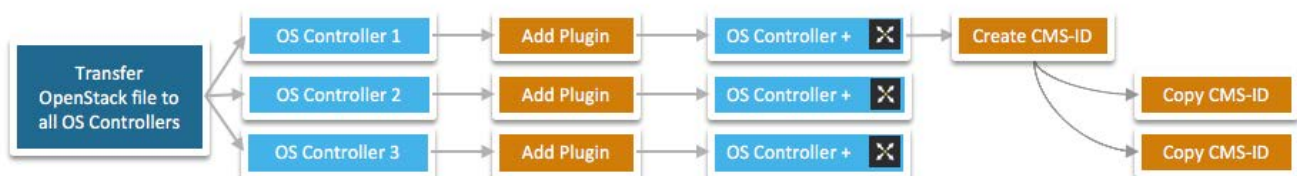
8. Restart the neutron server

Sample Output

```
service neutron-server restart
```

5.3.4.2 Install the Nuage Networks Neutron Plugin on Controllers 2 and 3

Repeat all of the previous steps on all OS controllers, except for the «CMS-ID» step «python set_and_audit_cms.py --neutron-config-file /etc/neutron/neutron.conf --plugin-config-file /etc/neutron/plugins/nuage/nuage_plugin.ini». Instead, you can paste the CMS-ID that was generated in the “nuage_plugin.in” file from the first controller “cms_id = b39014f5-99fa-4370-82af-edd6f5651ed9”



5.3.5 VRS Installation

The VRS is the final step of the installation and the steps (and flow) outlined below should be executed on all OS compute nodes. Required packages must be installed prior to installing any Nuage Networks components.



5.3.5.1 Deployment and Package Installation

- Unpack the package files on the OS compute node:
 - `tar xvfz Nuage-openstack-3.2.4.tar.gz`
 - `tar xvfz Nuage-VRS-3.2.4-133-ubuntu.14.04.tar.gz`
- Start the installation process and follow the steps below:
 - `sudo apt-get update`
 - `apt-get install qemu-kvm libvirt-bin libjson-perl python-twisted-core vlan`
 - `apt-get remove openvswitch-common`
 - `dpkg -i nuage-openvswitch-common_3.2.4-133_amd64.deb` ; if you encounter “dependency problems”, then resolve using the following
 - `apt-get -f install`
 - `apt-get install python-setproctitle`
 - `dpkg -i nuage-openvswitch-common_3.2.4-133_amd64.deb`
 - `dpkg -i nuage-python-openvswitch_3.2.4-133_all.deb`
 - `dpkg -i nuage-openvswitch-switch_3.2.4-133_amd64.deb`
- (Optional) Nuage Networks Metadata Agent Installation
 - `apt-get install -y python-novaclient`
 - `cd kilo/ubuntu-1404/`
 - `cd metadata-agent/`
 - `cd ubuntu-14.04/`
 - `dpkg -i nuage-metadata-agent_3.2.4-133_all.deb`

5.3.5.2 Configuration

- Edit the configuration file at /etc/default/openvswitch-switch to configure the systems personality (VRS or VRS-G) and the Controller's IP address (In this case we only have a one controller)
 - vi /etc/default/openvswitch-switch

Sample Configuration

```
# PERSONALITY: vrs/vrs-g/none (default: vrs)
PERSONALITY=vrs

..

# ACTIVE_CONTROLLER: Primary controller IP. Only valid IP addresses will be
# accepted. To delete the controller comment out the ACTIVE_CONTROLLER
# variable below
ACTIVE_CONTROLLER=10.20.0.9
#
# STANDBY_CONTROLLER: Secondary controller IP. Only valid IP addresses
# will be accepted. To delete the controller comment out the STANDBY_CONTROLLER
# variable below
#STANDBY_CONTROLLER=
```

- Edit the configuration file /etc/nova/nova.conf
 - Vi /etc/nova/nova.conf

Sample Configuration

```
#ovs_bridge=br-int
ovs_bridge=alubr0
#linuxnet_ovs_integration_bridge=br-int
network_api_class=nova.network.neutronv2.api.API
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtGenericVIFDriver
```

- Restart the relevant services
 - service nova-compute restart
 - service nuage-openvswitch-switch restart

5.3.5.3 System Checks

- Verify that the VRS has successfully establish a connection with the VSC
 - `ovs-vsctl show`

Sample Output

```
#ovs-vsctl show

24ba0dab-01b4-4a80-9d2b-70b9390e74ec
Bridge "alubr0"
  Controller "ctrl1"
    target: "tcp:10.20.0.10:6633"
    role: master
    is_connected: true
  Port "tap6cb95198-06"
    Interface "tap6cb95198-06"
  Port "tapda17a992-c8"
    Interface "tapda17a992-c8"
  Port "tap5cb261e6-71"
    Interface "tap5cb261e6-71"
  Port "tap6c9e0c16-e9"
    Interface "tap6c9e0c16-e9"
  Port "alubr0"
    Interface "alubr0"
      type: internal
  Port "tapf1259223-73"
    Interface "tapf1259223-73"
  Port "tap62be8a47-51"
    Interface "tap62be8a47-51"
  Port "tapbdd02221-dc"
    Interface "tapbdd02221-dc"
  Port "tap2a3544e0-9e"
    Interface "tap2a3544e0-9e"
  Port svc-pat-tap
    Interface svc-pat-tap
      type: internal
  Port "tap89aa37de-27"
    Interface "tap89aa37de-27"
  Port "svc-r1-tap1"
    Interface "svc-r1-tap1"
  Port "tap3e9983a5-a4"
    Interface "tap3e9983a5-a4"
  Port "tapb1e4807f-7d"
    Interface "tapb1e4807f-7d"
  Port "svc-r1-tap2"
    Interface "svc-r1-tap2"
  Port "tapfe812e28-ac"
    Interface "tapfe812e28-ac"

ovs_version: "3.2.4-133-nuage"
```

- Common Issues:
 - Iptables rules not set correctly:
 - `iptables -F`
 - `iptables-save > ./iptables.rules.backup`
 - Below are the rules needed for the VRS if you choose not to disable iptables

Sample Configuration

```
-P INPUT DROP ;Default drop
-P FORWARD ACCEPT ;Anything forward allow
-P OUTPUT ACCEPT Openflow (port 6633)
    JSON/RPC (port 7406)
    TCP and SSL (port 7407)
    to VSC and VSD Stats (29090)

-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT ;Return traffic from above connections
-A INPUT -p icmp -j ACCEPT ;Ping
-A INPUT -p gre -j ACCEPT ;For receiving MPLS/GRE traffic (only for return type GRE)
-A INPUT -p tcp -m tcp --dport 22 -j ACCEPT ;For SSH, can add 23 for Telnet
-A INPUT -p udp -m udp --dport 4789 -j ACCEPT ;VXLAN
-A INPUT -i lo -j ACCEPT ;Allow any traffic from loopback interface
-A INPUT -j DROP ;Default drop
```

- Additional useful troubleshooting commands:
 - `ovs-appctl vm/port-show`
 - `ls -lrt /var/log/nova`
 - `vi /var/log/nova/nova-compute.log`
 - `ovs-appctl ofproto/show alubr0`
 - `virsh dumpxml instance-000000fe`
 - `service nova-compute restart`
 - `ovs-appctl vm/port-show`

5.3.6 VRS-G Installation

In order to provide gateway functionality between the overlay network and the underlay network, the Nuage Networks software gateway is required (i.e. VRS-G), which is identical to the VRS with the exception of a configuration change («personality» set to «VRS-G» instead of «VRS»).

However, the VRS-G does not install on an OpenStack compute node, it has to be on a separate VM with a minimum of two vports (an access port and a network port ; access provides connectivity to the underlay and network provides the overlay connectivity). There are additional configuration steps on the VSD that will also be listed later in this section

5.3.6.1 Deployment and Package Installation

- Unpack the package files:
 - `tar xvfz Nuage-openstack-3.2.4.tar.gz`
 - `tar xvfz Nuage-VRS-3.2.4-133-ubuntu.14.04.tar.gz`
- Start the installation process and follow the steps below:
 - `sudo apt-get update`
 - `sudo apt-get upgrade`
 - `apt-get install qemu-kvm libvirt-bin libjson-perl python-twisted-core vlan`
 - `apt-get remove openvswitch-common`
 - `dpkg -i nuage-openvswitch-common_3.2.4-133_amd64.deb` ; if you encounter “dependency problems”, then resolve using the following
 - `apt-get -f install`
 - `apt-get install python-setproctitle`
 - `dpkg -i nuage-openvswitch-common_3.2.4-133_amd64.deb`
 - `dpkg -i nuage-python-openvswitch_3.2.4-133_all.deb`
 - `dpkg -i nuage-openvswitch-switch_3.2.4-133_amd64.deb`

5.3.6.2 Configuration

- Edit the configuration file at /etc/default/openvswitch-switch to configure the systems personality (VRS or VRS-G) and the Controller's IP address (In this case we only have a one controller)
 - vi /etc/default/openvswitch-switch

Sample Output

```
# PERSONALITY: vrs/vrs-g/none (default: vrs)
PERSONALITY=vrs-g

..

# ACTIVE_CONTROLLER: Primary controller IP. Only valid IP addresses will be
# accepted. To delete the controller comment out the ACTIVE_CONTROLLER
# variable below
ACTIVE_CONTROLLER=10.20.0.9
#
# STANDBY_CONTROLLER: Secondary controller IP. Only valid IP addresses
# will be accepted. To delete the controller comment out the STANDBY_CONTROLLER
# variable below
#STANDBY_CONTROLLER=
```

5.3.6.3 System Checks

- Verify that the VRS has successfully establish a connection with the VSC
 - ovs-vsctl show

Sample Output

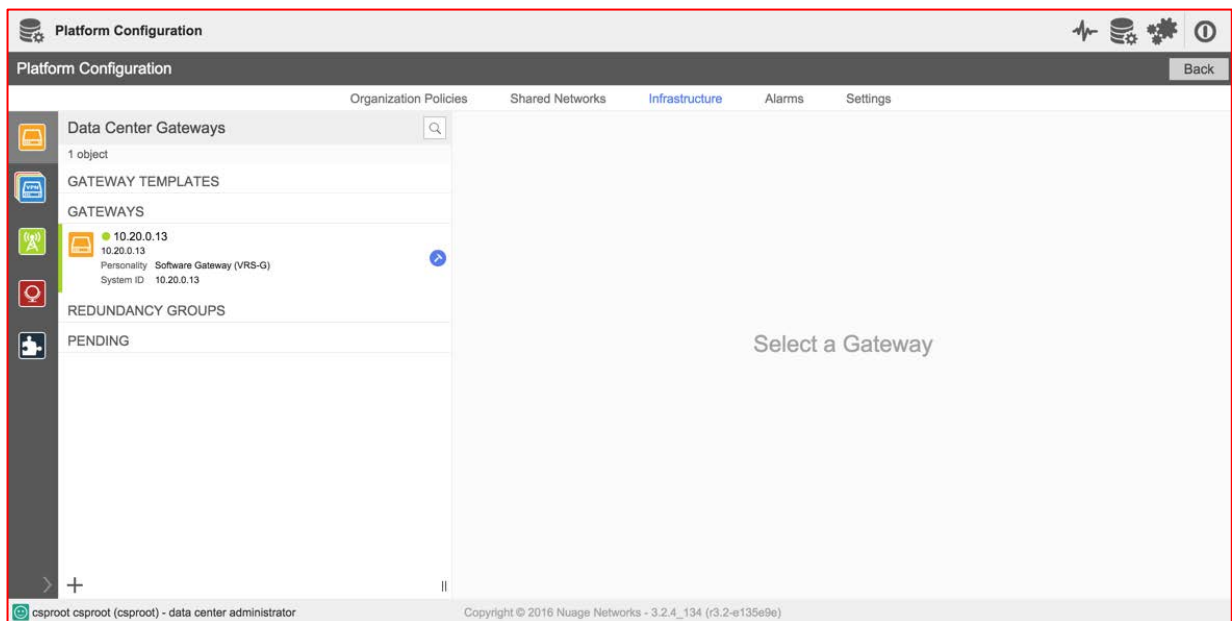
```
#ovs-vsctl show

e6f45d6c-97ca-461c-a14e-c7cc67b72d20
  Bridge "alubr0"
    Controller "ctrl1"
      target: "tcp:10.20.0.10:6633"
      role: master
      is_connected: true
    Port "alubr0"
      Interface "alubr0"
        type: internal
    Port "svc-rl-tap2"
      Interface "svc-rl-tap2"
    Port "svc-rl-tap1"
      Interface "svc-rl-tap1"
    Port "eth1"
      Interface "eth1"
    Port svc-pat-tap
      Interface svc-pat-tap
        type: internal
  ovs_version: "3.2.4-133-nuage"
```

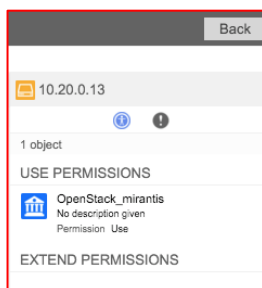

5.3.6.4 Gateway Setup on the VSD

The final steps in setting up the VRS-G include adding the appropriate permissions to the gateway, setting up the network and access port with the proper vlan settings. Then, adding the bridge interface/port to the appropriate network and a static route to enable VMs in the overlay from accessing the underlay.

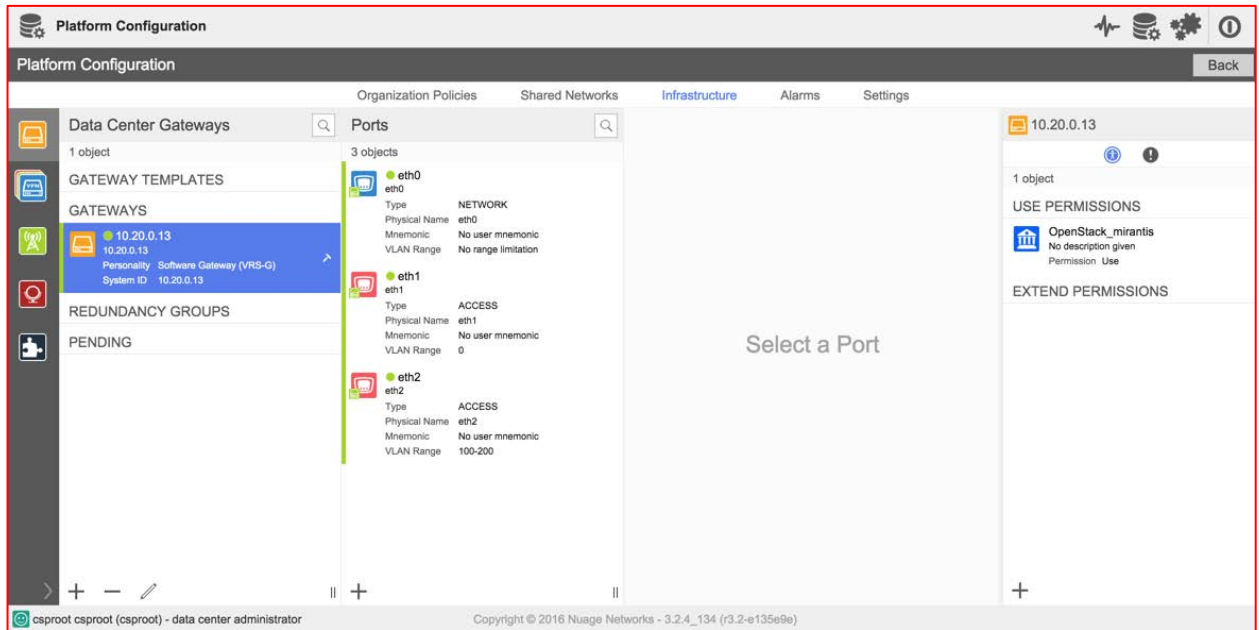
1. Proceed to the «Open Platform Configuration» section and then the «Infrastructure» tab and «Data Center Gateways» subsection, where you will see the «10.20.0.13» gateway in the «pending» section. Clicking on the blue arrow icon, will activate it and move it under the «gateways» section.



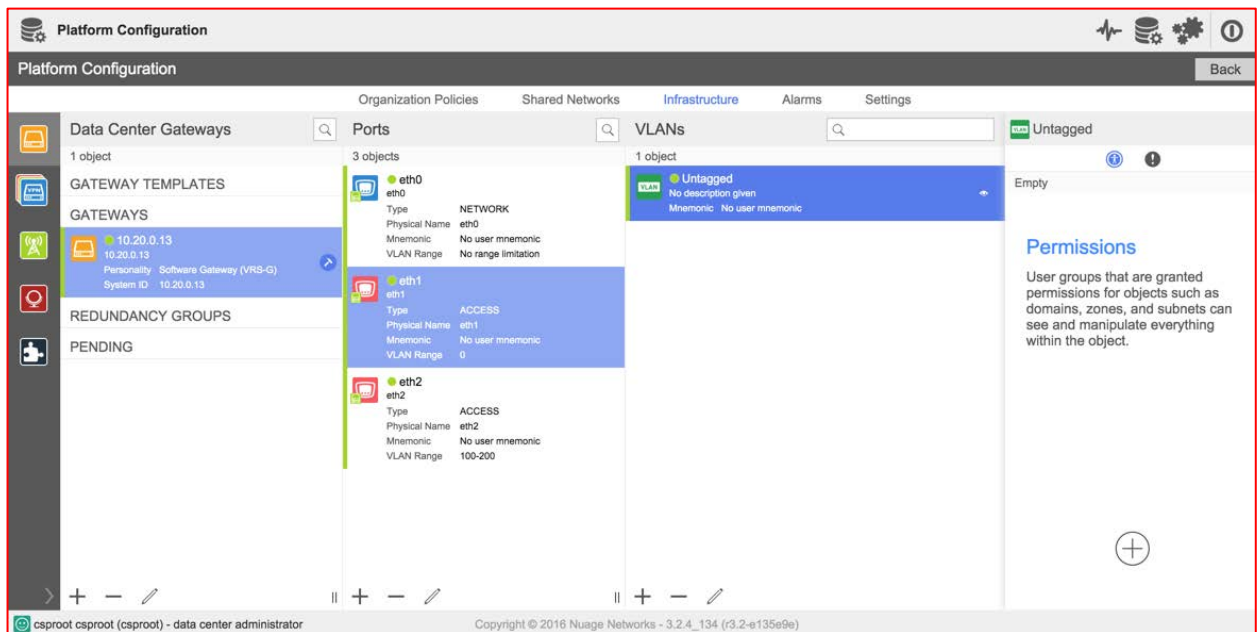
2. When selecting the «Gateway», ensure that the «user permissions» include the OpenStack organization configured for the OpenStack instance created.



3. Add or modify the «ports» associated with the selected gateway (ensuring that the naming matches the actual interfaces on the gateway itself). You can use the «+» icon at the bottom to add an interface and/or double click on the port to edit its content.



4. Next, enter the VLAN tags (or use untagged) for the «Access» interface(s). In our example we used eth1 access with untagged setting



Organization Policies

Ports

3 objects

Name	Type	Physical Name	Mnemonic	VLAN Range
eth0	NETWORK	eth0	No user mnemonic	No range
eth1	ACCESS	eth1	No user mnemonic	0
eth2	ACCESS	eth2	No user mnemonic	100-200

Edit NS Port

Name:

Description:

Physical Name:

Type:

VLAN Range:

Mnemonic:

If you set a value for mnemonic, users who can see the port will only see this value as a name. Other attributes will be masked. If you leave this field blank, they will see everything.

VLANs

1 object

Untagged

No description given

Mnemonic: No user mnemonic

Edit VLAN

VLAN Number:

Description:

Mnemonic:

If you set a value for mnemonic, then the user who can see the VLAN will only see this value as a name. Other attributes will be masked. If you leave this field blank, users will see everything.

NOTE

Depending on your specific setup, the details of the VLANs may differ, but once the setup is complete. You will need to ensure that you set the type and vlans on the actual port on the VRS-G VM

Examples:

- `nuage-vlan-config mod eth1 access 0 ; untagged`
- `nuage-vlan-config mod eth2 access 100-200 ; tagged VLAN range`

5. In the main window, select the «networks» tab in order to add the bridge vport to the «underlay» subnet. If you have not done that already, select the «subnet» icon from the «library» (bottom right of screen) and drop it in anywhere in the same L3 domain (the same zone was used for simplicity).

Subnet

Name:

Network:

Gateway:

- Now repeat the process for the «Bridge Vport», where you will be prompted to select the DC Gateway, Port and VLAN (example shown below Before / After)

VPort

Name

Gateway

Data Center Gateways

⛔
No Gateway

Port

No Port

VLAN

No VLAN

Create

VPort

Name

Gateway

Data Center Gateways

🔗

10.20.0.13
10.20.0.13

Port

🔗

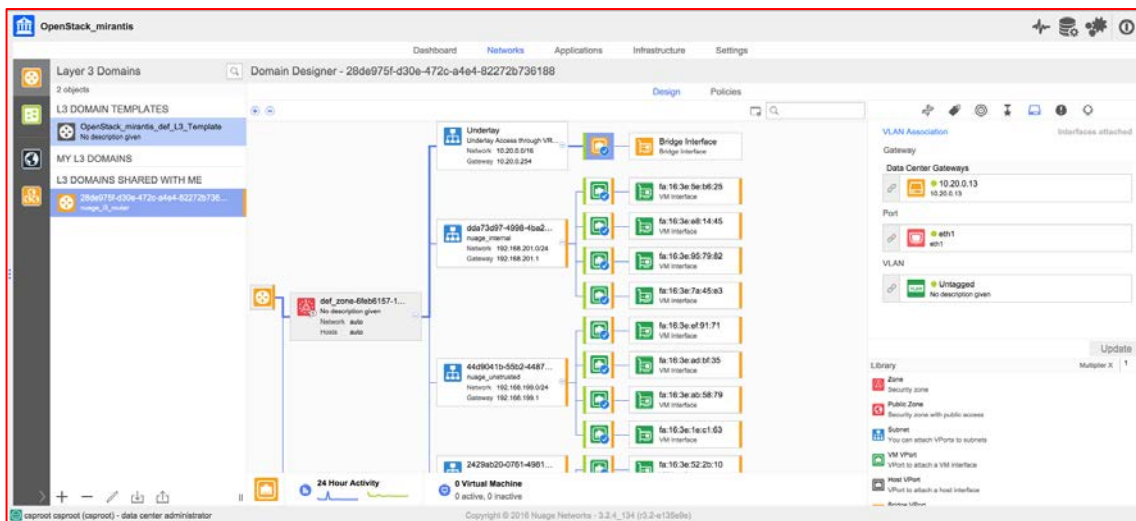
eth2
eth2

VLAN

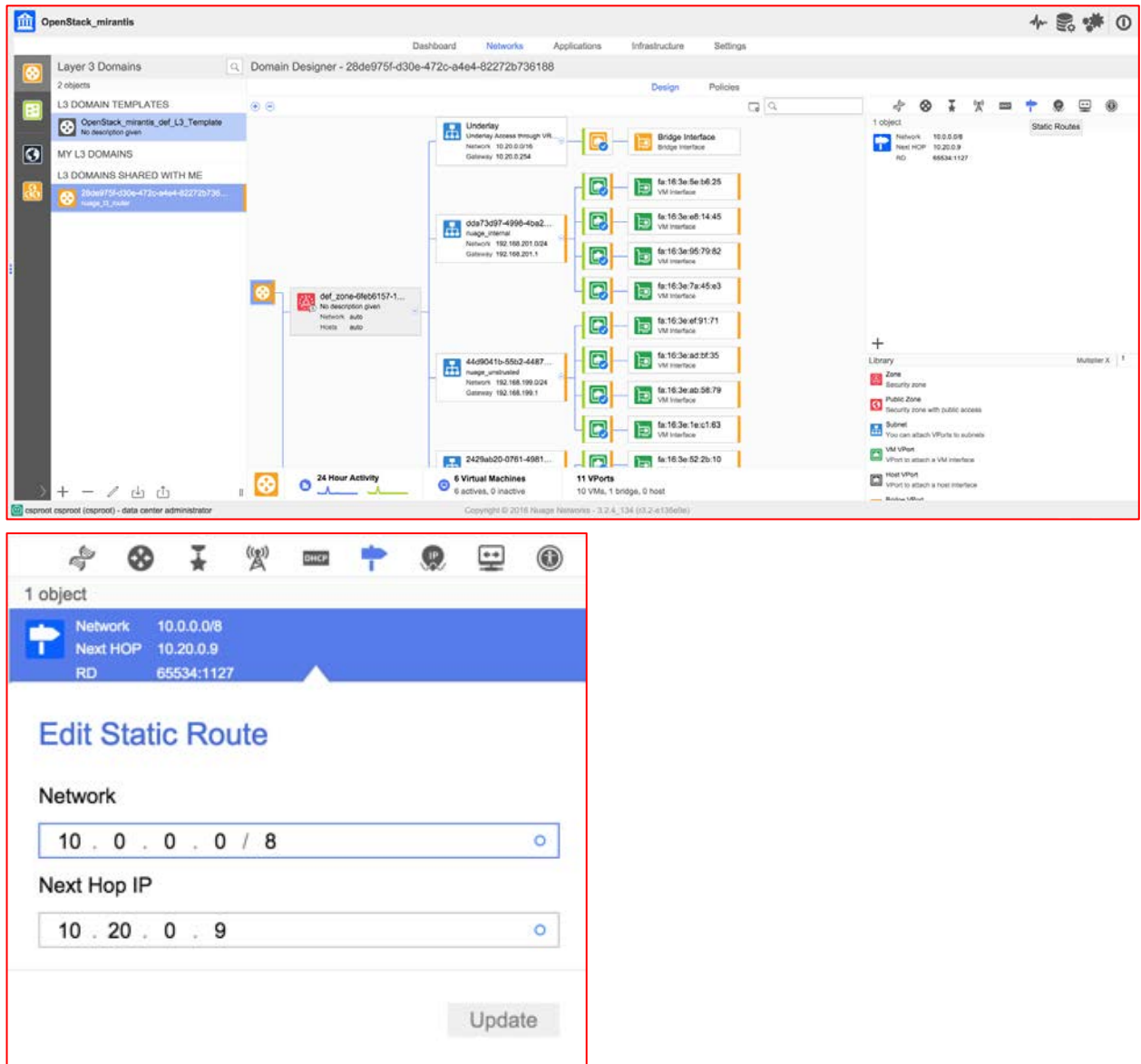
🔗

VLAN # 101
No description given

Create



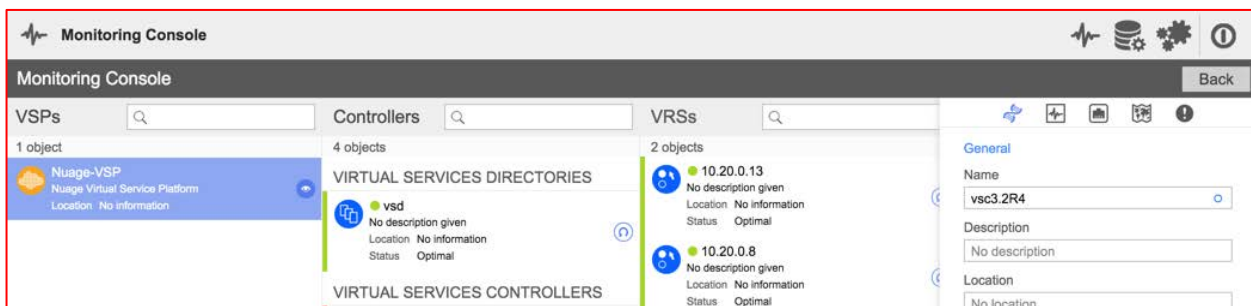
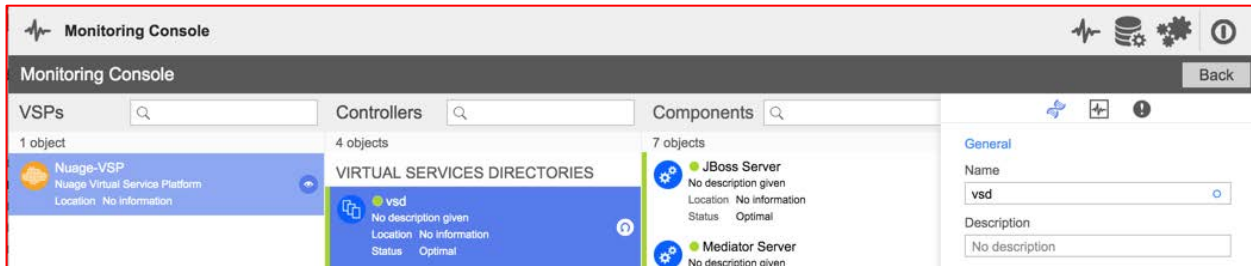
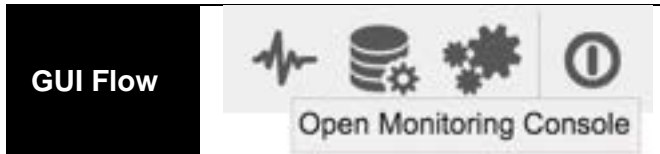
7. Select the domain icon, then the static route icon (blue road-sign icon top right), then add a static route to be used by VMs in the overlay network so that they can use an appropriate next-hop for the underlay addresses (in our example we used the 10.20.0.9 as an example)



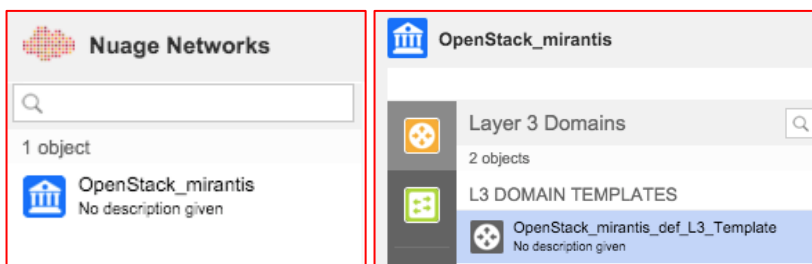
8. This concludes the setup of the VRS-G from the Nuage Networks side. You must also ensure that any underlay devices that need to communicate with the overlay, use the appropriate next-hop (10.20.0.254)

5.3.7 Installation & Setup Validation

To ensure that the Nuage Networks system is operating normally, you can use the VSD GUI to monitor the status of the VSD and all running processes as well as VSC and VRS.

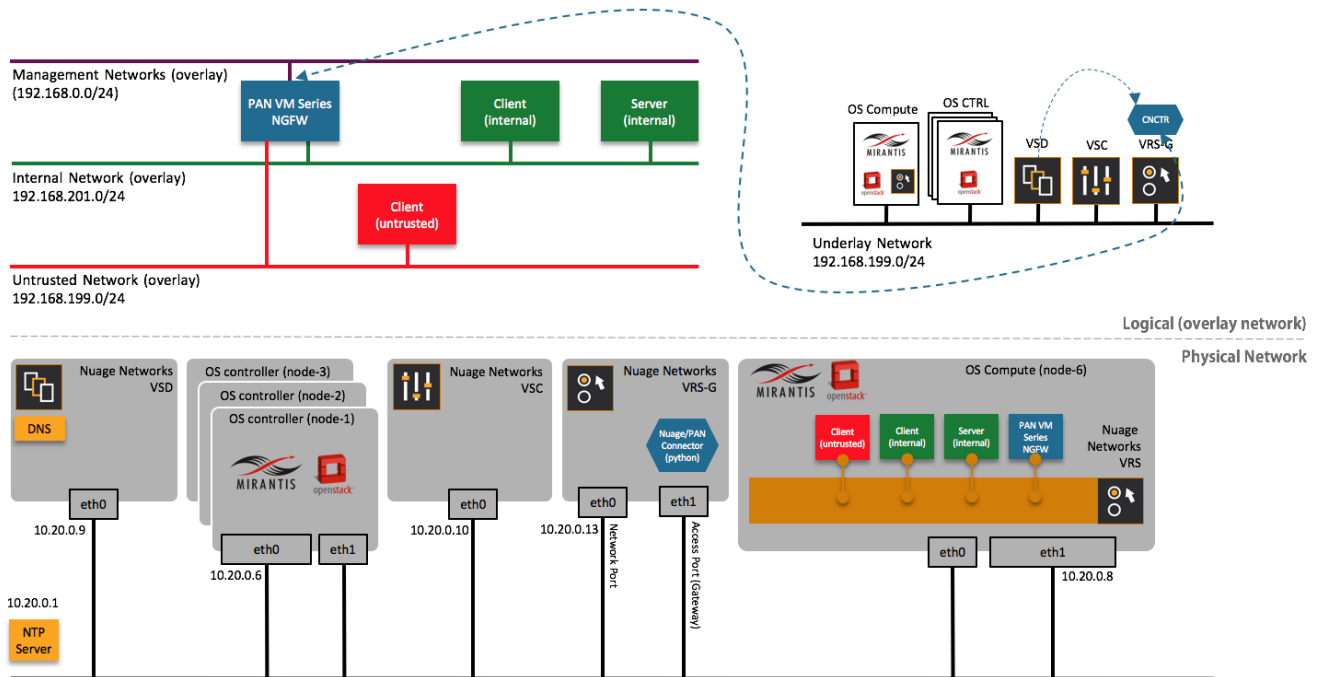


You should also see the «Enterprise» that the OpenStack controller pushed down to the Nuage Networks VSD «OpenStack_mirantis» as well as an L3 domain template. If they exist, then the installation was successful.

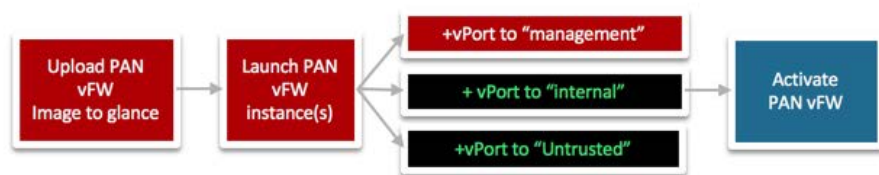


5.4 PAN vFW Setup & Configuration

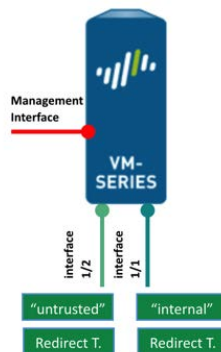
This section will cover the firewall configuration and setup needed for the deployment of the Palo Alto Networks VM series firewall and insertion into the data path using Mirantis OpenStack and Nuage Networks VSP. In our testing, the virtual firewall is set for “Virtual Router” mode. The diagram below shows the location of the virtual firewall (from an infrastructure perspective)



The initial setup and configuration of the Palo Alto Networks VM-series firewall can be done through the web dashboard (available using HTTPS from the PA-VM management interface) or the CLI interface. We will utilize the web dashboard to complete the setup and configuration. The deployment process is outlined below.



Below is a diagram that shows how the vFW is connected from an overlay networks prospective.



5.4.1 Deploying the PAN vFW

The PAN vFW (VM series) will be deployed using the OpenStack GUI. The VM series image should have been uploaded to glance so that it can be used to launch multiple vFW instances. In this example, we will use the VM series KVM-v7.0 image («m1.medium» flavor was used to meet the minimum requirements of the firewall). Please contact a Palo Alto Networks partner/reseller to download the software.

NOTE

It is important to be able to identify the management IP address and configure that correctly using CLI on the PAN vFW (Please refer to the PAN vFW Administrator's Guide for steps on how to complete that – for new users, we do recommend that you deploy the vFW with the management interface only first, then add additional interfaces after using the OpenStack CLI)

5.4.1.1 Launch PANW vFW instance in OpenStack

Launch a VM using the PA-VM-KVM-7.0.1 image and attach it to the three networks (nuage_management, nuage_internal and nuage_untrusted)

Launch Instance

Details * Access & Security Networking Application Post-Creation Advanced Options

Availability Zone

nova

Instance Name *

PANW-NGFW-01

Flavor *

m1.medium

Instance Count *

1

Instance Boot Source *

Boot from image

Image Name *

PA-VM-KVM-7.0.1 (1.4 GB)

Specify the details for launching an instance.

The chart below shows the resources used by this project in relation to the project's quotas.

Flavor Details

Name

m1.medium

VCPU

2

Root Disk

40 GB

Ephemeral Disk

0 GB

Total Disk

40 GB

RAM

4,096 MB

Project Limits

Number of Instances

0 of inf Used

Number of VCPUs

0 of inf Used

Total RAM

0 of inf MB Used

Cancel Launch

Launch Instance

Details * Access & Security Networking Application Post-Creation Advanced Options

Selected networks

NIC-1 nuage_management

NIC-2 nuage_internal

NIC-3 nuage_untrusted

Available networks

external_network

overlay_network

Select networks for your instance.

Chose Application instead of Networks.

Cancel Launch

Instances

Instance Name PANW Filter Launch Instance Terminate Instances More Actions

	Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
<input type="checkbox"/>	PANW-NGFW-02	PA-VM-KVM-7.0.1	nuage_untrusted 192.168.199.5 nuage_internal 192.168.201.9 nuage_management 192.168.202.5	m1.medium	-	Active	nova	None	Running	0 minutes	Create Snapshot
<input type="checkbox"/>	PANW-NGFW-01	PA-VM-KVM-7.0.1	nuage_untrusted 192.168.199.2 nuage_internal 192.168.201.4 nuage_management 192.168.202.4	m1.medium	-	Active	nova	None	Running	1 month	Create Snapshot

Displaying 2 items

v2.1

48

Tip

Using CLI to add interfaces to the PAN vFW in OpenStack

- neutron port-create nuage_untrusted
- neutron port-create nuage_untrusted
- nova interface-attach --port-id bdd02221-dc1c-4af5-b0e0-9f3511188525 PANW-NGFW-01

5.4.1.2 Activate PANW vFW

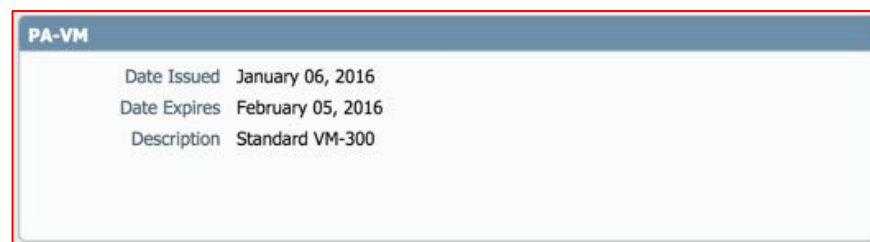
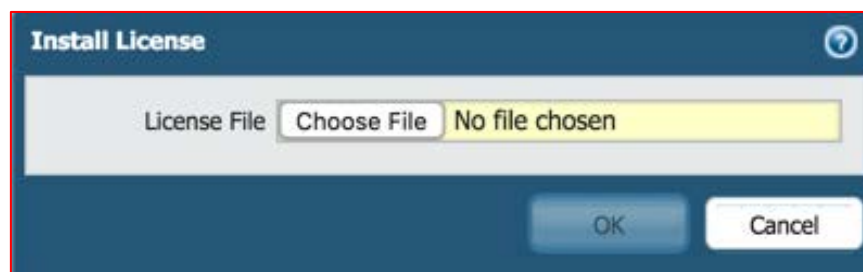
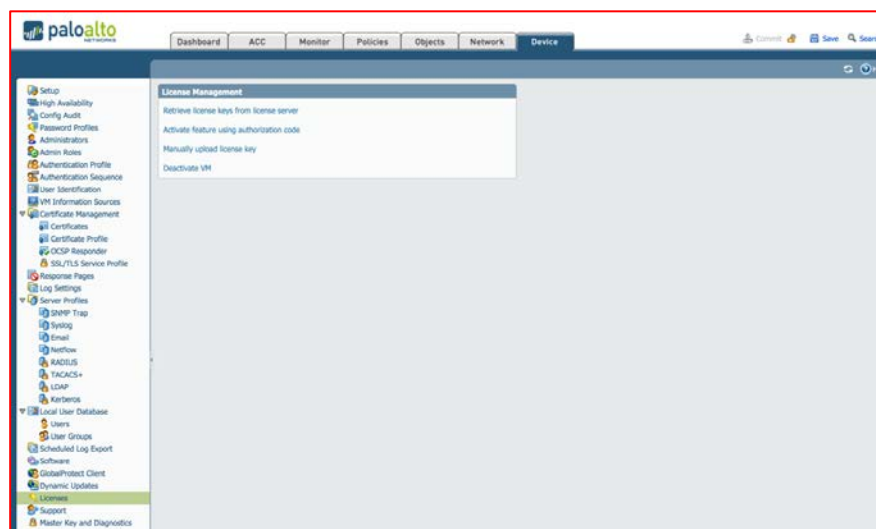
Once the PAN vFW instances have booted up, you can use the GUI through the management interface to complete the activation and configuration process. The default username/password is admin/admin



To activate the PAN vFW, the licensing server uses the UUID and the CPU ID of the virtual machine to generate a unique serial number for the VM-Series firewall. The capacity auth-code in conjunction with the serial number is used to validate your entitlement. (Refer to Palo Alto networks document [here](#) for instructions)

From the “Dashboard” tab, copy the CPU ID and UUID and use these on the licensing server to obtain your license file (for offline activation). Then go to the, **Device/Licenses** tab to manage device licenses. After uploading the license, the device will need to be rebooted





Now, that the PAN vFW has been activate, you can now insert the PAN vFW in to the data path first.

5.4.2 Nuage Networks VSP insertion of PAN vFW

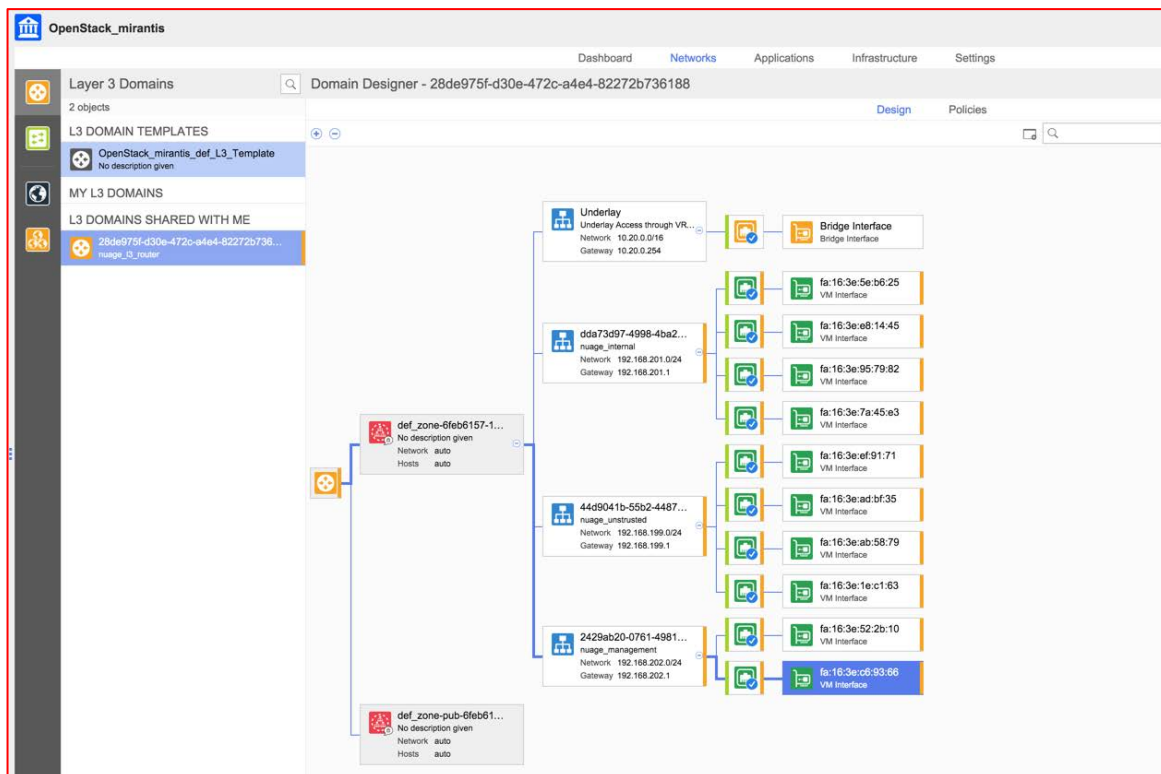
In this section, we will setup the overlay network so that the PAN vFW is inserted into the data path between the «internal» network and the «untrusted» network.



5.4.2.1 Disable Anti-spoofing

On the Nuage Networks VSD, identify the target vports (internal and untrusted) for the PAN vFW (record their UUIDs) and ensure that «Spoofing» is enabled on these ports so that redirected traffic can be processed correctly (You can use MAC addresses or IP addresses that have been assigned and shown from the the OpenStack Horizon view)


- Select the vPort icon in «Design» tab, then edit «Allow Source Address Spoofing» field in «General» tab to be «enabled»; repeat for all vFW ports that will receive traffic

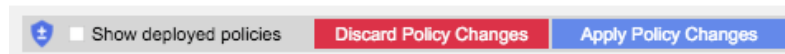


5.4.2.2 Configure Redirection Targets

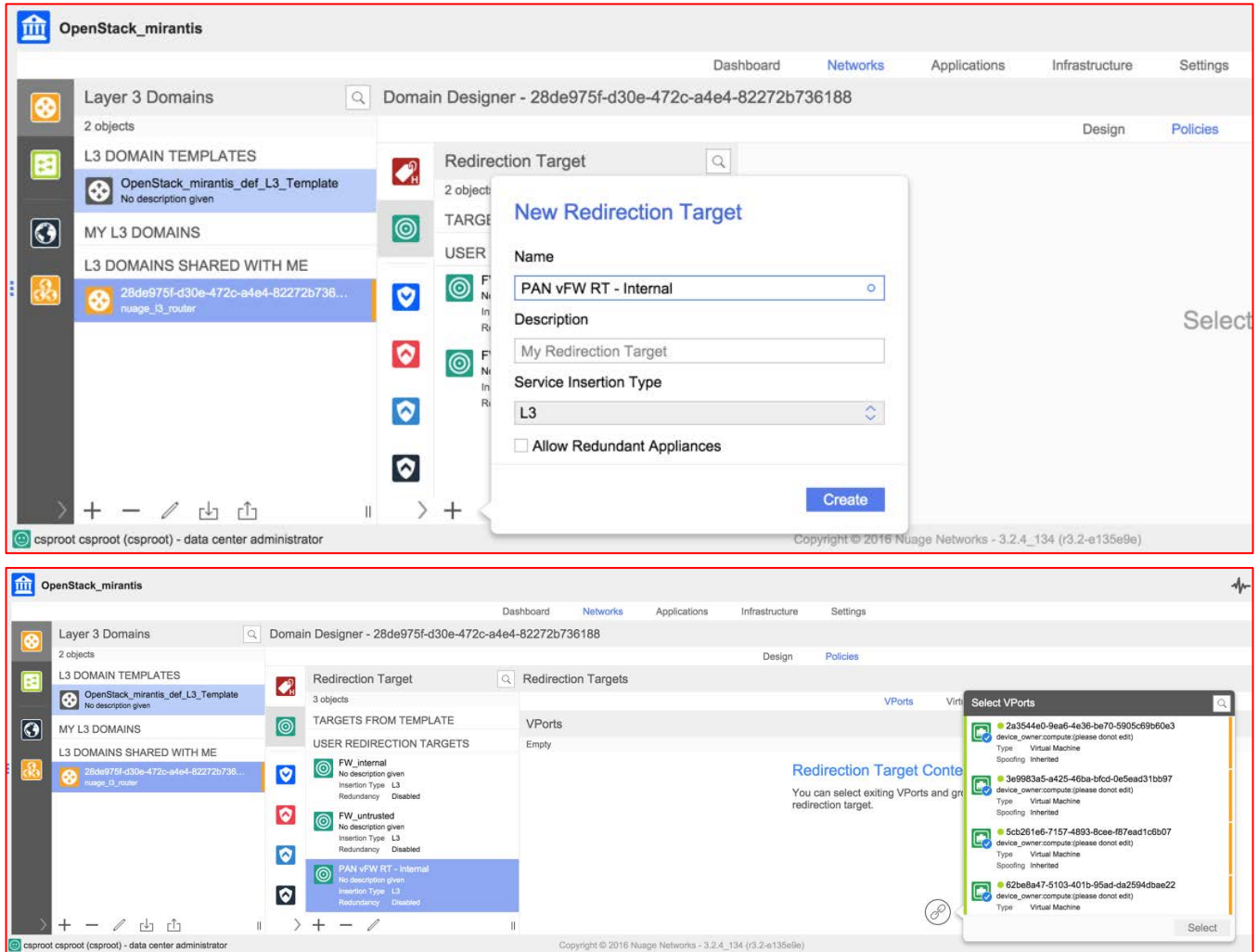
Create Redirection Targets for all PAN vFW vports and select the «insertion type» associated with each one and the redundancy support.

NOTE

When modifying security policies on Nuage Networks VSD, you must enter “Edit” mode by clicking the “lock” on the screen , make your changes and then “Apply” or “Discard” your changes.



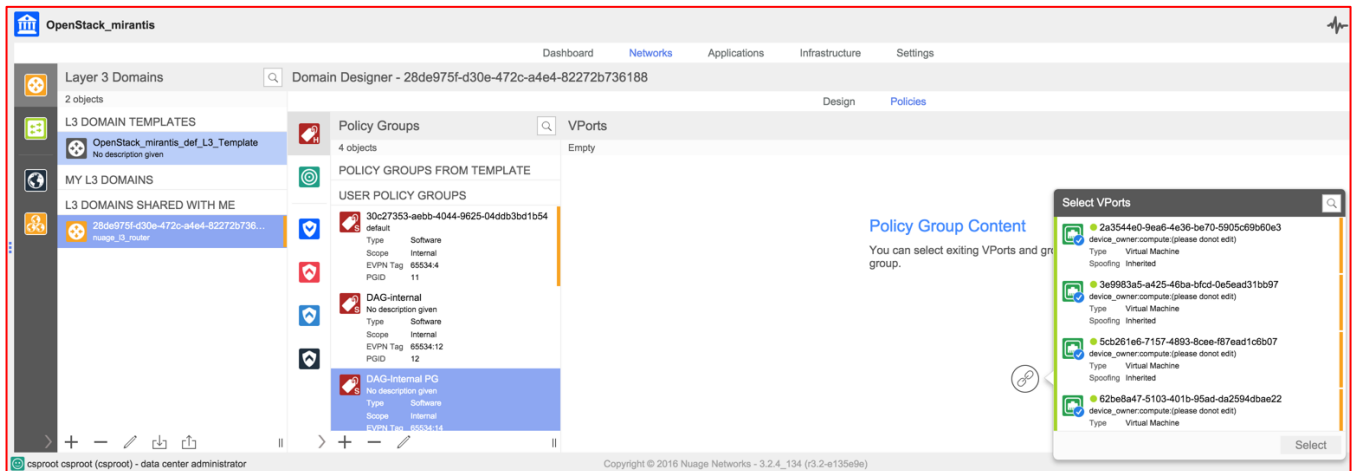
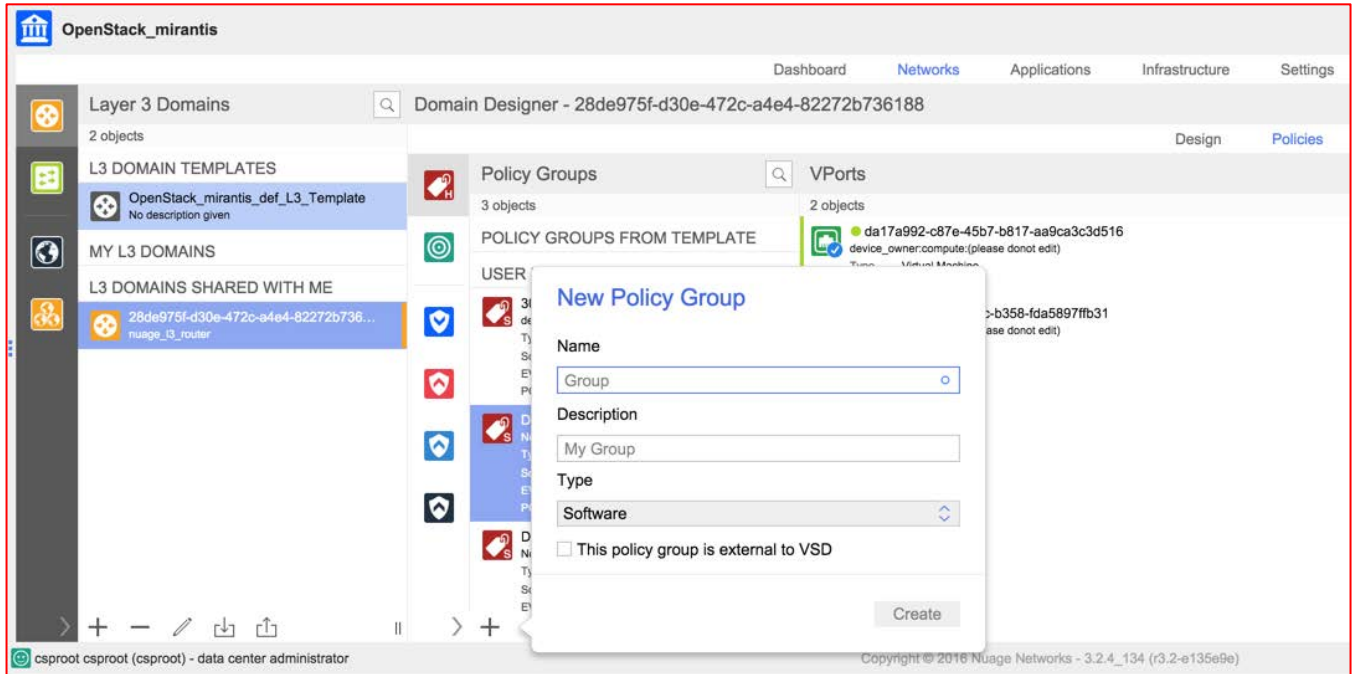
- In the «Policies» tab, select the «Redirection Target» objects section and «+» create a new Redirection Target; set «Service Insertion Type» to L3 ; uncheck the «Allow Redundant Appliance»
- Enter policy «Edit» mode
- Create one for each PAN vFW port («Internal» and «untrusted»)
- For each «Redirection Target», add the relevant PAN vFW vport
- «Accept» changes



5.4.2.3 Create Policy Groups

As part of the Micro-segmentation capability of the Nuage Networks VSP, «Policy Groups» are used to logically group VMs in the overlay networks so that security policy can be easily applied between the different segments. Please refer the Nuage Networks VSP User Guide for more details on Micro-segmentation and the use of Security Policies. For this test, we will create an «internal PG» and an «untrusted PG». The PG maps one-to-one with the Subnet, but it is not limited to that. In real DC deployments, the PG can group vports from different sub-networks.

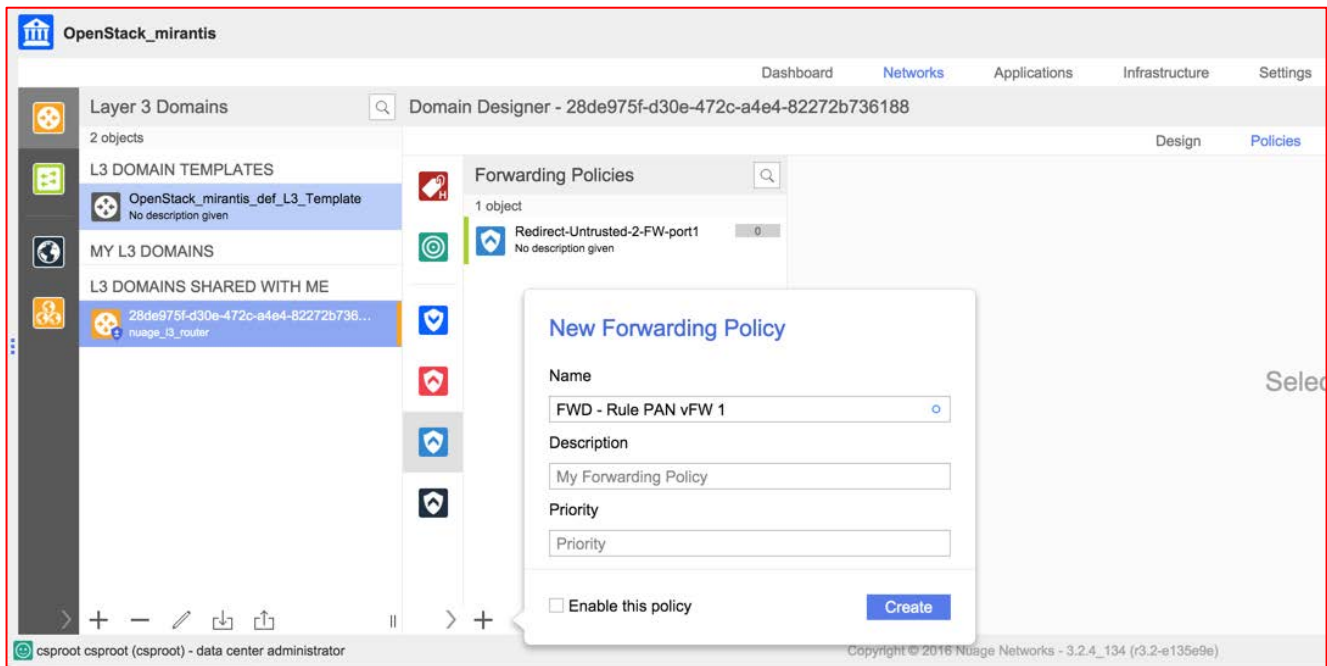
- In the «Policies» tab, select the «Policy Groups» objects section and «+» create a new Policy Group; set «Type» to Software ; uncheck the «This policy group is external to VSD»
- Create one for each logical group («Internal» and «untrusted»)
- For each «Policy Group», add the relevant VM ports that belong to that logical group (do not include the vFW ports, these are groups of Client/Server VMs)



5.4.2.4 Create & Set Forwarding Rules

The final step in the process of service insertion is to add the «Forwarding Policies» to redirect specific traffic flows to the PAN vFW. Typically, this is a subset of the traffic that is allowed to flow between segments. The embedded security features within the Nuage Networks VSP provide ACLs that will allow/deny traffic. However, customers may want the «allowed» traffic to flow to a security device for further inspection (L4-L7). In this section, we will create such rules.

- In the «Policies» tab, select the «Forwarding Policies» objects section and «+» create a new Forwarding Policy; uncheck the «Enable this policy» for the time being
- Enter policy «Edit» mode
- Create «Security Policy» entries that match the specific traffic that needs to be redirected to the vFW
 - Enter «Forward» match rule ; examples shown are to forward all traffic
 - Enter «Reverse» match rule ; examples shown are to forward all traffic
- Edit the Forwarding Policy to «Enable» it
- «Accept» changes



Edit Forwarding Policy Entry

Name:

Priority:

☐ Enable flow logging
☐ Enable statistics collection

Traffic Type

Ether Type: Source Port:
 Protocol: Destination Port:
 DSCP Marker: Source IP Match:

Traffic Path

Origin Location: Destination Network:
 Policy Group: Policy Group:
 DAG-untrusted No description given DAG-internal No description given

Traffic Management

Action: Forwarding Class Override:
 FW_untrusted Description Uplink Preference:

Edit Forwarding Policy Entry

Name:

Priority:

☐ Enable flow logging
☐ Enable statistics collection

Traffic Type

Ether Type: Source Port:
 Protocol: Destination Port:
 DSCP Marker: Source IP Match:

Traffic Path

Origin Location: Destination Network:
 Policy Group: Policy Group:
 DAG-internal No description given DAG-untrusted No description given

Traffic Management

Action: Forwarding Class Override:
 FW_internal Description Uplink Preference:

5.4.3 Establishing Nuage Networks VSP «sync» with PANW vFW

As part of the integration that has been completed between Palo Alto Networks vFW and Nuage Networks VSP, a management connector has been developed to synchronize objects between the two solutions. This is an additional package that can be provided by Palo Alto Networks or Nuage Networks. It can be setup anywhere in the network (i.e. overlay or underlay) as long as it has IP reachability to the Nuage Networks VSP and the PAN vFW and Panorama (if applicable). In our example, we deployed it on the Nuage Networks VRS-G. The steps to deploy the nuage-connector are listed below (please contact Nuage Networks representatives of Palo Alto Networks' to download this package):

1. Install required linux packages

```
[root@pan-sync ~]# yum install python-virtualenv python-pip gcc supervisor
```

2. Create a python virtualenv

```
[root@pan-sync ~]# cd /root
[root@pan-sync ~]# virtualenv pn-env
[root@pan-sync ~]# ./pn-env/bin/activate

(pn-env)[root@pan-sync ~]#
```

3. Copy the PAN “nuage-connector” tarball to the machine and extract it

```
(pn-env)[root@pan-sync ~]# cd /root
(pn-env)[root@pan-sync ~]# tar -xzf pan-nuage.tar.gz

pan-nuage/LICENSE
pan-nuage/nuage.py
pan-nuage/nuage.pyc
pan-nuage/pan-nuage.py
pan-nuage/README.md
```

4. Install python dependencies

```
(pn-env)[root@pan-sync ~]# pip install pan-python eventlet
```

5. Configure supervisord to manage the PAN “nuage-connector”

```
(pn-env)[root@pan-sync ~]# vi /etc/supervisord.d/pan-nuage.ini
```

```
[program:pan-nuage]
environment=PATH="/root/pn-env/bin"
command=/root/pan-nuage/pan-nuage.py --nuage-vsd 10.20.0.9:8443 --nuage-login csproot --nuage-
password csproot --nuage-organization csp --nuage-domain f75f73a8-4d5f-457c-896f-9dd3924aab8d
--pan-admin admin --pan-password admin --vmseries 192.168.202.4
stopwaitsecs=60
autostart=true
autorestart=true
redirect_stderr=true
killasgroup=true
```



```
stdout_logfile=/var/log/pan-nuage.log
```

6. Add routes to access the PAN vFW management IP address

```
(pn-env)[root@pan-sync ~]# route add -net 192.168.201.0/24 gw 10.20.0.254
(pn-env)[root@pan-sync ~]# route add -net 192.168.202.0/24 gw 10.20.0.254
```

7. Start supervisord and check PAN “nuage-connector” status

```
(pn-env)[root@pan-sync ~]# service supervisord start
(pn-env)[root@pan-sync ~]# supervisorctl status pan-nuage
pan-nuage                                RUNNING    pid 21031, uptime 0:00:19
```

8. Monitoring sync activity for troubleshooting

```
(pn-env)[root@pan-sync ~]# tail -f /var/log/pan-nuage.log
```

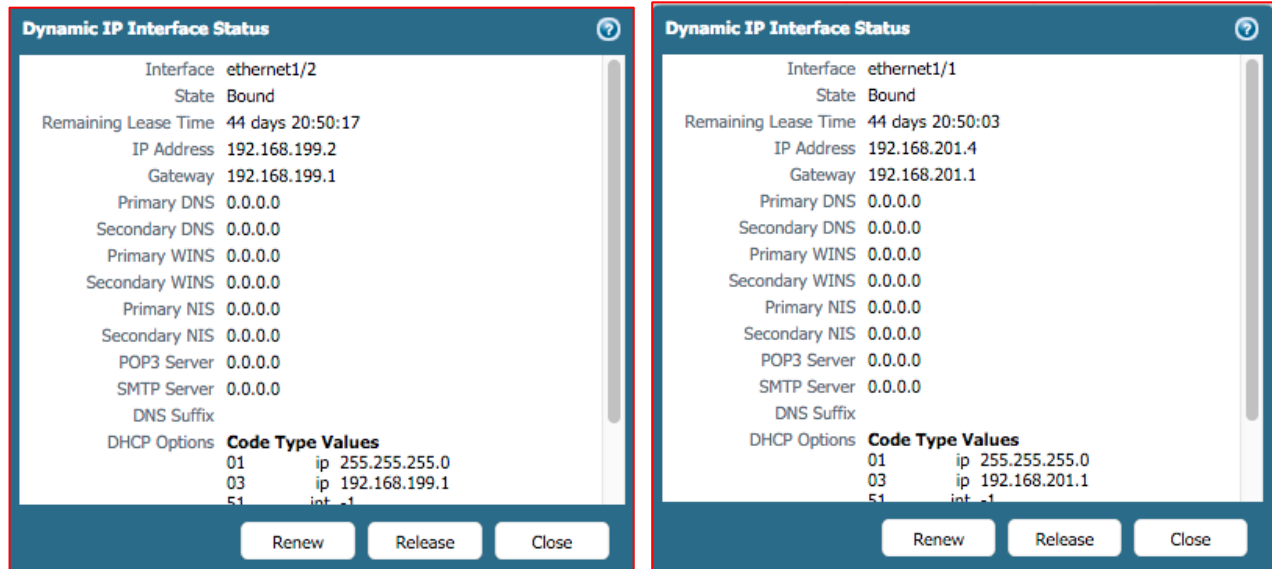
5.4.4 PAN vFW setup

The final step is to complete the configuration of the PAN vFW and add Dynamic Address Groups (DAGs) as well as security policies. The workflow is as follows:

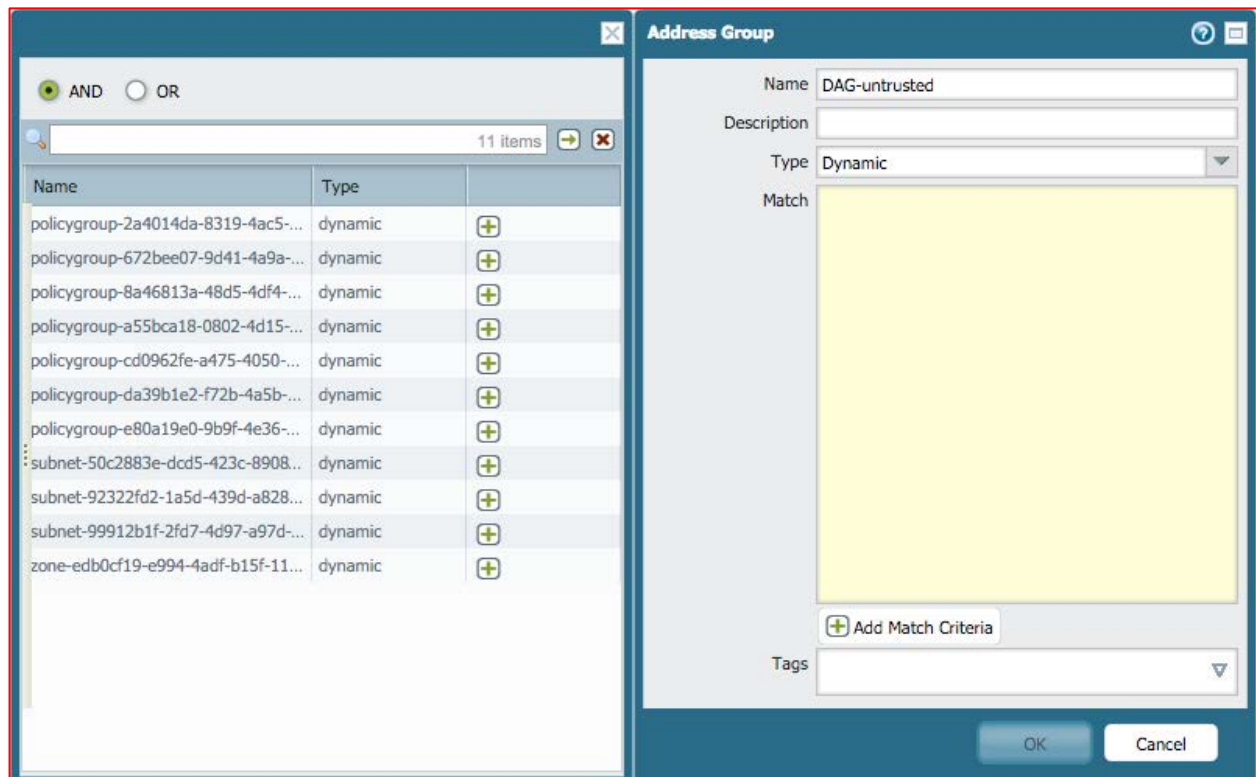


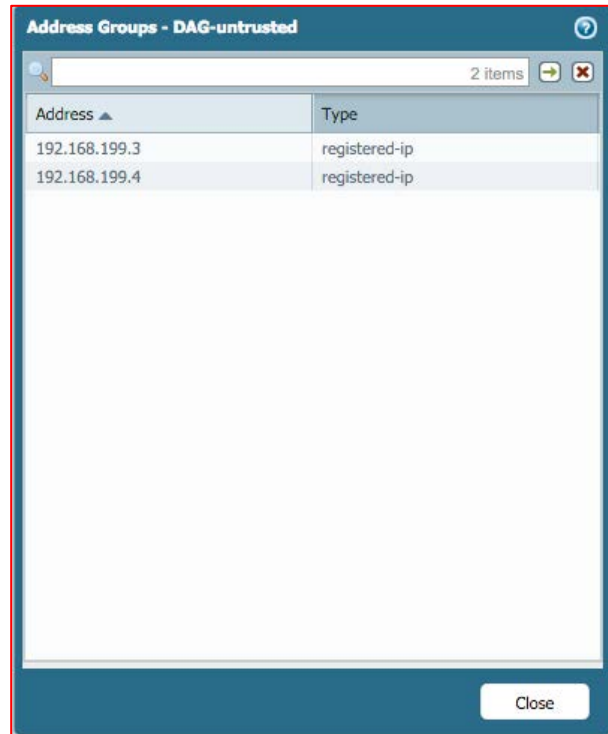
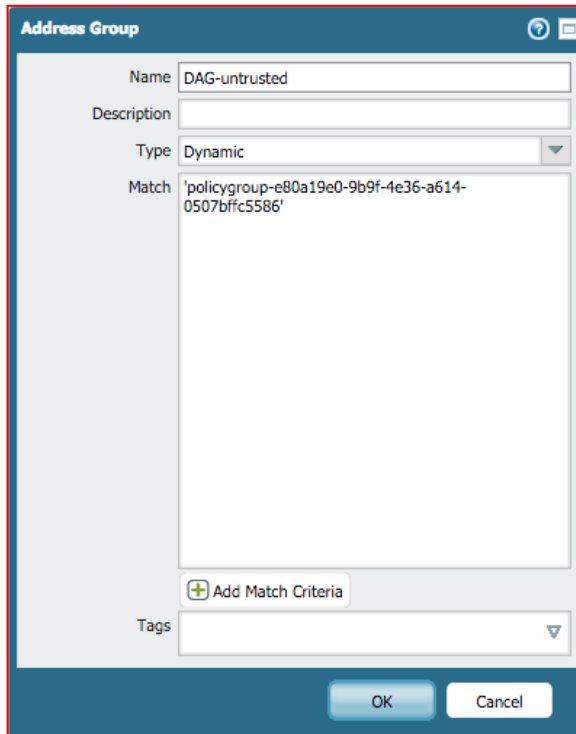
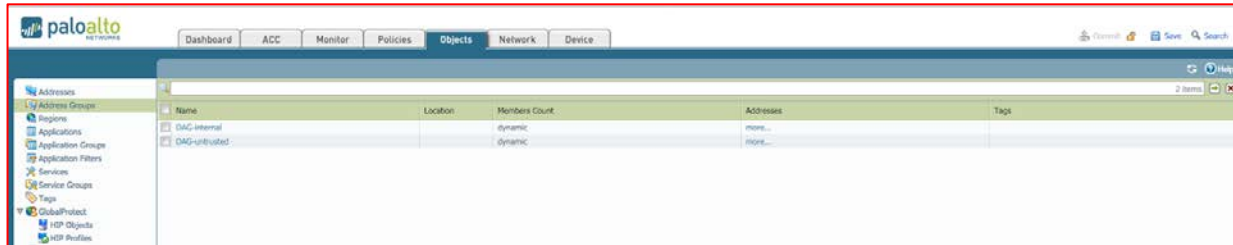
- In «Networks > Ethernet» configure Ethernet 1/1 so that the interface type is set to «Layer 3», the virtual router to «default», IPV4 «static or DHCP», «enable» it and the management profile. Repeat for Ethernet 1/2
- Confirm that the IP address is correctly assigned.

Interface	Interface Type	Management Profile	Link State	IP Address	Virtual Router	Tag	VLAN / Virtual-Wire	Security Zone	Features	Comment
ethernet1/1	Layer3	NUAGE - Management	Dynamic-DHCP Client		default	Untagged	none	AppZone		
ethernet1/2	Layer3	NUAGE - Management	Dynamic-DHCP Client		default	Untagged	none	AppZone		
ethernet1/3			none		none	Untagged	none	none		
ethernet1/4			none		none	Untagged	none	none		



- Under "Objects", add a new dynamic address groups and "add Match Criteria" then select the Nuage Networks VSD objects that are mapped to it (in our example we will map Policy Groups, ensure that the UUID is that of the "untrusted" PG/DAG. Once the mapping is complete, the system will automatically update the registered IP addresses, so confirm that the IP address of the vPorts that belong to the Nuage Networks VSD PG are present

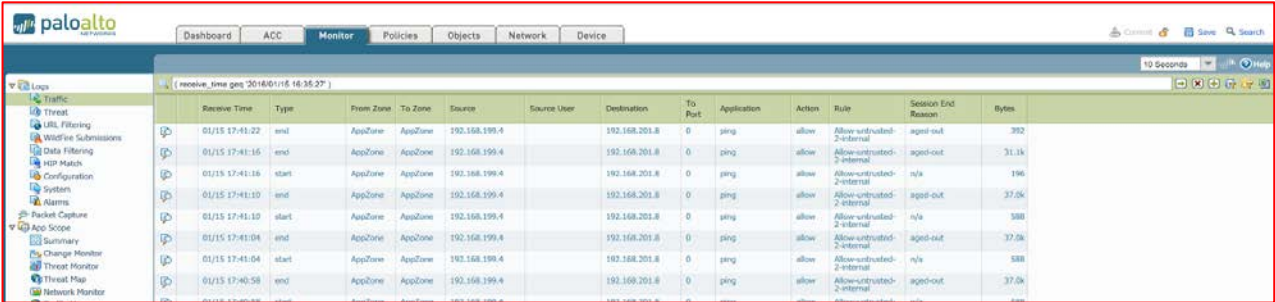




- Define security policies under “Policies” using the DAGs created earlier as the “Source” and “Destination”. In our example, we created rules to allow / deny rules. More details in the testing section about the specific rules applied. When completed, you must “commit” and “save” your changes



- At this point, any change to the network (adding VMs, deleting VMs) will automatically get pulled by the connector from VSD and pushed to the PAN vFW automatically, where security rules will be enforced for the redirected traffic. You can use the “Monitor” tab to log all of the activities (ensure that the policy defined has logging enabled)

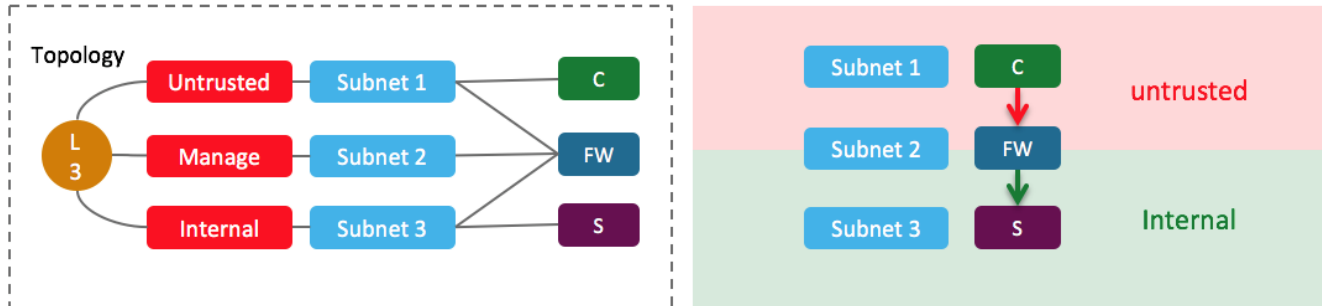


The screenshot shows the Palo Alto Networks Panorama interface. The top navigation bar includes 'Dashboard', 'ACC', 'Monitor' (selected), 'Policies', 'Objects', 'Network', and 'Device'. On the left, a sidebar lists various tools like Traffic, Threat, URL Filtering, Wildfire Submissions, Data Filtering, HSP Match, Configuration, System, Alarms, Packet Capture, App Scope, Summary, Change Monitor, Threat Monitor, Threat Map, and Network Monitor. The main area displays a traffic log table with the search filter '(receive_time geq 2016/01/15 16:35:27)'. The table has columns for Receive Time, Type, From Zone, To Zone, Source, Source User, Destination, To Port, Application, Action, Rule, Session End Reason, and Bytes. The log shows several ping requests from 192.168.199.4 to 192.168.201.8, all with an 'allow' action and 'aged-out' session end reason.

Receive Time	Type	From Zone	To Zone	Source	Source User	Destination	To Port	Application	Action	Rule	Session End Reason	Bytes
01/15 17:41:22	end	AppZone	AppZone	192.168.199.4		192.168.201.8	0	ping	allow	Allow-untrusted-2-internal	aged-out	392
01/15 17:41:16	end	AppZone	AppZone	192.168.199.4		192.168.201.8	0	ping	allow	Allow-untrusted-2-internal	aged-out	31.1k
01/15 17:41:16	start	AppZone	AppZone	192.168.199.4		192.168.201.8	0	ping	allow	Allow-untrusted-2-internal	n/a	196
01/15 17:41:10	end	AppZone	AppZone	192.168.199.4		192.168.201.8	0	ping	allow	Allow-untrusted-2-internal	aged-out	37.0k
01/15 17:41:10	start	AppZone	AppZone	192.168.199.4		192.168.201.8	0	ping	allow	Allow-untrusted-2-internal	n/a	588
01/15 17:41:04	end	AppZone	AppZone	192.168.199.4		192.168.201.8	0	ping	allow	Allow-untrusted-2-internal	aged-out	37.0k
01/15 17:41:04	start	AppZone	AppZone	192.168.199.4		192.168.201.8	0	ping	allow	Allow-untrusted-2-internal	n/a	588
01/15 17:40:58	end	AppZone	AppZone	192.168.199.4		192.168.201.8	0	ping	allow	Allow-untrusted-2-internal	aged-out	37.0k

5.5 Testing

Deployment scenarios will vary depending on the use-case, so we will focus on a test that utilizes the most common elements to illustrate the integration. The deployment of the test environment is shown below



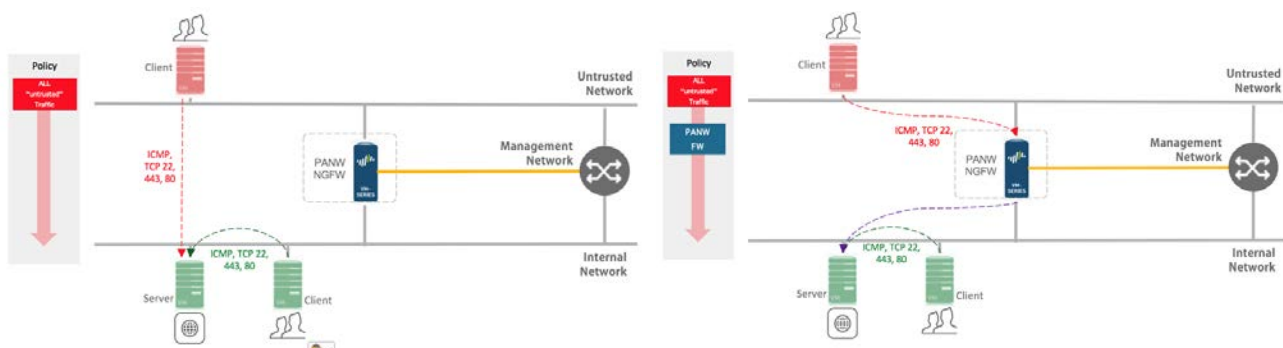
5.5.1 Test Tools

The following tools will be used during the testing

- SSH, Ping & wget
- An HTTP server

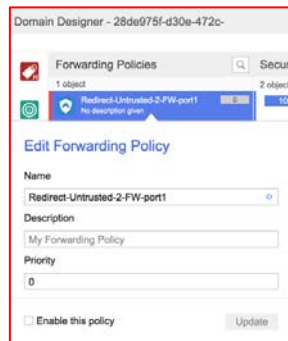
5.5.2 Target use case(s)

All traffic between the «untrusted networks» and the «internal network» will be blocked except for tcp port 22, 80 and 443. All these ports will get redirect to the Palo Alto Networks NGFW by the Nuage Networks VSP SDN. All traffic will be allowed to flow between VMs within the same network. This will illustrate three common scenarios, the first is the scenario where the SDN solution will completely block traffic. The second is the scenario where it will allow direct connectivity between specific end-points. The third will show the service insertion of a Palo Alto Networks virtual firewall into the data path for further inspection. The following logical diagrams illustrate the above:

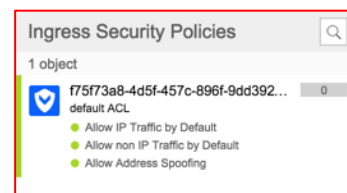
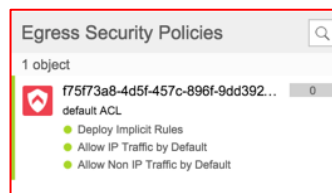
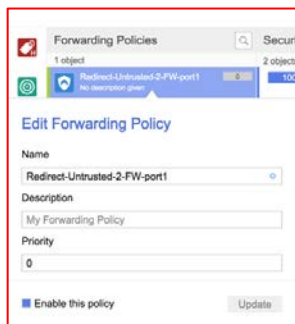


5.5.3 Test cases

- Deploy 3 VMs, two in the «internal» network and one in the «untrusted» network «CLIENT 1 – untrusted» and «CLIENT 1 – internal»
- Configure an HTTP server on a VM in the «internal» network «SERVER X»
- Test Case 1 - Set Nuage Networks policy to allow traffic to flow between segments (disabled Forwarding rule that redirects traffic to the vFW, and enabled flows)



- Run the following tests from a client in the «internal» network to the server in the «internal» network
 - ping
 - ssh
 - HTTP get
 - Run the following tests from a client in the «untrusted» network to the server in the «internal» network
 - ping
 - ssh
 - HTTP get
- Test Case 2 - Set Nuage Networks policy to redirect traffic (Enable on VSD) to the PAN vFW, and set the security policy on PAN vFW to deny the following applications (FTP, PING and SSH) and allow the following applications (WEB-BROWSING):



Name	Tags	Type	Source	Destination	Action	Profile	Options
1 Allow-untrusted-2-internal	none	universal	any DAG-untrusted Test-untrusted	DAG-internal Test-internal	Deny	none	
2 Allow-untrusted-2-internal-1	none	universal	any DAG-untrusted Test-untrusted	DAG-internal Test-internal	Allow	none	
3 Allow-internal-2-untrusted	none	universal	any DAG-internal Test-internal	DAG-untrusted Test-untrusted	Deny	none	
4 Allow-internal-2-untrusted-1	none	universal	any DAG-internal Test-internal	DAG-untrusted Test-untrusted	Allow	none	
5 intrazone-default	none	intrazone	any any	any any	Allow	none	
6 interzone-default	none	interzone	any any	any any	Allow	none	

- Run the following tests from a client in the «internal» network to the server in the «internal» network
 - ping
 - ssh
 - HTTP get
- Run the following tests from a client in the «untrusted» network to the server in the «internal» network
 - ping
 - ssh
 - HTTP get

5.5.4 Test Results

Test Case 1:

The Nuage Networks VSD has been set to «allow» traffic to flow between the two micro-segments «untrusted» and «internal» without redirection to the firewall. In this case, the expectation is that ALL tests will pass between segments and within a segment (ping, ssh and HTTP) as shown in the two screen shots that follow. The first is from the the «client-internal» and the second is from the «client-untrusted» VM.

```
$ ssh root@192.168.201.10
root@192.168.201.10's password:
root@192.168.201.10's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Jan 30 00:34:23 2016 from 192.168.199.6
root@server-int:~# exit
logout
$ ping 192.168.201.10
PING 192.168.201.10 (192.168.201.10): 56 data bytes
64 bytes from 192.168.201.10: seq=0 ttl=64 time=1.127 ms
64 bytes from 192.168.201.10: seq=1 ttl=64 time=0.283 ms

--- 192.168.201.10 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.283/0.705/1.127 ms
$ wget http://192.168.201.10
Connecting to 192.168.201.10 (192.168.201.10:80)
index.html 100% |*****| 11104 0:00:00 ETA
$
```

<client-internal console output>

```
root@client-EXT:~# ssh root@192.168.201.10
root@192.168.201.10's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Jan 30 00:12:54 2016 from 192.168.201.8
root@server-int:~# exit
logout
Connection to 192.168.201.10 closed.
root@client-EXT:~# wget http://192.168.201.10
--2016-01-30 00:34:34-- http://192.168.201.10/
Connecting to 192.168.201.10:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 11104 (11K) [text/html]
Saving to: 'index.html.9'

index.html.9      100%[=====] 10.84K  --.-KB/s  in 0s
2016-01-30 00:34:34 (25.5 MB/s) - 'index.html.9' saved [11104/11104]

root@client-EXT:~# _

root@client-EXT:~# ping 192.168.201.10
PING 192.168.201.10 (192.168.201.10) 56(84) bytes of data.
64 bytes from 192.168.201.10: icmp_seq=1 ttl=63 time=1.24 ms
64 bytes from 192.168.201.10: icmp_seq=2 ttl=63 time=0.332 ms
^C
--- 192.168.201.10 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
```

<client-untrusted console output>

Test Case 2 Results:

In this test, the Nuage Networks VSD has been set to «Redirect» traffic between the two micro-segments «untrusted» and «internal» to the PAN vFW. On the PAN vFW, policies have been put in place to «allow» certain applications and «deny» others. The policies used the «segments» as the source and destination and these were automatically pulled from the Nuage Networks VSD. «intra-segment» traffic has not changed from the previous test, because it was not redirected to the PAN vFW. However, the redirected traffic will be inspected by the PAN vFW and has to comply with the security policy that was set on the PAN vFW. In the test, HTTP traffic was the only traffic allowed to flow (based on the application setting) and PING and SSH traffic will get blocked as shown in the «client-untrusted» VM console as well as the logs (monitor output) on the PAN vFW.


```
$ ping 192.168.201.10
PING 192.168.201.10 (192.168.201.10): 56 data bytes
64 bytes from 192.168.201.10: seq=0 ttl=64 time=1.368 ms
64 bytes from 192.168.201.10: seq=1 ttl=64 time=0.259 ms
64 bytes from 192.168.201.10: seq=2 ttl=64 time=0.482 ms

--- 192.168.201.10 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.259/0.703/1.368 ms
$ ssh root@192.168.201.10
root@192.168.201.10's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Jan 30 00:12:18 2016 from 192.168.201.8
root@server-int:~# exit
logout
$ _
```

<client-internal console output>

```
root@client-EXT:~# ping 192.168.201.10
PING 192.168.201.10 (192.168.201.10) 56(84) bytes of data.

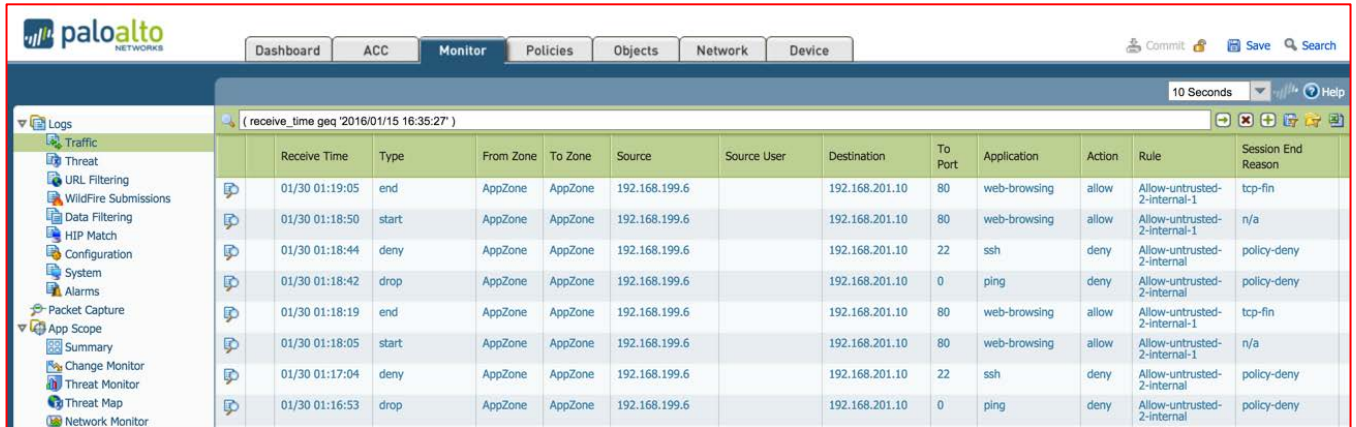
^C
--- 192.168.201.10 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2016ms

root@client-EXT:~# ssh root@192.168.201.10
^C
root@client-EXT:~# wget http://192.168.201.10
--2016-01-30 01:18:51-- http://192.168.201.10/
Connecting to 192.168.201.10:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 11104 (11K) [text/html]
Saving to: 'index.html.11'

index.html.11      100%[=====] 10.84K  --.-KB/s  in 0s
2016-01-30 01:18:51 (108 MB/s) - 'index.html.11' saved [11104/11104]

root@client-EXT:~# _
```

<client-untrusted console output>



The screenshot shows the Palo Alto Networks Firewall Monitor Logs interface. The top navigation bar includes Dashboard, ACC, Monitor (selected), Policies, Objects, Network, and Device. The left sidebar lists various log categories like Traffic, Threat, URL Filtering, etc. The main area displays a table of logs with columns: Receive Time, Type, From Zone, To Zone, Source, Source User, Destination, To Port, Application, Action, Rule, and Session End Reason. A search filter '(receive_time geq '2016/01/15 16:35:27')' is applied. The table shows several log entries for web-browsing and ssh applications.

Receive Time	Type	From Zone	To Zone	Source	Source User	Destination	To Port	Application	Action	Rule	Session End Reason
01/30 01:19:05	end	AppZone	AppZone	192.168.199.6		192.168.201.10	80	web-browsing	allow	Allow-untrusted-2-internal-1	tcp-fin
01/30 01:18:50	start	AppZone	AppZone	192.168.199.6		192.168.201.10	80	web-browsing	allow	Allow-untrusted-2-internal-1	n/a
01/30 01:18:44	deny	AppZone	AppZone	192.168.199.6		192.168.201.10	22	ssh	deny	Allow-untrusted-2-internal-1	policy-deny
01/30 01:18:42	drop	AppZone	AppZone	192.168.199.6		192.168.201.10	0	ping	deny	Allow-untrusted-2-internal-1	policy-deny
01/30 01:18:19	end	AppZone	AppZone	192.168.199.6		192.168.201.10	80	web-browsing	allow	Allow-untrusted-2-internal-1	tcp-fin
01/30 01:18:05	start	AppZone	AppZone	192.168.199.6		192.168.201.10	80	web-browsing	allow	Allow-untrusted-2-internal-1	n/a
01/30 01:17:04	deny	AppZone	AppZone	192.168.199.6		192.168.201.10	22	ssh	deny	Allow-untrusted-2-internal-1	policy-deny
01/30 01:16:53	drop	AppZone	AppZone	192.168.199.6		192.168.201.10	0	ping	deny	Allow-untrusted-2-internal-1	policy-deny



The screenshot shows the Detailed Log View for a specific log entry. It is divided into several sections: General, Source, Destination, Details, and Flags. The General section shows Session ID 1202, Action allow, Application web-browsing, and Rule Allow-untrusted-2-internal-1. The Source section shows User, Address 192.168.199.6, Country 192.168.0.0-192.168.255.255, Port 56365, Zone AppZone, and Interface ethernet1/2. The Destination section shows User, Address 192.168.201.10, Country 192.168.0.0-192.168.255.255, Port 80, Zone AppZone, and Interface ethernet1/1. The Details section shows Bytes 13127, Bytes Received 12149, Bytes Sent 978, and Repeat Count 1. The Flags section shows Captive Portal, Proxy Transaction, and Decrypted checkboxes, all of which are unchecked. Below these sections is a table with columns: PCAP, Receive Time, Type, Application, Action, Rule, Bytes, Severity, Category, and URL/FileNa... The table shows two entries for the same log entry.

PCAP	Receive Time	Type	Application	Action	Rule	Bytes	Severity	Category	URL/FileNa...
	2016/01/30 01:19:05	end	web-browsing	allow	Allow-untrusted-2-internal-1	13127		any	
	2016/01/30 01:18:50	start	web-browsing	allow	Allow-untrusted-2-internal-1	392		any	

Detailed Log View

General

Session ID

1201

Action

deny

Action Source

from-policy

Application

ssh

Rule

Allow-untrusted-2-internal

Session End Reason

policy-deny

Category

any

Virtual System

Device SN

IP Protocol

tcp

Log Action

Generated Time

Source

User

Address

192.168.199.6

Country

192.168.0.0-192.168.255.255

Port

58891

Zone

AppZone

Interface

ethernet1/2

Details

Bytes

312

Bytes Received

74

Bytes Sent

238

Repeat Count

1

Destination

User

Address

192.168.201.10

Country

192.168.0.0-192.168.255.255

Port

22

Zone

AppZone

Interface

ethernet1/1

Flags

Captive Portal

☐

Proxy Transaction

☐

Decrypted

☐

PCAP	Receive Time ▲	Type	Application	Action	Rule	Bytes	Severity	Category	URL/FileNa...
	2016/01/30 01:18:44	deny	ssh	deny	Allow-untrusted-2-internal	312		any	

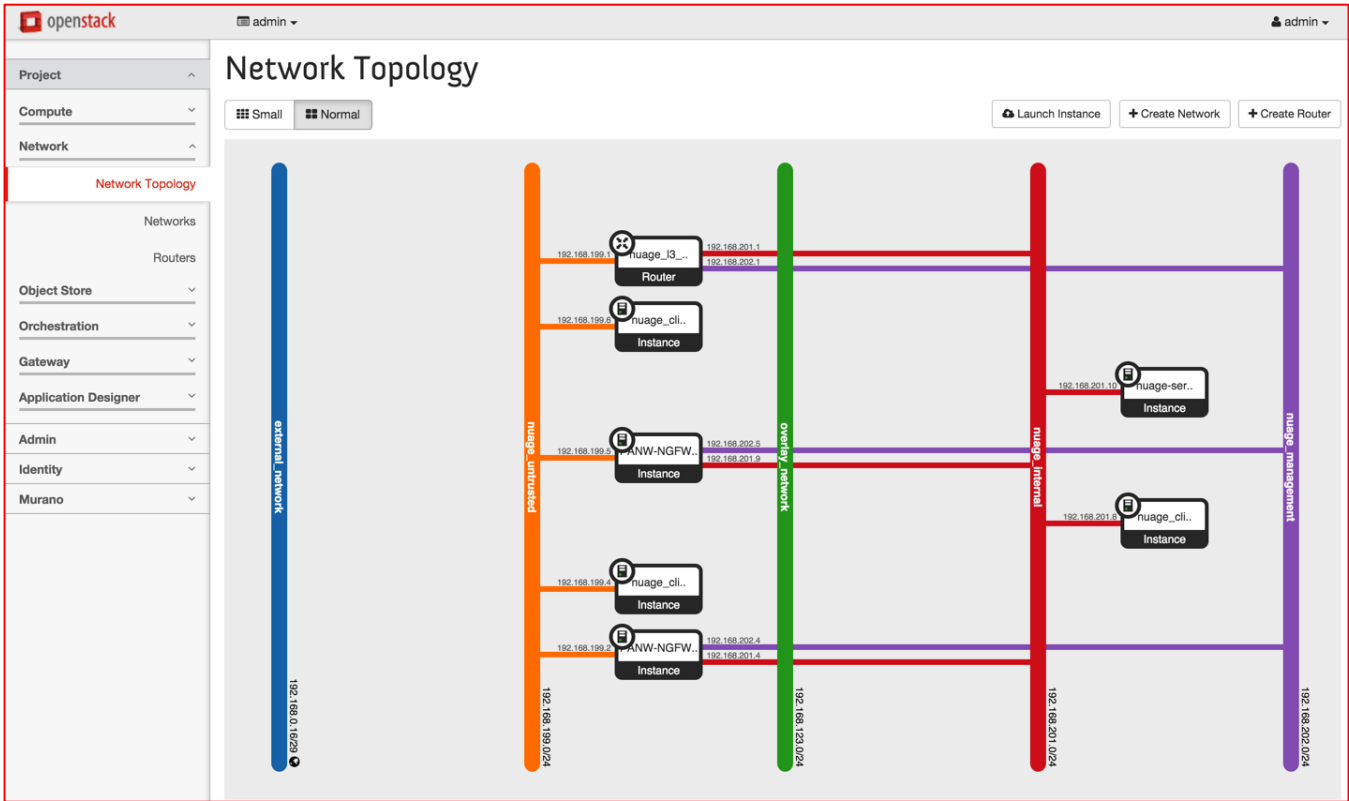
Close

v2.1

67

Appendix A: OpenStack Network Information (Horizon)

The following screenshots include information from the OpenStack setup used for the test in this runbook.



The screenshot shows the OpenStack Horizon interface for the 'Routers' page. The left sidebar has a navigation menu with 'Project', 'Compute', and 'Network' expanded. The main content area displays the 'Routers' page. At the top, there is a 'Filter' input field, a '+ Create Router' button, and a 'Delete Routers' button. Below this is a table with the following columns: Name, Status, External Network, Admin State, and Actions. The table contains one row for the router 'nuage_i3_router' with status 'Active' and admin state 'UP'. The 'Actions' column for this router has a 'Set Gateway' button. At the bottom of the table, it says 'Displaying 1 item'.

<input type="checkbox"/>	Name	Status	External Network	Admin State	Actions
<input type="checkbox"/>	nuage_i3_router	Active	-	UP	Set Gateway

openstack

admin

admin

Project

Compute

Network

Network Topology

Networks

Routers

Object Store

Orchestration

Gateway

Application Designer

Admin

Identity

Murano

Networks

Filter

<input type="checkbox"/>	Name	Subnets Associated	Shared	Status	Admin State	Actions
<input type="checkbox"/>	external_network	external_network 192.168.0.16/29	No	Active	UP	<input type="button" value="Edit Network"/>
<input type="checkbox"/>	nuage_untrusted	nuage_untrusted 192.168.199.0/24	No	Active	UP	<input type="button" value="Edit Network"/>
<input type="checkbox"/>	overlay_network	overlay_subnet 192.168.123.0/24	No	Active	UP	<input type="button" value="Edit Network"/>
<input type="checkbox"/>	nuage_internal	nuage_internal 192.168.201.0/24	No	Active	UP	<input type="button" value="Edit Network"/>
<input type="checkbox"/>	nuage_management	nuage_external 192.168.202.0/24	No	Active	UP	<input type="button" value="Edit Network"/>

Displaying 5 items

openstack

admin

admin

Project

Compute

Overview

Instances

Volumes

Images

Access & Security

Network

Object Store

Orchestration

Gateway

Application Designer

Admin

Identity

Murano

Instances

Instance Name Filter

<input type="checkbox"/>	Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
<input type="checkbox"/>	nuage_client2_untrusted	nuage-test-vm	192.168.199.6	m1.tiny	-	Active	nova	None	Running	2 days, 15 hours	<input type="button" value="Create Snapshot"/>
<input type="checkbox"/>	nuage-serverx-internal	nuage-test-vm	192.168.201.10	m1.tiny	-	Active	nova	None	Running	2 days, 15 hours	<input type="button" value="Create Snapshot"/>
<input type="checkbox"/>	PANW-NGFW-02	PA-VM-KVM-7.0.1	nuage_untrusted 192.168.199.5 nuage_internal 192.168.201.9 nuage_management 192.168.202.5	m1.medium	-	Active	nova	None	Running	4 days, 23 hours	<input type="button" value="Create Snapshot"/>
<input type="checkbox"/>	nuage_client1_internal	TestVM	192.168.201.8	m1.tiny	-	Active	nova	None	Running	2 weeks	<input type="button" value="Create Snapshot"/>
<input type="checkbox"/>	nuage_client1_untrusted	TestVM	192.168.199.4	m1.tiny	-	Active	nova	None	Running	2 weeks	<input type="button" value="Create Snapshot"/>
<input type="checkbox"/>	PANW-NGFW-01	PA-VM-KVM-7.0.1	nuage_untrusted 192.168.199.2 nuage_internal 192.168.201.4 nuage_management 192.168.202.4	m1.medium	-	Active	nova	None	Running	1 month, 1 week	<input type="button" value="Create Snapshot"/>

Displaying 6 items

Appendix B: Nuage Networks VSD VM xml

Deploying the VSD VM prior to the installation utilized the following XML file.

```
<domain type='kvm' id='3171'>
  <name>vsd_3.2R4</name>
  <uuid>170a8b32-2e61-3731-180d-227b9a0fec74</uuid>
  <memory unit='KiB'>8388608</memory>
  <currentMemory unit='KiB'>8388608</currentMemory>
  <vcpu placement='static'>6</vcpu>
  <resource>
    <partition>/machine</partition>
  </resource>
  <os>
    <type arch='x86_64' machine='pc-i440fx-trusty'>hvm</type>
    <boot dev='hd'>/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <pae/>
  </features>
  <clock offset='utc'>/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
  <devices>
    <emulator>/usr/bin/kvm-spice</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='qcow2'>/>
      <source file='/var/lib/libvirt/images/VSD-3.2.4_134.qcow2'>/>
      <target dev='vda' bus='virtio'>/>
      <alias name='virtio-disk0'>/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'>/>
    </disk>
    <controller type='usb' index='0'>
      <alias name='usb0'>/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x2'>/>
    </controller>
    <controller type='pci' index='0' model='pci-root'>
      <alias name='pci.0'>/>
    </controller>
    <interface type='network'>
      <mac address='52:54:00:3e:55:56'>/>
      <source network='fuel-pxe'>/>
      <target dev='vnet12'>/>
      <model type='virtio'>/>
      <alias name='net0'>/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'>/>
    </interface>
    <serial type='pty'>
      <source path='/dev/pts/16'>/>
      <target port='0'>/>
      <alias name='serial0'>/>
    </serial>
    <console type='pty' tty='/dev/pts/16'>
      <source path='/dev/pts/16'>/>
      <target type='serial' port='0'>/>
      <alias name='serial0'>/>
    </console>
    <input type='tablet' bus='usb'>
      <alias name='input0'>/>
    </input>
    <input type='mouse' bus='ps2'>/>
    <input type='keyboard' bus='ps2'>/>
  </devices>
</domain>
```

```
<graphics type='vnc' port='5906' autoport='yes' listen='0.0.0.0'>
  <listen type='address' address='0.0.0.0' />
</graphics>
<video>
  <model type='cirrus' vram='9216' heads='1' />
  <alias name='video0' />
  <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0' />
</video>
<memballoon model='virtio'>
  <alias name='balloon0' />
  <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0' />
</memballoon>
</devices>
<seclabel type='dynamic' model='apparmor' relabel='yes'>
  <label>libvirt-170a8b32-2e61-3731-180d-227b9a0fec74</label>
  <imagelabel>libvirt-170a8b32-2e61-3731-180d-227b9a0fec74</imagelabel>
</seclabel>
</domain>
```

[Optional]

Additional settings on the VSD specific for the test setup

```
iptables -F
iptables -L
sudo iptables -t nat -A PREROUTING -p tcp --destination-port 443 -j DNAT --to-destination 192.168.202.4:443
sudo iptables -t nat -A PREROUTING -p tcp --destination-port 443 -j DNAT --to-destination 192.168.202.5:443
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Appendix C: Nuage Networks VSC VM xml

Deploying the VSC VM prior to the installation utilized the following XML file.

```
<domain type='kvm' id='3176'>
  <name>vsc_3.2R4</name>
  <uuid>08b8516e-3726-487c-899f-e2f60b3482f0</uuid>
  <description>Timos VM</description>
  <memory unit='KiB'>4148224</memory>
  <currentMemory unit='KiB'>4147483</currentMemory>
  <vcpu placement='static'>4</vcpu>
  <cputune>
    <vcupin vcpu='0' cpuset='0' />
    <vcupin vcpu='1' cpuset='1' />
    <vcupin vcpu='2' cpuset='2' />
    <vcupin vcpu='3' cpuset='3' />
  </cputune>
  <resource>
    <partition>/machine</partition>
  </resource>
  <sysinfo type='smbios'>
    <system>
      <entry name='product'>Nuage Networks Virtual Services Controller</entry>
    </system>
  </sysinfo>
  <os>
    <type arch='x86_64' machine='pc-i440fx-trusty'>hvm</type>
    <smbios mode='sysinfo' />
  </os>
  <features>
    <apic />
  </features>
  <cpu>
    <topology sockets='4' cores='1' threads='1' />
  </cpu>
  <clock offset='utc'>
    <timer name='pit' tickpolicy='catchup' />
    <timer name='rtc' tickpolicy='catchup' />
  </clock>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>coredump-destroy</on_crash>
  <devices>
    <emulator>/usr/bin/kvm-spice</emulator>
    <disk type='file' device='disk' snapshot='no'>
      <driver name='qemu' type='qcow2' cache='writethrough' />
      <source file='/var/lib/libvirt/images/vsc_singledisk.qcow2' />
      <target dev='hda' bus='ide' />
      <boot order='1' />
      <alias name='ide0-0-0' />
      <address type='drive' controller='0' bus='0' target='0' unit='0' />
    </disk>
    <controller type='ide' index='0'>
      <alias name='ide0' />
      <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x1' />
    </controller>
    <controller type='usb' index='0'>
      <alias name='usb0' />
      <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x2' />
    </controller>
    <controller type='pci' index='0' model='pci-root'>
      <alias name='pci.0' />
    </controller>
    <interface type='network'>
      <mac address='52:54:00:8f:e6:85' />
    </interface>
  </devices>
</domain>
```



```
<source network='fuel-pxe' />
<target dev='vnet13' />
<model type='virtio' />
<alias name='net0' />
<address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0' />
</interface>
<serial type='pty'>
  <source path='/dev/pts/17' />
  <target port='0' />
  <alias name='serial0' />
</serial>
<console type='pty' tty='/dev/pts/17'>
  <source path='/dev/pts/17' />
  <target type='serial' port='0' />
  <alias name='serial0' />
</console>
<memballoon model='virtio'>
  <alias name='balloon0' />
  <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0' />
</memballoon>
</devices>
<seclabel type='none' model='apparmor' />
</domain>
```

Appendix D: Nuage Networks VRS-G VM xml

Deploying the VRS-G VM prior to the installation utilized the following XML file.

```
<domain type='kvm' id='3185'>
  <name>vrs_g_3.2R4</name>
  <uuid>b7467268-ad3e-b800-5617-c0844ca292a9</uuid>
  <memory unit='KiB'>2097152</memory>
  <currentMemory unit='KiB'>2097152</currentMemory>
  <vcpu placement='static'>2</vcpu>
  <resource>
    <partition>/machine</partition>
  </resource>
  <os>
    <type arch='x86_64' machine='pc-i440fx-trusty'>hvm</type>
    <boot dev='hd'>/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <pae/>
  </features>
  <clock offset='utc'>/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
  <devices>
    <emulator>/usr/bin/kvm-spice</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='qcow2'>/>
      <source file='/var/lib/libvirt/images/vrs_g.qcow2'>/>
      <target dev='vda' bus='virtio'>/>
      <alias name='virtio-disk0'>/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'>/>
    </disk>
    <controller type='usb' index='0'>
      <alias name='usb0'>/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x2'>/>
    </controller>
    <controller type='pci' index='0' model='pci-root'>
      <alias name='pci.0'>/>
    </controller>
    <interface type='network'>
      <mac address='52:54:00:eb:61:7d'>/>
      <source network='fuel-pxe'>/>
      <target dev='vnet4'>/>
      <model type='virtio'>/>
      <alias name='net0'>/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'>/>
    </interface>
    <interface type='network'>
      <mac address='52:54:00:15:b7:4e'>/>
      <source network='fuel-pxe'>/>
      <target dev='vnet5'>/>
      <model type='virtio'>/>
      <alias name='net1'>/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x0a' function='0x0'>/>
    </interface>
    <interface type='network'>
      <mac address='52:54:00:f6:06:31'>/>
      <source network='fuel-pxe'>/>
      <target dev='vnet10'>/>
      <model type='virtio'>/>
      <alias name='net2'>/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x0b' function='0x0'>/>
    </interface>
  </devices>
</domain>
```

```
</interface>
<serial type='pty'>
  <source path='/dev/pts/14'/>
  <target port='0'/>
  <alias name='serial0'/>
</serial>
<console type='pty' tty='/dev/pts/14'>
  <source path='/dev/pts/14'/>
  <target type='serial' port='0'/>
  <alias name='serial0'/>
</console>
<input type='tablet' bus='usb'>
  <alias name='input0'/>
</input>
<input type='mouse' bus='ps2'/>
<input type='keyboard' bus='ps2'/>
<graphics type='vnc' port='5902' autoport='yes' listen='0.0.0.0'>
  <listen type='address' address='0.0.0.0'/>
</graphics>
<video>
  <model type='cirrus' vram='9216' heads='1'/>
  <alias name='video0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'/>
</video>
<memballoon model='virtio'>
  <alias name='balloon0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'/>
</memballoon>
</devices>
<seclabel type='dynamic' model='apparmor' relabel='yes'>
  <label>libvirt-b7467268-ad3e-b800-5617-c0844ca292a9</label>
  <imagelabel>libvirt-b7467268-ad3e-b800-5617-c0844ca292a9</imagelabel>
</seclabel>
</domain>
```

Appendix E: Additional PAN vFW Initial Setup Commands

Initial setup session of a PAN vFW

```
Username & password: admin/admin
```

```
#configure
```

```
#set deviceconfig system ip-address 192.168.202.5 netmask 255.255.255.0 default-gateway  
192.168.202.1
```

```
#commit
```

```
#debug show vm-series interfaces all
```

```
Phoenix_interface Base-OS_port Base-OS_MAC PCI-IDmgt eth0 52:54:00:d7:91:52
```

```
0000:00:03.0Ethernet1/1 eth1 52:54:00:fe:8c:80 0000:00:06.0Ethernet1/2 eth2
```

```
0e:c6:6b:b4:72:06 0000:00:07.0Ethernet1/3 eth3 06:1b:a5:7e:a5:78 0000:00:08.0Ethernet1/4
```

```
eth4 26:a9:26:54:27:a1 0000:00:09.0Ethernet1/5 eth5 52:54:00:f4:62:13 0000:00:10.0
```

References:

- Palo Alto Administration Guide ([link](#))
- VMSeries Virtualization Guide ([link](#))
- Nuage Networks VSP 3.0.R5 Installation Runbook on MOS 6 ([link](#))