VIO Design Guide

Authors: C. Oliveira, M. Hernandez, X. Gao
Contributions: A. Pierce, R. Boswell

Reviewers: A. Gupta, D. Desmidt, M. Voelker, N. Jahagirdar
R. Gerganov

VIO Design Guide Overview

About this document

This document provides guidance in architecting and deploying an enterprise-grade OpenStack cloud based on VMware® best practices and real-world scenarios. VMware Integrated OpenStack (VIO) is an OpenStack distribution from VMware that makes it easy for IT organization to run an enterprise-grade OpenStack on top of vSphere infrastructure. VIO is an "integrated product" approach to OpenStack, which means that the components must be integrated to work well together. The components range from compute, storage and network products to management and monitoring services. VIO dramatically simplifies OpenStack deployment and operations, and delivers enhanced agility, rapid innovation, better economics, and scale. Our integrated approach enables our customers to:

- Deploy Interop compliant (formerly DefCore) OpenStack API
- Avoid private cloud snowflakes
- Leverage existing expertise and tools to operate the infrastructure
- Implement network virtualization
- Manage their cloud through a single pane of glass
- Call a single vendor for support in case needed

The following OpenStack core components are addressed in this document:

- Nova (Compute)
- Neutron (Network)
- Glance (Image Storage)
- Cinder (Block Storage)

In addition, we will address operational best practices and automation and orchestration options.

Objectives/outcomes:

At the end of this guide, readers will

- Understand VIO approach to OpenStack
- Design and architect a VIO deployment
- Operate and maintain a production VIO deployment
- Demonstrate VIO to a customer/prospective customer

Target Audience

This document is designed for Solution architects, Pre-Sales consultants, field consultants, advanced services specialists, and customers in building an OpenStack cloud on top of a battle tested VMware SDDC infrastructure. The ideal reader should have familiarity with:

- vSphere
- Basic NSX functionality
- Application deployment

- Core OpenStack components

Revision History

Version	Update	Comments
1.6.0	None	First Release
1.6.2	Edit into Word Document	Format changes
1.6.3	Merged comments from Reviewers	LBaaS and Conclusion are new contents

Design guide revision history

Contents

Table of Contents: Images, Tables and Screen Captures

Chapter 1: VMware Integrated OpenStack Overview

OpenStack is an open source framework that emerged as a tool for IT to provide application development teams with programmatic access to the infrastructure. Developers of next generation, cloud native applications have embraced API-based, programmatic access to public cloud infrastructure to develop mobile, social, big data and other applications. OpenStack delivers similar capabilities and user experience through industry standard APIs for on-premises, private clouds.

OpenStack itself does not provide the virtual technologies. It needs the underlying hypervisor, networking, and storage, which is provided by different vendors. VMware Integrated OpenStack (VIO) rapidly deploys production-grade OpenStack on top of industry-leading VMware technology. VIO leverages your existing VMware investment to simplify installation, upgrade, operations, monitoring, and more.

VIO is compliant with the OpenStack Foundation guidelines for a true OpenStack distribution, and is API compatible for all OpenStack services. Learn more about those guidelines here.

OpenStack is as good as the technological foundation it runs on. With VIO, an OpenStack cloud operates on top of VMware vSphere and VMware NSX, leveraging capabilities for security, stability, performance and reliability. Tools used to manage, monitor and troubleshoot existing private cloud can be used with VIO, allowing an organization to capitalize on existing expertise to build, run and troubleshoot an OpenStack environment. VMware is committed to integrating its industry-leading technologies with OpenStack to facilitate customer choice and promote open APIs. As a result, VMware is a major OpenStack code contributor and a gold member in the OpenStack Foundation.

1.1 Introduction to OpenStack

OpenStack is an open-source framework for building an Infrastructure-as-a-Service (IaaS), private and public clouds. OpenStack started in 2010 as a joint project between Rackspace and NASA. More than 500 companies actively participating in the OpenStack project. The OpenStack community overall has more than 60,000 registered members from 185 countries, and more than 1925 code contributors. OpenStack adoption is greatest in North America, followed by Asia and EMEA (See Figure 1.1 and Table 1.1). OpenStack is used under the terms of the Apache License. Since 2016, OpenStack is managed by the OpenStack Foundation.

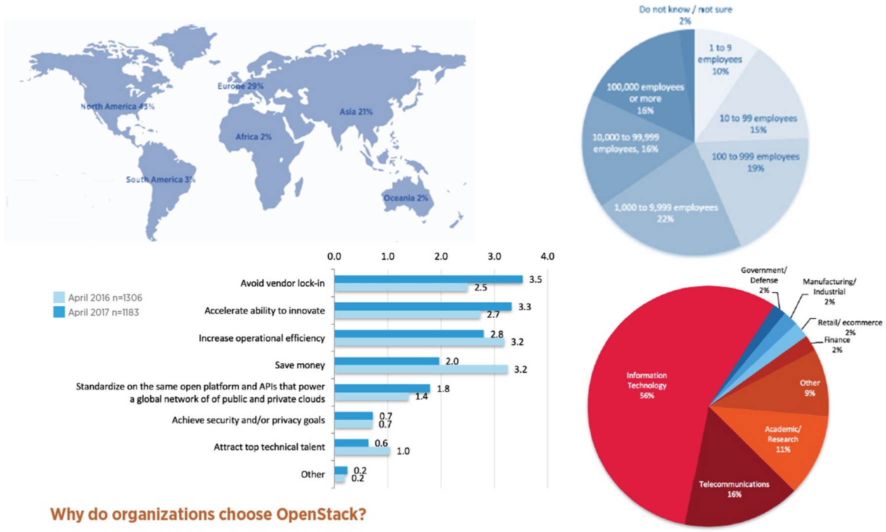


Figure 1.1: OpenStack Adoption by Regions, Company Size and Industries

Business Driver	Notes
Avoid Vendor Lock in	The ability to easily move from one vendor to another without making many changes to your applications. Organizations don't want to commit to using a specific vendor, and prefer to have the flexibility to move between vendors without high costs.
Ability to Innovate and Compete	DevOps empowerment. Provide developers with public cloud capabilities and use experience such as agility, flexibility, scale, etc.
Increase operational efficiency	Leverage automation and orchestration to automated workflows and reduce provisioning time.

Table 1.1: Top three drivers for OpenStack adoption based on the OpenStack User Survey

OpenStack emphasizes the importance of early and frequent release in creating a tight feedback loop between developers and users: OpenStack is developed and released in about 6-month cycles.

After the initial release, additional stable point releases are delivered in each release series (See Figure 1.2). A release reaches end of life approximately one year after its initial release date.

Companies of all sizes are adopting OpenStack. Leading reference customers who have deployed OpenStack in production include SAP, Bloomberg, Wells Fargo, Visa, Walmart, AT&T, eBay, and PayPal. OpenStack usage is under the terms of the Apache License.

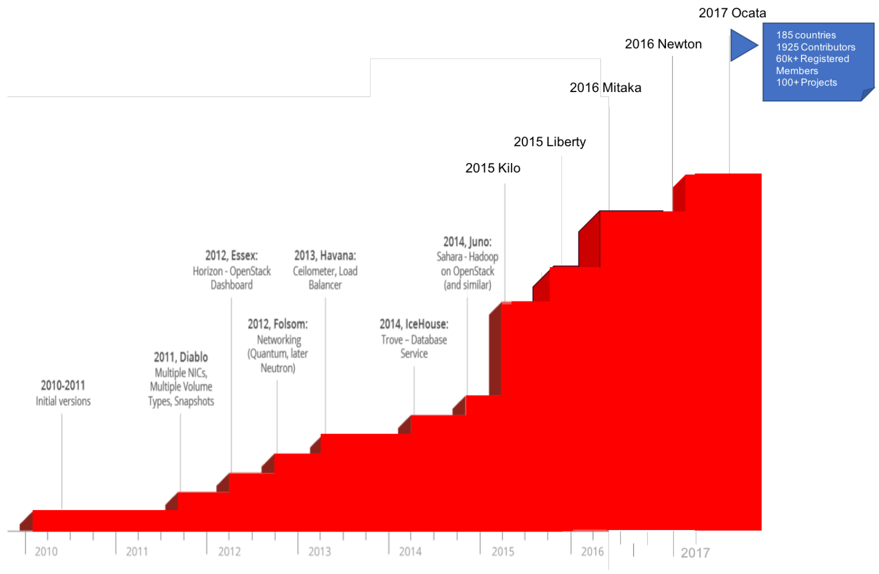


Figure 1.2: OpenStack Releases

Major OpenStack distribution vendors include:

- Red Hat
- SuSE – recently acquired HP’s OpenStack technologies.
- Mirantis - An early OpenStack vendor, launched its own distribution in 2013. It is popular in the telecom space.
- VMware – delivers an OpenStack distribution that runs on vSphere

1.2 VMware Commitment to OpenSource

VMware is a longtime top 10 contributor to OpenStack’s approved-release projects with several current and former core contributors and Project Team Leads (PTL). VMware is the founder of the Neutron Project in OpenStack (formerly known as Quantum). Beyond mainstream project contributions, VMware is contributing to OpenStack in other ways as well. For example:

- VMware co-chairs the Interop Working Group and its largest contributor. The Interop Working Group is an OpenStack Foundation Board of Directors working group whose primary task is to develop the interoperability guidelines that all products bearing the OpenStack trademark and logo must adhere to, thus guaranteeing a level of interoperability across the vendor ecosystem. Core developers contribute to the following OpenStack projects:
    - Bandit project - A security linter for Python that is now used in the CI gates for many OpenStack projects, and is used outside of OpenStack as well.
    - Senlin project - A cluster manager that is likely to take over auto-scaling functionality from Heat project and is attracting major interest in the telecommunications space.
    - Osprofiler - A profiler for OpenStack requests and services, which has helped track down countless bugs and performance bottlenecks across many projects and deployments.
    - Congress - A project for policy management, created by VMware.
- VMware is the cofounder and largest contributor to the Open vSwitch (OVS) community, and more recently, to the IOvisor community. These two virtual switching projects are both collaborative projects housed in the Linux

Foundation. OVS is the most popular deployment option for OpenStack networking (OVN), and has become a default choice for Linux-based virtual networking outside of OpenStack as well. Open Virtual Networking (OVN) provides a robust L3 networking option that is being co-developed and deployed in large deployments including IBM and Red Hat

- VMware is a gold member of the Linux Foundation. Our Chief Open Source Officer, Dirk Hohndel, sits on the Board of Directors at the Linux Foundation and has longstanding ties to the Linux community. VMware is also an inaugural member of the Cloud Native Computing Foundation and employs several Linux kernel subsystem contributors and maintainers. VMware is also a platinum member of ONAP (Open Network Automation Platform) and its predecessor Open-O at the Linux Foundation, a silver member of OPNFV, and a sponsor of the Core Infrastructure Initiative.
- Cloud Foundry, the industry-leading PaaS platform, was created at VMware, and Pivotal Labs was eventually spun out into its own company within the Dell Technologies family. Spring framework, one of the most popular OpenSource Java EE frameworks, was also part of the Pivotal labs spin out.

VMware has been very active in the container space as well. A few of our OpenSource projects here include: Harbor, a popular open source enterprise-ready container registries

- Admiral, a highly scalable and lightweight container management platform vSphere Integrated Containers Engine and Photon Controller
- Photon OS, a minimal Linux container host operating system
- Lightwave, which provides identity services for applications and containers Kubernetes and Docker contributions
- VMware has a long track record of open sourcing and co-developing toolkits for data center users. These range from tools such as Chaperone (an end-to-end installer for SDDC, OpenStack, and so on) to SDKs such as pyVmomi and goVmomi.

VMware has open sourced reusable building-block components for distributed systems projects like Xenon, a distributed control plane and microservices framework. VMware has also participated in other fundamental projects, such as Zookeeper and RabbitMQ.

VMware is a longtime contributor to open standards as well.

- VMware has co-authored the RFC for VXLAN and Geneve (the de facto and emerging standards for network virtualization).
- VMware has also helped guide the Open Virtualization Format (an ISO and ANSI standard that has very broad adoption across the industry).
- VMware open sourced the Clarity UX framework, a UI development kit built on Angular 2 components to craft exceptional user experiences.
- Outside of technical contribution, VMware also participates in, sponsors, and hosts OpenStack meetup groups. VMware often hosts the Bay Area meetup on its campus in Palo Alto. Triangle OpenStack Meetup in Research Triangle Park, NC was co-founded by VMware employees.
- VMware is also a longtime sponsor of the OpenStack Summit and other OpenStack regional events.

1.3 Introduction to VMware Integrated OpenStack

VMware Integrated OpenStack(VIO) is based on standard upstream OpenStack. Core projects that makes up VIO stack are:

- Neutron - Network Connectivity
- Cinder - Block Storage for Volumes
- Nova - Compute Services
- Glance - Image Repository
- Horizon - UI Portal
- Ceilometer - Telemetry
- Heat - Orchestration
- Keystone - Identity Management
- Swift - Object Store. Although Swift is a core project, VIO does not bundle Swift by default. Third-party integration is required, typically with SwiftStack (www.swiftstack.com).

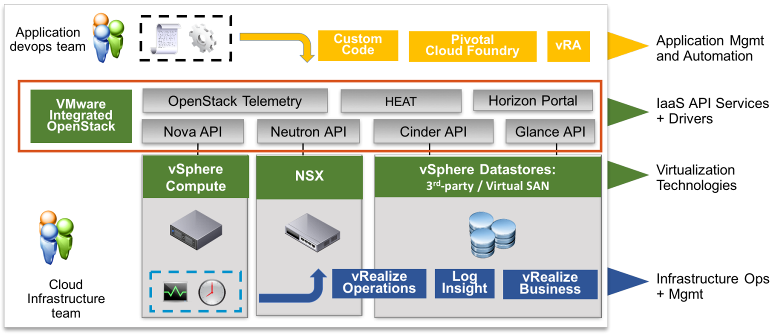


Figure 1.3: VIO Stack

VIO consists of the upstream OpenStack code preconfigured and optimized to use the VMware OpenStack drivers and tools required to install, upgrade, and operate an OpenStack cloud based on VMware technologies. There are four major pillars that make up the VIO infrastructure (Figure 1.3):

- vSphere Compute – Nove Compute API
- vSphere Networking with VDS, or NSX – Neutron Network API
- vSphere Storage – Cinder Storage or Glance Image API
- vReailze Monitoring – Day 2 monitoring and cloud governance using vRealize Automation, vRealize Operations (vROPS), vRealize Log Insight (vRLI), vRealize Network Insight (vRNI) an vRealize Cloud for Business (vRB).

Nova sees each VMware vSphere compute cluster as a single compute node (Figure 1.4) and selects the vSphere cluster to place the VM. Once a cluster is selected, vCenter uses DRS to optimally place the VM within Cluster.

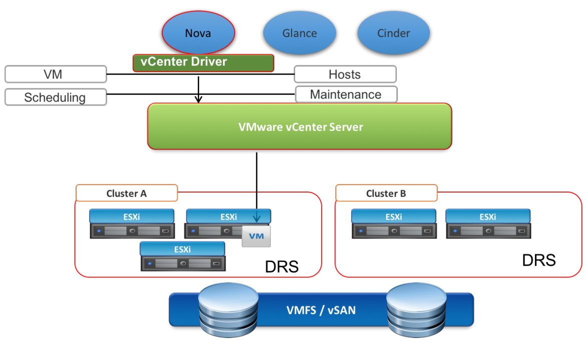


Figure 1.4: Nova and VMware vSphere compute cluster

Cinder executes block volume operations through VMDK driver (Figure 1.5). vCenter creates the volume and initially the volume belongs to a 'shadow VM'. When the volume is attached to a running VM, vCenter then changes the parent for the volume from the 'shadow VM' to the actual virtual machine.

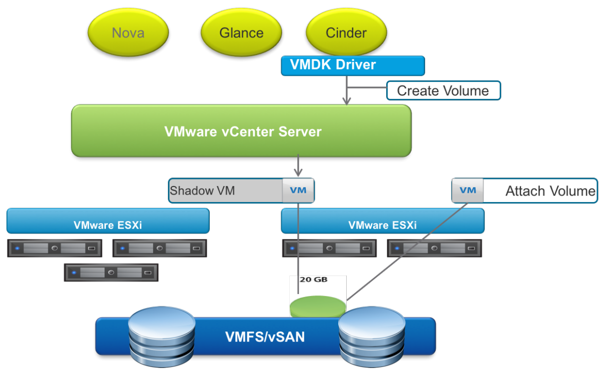


Figure 1.5: Cinder and VMware vCenter Server

Glance images are stored in a set of dedicated image service storage (Figure 1.6). VIO supports import of ISO, Raw, VDI, VHD, QCOW2, VMDK and OVA format images. Conversion into VMDK or OVA format happens automatically during the image import process. When a VM is booted, VMDK image is copied from the OpenStack Image Service to the vSphere datastore and gets cached in the datastore. Subsequent VM that boots from the same datastore will use the cached version.

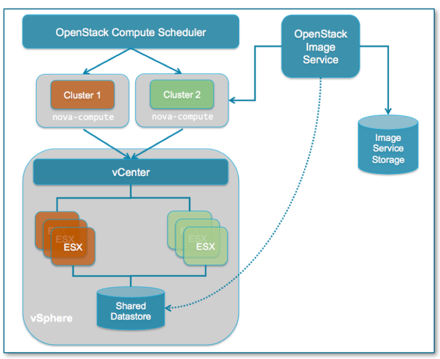


Figure 1.6: OpenStack Image Service

OpenStack Server uses NSX neutron plugin to communicate with the NSX manager or vCenter in case of VDS deployment (Figure 1.7). This Neutron plugin is open source and can be used with ANY OpenStack implementation. All OpenStack neutron operations maps directly to the NSX Manager. NSX edge devices functions as OpenStack L3 Agent, Metadata Server, DHCP L2 Agent, Tenant networking and Security Group policy enforcement. As a direct result of leveraging enterprise-grade virtualization with vSphere and enterprise grade networking with NSX, customers enjoy an enterprise-grade OpenStack layer, thus mitigating the risks and shortcomings of the reference implementation. When deploying VDS only model, neutron advanced features such as Security Group, L3 Agent, Tenant networking and etc are not supported.

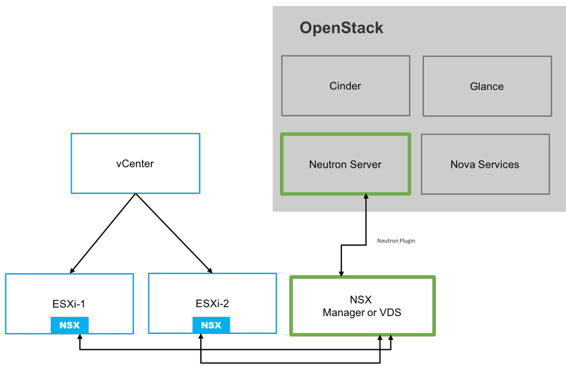


Figure 1.7: OpenStack and NSX

vRealize Log Insight (Figure 1.8) analyzes any structure and unstructured time-series data and configuration. It can digest any type of log data. The users just send their data to Log Insight, which automatically identifies structures in the data and creates a high-performance index for performing analytics. Unlike databases there is no need to engage database admins to Extract Transform and Load (ETL) the data. It can ingest TBs of data per node per day and has a configurable retention policy. Optionally log data can be written to an archive, Log Insight OpenStack Content pack ships with out of the box knowledge of OpenStack services. OpenStack Admins can leverage out of box OpenStack dashboards to filter and display time series events based on the OpenStack logs to simplify troubleshooting. Log Insight also includes Content Packs for most common storage, applications, compute and network devices.

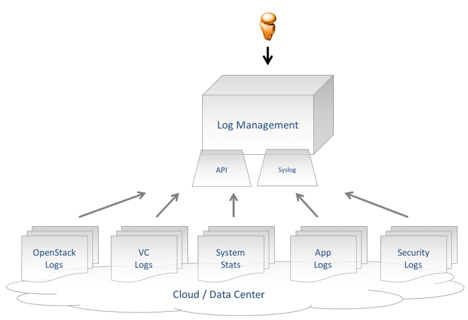


Figure 1.8: OpenStack and vRealize Log Insight

vRealize Operations Manager (Figure 1.9) provides proactive identification and remediation of emerging performance, capacity, and configuration issues. It provides a single glass pane visibility across applications and infrastructure. Out of box OpenStack Content Packs provide performance monitoring for Core OpenStack services, Databases, RabbitMQ and HAproxy, Cloud Admins can also use vROps to monitor OpenStack tenant consumption as well as automated OpenStack infrastructure capacity optimization and planning.

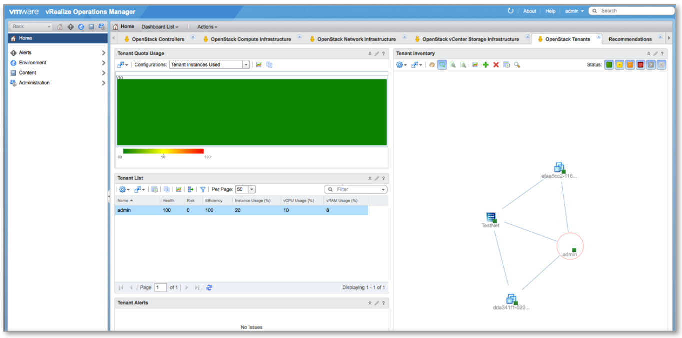


Figure 1.9: vRealize Operations Manager

vRealize Network Insight (Figure 1.10) provides Network Analytics by discover vCenter and NSX constructs. Based on workload characteristics, ports and common services, vRNi is able to generate automated security grouping to enable application level micro-segmentation. For OpenStack deployments, VMware recommends vRNI version 3.6 or later.

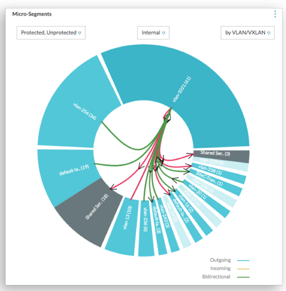


Figure 1.10: vRealize Network Insight

1.4 Benefits of Using OpenStack on VMware

VIO differentiates from its competition in by making install, upgrade and maintenance operations simple, and leveraging VMware enterprise grade infrastructure to provide the most stable release of OpenStack in the market. In addition to OpenStack distribution, VMware is also helping bridge gaps in traditional OpenStack management monitoring and logging by making VMware enterprise grade tools such as vRealize Operations Manager and Log Insight OpenStack aware with no customization.

- Standard DefCore Compliant OpenStack Distribution is delivered as an OVA. The implementation is fully supported by VMware

- The best foundational Infrastructure for IaaS is available with vSphere Compute (Nova), NSX Networking (Neutron), vSphere Storage (Cinder / Glance)
- OpenStack endpoint management and logging is simple and easy to perform with VMware vRealize Operations Manager for management, vRealize Log Insight for logging, and vRealize Business for chargeback analysis
- Best way to leverage existing VMware investment in People, Skills and Infrastructure

VIO is an integrated approach to deploy OpenStack. Integrated means that we are not here to do a custom fit. VIO is a bundled solution, customers need to consume the entire set to realize its true benefits. This implies that customer must leverage vSphere, NSX if they need security and micro segmentation, and vSphere Storage for Cinder and Glance. Our integrated approach offers stability and consistency across all deployments. The integrated approach helps our customers:

- Avoid private cloud snowflakes
- Leverage existing vSphere expertise & tools to operate the infrastructure
- Get a fully validated end to end solution. The software compatibility matrix can be used beyond the initial deployment.
- A phased approach to Network Virtualization
- Gain insights into the IaaS offerings that they're using, for both private and hybrid clouds
- Get support from a single vendor

Chapter 2: VIO Infrastructure Architecture and Requirements

2.1 vSphere Compute

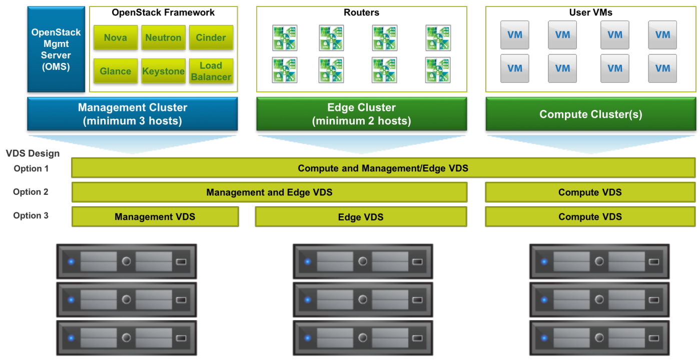


Figure 2.1: vSphere Compute Models

Based on SDDC best practices, VIO assumes following logical roles (Figure 2.1):

- Management
- Edge (not applicable in a vSphere Distributed Switch (VDS) only deployment)
- Compute

These logical roles are then mapped physically to the underlying vSphere infrastructure. Depending on types of deployment, the following mapping is possible:

- Integrated model – One vSphere cluster for consolidated Management, Edge and Compute.
- Consolidated model - One vSphere cluster for consolidated Management and Edge, and separate vSphere clusters for Compute.
- Dedicated model - Separate vSphere clusters for Management, Edge, and Compute.

A dedicated model is recommended, and provides the following design advantages:

- Simplicity in network and management capacity planning and scaling Isolation of organizational span of control
- Failure domain isolation
- Simplified lifecycle management of hardware resources
- Better management of hardware, such as CPU, memory, and NIC resources Simplified upgrade and migration process.

In small deployments where users want to limit the number of physical servers, Management and Edge clusters can be consolidated in a single vSphere cluster. The consolidated cluster for Management and Edge hosts both the control infrastructure for VIO and NSX, as well as NSX Edge devices. The Edge cluster is not applicable in a VDS only deployment.

At least three hosts are required in the management cluster in both the consolidated and dedicated model. This is so that management components can run on different hosts, and the failure of one host does not impact control plane availability.

In the dedicated model, a minimum of two hosts are required for the NSX Edge cluster.

There should be at least one workload/compute cluster. You can deploy additional compute clusters based on application, scale, and SLA requirements. A compute cluster can be as small as one node if you are only experimenting with VIO. While vCenter can support up to 64 hosts in a single cluster, this is not recommended for VIO, as large vSphere clusters add delay in VM scheduling and boot time. Typical production VIO cluster size is around 10 to 12 hosts. In a deployment with large number of concurrent operations, we recommend to create new clusters when you need additional compute capacity.

2.2 vSphere Networking

Foundations of VIO Networking starts with the VDS (VMware vSphere Distributed Switch) when using NSX-v. NSX-T does not have a VDS dependency. VDS provides a centralized interface from which you can configure, monitor and administer virtual machine access switching for the entire data center. NSX Networking can then be enabled on top of it. Specific to VDS deployment, the following networking models are possible:

- A single VDS that spans across all vSphere clusters (management, Edge, and workload/compute)
- A shared VDS between management and Edge, and a dedicated VDS for workload.
- A dedicated VDS for each cluster role.

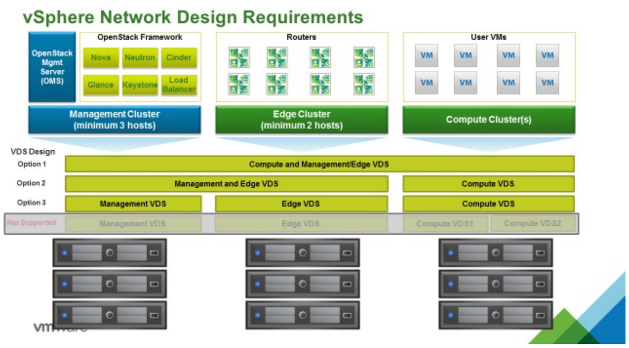


Figure 2.2: vSphere Networking Models

In version 3.x and earlier, all compute clusters must map to a single VDS, you cannot have a separate vDS for each compute cluster. As a VDS cannot span across multiple vSphere data centers, this also means that all VIO resources must be grouped under the same vSphere data center. In VIO 4.0 and later, we do not have this restriction.

The following VLANs are required for a standard VIO VXLAN based deployment (Figure 2.3):

- Management - Used for management access of VIO, ESXi hosts, NSX manager etc.
- API - Used by OpenStack users/tenants to access their OpenStack Project for provisioning and monitoring.
- External network - Routable IP address space within the customer organization. One or more addresses depending on customer needs.
- VXLAN transport - Used to enable tenant networking.
- vMotion - Deployed as part of vSphere Infrastructure, independent of VIO. Not shown in the diagram.
- Storage Access - Deployed as part of vSphere Infrastructure, independent of VIO. Not shown in the diagram.

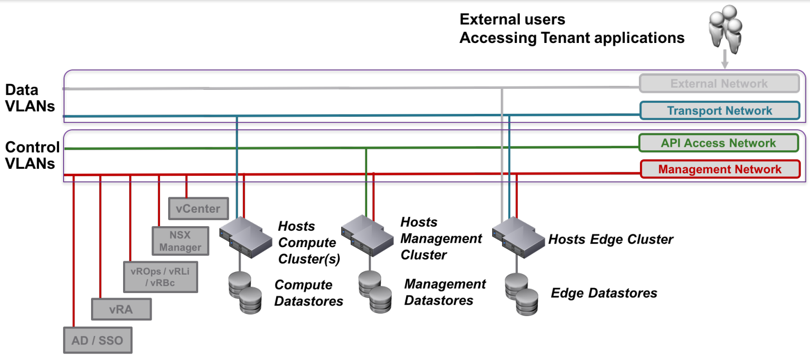


Figure 2.3: VLAN deployments

API and Management VLANs are intended to provide management and API access for cloud admins and OpenStack users. Both networks are control plane only. External and VXLAN transport VLANs are both data plane networks supporting communication to and from the VM. External networks are used to allow VMs access to the Internet or intranet, while VXLAN transport is used to enable tenant networking to support VM communication within the data center. The transport VLAN must be mapped to both the compute and edge cluster, while the external network needs to be mapped to Edge cluster only. With the exception of VXLAN transport network, all remaining OpenStack VLANs must be configured using routable IP address space within the customer organization.

An MTU size of 1600 is required for VXLAN end points. If using NSX-T and the Geneve encapsulation, an MTU of 1700 bytes is recommended to accommodate the overlay headers as well as the Distributed Network Encryption (DNE) headers, if used. VLAN requirements are the same for VDS based deployments, except for VXLAN transport VLAN. However, you might need to budget a much larger set of VLANs for OpenStack tenant worloads (Provider VLAN only). For more information, see the Neutron deep dive section..

To enable NSX networking on top of VDS, compute and Edge clusters must be prepared for NSX, and the transport zone must include these clusters. The management cluster does not have to be prepared for NSX or be part of a transport zone. A single NSX transport zone is supported with VIO. You must implement High Availability in all cases to provide redundancy to infrastructure components. For more information, see the Neutron Integration section.

2.3 vSphere Storage

Cinder makes use in VIO of a VMDK driver that leverages vSphere datastores to abstract the physical storage arrays presented to the hypervisors. As a result, VIO is directly compatible with any storage backend supported in vSphere.

The compute and Edge datastore is used by Cinder to provision persistent volumes for the Instances. Shared Datastore is a requirement for NSX edge and Workload/Compute clusters to take full advantage of the vSphere infrastructure DRS and HA capabilities. The management cluster where VIO infrastructure components reside requires three separate datastores for component redundancy so that components with HA are placed on separate datastore. This is to avoid single point of storage failure. We recommend management cluster datastore to be shared as well, accessible across all Management cluster compute nodes.

VIO Cinder is agnostic to the shared datastore backing, that is, you can use NFS, iSCSI, FC, FCoE, or VSAN. It's important to choose the correct storage based on your customer requirements, as the type of storage deployed greatly influences VM spin up and boot.

Cinder makes use in VIO of a VMDK driver that leverages vSphere datastores to abstract the physical storage arrays presented to the hypervisors. As a result, VIO is directly compatible with any storage backend supported in vSphere.

The VMware VMDK driver connects to vCenter, through which it can dynamically access all the data stores visible from the ESX hosts in the managed cluster.

When you create a volume, the VMDK driver creates a VMDK file on demand. The creation of the VMDK file is completed only when the volume is subsequently attached to an instance. The reason for this requirement is that data stores visible to the instance determine where to place the volume. Before the service creates the VMDK file, you must attach a volume to the target instance.

The running vSphere VM is automatically reconfigured to attach the VMDK file as an extra disk. After the disk is attached, you can log in to the running vSphere VM to rescan and discover the additional disk.

With the update to ESX version 6.0, the VMDK driver now supports NFS version 4.1.

2.3.1 VMDK Disk Type

The VMware VMDK drivers support the creation of VMDK disk file types thin, lazyZeroedThick (sometimes called thick or flat), or eagerZeroedThick.

- A thin virtual disk is allocated and zeroed on demand as the space is used. Unused space on a thin disk is available to other users.
- A lazy zeroed thick virtual disk has all space allocated at disk creation. This reserves the entire disk space, so it is not available to other users at any time.
- An eager zeroed thick virtual disk is similar to a lazy zeroed thick disk, in that the entire disk is allocated at creation. However, in this type, any previous data is wiped clean on the disk before the write. This causes the disk creation to take more time, but also prevents problems with stale data on physical media.

Use the vmware:vmdk_type extra spec key with the appropriate value to specify the VMDK disk file type. This table (Table 2.1) shows the mapping between the extra spec entry and the VMDK disk file type:

Disk File Type	Extra Spec Key	Extra Spec Value
thin	vmware:vmdk_type	thin
lazyZeroedThick	vmware:vmdk_type	thick
eagerZeroedThick	vmware:vmdk_type	eagerZeroedThick

Table 2.1: VMware:vmdk_type extra spec key with the appropriate value

If you do not specify a vmdk_type extra spec entry, the disk file type defaults to thin.

2.3.2 Clone Type

With the VMware VMDK drivers, you can create a volume from another source volume or a snapshot point. The VMware vCenter VMDK driver supports the full and linked/fast clone types. Use the vmware:clone_type extra spec key to specify the clone type. The following table (Table 2.2) captures the mapping for clone types:

Clone Type	Extra Spec Key	Extra Spec Value
full	vmware:clone_type	full
linked/fast	vmware:clone_type	linked

Table 2.2: Clone Type

If you do not specify the clone type, the default is full.

2.3.3 Storage Policy-Based Management SPDM

Using vCenter Storage Policies to Specify Back-End Datastores. In vCenter 5.5 and later, you can create one or more storage policies and expose them as a block storage volume-type to a VMDK volume. The storage policies are exposed to the VMDK driver through the extra spec property with the vmware:storage profile key. VIO fully supports Storage Policy-Based Management (SPBM). – Figure 2.4

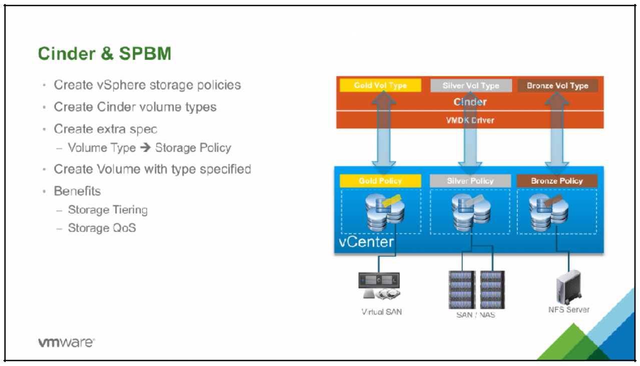


Figure 2.4: Storage Policy-Based Management (SPBM).

1. Administrators apply storage policies to VMware vSphere datastores.
2. Administrators create custom volume types in OpenStack that correspond to the vSphere storage policies.
3. When OpenStack users create their cinder volumes, they specify an extra specification with the required volume type VMware vSphere creates the volume on the appropriate datastore according to the policy.

2.4 VIO Control Plane

2.4.1 VIO Control Plane Overview

VIO control plane can be deployed in two ways:

1. Compact mode - 1 VM setup, (not including compute driver and OpenStack Management Server - OMS).
2. Full application HA - A 7 VM setup (not including compute driver and OpenStack Manager Server). Full HA mode provides high availability at the application layer. The HA deployment consists of 3 DB, 2 HAproxy and 2 Controller nodes. As of VIO 4.0, an extra 5 VMs will be deployed if the Ceilometer functionality is required.

OpenStack functionality are the same between deployment models. Compact mode was recently introduced in Version 3.0 and requires significantly fewer hardware resources and memory than full HA mode. You can deploy all OpenStack components in 1 VM with Compact mode. You can then achieve enterprise grade OpenStack high availability by using the HA capabilities of the vSphere infrastructure. Compact mode is useful for multiple, small deployments. Ceilometer can be enabled after completion of the VIO base stack deployment. – See tables 2.3 and 2.4.

Hardware Requirements - Full Application HA Deployment

Component	VMs	CPU	RAM (GB)	Disk Space (GB)

Integrated OpenStack Manager (OMS)	1	2 (2 per VM)	4 (4 per VM)	25
Load balancing service	2	4 (2 per VM)	8 (4 per VM)	40 (20 per VM)
Database service	3	12 (4 per VM)	48 (16 per VM)	240 (80 per VM)
Controllers	2	16 (8 per VM)	32 (16 per VM)	160 (80 per VM)
Compute service (Nova CPU)	1	2 (2 per VM)	4 (4 per VM)	20 (20 per VM)
DHCP service (VDS deployments only)	2	8 (4 per VM)	32 (16 per VM)	40 (20 per VM)
TOTAL (Not including DHCP)	11	36	96	485

Table 2.3: Hardware Requirements - Full Application HA Deployment

Hardware Requirements - Compact Mode

Component	VMs	vCPU	vRAM (GB)	vDisk Space (GB)
Integrated OpenStack Manager(OMS)	1	2 (2 per VM)	4 (4 per VM)	25 (25 per VM)
Controllers	1	8 (8 per VM)	16 (16 per VM)	80 (80 per VM)
Compute service (Nova CPU)	1	2 (2 per VM)	4 (4 per VM)	20 (20 per VM)
TOTAL	3	12	24 GB	120 GB

Table 2.4: Hardware Requirements – Compact Mode

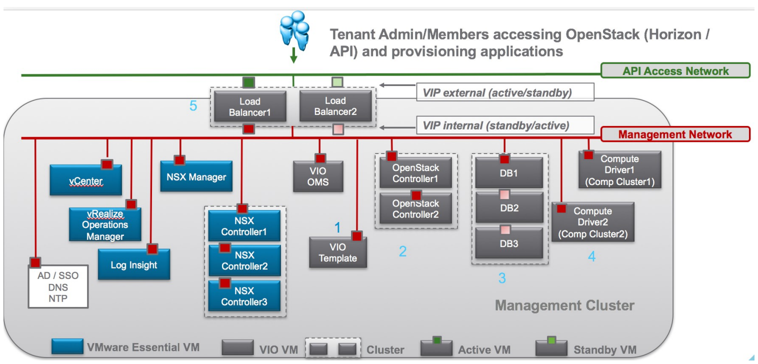


Figure 2.5: VIO Infrastructure deployments

Based on above diagram (Figure 2.5), VIO infrastructure deployments consists of following two categories of VMs:

VMware Essential VM -

- NSX Manager - The centralized network management component of NSX that provides an aggregated system view.
- NSX Controller - An advanced distributed state management system that controls virtual networks and overlay transport tunnels.
- vCenter - A Dedicated vCenter instance is not required by optimizes deployment
- vRealize Operations Manager (vROps) - Used by IT administrators monitor, troubleshoot, and manage the health and capacity of their OpenStack environment.
- vRealize Log Insight - Intelligent Log Management & Analytics tool for OpenStack and SDDC infrastructure. vRealize Log Insight is integrated with vROps to provide single pane of glass monitoring and logging of OpenStack and SDDC infrastructure.

VIO Core Services VM - VM required for VMware Integrated OpenStack in full application HA deployment. Core Services are integrated into a single Control VM in VIO Compact Mode.

2.5 Core VMware Integrated OpenStack(VIO) Components.

Below are summary descriptions of all VIO core components. Numbering corresponds to diagram above (see Figure 2.5).

2.5.1 VIO OpenStack Management Server (OMS)

OMS and VIO Template correspond to #1 on the diagram above (Figure 2.5). VIO Management Plane starts with the OpenStack Management Server (*OMS*). It serves as

- Secure jump host
- Single source of configuration truth
- Central repository for all day 2 related management tasks.

OMS registers with the vCenter and is where Ansible inventory maintains a list of VMs and corresponding roles required for upgrade and configuration update.

Day 2 management tasks are performed using VIOCLI service commands from the OMS server. VIOCLI is the main command line utility that Cloud Admins use to perform many maintenance and administration operations from the VIO Manager. viocli must always be run using sudo. Cloud Admins can use VIOCLI utility to verify db status, backup, recover and so on from the OMS node. The available subcommands can be retrieved with a simple viocli -h. The VIO OMS also stores credentials to access VIO core services securely, which avoids having to distribute plain text passwords. Private key required to access VIO control plane nodes are stored on the OMS server.

All VIO core components are created from a vSphere VM template called *VIO Template.* The *VIO Template* is made from the Ubuntu 16.04. operating system.

2.5.2 OpenStack Controller

OpenStack Controller nodes correspond to #2 on the diagram above (Figure 2.5). Controller nodes are the heart of OpenStack operations. The VIO OpenStack controllers contain:
All OpenStack API services - OpenStack API services are services that can be either consumed externally by an end user or internally by another OpenStack service. Services such as the Horizon UI, Nova, Glance, Keystone, Cinder, and Neutron are API services VIO OpenStack offers on the controller node. VIO will spin up separate Ceilometer or Mongo VMs when required. Nova communicates with vCenter using the VMwareVCDriver. Glance and Cinder uses the VMDK driver to support images and block storage. The Neutron service leverages the NSX plug-in to communicate with NSX manager. If NSX is not deployed, Neutron service communicates with the vCenter virtual distributed switch. For more information, see the OpenStack component sections.

It's important to understand that OpenStack controller nodes are the interface between OpenStack users and vSphere infrastructure. It takes OpenStack API requests and translates them to vSphere system calls. As OpenStack scales out horizontally both for scale and HA, all API services are front ended by a load balancer, which provides the boundary between external and internal API consumption.

All OpenStack schedulers - OpenStack schedulers are used to determine compute, network, and volume resource placement based on type of request. If a tenant wants to boot a VM with 4vCPU/8G Memory/20G volume, nova scheduler will look for a compute that has sufficient memory and CPU. If additional cinder services are required, cinder scheduler finds the datastore and create the volume.

Memcached - In VIO, Memcached is used to cache Keystone tokens. Memcached is an in-memory key-value store for small chunks of arbitrary data (strings, objects) from results of database calls, API calls, or page rendering. In VIO, Memcached is use to cache Keystone tokens. Token binding embeds information from an external authentication mechanism inside a token. All transactions in OpenStack start with client authenticate to keystone to fetch a token. After a token is granted, the client accesses the required OpenStack resources based on the Service Catalog using newly issued token.

VIO scales out these services by implementing an active-active mode with load balancing using a virtual IP address (HAProxy and Keepalived). One pair of Controller nodes are deployed.

2.5.3 Database and RabbitMQ Nodes

OpenStack Database and RabbitMQ resides on the same set of virtual machines. It corresponds to #3 on the diagram above (Figure 2.5).

VIO database is based on MariaDB implementation, which makes use of a three-node Galera cluster for redundancy. Database cluster is active-active-active. However, as some OpenStack services enforce table locking, reads and writes are directed to a single node through load balancers. Whenever a client consumes a resource in OpenStack, one or more of the following databases are updated (Table 2.5):

Database	Description
glance	Information about Image status (public, private, active, deleted, and so on), location, type, owner, format, and so on are stored here
heat	Information about Heat stack status - Active, deleted, timestamp of creation/deletion, tenant mapping and so on
keystone	Information about Service catalog, region, Identity provider, user, project, and federation grouping
neutron	Information about NSX DHCP binding, NSX network-subnet-port mapping, and NSX router binding based on tenant
nova	Information about compute nodes, aggregate mapping, instance, SSH key pairs, user quota, and instance metadata

Table 2.5: Database Nodes

For more information about the Galera multi-master implementation, go to http://galeracluster.com/products/technology/

2.5.4 RabbitMQ

The RabbitMQ cluster also runs on the database node. It is the default AMQP server used by all VIO services. It is an intermediary for messaging, providing applications a common platform to send and receive messages. Messages are placed into a queue and clear only after they are acknowledged. Similar to MariaDB, the VIO RabbitMQ implementation also runs in active-active-active mode. Unlike MariaDB, RabbitMQ is not front ended by a load balancer, but in a cluster environment with queues mirrored between three Rabbit nodes. Each Client, OpenStack service, is configured with IP addresses of the RabbitMQ cluster members and designates one node as primary. If the primary node isn't reachable, Rabbit Client uses one of the remaining nodes. Because queues are mirrored, messages are consumed identically regardless of the node to which a client connects. RabbitMQ can scale up to millions of connection (channels) between endpoints. Channels are TCP based, brought up/ teared down based on type of requests. RabbitMQ is not CPU intensive, but requires sufficient memory and strict network latency.

We do not recommend split RabbitMQ nodes to span across multiple data centers unless there's guaranteed network latency. RabbitMQ stores cluster information in the Mnesia database. it is an extremely light weight database that includes cluster name, credentials, and status of member nodes.

Operational stability of OpenStack is as stable as the RabbitMQ cluster itself. In an environment where there's little network stability, RabbitMQ cluster can enter into a partitioned state. Partition state happens when node members loses contact with each other. Detection and recovery from cluster partitioning is the responsibility of the Mnesia database. For more information about partition detection and recovery, go to https://www.rabbitmq.com/partitions.html.

VIO implements "Pause Minority" mode, The Erlang VM on the paused nodes continues running, but the nodes do not listen on any ports or perform any other task. They check once per second to see whether the rest of the cluster has reappeared, and start up again if it has.

The following services are connected to the messaging backend (RabbitMQ) and can be scaled-out:

- nova-scheduler
- nova-conductor
- cinder-scheduler
- cinder-volume
- neutron-server
- nova-compute consoleauth
- heat ceilometer

As a generalization, you can safely assume all core OpenStack components connects into the RabbitMQ message bus. The OpenStack subcomponents (nova-scheduler to nova-compute, for example) communicate among themselves using this hosted message queue service. They also utilize the hosted Memcached services for caching authentication tokens, for example. Component-to-component communications (Nova to Neutron for example) is done by using REST through Haproxy based on the internal Virtual IP (VIP).

For more information about RabbitMQ, go to http://www.rabbitmq.com/ha.html

2.5.5 Compute Nodes:

Compute nodes are labeled #4 on the diagram above (Figure 2.5). Compute node is the OpenStack hypervisor. Unlike traditional KVM based approach where each hypervisor is represented as a nova compute, VIO nova-compute represents a single ESXi cluster. The VMware vCenter driver enables the nova-compute service to communicate with a VMware vCenter server. This setting is defined in nova-compute.conf file for VIO 3.x based release (Future releases of VIO, post 3.x, nova-compute.conf will be removed and driver settings will be part of nova.conf).

2.5.6 Load Balancer:

VIO Load Balancers are based on a HAProxy implementation with Keepalived for high availability. NSX Load Balancers are not used here, as to not create a circular dependency. There are two sets of endpoints exposed by HAproxy, Internal and External. External endpoints are used for external tenant access (part of API Access network - example horizon), while internal endpoints are for inter-component communication (part of Management network). SSL termination is required for External endpoint, while Internal endpoints are un-encrypted. Most of OpenStack services Active/Active across the HAproxy. For the Database Service, the load balancer is configured to use a primary DB VM. In case of failure it will switch to one of the two backup DB VMs.

2.6 VIO Use Cases

The amount of control that an IT admin allows leads to four types of distinct OpenStack use cases (Table 2.6):

Traditional Enterprise Model - Based on the maturity of the development team or the criticality of aligning with central corporate policies, an IT admin may expose a subset of the OpenStack API or completely abstract the API access from the users. In such situations, the IT admin maintains control of network and security policies, while exposing VM, image, and storage consumption to OpenStack developers. The IT admin develops and shares a vRealize Automation blueprint to create OpenStack users and projects, the Provider VLAN, and Neutron Security groups, while developers integrate those predefined constructs within their application so that applications can be developed and deployed within corporate securities at all times.

Dev Cloud - IT admins give developers complete access to the infrastructure. That is, developers have access to full network, storage, and compute virtualization, similar to that with AWS. Mature public cloud consumers usually adopt this model.

Deployments ultimately go to either public or private clouds (AWS or VIO). Enterprises deploy OpenStack to meet the demands of developers and their tools.
Developers want to turn their infrastructure requests into formatted code that is included in their application code. Instead of submit a request, infrastructure teams look at the request, manually provision the infrastructure, they want documented vendor neutral APIs, support for common tools and an ecosystem of knowledge. Instead of writing different code for Dev, Test, and Production, developers want the same code to be used in Dev, Test, and Production.

NFV - Application virtualization standardized on OpenStack due to availability of vendor-neutral APIs. Primarily, traditional telecom companies adopt this model. When launching services inside or outside their footprint, performance / stability, as well as assurance infrastructure components are deployed in a consistent way are extremely critical.

Thin IaaS - A new trend in OpenStack industry, primarily driven by container adoption. Instead of providing all the features and capabilities of an OpenStack distribution, cloud admins deploy a small subset of OpenStack components (Thin IaaS, such as nova, cinder, glance, and neutron), to provide infrastructure for container orchestration. Developers have access to container orchestration API and VM, but not to OpenStack APIs. Infrastructure admins consume the OpenStack API through Heat or Terraform to create a complete set of master and slave nodes with container orchestration preinstalled. This use case provides workload-container isolation between development teams while taking advantage of Infrastructure optimization such as vMotion and DRS.

Traditional Enterprise	Dev Cloud	NFV	Thin IaaS
• Centralized IT creates the blueprints “primitives” ◦ VM Images ◦ Network architecture/features ◦ Security Profiles • Users consume the primitives • Governed approach	• Users have direct access to infrastructure ◦ Images/application templates/etc ◦ Create any network topology ◦ Security can be created by app developer • Users create & consume primitives • Restrictions with quotas	• Telco solution for virtualizing core infrastructure components ◦ VoIP, VoLTE • Must run certified applications (VNFs) • Performance and scalability is top of mind	• Users have no access to OpenStack API ◦ VMs spun up by automation tool controlled by Admin ◦ Terraform or heat • Users create & consume K8S containers • Workload Isolation and authorization
VIO			

Table 2.6: VIO Use Cases

2.7 VIO Infrastructure - Sequence of Operations to Deploy

Before you deploy VIO, it's important to ensure underlying infrastructure is prepared based on SDDC recommendations. This typically involves following steps:

Prepare your vSphere Platform (see Infrastructure Requirements section).

- Create a single vSphere data center.
- Create required VDS instances based on your design.
- Create VLAN mapping on corresponding VDS (see Infrastructure Requirements section for required VLANs).
- Deploy the management and Edge cluster (at least three hosts for Management, and two hosts for Edge).
- Deploy the compute cluster.
- Add management, Edge, and compute cluster to the corresponding VDS.
- Map datastore to ESXi clusters (at least three datastores required for the management cluster).

Deploy an NTP server for ESXi hosts and the vCenter Server to ensure time consistency.

Deploy the NSX Manager Appliance OVA on the management cluster.

- Register the NSX Manager instance with the vCenter Server. Navigate to Manage vCenter Registration from NSX Manager portal.
- Configure lookup Service to the IP Address of your Platform Services Controller and the vCenter Server connection to your vCenter Server.

Deploy NSX Controller on the management cluster. Open the vCenter Client Home page, and click **Network and Security > Installation** (three NSX Controllers are required for redundancy).

- Use Management Cluster.
- use Management PortGroup.
- Allocate three IP address from Management subnet for IP pool configuration.

Complete NSX Host Preparation. Navigate to **Network and Security > Installation > Host Preparation** (the host does not need to be in maintenance mode).

- Perform Host Preparation on the Edge and compute cluster, and not on management.
- Define Tunnel Endpoints (VTEP Interfaces). Click **Not Configured** in the **VXLAN** tab, configure VXLAN settings, and add an IP Pool for VTEP interfaces.
- Navigate to **Network and Security > Installation > Logical Network Preparation > Segment ID**, and configure the Segment ID pool to the number of VXLANs you are planning to use.

To define replication boundaries of a VXLAN network by defining a transport zone, navigate to **Network and Security > Installation > Logical Network Preparation > Transport Zones**, and click the Add (+) icon.

- Add a Global Transport Zone, set the replication mode to Unicast, and select compute and Edge clusters to be part of the Transport Zone.

2.8 VIO License and Support

VIO is available in two editions:

- DataCenter Edition (DC)
- Carrier Edition (CE)

Carrier Edition is a super set of the DataCenter edition. VIO CE can not be purchased standalone; it must be part of the vCloud NFV for OpenStack bundle.

Following table highlights features available in each edition:

Feature	DataCenter	Carrier
HW passthrough (SRIOV, PCI passthrough, GPU passthrough)	No	Yes
Tenant vDC	No	Yes
All Other Features	Yes	Yes

Table 2.7: VIO CE Features by Edition

VIO Carrier Edition will also entitle customers to Carrier Grade Support offered by VMware GSS.

Depending on the use case, VMware Integrated OpenStack requires either vSphere Enterprise Plus or NSX Advanced with vSphere Standard license.

Users who implement VXLAN overlays require NSX Advanced or Enterprise license. vSphere Standard license is sufficient when used with the NSX.
Users who are not using NSX will be limited to provider VLAN only using VDS. VDS support requires the vSphere Enterprise Plus license.
Enterprise Plus license is required if DRS support is required.

For more information on pricing, please reach out to your VMware sales team.

Section 3: Nova Integration

Nova is about accessing and managing OpenStack compute resources. With VMware Integrated OpenStack, Nova is fully integrated to work together with VMware vCenter to provision and manage Virtual Machines as well as the overall availability of the virtual infrastructure (Figure 3.1). Using the vCenter driver, Nova sees each vSphere Cluster as a single compute node. Virtual Machine Placement at the Nova level is based on vSphere Cluster, vCenter then uses DRS to place VM within the cluster. vCenter ESXi Cluster resource management can also perform directly from OpenStack Nova. Multiple ESXi compute clusters can be grouped to form a single host or multiple compute aggregates. Host aggregates can be regarded as a mechanism to partition an availability zone. Each Nova compute aggregate can be assigned different key-value pairs to enable advanced VM scheduling placement.

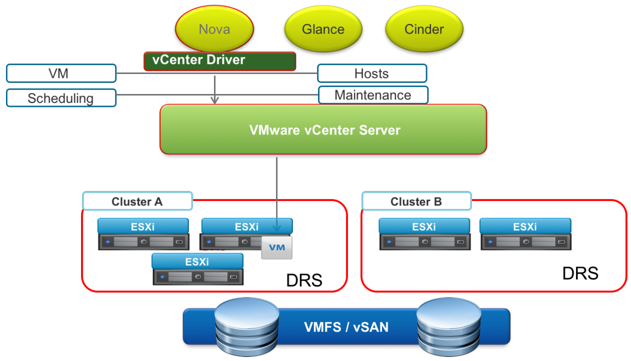


Figure 3.1: Nova Components

Nova Integration chapter will review following topics in detail:

- VIO Nova Components and vSphere Equivalent
- Nova Scheduler interaction and Over Subscription Metadata Service
- How to Place Instances into a Specific Cluster Nova Compute Scaling
- Brownfield migration options
- VM Boot and Destruction Workflow

3.1 VIO Nova Compute Components and vSphere Equivalent

As discussed in the Overview section, following components make up the OpenStack Nova stack:

- nova-api - Daemon that accepts and responds to end-user compute API requests such as VM boot, reset, resize and etc.
- nova-compute - Daemon that creates and terminates virtual machine instances.
- nova-scheduler - Daemon that takes a virtual machine instance request from the queue and determines where compute should run it on
- nova-conductor - Daemon that handles requests that need coordination, and acts as a database proxy. Nova conductor talks between Nova process.

OpenStack users consume nova services by sending requests to the nova-api service. Based on the request, nova-api either communicates externally to non-nova services through REST API or communicates with other nova services through Oslo. Nova conductor is effectively a RPC server for other nova services, it is stateless and exposes a set of APIs, most of which are database APIs, and the remaining are nova API. Client side of the RPC is typically within the nova compute. An example of the nova conductor function is when a nova compute daemon needs to update the state of a VM, instead of accessing the database directly, it coordinates the update with the nova conductor. The nova conductor acts as a security and access control layer for nova. OpenStack end users do not have direct access to nova conductors.

3.1.1 Nova Compute

Unlike a traditional KVM based approach, where each hypervisor is represented as a nova compute, the VMware vCenter driver enables the nova-compute service to communicate with a VMware vCenter Server instance. This setting is defined in nova-compute.conf file for VIO 3.x based release. In releases later than VIO 3.x, driver settings will be part of nova.conf (and nova-compute.conf will be removed). Below is a screen capture (Capture 3.1) of nova compute driver definition:

```
root@compute01:/etc/nova# more nova-compute.conf
# Ansible managed file, do not edit directly

[DEFAULT]
compute_driver = vmwareapi.VMwareVCDriver
root@compute01:/etc/nova#
```

Capture 3.1: Nova Compute driver definition

The VMware VCD Driver aggregates all ESXi hosts within each cluster and present one large hypervisor to the nova scheduler. VIO deploys a nova-compute VM for each vSphere ESXi cluster that it manages. Because individual ESXi hosts are not exposed to the nova scheduler, schedule assigns hypervisor compute hosts at granularity of the vSphere clusters, and vCenter selects the actual ESXi host within the cluster based on advanced DRS placement settings. Both automated or partially automated DRS are supported for standard VM workloads. DRS must be disabled in the case of SRIOV.

By abstracting the underlying infrastructure into the nova-compute VM, VIO is the only OpenStack distribution that implements complete control and data plane separation. Because of this separation, cloud admins do not need to visit every single hypervisor and upgrade OpenStack agents (OVS, nova, Cinder, Glance, and Ceilometer, among others) on each node. Most OpenStack upgrades can be performed within a 30 minute window as you no longer need to migrate running VMs between hypervisors to avoid downtime. OpenStack managed VMs continue to operate during any type of OpenStack upgrades. Cloud admins need not deploy excess capacity or address interoperability between various OpenStack versions.

```
viouser@prme-haas-2-vio:~$ openstack host list
+--------------+-------------+----------+
| Host Name    | Service     | Zone     |
+--------------+-------------+----------+
| controller02 | conductor   | internal |
| controller01 | conductor   | internal |
| controller02 | scheduler   | internal |
| controller01 | scheduler   | internal |
| controller02 | consoleauth | internal |
| controller01 | consoleauth | internal |
| compute01    | compute     | nova     |
| compute02    | compute     | nova     |
+--------------+-------------+----------+
viouser@prme-haas-2-vio:~$ openstack hypervisor list
+----+---------------------------------------------------+
| ID | Hypervisor Hostname                               |
+----+---------------------------------------------------+
|  4 | domain-c13.b1096bc0-9c09-41ca-bfff-31a1e54a3047   |
| 31 | domain-c3597.b1096bc0-9c09-41ca-bfff-31a1e54a3047 |
+----+---------------------------------------------------+
viouser@prme-haas-2-vio:~$
```

Capture 3.2: Nova Compute host list

3.1.2 Nova Host Aggregate

A nova host aggregate is a grouping of hypervisor or nova computes. Groupings can be done based on host hardware similarity. For example, all clusters with SSD storage backing can be grouped in one aggregate, clusters with magnetic storage in another. If hardware attributes are similar, grouping can also be based on physical location of the cluster. If there are N data centers, all ESXi clusters within a datacenter can be grouped into a single aggregate. An ESXi cluster can be in more than one host aggregate.

Host Aggregates provide a mechanism to allow administrators to assign key-value pairs, also known as metadata, to groups of computes. This key-value pair, metadata, can be used by the nova scheduler to pick the hardware that matches the client request. Host aggregates are only visible to administrators, users consume aggregates based on VM flavor definition and availability zone.

```
viouser@prme-haas-2-vio:~$
viouser@prme-haas-2-vio:~$
viouser@prme-haas-2-vio:~$ nova aggregate-create ssd nova
+----+------+-------------------+-------+--------------------------+
| Id | Name | Availability Zone | Hosts | Metadata                 |
+----+------+-------------------+-------+--------------------------+
| 7  | ssd  | nova              |       | 'availability_zone=nova' |
+----+------+-------------------+-------+--------------------------+
viouser@prme-haas-2-vio:~$ nova aggregate-set-metadata 7 ssd=true
Metadata has been successfully updated for aggregate 7.
+----+------+-------------------+-------+--------------------------------------+
| Id | Name | Availability Zone | Hosts | Metadata                             |
+----+------+-------------------+-------+--------------------------------------+
| 7  | ssd  | nova              |       | 'availability_zone=nova', 'ssd=true' |
+----+------+-------------------+-------+--------------------------------------+
viouser@prme-haas-2-vio:~$ nova aggregate-add-host 7 compute01
Host compute01 has been successfully added for aggregate 7
+----+------+-------------------+-------------+--------------------------------------+
| Id | Name | Availability Zone | Hosts       | Metadata                             |
+----+------+-------------------+-------------+--------------------------------------+
| 7  | ssd  | nova              | 'compute01' | 'availability_zone=nova', 'ssd=true' |
+----+------+-------------------+-------------+--------------------------------------+
```

Capture 3.3: Nova host aggregate

3.1.3 Nova Flavor

To consume a nova host aggregate, cloud admins needs to create and expose VM flavors so that users can request nodes that match their application characteristics (SSD, in this example). In OpenStack, a flavor defines the compute, memory, and storage capacity of the computing instances, another words size of the virtual server along with required hardware that can be launched. Nova flavor extra-spec are key-value pairs that define which compute aggregate a flavor can run. The nova scheduler finds a grouping of hardware that matches the corresponding
key-value (metadata) pairs on the compute node aggregate.

In the Host Aggregate section, we created a host aggregate with key-value (metadata) pair `ssd=true`. To create a nova flavor to consume this host aggregate, use the nova flavor-create command to create a new flavor called ssd.2v8G20G with a flavor ID of 23, 8 GB of RAM, 20 GB root SSD disk, and 2 vCPUs. To ensure that nova scheduler selects hosts in the SSD aggregate, set the flavor key-value pairs to match the host aggr egates (ssd=true). You can set a key-value pair on a flavor by using the nova flavor-key command.

```
viouser@prme-haas-2-vio:~$ nova flavor-create ssd.2v8G20G 23 8192 20 2
+----+-------------+-----------+------+-----------+------+-------+-------------+-----------+
| ID | Name        | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public |
+----+-------------+-----------+------+-----------+------+-------+-------------+-----------+
| 23 | ssd.2v8G20G | 8192      | 20   | 0         |      | 2     | 1.0         | True      |
+----+-------------+-----------+------+-----------+------+-------+-------------+-----------+
viouser@prme-haas-2-vio:~$ nova flavor-key ssd.2v8G20G set ssd=true
viouser@prme-haas-2-vio:~$ nova flavor-show ssd.2v8G20G
+----------------------------+-----------------+
| Property                   | Value           |
+----------------------------+-----------------+
| OS-FLV-DISABLED:disabled   | False           |
| OS-FLV-EXT-DATA:ephemeral  | 0               |
| disk                       | 20              |
| extra_specs                | {"ssd": "true"} |
| id                         | 23              |
| name                       | ssd.2v8G20G     |
| os-flavor-access:is_public | True            |
| ram                        | 8192            |
| rxtx_factor                | 1.0             |
| swap                       |                 |
| vcpus                      | 2               |
+----------------------------+-----------------+
viouser@prme-haas-2-vio:~$
```

Capture 3.4: Nova host flavor

3.1.4 Nova Scheduler Interaction and Over Subscription

OpenStack uses the nova-scheduler service to determine where to place a new workload or modification to an existing workload request, for example during a live migration or when a new VM starts up.

Another simplistic way to describe a nova-scheduler is that it's a filter (Figure 3.2). Based on the type of request, it eliminates nova-computes that can not satisfy the workload request and returns those that could satisfy the workload request.

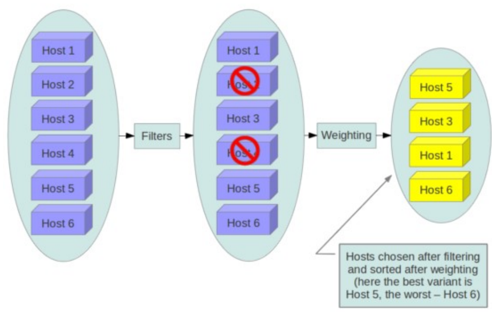


Figure 3.2: Nova Scheduler

Nova scheduler settings are part of /etc/nova/nova.conf file. In the default VIO configuration, this scheduler makes available

#nova_scheduler_default_filters: RetryFilter, AvailabilityZoneFilter, RamFilter, ComputeFilter, ComputeCapabilitiesFilter, ImagePropertiesFilter, ServerGroupAntiAffinityFilter, ServerGroupAffinityFilter

hosts for operation that meet all of following criterias, also known as filters:

The following definitions are sourced from Openstack.org:

- Have not been attempted for scheduling purposes (RetryFilter).

- Are in the requested availability zone (AvailabilityZoneFilter).
- Have sufficient RAM available (RamFilter).
- Can service the request (ComputeFilter).
- Satisfy the extra specs associated with the instance type (ComputeCapabilitiesFilter).
- Satisfy any architecture, hypervisor type, or virtual machine mode properties specified on the instance's image properties (ImagePropertiesFilter).
- Are on a different host than other instances of a group (if AntiAffinity is requested as part of VM boot request - ServerGroupAntiAffinityFilter).
- Are in a set of group hosts (if Affinity is requested as part of VM boot request - ServerGroupAffinityFilter).

For more information, go to upstream Nova Scheduler Filter documentation.

Nova scheduler also controls host CPU, memory and disk over-subscription. Over-subscription is the ability to place multiple devices to the same physical resource to optimize usage. The following global settings in nova.conf impacts all OpenStack hosts.

```
# Virtual CPU to physical CPU allocation ratio which affects all CPU filters.
#nova_cpu_allocation_ratio: 16
# Virtual Memory to physical Memory allocation ratio which affects all CPU filters.
#nova_ram_allocation_ratio: 1.5
# This is the virtual disk to physical disk allocation ratio used by the
# disk_filter.py script to determine if a host has sufficient disk space to fit
# a requested instance. This can be set per-compute, or if set to 0.0, the
# value set on the scheduler node(s) will be used and defaulted to 1.0
#disk_allocation_ratio = 0.0
```

Capture 3.5: nova.conf settings

You can also control over-subscription for each host aggregate. Host aggregates are a grouping of hosts with similar hardware, Over-subscription and VM placement characteristics. The following filters are required to enable host aggregates level over-subscription management:

AggregateCoreFilter - Filters host by CPU core numbers with a per-aggregate cpu_allocation_ratio value. If the per-aggregate value is not found, the value falls back to the global setting. If the host is in more than one aggregate and more than one value is found, the minimum value will be used.

AggregateDiskFilter - Filters host by disk allocation with a per-aggregate disk_allocation_ratio value. If the per-aggregate value is not found, the value falls back to the global setting. If the host is in more than one aggregate and more than one value is found, the minimum value will be used.

AggregateRamFilter - Filters host by RAM allocation of instances with a per-aggregate ram_allocation_ratio value. If the per-aggregate value is not found, the value falls back to the global setting. If the host is in more than one aggregate and thus more than one value is found, the minimum value will be used.
Below is an example of no CPU over-subscription and 1.5X memory over-subscription:

```
viouser@prme-haas-2-vio:~$ nova aggregate-details 19
+----+-------------+-------------------+-------------+-----------------------------------------+
| Id | Name        | Availability Zone | Hosts       | Metadata                                |
+----+-------------+-------------------+-------------+-----------------------------------------+
| 19 | new-cluster | nova              | 'compute02' | 'availability_zone=nova', 'ssd=true'    |
+----+-------------+-------------------+-------------+-----------------------------------------+
viouser@prme-haas-2-vio:~$ nova aggregate-set-metadata 19 cpu_allocation_ratio=1.0 ram_allocation_ratio=1.5
Metadata has been successfully updated for aggregate 19.
+----+-------------+-------------------+-------------+-----------------------------------------------------------------------------------------------------+
| Id | Name        | Availability Zone | Hosts       | Metadata                                                                                            |
+----+-------------+-------------------+-------------+-----------------------------------------------------------------------------------------------------+
| 19 | new-cluster | nova              | 'compute02' | 'availability_zone=nova', 'cpu_allocation_ratio=1.0', 'ram_allocation_ratio=1.5', 'ssd=true'        |
+----+-------------+-------------------+-------------+-----------------------------------------------------------------------------------------------------+
viouser@prme-haas-2-vio:~$
```

Capture 3.6: no CPU over-subscription and 1.5X memory over-subscription

Below is an example of 4XCPU over-subscription and no memory over-subscription

```
viouser@prme-haas-2-vio:~$ nova aggregate-set-metadata 7 cpu_allocation_ratio=4.0 ram_allocation_ratio=1.0
Metadata has been successfully updated for aggregate 7.
+----+------+-------------------+--------------------------+------------------------------------------------------------------------------------------------+
| Id | Name | Availability Zone | Hosts                    | Metadata                                                                                       |
+----+------+-------------------+--------------------------+------------------------------------------------------------------------------------------------+
| 7  | ssd  | nova              | 'compute01', 'compute02' | 'availability_zone=nova', 'cpu_allocation_ratio=4.0', 'ram_allocation_ratio=1.0', 'ssd=true' |
+----+------+-------------------+--------------------------+------------------------------------------------------------------------------------------------+
viouser@prme-haas-2-vio:~$
```

Capture 3.7: 4XCPU over-subscription and no memory over-subscription

However, compute02 is part of both aggregates, and so, only the minimal value is used. As a result, compute02 is treated as

```
cpu_allocation_ratio=1, ram_allocation_ratio=1
```

For a standard out of box design or implementation, do not allocate nova compute across aggregates for ease of day 2 support.

3.1.5 Post Deployment Nova Scheduler Filter Updates

For Aggregated related filtering, you need to update Nova scheduler filters. To generalize this process using Aggregate filter as an example:

- If you have not used the custom.yml file, copy the /var/lib/vio/ansible/custom/custom.yml.sample to /opt/vmware/vio/custom/custom.yml
- Append AggregateInstanceExtraSpecsFilter, AggregateCoreFilter, AggregateRamFilter, and AggregateDiskFilter the nova_scheduler_default_filters in /opt/vmware/vio/custom/custom.yml
- Remove the "#" to uncomment the following line:
  - nova_scheduler_default_filters: RetryFilter, AvailabilityZoneFilter, RamFilter, ComputeFilter, ComputeCapabilitiesFilter, ImagePropertiesFilter, ServerGroupAntiAffinityFilter, ServerGroupAffinityFilter, PciPassthroughFilter,AggregateInstanceExtraSpecsFilter,AggregateCoreFilter, AggregateRamFilter and AggregateDiskFilter
- To save and re-run the Ansible deployment, run the following command:

```
viocli deployment configure --tags nova_api_config
```

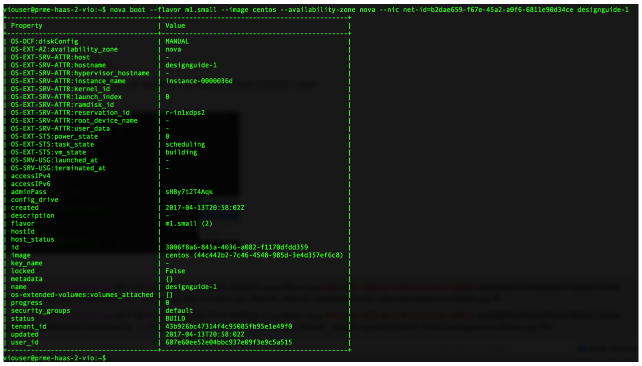
At the opposite end of the spectrum, instead of over-subscription, Cloud Administrators can assign dedicated compute hosts by tenants. You can use AggregateMultiTenancyIsolation filter to control tenant to host placement. If an aggregate that has the filter_tenant_id metadata key, the hosts in the aggregate create instances from only that tenant or list of tenants. No other tenant will be allowed on these hosts.

3.1.6 Examples of Nova Scheduler Operations and Output

We will highlight details of OpenStack Nova Scheduler through the use of VM boot up. We will demonstrate a successful VM boot up where a valid host is found, and we will also demonstrate a failed boot up where requested resources are not available.

3.1.6.1 Successful VM Boot-up

The following is an example of a successful VM boot request and the corresponding nova scheduler output. The command in the example attempts to boot a generic m1.small CentOS VM:



Capture 3.8: boot a generic m1.small CentOS VM

Nova schedule is using the default filter. The following debug information shows that two hosts were found with each matching nova filter.

/var/log/nova/nova-scheduler.log:2017-04-13 20:09:38.739 31151 DEBUG nova.filters [req-7f5d1c34-1552-4e1d-9f1d-07cdb17d09b5 b6db8b00c83546a090e4766b8e12d4d5 60da9d589b5a4f848523576d6d0a5a1b - - -] Starting with 2 host(s) get_filtered_objects /usr/lib/python2.7/dist-packages/nova/filters.py:70

/var/log/nova/nova-scheduler.log:2017-04-13 20:09:38.739 31151 DEBUG nova.filters [req-7f5d1c34-1552-4e1d-9f1d-07cdb17d09b5 b6db8b00c83546a090e4766b8e12d4d5 60da9d589b5a4f848523576d6d0a5a1b - - -] Filter RetryFilter returned 2 host(s) get_filtered_objects /usr/lib/python2.7/dist-packages/nova/filters.py:104

/var/log/nova/nova-scheduler.log:2017-04-13 20:09:38.740 31151 DEBUG nova.filters [req-7f5d1c34-1552-4e1d-9f1d-07cdb17d09b5 b6db8b00c83546a090e4766b8e12d4d5 60da9d589b5a4f848523576d6d0a5a1b - - -] Filter AvailabilityZoneFilter returned 2 host(s) get_filtered_objects /usr/lib/python2.7/dist-packages/nova/filters.py:104

/var/log/nova/nova-scheduler.log:2017-04-13 20:09:38.741 31151 DEBUG nova.filters [req-7f5d1c34-1552-4e1d-9f1d-07cdb17d09b5 b6db8b00c83546a090e4766b8e12d4d5 60da9d589b5a4f848523576d6d0a5a1b - - -] Filter RamFilter returned 2 host(s) get_filtered_objects /usr/lib/python2.7/dist-packages/nova/filters.py:104

/var/log/nova/nova-scheduler.log:2017-04-13 20:09:38.741 31151 DEBUG nova.filters [req-7f5d1c34-1552-4e1d-9f1d-07cdb17d09b5 b6db8b00c83546a090e4766b8e12d4d5 60da9d589b5a4f848523576d6d0a5a1b - - -] Filter DiskFilter returned 2 host(s) get_filtered_objects /usr/lib/python2.7/dist-packages/nova/filters.py:104

/var/log/nova/nova-scheduler.log:2017-04-13 20:09:38.742 31151 DEBUG nova.filters [req-7f5d1c34-1552-4e1d-9f1d-07cdb17d09b5 b6db8b00c83546a090e4766b8e12d4d5 60da9d589b5a4f848523576d6d0a5a1b - - -] Filter ComputeFilter returned 2 host(s) get_filtered_objects /usr/lib/python2.7/dist-packages/nova/filters.py:104

/var/log/nova/nova-scheduler.log:2017-04-13 20:09:38.742 31151 DEBUG nova.filters [req-7f5d1c34-1552-4e1d-9f1d-07cdb17d09b5 b6db8b00c83546a090e4766b8e12d4d5 60da9d589b5a4f848523576d6d0a5a1b - - -] Filter ComputeCapabilitiesFilter returned 2 host(s) get_filtered_objects /usr/lib/python2.7/dist-packages/nova/filters.py:104

/var/log/nova/nova-scheduler.log:2017-04-13 20:09:38.743 31151 DEBUG nova.filters [req-7f5d1c34-1552-4e1d-9f1d-07cdb17d09b5 b6db8b00c83546a090e4766b8e12d4d5 60da9d589b5a4f848523576d6d0a5a1b - - -] Filter ImagePropertiesFilter returned 2 host(s) get_filtered_objects /usr/lib/python2.7/dist-packages/nova/filters.py:104

/var/log/nova/nova-scheduler.log:2017-04-13 20:09:38.743 31151 DEBUG nova.filters [req-7f5d1c34-1552-4e1d-9f1d-07cdb17d09b5 b6db8b00c83546a090e4766b8e12d4d5 60da9d589b5a4f848523576d6d0a5a1b - - -] Filter ServerGroupAntiAffinityFilter returned 2 host(s) get_filtered_objects /usr/lib/python2.7/dist-packages/nova/filters.py:104

/var/log/nova/nova-scheduler.log:2017-04-13 20:09:38.744 31151 DEBUG nova.filters [req-7f5d1c34-1552-4e1d-9f1d-07cdb17d09b5 b6db8b00c83546a090e4766b8e12d4d5 60da9d589b5a4f848523576d6d0a5a1b - - -] Filter ServerGroupAffinityFilter returned 2 host(s) get_filtered_objects /usr/lib/python2.7/dist-packages/nova/filters.py:104

/var/log/nova/nova-scheduler.log:2017-04-13 20:09:38.744 31151 DEBUG nova.filters [req-7f5d1c34-1552-4e1d-9f1d-07cdb17d09b5 b6db8b00c83546a090e4766b8e12d4d5 60da9d589b5a4f848523576d6d0a5a1b - - -] Filter PciPassthroughFilter returned 2 host(s) get_filtered_objects /usr/lib/python2.7/dist-packages/nova/filters.py:104

Capture 3.9: Nova debug

In this situation, Nova scheduler compares the two hosts based on relative weight:

Compute02 has a weight of 1.9986 as there's only a single running instance.

Compute01 has a weight of 1.8910 as there are three running instances

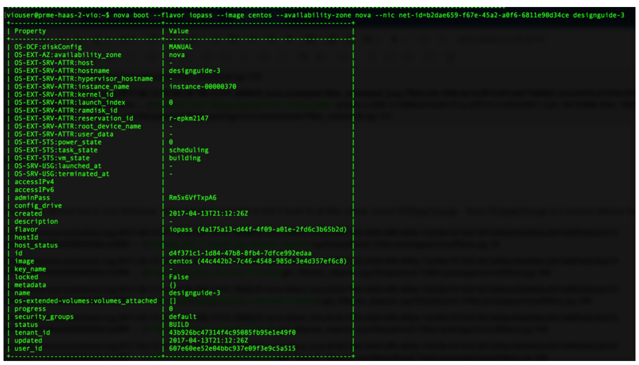
As a result, Compute02 is preferred as it has a higher weight.

```
/var/log/nova/nova-scheduler.log:2017-04-13 20:09:38.745 31151 DEBUG nova.scheduler.filter_scheduler [req-7f5d1c34-1552-4e1d-9f1d-07cdb17d09b5 b6db8b00c83546a090e4766b8e12d4d5 60da9d589b5a4f848523576d6d0a5a1b - - -] Filtered [(compute02, domain-c3597.b1096bc0-9c09-41ca-bfff-31a1e54a3047) ram: 94722MB disk: 790528MB io_ops: 0 instances: 1, (compute01, domain-c13.b1096bc0-9c09-41ca-bfff-31a1e54a3047) ram: 93962MB disk: 711680MB io_ops: 0 instances: 3] _schedule /usr/lib/python2.7/dist-packages/nova/scheduler/filter_scheduler.py:118

/var/log/nova/nova-scheduler.log:2017-04-13 20:09:38.746 31151 DEBUG nova.scheduler.filter_scheduler [req-7f5d1c34-1552-4e1d-9f1d-07cdb17d09b5 b6db8b00c83546a090e4766b8e12d4d5 60da9d589b5a4f848523576d6d0a5a1b - - -] Weighed [WeighedHost [host: (compute02, domain-c3597.b1096bc0-9c09-41ca-bfff-31a1e54a3047) ram: 94722MB disk: 790528MB io_ops: 0 instances: 1, weight: 1.99867466831], WeighedHost [host: (compute01, domain-c13.b1096bc0-9c09-41ca-bfff-31a1e54a3047) ram: 93962MB disk: 711680MB io_ops: 0 instances: 3, weight: 1.89103954151]] _schedule /usr/lib/python2.7/dist-packages/nova/scheduler/filter_scheduler.py:123

/var/log/nova/nova-scheduler.log:2017-04-13 20:09:38.746 31151 DEBUG nova.scheduler.filter_scheduler [req-7f5d1c34-1552-4e1d-9f1d-07cdb17d09b5 b6db8b00c83546a090e4766b8e12d4d5 60da9d589b5a4f848523576d6d0a5a1b - - -] Selec ted host: WeighedHost [host: (compute02, domain-c3597.b1096bc0-9c09-41ca-bfff-31a1e54a3047) ram: 94722MB disk: 790528MB io_ops: 0 instances: 1, weight: 1.99867466831] _schedule /usr/lib/python2.7/dist-packages/nova/scheduler/filter_scheduler.py:131
```

Capture 3.10: Nova scheduler compares the two hosts based on relative weight

3.1.6.2 **Failed VM Boot-up**

The following is an example of a VM boot failure because nova Scheduler is unable to find the required resources. The VM requested is a CentOS that has direct access to Physical PCI slot.



Capture 3.11: Example of VM boot failure

```
viouser@prme-haas-2-vio:~$ nova flavor-show iopass
+----------------------------+--------------------------------------+
| Property                   | Value                                |
+----------------------------+--------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                |
| OS-FLV-EXT-DATA:ephemeral  | 0                                    |
| disk                       | 20                                   |
| extra_specs                | {"pci_passthrough:alias": "sriov:1"} |
| id                         | 4a175a13-d44f-4f09-a01e-2fd6c3b65b2d |
| name                       | iopass                               |
| os-flavor-access:is_public | False                                |
| ram                        | 2048                                 |
| rxtx_factor                | 1.0                                  |
| swap                       |                                      |
| vcpus                      | 1                                    |
+----------------------------+--------------------------------------+
viouser@prme-haas-2-vio:~$
```

Capture 3.12: VM boot failure

Nova schedule is using the default filter. The following debug information shows that Nova scheduler was able to find two hosts for all filter criteria except PCIPassThrough. Since, PCIpassThrough is a required attribute for the new machine, boot failed.

```
/var/log/nova/nova-scheduler.log:2017-04-13 21:12:26.391 31151 DEBUG nova.filters [req-8c5e739d-45df-4ffb-806a-72d39e7013d3 607e60ee52e04bbc937e09f3e9c5a515 43b926bc47314f4c95085fb95e1e49f0 - - -] Starting with 2 host(s) get_filtered_objects /usr/lib/py thon2.7/dist-packages/nova/filters.py:70

/var/log/nova/nova-scheduler.log:2017-04-13 21:12:26.391 31151 DEBUG nova.filters [req-8c5e739d-45df-4ffb-806a-72d39e7013d3 607e60ee52e04bbc937e09f3e9c5a515 43b926bc47314f4c95085fb95e1e49f0 - - -] Filter RetryFilter returned 2 host(s) get_filtered_objec ts /usr/lib/python2.7/dist-packages/nova/filters.py:104

/var/log/nova/nova-scheduler.log:2017-04-13 21:12:26.392 31151 DEBUG nova.filters [req-8c5e739d-45df-4ffb-806a-72d39e7013d3 607e60ee52e04bbc937e09f3e9c5a515 43b926bc47314f4c95085fb95e1e49f0 - - -] Filter AvailabilityZoneFilter returned 2 host(s) get_filtered_objects /usr/lib/python2.7/dist-packages/nova/filters.py:104

/var/log/nova/nova-scheduler.log:2017-04-13 21:12:26.392 31151 DEBUG nova.filters [req-8c5e739d-45df-4ffb-806a-72d39e7013d3 607e60ee52e04bbc937e09f3e9c5a515 43b926bc47314f4c95085fb95e1e49f0 - - -] Filter RamFilter returned 2 host(s) get_filtered_object s /usr/lib/python2.7/dist-packages/nova/filters.py:104

/var/log/nova/nova-scheduler.log:2017-04-13 21:12:26.393 31151 DEBUG nova.filters [req-8c5e739d-45df-4ffb-806a-72d39e7013d3 607e60ee52e04bbc937e09f3e9c5a515 43b926bc47314f4c95085fb95e1e49f0 - - -] Filter DiskFilter returned 2 host(s) get_filtered_object s /usr/lib/python2.7/dist-packages/nova/filters.py:104

/var/log/nova/nova-scheduler.log:2017-04-13 21:12:26.393 31151 DEBUG nova.filters [req-8c5e739d-45df-4ffb-806a-72d39e7013d3 607e60ee52e04bbc937e09f3e9c5a515 43b926bc47314f4c95085fb95e1e49f0 - - -] Filter ComputeFilter returned 2 host(s) get_filtered_o bjects /usr/lib/python2.7/dist-packages/nova/filters.py:104

/var/log/nova/nova-scheduler.log:2017-04-13 21:12:26.393 31151 DEBUG nova.filters [req-8c5e739d-45df-4ffb-806a-72d39e7013d3 607e60ee52e04bbc937e09f3e9c5a515 43b926bc47314f4c95085fb95e1e49f0 - - -] Filter ComputeCapabilitiesFilter returned 2 host(s) ge t_filtered_objects /usr/lib/python2.7/dist-packages/nova/filters.py:104

/var/log/nova/nova-scheduler.log:2017-04-13 21:12:26.394 31151 DEBUG nova.filters [req-8c5e739d-45df-4ffb-806a-72d39e7013d3 607e60ee52e04bbc937e09f3e9c5a515 43b926bc47314f4c95085fb95e1e49f0 - - -] Filter ImagePropertiesFilter returned 2 host(s) get_filt ered_objects /usr/lib/python2.7/dist-packages/nova/filters.py:104

/var/log/nova/nova-scheduler.log:2017-04-13 21:12:26.395 31151 DEBUG nova.filters [req-8c5e739d-45df-4ffb-806a-72d39e7013d3 607e60ee52e04bbc937e09f3e9c5a515 43b926bc47314f4c95085fb95e1e49f0 - - -] Filter ServerGroupAntiAffinityFilter returned 2 host(s) get_filtered_objects /usr/lib/python2.7/dist-packages/nova/filters.py:104

/var/log/nova/nova-scheduler.log:2017-04-13 21:12:26.396 31151 DEBUG nova.filters [req-8c5e739d-45df-4ffb-806a-72d39e7013d3 607e60ee52e04bbc937e09f3e9c5a515 43b926bc47314f4c95085fb95e1e49f0 - - -] Filter ServerGroupAffinityFilter returned 2 host(s) get_filtered_objects /usr/lib/python2.7/dist-packages/nova/filters.py:104

/var/log/nova/nova-scheduler.log:2017-04-13 21:12:26.396 31151 DEBUG nova.scheduler.filters.pci_passthrough_filter [req-8c5e739d-45df-4ffb-806a-72d39e7013d3 607e60ee52e04bbc937e09f3e9c5a515 43b926bc47314f4c95085fb95e1e49f0 - - -] (comput e02, domain-c3597.b1096bc0-9c09-41ca-bfff-31a1e54a3047) ram: 94024MB disk: 790528MB io_ops: 0 instances: 3 doesn't have the required PCI devices (InstancePCIRequests(instance_uuid=d4f371c1-1d84-47b8-8fb4-7dfce992edaa,requests=[InstancePCIRequest])) host_passes /usr/lib/python2.7/dist-packages/nova/scheduler/filters/pci_passthrough_filter.py:51

/var/log/nova/nova-scheduler.log:2017-04-13 21:12:26.397 31151 DEBUG nova.scheduler.filters.pci_passthrough_filter [req-8c5e739d-45df-4ffb-806a-72d39e7013d3 607e60ee52e04bbc937e09f3e9c5a515 43b926bc47314f4c95085fb95e1e49f0 - - -] (comput e01, domain-c13.b1096bc0-9c09-41ca-bfff-31a1e54a3047) ram: 93969MB disk: 711680MB io_ops: 0 instances: 3 doesn't have the required PCI devices (InstancePCIRequests(instance_uuid=d4f371c1-1d84-47b8-8fb4-7dfce992edaa,requests=[InstancePCIRequest])
) host_passes /usr/lib/python2.7/dist-packages/nova/scheduler/filters/pci_passthrough_filter.py:51

/var/log/nova/nova-scheduler.log:2017-04-13 21:12:26.397 31151 INFO nova.filters [req-8c5e739d-45df-4ffb-806a-72d39e7013d3 607e60ee52e04bbc937e09f3e9c5a515 43b926bc47314f4c95085fb95e1e49f0 - - -] Filter PciPassthroughFilter returned 0 hosts

/var/log/nova/nova-scheduler.log:2017-04-13 21:12:26.398 31151 DEBUG nova.filters [req-8c5e739d-45df-4ffb-806a-72d39e7013d3 607e60ee52e04bbc937e09f3e9c5a515 43b926bc47314f4c95085fb95e1e49f0 - - -] Filtering removed all hosts for the request with instance ID 'd4f371c1-1d84-47b8-8fb4-7dfce992edaa'. Filter results: [('RetryFilter', [(u'compute02', u'domain-c3597.b1096bc0-9c09-41ca-bfff-31a1e54a3047'), (u'compute01', u'domain-c13.b1096bc0-9c09-41ca-bfff-31a1e54a3047')]), ('AvailabilityZoneFilter', [(u'compute02', u'domain-c3597.b1096bc0-9c09-41ca-bfff-31a1e54a3047'), (u'compute01', u'domain-c13.b1096bc0-9c09-41ca-bfff-31a1e54a3047')]), ('RamFilter', [(u'compute02', u'domain-c3597.b1096bc0-9c09-41ca-bfff-31a1e54a3047'), (u'compute01', u'domain-c13.b1096bc0-9c09-41ca-bfff-31a1e54a3047')]), ('DiskFilter', [(u'compute02', u'domain-c3597.b1096bc0-9c09-41ca-bfff-31a1e54a3047'), (u'compute01', u'domain-c13.b1096bc0-9c09-41ca-bfff-31a1e54a3047')]), ('ComputeFilter', [(u'compute02', u'domain-c3597.b1096bc0-9c09-41ca-bfff-31a1e54a3047'), (u'compute01', u'domain-c13.b1096bc0-9c09-41ca-bfff-31a1e54a3047')]), ('ComputeCapabilitiesFilter', [(u'compute02', u'domain-c3597.b1096bc0-9c09-41ca-bfff-31a1e54a3047'), (u'compute01', u'domain-c13.b1096bc0-9c09-41ca-bfff-31a1e54a3047')]), ('ImagePropertiesFilter', [(u'compute02', u'domain-c3597.b1096bc0-9c09-41ca-bfff-31a1e54a3047'), (u'compute01', u'domain-c13.b1096bc0-9c09-41ca-bfff-31a1e54a3047')]), ('ServerGroupAntiAffinityFilter', [(u'compute02', u'domain-c3597.b1096bc0-9c09-41ca-bfff-31a1e54a3047'), (u'compute01', u'domain-c13.b1096bc0-9c09-41ca-bfff-31a1e54a3047')]), ('ServerGroupAffinityFilter', [(u'compute02', u'domain-c3597.b1096bc0-9c09-41ca-bfff-31a1e54a3047'), (u'compute01', u'domain-c13.b1096bc0-9c09-41ca-bfff-31a1e54a3047')]), ('PciPassthroughFilter', None)] get_filtered_objects /usr/lib/python2.7/dist-packages/nova/filters.py:129
```

```
/var/log/nova/nova-scheduler.log:2017-04-13 21:12:26.398 31151 INFO nova.filters [req-8c5e739d-45df-4ffb-806a-72d39e7013d3 607e60ee52e04bbc937e09f3e9c5a515 43b926bc47314f4c95085fb95e1e49f0 - - -] Filtering removed all hosts for the request with instance ID 'd4f371c1-1d84-47b8-8fb4-7dfce992edaa'. Filter results: ['RetryFilter: (start: 2, end: 2)', 'AvailabilityZoneFilter: (start: 2, end: 2)', 'RamFilter: (start: 2, end: 2)', 'DiskFilter: (start: 2, end: 2)', 'ComputeFilter: (start: 2, end: 2)', 'ComputeCapabilitiesFilter: (start: 2, end: 2)', 'ImagePropertiesFilter: (start: 2, end: 2)', 'ServerGroupAntiAffinityFilter: (start: 2, end: 2)', 'ServerGroupAffinityFilter: (start: 2, end: 2)', 'PciPassthroughFilter: (start: 2, end: 0)']

/var/log/nova/nova-scheduler.log:2017-04-13 21:12:26.399 31151 DEBUG nova.scheduler.filter_scheduler [req-8c5e739d-45df-4ffb-806a-72d39e7013d3 607e60ee52e04bbc937e09f3e9c5a515 43b926bc47314f4c95085fb95e1e49f0 - - -] There are 0 hosts available but 1 instances requested to build. select_destinations /usr/lib/python2.7/dist-packages/nova/scheduler/filter_scheduler.py:71
```

Capture 3.13: Boot failed because Nova scheduler coundn't find required attribute for the new machine

3.1.8 Cell

In Ocata and later OpenStack release, OpenStack computes and Nova aggregates are organized into cells (Fig 3.3). Only a single cell is supported in Ocata. We are working actively with upstream OpenStack community and with our largest customer to see when multi-cell (v2) deployments are ready to meet production needs. With cells, Nova creates a hierarchy of Nova topologies each with dedicated database, message queue, and compute nodes. Information applicable to all cells such as Nova flavor, resource providers, user keys, and so on is stored in the API table or global cell. Instance-specific details such as security group, floating IP, and host aggregates are part of the cell-level tables.

To schedule instance builds to compute hosts, Nova and the scheduler need to take into account that hosts are grouped into cells. To create an instance:

- We first need to know which cell to create host in - Nova scheduler filters and node allocation described in the previous sections remains the same. The scheduler continues to return a (host, node) tuple, and the calling service looks up the host in a mapping table to determine which cell it is in.
- The conductor calls the scheduler, creates the instance in the cell, and passes the build request to it. The new boot workflow is similar to the following:

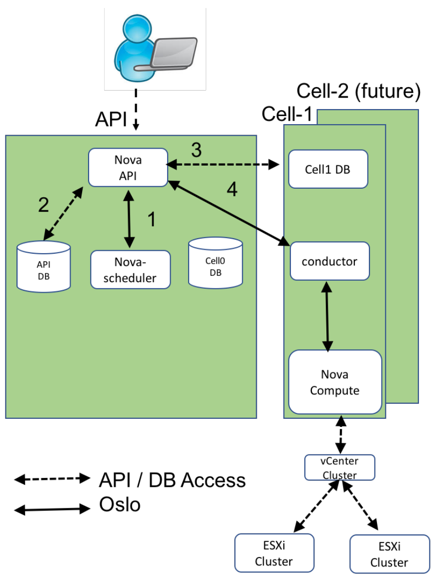


Figure 3.3: OpenStack computes and Nova aggregates are organized into cells

1. Schedule the instance
2. Record the cell to which the instance is scheduled
3. Create instance record
4. Send RPC message to the conductor to Build

3.1.9 Placement API

With Ocala and later releases, nova-placement-api is separated from nova-api code, and runs as a separate service. nova-placement-api is used to track resource provider inventories and usage:

Compute node Share storage pool External IP pool

For example, an instance created on a compute node might consume resources such as RAM and CPU from a compute node resource provider, disk from an external shared storage pool resource provider, and IP addresses from an external IP pool resource provider. The types of resources consumed are tracked as classes. The service provides a set of standard resource classes (for example —DISK_GB, —MEMORY_MB, and —V CPU) and provides the ability to define custom resource classes as needed.

OpenStack scheduler queries the Placement API service as part of scheduling process. In Ocata based VIO releases, if the scheduler is unable to make requests to the Placement API, NoValidHost errors occur.

The following output displays a successful boot sequence with Placement API and Nova Scheduler:==> nova-placement-api.log <==

2017-04-19 18:36:46.960 31996 INFO nova.api.openstack.placement.requestlog [req-0c77486e-e24a-4073-b338-846301fa0b30 a563582e1 1.4

==> nova-scheduler.log <==

2017-04-19 18:36:46.988 28704 DEBUG nova.filters [req-b298ca23-c541-407a-83c4-5eda07fd850a 4a793f3cac1e44b6957f7a91f0af7457 6c594

2017-04-19 18:36:46.988 28704 DEBUG nova.scheduler.filters.retry_filter [req-b298ca23-c541-407a-83c4-5eda07fd850a 4a793f3cac1e44b6957

2017-04-19 18:36:46.989 28704 DEBUG nova.filters [req-b298ca23-c541-407a-83c4-5eda07fd850a 4a793f3cac1e44b6957f7a91f0af7457 6c594

2017-04-19 18:36:46.990 28704 DEBUG nova.filters [req-b298ca23-c541-407a-83c4-5eda07fd850a 4a793f3cac1e44b6957f7a91f0af7457 6c594

2017-04-19 18:36:46.991 28704 DEBUG nova.filters [req-b298ca23-c541-407a-83c4-5eda07fd850a 4a793f3cac1e44b6957f7a91f0af7457 6c594

2017-04-19 18:36:46.992 28704 DEBUG nova.filters [req-b298ca23-c541-407a-83c4-5eda07fd850a 4a793f3cac1e44b6957f7a91f0af7457 6c594

2017-04-19 18:36:46.992 28704 DEBUG nova.filters [req-b298ca23-c541-407a-83c4-5eda07fd850a 4a793f3cac1e44b6957f7a91f0af7457 6c594

2017-04-19 18:36:46.993 28704 DEBUG nova.filters [req-b298ca23-c541-407a-83c4-5eda07fd850a 4a793f3cac1e44b6957f7a91f0af7457 6c594

2017-04-19 18:36:46.994 28704 DEBUG nova.filters [req-b298ca23-c541-407a-83c4-5eda07fd850a 4a793f3cac1e44b6957f7a91f0af7457 6c594

```
2017-04-19 18:36:46.995 28704 DEBUG nova.filters [req-b298ca23-c541-407a-83c4-5eda07fd850a
4a793f3cac1e44b6957f7a91f0af7457 6c594

2017-04-19 18:36:46.996 28704 DEBUG nova.filters [req-b298ca23-c541-407a-83c4-5eda07fd850a
4a793f3cac1e44b6957f7a91f0af7457 6c594

2017-04-19 18:36:46.996 28704 DEBUG nova.filters [req-b298ca23-c541-407a-83c4-5eda07fd850a
4a793f3cac1e44b6957f7a91f0af7457 6c594

2017-04-19 18:36:46.997 28704 DEBUG nova.scheduler.filter_scheduler [req-b298ca23-c541-407a-83c4-5eda07fd850a
4a793f3cac1e44b6957f

2017-04-19 18:36:46.999 28704 DEBUG nova.scheduler.filter_scheduler [req-b298ca23-c541-407a-83c4-5eda07fd850a
4a793f3cac1e44b6957f

==> nova-conductor.log <==

2017-04-19 18:36:47.007 28626 DEBUG oslo_messaging._drivers.amqpdriver [-] received reply msg_id:
60cad8b85f38477dbd3c7a1031f2db0a

2017-04-19 18:36:47.053 28626 DEBUG oslo_db.sqlalchemy.engines [req-b298ca23-c541-407a-83c4-5eda07fd850a
4a793f3cac1e44b6957f7a
STRICT_TRANS_TABLES,STRICT_ALL_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_B
Y_ZERO,TRADITIONA

2017-04-19 18:36:47.124 28626 DEBUG nova.conductor.manager [req-b298ca23-c541-407a-83c4-5eda07fd850a
4a793f3cac1e44b6957f7a91f
[BlockDeviceMapping(boot_index=0,connection_info=None,created_at=<?>,delete_on_termination=False,deleted=<?>,deleted_at=<?>
,destinati
_create_block_device_mapping /usr/lib/python2.7/dist-packages/nova/conductor/manager.py:792

2017-04-19 18:36:47.146 28626 DEBUG oslo_db.sqlalchemy.engines [req-b298ca23-c541-407a-83c4-5eda07fd850a
4a793f3cac1e44b6957f7a
STRICT_TRANS_TABLES,STRICT_ALL_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_B
Y_ZERO,TRADITIONA

2017-04-19 18:36:47.215 28626 DEBUG oslo_messaging._drivers.amqpdriver [req-b298ca23-c541-407a-83c4-5eda07fd850a
4a793f3cac1e44b


==> nova-placement-api.log <==

2017-04-19 18:36:47.719 32000 DEBUG nova.api.openstack.placement.requestlog [req-7204d480-68b3-423f-86d8-
40fc2c2849ed a563582e120
/usr/lib/python2.7/distpackages/nova/api/openstack/placement/requestlog.py:38

2017-04-19 18:36:47.728 32000 INFO nova.api.openstack.placement.requestlog [req-7204d480-68b3-423f-86d8-40fc2c2849ed
a563582e120d4

2017-04-19 18:36:47.737 31999 DEBUG nova.api.openstack.placement.requestlog [req-6d63148b-8998-4ceb-9a03-
9cf35ddf1e34 a563582e120
/usr/lib/python2.7/dist-packages/nova/api/openstack/placement/requestlog.py:38

2017-04-19 18:36:47.761 31999 DEBUG nova.api.openstack.placement.handlers.allocation [req-6d63148b-8998-4ceb-9a03-
9cf35ddf1e34 a563 AllocationList[Allocation(consumer_id=79a5e490-ca87-4c22-8d08-
e8e588a4aee7,id=<?>,resource_class='MEMORY_MB',resource_provider=R
/usr/lib/python2.7/dist-packages/nova/api/openstack/placement/handlers/allocation.py:255
```

Capture 3.14: displays a successful boot sequence with Placement API and Nova Scheduler

3.1.10 Metadata Service

OpenStack provides a metadata service for cloud instances. This metadata is useful for accessing instance-specific information from within the instance (Figure 3.4). The primary purpose of this capability is to apply customizations to the instance during boot time if cloud-init or cloudbase-init is configured on your Linux or Windows image respectively. However, instance metadata can be accessed at any time after the instance boots by the user or by applications running on the instance. The metadata service is implemented by the nova-api service on each controller node:

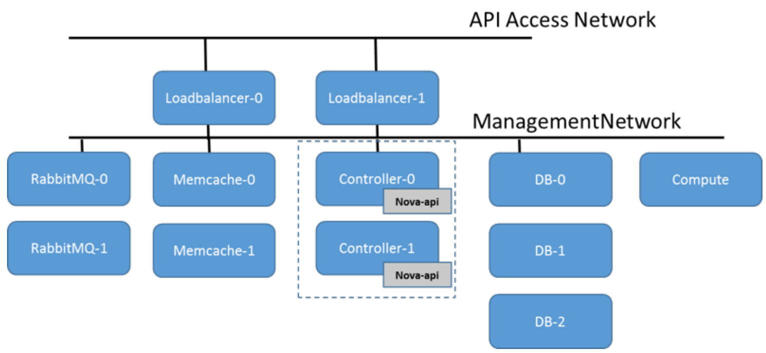


Figure 3.4: OpenStack metadata service for cloud instances

Instances can access the metadata via HTTP at the link-local IP address of 169.254.169.254. To manually access the metadata, use curl or wget to make a request and see the content that is available:

```
#curl http://169.254.169.254/openstack/latest
meta_data.json
user_data
password
vendor_data.json
```

Capture 3.15: use curl or wget to make a request and see the content

3.1.10.1 VIO Deployment Details

In VMware Integrated OpenStack, we deploy two NSX Edge Routers, known as metadata proxy routers, in a load-balanced configuration, to serve as a proxy for metadata service requests:

edge-134	metadata_proxy_router-8824f1a1-68f1-4015-aaa0-d2cd28b33d33	NSX Edge
edge-135	metadata_proxy_router-df5296e9-90d9-44aa-972d-45db20d0b7ff	NSX Edge

Capture 3.16: Proxy for metadata service requests

Each metadata proxy router is visible to administrative users within OpenStack:

	Name	Status	Distributed	Router Type	External Network
☐	metadata_proxy_router	Active	No	Exclusive	-
☐	metadata_proxy_router	Active	No	Exclusive	-

Capture 3.16: metadata proxy router

Within OpenStack, an inter-edge network with subnet 169.254.128.0/17 is created and each metadata proxy router is connected to this network:

Networks

Project	Network Name	Subnets Associated
-	inter-edge-net	**inter-edge-subnet** 169.254.128.0/17

Capture 3.18: metadata proxy router

There are two ways metadata information can be fetched:

If tenant router does not exist:
VM > DHCP Edge > Metadata Proxy Edge > Management Network > Nova If tenant router exists:
VM > Edge Tenant Router > Metadata Proxy Edge > Management Network > Nova

Some details on this workflow (Figure 3.5):

1) A VM sends HTTP request to 169.259.169.254
2) The request is forwarded over the inter-edge-net network to metadata proxy edge
    a. The request is then sent to the Nova metadata service across the Haproxy. Metadata service provides following treatments:
    b. If instance information is already cached. Fetch and return cached data.
3) New instances, query new query to the MySQL DB
4) Once the desired metadata is fetched from DB, return the metadata to the instance that requested it.

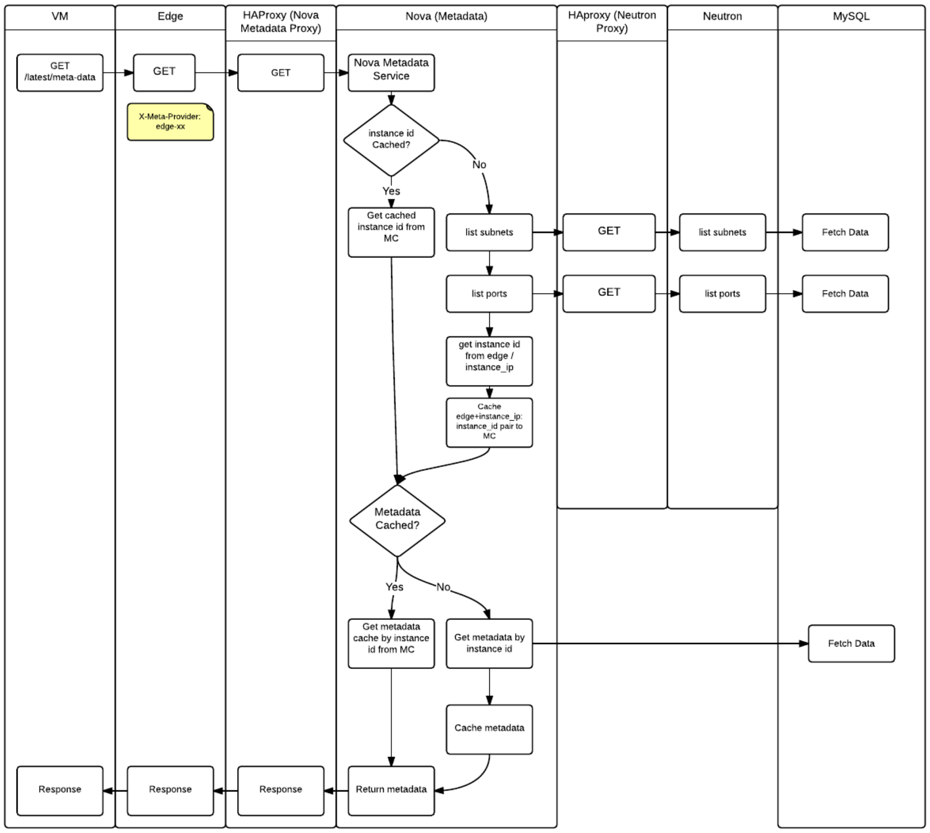


Figure 3.5: OpenStack metadata service for cloud instances

3.1.11 How to Place VM Instances into a Specific Nova Aggregate

The following example demonstrates the VM placement concept:

- Customer has 2 vSphere clusters. One cluster is for production, and another for non-production tasks. Each cluster maps to a nova compute in OpenStack.
- Customer follows a strict CICD model that moves from the non-production cluster to the production cluster. Development workload must deploy to the non-production cluster.
- Production workloads must deploy to the production cluster.

This example below uses a combination of nova flavor extra spec and host aggregate to solve this. In the OpenStack Scheduler, AggregateInstanceExtraSpecsFilter, AggregateCoreFilter, AggregateRamFilter are required. For more information, see the Nova Scheduler Interaction and Over Subscription sections.   New OpenStack CLI command line syntax will be used.

1. Create the respective aggregates. use the "openstack aggregate create" command to create three zones, Prod, Dev and Test zone. Legacy "nova aggregate-create" nova command works just as well.

```
viouser@prme-haas-2-vio:~$ openstack aggregate create --zone nova Prod
+-------------------+----------------------------+
| Field             | Value                      |
+-------------------+----------------------------+
| availability_zone | nova                       |
| created_at        | 2017-04-20T16:44:33.888528 |
| deleted           | False                      |
| deleted_at        | None                       |
| id                | 22                         |
| name              | Prod                       |
| updated_at        | None                       |
+-------------------+----------------------------+
viouser@prme-haas-2-vio:~$ openstack aggregate create --zone nova Dev
+-------------------+----------------------------+
| Field             | Value                      |
+-------------------+----------------------------+
| availability_zone | nova                       |
| created_at        | 2017-04-20T16:44:40.251135 |
| deleted           | False                      |
| deleted_at        | None                       |
| id                | 25                         |
| name              | Dev                        |
| updated_at        | None                       |
+-------------------+----------------------------+
viouser@prme-haas-2-vio:~$ openstack aggregate create --zone nova Test
+-------------------+----------------------------+
| Field             | Value                      |
+-------------------+----------------------------+
| availability_zone | nova                       |
| created_at        | 2017-04-20T16:44:44.203817 |
| deleted           | False                      |
| deleted_at        | None                       |
| id                | 28                         |
| name              | Test                       |
| updated_at        | None                       |
+-------------------+----------------------------+
viouser@prme-haas-2-vio:~$
```

Capture 3.19: Create three zones: Prod, Dev and Test zone

2. Add Nova computes to the aggregate. use the "openstack aggregate add host" command to add compute hosts to respective compute aggregates. Legacy version of the nova command is "nova aggregate-add-host".

```
viouser@prme-haas-2-vio:~$ openstack aggregate add host Prod compute01
+-------------------+--------------------------------+
| Field             | Value                          |
+-------------------+--------------------------------+
| availability_zone | nova                           |
| created_at        | 2017-04-20T16:44:33.000000     |
| deleted           | False                          |
| deleted_at        | None                           |
| hosts             | [u'compute01']                 |
| id                | 22                             |
| metadata          | {u'availability_zone': u'nova'} |
| name              | Prod                           |
| updated_at        | None                           |
+-------------------+--------------------------------+
viouser@prme-haas-2-vio:~$ openstack aggregate add host Dev compute02
+-------------------+--------------------------------+
| Field             | Value                          |
+-------------------+--------------------------------+
| availability_zone | nova                           |
| created_at        | 2017-04-20T16:44:40.000000     |
| deleted           | False                          |
| deleted_at        | None                           |
| hosts             | [u'compute02']                 |
| id                | 25                             |
| metadata          | {u'availability_zone': u'nova'} |
| name              | Dev                            |
| updated_at        | None                           |
+-------------------+--------------------------------+
viouser@prme-haas-2-vio:~$
```

Capture 3.20: Openstack aggregate add host

3. Map the corresponding metadata to aggregate. Metadata data setting will be used during scheduling to find the compute hardware that matches requested flavor. Legacy version of the command is "nova aggregate-set-metadata"

```
viouser@prme-haas-2-vio:~$ openstack aggregate set --property prod=true Prod
viouser@prme-haas-2-vio:~$ openstack aggregate set --property dev=true Dev
viouser@prme-haas-2-vio:~$ openstack aggregate show Prod
+-------------------+----------------------------+
| Field             | Value                      |
+-------------------+----------------------------+
| availability_zone | nova                       |
| created_at        | 2017-04-20T16:44:33.000000 |
| deleted           | False                      |
| deleted_at        | None                       |
| hosts             | [u'compute01']             |
| id                | 22                         |
| name              | Prod                       |
| properties        | prod='true'                |
| updated_at        | None                       |
+-------------------+----------------------------+
viouser@prme-haas-2-vio:~$ openstack aggregate show Dev
+-------------------+----------------------------+
| Field             | Value                      |
+-------------------+----------------------------+
| availability_zone | nova                       |
| created_at        | 2017-04-20T16:44:40.000000 |
| deleted           | False                      |
| deleted_at        | None                       |
| hosts             | [u'compute02']             |
| id                | 25                         |
| name              | Dev                        |
| properties        | dev='true'                 |
| updated_at        | None                       |
+-------------------+----------------------------+
viouser@prme-haas-2-vio:~$
```

Capture 3.21: Map the corresponding metadata to aggregate

4. Create the Nova flavor. Both Prod and Dev web servers will require 4G Memory, 2 vCPU and 40 GB disk. Legacy nova command is "nova flavor-create"

```
viouser@prme-haas-2-vio:~$ openstack flavor create --ephemeral 40 --ram 4096 --vcpus 2 Prod.Web
+----------------------------+--------------------------------------+
| Field                      | Value                                |
+----------------------------+--------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                |
| OS-FLV-EXT-DATA:ephemeral  | 40                                   |
| disk                       | 0                                    |
| id                         | fadf6312-2075-4e1d-a88a-c2a6bf7e5cc6 |
| name                       | Prod.Web                             |
| os-flavor-access:is_public | True                                 |
| ram                        | 4096                                 |
| rxtx_factor                | 1.0                                  |
| swap                       |                                      |
| vcpus                      | 2                                    |
+----------------------------+--------------------------------------+
viouser@prme-haas-2-vio:~$ openstack flavor create --ephemeral 40 --ram 4096 --vcpus 2 Dev.Web
+----------------------------+--------------------------------------+
| Field                      | Value                                |
+----------------------------+--------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                |
| OS-FLV-EXT-DATA:ephemeral  | 40                                   |
| disk                       | 0                                    |
| id                         | 8990517e-d06e-40c7-b0f1-0ec9a44302b3 |
| name                       | Dev.Web                              |
| os-flavor-access:is_public | True                                 |
| ram                        | 4096                                 |
| rxtx_factor                | 1.0                                  |
| swap                       |                                      |
| vcpus                      | 2                                    |
+----------------------------+--------------------------------------+
viouser@prme-haas-2-vio:~$
```

Capture 3.22: Create the Nova flavor

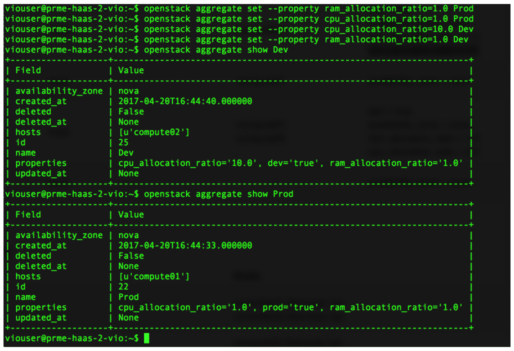
5. Add extra specs data to new flavors. Extra spec should match the value set for the compute aggregate.

```
viouser@prme-haas-2-vio:~$ openstack flavor set --property dev=true Dev.Web
viouser@prme-haas-2-vio:~$ openstack flavor set --property prod=true Prod.Web
viouser@prme-haas-2-vio:~$ openstack flavor show Dev.Web
+----------------------------+--------------------------------------+
| Field                      | Value                                |
+----------------------------+--------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                |
| OS-FLV-EXT-DATA:ephemeral  | 40                                   |
| disk                       | 0                                    |
| id                         | 8990517e-d06e-40c7-b0f1-0ec9a44302b3 |
| name                       | Dev.Web                              |
| os-flavor-access:is_public | True                                 |
| properties                 | dev='true'                           |
| ram                        | 4096                                 |
| rxtx_factor                | 1.0                                  |
| swap                       |                                      |
| vcpus                      | 2                                    |
+----------------------------+--------------------------------------+
viouser@prme-haas-2-vio:~$ openstack flavor show Prod.Web
+----------------------------+--------------------------------------+
| Field                      | Value                                |
+----------------------------+--------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                |
| OS-FLV-EXT-DATA:ephemeral  | 40                                   |
| disk                       | 0                                    |
| id                         | fadf6312-2075-4e1d-a88a-c2a6bf7e5cc6 |
| name                       | Prod.Web                             |
| os-flavor-access:is_public | True                                 |
| properties                 | prod='true'                          |
| ram                        | 4096                                 |
| rxtx_factor                | 1.0                                  |
| swap                       |                                      |
| vcpus                      | 2                                    |
+----------------------------+--------------------------------------+
viouser@prme-haas-2-vio:~$
```

Capture 3.23: Add extra specs data to new flavors

6. (Optional) Set the over-subscription ratio on the individual aggregate. Oversubscription ratio at the aggregate level is Optional. Will default to global oversubscription if not set at the aggregate level. Global allocation ratios are:

   - CPU: 10:1
   - Memory: 1.5:1
   - Disk: 1.0



Capture 3.24: Set the over-subscription ratio

3.1.12 Dedicating Compute Aggregates or Hosts by Tenants

The following example demonstrates the concept of dedicating compute host based on OpenStack tenant:

- Customer has 2 vSphere clusters. One cluster is for Coke, and another for Pepsi. Each cluster maps to a nova compute in OpenStack. Coke can only boot VM on the Coke cluster.
- Pepsi can only boot VM on the Pepsi cluster.
- Anytime a new Coke or Pepsi department is added, the newly created OpenStack project can only boot VMs on the cluster dedicated to each company.

The example below is nearly identical to the previous example, with follow minor differences:

- nova scheduler configuration in /etc/nova/nova.conf includes AggregateMultiTenancyIsolation instead of AggregateInstanceExtraSpecsFilter, AggregateCoreFilter, AggregateRamFilter filters.
- Instead prod=true, or dev=true, set "fiter_tenant_id" metadata tag on the host aggregate with the id of Coke and Pepsi's OpenStack tenant.

Summary Commands are:

1. Create the respective aggregates. use the "openstack aggregate create" command to create two aggregates:

   a. **openstack aggregate create --zone nova Cokeopenstack aggregate create --zone nova Pepsi**

2. Use the "openstack aggregate add host" command to add compute hosts to respective compute aggregates.

   a. **openstack aggregate add host Coke compute01 openstack aggregate add host Pepsi compute02**

3. Use the "openstack aggregate set" command to set the fiter_tenant_id of the aggregate

```
openstack aggregate set --property fiter_tenant_id=<UUID Coke> Coke
openstack aggregate set --property fiter_tenant_id=<UUID Pepsi> Pepsi
```

Finally update the /etc/nova/nova.conf

```
scheduler_default_filters =RetryFilter,AvailabilityZoneFilter,RamFilter,DiskFilter,ComputeFilter,ComputeCapabilitiesFilter,ImagePropertiesFilter,ServerGroupAntiAffinityFilter,ServerGroupAffinityFilter, AggregateMultiTenancyIsolation
```

The ability to exclude a tenant from an aggregate does not currently exist in OpenStack. If a third host aggregate exists, which does not have any filter_tenant_id, then it is possible that both Coke and Pepsi will launch VM instances on the third cluster.

3.2 VIO Nova Compute Scaling

As workloads increase, your Cluster must scale to meet new capacity demands. While vCenter Server can have a maximum cluster size of 64 ESXi hosts, VIO Cluster scaling can be different depending on use case. You can add new capacity to a VIO deployment in two ways:

- Vertical scaling - Increase number of hosts in a Cluster.
- Horizontal scaling - Deploy a new vCenter Server Cluster and add the Cluster as a new nova-compute node to an existing or new nova compute aggregate.

The decision to implement vertical or horizontal scaling should be based on the VIO use case and the number of concurrent operations against the OpenStack API. The VIO Use Cases section outlines the following four most frequent scenarios:

- Traditional enterprise model
- API access to the infrastructure
- NFV
- Thin IaaS

The following are general guidelines for scaling models you can use with the scenarios:

Consumption Model	Expected Parallel OpenStack Operations	Scaling Model
Traditional Enterprise	Low	Vertical or Horizontal
API access to the Infrastructure	Low	Vertical or Horizontal
API access to the Infrastructure	High	Horizontal
NFV	Low	Vertical or Horizontal
Thin IaaS	Low	Vertical or Horizontal

Table 3.1: Nova Compute Scaling Model

We recommend all VIO deployments to leverage vRealize Operations Manager and vRealize Log Insight for capacity trend analysis as well as planning. When in doubt, consult with the VMware Professional Services Organization (PSO) to decide the right expansion strategy for your deployment.

For more information about how to integrate vRealize Operations Manager and vRealize Log Insight into your VIO deployment, see the Operational Monitoring and Logging section of this design guide.

3.2.1 Add a Host to an Existing Cluster

This is a vSphere operation. From your vCenter:

1. Add a host to vCenter as a standalone host

2. Add the host to Distributed Virtual Switch (VDS)
    a. Home -> Networking
    b. Select and right-click on the VDS -> Add and Manage Hosts
3. Add the host to a Cluster
    a. Home -> Hosts and Clusters
    b. Select and right-click on the Cluster -> Add Hosts

The required NSX software is automatically installed on the newly added host.

No additional work is required on VIO. Based on Cluster DRS settings, running workloads will be automatically migrated to the newly added ESXi host. Cluster capability will be updated in VIO automatically as part of periodic resource synchronization.

3.2.2 Create a New Compute Cluster

You can increase the number of compute Clusters in your VMware Integrated OpenStack deployment to increase CPU capacity. There is a 1:1 relationship between Nova Compute nodes on the VIO Management Plane and the vSphere Clusters you want to expose to VIO. Nova will think that each vSphere Cluster you expose as an OpenStack hypervisor: this simplification on the Nova side allows us to leverage all the features a vSphere Cluster has (HA, DRS, etc.). Adding a new Cluster to VIO is accomplished using the VIO plugin UI:

- From vCenter Home, click VMware Integrated OpenStack > Getting Started.
- Under Manage, click Nova Compute, and click Add to add Clusters to OpenStack.
- In the Add Clusters to Openstack wizard, select Cluster and storage that you want to add.

Note: The nova process will restart when creating additional compute Clusters.

Newly added Cluster will show up as a new OpenStack Nova Compute node. Cloud admins have the option of adding the new vSphere Cluster to an existing aggregate or creating a new aggregate. Host aggregates as stated in earlier sections can be regarded as a mechanism to further partition an availability zone. Each Nova compute aggregate can be assigned different key-value pairs to enable advanced VM scheduling placement. As a general guideline, it is recommended to keep number of host aggregates static. To simplify operational support, consider increase number of host aggregates only if

- Different classes of hardware are made available
- Change in Oversubscription ratio
- Change in Virtual Machine Placement requirement (Example: Production vs. Dev)

Below is an example of adding vSphere Cluster to an existing host aggregate:

```
root@prme-haas-2-vio:/home/viouser# nova aggregate-details 7
+----+------+-------------------+-------------+--------------------------------------+
| Id | Name | Availability Zone | Hosts       | Metadata                             |
+----+------+-------------------+-------------+--------------------------------------+
| 7  | ssd  | nova              | 'compute01' | 'availability_zone=nova', 'ssd=true' |
+----+------+-------------------+-------------+--------------------------------------+
root@prme-haas-2-vio:/home/viouser# nova aggregate-add-host  7 compute02
Host compute02 has been successfully added for aggregate 7
+----+------+-------------------+--------------------------+--------------------------------------+
| Id | Name | Availability Zone | Hosts                    | Metadata                             |
+----+------+-------------------+--------------------------+--------------------------------------+
| 7  | ssd  | nova              | 'compute01', 'compute02' | 'availability_zone=nova', 'ssd=true' |
+----+------+-------------------+--------------------------+--------------------------------------+
root@prme-haas-2-vio:/home/viouser#
```

Capture 3.25: Set the over-subscription ratio

Below is an example of creating new host aggregate and adding vSphere Cluster to the new host aggregate:

```
root@prme-haas-2-vio:/home/viouser# nova aggregate-create new-cluster nova
+----+-------------+-------------------+-------+--------------------------+
| Id | Name        | Availability Zone | Hosts | Metadata                 |
+----+-------------+-------------------+-------+--------------------------+
| 19 | new-cluster | nova              |       | 'availability_zone=nova' |
+----+-------------+-------------------+-------+--------------------------+
root@prme-haas-2-vio:/home/viouser# nova aggregate-set-metadata 19 ssd=true
Metadata has been successfully updated for aggregate 19.
+----+-------------+-------------------+-------+--------------------------------------+
| Id | Name        | Availability Zone | Hosts | Metadata                             |
+----+-------------+-------------------+-------+--------------------------------------+
| 19 | new-cluster | nova              |       | 'availability_zone=nova', 'ssd=true' |
+----+-------------+-------------------+-------+--------------------------------------+
root@prme-haas-2-vio:/home/viouser# nova aggregate-add-host  19 compute02
Host compute02 has been successfully added for aggregate 19
+----+-------------+-------------------+-------------+--------------------------------------+
| Id | Name        | Availability Zone | Hosts       | Metadata                             |
+----+-------------+-------------------+-------------+--------------------------------------+
| 19 | new-cluster | nova              | 'compute02' | 'availability_zone=nova', 'ssd=true' |
+----+-------------+-------------------+-------------+--------------------------------------+
root@prme-haas-2-vio:/home/viouser#
```

Capture 3.26: Adding vSphere Cluster to the new host aggregate

When you either add new vSphere Cluster to an existing or a new Nova host aggregate, it's important to point out:

- Nova scheduler is responsible to place new workload requests to the newly added resource. If an existing host aggregates is used, there will be no change to the nova scheduler filter. Nova scheduler filer updates maybe required if new host aggregates are deployed. Refer to the Nova Scheduler Interaction and Over Subscription section for required filter updates.
- Existing workloads do not automatically migrate to the new vSphere Cluster. Cloud Admin can leverage the OpenStack live-migrationco mmand to balance workloads between existing and new vSphere Clusters.

3.3 Brownfield Migration Options

Brownfield Migration, in the context of VIO, is the ability to import vSphere workloads (VMs) or VM templates from vCenter managed to OpenStack managed. The current implementation of VIO only supports import from vCenter server controlled by VIO, VM belonging to vCenter Server not managed by VIO can not be imported. By importing vSphere VM workloads or Glance images into OpenStack and running critical Day 2 operations against them by using OpenStack APIs, you can quickly move existing development projects or production workloads to the OpenStack Framework.

3.3.1 Add VM Templates as Glance Images to VMware Integrated OpenStack Deployment

You can add existing VM templates to your VMware Integrated OpenStack deployment as Glance images. You can use this capability to boot instances, create bootable block storage volumes, and perform other functions available to Glance images.

Prerequisites

- Single disk only
- No CD-ROM or floppy disk Drive
- VM template resides on the same vCenter Server as your VMware Integrated OpenStack deployment

Procedure

1. To create a Glance image, run the following command on the OMS server:

```
glance image-create --name <Name>  --disk-format vmdk --container-format bare \
  --property vmware_ostype=ubuntu64Guest --property hw_vif_model=VirtualVmxnet3
```

Capture 3.27: Create a Glance Image

2. Note the UUID of the newly created image. The UUID is required as input for below step

glance location-add <Glance Image UUID> --url "vi://<vcenter-host>/<datacenter-path>/vm/<sub-folders>/<template_name>"

The location-add command points to the inventory path for the VM template and can refer to either the VM or the host. For example:

- *vi://<datacenter-path>/vm/<template_name> vi://<datacenter-*
- *path>/host/<host_name>/<template_name>*

The VM and host keywords in the inventory path represent the VM and Templates view and the Host and Cluster view hierarchy in your vSphere Web Client.

Capture 3.28: View UUID of newly created image

3.3.2 Import VMs from vSphere

You can import VMs from vSphere into your VMware Integrated OpenStack deployment and manage them like OpenStack instances.

You import VMs using the Datacenter Command Line Interface (DCLI), which is packaged with the VMware Integrated OpenStack management server, and is powered by the VMware Integrated OpenStack vAPI provider.

Though imported VMs become OpenStack instances, they remain distinct in several ways:

1. If there are multiple disks on the imported VM, Nova snapshot creation is not supported.
2. If there are multiple disks on the imported VM, the Nova resizing operation is not supported.
3. Existing networks are imported as provider network type port group, with subnets created with DHCP disabled. This prevents conflict between the DHCP node in OpenStack and the external DHCP server.
4. The flavor for the imported VM shows the correct CPU and memory, but the root disk. Root disk size incorrectly displays as having 0 GB.

Procedure

1. Add the clusters containing the VMs to be imported to the VMware Integrated OpenStack deployment.
   a. In the vSphere Web Client, identify the cluster containing the VMs to be imported.
   b. Add the cluster to the VMware Integrated OpenStack deployment as a Nova compute cluster.
   c. Repeat for multiple clusters, if necessary.
2. Using SSH, log in to the VMware Integrated OpenStack manager, and connect to the VMware Integrated OpenStack vAPI endpoint.

```
dcli +server http://localhost:9449/api +i
```

1. List all namespaces in the VMware Integrated OpenStack vAPI provider

```
dcli> com vmware vio
The vio namespace provides namespaces to manage components related to OpenStack and vSphere
Available Namespaces:
vm
```

2. List all unmanaged VMs in a specific target cluster that you added to the Nova Compute node.

```
com vmware vio vm unmanaged list --cluster <vcenter cluster mor-id>
```

3. Import VMs into VMware Integrated OpenStack. To import a specific VM, run the following command:

```
com vmware vio vm unmanaged importvm [-h] \ --vm VM [--tenant TENANT] [--nic-mac-address NIC_MAC_ADDRESS] \ [--nic-ipv4-address NIC_IPV4_ADDRESS]
```

3.4 VIO: VM Boot and Destruction Workflow

3.4.1 VM Boot Workflow

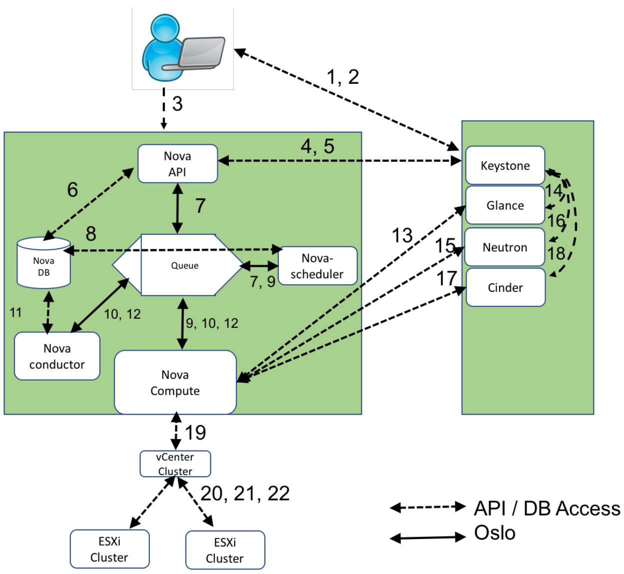


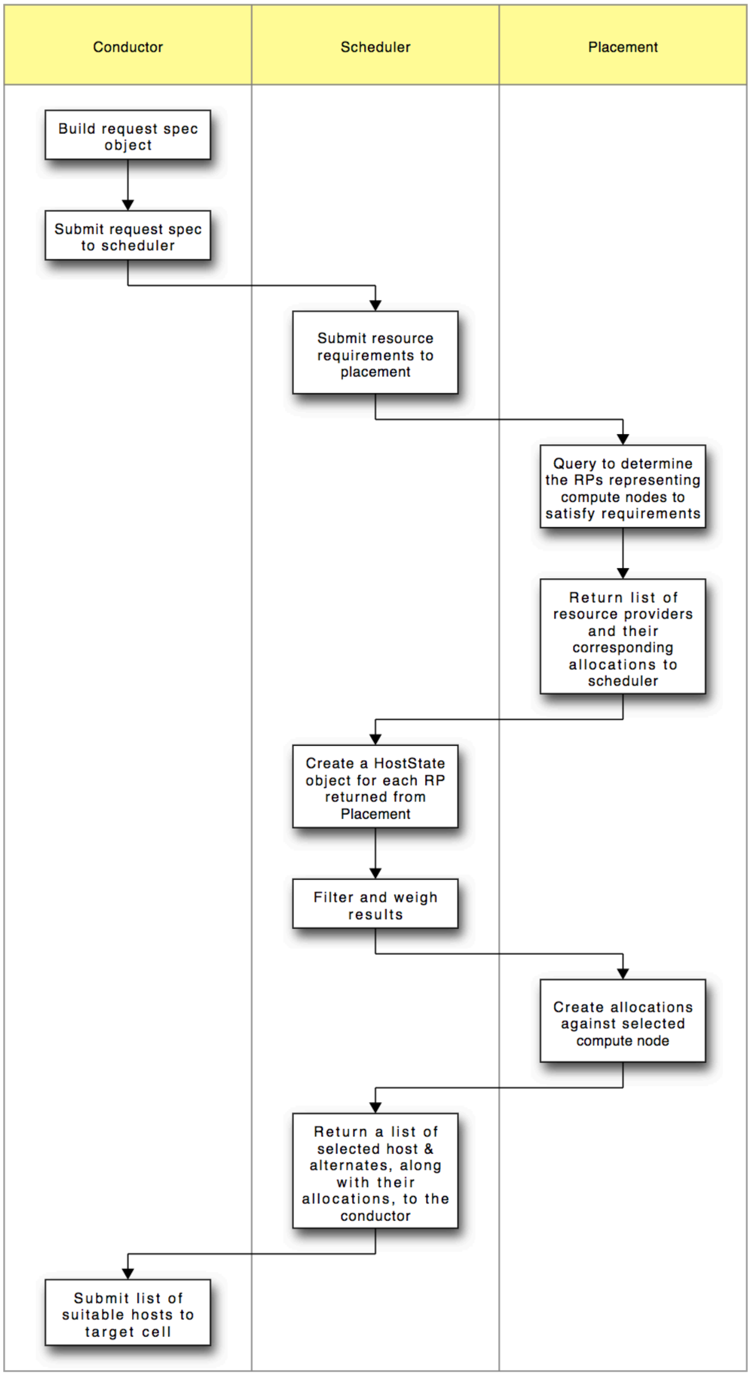
Figure 3.6 VM Boot Workflow

The following steps show the VM Boot Workflow (Figure 3.6).

1. An end user issues a REST API call to Keystone for user authentication and auth-token access.
2. Keystone authenticates the user and returns an auth-token, which is used for sending request to other components through REST-call.
3. The user sends a request to nova-api for booting a VM using REST API along with auth-token obtained from step #2.
4. nova-api validates auth-token and access permissions with keystone.
5. Keystone validates the auth-token and passes back permission information to nova-api.
6. nova-api creates an initial database entry for the new VM instance.
7. nova-api sends the request to nova-scheduler

8. nova-scheduler queries nova-database to find an appropriate host by filtering and weighing based on requested Nova flavor size.
9. nova-scheduler sends request to nova-compute for launching the instance on an appropriate host .
10. nova-compute sends request to nova-conductor to fetch the instance information from database.
11. nova-conductor interacts with nova-database and returns the instance information.
12. nova-conductor updates nova-compute with instance information.
13. nova-compute, through REST call, passes auth-token to glance-api to get the Image URI by Image ID from glance and upload image from image storage.
14. glance-api validates the auth-token with keystone. nova-compute gets the image metadata.
15. nova-compute, through REST call, passes auth-token to neutron-server API to allocate and configure the network such that instance gets the IP address and associated neutron port.
16. Neutron-server validates the auth-token with keystone and returns the newly allocated network information to nova-compute.
17. nova-compute, through REST call, passes auth-token to cinder-volume API to attach volumes to instance.
18. cinder-api validates the auth-token with keystone and returns the block storage information to nova-compute.
19. nova-compute generates VM boot data for the hypervisor driver and executes VM boot request against the vCenter Server.
20. The vCenter Server receives the VM boot request. If glance image with corresponding flavor is requested for the very first time, vCenter creates a local copy of the VMDK from glance in the nova datastore in the form of a shadow VM based on requested flavor size. This shadow VM does not consume CPU cycles, but counts towards vSphere license usage. vCenter Server use the shadow VM to create a linked clone of subsequent requests for the VM flavor and image combination. The total number of shadow VMs in a VIO deployment can be determined by the following formula:
    - Number of flavor * Number of glance image * Number of Nova datastore
21. The vCenter Server creates a linked clone of the VM based on DRS reservation setting and other information contained in the nova flavor extra-spec.
22. The vCenter Server renames and powers on the new VM. An instance is deleted if it cannot be powered on.
23. nova-api updates project quota based instance RAM and core count.

Nova Scheduler has gone through many changes since Mitaka release. Below is an overview of how scheduling works in nova from Pike onwards.


Conductor
Scheduler
Placement
Build request spec object
Submit request spec to scheduler
Submit resource requirements to placement
Query to determine the RPs representing compute nodes to satisfy requirements
Return list of resource providers and their corresponding allocations to scheduler
Create a HostState object for each RP returned from Placement
Filter and weigh results
Create allocations against selected compute node
Return a list of selected host & alternates, along with their allocations, to the conductor
Submit list of suitable hosts to target cell

Detailed sequence of steps are explained here

3.4.2 VM Termination Workflow

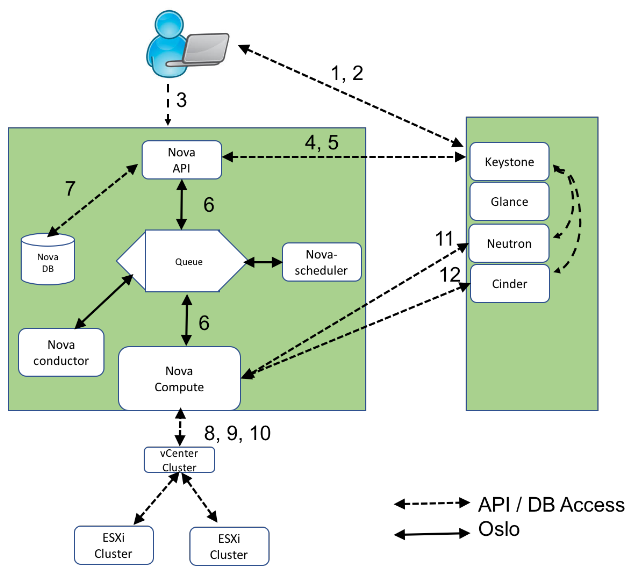


Figure 3.7 VM Termination Workflow

The following steps show the VM termination workflow (Figure 3.7)

1. An end user issues a REST API call to Keystone for user authentication and auth-token.
2. Keystone generates & sends back auth-token (used for sending request to other Components through REST-call).
3. The user sends through REST API along with auth token to nova-api to delete an VM.
4. nova-api validates auth-token and access permission with keystone.
5. Keystone validates the token and passes the permission information back to nova-api.
6. nova-api processes the delete request by calling nova-compute to fetch and terminate the instance by UUID.
7. nova-api updates the project quota based deleted instance RAM and core count.
8. nova-compute calls the hypervisor driver and runs VirtualMachine.PowerOffVM_Task against the vCenter Server.
9. After the VM is powered off, nova-compute issues VirtualMachine.Destroy_Task against the vCenter Server to delete the VM.
10. vCenter delete VM from the vSphere datastore.
11. nova-compute, through REST, calls the Neutron API to delete network by using instance _deallocate_network. Neutron-server deletes the port assigned to the VM.
12. nova-compute, through REST, calls cinder-volume API to terminate bdm BlockDeviceMapping. Cinder-volume deletes the block storage mapping information.

Section 4: VIO Networking Configuration

This section provides a deeper look into the network infrastructure best practices for designing and delivering VIO.

VMware Integrated OpenStack has a number of networking requirements that must be satisfactorily met for proper deployment and operation of the solution. While NSX is not strictly required in VIO, these connectivity prerequisites are aligned with the best practices for integrating NSX, as this is the recommended deployment model for production environments. It is important to remember that NSX enables multicast free, controller-based overlay networks, as well as VLAN-backed networks for tenants and providers. Its extreme flexibility enables it to work with:

- Any type of physical topology, such as PoD, routed, or spine-leaf
- Any vendor physical switch
- Any install base with any topology, such as L2, L3, or a converged topology mixture
- Any other fabric technology with proprietary controls

The only two strict requirements for NSX from the physical network are IP connectivity and jumbo MTU support. OpenStack defines a number of networks:

- External Network: Data network used by the user environment to gain access to the external world (Internet/Intranet)
  - Can only be configured by the OpenStack Admin
  - Routable IP address space within the customer organization Neutron routers are uplinked to this network
  - Floating IP Pool (DNAT) belongs to this network

- API Network: network segment used by OpenStack consumers, users, and tenants to access their OpenStack Project for provisioning operations
  - Routable IP address space within the customer organization Not recommended to combine API and External networks

- Tenant Network: Data network used by tenants to connect OpenStack instances Can be self-service or administrator-controlled
  - Can be completely isolated
  - Support for overlapping IP addresses as well as non-overlapping IP addresses

The following infrastructure VLANs are required for a successful VIO deployment:

- Management VLAN - Used for the OpenStack Management Server (OMS) and the VIO control plane components. This VLAN is often shared with other management and control plane infrastructure, such as NSX Manager, NSX Controllers, vCenter Server, logging, analytics, and so on. OMS has an adjacency requirement with the VMs that make up the VIO Control plane and must be deployed on the same IP subnet.
- API VLAN - A routable network that is used by the OpenStack tenants and admins to interact with the various API services available in the private cloud.
- External VLAN - A routable network (or networks, as multiple External subnets are supported) that is used to connect the uplinks of the Neutron routers (served in the back-end by NSX Edge Gateways). In NAT topologies, this network (or networks) also hosts the floating IP range, which is a pool of IP addresses dedicated to providing external-to-internal connectivity via destination NAT (DNAT) configured on the Neutron routers.

Note: If routable IP addresses are scarce in the environment and VLAN provisioning is challenging, the External and API networks can be combined. It must be understood that by doing this API and application traffic will share the same infrastructure from a connectivity perspective.

- Transport Network - One or more VLANs that make up the transport infrastructure for encapsulated overlay traffic used by NSX. Additional Provider VLANs - If the deployment calls for the use of VLAN-backed provider networks, these VLANs must be trunked to the compute and Edge clusters as OpenStack has a strict L2-adjacency requirement for DHCP services and tenant instances (no DHCP relay support in Neutron as of the writing of this document).

When using NSX, OpenStack administrators have the added flexibility of enabling overlay technology for External and Provider networks. This allows for a very simple underlay topology and puts the NSX admin in control of the entire baseline infrastructure required by VIO.

The following diagram (Figure 4.1) is a representation of the required VLANs and networks in the VIO Management Cluster:

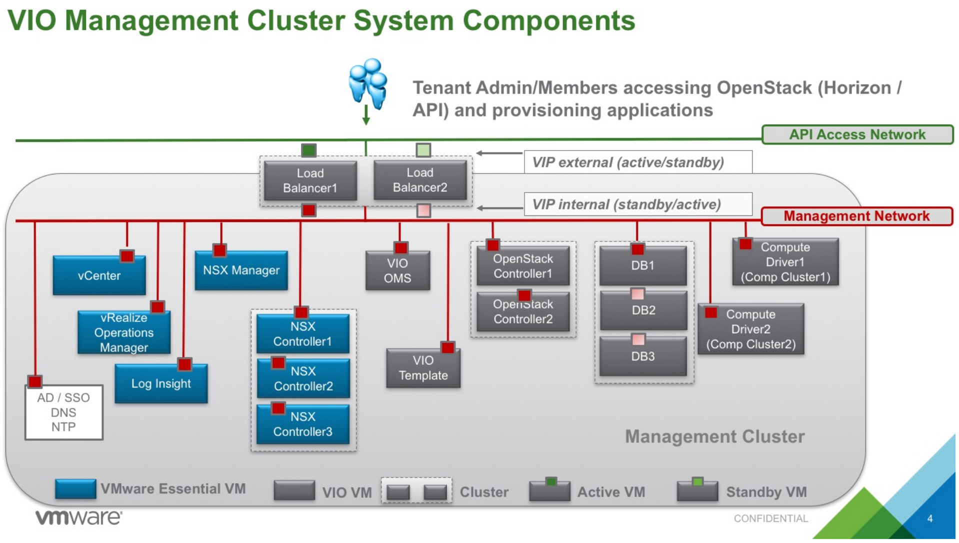


Figure 4.1 VIO Management Cluster System Components

The following diagram (Figure 4.2) is a representation of the complete solution with all the aforementioned functional clusters:

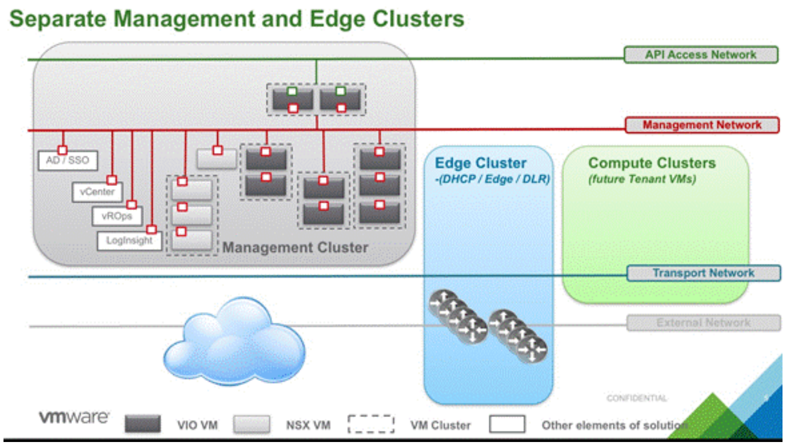


Figure 4.2 Separate Management and Edge Clusters

Note: You can also combine the Management and Edge Clusters. In such cases, all management and control plane components share infrastructure with the OpenStack routers, load balancers, and DHCP servers. In NSX vSphere, these elements correspond to Edge Gateways that are deployed with capacity reservations to ensure performance and scalability even if they are sharing the underlying servers with other Cloud services.

4.1 Neutron Components and NSX Equivalents

Neutron contemplates a number of basic network workflows that are considered "table stakes". These include:

- **L2 services:** The ability for tenants to create and consume their own L2 networks.
- **L3 services:** The ability for tenants to create and consume their own IP subnets and routers. These routers can connect intra-tenant application tiers and can also connect to the external world via NATed and non-NATed topologies.
- **Floating IPs:** A DNAT rule, living on the Neutron router, that maps a routable IP sitting on the external side of that router (External network) to a private IP on the internal side (Tenant network). This floating IP forwards all ports and protocols to the corresponding private IP of the instance (VM) and is typically used in cases where there is IP overlap in tenant space.
- **DHCP Services:** The ability for tenants to create their own DHCP address scopes.
- **Security Groups:** The ability for tenants to create their own firewall policies (L3 or L4) and apply them directly to an instance or a group of instances.
- **Load Balancing as-a-Service, LBaaS:** The ability for tenants to create their own load balancing rules, virtual IPs, and load balancing pools.

The following diagram (Figure 4.3) shows these basic workflows and their situation as it relates to the application, as well as the corresponding NSX element that is leveraged each time. For more information about NSX, see the official VMware NSX product page.

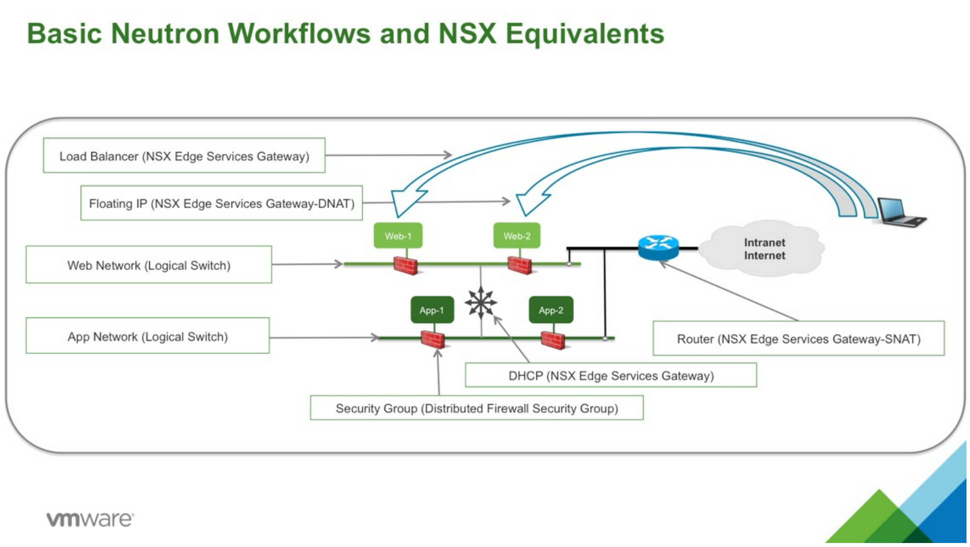


Figure 4.3 Basic Neutron Workflows and NSX Equivalents

By leveraging the NSX Neutron plug-in developed by VMware, cloud administrators can onboard NSX into their OpenStack environment and offer their users and developers the open APIs that they require, without compromising uptime, stability, and scalability.

4.2 VIO Neutron NSX Integration

As OpenStack becomes more prevalent in the industry, cloud architects are looking for ways to provide enterprise-grade network and security services to their consumers without compromising the primary objectives of an OpenStack-based private cloud, which include:

- Vendor-neutral APIs.
- Infrastructure choice and flexibility.
- Amazon-like experience.

Neutron, the networking project in OpenStack, has come a long way in the last few years, adding powerful capabilities at a very fast pace while enabling rich network workflows and a variety of use cases. Long gone are the days of flat Nova networking. There is general consensus in the OpenStack community that a cloud that lacks rich network functionality is a mediocre cloud.

As more features are added to Neutron, its architecture becomes more complex. Wisely, the Neutron community has decided, as of Kilo release, to “decompose” Neutron. The general idea is that Neutron will remain focused on core L2 and L3 services, while all services in the L4-L7 layers will be “pluggable” and abide by a well-known extensible data model. For vendors such as VMware, this is great news. We now have a reference architecture to develop against, promote our value-addition, and expose the unique capabilities of NSX, all the while honoring a consumption model that prioritizes these desirable northbound OpenStack APIs.

With all that said, it is important to note (and know) that Neutron core is developed using a reference implementation based on open source components, including:

- Open vSwitch - Hypervisor-level networking
- dnsmasq - DHCP and DNS services
- Linux iptables - Security groups
- L3-agent - Routing services
- HAProxy - Load balancing

In production, at scale, the reference implementation typically suffers. This is widely recognized as factual and comes at no surprise once you understand how Neutron core is developed and tested. The community is doing a great job at defining the core APIs and the extensibility model, but it’s been up to the vendors, for the most part, to define and test the scalability, reliability, and upgradeability of an OpenStack-based solution using their own “productized” distributions. Often, as it is the case with VMware NSX, vendors replace the reference open-source components with their own technology. OpenStack consumers don’t notice the difference; they still interact with the northbound Neutron APIs. Through plugin and drivers, Neutron API calls are “translated” to private, southbound calls. For NSX, the integration looks like this (Figure 4.4):

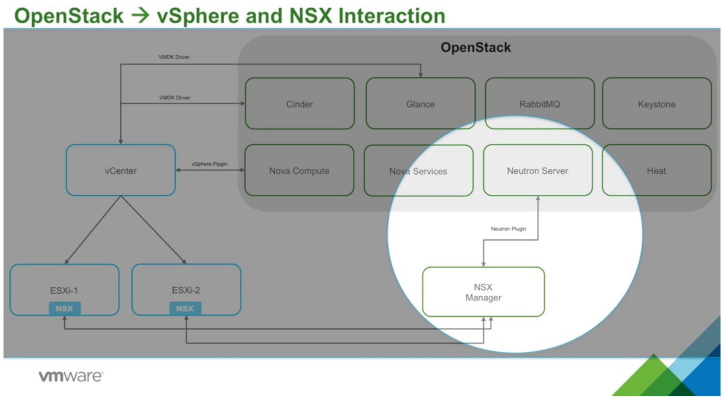


Figure 4.4: vSphere and NSX Interaction

Neutron services leverage a VMware-developed plug-in that instigates API calls to NSX Manager, the API provider and management plane of NSX. This Neutron plug-in is itself an open source project and can be used with any OpenStack implementation (DIY, off-the-shelf, or both). VMware offers its own OpenStack distribution, VMware Integrated OpenStack, which natively integrates the NSX-Neutron plug-in, in addition to other plug-ins and drivers that connect OpenStack compute and storage services to vSphere. As a direct result of leveraging enterprise-grade virtualization with vSphere and enterprise grade networking with NSX, customers will enjoy an enterprise-grade OpenStack layer, thus mitigating the risks and shortcomings of the reference implementation.

4.3 Benefits of NSX

4.3.1 Why Is NSX Essential to OpenStack?

Automation is essential for IT agility and consistency, which in turn significantly improve overall operational savings. However, IT organizations that are still constrained by hardware cannot implement a meaningful automation strategy that meets the often competing goals of the organization. Networking hardware in particular depends heavily on error-prone manual configuration and maintenance of a sprawling library of scripts. The result is a labor-intensive process that impacts an IT department's ability to support the business as it moves quickly to seize emerging opportunities. NSX completely removes this hardware-centric barrier to the automation of networking operations. By moving networking and security services into the data center virtualization layer, NSX delivers the same automated operational model of a VM, but for the entire network. With OpenStack, NSX is able to automate a number of processes, significantly accelerating service delivery and reducing provision times from months to minutes (Figure 4.5). The positive business impacts of this cannot be overstated and include dramatically reduced operational complexity and cost, as well as improved governance, compliance, and consistency.

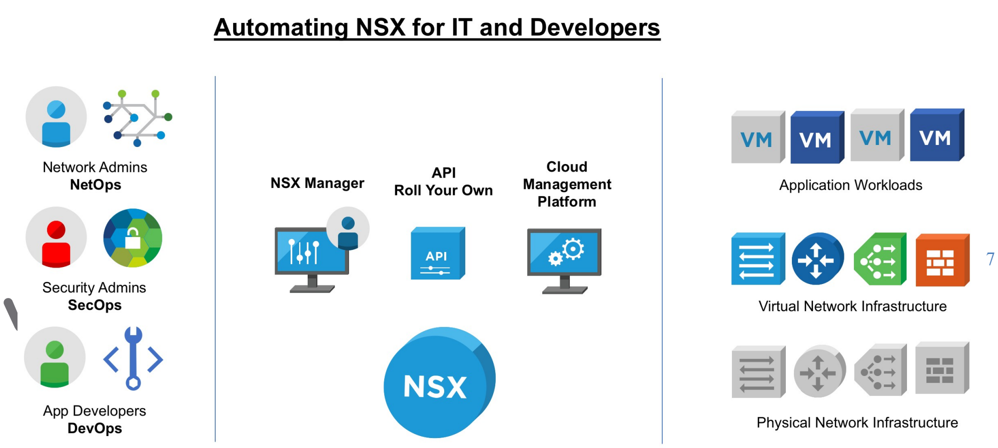


Figure 4.5: Automation from NSX

The benefits of NSX align with the requirements of a robust OpenStack private cloud implementation, which are:

- Agility – Networking at the speed of apps.
- Mobility – Provision anywhere, move anywhere.
- Security – Micro-segment, detect anywhere, detect early
- Multi-tenancy – Share hardware across multiple tenants.
- Simplified operations – Centrally manage, monitor everywhere.

Implementing Neutron is one of the most difficult challenges on an OpenStack Cloud, NSX simplifies such implementation in the following areas:

- Simpler implementation of Neutron Services
- Vendor support for the most critical services in the Cloud
- Higher performance
- Management, Operations, Troubleshooting tools natively on NSX
- Stability at scale and High-Availability

Despite being part of the VMware Integrated Openstack Design Guide, this section can be leverage by any Openstack distribution using NSX. NSX-v, or NSX-T Plugins are available as opensource at VMware Github page that can be integrated in any Openstack distribution.

4.3.2 VMware NSX Network Virtualization and Security Platform

You can use NSX for creation of entire networks in software and embeds them in the hypervisor layer, abstracted from the underlying physical hardware. All network components can be provisioned in minutes, without the need to modify the application.

4.4 NSX Use Cases

Automation

NSX treats your physical network as a pool of transport capacity, with network and security services attached to workloads using a policy-driven approach. This automates networking operations and eliminates bottlenecks associated with hardware-based networks.

Security

NSX embeds security functions right into the hypervisor. It delivers micro-segmentation and granular security to the individual workload, enabling a more secure data center. Security policies travel with the workloads, independent of where workloads are in the network topology.

Application Continuity

NSX abstracts networking from the underlying hardware and attaches networking and security policies to their associated workloads. Applications and data can reside and be accessible anywhere.

Compliance

NSX enables micro-segmentation and granular security of workloads in virtualized networks, isolating sensitive systems and reducing both risk and scope of compliance. Use NSX to help ensure and demonstrate compliant operations with many regulations such as PCI DSS, HIPAA, FedRAMP, SOC, CJIS, DISA STIG, and more.

4.5 NSX-v and NSX-T Feature Comparison

There are two main NSX platforms an OpenStack Admin can select from, NSX vSphere (NSX-v) or NSX Transformers (NSX-T). NSX-V integrates directly into vCenter. NSX-T is a product which is built to support multiple platforms, as such, NSX Manager is standalone. Just like NSX-v, NSX-T architecture has built-in separation of the data plane, control plane and management plane. This separation delivers multiple benefits, including scalability, performance, resiliency and heterogeneity.

The main differences between NSX-v and NSX-T are:

- The NSX-T control plane encompasses a clustered control-plane (CCP) running on controller nodes and a localized control plane (LCP) on compute endpoints.
- The NSX-T data plane can be enabled on ESXi, KVM hypervisors and appliances providing gateway functionality called edge nodes to provide rich networking and security services. (NSX-T Edge Nodes will be cover in a part II of this series)
- The NSX-T data plane introduces a host switch (rather than relying on the vSwitch), which decouples it from the compute manager and normalizes networking connectivity.
- NSX-T encapsulation is based on GENEVE instead of VXLAN (strongly recommend reading this post of Bruce Davie - VMware APJ VP & CTO - about GENEVE)

The following table (Table 4.1) explains which features are available in NSX-v and NSX-T. Please be aware that some functionalities might not be available through Neutron APIs:

Feature	NSX-v (6.3)	NSX-T (2.1)
Logical Switches	Yes	Yes
Distributed Logical Routers	Yes	Yes
Edge Routers	Yes	Yes
NAT	Yes	Yes
Distributed Firewall	Yes	Yes
Edge Firewall	Yes	Yes
Load Balancer	Yes	Yes
VPN	Yes	No
Hypervisors Supported	vSphere	vSphere, KVM
Encapsulation Protocol	VXLAN	GENEVE
vRNI Support	Yes	No
Security Partners Integration	Yes	No
L2 Bridging (VXLAN-VLAN)	Yes	Yes
Hardware VTEP Integration	Yes	No

DPDK Support	No	Yes
vRealize Automation Integration	Yes	Yes*
Neutron Plugin	Yes	Yes
Operational Tools (Traceflow, Port Connection)	Yes	Yes

Table 4.1: Features in NSX-v and NSX-T

Note: vRealize Integration is through OpenStack API, not direct NSX-API

4.6 NSX-v Supported Topologies and Integration

The following table (Table 4.2) summarizes the topologies supported by the NSX-Neutron plug-in:

Use Case		Comments
VLAN-backed L2, no L3 services	Micro-segmentation only	Uses no overlays. Security Groups leverage Distributed Firewall policies
VLAN-backed L2, L3 services, LBaaS optional	Leverage VLANs for L2, NSX Edge for L3	Uses no overlays. L3 provided by NSX Edge. No distributed routing support. Static routes only
L2/L3 overlays, no NAT, LBaaS optional	Enterprise customers that don't need overlapping IP addresses	Can use distributed router, NSX Edge, or both. No overlapping IPs allowed. Static routes only. Very efficient. Preferred enterprise model
L2/L3 overlay, NAT, LBaaS optional	Enterprise customers that need overlapping IPs	Can use distributed router, NSX Edge, or both. Static routes only. Very efficient. Preferred cloud provider and service provider model

Table 4.2: Topologies supported by the NSX-Neutron plug-in

4.6.1 NSX-v Microsegmentation and Security Groups

Neutron Security Groups have historically implemented Linux iptables when running on KVM, or Open vSwitch stateless matches to filter traffic at the hypervisor level. Both approaches have proven inadequate, and there is serious work underway aimed at addressing these issues (VMware is a contributor to these efforts).

When using NSX and vSphere, we deploy a stateful firewall on every ESXi host. That means that every hypervisor protects the microcosm of virtual machines that it hosts, providing the notion of a distributed dataplane. This is called a distributed firewall, or DFW. The NSX DFW runs in the kernel of ESXi and enables granular security controls at the VM vNIC level. When using Neutron Security Groups, the NSX DFW is configured, through the plug-in integration. Neutron Security Groups are mapped to instances, meaning the NSX DFW **protect the VM unit (Figure 4.7)**.

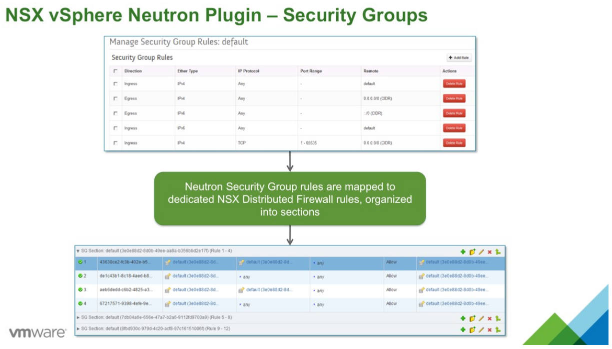


Figure 4.7: NSX vSphere Neutron Plugin – Security Groups

Running an actual firewall on each hypervisor within your OpenStack cloud has the following benefits:

- The attributes of micro-segmentation are readily available to the OpenStack admins and tenants.
- Compliance requirements (such as PCI, or HIPAA) can be met, without sacrificing the openness of OpenStack as your IaaS layer.
- Advanced L4-L7 security services through service insertion and service chaining.

The NSX firewall scales as your ESXi footprint grows. Increasing your compute capacity due to the organic growth of your business, automatically means that you are also adding security and compliance to your virtual infrastructure.

It is important to note that Neutron Security Groups and NSX micro-segmentation can be used standalone, without adopting L2 overlays and L3 virtualization. While not as flexible as a full network virtualization implementation, the micro-segmentation only use case is very popular with our customers and provides a great insertion vector for OpenStack and NSX, without disrupting any operational model that is already in place.

4.6.2 NSX-v Edge Integration

Tenants in OpenStack are allowed, by default, to create their own IP subnets and routers. We will cover some of the NSX capabilities available with the Neutron plug-in. Before we do that though, just a quick parenthesis about self-service in general: As OpenStack gains more traction in the Enterprise, we are learning that these self-service capabilities might not be desirable. Administrators might want to remain in control of the IP subnetting, for example, especially if the use case calls for routable IP address space everywhere. OpenStack lacks the necessary controls to enforce this type of restriction, so short of forbidding API access to specific functions or relying on the good-old honor system, customers have little to no choice when it comes to the built-in OpenStack governance. Projects such as OpenStack Congress are attempting to bridge this gap, and some commercial products are already providing the controls that IT requires. vRealize Automation is a VMware platform that offers comprehensive, scalable governance and can be deployed on top of OpenStack in order to satisfy the needs of IT for checks and balances.

Back to the L3 services discussion, we stated that a tenant could create Neutron routers. The NSX-Neutron plug-in translates this provisioning request and signal NSX Manager to create an NSX Edge Services Gateway (ESG). The ESG is a network appliance (Figure 4.8) that supports several network features (of which some are visible to OpenStack, by the way) and that is broadly used in our integration.

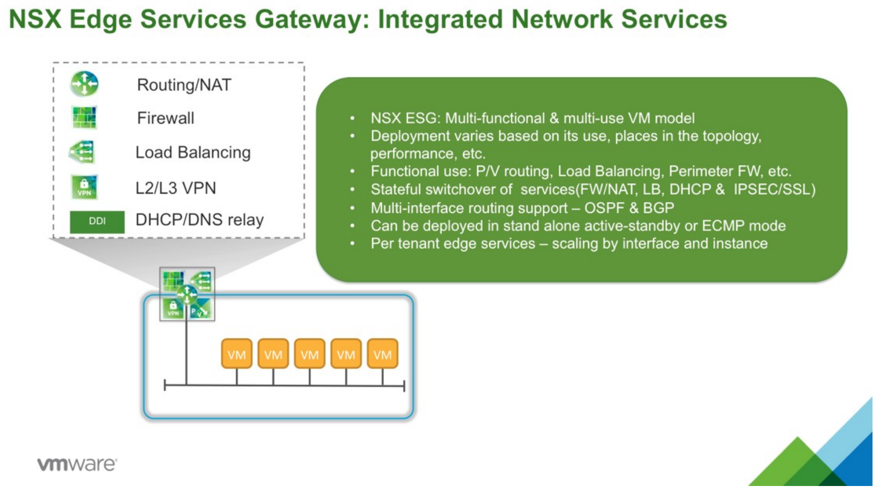


Figure 4.8: NSX Edge Services Gateway

After the Neutron router is created, you can connect previously provisioned Neutron networks (L2 segments) to it and make routing available between them.

The uplink of a Neutron router can be connected to an **External network**. This is also known as **setting the gateway**. This External network must sit on routable IP address space within the organization and is also the network where floating IPs reside. If the tenant networks sit on an RFC1918 space, then the Neutron router must perform **Network Address Translation**, or NAT (source NAT for internal to external access, and DNAT for floating IPs). If the tenant networks sit on routable subnets, then the router does not have to perform NAT.

The tenant networks can also be backed by VLAN, instead of VXLAN. If the tenant wants to or must use VLANs instead of VXLANs, then the administrator must create these networks on behalf of the tenant.

Tenant routers can be **exclusive** (defined at provisioning time using an API extension) or **shared** (default behavior). Exclusive and Shared refers to the fact that a router VM (NSX Edge Services Gateway) can be shared across Tenants or be dedicated to a Tenant. Depending on your performance and scalability expectations, you can choose one or the other.

When using NSX, the Neutron L3 services might include a **distributed router (Figure 4.9)**, which is a very powerful capability in NSX that allows for the optimization of East-West traffic in routed topologies. This is a good example of an enterprise-grade capability of NSX and differentiator from the reference implementation, which can be leveraged without compromising the basic tenet of OpenStack in keeping the API open. A distributed router sends traffic from the source hypervisor to the destination hypervisor without hairpinning the packets through an NSX ESG or a physical router SVI. This increases performance significantly and streamlines traffic engineering within the data center.

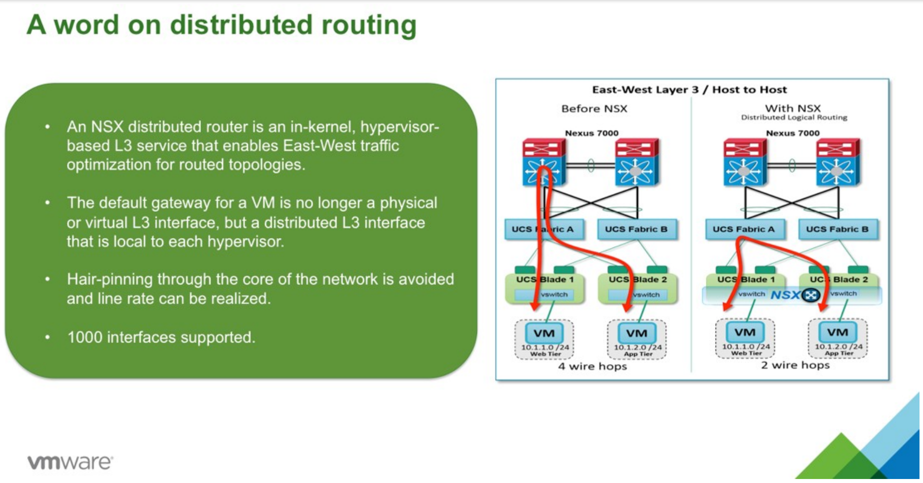


Figure 4.9: Distributed Routing

Finally, up until VIO 3.1, our Neutron implementation only supported static routing, which meant that when using NSX with your OpenStack implementation, tenants could not make use of dynamic routing, even though NSX has supported both OSPF and BGP since day 1. A **BGP speaker** is available in the **Liberty** release of OpenStack Neutron. BGP has been added to the Ocata release of Neutron plug-in in VIO 4.0.
The picture below shows the basic supported topologies by the NSX-Neutron plugin (Figure 4.10):

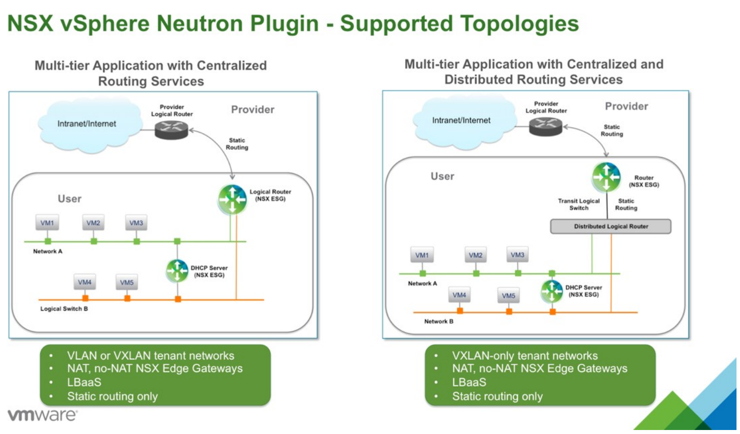


Figure 4.10: NSX vSphere Neutron Plugin – supported topologies

DHCP Services

In our implementation of DHCP, we replace the dnsmasq process that is used by the reference implementation with an NSX Edge Services Gateway configured with static DHCP bindings This approach has proven to be much more reliable at scale (thousands of VMs).

Logic in the NSX-Neutron plug-in automatically determines how to use an Edge Services Gateway for DHCP services. Depending on the use case (overlapping IPs or non-overlapping IPs) the same ESG can be reused for multiple tenant networks, as shown in the following diagram (Figure 4.11)

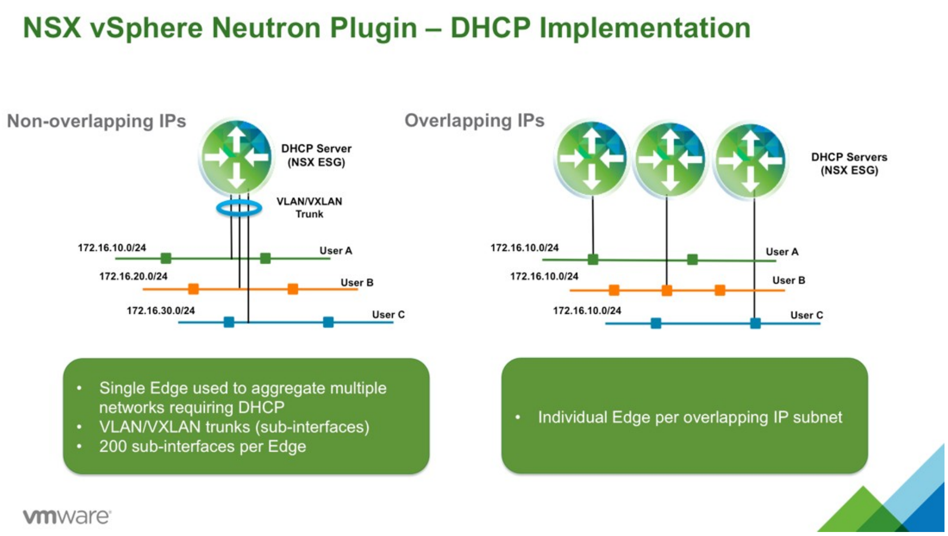


Figure 4.11: NSX vSphere Neutron Plugin – DHCP implementation

A future enhancement to the NSX ESG will allow for separate namespaces per DHCP scope, using something similar to the virtual routing and forwarding function (VRF) found in most commercial router implementations. This will reduce the number of ESGs that are used in an overlapping IP scenario, resulting in a more optimal utilization of the resources dedicated to the cluster where the ESGs reside.

4.6.3 NSX-v LBaaS integration

As of the Mitaka version of the NSX vSphere plug-in, Neutron LBaaS v2.0 support has been incorporated. The workflow includes the following capabilities:

- Tenants can create application pools (initially empty).
- Tenants add several members to the pool (instance IP address).
- Tenants create one or more health monitors.
- Tenants associate the health monitors with the pool.
- Tenants create a virtual IP (VIP) with the pool.

Supported protocols: TCP, HTTP, and HTTPS.

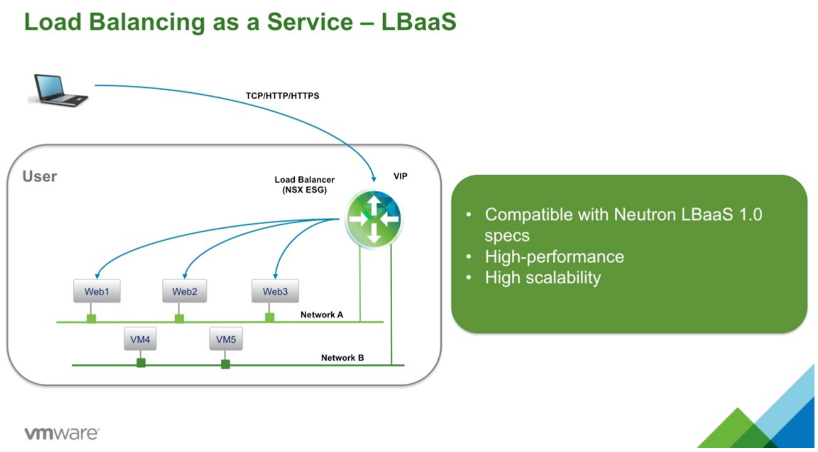


Figure 4.12: Load Balancing as a Service

As with other network services in our implementation, we leverage the NSX ESG as a one-arm balance in addition to a required Neutron router (Figure 4.12).

4.7 NSX-v Policy Redirection

OpenStack is quickly and steadily positioning itself as a great Infrastructure-as-a-Service solution for the Enterprise. Originally conceived for that proverbial DevOps Cloud use case (and as a private alternative to AWS), the OpenStack framework has evolved to add rich Compute, Network and Storage services to fit several enterprise use cases. This evolution can be evidenced by the following initiatives:

1. Higher number of commercial distributions are available today, in addition to Managed Services and/or DIY OpenStack.
2. Diverse and expanded application and OS support vs. just Cloud-Native apps (a.k.a “pets vs. cattle”).
3. Advanced network connectivity options (routable Neutron topologies, dynamic routing support, etc.).
4. More storage options from traditional Enterprise storage vendors.

The only robust option for application security offered in OpenStack are Neutron Security Groups. The basic idea is that OpenStack tenants can be in control of their own firewall rules, which are then applied and enforced in the dataplane by technologies such as Linux IP Tables, OVS conntrack or, as it is the case with NSX vSphere, a stateful and scalable Distributed Firewall with vNIC-level resolution operating on every ESXi host.

Neutron Security Groups are designed for intra-tier and inter-tier L3/L4 protection within the same application environment (the so-called “East-West” traffic).

In addition to Neutron Security Groups, projects such as Firewall-as-a-Service (FWaaS) are also attempting to onboard next generation security services onto these OpenStack Clouds and there is an interesting roadmap taking form on the horizon. The future looks great, but while OpenStack gets there, what are the implementation alternatives available today? How can Cloud Architects combine the benefits of the OpenStack framework and its appealing API consumption model, with security services that provide more insight and visibility into the application traffic? In other words, how can OpenStack Cloud admins offer next generation security right now, beyond the basic IP/TCP/UDP inspection offered in Neutron?

The answer is: With VMware NSX.

NSX natively supports and embeds an in-kernel redirection technology called Network Extensibility, or NetX. Third-party ecosystem vendors write solutions against this extensibility model, following a rigorous validation process, to deliver elegant and seamless integrations. When the solution is implemented, the notion is simply beautiful: leverage the NSX policy language, the same language that made NSX into an effective solution for micro-segmentation, to “punt” interesting traffic toward the partner solution in question. This makes it possible to have protocol-level visibility for East-West traffic. You can also follow this approach to create a firewall rule-set **that looks like your business and not like your network.** Applic ation attributes such as VM name, OS type, or any other vCenter object can be used to define said policies, irrespective of location, IP address, or network topology. When the partner solution receives the traffic, the security administrators can apply deep traffic inspection, visibility, and monitoring techniques to it.

NSX Service Redirection works in VIO in the following manner:

1. Deploy VMware Integrated OpenStack and NSX following the best practices outlined in this guide and in the official VIO documentation.
2. Follow documented configuration best practices to integrate NSX and the Partner Solution. The list of active ecosystem partners can be found here.
3. Create an NSX security policy to classify the application traffic by using the policy language mentioned above. This approach follows a wizard-based provisioning process to select which VMs will be subject to deep level inspection with Service Composer.

The cloud admin user updates the nsx.ini file to enable policy integration, and select one of the policies as the default, and set it in the nsx.ini file, and restart neutron:

[nsxv]

use_nsx_policies = True default_policy_id = policy-6

allow_tenant_rules_with_policy = True / False

IMPORTANT: For detailed configuration steps that can persist after an upgrade/update, please refer to the VIO documentation. The changes in nsxv.ini should be implmented using the custom.yml file.

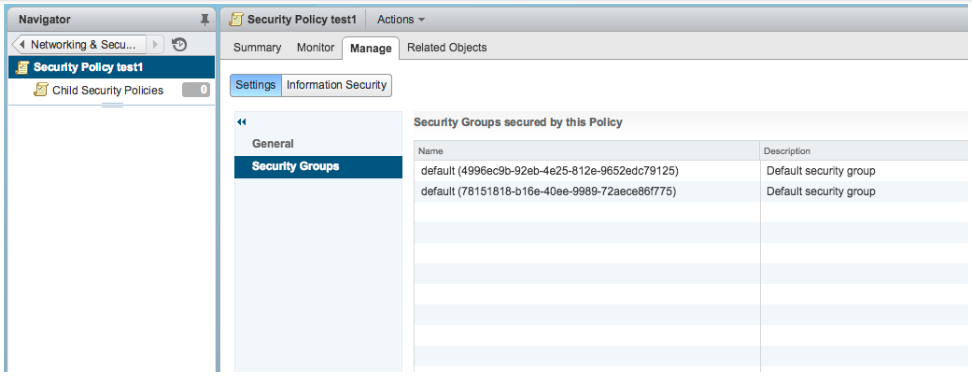
Now there will be 4 types of security groups for each tenant:

1. A default security group using the policy ID from nsx.ini. The cloud admin can select another policy or multiple policies as default.
2. A Provider security group with a policy, added automatically to each compute ports. The tenant cannot remove this group from the port.
3. An Optional security group with a policy, added manually to each compute ports. The tenant can select the groups to use for each port.
4. Allow_tenant_rules_with_policy is set to True, Regular security groups. The tenant can create such groups with rules, and attach them to ports in addition.

The cloud admin uses OpenStack or VIO to create or update the policy security groups per tenant, to use a specific policy:

neutron security-group-create/update --policy=<nsx-policy-id> <neutron-sec-group-id>

The plug-in creates an nsx security group which is applied to this policy, and saves it in the db security group mapping (like a regular security group). Looking at the vsphere you can see it on the policy's Security Groups tab:



Capture 4.1: nsx security group

When a VM is booted, the default security group (which now uses the policy) or a specific security-group is used as usual. In addition, the provider security groups of this tenant are also used. You cannot remove the provider security groups.

In the openstack API, the user can see that a specific VM port is assigned to one or multiple security groups, and can verify that a specific security group is assigned to a policy:

```
neutron security-group-show 4996ec9b-92eb-4e25-812e-9652edc79125

+----------------------+--------------------------------------------------------------------+

| Field                | Value                                                              |

+----------------------+--------------------------------------------------------------------+

| created_at           | 2016-10-06T12:22:41Z                                               |

| description          | Default security group                                             |

| id                   | 4996ec9b-92eb-4e25-812e-9652edc79125                               |

| logging              | False                                                              |

| name                 | default                                                            |

| policy               | policy-6                                                           |

| project_id           | 8c2704a5694c480daabdeed0d0b435e6                                   |

| provider             | False                                                              |

| revision_number      | 1                                                                  |

| security_group_rules |                                                                    |

| tenant_id            | 8c2704a5694c480daabdeed0d0b435e6                                   |

| updated_at           | 2016-10-06T12:22:41Z                                               |

+----------------------+--------------------------------------------------------------------+
```

Implementation details:

Openstack plugin Configuration:

- New nsxv parameters in the nsx.ini:
  - Use_nsx_policies - True/False (default - False). When it is set, the plugin will work in 'Policy-Mode' (as opposed to the regular 'Rules-Mode')

- Default_policy_id -The configured policy will be used as the default policy for the default tenant security groups.
- Allow_tenant_rules_with_policy - True/False (default: False) - When True, the tenant can also create security groups with rules.

Following the initial integration, the security admin can use the Security Partner management console to create protocol-level security policies, such as application level firewalling, web reputation filtering, malware protection, antivirus protection, and so on.

There are some Neutron enhancements in the works, such as Flow Classifier and Service Chaining, that are looking “split” the security consumption between admins and tenants, by promoting these redirection policies to the Neutron API layer, thus allowing a tenant (or a security admin) to selectively redirect traffic without bypassing Neutron itself. This implementation, however, is very basic when compared to what NSX can perform natively. We are actively monitoring this work and studying opportunities for future integration. In the meantime, the approach outlined above can be used to get the best of both worlds: the APIs you want (OpenStack) with the infrastructure you trust (vSphere and NSX).

4.8 NSX-v Admin Policy

Starting with VIO 3.1, a new security service has been made available when using NSX (both vSphere and T). This service is called "admin policy" and it allows an OpenStack admin to create a new kind of per-project Neutron security group to explicitly block unwanted traffic, where all rules have implicit deny action, instead of the current state where security-groups rules have implicit allow action.

As mentioned above, these security groups can only be managed by the an OpenStack admin, and this behavior is enforced via *policy.json* restric tions. As with normal security-groups, logical-ports are associated with one or more of these "strict" security-groups. When a packet reaches a port (inbound or outbound), it is matched against the strict security-groups rules. If no strict rule is hit, the packet is processed against the other (normal) security-groups rules, as usual.

This capability is implemented in the NSX plugin. Strict security-groups rules are organized in Distributed Firewall sections, which are added at the top of the DFW. Normal security-groups DFW sections are always created underneath the strict sections.

One application of this capability is a strict policy that widely restricts Telnet in an OpenStack Cloud, for example. The admin would create a strict security group and thencreate a deny rule for TCP port 23 (inbound and outbound, for example). Every instance launched after the creation of the strict rule will be mapped to the Neutron Security Group, even if the user launching the instance does not select it. Additionally, if the tenant allows TCP port 23 in their own security group, that rule will never be honored as the strict rule will always take precedence.

This capability offers a programatic way to enable blanket policy by leveraging the Neutron APIs and the microsegmentation benefits of the NSX Distributed Firewall.

4.9 NSX-v Neutron End-user Workflow

The following section covers the specifics of the Neutron and NSX vSphere integration. We will describe, one by one, the Neutron services typically leveraged in an OpenStack Cloud and then show the corresponding NSX construct that gets provisioned or configured in the back end. Whenever possible, we will show the Horizon workflow to provide the construct, but some operations are only available from the OpenStack CLI client, in which case we will show the necessary command (or commands) to provide that particular network service.

As a reminder, when using NSX vSphere (NSX-v) as the SDN back end for VIO, the following services are available to your OpenStack tenants and admins:

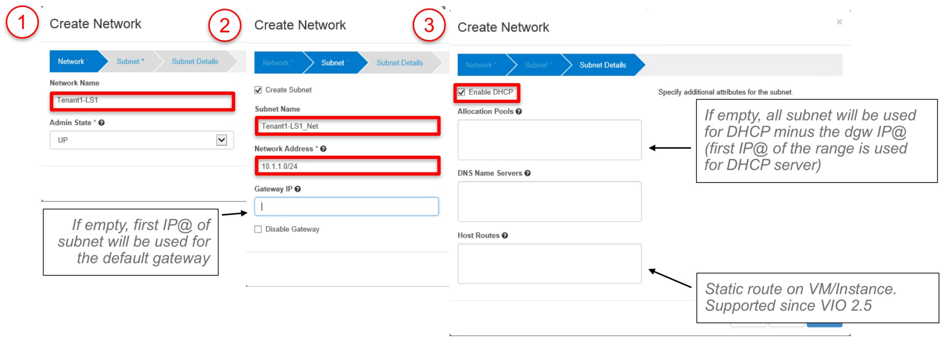
- L2 services: VLAN-backed Provider networks, VXLAN-backed Provider and Tenant networks (allowing support of any L2/L3 underlay fabric), DHCP server for Nova Instances,with overlapping and non-overlapping IP address support and L2 gateways for overlay-to-VLAN bridging.
- L3 services: VLAN-backed and VXLAN-backed External Networks, Logical Routers (Centralized-Shared, Centralized-Exclusive, and Distributed), static and dynamic routing (BGP), global Source NAT (SNAT) and Floating IPs (SNAT/DNAT), and no-NAT routers.
- Security services: Neutron Security Groups for tenants and admins (East-West firewalling), Firewall-as-a-Service (FWaaS) for North-South firewalling and Neutron Port Security (spoofguard).
- Load balancing services: Neutron LBaaS 2.0.

- Metadata proxies: Provide seamless connectivity to Nova Metadata Services leveraging the coud-init agent.
- Neutron availability zones: Provide the ability to do deterministic placement of network elements for added redundancy and availability.

4.9.1 L2 Services

Tenant Networks and DHCP (UI)

Under "Project - Network – Networks", Create Network



Capture 4.2: Tenant Networks and DHCP

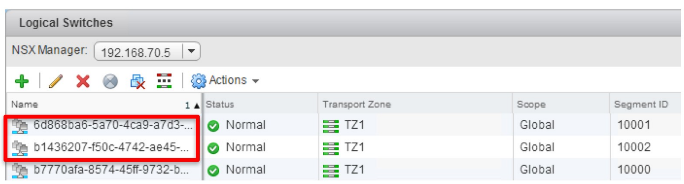
Tenant Networks and DHCP (CLI)

```
# neutron net-create Tenant1-LS2 Created a new network:

+---------------------------+-------------------------------------+

| Field                     | Value                               |

+---------------------------+------------------------------------ +

| admin_state_up            | True                                |

| id                        | b1436207-f50c-4742-ae45-58226f8dd631 |

| name                      | Tenant1-LS2                         |

| port_security_enabled     | True                                |

| router:external           | False                               |

| shared                    | False                               |

| status                    | ACTIVE                              |

| subnets                   |                                     |

| tenant_id                 | 40e97bec2b06462098b241f04a224167    |

+----------------------+------------------------------------------+


# neutron subnet-create --name Tenant1-LS2_Net Tenant1-LS2 10.1.2.0/24
```

```
[--dns-nameservers list=true 10.33.38.1 10.33.38.2 --dns_search_domain mydomain.local] Created a new subnet:
+-------------------+--------------------------------------------+
| Field             | Value                                      |
+-------------------+--------------------------------------------+
| allocation_pools  | {"start": "10.1.2.2", "end": "10.1.2.254"} |
| cidr              | 10.1.2.0/24                                |
| dns_nameservers   | 10.33.38.1                                 |
|                   | 10.33.38.2                                 |
| enable_dhcp       | True                                       |
| gateway_ip        | 10.1.2.1                                   |
| host_routes       |                                            |
| id                | d7eb0562-95d9-42ab-99b7-cfb40519b45c       |
| ip_version        | 4                                          |
| ipv6_address_mode |                                            |
| ipv6_ra_mode      |                                            |
| name              | Tenant1-LS2_Net                            |
| network_id        | b1436207-f50c-4742-ae45-58226f8dd631       |
| subnetpool_id     |                                            |
| tenant_id         | 40e97bec2b06462098b241f04a224167           |
+-------------------+--------------------------------------------+
```

- For each OpenStack Network created:
  1 – 1 Logical Switch is created in NSX

    Under "NSX – Logical Switches"



Note: The Logical Switch name is the OpenStack Network UUID.
You can find the OpenStack Network UUID via Horizon (edit the network) or CLI

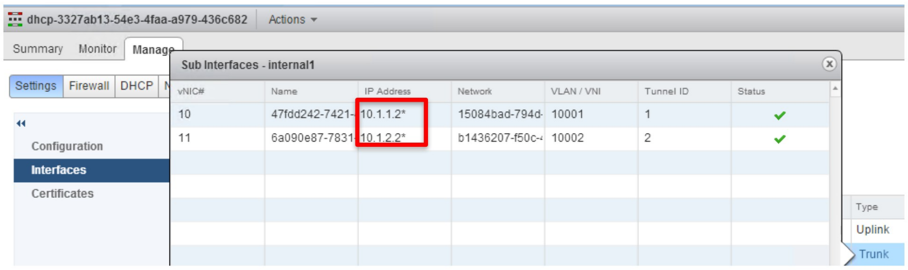
Capture 4.3: Create logical switches

- For each OpenStack Network created:

2 – The DHCP Edge is updated

Under "NSX – NSX Edges"

- with LS in trunk interface
- with IP@ = first IP@ of DHCP range



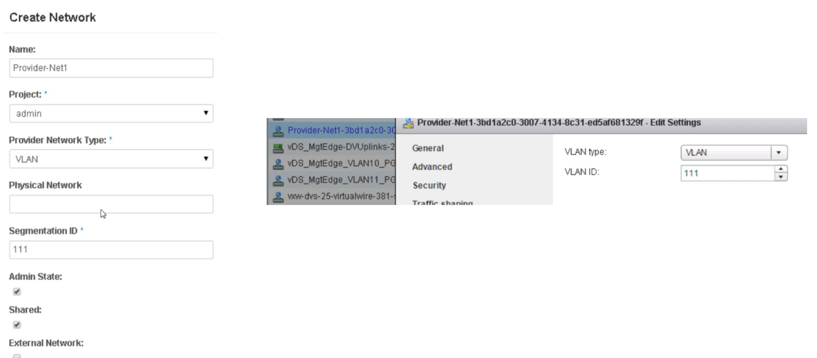
Capture 4.4: Create DHCP Bridge

A new DHCP Edge is created (from the backup Edge Pool) if one of the following circumstances arises:

- More than 160 logical networks
- New subnet is IP overlapping with an existing network on the DHCP Edge
- The subnet is attached to a Distributed Logical Router

VLAN-backed Provider Network (Horizon)

– Under "Admin - System Panel - Networks - Create Network" with option "shared". Notice that a new distributed Portgroup is created in the VDS. If separate VDSes are being used, a dvPG will be created on each one (CLI only)



Capture 4.5: Create VLAN-backed provider network

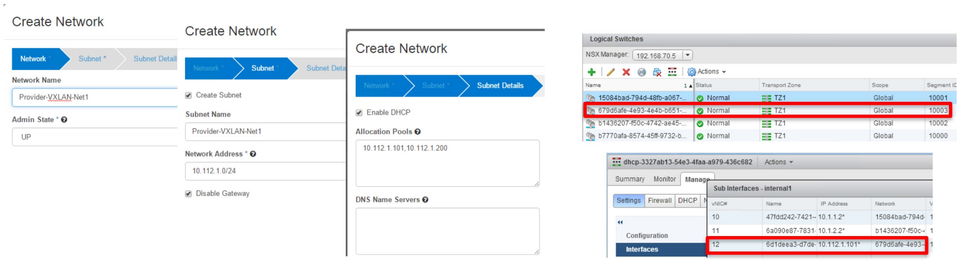
After this network is created, the admin can proceed to create the corresponding Neutron subnet.

vmware®

VXLAN-backed Provider Network (Horizon)

Create an admin VXLAN Network + Subnet

- This step is NOT done under "Admin - System Panel - Networks - Create Network"
- This step is done under "Project – Network – Create Network"

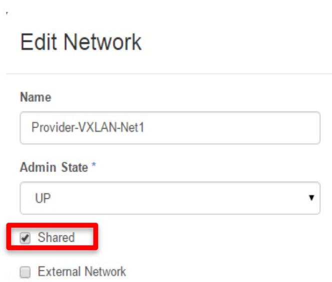


Capture 4.6: Create VXLAN network subnet

Update that Network to be Shared (as admin)

- Under "Admin - Network", edit that created network and enable "Shared"

The network is now a VXLAN provider network



Capture 4.7: Update VXLAN provider network to be shared

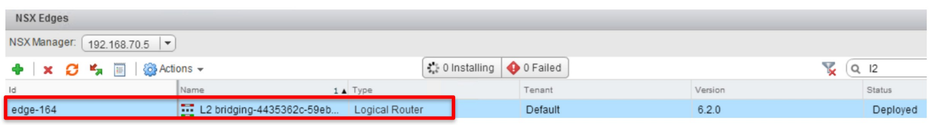
L2 Bridging (VXLAN-VLAN)

This operation is only possible through the Python CLI. Using the vCenter Management Object Base (MOB), find the MoRef for the dvPG associated with the VLAN in question, which must be trunked to the Edge Cluster. In this example, we are using dvportgroup-434.

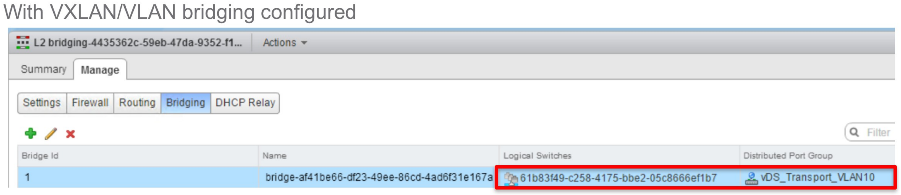
```
# neutron-l2gw l2-gateway-create L2GW1 --device name=L2GW1,interface_names=dvportgroup-434 Created a new

l2_gateway:
```

```
+-----------+------------------------------------------------------------------------+
| Field     | Value                                                                  |
+-----------+------------------------------------------------------------------------+
| devices   | {"interfaces": [{"segmentation_id": [], "name": "dvportgroup-434"}], "id": "21d7078c-10b9-
40bc-bbb3-418d296eac3c", "device_name": "edge-164"}  |
| id        | 84c59e94-b0ce-43e7-9c52-a0a5f88b7014                                   |
| name      | L2GW1                                                                  |
| tenant_id | b0131a0680144c079eeeca26c2265ec0                                       |
+-----------+----------------------------------------------------------------------+
# neutron-l2gw l2-gateway-connection-create L2GW1 Tenant1-LS1 --default-segmentation-id 10
Created a new l2_gateway_connection:
+-----------------+--------------------------------------+
| Field           | Value                                |
+-----------------+--------------------------------------+
| id              | ddf11ca0-16ed-40ab-8892-a208d532627a |
| l2_gateway_id   | 84c59e94-b0ce-43e7-9c52-a0a5f88b7014 |
| network_id      | 61b83f49-c258-4175-bbe2-05c8666ef1b7 |
| segmentation_id | 10                                   |
| tenant_id       | b0131a0680144c079eeeca26c2265ec0     |
+-----------------+----------------------------+
```

- For each VXLAN/VLAN bridging created
  - One NSX DLR is created



  - With VXLAN/VLAN bridging configured

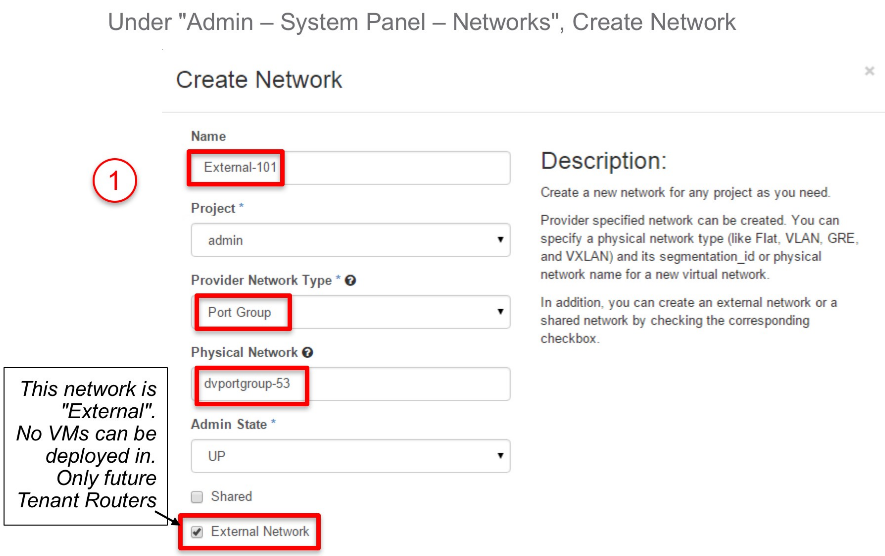


Capture 4.8: VXLAN/VLAN Bridging Configured

4.9.2 L3 Services

External Network VLAN-backed (Horizon)

Find the MoRef for the dvPG associated with the VLAN in question.



Capture 4.9: Create External Network

Proceed to create the External Subnet after this step.

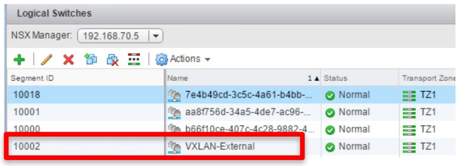
External Network VLAN-backed (CLI)

```
# neutron net-create External-102 -- --provider:network_type=portgroup
--provider:physical_network=dvportgroup-1443 --router:external=True (requires admin credentials)
Created a new network:
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | 5f4a2f7a-b888-4d99-8089-ed2f93178476 |
| name                      | External-102                         |
| port_security_enabled     | True                                 |
| provider:network_type     | portgroup                            |
| provider:physical_network | dvportgroup-187                      |
| provider:segmentation_id  | 0                                    |
| router:external           | True                                 |
```

```
| shared                      | False                                  |
| status                      | ACTIVE                                 |
| subnets                     |                                        |
| tenant_id                   | 423a48a7cf7b43689fa529692299e7ab       |
+-----------------------------+----------------------------------------+
```

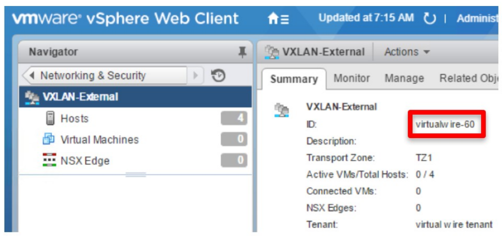
External Network VXLAN-backed (Horizon):

1. Get a Logical Switch created from NSX-v

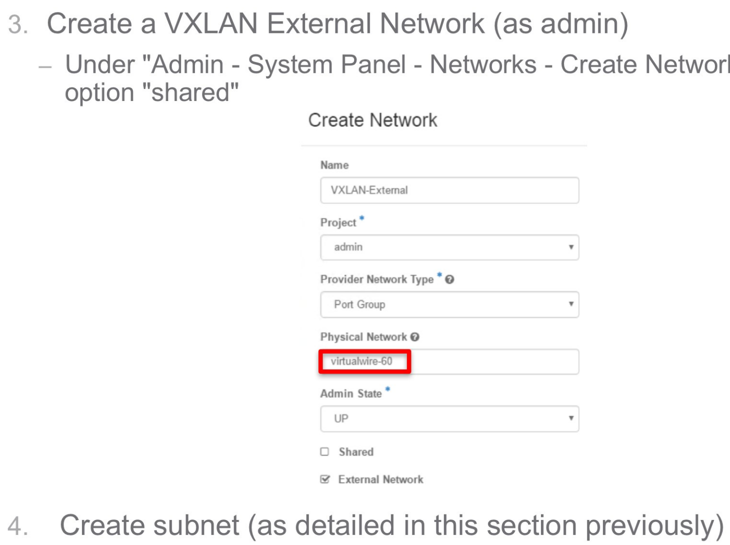


2. Find the NSX-v VXLAN VirtualWire-ID

   Edit VXLAN Logical Switch



Capture 4.10: Create and edit logical switch

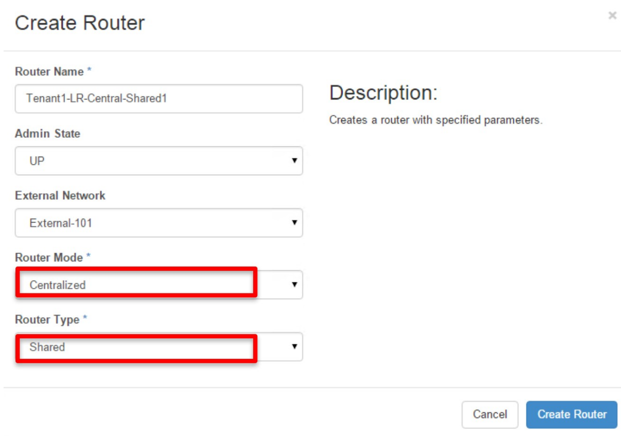


Capture 4.11: Create VXLAN external network (as admin)

Logical Routers - Centralized, Distributed, Exclusive and Shared (Horizon)

When using NSX-v with VIO, tenants and admins can configure routers that share the same router VM (NSX Edge) across tenants (shared), or dedicate the router VM to a single tenant (exclusive). Routers can also be centralized (NSX Edge processes the routing functions) and distributed (source hypervisor processes the routing function to optimize East-West traffic flows). You can use VIO to create these routers from Horizon (Figure 4.23):

- Create Router
  Under "Project - Network – Routers"



Capture 4.12: Create Logical Router

After router creation, you can proceed to add interfaces as usual. In the NSX back end, the NSX Edge Services Gateway is leveraged for centralized routing functions, and the Distributed Logical Router (DLR) is leveraged for distributed routing functions.

No-NAT Topologies (works with Centralized and Distributed Routers)

To disable NAT in a Neutron router, run the following CLI:

#neutron router-update Tenant1-LR-Central-Exclusive1 --external_gateway_info type=dict network_id=ccee6823-360d-43d7-99b0-a7e22b82433f,enable_snat=False

Updated router: Tenant1-LR-Central-Exclusive1

Additional L3 Services

We are not showing the UI or CLI to create and assign floating IP addresses, connect multiple routers to the same network, or perform static routes and BGP. These services are extensively documented in the official OpenStack docs repository, available at docs.openstack.org.

4.9.3 Security Services

Tenant Security Groups (Horizon):

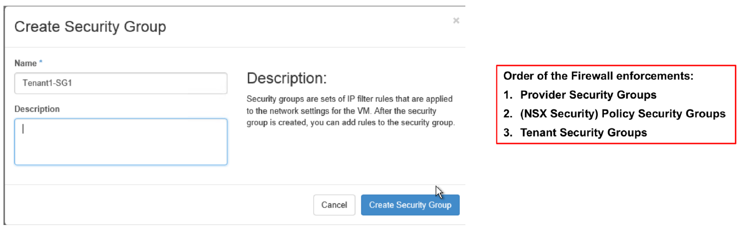
Tenant Security Groups have the following characteristics:

- Every rule has an ALLOW action, and there is an implicit any-any-any-block rule at the end.
- Tenants can create or delete Tenant Security Groups.

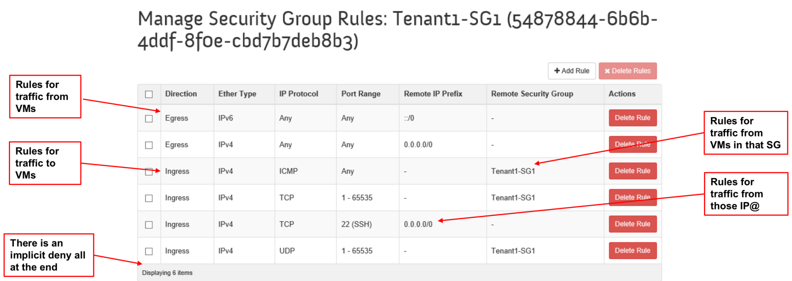
- Every tenant can have one or more Tenant Security Groups associated to a VM (instance).
- Tenant Security Groups have lower precedence than Provider and Policy Security Groups.

These Neutron Security Groups may be utilized by the tenants, for example, to control what traffic is allowed within their own applications.

- Create a Security Group
  Under "Project – Compute – Access & Security", Create "Security Group"



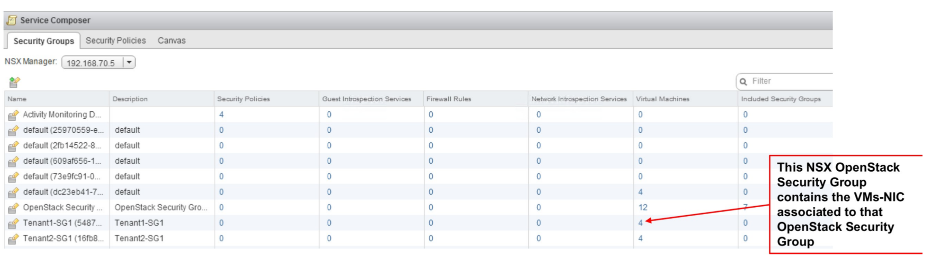
- Manage rules in the Security Group
  Under "Project – Compute – Access & Security – Manage Rules"



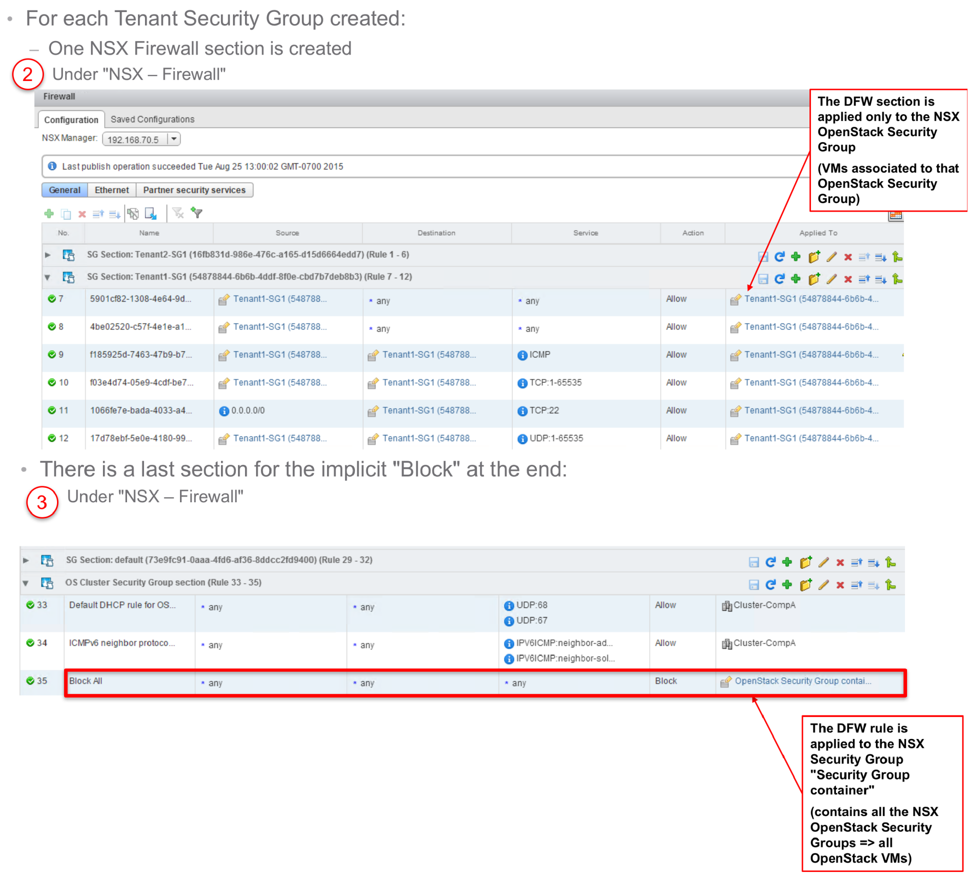
Capture 4.11: Manage security group rules

After this, instances can be launched with the corresponding Security Group attachment. Here is what happens in NSX:

- For each Tenant Security Group created:
  - One NSX Security Group is created
  - (1) Under "NSX – Service Composer – Security Groups"



Capture 4.12: Create NSX Security Groups



Capture 4.12: Create NSX Security Groups

Provider Security Groups (Horizon)

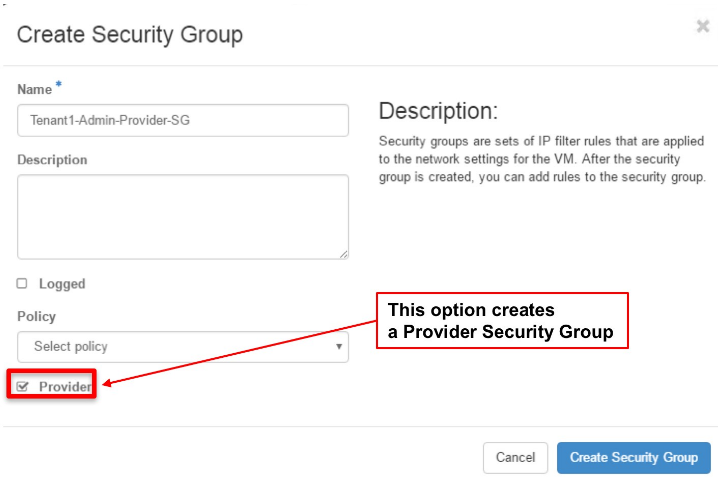
Provider Security Groups have the following characteristics:

- Every rule has a BLOCK action, instead of ALLOW.
- Only an administrator can create or delete Provider Security Groups.
- Every tenant can have one or more Provider Security Group, which must be explicitly created by an administrator.
- Provider Security Group have precedence over Tenant Security Groups so that Provider Security Rules are always applied.
- Provider Security Groups are automatically attached to new tenant instances.

They may be utilized by the admin in a number of situations:

Blacklisting IP addresses, for example a bogon list
Prohibiting the use of specific protocols by a tenant, for example Telnet

- Create a Security Group
  Under "Project – Compute – Access & Security", Create "Security Group"

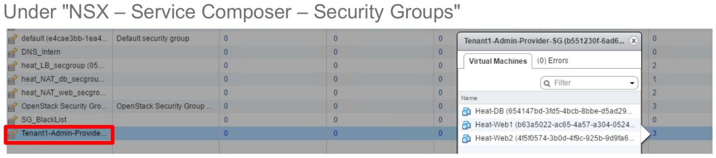


Capture 4.13: Create Provider Security Groups

After creating rules, you can manage them in the Provider Security Group for Tenant1 as described in the Tenant Security Group section earlier. Here is what happens in NSX:

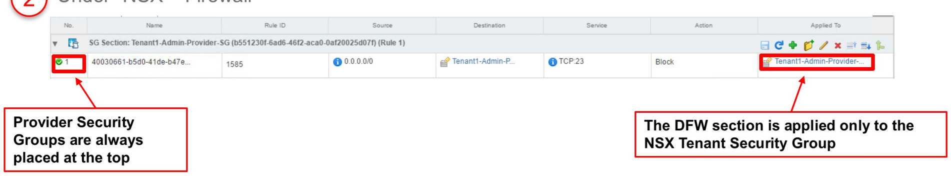
- For each Provider Security Group created:
  - One NSX Security Group is created containing all the Tenant VMs

(1) Under "NSX – Service Composer – Security Groups"



  - One NSX DFW Section is created at the top

(2) Under "NSX – Firewall"



Capture 4.14: Create NSX Tenant Security Groups

Advanced Service Insertion: Policy Security Groups

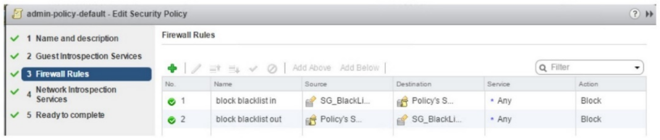
Policy Security Group have the following characteristics:

- The NSX admin creates a Policy security with guest introspection service, firewall rules, and/or network introspection service for each tenant or a group of tenants.
- The cloud administrator defines one of those policies as the default for new tenants (in the nsx.ini file).
  The cloud administrator can define some policies as mandatory for some tenants (using provider option), and other policies as optional for some tenants.
- New VMs of this tenant belong to the default policy automatically, and also get all the mandatory policies.
- Each policy can be used for multiple tenants, and also for multiple security groups of the same tenant.

In addition, there will be an option (disabled by default) to allow the tenants to add their own tenant rules in their security groups (which will be evaluated after the policies). This may be utilized by the admin for number of use-cases, for example: Admin / Tenant wants to offer advanced firewall policies (IPS/IDS for instance)

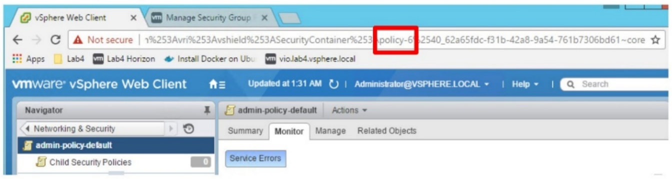
- Create Default Policy Security Group
  - In NSX, create a Security Policy

(1) Under "NSX – Service Composer – Security Policy"



- In NSX, get Security Policy ModID

Under "NSX – Service Composer – Security Policy", open Security Policy and look at the policy-id in the URL



- Create Default Policy Security Group
  - In Neutron NSX-v plugin,

(2) Edit the Policy settings

Mandatory to use Policy Security Groups

```
root@controller01:~# vi /etc/neutron/plugins/vmware/nsxv.ini
# Configure neutron security groups to use NSX policies
use_nsx_policies = True

# (Optional) If use_nsx_policies is true, this policy will be used as the
# default policy for new tenants.
default_policy_id = policy-6

# (Optional) If use_nsx_policies is True, this value will determine if the
# tenants can add rules to their security groups.
allow_tenant_rules_with_policy = True
```

To make those changes persistent currently use the "custom.yml" file.

(Optional) To allow or not Tenants to create their own Tenant Security Groups too

- Restart Neutron

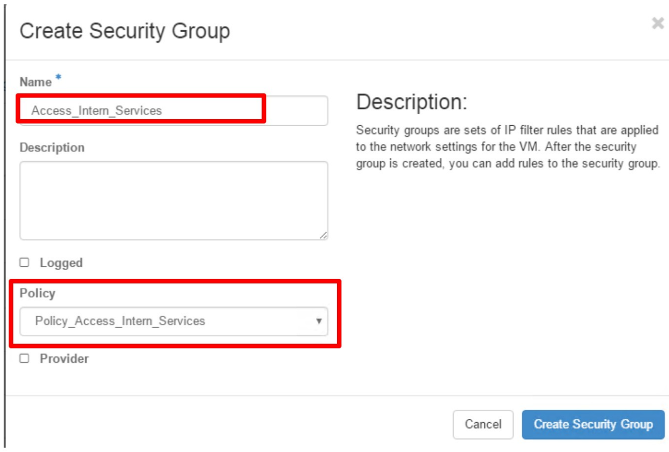
```
root@controller01:~# service neutron-server restart
```

- Now all VMs using the Tenant default Security Group will have the NSX Policy applied

Capture 4.15: Create Default Policy Security Groups

3 • Create a Policy Security Group

Under "Project – Compute – Access & Security", Create "Security Group"

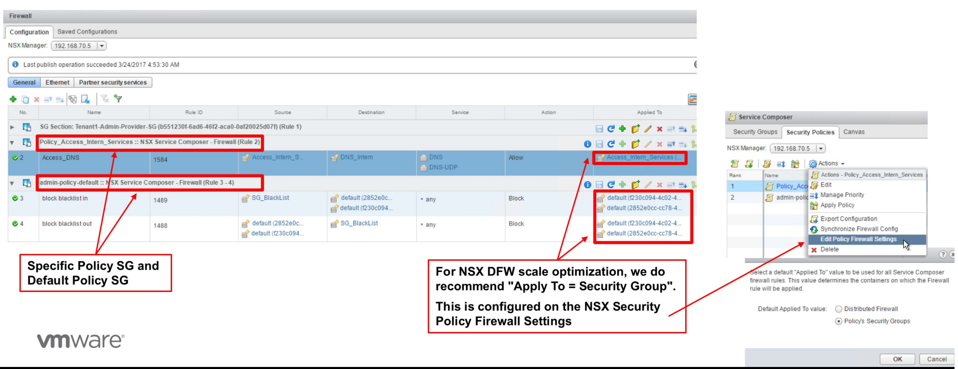


Capture 4.16: Create Policy Security Group

There is also a way to create and assign specific Policy Groups to Tenants that are created after enabling the service in nsxv.ini. Here is what happens in NSX after the creation of these Policy Groups (default and individual):

- For each Provider Security Group created:
  - One NSX DFW Section is created between Provider SG and Tenant SG

    Under "NSX – Firewall"



Capture 4.17: Create specific policy groups after enabling the service in nsxv.ini

Port Security (Horizon and CLI)

Port Security protects against users changing the IP address of their instance, and is enabled by default. Port Security can be configured to allow multiple IP addresses per VM. Some other attributes of this feature are:

- Port Security and Security Groups can be disabled directly on a Neutron Port (Logical Switch Port)
- Port Security can be disabled while a Security Group is still active: per Neutron Port (Logical Switch Port) per Neutron Network (Logical Switch)

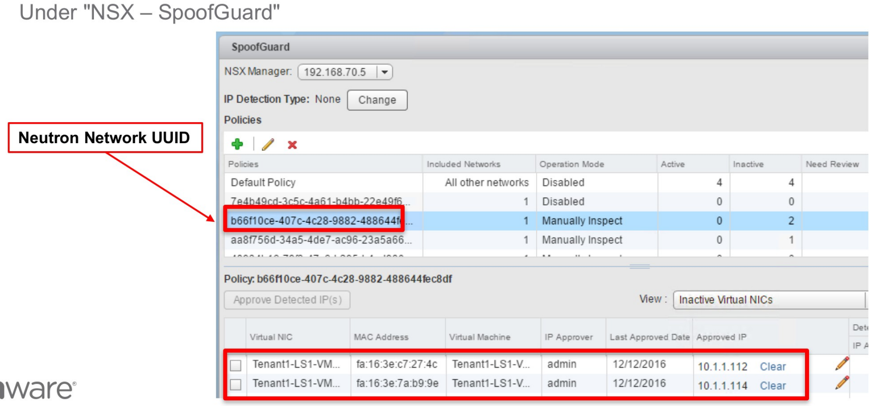
- Visualize Port Security
  - Horizon (UI)
    - Edit Logical Switch Port

      Under "Project – Network – Networks", Edit Network, Go on "Ports" tab, and Select a Port

Capture 4.18: Edit logical switch port

- For each Network created
  - One NSX SpoofGuard Policy is created
- For each Instance created
  - One NSX SpoofGuard Policy entry is created

    Under "NSX – SpoofGuard"



Capture 4.19: Create NSX Spoofguarp policy

To disable Port Security:

```
# neutron port-show 94040c5e-d60b-4f68-a382-eff0ebc7f5db

+-----------------------+-----------------------------------------------------------+
```

```
| Field                 | Value                                                        |
+-----------------------+--------------------------------------------------------------+
| admin_state_up        | True                                                         |
| allowed_address_pairs |                                                                            |
<snip>
| fixed_ips             | {"subnet_id": "75b96d2c-1926-4910-8394-abce2a975b2d", "ip_address":
"10.1.1.112"}                                                                           |
| id                    | 94040c5e-d60b-4f68-a382-eff0ebc7f5db                               |
| mac_address           | fa:16:3e:c7:27:4c                                                  |
| name                  |                                                                    |
| network_id            | b66f10ce-407c-4c28-9882-488644fec8df                               |
| port_security_enabled | True                                                               |
| security_groups       | 4f7e4863-d2bf-4c9e-be85-22639c7e4a27                               |
| status                | ACTIVE                                                             |
# neutron port-update 94040c5e-d60b-4f68-a382-eff0ebc7f5db --port_security_enabled=false Updated
port: 94040c5e-d60b-4f68-a382-eff0ebc7f5db

# neutron port-show 94040c5e-d60b-4f68-a382-eff0ebc7f5db
| Field                 | Value                                                             |
+-----------------------+-------------------------------------------------------------------+
<snip>
| fixed_ips             | {"subnet_id": "75b96d2c-1926-4910-8394-abce2a975b2d", "ip_address":
"10.1.1.112"}                                                                          |
<snip>
| port_security_enabled | False                                                             |
| security_groups       | 4f7e4863-d2bf-4c9e-be85-22639c7e4a27                              |
| status                | ACTIVE                                                            |
```

4.9.4 Load Balancing (CLI)

The first step to create a Load Balancer in Neutron is to create an exclusive router. This guarantees performance and availability of LB services for the application. The router-size parameter allows the tenant to assign more resources or less, depending on the throughput needs. See the NSX documentation for additional information regarding performance of the various NSX Edges.

```
# neutron router-create Tenant1-LR-Exclusive1 --router_type=exclusive [--router-
size=compact|large|xlarge|quadlarge]
Created a new router:
+-----------------------+--------------------------------------+
| Field                 | Value                                |
```

```
+-----------------------+--------------------------------------+
| admin_state_up        | True                                 |
| distributed           | False                                |
| external_gateway_info |                                      |
| id                    | 49211ba7-589a-46b2-aabe-83b154e7db89 |
| name                  | Tenant1-LR-Exclusive1                |
| router_type           | exclusive                            |
| routes                |                                      |
| status                | ACTIVE                               |
| tenant_id             | 40e97bec2b06462098b241f04a224167     |
+-----------------------+--------------------------------------+
```

Then, we proceed to add the internal interfaces:

```
# neutron router-interface-add Tenant1-LR-Exclusive1 Tenant1-LS1_Net
Added interface 6c784dae-38f7-4025-85ac-e487987c7f35 to router Tenant1-LR-Exclusive1.

# neutron router-interface-add Tenant1-LR-Exclusive1 Tenant1-LS2_Net
Added interface eb367864-b5bd-48c3-a2fb-481faadbab01 to router Tenant1-LR-Exclusive1.
```

Next, connect the router to the appropriate External network:

```
# neutron router-gateway-set Tenant1-LR-Exclusive1 External-101 Set gateway for
router Tenant1-LR-Exclusive1
```

Create the Load Balancer:

```
root@controller01:~# neutron subnet-list
+--------------------------------------+--------------------+---------------+--------------------------------------------------+
| id                                   | name               | cidr          | allocation_pools                                 |
+--------------------------------------+--------------------+---------------+--------------------------------------------------+
| 37b57796-f919-461d-bdc7-5057eabcbf8f | Provider-VXLAN-Net1 | 10.112.1.0/24 | {"start": "10.112.1.101", "end": "10.112.1.200"} |
| a8175a61-9162-4253-971d-9e675aba3cbf | Tenant1-LS1_Net    | 10.1.1.0/24   | {"start": "10.1.1.2", "end": "10.1.1.254"}       |
| d7eb0562-95d9-42ab-99b7-cfb40519b45c | Tenant1-LS2_Net    | 10.1.2.0/24   | {"start": "10.1.2.2", "end": "10.1.2.254"}       |
+--------------------------------------+--------------------+---------------+--------------------------------------------------+
```

Created a new Load Balancer:

```
# neutron lbaas-loadbalancer-create --name Tenant1-LB1 Tenant1-LS1_Net
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| admin_state_up      | True                                 |
| description         |                                      |
| id                  | 142a8f47-47fb-40c8-837a-8c057518783a |
| listeners           |                                      |
| name                | Tenant1-LB1                          |
| operating_status    | ONLINE                               |
| pools               |                                      |
| provider            | vmwareedge                           |
| provisioning_status | ACTIVE                               |
| tenant_id           | 971c3aa8f8da4e80884894dcf244a939     |
| vip_address         | 10.1.1.115                           |
| vip_port_id         | 2a81c548-b8ad-4471-806e-ea44b7c8883d |
| vip_subnet_id       | 75b96d2c-1926-4910-8394-abce2a975b2d |
+---------------------+--------------------------------------+
```

Create a Virtual IP (VIP):

```
# neutron lbaas-listener-create --loadbalancer Tenant1-LB1 --protocol TCP --protocol-port 80 --name
WebListener1
Created a new listener:
+---------------------------+------------------------------------------------+
| Field                     | Value                                          |
+---------------------------+------------------------------------------------+
| admin_state_up            | True                                           |
| connection_limit          | -1                                             |
| default_pool_id           |                                                |
| default_tls_container_ref |                                                |
| description               |                                                |
| id                        | 796d978d-956c-4402-ad54-a7c724f40167           |
| loadbalancers             | {"id": "142a8f47-47fb-40c8-837a-8c057518783a"} |
| name                      | WebListener1                                   |
| protocol                  | TCP                                            |
| protocol_port             | 80                                             |
| sni_container_refs        |                                                |
| tenant_id                 | 971c3aa8f8da4e80884894dcf244a939               |
```

+---------------------------+--+

Create a Pool:

```
# neutron lbaas-pool-create --lb-algorithm ROUND_ROBIN --listener WebListener1 --protocol TCP --name
WebPool1

Created a new pool:

+---------------------+------------------------------------------------+

| Field               | Value                                          |

+---------------------+------------------------------------------------+

| admin_state_up      | True                                           |

| description         |                                                |

| healthmonitor_id    |                                                |

| id                  | edad563d-0576-4cc0-92ef-2c7ef3e6e1f1           |

| lb_algorithm        | ROUND_ROBIN                                    |

| listeners           | {"id": "796d978d-956c-4402-ad54-a7c724f40167"} |

| loadbalancers       | {"id": "142a8f47-47fb-40c8-837a-8c057518783a"} |

| members             |                                                |

| name                | WebPool1                                       |

| protocol            | TCP                                            |

| session_persistence |                                                |

| tenant_id           | 971c3aa8f8da4e80884894dcf244a939               |

+---------------------+------------------------------------------------+
```

Populate the pool with the IP addresses of the instances and specify the protocol:

```
# neutron lbaas-member-create --subnet Tenant1-LS1_Net --address 10.1.1.112 --protocol-port 80 WebPool1

Created a new member:

+----------------+--------------------------------------+

| Field          | Value                                |

+----------------+--------------------------------------+

| address        | 10.1.1.112                           |

| admin_state_up | True                                 |

| id             | 60154945-a70d-4b6c-baa7-bf8f9bdf6d68 |

| name           |                                      |

| protocol_port  | 80                                   |

| subnet_id      | 75b96d2c-1926-4910-8394-abce2a975b2d |

| tenant_id      | 971c3aa8f8da4e80884894dcf244a939     |

| weight         | 1                                    |

+----------------+--------------------------------------+
```

Create a Health Monitor and associate it with the pool:

```
# neutron lbaas-healthmonitor-create --delay 5 --max-retries 2 --timeout 10 --type HTTP --pool
WebPool1
--name HC_HTTP1

Created a new healthmonitor:

+----------------+-----------------------------------------------+

| Field          | Value                                         |

+----------------+-----------------------------------------------+

| admin_state_up | True                                          |

| delay          | 5                                             |

| expected_codes | 200                                           |

| http_method    | GET                                           |

| id             | f5a7c18c-926d-449f-90b9-07f1d7cb578c          |

| max_retries    | 2                                             |

| name           | HC_HTTP1                                      |

| pools          | {"id": "edad563d-0576-4cc0-92ef-2c7ef3e6e1f1"} |

| tenant_id      | 971c3aa8f8da4e80884894dcf244a939              |

| timeout        | 10                                            |

| type           | HTTP                                          |

| url_path       | /                                             |

+----------------+----------------------------------------------+
```

IMPORTANT: Neutron Security Groups may need to be updated to allow the traffic for Health Check into the Instances.

Finally, if NAT is being used, configure the Floating IP for the VIP:

```
# neutron floatingip-list

+--------------------------------------+------------------+---------------------+---------+

| id                                   | fixed_ip_address | floating_ip_address | port_id |

+--------------------------------------+------------------+---------------------+---------+

| 5024e099-c4ae-4557-9474-933aa4ed4fab |                  | 20.20.20.107        |         |

+--------------------------------------+------------------+---------------------+---------+


# neutron port-list | grep loadbalancer

| 2a81c548-b8ad-4471-806e-ea44b7c8883d | loadbalancer-142a8f47-47fb-40c8-837a-8c057518783a |
fa:16:3e:2d:32:44 | {"subnet_id": "75b96d2c-1926-4910-8394-abce2a975b2d", "ip_address":
"10.1.1.115"} |



# neutron floatingip-associate 5024e099-c4ae-4557-9474-933aa4ed4fab 2a81c548-b8ad-4471-806e-

ea44b7c8883d Associated floating IP 5024e099-c4ae-4557-9474-933aa4ed4fab
```

4.9.5 Firewall-as-a-Service (CLI)

To list firewalls, firewall_policies, firewall_rules:

```
# neutron firewall-list
# neutron firewall-policy-list
# neutron firewall-rule-list
```

Create firewall rule:

```
# neutron firewall-rule-create --protocol tcp --destination-port 80 --action allow Created a new
firewall_rule:
+------------------------+--------------------------------------+
| Field                  | Value                                |
+------------------------+--------------------------------------+
| action                 | allow                                |
| description            |                                      |
| destination_ip_address |                                      |
| destination_port       | 80                                   |
| enabled                | True                                 |
| firewall_policy_id     |                                      |
| id                     | 1283a548-9ca8-4a7b-a187-fc21c7fefe8e |
| ip_version             | 4                                    |
| name                   |                                      |
| position               |                                      |
| protocol               | tcp                                  |
| shared                 | False                                |
| source_ip_address      |                                      |
| source_port            |                                      |
| tenant_id              | baaaf4da44874e3f82ff93beba64117e     |
+------------------------+--------------------------------------+
```

Create firewall policy with rules:

```
# neutron firewall-policy-create --firewall-rules "1283a548-9ca8-4a7b-a187-fc21c7fefe8e ef9fe8d1-
1d79-485b-9d90-d1dd4bf228b5" test-policy
Created a new firewall_policy:
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| audited        | False                                |
| description    |                                      |
| firewall_rules | 1283a548-9ca8-4a7b-a187-fc21c7fefe8e |
|                | ef9fe8d1-1d79-485b-9d90-d1dd4bf228b5 |
| id             | 257f0a59-5b16-486b-aae2-b57c60e2053f |
| name           | test-policy                          |
| shared         | False                                |
| tenant_id      | baaaf4da44874e3f82ff93beba64117e     |
+----------------+--------------------------------------+
```

Create the firewall with the policy association:

```
# neutron firewall-create 257f0a59-5b16-486b-aae2-b57c60e2053f Created a new firewall:
+--------------------+--------------------------------------+
| Field              | Value                                |
+--------------------+--------------------------------------+
| admin_state_up     | True                                 |
| description        |                                      |
| firewall_policy_id | 257f0a59-5b16-486b-aae2-b57c60e2053f |
| id                 | 28530399-d8ee-4700-9685-ee5d645f4d59 |
| name               |                                      |
| status             | PENDING_CREATE                       |
| tenant_id          | baaaf4da44874e3f82ff93beba64117e     |
+--------------------+--------------------------------------+
```

Check that the firewall is in ACTIVE state before the next operation can be performed on the firewall:

```
# neutron firewall-show 28530399-d8ee-4700-9685-ee5d645f4d59
+--------------------+--------------------------------------+
| Field              | Value                                |
+--------------------+--------------------------------------+
| admin_state_up     | True                                 |
| description        |                                      |
| firewall_policy_id | 257f0a59-5b16-486b-aae2-b57c60e2053f |
| id                 | 28530399-d8ee-4700-9685-ee5d645f4d59 |
| name               |                                      |
| status             | ACTIVE                               |
| tenant_id          | baaaf4da44874e3f82ff93beba64117e     |
+--------------------+--------------------------------------+
```

Delete the firewall:

```
# neutron firewall-delete 28530399-d8ee-4700-9685-ee5d645f4d59 Deleted firewall: 28530399-d8ee-4700-9685-
ee5d645f4d59
```

4.10 NSX-T Neutron Integration

VMware NSX-T is designed to address emerging application frameworks and architectures that have heterogeneous endpoints and technology stacks. In addition to vSphere, these environments may also include other hypervisors, containers, bare metal, and public clouds. IT and development teams can use NSX-T to choose the technologies best suited for their particular applications. NSX-T is also designed for management, operations, and consumption by development organizations in addition for IT.

NSX-T is built to support the increasingly heterogeneous and app-driven realities of digital business environments. As IT's role in this landscape becomes more complex, NSX-T provides an approach to support, manage, and secure workloads running inside multiple clouds, containers, and hypervisors, distributed across microservice architectures. By enabling developers to consume networking resources through APIs or natively by using Openstack NSX-T Neutron Plug-in, NSX-T lets them treat infrastructure-as-code (IaC) within the context of their build and CI/CD pipeline, and get more work done, faster.

NSX-T Key Takeaways

- NSX-T architecture is designed for heterogeneity and choice, and will evolve to provide networking and security services to workloads running anywhere.
- NSX-T provides a rich set of networking and security services with enterprise-grade quality and scale for workloads running on ESXi or KVM hosts and hypervisors.

- NSX-T provides a single pane of glass and tools to configure, monitor, and troubleshoot networking and security on heterogeneous infrastructures.
- The NSX-T REST API is powerful and comes with an OpenAPI specification that enables various language bindings and plug-ins to integrate with the CMP of your choice, including OpenStack.
- NSX-T OpenStack plug-in is available for developers who want to build and maintain multi-tenant developer clouds with advanced services.
- NSX-T edge nodes provide extreme gateway and services performance using innovation in x86 forwarding and Intel DPDK technology. NSX-T is vCenter agnostic and can be consumed by vCenter, other compute managers, as well as PaaS platforms.

Note: Although NSX-T supports multiple hypervisors, when using VMware Integrated Openstack, NSX-T will be supported and used for VMware vSphere.

NSX-T offers to Openstack

- L2 Services with Overlay, DHCP, support for overlapping IP addresses, and an L2 Gateway for Overlay or VLAN Bridging.
- L3 Services with Distributed Routing, External Network, Floating IP, no-NAT support and integrated dynamic routing for (floating IP and/or tenant subnets are automatically advertised to physical world without config change on the physical routers).
- Security Groups leveraging Stateful Distributed Firewall on NSX-T.
- LBaaS v 2.0 and QoS.
- Metadata Services that allow instances to access instance-specific metadata, such as host name, SSH key, DNS info, and so on. Metadata Proxy is offered by the NSX-T platform and not by Neutron.
- RBAC for Networks. Different Tenants share same Networks or specific External Networks are for specific Tenants only

4.10.1 NSX-T Features

Feature	**NSX-T Neutron Plug-In Support**
Overlay Provider Networks	Yes
Overlay Tenant Networks	Yes
Metadata Proxy Service	Yes
DHCP Server	Yes
Neutron Router - Distributed	Yes
Floating IP Support	Yes
No-NAT Neutron Routers	Yes
Neutron Security Groups using Stateful Firewall	Yes
Port Security	Yes
Neutron L2 Gateway	Yes
Admin Utility (Consistency Check, Cleanup)	Yes
QoS	Yes
RBAC for Networks	Yes

Table 4.3: NXS-T Features

4.10.2 NSX-T Architecture

The NSX-T architecture (Figure 4.13) has built-in separation of the data plane, control plane, and management plane. This separation delivers multiple benefits, including scalability, performance, resilience, and heterogeneity.

- Management plane: The NSX-T management plane is designed from the ground up with advanced clustering technology, which the platform can use to process large-scale concurrent API requests.
- Control plane: The NSX-T control plane keeps track of the real-time virtual networking and security state of the system. NSX-T control plane separates the control plane into a central clustered control plane (CCP) and a local control plane (LCP). This simplifies the job of the CCP significantly and enables the platform to extend and scale for heterogeneous endpoints.
- Data plane: The NSX-T data plane introduces a host switch (rather than relying on the vSwitch), which decouples it from the compute manager and normalizes networking connectivity. All create, read, update, and delete (CRUD) operations are performed through the NSX-T Manager

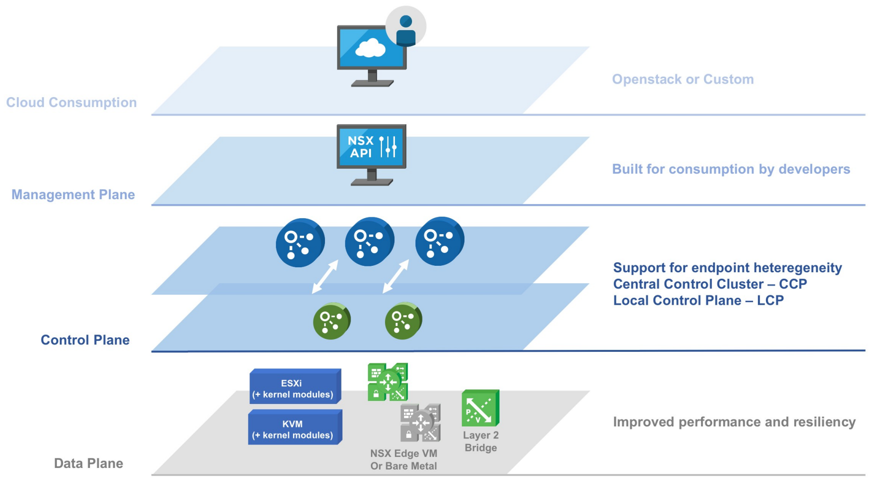


Figure 4.13: NSX-T Architecture

Management Plane

The NSX-T management plane provides secure concurrent entry points to the system via a full-featured REST API or a browser-based, high-performance, and modern HTML5 Graphical User Interface. The management plane is designed from the ground up for performance to process large-scale concurrent API calls from cloud management platforms (CMPs). The system is able to be integrated into any CMP and is fully supported OpenStack Neutron plug-in. As the system scales, the management plane has the ability to scale out using advanced clustering technology.

Control Plane

The NSX-T control plane encompasses a clustered control-plane (CCP) running on controller nodes and a localized control plane (LCP) on compute endpoints.

- The NSX-managed compute endpoints are known as transport nodes.
- The CCP computes and disseminates the ephemeral runtime state based on configuration from the management plane and topology information reported by the data plane elements.
- The LCP runs on the compute endpoints. It computes the local ephemeral runtime state for the endpoint based on updates from CCP and local data plane information. The LCP pushes stateless configuration to forwarding engines in the data-plane as well as reports the information back to the CCP. This simplifies the job of the CCP significantly. As a result, the platform can scale to thousands of heterogeneous endpoints (hypervisor, container host, bare metal, or public cloud).

Data Plane

NSX-T includes a number of capabilities in the data plane that improve the performance and resiliency of the platform.

The NSX-T data plane can be enabled on ESXi, KVM hypervisors, and appliances providing gateway functionality called Edge Nodes ensuring a rich set of networking and security services. Some of these services include logical switching, distributed logical routing, distributed firewalls, and network services such as NAT, DHCP Relay, DHCP Server and MetaData Proxy functionality. Data plane forwarding and transformation decisions are made based on the local tables populated by the control plane.

4.11 NSX-T Supported Topologies and Integration

The NSX-T Neutron plug-in supports the following topologies (Table 4.4):

Comment [XG1]: format

Comment [AG2R1]:

Use Case		Comments
L2 overlay only, no L3 services and no security	L2 Overlay only based on Geneve encapsulation	Overlays are bridged with VLANs. No Distributed Firewall policies
L2 overlay and security only, no L3 services	L2 Overlay and Micro-segmentation only	Overlays are bridged with VLANs. Security Groups leverage Distributed Firewall policies
L2/L3 overlays, no NA	Enterprise customers that don't need overlapping IP addresses	No overlapping IPs allowed. Very efficient. Preferred Enterprise model
L2/L3 overlays, NAT	Enterprise customers that need overlapping IPs	Overlapping IPs allowed. Very efficient. Preferred Cloud Provider and Service Provider model

Table 4.4: NXS-T Features

L2 Overlay only

The L2 overlay (Figure 4.14) only topology is supported and recommended for laboratory environments and tiny deployments.

NSX-T only performs L2 tasks, and all others features such as routing, security, load balancing and others need to be pre-provisioned or created manually in the physical infrastructure.

For every new Network created in OpenStack, a Logical Switch is created in NSX-T, and an L2 Bridge is established in NSX-T for a specific VLAN.

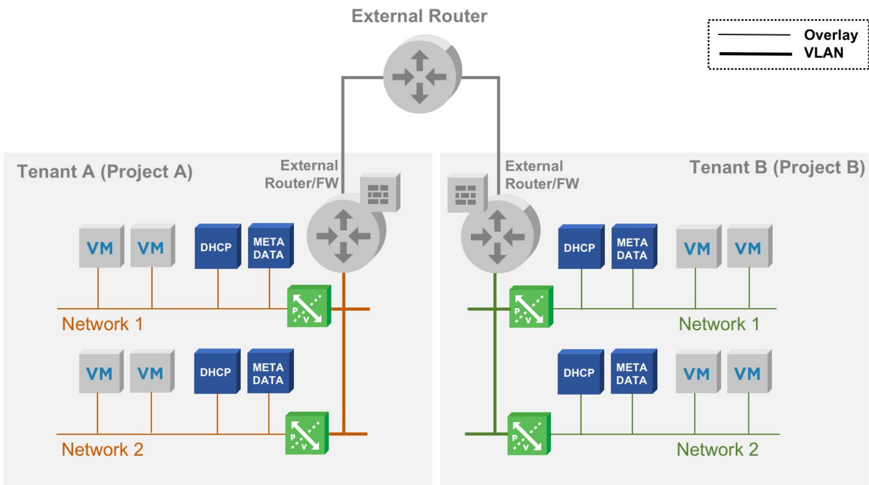


Figure 4.13: L2 Overlay

L2 Overlay and Security only

This following topology is supported and recommended for non-production deployments.

NSX-T only performs L2 tasks and security, and all others features like routing, load balancing and others need to be either pre-provisioned or created manually in the physical infrastructure.

For every new Network created in OpenStack, a Logical Switch is created in NSX-T and a L2 Bridge is established in NSX-T for a specific VLAN (Figure 4.15). Regarding Security, NSX-T plug-in translates all Security Groups created on Neutron to NSX-T Distributed Firewall rules and NSX-T NSGroups. You can dig into this in the chapter regarding NSX-T Security.

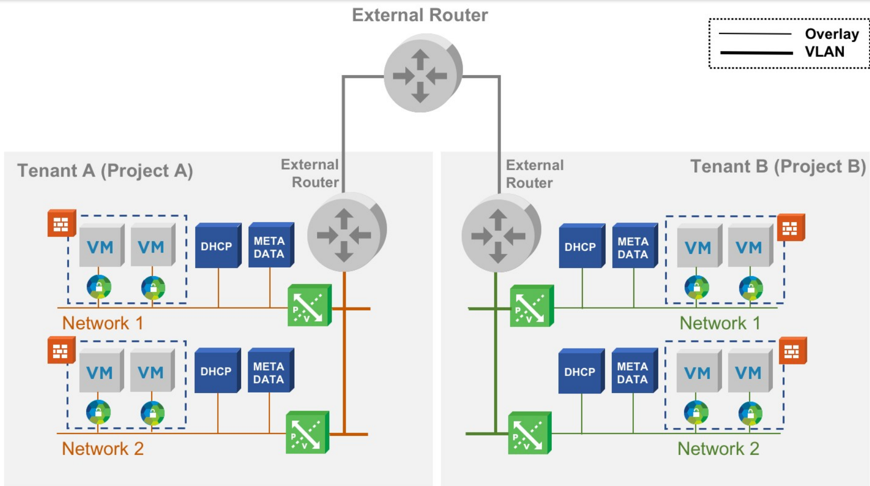


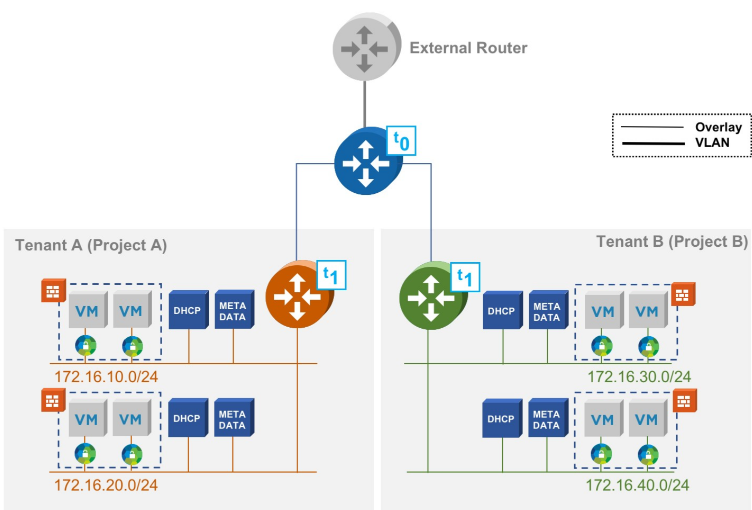
Figure 4.15: L2 Overlay and Security

L2 Overlays with Security and L3 Tenant Routing - no NAT Deployment

This following topology is supported and recommended for most of Enterprise deployments.

NSX-T performs L2, security, routing, load balancing, and other tasks, and no pre-provisioning or manual creation is required in the physical infrastructure. In this topology overlapping IP cannot be used (Figure 4.16).

LBaaS is optional and delivered by NSX-T



vmware®

Figure 4.16: L2 Overlays with Security and L3 Tenant Routing

L2/L3 Overlays with Security - NAT Deployment

This following topology (Figure 4.17) is supported and recommended for most of Cloud and Service Providers deployments.

NSX-T is performing L2 functionalities, security, routing, load balancing and others and there is no need to be pre-provisioned or created manually in the physical infrastructure. Regarding IP overlapping, this topology does support it and NAT is used to tenants/projects communicate between each other or to the external world. LBaaS is optional and delivered by NSX-T Load balancer service.

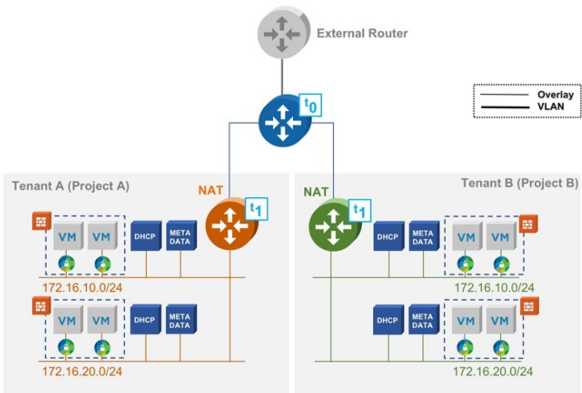


Figure 4.17: L2/L3 Overlays with Security - NAT Deployment

4.12 NSX-T Security Microsegmentation

Distributed Firewall, Microsegmentation and Security Groups

NSX-T introduces support for distributed firewall (DFW) functionality for workloads running on both ESXi and KVM hosts/hypervisors (Figure 4.18). The NSX DFW provides the capability to enforce firewalling functionality directly at the workload vNIC layer, providing an optimal micro-segmented environment. Both stateful and stateless firewall rules are supported. Firewall rule provisioning will be proliferated in the system by the CCP for more redundancy and scale. Users will have the ability to craft their micro-segmentation model based on IP address sets (L3), MAC identifier sets (L2), logical switches, logical ports or advanced security policies based on security groups. CMPs and PaaS platforms will be able to provision policies on NSX-managed workloads using the advanced REST API or pluggable modules. Operators will be able to view provisioned policies in a single pane of glass in the NSX Manager GUI or via the API.

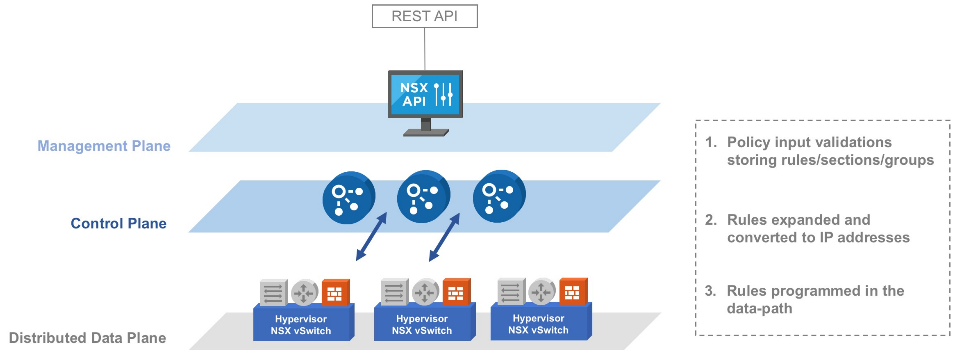


Figure 4.18: Distributed Firewall, Microsegmentation and Security Groups

Neutron Security Groups have historically implemented Linux iptables when running on KVM, or Open vSwitch stateless matches to filter traffic at the hypervisor level. Both approaches have proven inadequate and there is serious work underway aimed at addressing these issues (VMware is a contributor to these efforts).

When using NSX, we deploy a stateful firewall on each and every hypervisor host. That means that every hypervisor will protect the microcosm of virtual machines that it hosts, providing the notion of a distributed dataplane. We call this a distributed firewall, or DFW. The NSX DFW enables granular security controls at the VM vNIC level. When using Neutron Security Groups, the NSX DFW is configured, via the plugin integration. Neutron Security Groups are mapped to instances, meaning the NSX DFW will protect the VM unit.

Running an actual stateful firewall on each hypervisor within your OpenStack cloud has the following benefits:

- The attributes of micro-segmentation are readily available to the OpenStack admintrators and tenants.
- Compliance requirements (like PCI, or HIPAA) can be met, without sacrificing the openness of OpenStack as your IaaS layer.

The NSX firewall scales as your hypervisor footprint grows. The mere act of increasing your compute capacity due to the organic growth of your business, automatically means you are also adding security and compliance to your virtual infrastructure.

Similar to NSX-v, NSX-T implements for every Neutron Security Group a NSX-T NSGroup. Membership criteria (Figure 4.19) for the NSGroup is based in the Logical Switch Port Tag on NSX-T. Under NSX-T Firewall Rules, a section is created per tenant (project) and NSGroups are used as Sources and/or Destinations. Those rules are applied only for the NSGroups related to the rules creating a true micro-segmented architecture per tenant. At NSX-T Firewall Rules, there is a Default Section that are applied to all NSX-T Logical Switch Ports that have an Implicit Drop in the end. At this last section, DHCP rules are allowed for NSX-T DHCP Server to work as well.

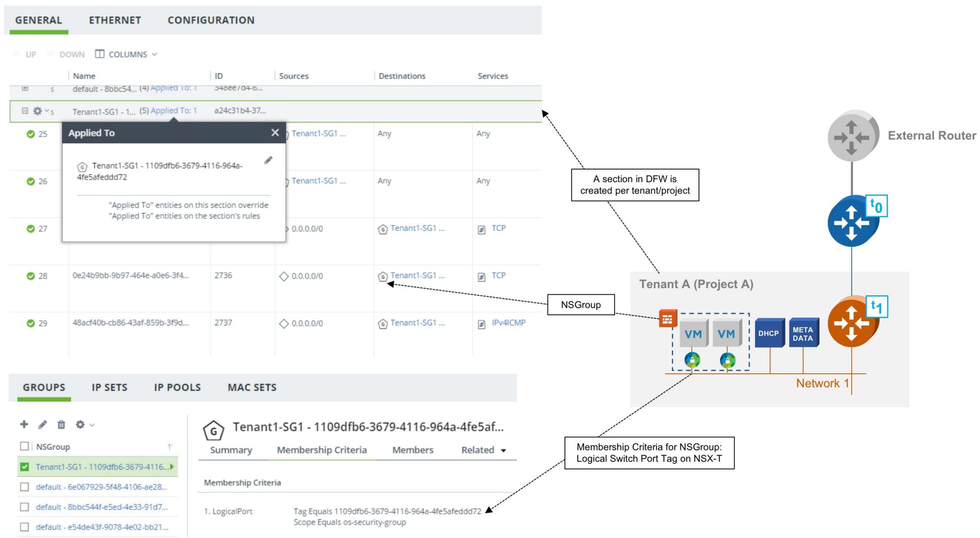


Figure 4.19: Membership Criteria Rules

Logging

NSX-T Distributed Firewall per default configure firewall rules with logging. There are 3 options for logging:

1. Enable logging for the last "Block-All" rule in the last section
2. Enable logging for all Tenants Security Groups (NSX-T NSGroups) allow rules
3. Enable logging for Specific Tenants Security Groups (NSX-T NSGroups) allow rules

Port-Security

Port-Security protects against users changing the IP address of the instance creating a table correlating MAC and IP. Port-

Security is enabled by default but can be disabled.

For each instance created, one NSX-T Address Binding is created on the related Logical Switch Port and a SpoofGuard Profile is associated with this port.

Depending on the use case, it's possible to associate multiple IP Addresses to a instance.

Provider Security Group

As an admin, starting at Newton release, you have the ability to create Provider Security Groups. The idea is to block specific traffic to Tenant VMs.

For example, as an admin, you can deny SSH traffic from outside to the VMs inside the tenant.

At NSX-T, in Firewall Rules, a section is created on top of all Tenant sections, for Provider Security Groups and these deny rules have precedence of Tenant Security Groups created even if the Tenant decided to allow the traffic with specific rules.

This is a powerful tool provided by Neutron and NSX-T that can be leverage to add more security and control to the Openstack environment.

4.13 NSX-T Edge Integration

4.13.1 Multi-Tenant Routing Protocol

NSX-T supports a multi-tiered routing model with logical separation between the provider router function (known in NSX as a Tier0 router) and the tenant router function (known in NSX as a Tier1 router) (Figure 4.19)

- The Tier0 logical router, a routing layer that can be controlled by the cloud provider, is capable of integrating with the physical infrastructure.
- The Tier1 logical router, a routing layer provided to cloud tenants, can be provisioned by GUI/API/CMPs for each tenant and can be attached to the Tier0 routers.

NSX-T instantiates distributed routers on the hypervisors for optimal multi-tier East-West routing.

Connectivity between the tenant routers and the provider routers is managed by the NSX-T management plane and the NSX-T control plane, and there is no requirement to run complex routing protocols or control VM appliances to disseminate connectivity information.

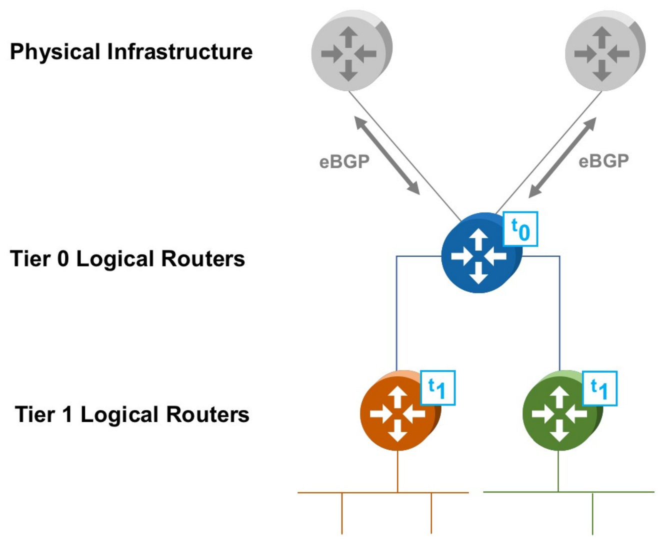


Figure 4.19: Multi-Tenant Routing Model

4.13.2 High Performance Edge Nodes

NSX-T enables high-performance Edge nodes to provide Tier0 connectivity to the physical infrastructure. Edge nodes also provide capacity to run tenant-based or provider-based centralized network services such as NAT.
Edge nodes can be deployed as both VM or bare-metal form factor. These nodes leverage improvements in x86 forwarding technology such as Intel's DPDK Libraries (Data Plane Development ToolKit) for faster packet processing. For more

information, go to http://www.intel.com/go/DPDK. These improvements enable line-rate gateway performance at small packet sizes. The Edge Nodes can further scale out to provide several Gbps of throughput.

Data Plane Scale Out

The Edge Nodes can be grouped into a pool of capacity (known as Edge Cluster) to provide scale out, redundant and high-throughput gateway functionality for the logical networks – see Figure 4.20. Scale out from the logical networks to the Edge Nodes is achieved using equal cost multi path (ECMP). The Edge Nodes can also host stateful services and provide scale and redundancy to run these services at scale. The active standby model is also available to provide flexibility and choice.

Multiple tenant-based or provider-based services can run as individual contexts on any node in the Edge Cluster as shown in the figure.

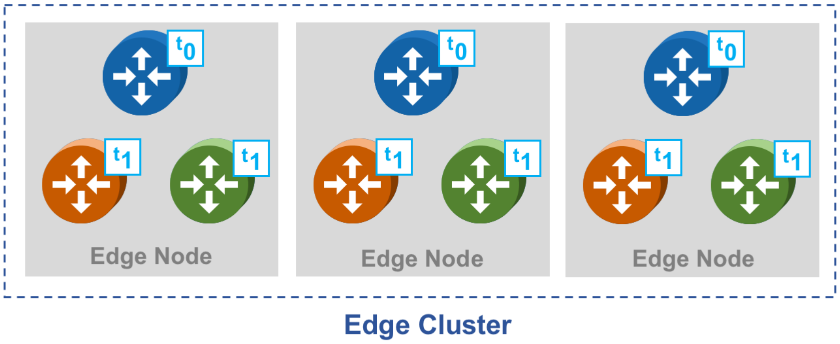


Figure 4.20: Edhe Cluster

Routing and Convergence

NSX-T supports protocol-standards-based static and dynamic routing to peer with the physical networking world. The Edge Node is designed to run a large number of tenant routing contexts within a virtual or bare-metal appliance and leverage advanced routing functionality, while also supporting BGP routing and attribute-manipulation capabilities using route maps and prefix lists.

For faster detection of link or node failures and fast routing convergence, bidirectional forwarding detection (BFD) towards the physical space is available. BFD is also used internally for fault detection and fast failover of nodes.

Neutron Integration with NSX-T Tier0 and Tier1 Routers

NSX-T Tier0 and Tier1 router provides an intelligent model for Neutron. NSX-T Tier0 and Tier1 auto-plumbing mechanism creates an easy way to handle network traffic. Both Tier0 and Tier1 routers can be distributed in the kernel of each hypervisor and also be in the Edge Node as part of an Edge Cluster. This distributed, automated, and flexible model provides the most efficient way for packets to flow inside or outside tenants in OpenStack.

NSX-T Tier0 router can be connect using BGP (recommended) or static routes with physical routers. If you use static routes, for every new external network created, you need to manually add the external network to the Tier0 router peering with the physical routers. (Figure 4.21)

Whenever a external network is created in OpenStack, nothing happens in NSX-T at this moment. External networks must be preconfigured manually in NSX-T to be advertised to the external world in one of two ways:

- Tier0 selectively advertise Tier1 NAT IPs (Example: Tenant A has external access, Tenant B, does not require external access, Tenants A and B need to communicate between themselves. Tier0 router advertises Tenant A external IP, but not Tenant B external IP)

- Tier0 advertise the whole external network

Changes occur only when a Neutron router is created (translated as a NSX-T Tier1 router with an uplink interface at the external network with a SNAT rule) or when a floating IP is created (translated as an SNAT rule in the NSX-T Tier1 router and DNAT rule for the internal instance).
Whenever a NSX-T Tier1 router is created, it is automatically connected to the NSX-T Tier0 router.

Note: In a no-NAT topology, SNAT and DNAT for Tier1 routers are not configured. Also, the Tier0 router advertises connected networks from the Tier1 routers to the external world.

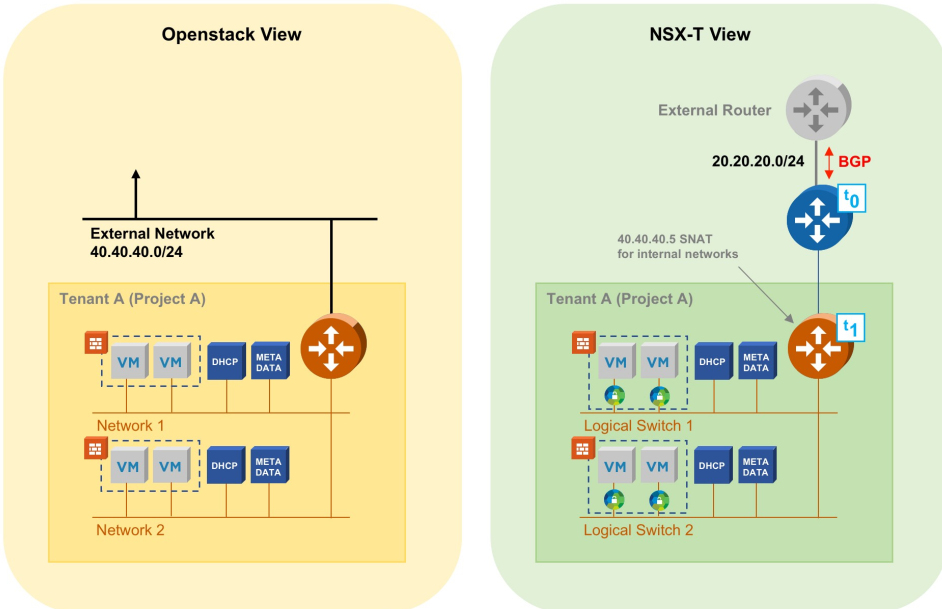


Figure 4.21: Neutron Integration with NSX-T Tier0 and Tier1 Routers

DHCP Services

Starting with Newton, NSX-T natively offers DHCP server support for any Openstack distribution, including VIO.

DHCP servers are located on the NSX-T Edge Node. For High Availability (HA), DHCP service is configured on the Edge Cluster, offering HA automatically for VIO implementation.

For each Neutron OpenStack Network created, one DHCP Instance is created and connected on the NSX-T Logical Switch associated with that network.

Metadata Services

Starting with Newton, NSX-T natively offers Metadata Proxy Service for any OpenStack distribution, including VIO.

For each network (NSX-T Logical Switch), the NSX-T Plug-in creates one Logical Switch Port for Metadata Proxy in the Edge Nodes and Edge Nodes forward the traffic to the Metadata Server. (Figure 4.22)

Note: Edge Nodes must have IP connectivity to Metadata Server via Management Interface

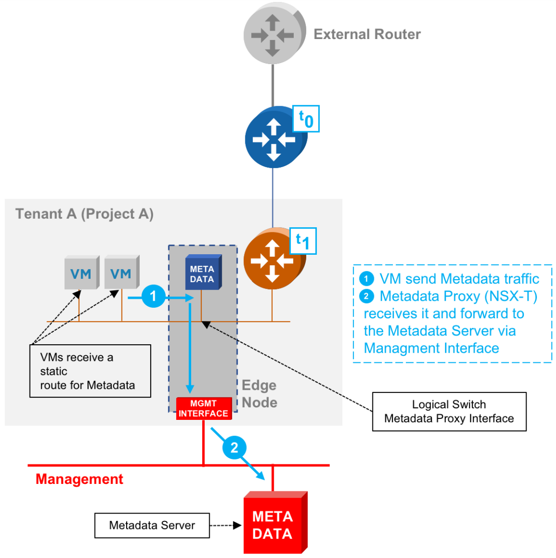


Figure 4.22: Metadata Services

4.13.3 NSX-T LBaaS Integration

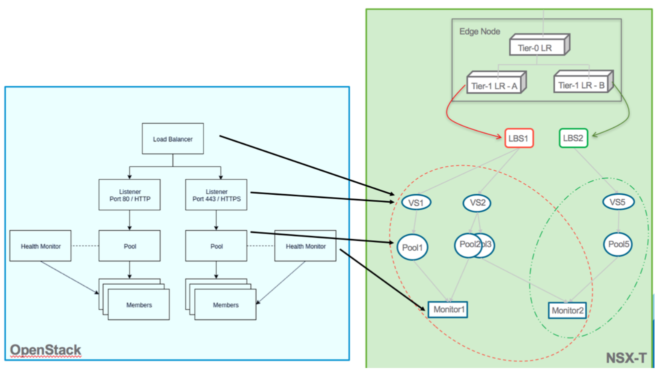


Figure 4.23: NSX-T LBaaS Services

Load Balancer Service (LBS)
NSX-T Edge Node provides the LBaaS functionalities. More specifically, load balancing is one of the services running within the context of a Service Router (SR). In the initial NSX-T release, Load Balancer Services (LBS) maps to a Tier-1 LR (logical router) only.. As an NSX edge node can host multiple LRs, there can be multiple LBSs running on a

single edge node, attached to different LRs. One LBS can be attached to a single Tier-1 SR, multiple neutron Loadbalancer can share the same LBS if they are on the same Tier-1 router. When an LBS is attached to a tier-1 router, NSX Manager creates a separate Linux namespace for that SR, if one does not already exist. It then starts NGINX within that namespace.

Virtual Server (VS)

A virtual server is identified by IP address, port, and protocol. All client connections are received by the virtual server and distributed among the backend servers. A virtual server supports either TCP or UDP protocol, but not both. If support for both TCP and UDP protocols is desired for the same IP and port (e.g., DNS), then two virtual servers must be created, one for each protocol. With NSX-T load balancer service, the VIP is associated with virtual server attached to the LB service. A load balancer maps to a Load Balancer Service (LBS) on the NSX-T backend.

Pool

A server pool consists of one or more servers, also referred to as members that are similarly configured and are running the same application. Multiple pools can be bound to an L7 virtual server using LB Rules. A single pool can be bound to more than one virtual server. In such a case, server pool statistics, pool member statistics, etc. are maintained separately for each virtual server.

Load Balancing Algorithm

Load balancing algorithm chooses a server for each new connection by going through the list of servers in the pool. It is configured per pool. Currently, following load balancing algorithms are supported:

1. Round-Robin: selects a server in a round-robin fashion. Ignores member weights even if they are configured.
2. Weighted Round-Robin: selects a server in a weighted round-robin fashion. Default weight of 1 is used if weight is not configured for a member.
3. Least Connections: selects a server that currently has the least number of connections. Ignores member weights even if they are configured.
4. Source IP Hash: Consistent hash is performed on the source IP address of the incoming connection to select a server.

Round-Robin is the default Algorithm. L7 load balancing algorithms such as URL and Host-header based server/pool selection are supported only by LB Rules. LB Rules can be used to select either a server pool or a server directly. If a server pool is selected using LB Rules, then a server within the pool is selected using the load balancing algorithm configured for that pool.

High Availability (HA)

Load Balancer Service (LBS) is always deployed in an Active/Hot-Standby mode, with the active and standby SRs running on two different edge nodes of the edge cluster. Active-Active is not supported.

Monitor

Load balancers monitor the health of backend servers to ensure traffic is not blackholed. There are two types of health checks: active and passive. In case of active health checks, load balancer itself initiates new connections (or sends ICMP ping) to the servers periodically to check their health, completely independent of any data traffic. NSX-T is capable of supporting both Active / Passive health check. OpenStack does not support passive health check.

Scalability

To address varies customer performance and scalability requirements, different sizes of LBS are supported: LARGE, MEDIUM and SMALL. Performance and scalability details are outlined below:

	Small LB	Medium LB	Large LB

# Virtual Servers	10	100	1000
# Pool Members	30	300	3000

Table 4.5: NXS-T LBaaS performance properties

The number of LBS instances of each type that can be supported per edge and the performance and scalability details of each LBS size are listed below:

	Small LB	Medium LB	Large LB
Edge VM (4 vCPU, 8GB)	1	Not Supported	Not Supported
Edge VM (8 vCPU, 16GB)	4	1	Not Supported
Edge Bare Metal	100	10	1

Table 4.6: LBS instances per Edge Type

The default mapping between neutron flavor and NSX-T load balancer size are specified in the table below. If the user doesn't specify a flavor when creating a load balancer, the small size will be used by default.

OpenStack Network Flavor Name	NSX-T Load Balancer Service Size
small	SMALL
medium	MEDIUM
large	LARGE

Table 4.7: OperStack Flavor Mapping to NSX-T

As LBS is instantiated on the SR it is attached to, the actual edge nodes on which it is instantiated depends on the placement of SR itself. As load-balancer utilization increases, the need to rebalance the SR to edge node mapping may be required.

Interfaces with Other Components

Same SR can have LB, NAT and FW enabled. If all three features have rules that can match an incoming packet, then only LB rule hit will be honored, and the other two features will not be performed for that packet. As a general rule, features will act on a given flow with the precedence: LB, NAT, and FW. NSX-T platform does not support OpenStack Floating IP (NAT) on LB-VIP.

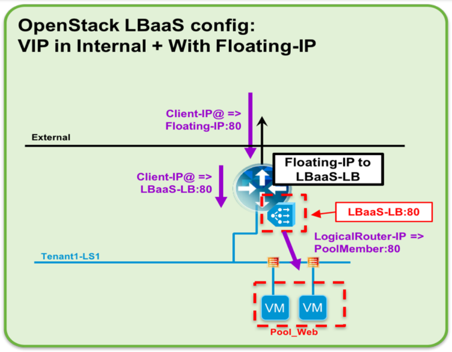


Figure 4.24: Internal VIP with NAT

in the event, floating IP is assigned to a LB-VIP (see figure 4.24), the Neutron NSX-T plugin will translate NSX-T LB-VIP to the OpenStack Floating-IP. "NAT + LB-VIP" configured in NSX-T (see figure below):

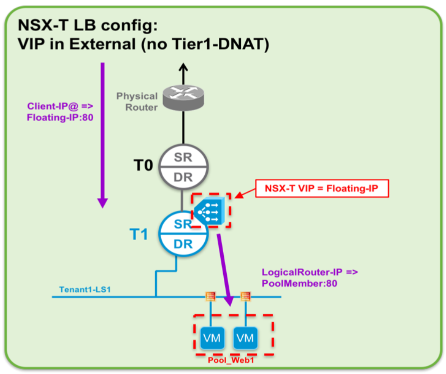


Figure 4.25: LB-VIP to Floating IP

A NSX-T logical router is required for LBaaS. Below topologies are note supported:

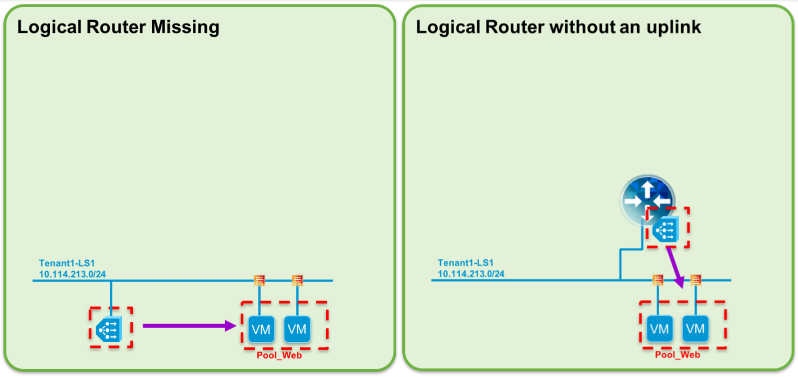


Figure 4.26: Unsupported Topologies

The LBaaS workflow includes the following capabilities, both inline and one-arm mode are supported. Make sure the load balancer is attached to a Tier-1 router for full functionality.

1. Create a Load Balancer.
2. Create a Listener.
3. Create a Pool.
4. Create Pool Members.
5. Create a Health Monitor.
6. Configure Security Groups to allow Health Monitor to work from Load Balancer to Pool Members.

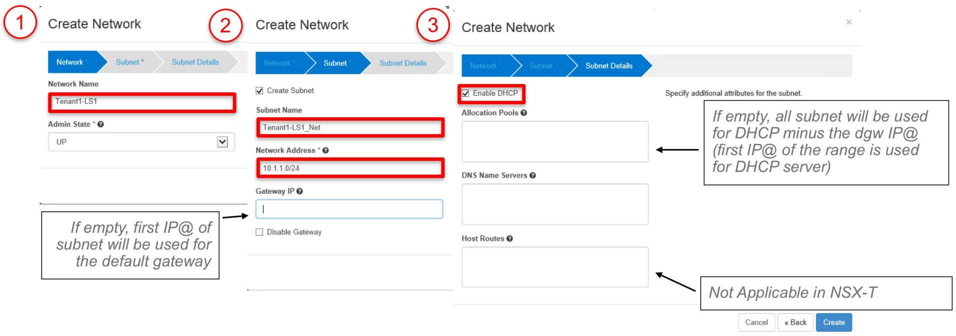
4.14 NSX-T Neutron End-user Workflow

The following section covers the specifics of the Neutron and NSX-T integration. We will describe, one by one, the Neutron services typically leveraged in an OpenStack Cloud and then show the corresponding NSX-T construct that gets provisioned or configured in the back end.
Whenever possible, we will show the Horizon workflow to provide said construct, but some operations are only available from the OpenStack CLI client, in which case we will show the necessary command (or commands) to provide that particular network service.

4.14.1 L2 Services - Switching

UI

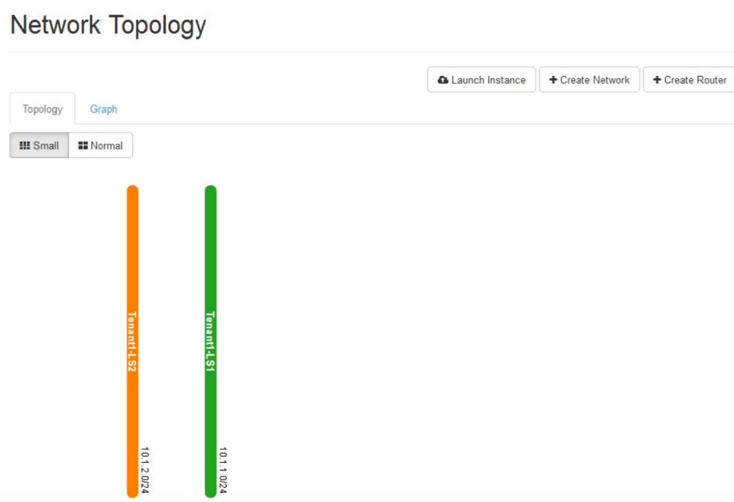
Under "Project - Network – Networks", Create Network



Capture 4.20: Neutron End-User Workflow

Verification

Under "Project - Network – Network Topology"



Capture 4.21: Network Topology

CLI

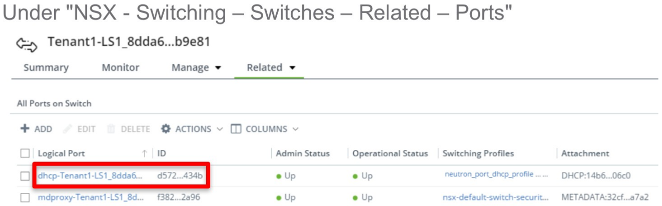
```
root@controller01:~# neutron net-create Tenant1-LS2 Created a new network:

+-----------------------+-------------------------------------+

| Field                 | Value                               |
```

```
+----------------------+-------------------------------------+
| admin_state_up       | True                                |
| id                   | 73232656-058e-418c-845d-e8f2dc03b378 |
| name                 | Tenant1-LS2                         |
| port_security_enabled | True                               |
| router:external      | False                               |
| shared               | False                               |
| status               | ACTIVE                              |
| subnets              |                                     |
| tenant_id            | 4d238862a193465985cde43b3f0c6500    |
+----------------------+-------------------------------------+
root@controller01:~# neutron subnet-create --name Tenant1-LS2_Net Tenant1-LS2 10.1.2.0/24
--dns-nameservers list=true 10.33.38.1 10.33.38.2 Created a new subnet:
+-------------------+------------------------------------------+
| Field             | Value                                    |
+-------------------+------------------------------------------+
| allocation_pools  | {"start": "10.1.2.2", "end": "10.1.2.254"} |
 | cidr             | 10.1.2.0/24                              |
 | dns_nameservers  | 10.33.38.1                               |
 |                  | 10.33.38.2                               |
 | enable_dhcp      | True                                     |
 | gateway_ip       | 10.1.2.1                                 |
 | host_routes      |                                          |
 | id               | 9734deb0-f073-436d-820f-d5c26bdd546c     |
 | ip_version       | 4                                        |
 | ipv6_address_mod |                                          |
 e
 | ipv6_ra_mode     |                                          |
 | name             | Tenant1-LS2_Net                          |
 | network_id       | 73232656-058e-418c-845d-e8f2dc03b378     |
 | subnetpool_id    |                                          |
 | tenant_id        | 4d238862a193465985cde43b3f0c6500         |
+----------------+----------------------------------+
```

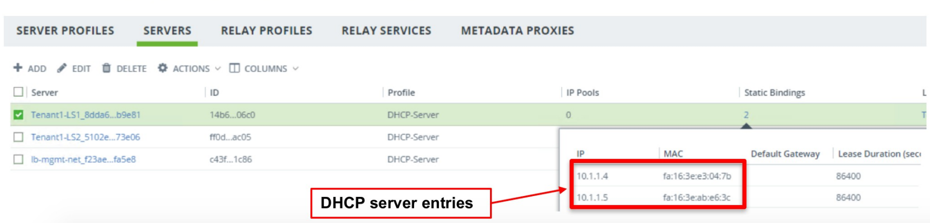
What happens in NSX-T?

- For each OpenStack Network created:
  2 – One DHCP instance is created and plugged on the LS

  Under "NSX - Switching – Switches – Related – Ports"



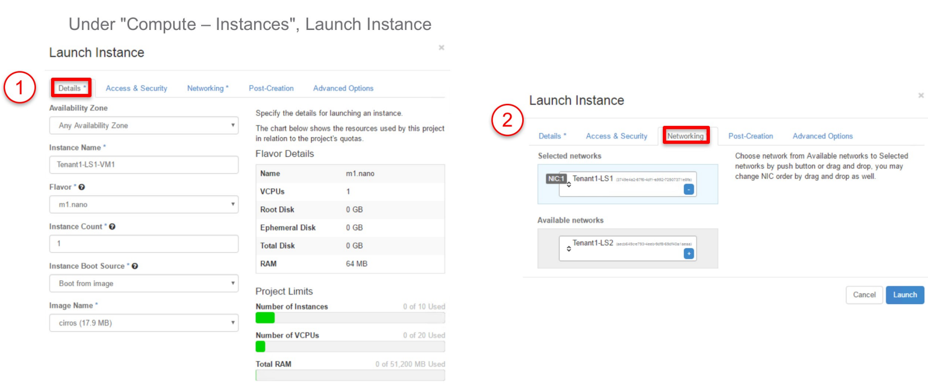
Under "NSX – DHCP – Servers"



Capture 4.22: Create DHCP Instance

4.14.2 L2 Services - Switch Port

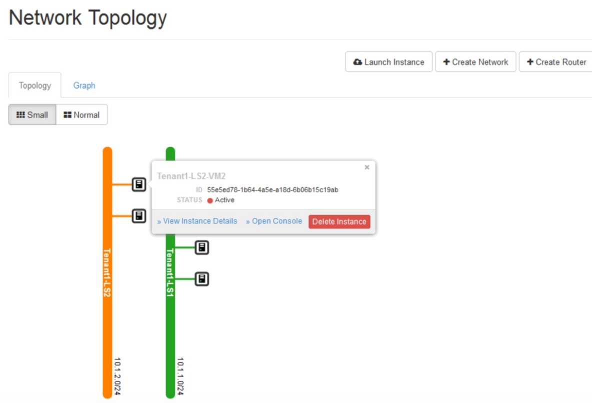
UI



Capture 4.23: L2 Services – Switch Port

Verification

Under "Project - Network – Network Topology"



Capture 4.24: Network Topology

CLI

```
root@controller01:~# nova boot --image cirros --flavor m1.nano --nic net-
id=03cb1772-d175-44b7-9e32-935f445610a0  Tenant1-LS1-VM2

+--------------------------------+--------------------------------------+
| Property                       | Value                                |
+--------------------------------+--------------------------------------+
 | OS-DCF:diskConfig             | MANUAL                               |
 | OS-EXT-AZ:availability_zone   |                                      |
 | OS-EXT-STS:power_state        | 0                                    |
 | OS-EXT-STS:task_state         | scheduling                           |
 | OS-EXT-STS:vm_state           | building                             |
| OS-SRV-USG:terminated_at       | -                                    |
| accessIPv4                     |                                      |
| accessIPv6                     |                                      |
| adminPass                      | 9zZiywWiVTm2                         |
| config_drive                   |                                      |
| created                        | 2016-11-23T16:37:42Z                 |
| description                    | -                                    |
| flavor                         | m1.tiny (1)                          |
| hostId                         |                                      |
| id                             | 4b5df5b4-ff68-42b1-8d94-c718a6c98e92 |
| image                          | cirros (c4a23f19-7957-4f13-932c-f6333d5b5da4) |
| key_name                       | -                                    |
```

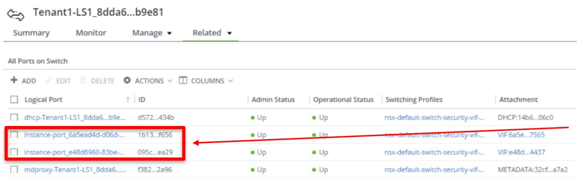
```
| locked                               | False                                          |
| metadata                             | {}                                             |
| name                                 | Tenant1-LS1-VM2                                |
| os-extended-volumes:volumes_attached  | []                                             |
| progress                             | 0                                              |
| security_groups                      | default                                        |
| status                               | BUILD                                          |
| tags                                 | []                                             |
| tenant_id                            | eee4ee94bcaa460a8d775c02d5ff60e9               |
| updated                              | 2016-11-23T16:37:42Z                           |
| user_id                              | 4a3454a70cf640d589327b857d174c07               |
+--------------------------------------+------------------------------------------------+
```

Verification

```
root@controller01:~# nova list
+--------------------------------------+-----------------+--------+------------+-------------+
| ID                                   | Name            | Status | Task State | Power State | Networks
                       |
+--------------------------------------+-----------------+--------+------------+-----+
 | 4b5df5b4-ff68-42b1-8d94-c718a6c98e92 | Tenant1-LS1-VM1 | ACTIVE | -          | Running     |
 Tenant1-LS1=10.1.1.4  |
 | f64c9bb3-5d72-49e9-955c-674128983b99 | Tenant1-LS1-VM2 | ACTIVE | -          | Running     |
 Tenant1-LS1=10.1.1.5  |
 | c8417bff-2a06-4d3b-80d5-31858fa2ec9e | Tenant1-LS2-VM1 | ACTIVE | -          | Running     |
 Tenant1-LS2=10.1.2.5  |
 | 55e5ed78-1b64-4a5e-a18d-6b06b15c19ab | Tenant1-LS2-VM2 | ACTIVE | -           | Running     |
 Tenant1-LS2=10.1.2.12 |
+------------------------------+-------------+------+----------+----------+
```

What happens in NSX-T?

- For each VM created:
  - (1) – 1 Logical Switch Port is created in NSX-T
    Under "NSX - Switching – Switches"

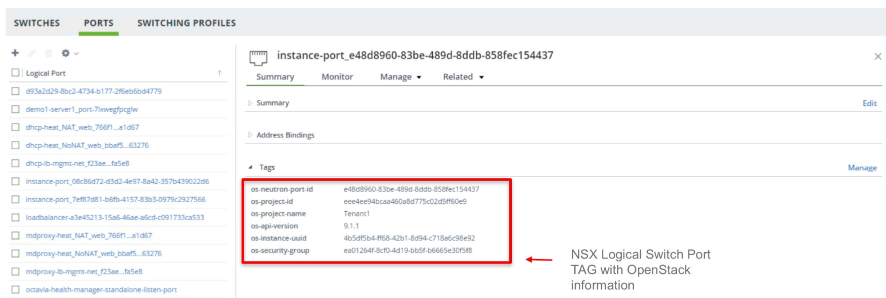



```
root@controller01:~# neutron port-list
+--------------------------------------+------+-------------------+--------------------------------------------------------------------------------------+
| id                                   | name | mac_address       | fixed_ips                                                                            |
+--------------------------------------+------+-------------------+--------------------------------------------------------------------------------------+
| 02374ea5-556b-40db-b16c-e1cd0cfd667b |      | fa:16:3e:f2:0d:7e | {"subnet_id": "d17090b8-6588-458a-829c-4cbeb1e87745", "ip_address": "10.1.2.5"}
| 6a5ead4d-d06d-460c-a5d4-ff1509547565 |      | fa:16:3e:ab:e6:3c | {"subnet_id": "271408e7-fbd9-4830-8808-68129d418c01", "ip_address": "10.1.1.5"}
| 94b925ec-a729-44b9-9f5f-fb861060253d |      | fa:16:3e:dc:7f:23 | {"subnet_id": "d17090b8-6588-458a-829c-4cbeb1e87745", "ip_address": "10.1.2.2"}
| b3e7bdb5-5357-4392-a939-61bc10818937 |      | fa:16:3e:71:ae:ac | {"subnet_id": "271408e7-fbd9-4830-8808-68129d418c01", "ip_address": "10.1.1.2"}
| d004b478-597f-4948-a422-5ac8bedf4dbb |      | fa:16:3e:ef:64:fa | {"subnet_id": "d17090b8-6588-458a-829c-4cbeb1e87745", "ip_address": "10.1.2.12"}
| e48d8960-83be-489d-8ddb-858fec154437 |      | fa:16:3e:e3:04:7b | {"subnet_id": "271408e7-fbd9-4830-8808-68129d418c01", "ip_address": "10.1.1.4"}
+--------------------------------------+------+-------------------+--------------------------------------------------------------------------------------+
```

Capture 4.25: Create logical port in NSX-T

- For each VM created:
  - 1 Logical Switch Port is created in NSX-T
    Under "NSX - Switching – Switches"

(2)



Capture 4.26: Create logical switch port in NSX-T

4.14.3 L2 Services - Overlay/VLAN Bridging

CLI

1) List L2 Gateway (created automatically by the plugin) - Requires admin credentials

```
root@controller01:~# neutron l2-gateway-list
+--------------------------------------+--------------+------------------------------+
| id                                   | name         | devices
+--------------------------------------+--------------+------------------------------+
| 3135e9bf-eebc-4d73-8f4e-261acb71b71a | default-l2gw | {"interfaces": [{"segmentation_id": [], "name":
"default-bridge-cluster"}], "id": "cbc72e06-0a94-4773-b5b5-ce4504f6552c", "device_name":
```

```
"4292e88f-0bab-4dec-8643-7e822c6f327a"}                              |

+--------------------------------------+--------------+------------------------------+
```

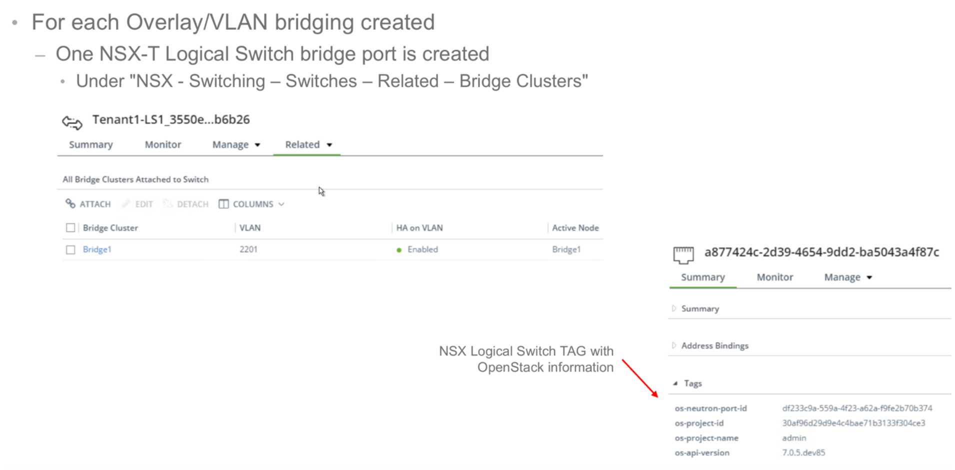
2) Create Overlay/VLAN Bridging - Requires admin credentials

```
root@controller01:~# neutron l2-gateway-connection-create default-l2gw Tenant1-LS1
--default-segmentation-id=2201 Created a new l2_gateway_connection:

+-----------------+--------------------------------------+

| Field           | Value                                |

+-----------------+--------------------------------------+

| id              | dccc1b79-87dd-4e61-a7ee-46d36de94921 |

| l2_gateway_id   | 3135e9bf-eebc-4d73-8f4e-261acb71b71a |

| network_id      | 1675ec95-b5dd-4f91-aa24-2af391733652 |

| segmentation_id | 2201                                 |

| tenant_id       |                                      |

+-----------------+--------------------------------------+
```

3) Verification

```
root@controller01:~# neutron l2-gateway-connection-list

+--------------------------------------+--------------------------------------+------+

| id                                   | l2_gateway_id                        |
network_id                           | segmentation_id |

+--------------------------------------+--------------------------------------+------+

| 795fe188-9a86-4071-9dcb-5c388f6f4852 | 3135e9bf-eebc-4d73-8f4e-261acb71b71a | 3550e54d-eabb-433d-806e-
3b1d2f7b6b26 |                                     2201 |

+--------------------------------------+--------------------------------------+------+

root@controller01:~# neutron l2-gateway-connection-show 795fe188-9a86-4071-9dcb-5c388f6f4852

+-----------------+--------------------------------------+

| Field           | Value                                |

+-----------------+--------------------------------------+

| id              | 795fe188-9a86-4071-9dcb-5c388f6f4852 |

| l2_gateway_id   | 3135e9bf-eebc-4d73-8f4e-261acb71b71a |

| network_id      | 3550e54d-eabb-433d-806e-3b1d2f7b6b26 |

| segmentation_id | 2201                                 |

| tenant_id       |                                      |

+-----------------+--------------------------------------+
```

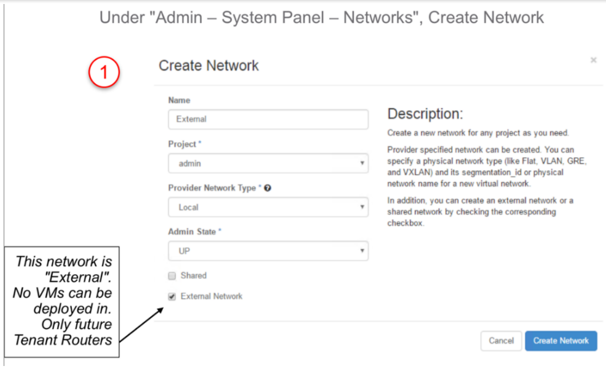
What happens in NSX-T?



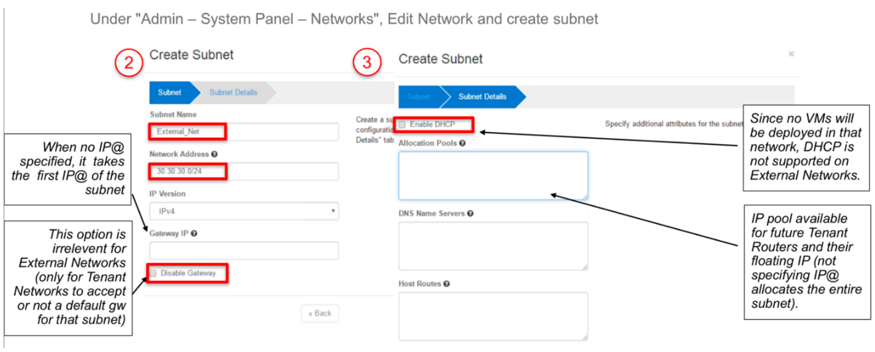
Capture 4.27: Create NSX-T logical switch bridge for overlay/VLAN bridge

4.14.4 L3 Services - External Network

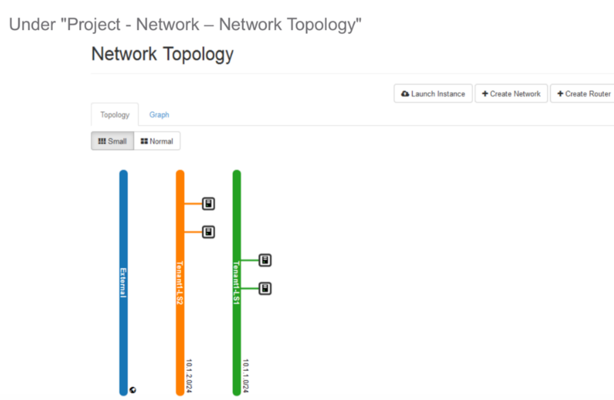
UI



Capture 4.28: Create external network



Capture 4.29: Create external subnet



Verification

vmware®

Capture 4.30: Network Topology

CLI

```
root@controller01:~# neutron net-create External --router:external Created a new network:

+-------------------------+-------------------------------------+

| Field                   | Value                               |

+-------------------------+-------------------------------------+

| admin_state_up          | True                                |

| availability_zone_hints |                                     |

| availability_zones      |                                     |

| created_at              | 2016-11-23T17:10:23Z                |

| description             |                                     |


|  id is_default name     |  82d5f5ac-91be-4749-8bee-0941a3ac5fed   |

|  port_security_enabled  |  False                                  |

|                         |  External                               |

|                         |  True                                   |

|  project_id             |  55e7a89f46ff41c1973169b25a634e06       |

|  revision_number        |  3                                      |

|  router:external        |  True                                   |

|  shared                 |  False                                  |

|  status                 |  ACTIVE                                 |

|  subnets                |                                         |

|  tags                   |                                         |

|  tenant_id              |  55e7a89f46ff41c1973169b25a634e06       |

|  updated_at             |  2016-11-23T17:10:24Z                   |

+-------------------------+--------------------------+

root@controller01:~# neutron subnet-create --name External_Net External 40.40.40.0/24 --no-gateway
--disable-dhcp Created a new subnet:

+-------------------+----------------------------------------------+

| Field             | Value                                        |

+-------------------+----------------------------------------------+

| allocation_pools  | {"start": "40.40.40.1", "end": "40.40.40.254"} |

| cidr              | 40.40.40.0/24                                |

| created_at        | 2016-11-23T07:37:33Z                         |

| description       |                                              |

| dns_nameservers   |                                              |

| enable_dhcp       | False                                        |

| gateway_ip        |                                              |

| host_routes       |                                              |
```

```
| id                | 9af7979a-122b-4b79-b321-7e9f055514f9           |
| ip_version        | 4                                              |
| ipv6_address_mode |                                                |
| ipv6_ra_mode      |                                                |
| name              | External_Net                                   |
| network_id        | 82d5f5ac-91be-4749-8bee-0941a3ac5fed           |
| project_id        | 55e7a89f46ff41c1973169b25a634e06               |
| revision_number   | 2                                              |
| subnetpool_id     |                                                |
| tenant_id         | 55e7a89f46ff41c1973169b25a634e06               |
| updated_at        | 2016-11-23T07:37:33Z                           |
+-------------------+------------------------------------------------
```

Verification

```
root@controller01:~# neutron net-show External
+-------------------------+--------------------------------------+
| Field                   | Value                                |
+-------------------------+--------------------------------------+
| admin_state_up          | True                                 |
| availability_zone_hints |                                      |
| availability_zones      |                                      |
| created_at              | 2016-11-23T07:37:00Z                 |
| description             |                                      |
| id                      | 82d5f5ac-91be-4749-8bee-0941a3ac5fed |
| is_default              | False                                |
| name                    | External                             |
| port_security_enabled   | True                                 |
| project_id              | 55e7a89f46ff41c1973169b25a634e06     |
| revision_number         | 5                                    |
| router:external         | True                                 |
| shared                  | False                                |
| status                  | ACTIVE                               |
| subnets                 | 9af7979a-122b-4b79-b321-7e9f055514f9 |
| tags                    |                                      |
| tenant_id               | 55e7a89f46ff41c1973169b25a634e06     |
| updated_at              | 2016-11-23T07:37:33Z                 |
+-------------------------+--------------------------------------+
root@controller01:~# neutron subnet-show External_Net
+-------------------+------------------------------------------------+
| Field             | Value                                          |
```

```
+-------------------+---------------------------------------------+
| allocation_pools  | {"start": "40.40.40.1", "end": "40.40.40.254"} |
| cidr              | 40.40.40.0/24                                |
| created_at        | 2016-11-23T07:37:33Z                         |
| description       |                                              |
| dns_nameservers   |                                              |
| enable_dhcp       | False                                        |
| gateway_ip        |                                              |
| host_routes       |                                              |
| id                | 9af7979a-122b-4b79-b321-7e9f055514f9         |
| ip_version        | 4                                            |
| ipv6_address_mode |                                              |
| ipv6_ra_mode      |                                              |
| name              | External_Net                                 |
 | network_id       | 82d5f5ac-91be-4749-8bee-0941a3ac5fed         |
 | project_id       | 55e7a89f46ff41c1973169b25a634e06             |
 | revision_number  | 2                                            |
 | subnetpool_id    |                                              |
 | tenant_id        | 55e7a89f46ff41c1973169b25a634e06             |
 | updated_at       | 2016-11-23T07:37:33Z                         |
+-------------------+---------------------------------------------+
```

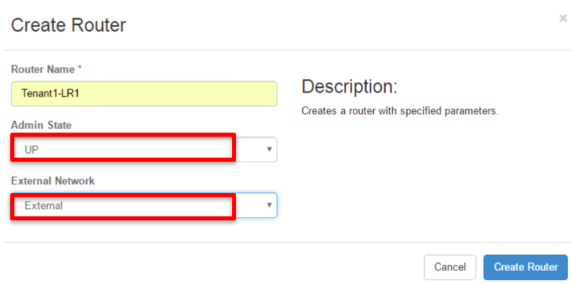
What happens in NSX-T?

Nothing

4.14.5 L3 Services - Logical Routing

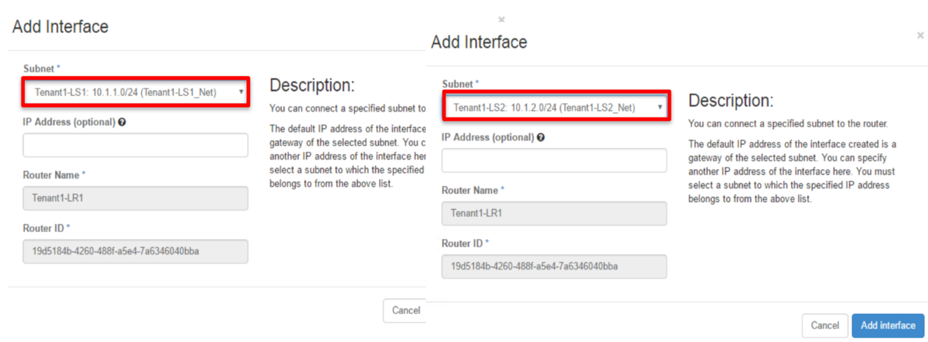
UI

Under "Project - Network – Routers", Create Router



Capture 4.31: Create Logical Router

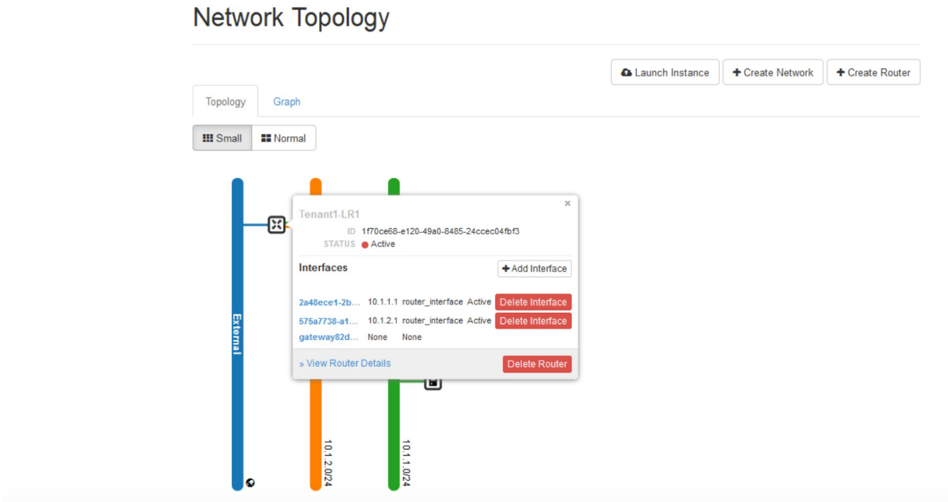
Under "Project - Network – Routers", Edit Router and add interface



Capture 4.32: Edit Router

Verification

Under "Project - Network – Network Topology"



Capture 4.33: Network Topology

CLI

```
root@controller01:~# neutron router-create Tenant1-LR1 Created a new router:

+-----------------------+--------------------------------------+

| Field                 | Value                                |

+-----------------------+--------------------------------------+

| admin_state_up        | True                                 |

| created_at            | 2016-11-23T17:25:39Z                 |

| description           |                                      |

| external_gateway_info |                                      |

| id                    | 1f70ce68-e120-49a0-8485-24ccec04fbf3 |

| name                  | Tenant1-LR1                          |

| project_id            | eee4ee94bcaa460a8d775c02d5ff60e9     |

| revision_number       | 2                                    |

| routes                |                                      |

| status                | ACTIVE                               |

| tenant_id             | eee4ee94bcaa460a8d775c02d5ff60e9     |

| updated_at            | 2016-11-23T17:25:39Z                 |

+-----------------------+--------------------------------------+
```

```
root@controller01:~# neutron router-interface-add Tenant1-LR1 Tenant1-LS1_Net Added interface 2a48ece1-
2b8e-4335-8056-92e5b875e842 to router Tenant1-LR1. root@controller01:~# neutron router-interface-add
Tenant1-LR1 Tenant1-LS2_Net

Added interface 575a7738-a17c-45f0-b55e-5dbb4d7f2aed to router Tenant1-LR1. root@controller01:~# neutron
router-gateway-set Tenant1-LR1 External
Set gateway for router Tenant1-LR
```

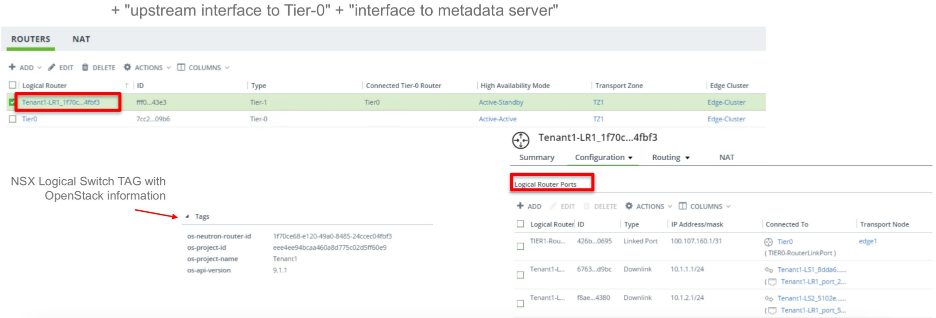
Verification

```
root@controller01:~# neutron router-list
+--------------------------------------+-------------+---------------------------+
| id                                   | name        |
external_gateway_info                                                            |
+--------------------------------------+-------------+---------------------------+
| 1f70ce68-e120-49a0-8485-24ccec04fbf3 | Tenant1-LR1 | {"network_id":
"82d5f5ac-91be-4749-8bee-0941a3ac5fed", "enable_snat": true, "external_fixed_ips":     |
|                                      |             | [{"subnet_id":
"9af7979a-122b-4b79-b321-7e9f055514f9", "ip_address": "40.40.40.5"}]}                  |
+--------------------------------------+-------------+---------------------------+
root@controller01:~# neutron router-show Tenant1-LR1
+-----------------------+--------------------------------------------------------+
| Field                 |
Value                   |
+-----------------------+--------------------------------------------------------+
 | admin_state_up        | True                                                                          |
 | created_at            | 2016-11-23T17:25:39Z                                                          |
 | description           |                                                                               |
 | external_gateway_info | {"network_id": "82d5f5ac-91be-4749-8bee-0941a3ac5fed", "enable_snat": true,
 "external_fixed_ips": [{"subnet_id": "9af7979a-122b-     |
 |                       | 4b79-b321-7e9f055514f9", "ip_address": "40.40.40.5"}]}                        |
 | id                    | 1f70ce68-e120-49a0-8485-24ccec04fbf3                                          |
 | name                  | Tenant1-LR1                                                                   |
 | project_id            | eee4ee94bcaa460a8d775c02d5ff60e9                                              |
 | revision_number       | 7                                                                             |
 | routes                |                                                                               |
 | status                | ACTIVE                                                                        |
 | tenant_id             | eee4ee94bcaa460a8d775c02d5ff60e9                                              |
 | updated_at            | 2016-11-23T17:28:00Z                                                          |
+-----------------------+--------------------------------------------------------+
root@controller01:~# neutron router-port-list Tenant1-LR1
+--------------------------------------+------+-------------------+---------------+
| id                                   | name | mac_address       |fixed_ip       |
+--------------------------------------+------+-------------------+---------------+
```

```
| 2a48ece1-2b8e-4335-8056-92e5b875e842 |       | fa:16:3e:0d:bc:d3 | {"subnet_id": "271408e7-fbd9-4830-8808-68129d418c01", "ip_address": "10.1.1.1"} |
| 575a7738-a17c-45f0-b55e-5dbb4d7f2aed |       | fa:16:3e:52:95:db | {"subnet_id": "d17090b8-6588-458a-829c-4cbeb1e87745", "ip_address": "10.1.2.1"} |

+--------------------------------------+------+-------------------+---------------+

root@controller01:~# neutron router-port-list Tenant1-LR1 (requires admin credentials)

+--------------------------------------+------+-------------------+---------------+

| id                                   | name | mac_address       |
fixed_ips                                                                          |

+--------------------------------------+------+-------------------+---------------+

| 2a48ece1-2b8e-4335-8056-92e5b875e842 |       | fa:16:3e:0d:bc:d3 | {"subnet_id": "271408e7-fbd9-4830-8808-68129d418c01", "ip_address": "10.1.1.1"} |

| 575a7738-a17c-45f0-b55e-5dbb4d7f2aed |       | fa:16:3e:52:95:db | {"subnet_id": "d17090b8-6588-458a-829c-4cbeb1e87745", "ip_address": "10.1.2.1"} |

| e2644db4-6462-4e33-aa10-124eb8980c63 |       | fa:16:3e:42:08:f9 | {"subnet_id": "9af7979a-122b-4b79-b321-7e9f055514f9", "ip_address": "40.40.40.5"} |

+--------------------------------------+------+-------------------+---------------+
```

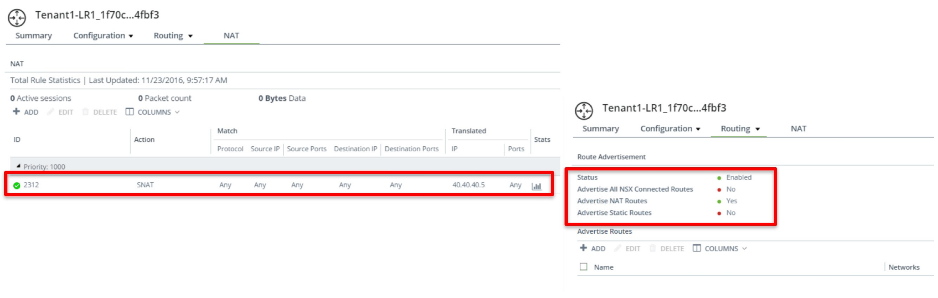
What happens in NSX-T?

- For each OpenStack Logical Router:
  1. – One NSX-T Tier1 Router is created
     Under "NSX – Routing"
     - with "LS internal interfaces"
       + "upstream interface to Tier-0" + "interface to metadata server"



Capture 4:34: Create NSX-T Tier 1 Router

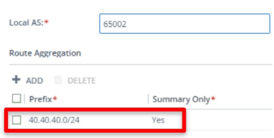
- For each OpenStack Logical Router:
  2 – One NSX-T Tier1 Router is created
  Under "NSX – Routing"
    - with SNAT rules for "South/North"



Capture 4:35: Update NSX-T Tier 1 Router

- Tier-0 router must be pre-configured to advertise external to physical.
  3 – Option1: Tier-0 advertise each /32 Tier-1 NAT
  Under "NSX – Routing", select Tier-0 and "Routing – Route Redistribution"

  – Option2: Tier-0 advertise the whole external subnet instead of /32 IP@ (route aggregation)
  In addition to Option1, under "NSX – Routing", select Tier-0 and "Routing – BGP", Edit BGP configuration



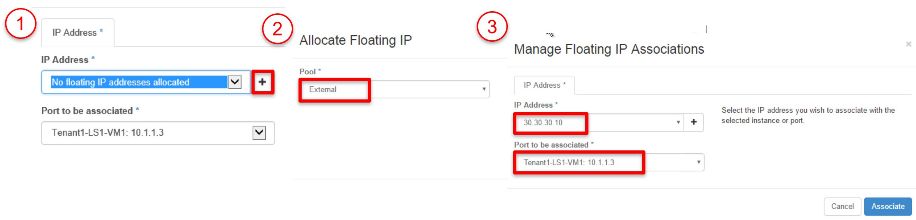
Capture 4:36: Configure NSX-T Tier 1 Router

4.14.6 L3 Services - Floating IP

GUI

Under "Project – Compute – Instances", on Instance "More – Associate Floating IP"

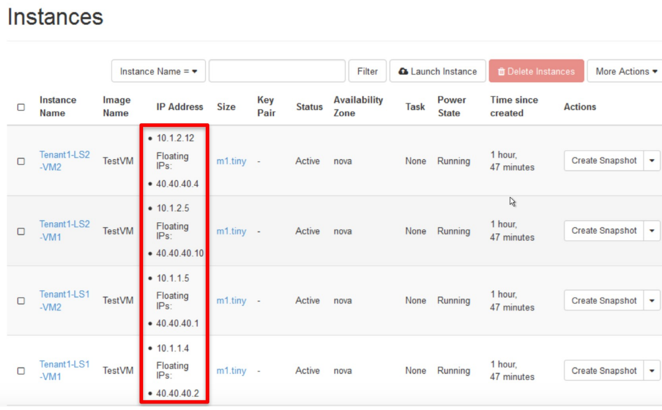
Manage Floating IP Associations



Capture 4:37: Manage floating IP associations

Verification

Under "Project – Compute – Instances"



Capture 4:38: Update floating IP instances

CLI

```
root@controller01:~# neutron net-external-list

+--------------------------------------+----------+--------------------------------------+

| id                                   | name     | subnets                              |

+--------------------------------------+----------+--------------------------------------+

| 82d5f5ac-91be-4749-8bee-0941a3ac5fed | External | 9af7979a-122b-4b79-b321-7e9f055514f9 |

+--------------------------------------+----------+--------------------------------------+

root@controller01:~# neutron floatingip-create External Created a new floatingip:
```

```
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| created_at          | 2016-11-23T18:20:40Z                 |
| description         |                                      |
| floating_ip_address | 40.40.40.2                           |
| floating_network_id | 82d5f5ac-91be-4749-8bee-0941a3ac5fed |
| id                  | a091276d-ea28-4034-b6b1-e3dd2b994f46 |
| port_id             |                                      |
| project_id          | eee4ee94bcaa460a8d775c02d5ff60e9     |
| revision_number     | 1                                    |
| router_id           |                                      |
| status              | DOWN                                 |
| tenant_id           | eee4ee94bcaa460a8d775c02d5ff60e9     |
| updated_at          | 2016-11-23T18:20:40Z                 |
+---------------------+--------------------------------------+
root@controller01:~# neutron port-list
+--------------------------------------+------+-------------------+---------------+
| id                                   | name | mac_address       |
fixed_ips                                                        |
+--------------------------------------+------+-------------------+---------------+
| 02374ea5-556b-40db-b16c-e1cd0cfd667b |      | fa:16:3e:f2:0d:7e | {"subnet_id": "d17090b8-6588-458a-829c-4cbeb1e87745", "ip_address": "10.1.2.5"}  |
| 2a48ece1-2b8e-4335-8056-92e5b875e842 |      | fa:16:3e:0d:bc:d3 | {"subnet_id": "271408e7-fbd9-4830-8808-68129d418c01", "ip_address": "10.1.1.1"}  |
| 575a7738-a17c-45f0-b55e-5dbb4d7f2aed |      | fa:16:3e:52:95:db | {"subnet_id": "d17090b8-6588-458a-829c-4cbeb1e87745", "ip_address": "10.1.2.1"}  |
| 6a5ead4d-d06d-460c-a5d4-ff1509547565 |      | fa:16:3e:ab:e6:3c | {"subnet_id": "271408e7-fbd9-4830-8808-68129d418c01", "ip_address": "10.1.1.5"}  |
| 94b925ec-a729-44b9-9f5f-fb861060253d |      | fa:16:3e:dc:7f:23 | {"subnet_id": "d17090b8-6588-458a-829c-4cbeb1e87745", "ip_address": "10.1.2.2"}  |
| b3e7bdb5-5357-4392-a939-61bc10818937 |      | fa:16:3e:71:ae:ac | {"subnet_id": "271408e7-fbd9-4830-8808-68129d418c01", "ip_address": "10.1.1.2"}  |
| d004b478-597f-4948-a422-5ac8bedf4dbb |      | fa:16:3e:ef:64:fa | {"subnet_id": "d17090b8-6588-458a-829c-4cbeb1e87745", "ip_address": "10.1.2.12"} |
| e48d8960-83be-489d-8ddb-858fec154437 |      | fa:16:3e:e3:04:7b | {"subnet_id": "271408e7-fbd9-4830-8808-68129d418c01", "ip_address": "10.1.1.4"}  |
+--------------------------------------+------+-------------------+---------------+
root@controller01:~# neutron floatingip-associate a091276d-ea28-4034-b6b1-e3dd2b994f46 e48d8960-83be-489d-8ddb-858fec154437
Associated floating IP a091276d-ea28-4034-b6b1-e3dd2b994f46
```

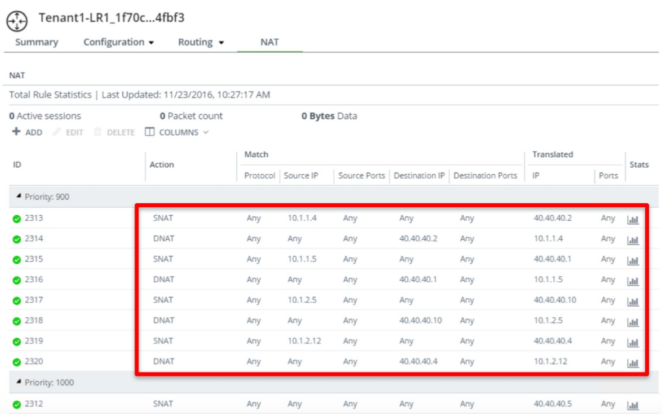
Verification

```
root@controller01:~# neutron floatingip-list
```

```
+--------------------------------------+------------------+---------------------+

| id                                   | fixed_ip_address | floating_ip_address |
port_id                              |
 +--------------------------------------+------------      | 40.40.40.1          |
 ------+-----------------+| 17ba67d4-f637-44e5-a45e-
 98e8416377c8 | 10.1.1.5
 6a5ead4d-d06d-460c-a5d4-ff1509547565 |
 | 4b36108b-bb67-4873-8df5-d954187d575c | 10.1.2.5         | 40.40.40.10         |
 02374ea5-556b-40db-b16c-e1cd0cfd667b |

 | a091276d-ea28-4034-b6b1-e3dd2b994f46 | 10.1.1.4         | 40.40.40.2          |
 e48d8960-83be-489d-8ddb-858fec154437 |

 | a42cf571-3fa9-4aaa-b169-59d54c7bafc0 | 10.1.2.12        | 40.40.40.4          |
 d004b478-597f-4948-a422-5ac8bedf4dbb |

+--------------------------------------+------------------+---------------------+
```

What happens in NSX-T?

- For each Floating IP@ created:
  - The Tenant NSX-T Tier1 Router is updated
    - Under "NSX – Routing – Router - NAT"
      - with SNAT + DNAT rules for "South/North" and "North/South"



Capture 4:39: Create floating IP in NSX-T

4.14.7 L3 Services - No-NAT

CLI

```
root@controller01:~# neutron net-list

+--------------------------------------+-------------+--------------------------------------------------+

| id                                   | name        | subnets                                          |

+--------------------------------------+-------------+---------------------------------------------------+
```

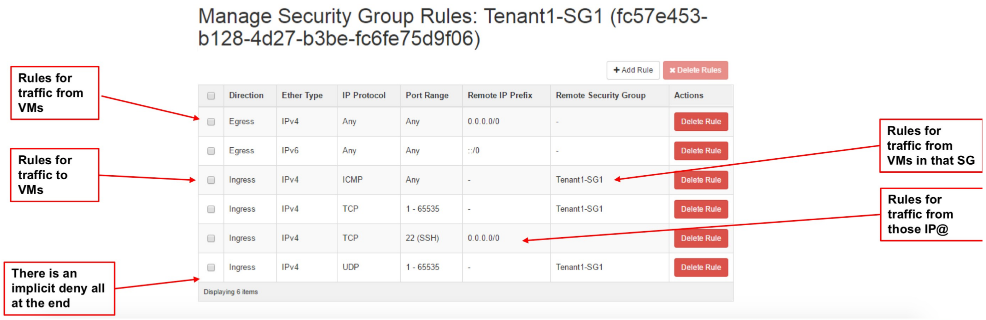
```
| 5102e155-8465-490b-ba7f-966a6ef73e06 | Tenant1-LS2 | d17090b8-6588-458a-829c-4cbeb1e87745 10.1.2.0/24|

| 82d5f5ac-91be-4749-8bee-0941a3ac5fed | External    | 9af7979a-122b-4b79-b321-7e9f055514f9            |

| 8dda6b07-5b8d-420b-b715-f891226b9e81 | Tenant1-LS1 | 271408e7-fbd9-4830-8808-68129d418c01 10.1.1.0/24|

+--------------------------------------+-------------+------------------------------------------------+

root@controller01:~# neutron router-update Tenant1-LR1 --external_gateway_info type=dict
network_id=82d5f5ac-91be-4749-8bee-0941a3ac5fed,enable_snat=False

Updated router: Tenant1-LR1
```

Verification

```
root@controller01:~# neutron router-show Tenant1-LR1

+-----------------------+---------------------------------------------------------+

| Field                 | Value                                                   |

+-----------------------+---------------------------------------------------------+

| admin_state_up        | True                                                    |

| created_at            | 2016-11-23T17:25:39Z                                    |

| description           |                                                         |

| external_gateway_info | {"network_id": "82d5f5ac-91be-4749-8bee-0941a3ac5fed", "enable_snat": false,
"external_fixed_ips": [{"subnet_id": "9af7979a-122b-                            |

|                       | 4b79-b321-7e9f055514f9", "ip_address": "40.40.40.5"}]  |

| id                    | 1f70ce68-e120-49a0-8485-24ccec04fbf3                    |

| name                  | Tenant1-LR1                                             |

| project_id            | eee4ee94bcaa460a8d775c02d5ff60e9                        |

| revision_number       | 8                                                       |

| routes                |                                                         |

| status                | ACTIVE                                                  |

| tenant_id             | eee4ee94bcaa460a8d775c02d5ff60e9                        |

| updated_at            | 2016-11-23T18:29:37Z                                    |

+-----------------------+---------------------------------------------------------+
```

What happens in NSX-T?

Under "Project – Compute – Access & Security – Manage Rules"

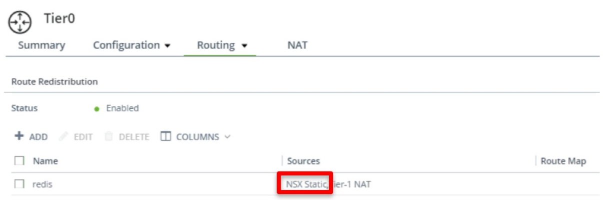


	Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Actions
	Egress	IPv4	Any	Any	0.0.0.0/0	-	Delete Rule
	Egress	IPv6	Any	Any	::/0	-	Delete Rule
	Ingress	IPv4	ICMP	Any	-	Tenant1-SG1	Delete Rule
	Ingress	IPv4	TCP	1 - 65535	-	Tenant1-SG1	Delete Rule
	Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0		Delete Rule
	Ingress	IPv4	UDP	1 - 65535	-	Tenant1-SG1	Delete Rule

Note: If Floating VIP are also configured those are applied on NSX (but in case of no-NAT, generally Floating-IP are not used).

Capture 4:40: Manage security group rules

- Tier-0 router must be pre-configured to advertise external to physical.
  2 – Option1: Tier-0 advertise Tier-1 advertised connected-subnets
    Under "NSX – Routing", select Tier-0 and "Routing – Route Redistribution"

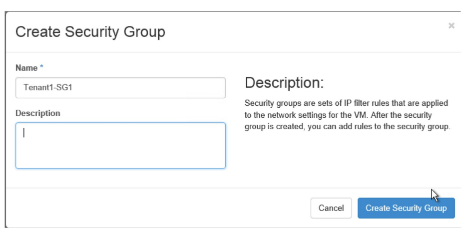


Capture 4:41: Configure Tier 0 router

4.14.8 Security Services - Security Groups

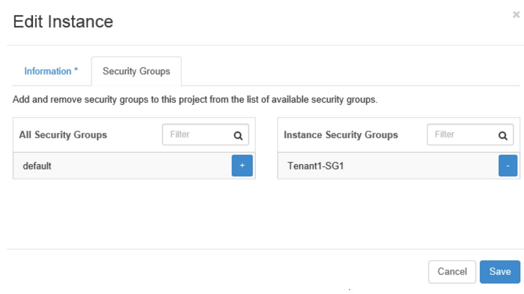
GUI

Under "Project – Compute – Access & Security", Create "Security Group"



Capture 4:42: Create security groups

Under "Project – Compute – Instances", on Instance "More – Edit Security Groups"



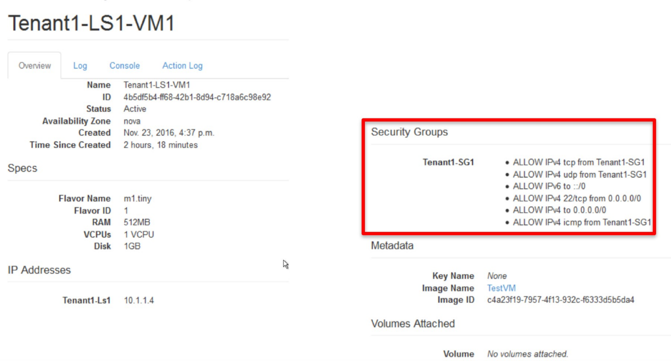
Note: In case of VM with multiple NIC, each VM-NIC receives that policy.

Via CLI/API, it's possible to apply a Security Group to a specific VM-NIC (see CLI section below).

Capture 4:43: Edit Security Group

Verification

Under "Project – Compute – Instances", Edit Instance



Capture 4:44: Edit instances

CLI

```
root@controller01:~# neutron security-group-create Tenant1-SG1

+----------------------+---------------------------------------------------------+
| Field                |Value                                                    |
+----------------------+---------------------------------------------------------+
| created_at           | 2016-11-23T18:43:21                                     |
| description          |                                                         |
| id                   |75f7d59e-868a-47df-b7f1-10d29c7e7dc1                     |
| logging              |False                                                    |
| name                 |Tenant1-SG1                                              |
```

```
| project_id           |eee4ee94bcaa460a8d775c02d5ff60e9                            |

| provider             |False                                                       |

| revision_number      | 1                                                          |

| security_group_rules | {"local_ip_prefix": null, "direction": "egress", "protocol": null,
"description": null, "remote_group_id": null, "ethertype": "IPv4",                 |

|                      | "remote_ip_prefix": null, "port_range_max": null, "updated_at": "2016-11-
23T18:43:21Z", "security_group_id": "75f7d59e-868a-47df-                           |

|                      | b7f1-10d29c7e7dc1", "port_range_min": null, "revision_number": 1, "tenant_id":
"eee4ee94bcaa460a8d775c02d5ff60e9", "created_at":                                  |

|                      | "2016-11-23T18:43:21Z", "project_id": "eee4ee94bcaa460a8d775c02d5ff60e9", "id":
"a9958f2f-c9bb-4a55-af40-e6d3bdc76160"}                                            |

|                      | {"local_ip_prefix": null, "direction": "egress", "protocol": null,
"description": null, "remote_group_id": null, "ethertype": "IPv6",                 |

|                      | "remote_ip_prefix": null, "port_range_max": null, "updated_at": "2016-11-
23T18:43:21Z", "security_group_id": "75f7d59e-868a-47df-                           |

|                      | b7f1-10d29c7e7dc1", "port_range_min": null, "revision_number": 1, "tenant_id":
"eee4ee94bcaa460a8d775c02d5ff60e9", "created_at":                                  |

|                      | "2016-11-23T18:43:21Z", "project_id": "eee4ee94bcaa460a8d775c02d5ff60e9", "id":
"3a730dd0-89de-4099-80be-3f4c09bb07ad"}                                            |

| tenant_id            |eee4ee94bcaa460a8d775c02d5ff60e9                            |

| updated_at           | 2016-11-23T18:43:21Z                                       |

+----------------------+------------------------------------------------------------+

root@controller01:~# neutron security-group-rule-create --direction ingress

--protocol icmp --remote-group-id Tenant1-SG1 Tenant1-SG1

+-------------------+--------------------------------------+

| Field             | Value                                |

+-------------------+--------------------------------------+

| created_at        | 2016-11-23T18:43:46Z                 |

| description       |                                      |

| direction         | ingress                              |

| ethertype         | IPv4                                 |

| id                | f001ad75-87a3-45fb-95d5-c8d5d96fc034 |

| local_ip_prefix   |                                      |

| port_range_max    |                                      |

| port_range_min    |                                      |

| project_id        | eee4ee94bcaa460a8d775c02d5ff60e9     |

| protocol          | icmp                                 |

| remote_group_id   | 75f7d59e-868a-47df-b7f1-10d29c7e7dc1 |

| remote_ip_prefix  |                                      |

| revision_number   | 1                                    |

| security_group_id | 75f7d59e-868a-47df-b7f1-10d29c7e7dc1 |

| tenant_id         | eee4ee94bcaa460a8d775c02d5ff60e9     |

| updated_at        | 2016-11-23T18:43:46Z                 |
```

```
+------------------+---------------------------------------+

root@controller01:~# neutron security-group-rule-create --direction ingress --protocol tcp
--remote-group-id Tenant1-SG1 Tenant1-SG1
root@controller01:~# neutron security-group-rule-create --direction ingress --protocol udp
--remote-group-id Tenant1-SG1 Tenant1-SG1
root@controller01:~# neutron security-group-rule-create --direction ingress --protocol tcp
--port_range_min 22 --port_range_max 22 Tenant1-SG1 root@controller01:~# nova secgroup-list
+--------------------------------------+-------------+------------------------+
| Id                                   | Name        | Description            |
+--------------------------------------+-------------+------------------------+
| 75f7d59e-868a-47df-b7f1-10d29c7e7dc1 | Tenant1-SG1 |                        |
| ea01264f-8cf0-4d19-bb5f-b6665e30f5f8 | default     | Default security group |
+--------------------------------------+-------------+------------------------+
root@controller01:~# nova add-secgroup Tenant1-LS1-VM1 Tenant1-SG1 root@controller01:~# nova add-
secgroup Tenant1-LS1-VM2 Tenant1-SG1 root@controller01:~# nova add-secgroup Tenant1-LS2-VM1 Tenant1-SG1
root@controller01:~# nova add-secgroup Tenant1-LS2-VM2 Tenant1-SG1 root@controller01:~# nova remove-
secgroup Tenant1-LS1-VM1 default root@controller01:~# nova remove-secgroup Tenant1-LS1-VM2 default

root@controller01:~# nova remove-secgroup Tenant1-LS2-VM1 default root@controller01:~# nova remove-
secgroup Tenant1-LS2-VM2 default
```

Verification

```
root@controller01:~# nova show Tenant1-LS1-VM1
+--------------------------------------+----------------------------------------------------------+
| Property                             | Value                                                    |
+--------------------------------------+----------------------------------------------------------+
| OS-DCF:diskConfig                    | MANUAL                                                   |
| OS-EXT-AZ:availability_zone          | nova                                                     |
| OS-EXT-STS:power_state               | 1                                                        |
| OS-EXT-STS:task_state                | -                                                        |
| OS-EXT-STS:vm_state                  | active                                                   |
| OS-SRV-USG:launched_at               | 2016-11-23T16:37:46.000000                               |
| OS-SRV-USG:terminated_at             | -                                                        |
| Tenant1-LS1 network                  | 10.1.1.4                                                 |
| accessIPv4                           |                                                          |
| accessIPv6                           |                                                          |
| config_drive                         |                                                          |
| created                              | 2016-11-23T16:37:42Z                                     |
| description                          | -                                                        |
| flavor                               | m1.tiny (1)                                              |
```

```
| hostId                               | 9767c7fc0f3b83c271bb4f7ab1b99274e3b594b9065178549895b991 |
| id                                   | 4b5df5b4-ff68-42b1-8d94-c718a6c98e92                     |
| image                                | TestVM (c4a23f19-7957-4f13-932c-f6333d5b5da4)            |
| key_name                             | -                                                        |
| locked                               | False                                                    |
| metadata                             | {}                                                       |
| name                                 | Tenant1-LS1-VM1                                          |
| os-extended-volumes:volumes_attached | []                                                       |
| progress                             | 0                                                        |
| security_groups                      | Tenant1-SG1                                              |
| status                               | ACTIVE                                                   |
| tags                                 | []                                                       |
```

```
| tenant_id                    | eee4ee94bcaa460a8d775c02d5ff60e9                                   |
| updated                      | 2016-11-23T16:37:46Z                                               |
| user_id                      | 4a3454a70cf640d589327b857d174c07                                   |
+-------------------------------------------------------------------------------------------------+
```

What happens in NSX-T?

For each Security Group created:

- One NSX-T NSGroup is created

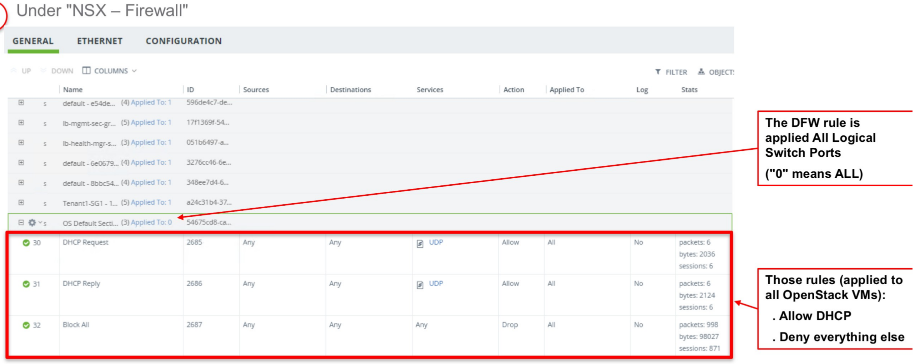
G) Under "**NSX** - Inventory - Groups"

GROUPS IP SETS IP POOLS MAC SETS

@ Tena nt1-SG1 -...-4116-964a-4fe5af...

- There is a last section for the implicit Drop at the end:

3 Under "NSX – Firewall"



Capture 4:46: NSX-firewall

4.14.9 Security Services - Port-Security

It's recommended and enabled by default. You can disable it if you need to.

GUI

Verification

Under "Project – Network – Networks", Edit Network, Go on "Ports" tab, and Select a Port

Capture 4:47: Edit Networks

CLI

Verification

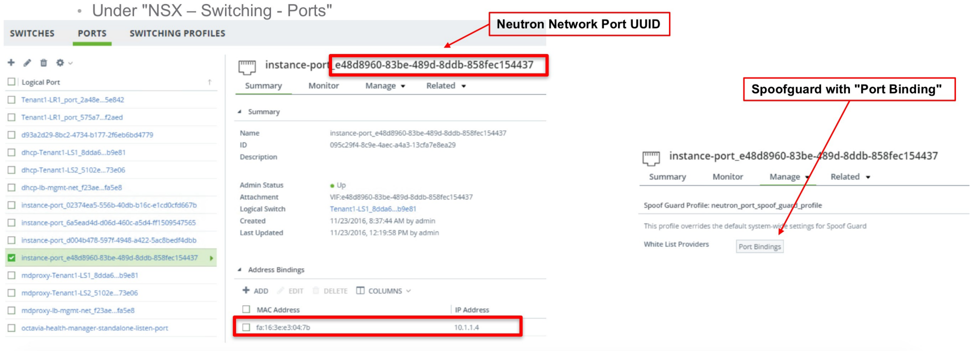
```
root@controller01:~# neutron port-list

+--------------------------------------+------+-------------------+-----------------+

| id                                   | name | mac_address       |fixed_ips        |

+--------------------------------------+------+-------------------+-----------------+

| 02374ea5-556b-40db-b16c-e1cd0cfd667b |      | fa:16:3e:f2:0d:7e | {"subnet_id": "d17090b8-6588-458a-829c-4cbeb1e87745", "ip_address": "10.1.2.5"}  |

| 2a48ece1-2b8e-4335-8056-92e5b875e842 |      | fa:16:3e:0d:bc:d3 | {"subnet_id": "271408e7-fbd9-4830-8808-68129d418c01", "ip_address": "10.1.1.1"}  |
| 575a7738-a17c-45f0-b55e-5dbb4d7f2aed |      | fa:16:3e:52:95:db | {"subnet_id": "d17090b8-6588-458a-829c-4cbeb1e87745", "ip_address": "10.1.2.1"}  |

| 6a5ead4d-d06d-460c-a5d4-ff1509547565 |      | fa:16:3e:ab:e6:3c | {"subnet_id": "271408e7-fbd9-4830-8808-68129d418c01", "ip_address": "10.1.1.5"}  |

| 94b925ec-a729-44b9-9f5f-fb861060253d |      | fa:16:3e:dc:7f:23 | {"subnet_id": "d17090b8-6588-458a-829c-4cbeb1e87745", "ip_address": "10.1.2.2"}  |

| b3e7bdb5-5357-4392-a939-61bc10818937 |      | fa:16:3e:71:ae:ac | {"subnet_id": "271408e7-fbd9-4830-8808-68129d418c01", "ip_address": "10.1.1.2"}  |

| d004b478-597f-4948-a422-5ac8bedf4dbb |      | fa:16:3e:ef:64:fa | {"subnet_id": "d17090b8-6588-458a-829c-4cbeb1e87745", "ip_address": "10.1.2.12"} |

| e48d8960-83be-489d-8ddb-858fec154437 |      | fa:16:3e:e3:04:7b | {"subnet_id": "271408e7-fbd9-4830-8808-68129d418c01", "ip_address": "10.1.1.4"}  |

+--------------------------------------+------+-------------------+-----------------+

root@controller01:~# neutron port-show e48d8960-83be-489d-8ddb-858fec154437

+-------------------------+---------------------------------------------------------+

| Field                   |Value                                                             |

+-------------------------+---------------------------------------------------------------+
```


vmware®

```
|Admin_state_up                                |  True                                             |
|binding:vnic_type_normal                      |                                                   |
| fixed_ips                                    | {"subnet_id": "271408e7-fbd9-4830-8808-68129d418c01",
"ip_address": "10.1.1.4"} |
| id                                           |e48d8960-83be-489d-8ddb-858fec154437               |
| mac_address                                  |fa:16:3e:e3:04:7b                                  |
| port_security_enabled                        |True                                               |
| project_id                                   |eee4ee94bcaa460a8d775c02d5ff60e9                   |
| security_groups                              |75f7d59e-868a-47df-b7f1-10d29c7e7dc1               |
+-------------------------+------------------------------------------------------------+
```

What happens in NSX-T?

- For each Instance created
  - One NSX-T Address Binding is created on Logical Switch Port
  - And a "Port Binding" spoofguard profile is associated to Logical Switch Port
    - Under "NSX – Switching - Ports"

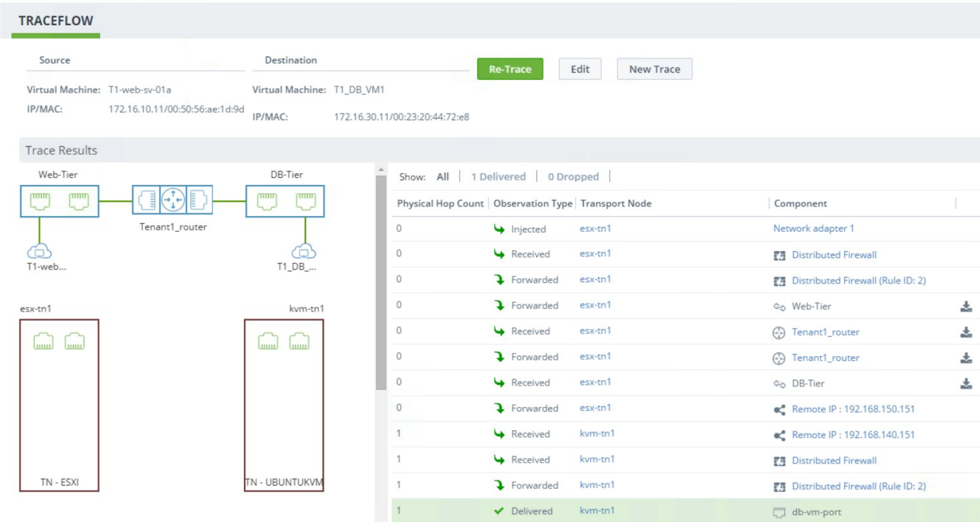


Capture 4.48: Associate logical switch port with NSX-T address binding

4.15 NSX-T Operations

In heterogeneous environments, having consistent operations, monitoring, and troubleshooting becomes even more crucial than in homogeneous deployments. NSX-T has the operational toolbox for managing and troubleshooting complex environments running on heterogeneous infrastructures.

Whenever you have a complex and multi-tenant environment like OpenStack, native operations tools are extremely useful for troubleshooting and monitoring.
NSX-T has tools such as Port Connectivity and Traceflow, which help users trace connectivity through virtual and physical devices and detect failures.



Capture 4.48: NSX-T Operations traceflow

NSX-T provides granular flow and packet-level visibility through standards tools such as IPFIX and port mirroring. This enables customers to use their existing monitoring and troubleshooting tools for network visibility when troubleshooting.

Section 5: VIO Storage Integration

5.1 Block storage

5.1.1 Overview

Block storage in VMware Integrated OpenStack is provided by Cinder. The Cinder services (cinder-api, cinder-volumes, cinder-scheduler) are hosted on each of the VMware Integrated OpenStack controllers; the cinder-api is load-balanced both for internal traffic and external traffic by the VIO management plane load-balancer running on loadbalancer01 and loadbalancer02.

- cinder-api: Provides a RESTful API used to interact with the Cinder services
- cinder-scheduler: Determines volume placement for newly provisioned block storage and forwards the request to cinder-volume
- cinder-volume: Consumes the block storage management request and routes it to the designated storage backend using a specialized driver

The Cinder services functionality has not been modified from OpenStack mainline and a VMware vSphere-specific driver has been developed to support vSphere-centric storage platforms. In OpenStack Cinder services rely on first-class block storage, however, the VIM object model and API does not support block storage in a manner that is conducive to native usage in OpenStack. Required functionality is provided in VMware Integrated OpenStack by mimicking a first-class block storage object using the following steps to create a new volume:

- At the time of Cinder volume provisioning the vSphere Cinder driver will create a new Cinder volume object in the OpenStack database.
- At the time of the attachment of the volume to a virtual instance the Cinder driver will request provisioning of a new virtual instance, referred to as a shadow VM.
- The shadow VM is provisioned with a single VMDK volume whose size matches the requested Cinder volume size.
- Once the shadow VM is provisioned successfully Cinder will attach the shadow VM's VMDK file to the target virtual instance and Cinder will treat the VMDK as the Cinder volume.

Management of the Cinder volume is the same as OpenStack mainline.

5.1.2 Design Considerations General

vSphere storage and IP storage networking best practices should be followed wherever possible. The block storage consumption model used by the Cinder server under VIO does not require any additional considerations.

cinder-api

Generally, the cinder-api service requires no additional configuration under the standard usage pattern. In special cases, there are options that can be modified to tune the API pipeline but these options should only be modified if the impacts are fully understood.

- osapi_volume_workers: Sets the number of cinder-api workers and defaults to the number of CPU cores reported by the operating system. The best practice is to leave this option unchanged, however, there may be extenuating circumstances, out-of-scope of this document, that require modification. Modification of this option can lead to performance impacts.
- api_rate_limit: Enables or disables the rate limit of the cinder-api. The default value is disabled. When enabled cinder-api will any requests that violate the value provided and therefore the best practice is to leave this option disabled.

cinder-scheduler

The cinder-scheduler service requires no additional configuration under the standard usage pattern. By default, the cinder-scheduler service enables the following filters:

- AvailabilityZoneFilter - filters the storage backends by availability zone.

- CapacityFilter
- CapabilitiesFilter

The cinder-scheduler service enables the following weigher:

- ChanceWeigher - assigns random weights to hosts, used to distribute volumes randomly across a list of equally suitable hosts

VMware Integrated OpenStack does not support storage over-subscription (via the *max_over_subscription_ratio* option) or enabling thin or thick provisioning support (via the *thin_provisioning_support* and *thick_provisioning_support* options) from within the Cinder VMDK driver and therefore these options are hardcoded in cinder-scheduler. The Cinder VMDK driver also presents all of the backend storage as a single pool to the scheduler. However, this configuration does not affect over-subscription or thin and thick provisioning support at the underlying storage array.

cinder-volume

The vSphere Cinder driver supports the following datastore types:

- NFS
- VMFS
- VSAN
- VVOL

The Cinder service uses cinder-scheduler to tell it where to place new new volumes. There are nuances to how the VMware SDDC stack interacts with VMware Integrated OpenStack:

- Storage vMotion and Storage DRS are supported when running VMware Integrated OpenStack. However, there is no integration between cinder-scheduler and and neither scheduler has information about the other nor understands each other.
- vSphere SPBM policies can be applied to provide storage tiering by using metadata. Snapshots of cinder volumes are placed on the same datastore as the master volume.
- Backups should also be configured to provide disaster recovery services for Cinder volumes and prevent loss of data. The Cinder services can be configured to back up volumes to a NFS share or Swift Object Store.

Cinder Availability Zones

Cinder Availability Zones are fully supported and can be utilized to segment Cinder volumes. The standard OpenStack management process should be used.

Cinder Consistency Groups

Consistency groups are not supported by the Cinder VMDK driver.

Cinder Volume Replication or Migration

Volume replication or volume migration that is controlled and managed at the VMware Integrated OpenStack layer is not supported. However, volume replication or volume migration can be achieved via an array-based replication solution. VMware Integrated OpenStack does not support importing of Cinder volumes, and so, any replication or migration of a Cinder volume does not allow for reuse within the same or different VMware Integrated OpenStack instance.

Cinder QoS

QoS management on Cinder volumes is not supported via the VMware Cinder VMDK driver. However, through the use of OpenStack flavor or image metadata, vSphere Storage Policy Based Management (SPBM) policies can be applied to instances that will ensure placement of a Cinder volume onto a SPBM-managed vSphere datastore.

5.2 Object Storage

VMware Integrated OpenStack provides object storage by the OpenStack-standard service Swift, but in an optional, unsupported manner. Industry best practices should be followed for the design of the backing Object Store.

5.2.1 Ephemeral Instance Storage

All nonpersistent virtual instance storage is provided by Nova datastores and is collocated with the virtual instances. VMware SDDC best practices should be followed.

5.3 Image Storage

The Glance services provide an image catalog that can be used to deploy new virtual instances. Glance uses standard vSphere datastores to host the images via the VMware VMDK driver.

Section 6: VIO Image Maintenance (Glance)

6.1 Introduction

The OpenStack Image Service called Glance provides discovery, registration, and delivery services for disk and server images. To support rapid provisioning of instances (VMs) are instantiated from a pre-built operating system image. For vSphere administrators, a very good analogy would be the VM template from which you clone. VMware Integrated Openstack provides an Image Service (Glance) for storage and management of OpenStack images. (Figure 6.1) There are several administration options, with both GUI and CLI based methods available.
VIO stores uploaded images to a designated vSphere datastore or multiple datastores.

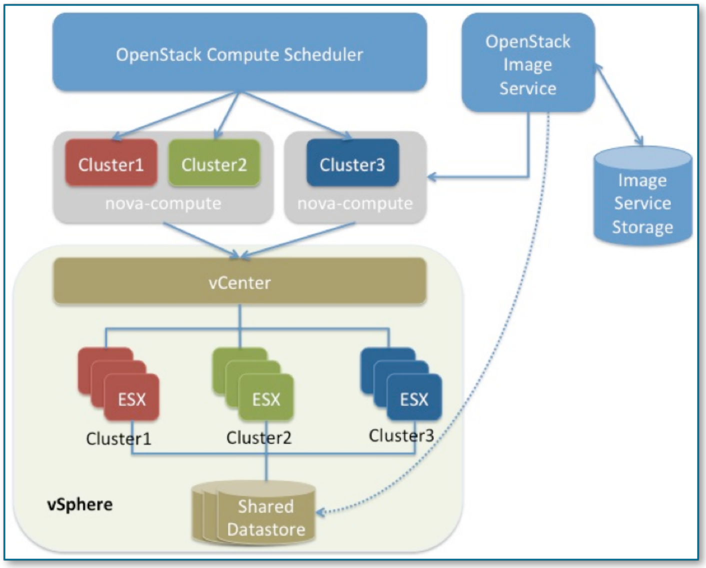


Figure 6.1: Openstack Image Service

Glance has the ability to copy (or snapshot) a server image, and then to store it promptly. Stored images then can be used as templates to get new servers up and running quickly, and can also be used to store and catalog unlimited backups. When an image is stored, it can either be stored as a public image or a private image.

- A public image is accessible to all projects (tenants), and is typically provided by an OpenStack administrator.
- A private image is created by an individual user, and will only be available to that project.

VMware Integrated OpenStack is pre-bundled with a Ubuntu image. That Ubuntu image is a public image. Glance includes the following components:

- Glance-api: Accepts Image API calls for image discovery, retrieval and storage.
- Glance-registry: Stores, processes, and retrieves metadata for images. Metadata includes size, type, and so on.
- Database: Stores image metadata.
- Storage repository: Supports normal file systems, or any datastore VMFS that is supported on the HCL (hardware compatibility list) for storage on vSphere.

For information about import images, see the latest VIO Admin guide. Go to https://www.vmware.com/support/pubs, select VMware Integrated Openstack from the product list on the right, and then select the administration guide from the “view VMware Integrated Openstack documentation center" link.

6.2 Understanding the difference between Instances, Images, & Flavors

Images: File representation of an Operating System. Can be a Public image (admin created) or Private image (user created per project)

Flavors: Specify CPU, Memory, Disk. A flavor can be a default (T-Shirt Sized) or user specified

Instance: The running image in the specified flavor size.

6.3 Supported Image Formats for Glance

The Glance image service component in VMware Integrated OpenStack natively supports images that are packaged in the formats ISO, OVA, and VMDK.
However, if you have an image in a different format, you can import the image into the Glance Service component of VIO. Currently, you can import the following formats into the Glance service of VMware Integrated OpenStack

- VMDK
- OVA
- RAW
- QCOW2
- VDI
- VHD
- ISO

You can import images to glance either via the Horizon Dashboard or via the CLI. Details of how to do this are described later in this chapter.

Note: If you are importing an image using a RAW, QCOW2, VDI, or VHD source image format, verify that the source image is hosted on a server without credentials to allow plain HTTP requests. Otherwise the transfer will fail.

6.4 Installation of Glance CLI on a Linux Machine

There are two ways to interact with the Glance API. Firstly via the Horizon GUI and secondly via the Glance CLI. To interact with the Glance CLI you can install the CLI tool. Horizon functionality is limited compared to CLI tools in the case of Glance. So, we recommend that you install the Glance CLI tools.
We will begin by installing the glance CLI client to an existing Linux Workstation. It is possible to install the tools on other operating systems, but I want to keep the process as simple as possible.

Section A Option 1 - Debian based Workstation Requirements such as Ubuntu:

1. Install a debian based Linux Distribution, such as Ubuntu 16.04 LTS.
2. Ensure Python 2.7 or later is installed by issuing the following command.

   apt-get install python

3. Ensure that Python 2.7 has been installed on the linux workstation.

   python –version

4. Add a package installation

repository.add-apt-repository -y cloud-archive:mitaka

5. Get the latest version of the CLI tool to add a package installation repository.

   apt-get update && apt-get install python-glanceclient

6. Check the version of Glance. glance --version

Section A Option 2 - Fedora based Workstation Requirements such as Fedora, Redhat or Centos:

If your Linux Workstation is an Enterprise Linux derivative, use the yum package management tool instead: Ensure you have a linux version installed.

1. Install the rdo release, we will update to the latest release in a later step so dont worry about the package being a kilo release

   yum install –y https://repos.fedorapeople.org/repos/openstack/openstack-kilo/rdo-release-kilo-1.noarch.rpm

2. Update the package.

   yum update –y

3. Install the python-glance client.

   yum install python-glanceclient

4. Verify that the version of glance is is later version 2.7 (output should be greater than 0.12.0)

   glance --version

Section B - Install Certificate

We now need to copy a certificate from the load balancer to the OMS server. Run the following command from the OMS server to copy the certificate.

scp -i ~/.ssh/ida_rsa <loadbalancer01>:/etc/ssl/vio.pem .

Where <loadbalancer01> is the dns name of your load balancer appliance.

Note: Include the period at the end of the command. This copies it into the working directory.

6.4.1 Creating an Image from a QCOW2 Image Format

You might need to convert an image from QCOW2 format, because there are a large number of public pre-built OpenStack images available. These images normally exist in QCOW2 image format. This is not an ESXi-friendly image, and therefore needs to be converted.

This is also a great exercise for migrating from a pre-existing development cloud where other hypervisors are in use, and therefore, so are QCOW2 images. After conversion, we upload or import to the Glance repository. The behavior of the instance is exactly the same, but it is now running on vSphere with the ability to leverage all of the underlying platform technology, such as HA, vMotion, and DRS.

6.4.2 Procedure

1. Using SSH, log in to the VMware Integrated OpenStack manager.

2. From the VMware Integrated OpenStack manager, use SSH to log in to the controller01 node.
3. Switch to root user.

   sudo su -

4. Using your preferred text editor (vi, vim, joe), edit the cloudadmin.rc file to include the following line export OS_AUTH_URL=http://INTERNAL_VIP:35357/v2.0

Important: Use the FQDN of the internal VIP that you have in your environment. For example:

Capture 6.1: Creating an Image from a QCOW2 Image Format

5. Save the file.
6. Now run the following command

   source cloudadmin.rc

7. To import the image, run the glance-import command. For details and options of the command, see the latest version of the administration guide, available at https://docs.vmware.com/en/VMware-Integrated-OpenStack/index.html then select the administration guide.

For example: Note that the file is accessible via http

glance-import cirros-img qcow2 http://launchpad.net/cirros/trunk/0.3.0/+download/cirros-0.3.0-x86_64-disk.img

The CLI displays the task information and status, including the task ID and image ID.

Created import task with id 5cdc4a04-5c68-4b91-ac44-37da07ec82ec

Waiting for Task 5cdc4a04-5c68-4b91-ac44-37da07ec82ec to finish.

Current Status.. SUCCESS

Image cirros-img created with ID: 2120de75-0717-4d61-b5d9-2e3f16e79edc

8. (Optional) To verify that the image is included in the Image Service, run the following CLI command glance image-list. The command returns a list of all images that are available in the Image Service.

6.4.3 Adding a vCenter VM Template as an OpenStack Image

One of the first elements you want to seed your cloud with is images. So, users and developers can start building applications. In a private cloud environment, cloud admins expose a list of standard Operating System images to be used to that end, in other words Operating System master images.

When VIO is deployed on top of an existing vSphere environment, these Operating System master images are generally already present in the virtualization layer as vSphere templates and a great deal of engineering hours have gone into creating and configuring those images to reflect the needs of a given corporate organization in terms of security, compliance or regulatory requirements, for example Operating System hardening, customization, agents installation, and patching.

VMware Integrated Openstack supports the capability to leverage their existing vSphere templates by adding them to their OpenStack deployment as Glance images, which can then be booted as OpenStack instances or used to create bootable Cinder volumes.

The beauty of this feature is that it is done without copying the template into the Glance datastore. The media only exists in one place, which is the original datastore where the template is stored, and you actually create a pointer from the OpenStack image object towards the vSphere template, thus eliminating the potentially lengthy process of copying media from one location to another (OS images tend to be pretty large in corporate environments).

This feature is only available through the Glance CLI only and that the templates that you wish to add have the following prerequisites.

1. Ensure that the existing VMs template resides in the same vCenter as your VMware Integrated OpenStack deployment.
2. Verify that the VM template does not have multiple disks.
3. Verify that the VM template does not have a CD-ROM drive.
4. Verify that the VM template does not have a floppy disk drive.

The high-level steps that need to be performed to create an image are as follows:

1. Create an OpenStack image (with no location)
2. Note the image ID and specify a location pointing towards the vSphere template
3. In the images section of the Horizon dashboard for example, might display a new image called corporate-windows-2012-r2, from which instances can be launched.

Note: cloud admins will have to make sure those OS images have the cloud-init package installed on them before they can be fully used in the OpenStack environment. If cloud-init needs to be installed, this can be done either pre- or post- the import process into Glance.

A link to a video demonstrating the process is available here: https://player.vimeo.com/video/190155185

6.5 Best Practices for Glance

6.5.1 Create a Common Naming Convention for stored images in glance

As images take up a resonably large amount of storage space on filesystems it is therefore recommended to try to reduce the amount of duplication of these images that occurs. VMware recommends that you try to work with all parties to ensure that you have a consistent naming policy of images to reduce the amount of duplicated images that are stored. Remember that this will be your organisations image repository, and it makes sense to keep the repository to a minimum size. VMware recommend to have a standard naming convention, thereby all tenants and users can find an image that is available to them easily and quickly.

6.5.2 Glance Image Conversion Best Practices

To import QCOW2 native cloud images into VMware Integrated OpenStack with ease, there are number of recommended best practices. Best practices are designed to workaround problems caused by older upstream tooling or simple adjustments required in the cloud image to match the vSphere environment. Specifically:

- Some storage vendors need StreamOptimized image format.
- Guest Images are attempting to write boot log to ttyS0, but the serial interface is not available on the VM.
- Defects in earlier versions of the qemu-img tool while creating streamOptimized images.
- DHCP binding failure caused by Predictive Network Interface Naming.

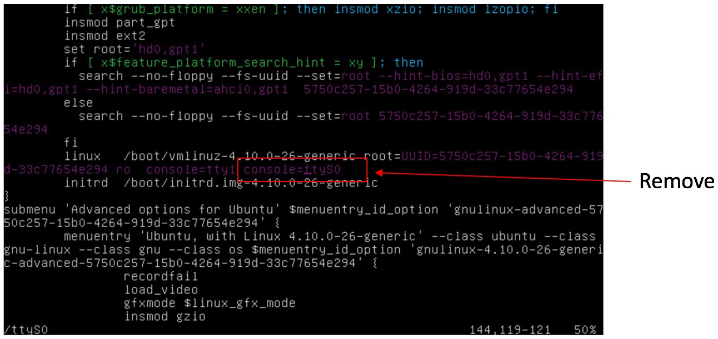
1. VIO 3.x and earlier, serial console output is not enabled. When booting an image that requires serial console support, use libguestfs to edit the grub.cfg and remove all references to "console=ttyS0". Libguestfs provides a suite of tools for accessing and editing VM disk images. Once installed the "guest mount" command-line tool can be used to mount qcow2 based images. By default, the disk image mounts in read-write mode. More info on Libguestfs here

    # guestmount -a xxx-cloudimg-amd64.img -m /dev/sda1 /mnt

    # vi /mnt/boot/grub/grub.cfg

    # umount mnt

See below screen Capture:



Capture 6.2: Disk image mounts

2. VMware vSAN requires all images to be in streamOptimized format. When converting to VMDK format, use the –o flag to specify the subformat as streamOptimized:

    # qemu-img convert -f qcow2 -O vmdk -o subformat=streamOptimized -o adapter_type=lsilogic xxx-server-cloudimg-amd64.img xxx-server-cloudimg-amd64.vmdk ; printf '\x03' | dd conv=notrunc of=xxx-server-cloudimg-amd64.vmdk bs=1 seek=$((0x4))

A few additional items to call out:

a) "lsilogic" is the recommended adapter type. Although it is possible to set the adapter type during image upload into glance, we recommend as a good practice to always set the adapter type as part of the image conversion process.

b) Older versions of the qemu-img tool contain a bug that causes problems with the streamOptimized subformat. The following command can be run after converting an image to correct the problem: printf '\x03' | dd conv=notrunc of=xxx-server-cloudimg-amd64.vmdk bs=1 seek=$((0x4)). It is harmless to execute the printf even if you're using a version of qemu-tools that has the fix: all the command does is set the VMDK version to "3" which correct version of qemu-img will already have done. If you are not sure what version of qemu-tools you have, apply the printf command.

3. In the case of CentOS, Udev rule ln -s /dev/null /etc/udev/rules.d/80-net-name-slot.rules as part of the image bundle is ignored during CentOS image boot up and Predictable Network Interface Naming is enabled as a result. Our recommendation is to disable predictive naming using grub. You can find more information here.

4. Finally, with Cirros QOCW image, preserve the adapter type as 'ide' during the QCOW to VMDK conversion process. This is to avoid an open Upstream bug.

    # qemu-img convert –f qcow2 –O vmdk /var/www/images/cirros-0.3.5-x86_64-disk.img /var/www/images/cirros-0.3.5-x86_64-disk.idk.vmdk

6.6 Configure QoS Resource Alocation for Instances Using Image Metadata

Resource and over-subscription management are always the most challenging tasks facing a Cloud Admin. To deliver a guaranteed SLA, one method OpenStack Cloud Admins have used is to create separate compute aggregates with different allocation / over-subscription ratios. Production workloads that require guaranteed CPU, memory, or storage would be placed into a non-oversubscribed aggregate with 1:1 Over-subscription, dev workloads may be placed into a best effort aggregate with N:1 over-subscription. While this simplistic model accomplishes its purpose of an SLA guarantee on paper, it comes with a huge CapEx and/or high overhead for capacity management / augmentation. Worst yet, because host aggregate level over-subscription in OpenStack is simply static metadata consumed by the nova scheduler during VM placement, not real time VM state or consumption, huge resource imbalances within the compute aggregate and noisy neighbor issues within a nova compute host are common occurrences.

New workloads can be placed on a host running close to capacity (real time consumption), while remaining hosts are running idle due to differences in application characteristics and usage pattern. Lack of automated day 2 resource re-balance (management) further exacerbates the issue. To provide white glove treatment to critical tenants and workloads, Cloud Admins must deploy additional tooling to discover basic VM to Hypervisor mapping based on OpenStack project IDs. This is both expensive and ineffective in meeting SLAs.

Over-subscription works if resource consumption can be tracked and balanced across a compute cluster. Noisy neighbor issues can be solved only if the underlying infrastructure supports quality of service (QoS). By leveraging OpenStack Nova flavor extra-spec extensions along with

vSphere industry proven per VM resource reservation allocation (expressed using shares, limits and reservations), OpenStack Cloud Admins can deliver enhanced QoS while maintaining uniform consumption across a compute cluster.

The VMware Nova flavor extension to OpenStack was first introduced upstream in Kilo and is officially supported in VIO release 2.0 and above. Additional requirements are outlined below:

- Requires VMware Integrated OpenStack version 2.0.x or greater
- Requires vSphere version 6.0 or greater
- Network Bandwidth Reservation requires NIOC version 3 VMware Integrated OpenStack access as a cloud administrator

Resource reservations can be set for following resource categories:

- CPU (MHz)
- Memory (MB)
- Disk IO (IOPS)
- Network Bandwidth (Mbps)

Within each resource category, Cloud Admin has the option to set:

- Limit – Upper bound, not to exceed limit resource utilization
- Reservation – Guaranteed minimum reservation
- Share Level – The allocation level. This can be ‘custom’, ‘high’ ‘normal’ or ‘low’.
- Shares Share – In the event that ‘custom’ is used, this is the number of shares.

Image extra-spec metadata can be set on a Glance images or Nova flavors. Complete Nova flavor extra-spec details and deployment options can be found here. vSphere Resource Management capabilities and configuration guidelines is a great reference as well and can be found here.

Let’s look at an example using Hadoop to demonstrate VM resource management with flavor extra-specs. Data flows from Kafka into HDFS, every 30 minutes there’s a batch job to consume the newly ingested data. Exact details of the Hadoop workflow are outside the scope of the design guide. If you are not familiar with Hadoop, some details can be found here. Resources required for this small scale deployment are outlined below in table 6.1:

Node Type	Core (reserved – Max)	Memory (reserved – Max)	Disk	Network Limit
Master / Name Node	4	16 G	70 G	500 Mbps
Data Node	4	16 G	70 G	1000 Mbps
Kafka	0.4-2	2-4 G	25 G	100 Mbps

Table 6.1: VM resource management with flavor extra-specs

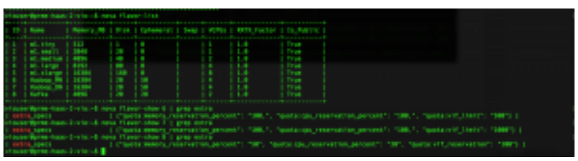
Based on above requirements, Cloud Admin needs to create Nova flavors to match maximum CPU / Memory / Disk requirements for each Hadoop component. Most of OpenStack Admins should be very familiar with this process:

Capture 6.3: Nova Flavor

Based on the reservation amount, attach corresponding nova extra specs to each flavor or to glance images associated with corresponding node type:

Capture 6.4: Flavor Extra Specs

Once extra specs are mapped, confirm setting using the standard *nova flavor-show* command or glance image-show:



Capture 6.5: Flavor Verfication

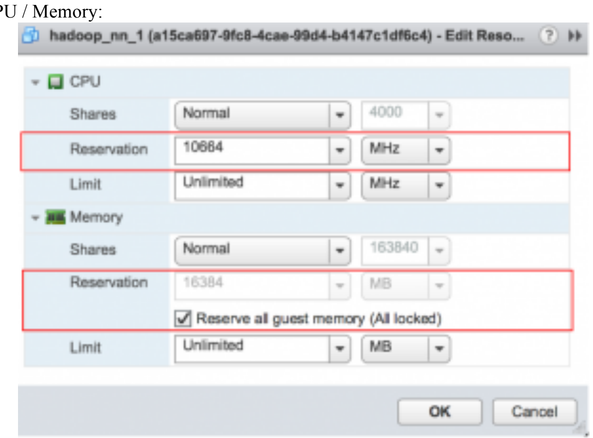
Any new VM consumed using new flavors from OpenStack (API, command line or Horizon GUI) will have resource requirements passed to vSphere (VMs can be migrated using the nova rebuild VM feature).

To set the extra spec on on a glance image, following commands can be used:

```
glance image-update --property quota_cpu_reservation_percent=20 --property quota_memory_reservation_percent=50 --property quota_vif_reservation=100 e5198d04-ceb6-421e-b922-21b702138636

where e5198d04-ceb6-421e-b922-21b702138636 is the UUID of the Image
```

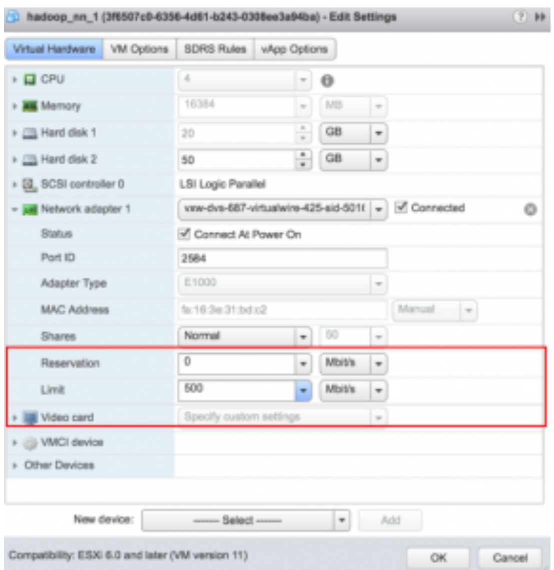
Instead of best effort, vSphere will guarantee resources based on nova flavor extra-spec definition. Specific to our example, 4 vCPU / 16G / Max 1G network throughput will be reserved for each DataNode, NameNode with 4 vCPU / 16G / Max 500M throughput and Kafka nodes will have 20% vCPU / 50% Memory reserved. Instances boot into "Error" state if requested resources are not available, ensuring existing workload application SLAs are not violated. You can see that the resource reservation created by the vSphere Nova driver are reflected in the vCenter interface:

- Name Node CPU / Memory:



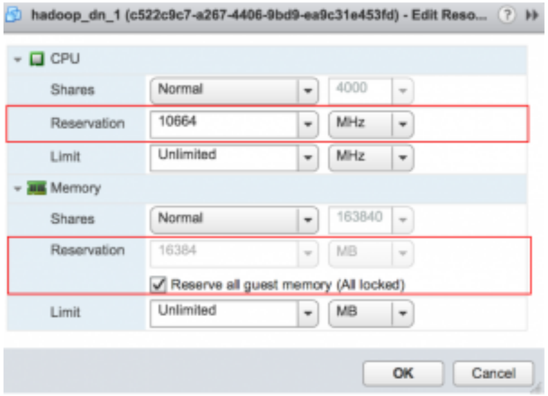
Capture 6.6: Name Node CPU/Memory

- Name Node Network Bandwidth:



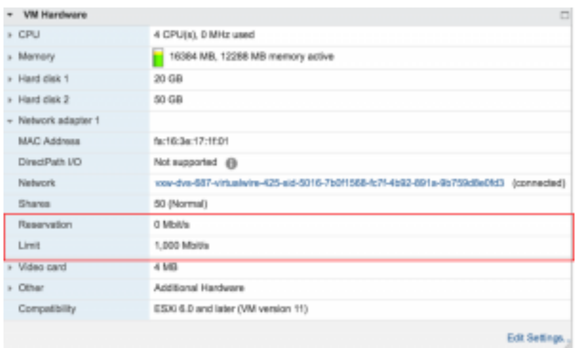
Capture 6.7: Name Node Network Bandwidth

- Data Node CPU / Memory:



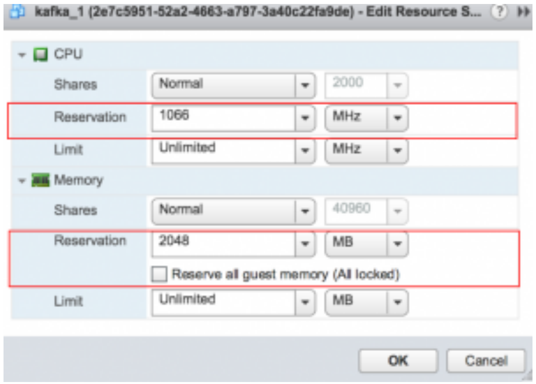
Capture 6.8: Data Node CPU / Memory

- Data Node Network Bandwidth:



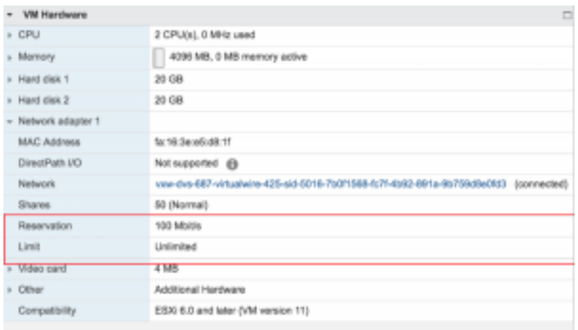
Capture 6.9: Data Node Network Bandwidth

- Kafka Node CPU / Memory:



Capture 6.10: Kafka Node CPU / Memory

- Kafka Network Bandwidth:



Capture 6.11: Kafka Network Bandwidth

vSphere will enforce strict admission control based on real time resource allocation and load. New workloads will be admitted only if SLA can be honored for new and existing applications. Once a workload is deployed, in conjunction with vSphere DRS, workload rebalance can happen automatically between hypervisors to ensure optimal host utilization and avoid any noisy neighbor issues. Both features are available out of box, no customization is required.

Section 7: VIO Availability Zone Design

OpenStack availability zone is a logical subdivision of resources into failure domains. You can define the failure boundary based on physical attributes such as power source, rack location, or data center location. A subdivision could also be grouped to address unplanned as well as planned failure. DevOps engineers can design their applications to take advantage of this grouping to achieve maximum availability for their implementation. The value added by availability zones will depend on the likelihood of failure. Other factors to consider when designing availability zone are:

- The number of Availability zones to manage, and impact on resource utilization. If AZ zones are too granular, per rack for example, it becomes a burden to select the AZ on which to place an application. Since AZ is a method to split resources, managing capacity within each AZ becomes difficult. If there is already high level of redundancy built into a rack, the benefit of using AZ to deal with unplanned failure is marginal.
- Planned maintenance. A Cloud Admin can leverage AZ to work around routine maintenance by aligning AZ implementation to match most common planned outage scenarios. Planned failures are much more common than unplanned. AZ design that aligns maintenance scenarios can be used to offer higher application SLA.
- Tenant application Requirements. This is specific to the type of implementation. An application with a single point of failure cannot take advantage of AZ.
- Network latency and bandwidth. AZ implementation needs to factor in application network latency or bandwidth requirements.

Design availability zones according to the following guidelines:

- Limit the number of AZs to maximize resource utilization within each AZ.
- Use application requirements to determine AZ grouping.
- Have a complete understanding of physical layout and dependencies within each DC

7.1 Nova Availability Zone

Nova ties availability zone implementation to host aggregates. Not all host aggregates are required to be mapped to availability zones. Nova computes not placed in a user defined AZ are placed into default_availability_zone, and the internal_service_availability_zone where other Nova services live. This is how it's defined in VIO (from /etc/nova/nova.conf):

```
#default_availability_zone = nova
#internal_service_availability_zone = internal
```

A single Nova compute host can only belong to a single availability zone. After the host is assigned to an AZ, Nova does not allow the assignment of the same compute host to a different AZ.

For example, compute01 and 02 are added from the VIO management console. By default, they are mapped to default availability zone nova. Nova services live in the internal availability zone.

```
viouser@openstack-client:~/vio2$ nova availability-zone-list
+-----------------------+----------------------------------------+
| Name                  | Status                                 |
+-----------------------+----------------------------------------+
| internal              | available                              |
| |- controller01       |                                        |
| | |- nova-conductor   | enabled :-) 2017-04-25T20:30:54.000000 |
| | |- nova-scheduler   | enabled :-) 2017-04-25T20:30:55.000000 |
| | |- nova-consoleauth | enabled :-) 2017-04-25T20:30:58.000000 |
| |- controller02       |                                        |
| | |- nova-conductor   | enabled :-) 2017-04-25T20:30:54.000000 |
| | |- nova-scheduler   | enabled :-) 2017-04-25T20:30:57.000000 |
| | |- nova-consoleauth | enabled :-) 2017-04-25T20:30:53.000000 |
| nova                  | available                              |
| |- compute01          |                                        |
| | |- nova-compute     | enabled :-) 2017-04-25T20:30:54.000000 |
| |- compute02          |                                        |
| | |- nova-compute     | enabled :-) 2017-04-25T20:30:56.000000 |
+-----------------------+----------------------------------------+
viouser@openstack-client:~/vio2$
```

Capture 7.1: Nova availability zone

You can add compute01 node to a previously created host aggregate with availability zone AZ1:

```
viouser@openstack-client:~/vio2$ nova aggregate-add-host gold-1 compute01
Host compute01 has been successfully added for aggregate 2
+----+--------+-------------------+-------------+---------------------------+
| Id | Name   | Availability Zone | Hosts       | Metadata                  |
+----+--------+-------------------+-------------+---------------------------+
| 2  | gold-1 | AZ1               | 'compute01' | 'availability_zone=AZ1'   |
+----+--------+-------------------+-------------+---------------------------+
viouser@openstack-client:~/vio2$ nova availability-zone-list
+-----------------------+----------------------------------------+
| Name                  | Status                                 |
+-----------------------+----------------------------------------+
| internal              | available                              |
| |- controller01       |                                        |
| | |- nova-conductor   | enabled :-) 2017-04-25T20:36:34.000000 |
| | |- nova-scheduler   | enabled :-) 2017-04-25T20:36:35.000000 |
| | |- nova-consoleauth | enabled :-) 2017-04-25T20:36:38.000000 |
| |- controller02       |                                        |
| | |- nova-conductor   | enabled :-) 2017-04-25T20:36:44.000000 |
| | |- nova-scheduler   | enabled :-) 2017-04-25T20:36:37.000000 |
| | |- nova-consoleauth | enabled :-) 2017-04-25T20:36:43.000000 |
| AZ1                   | available                              |
| |- compute01          |                                        |
| | |- nova-compute     | enabled :-) 2017-04-25T20:36:44.000000 |
| nova                  | available                              |
| |- compute02          |                                        |
| | |- nova-compute     | enabled :-) 2017-04-25T20:36:37.000000 |
+-----------------------+----------------------------------------+
viouser@openstack-client:~/vio2$
```

Capture 7.2: Adding a node to a previously created availability zone

You can create a second AZ2, and add compute02 to the new AZ:

```
viouser@openstack-client:~/vio2$ nova aggregate-create gold-2 AZ2
+----+--------+-------------------+-------+-------------------------+
| Id | Name   | Availability Zone | Hosts | Metadata                |
+----+--------+-------------------+-------+-------------------------+
| 8  | gold-2 | AZ2               |       | 'availability_zone=AZ2' |
+----+--------+-------------------+-------+-------------------------+
viouser@openstack-client:~/vio2$ nova aggregate-add-host gold-2 compute02
Host compute02 has been successfully added for aggregate 8
+----+--------+-------------------+-------------+-------------------------+
| Id | Name   | Availability Zone | Hosts       | Metadata                |
+----+--------+-------------------+-------------+-------------------------+
| 8  | gold-2 | AZ2               | 'compute02' | 'availability_zone=AZ2' |
+----+--------+-------------------+-------------+-------------------------+
viouser@openstack-client:~/vio2$
```

Capture 7.3: Adding a new node to a newly created availability zone

If you attempt to add compute01 to AZ2, it is rejected. From a scheduling perspective, VIO does not set the default availability zone

Availability zone to use when user doesn't specify one (string value)
default_schedule_zone = **<None>**

When the default availability zone isn't set, users who don't care about availability zones can omit the availability zone flag from their API calls. Nova scheduler places the workload to any availability zone that has sufficient capacity.

7.2 Neutron Availability Zone

In VIO 3.0 and later, NSXV plug-in supports neutron availability zones. This means that Cloud Admins can configure several sets of ESXi Edge resource pools and corresponding datastores. For each network or router, OpenStack users can choose where to deploy the NSX edge device. I n the nsxv.ini file, under [nsxv], a Cloud Admin can add a new availability_zones parameter, with the following convention:

<availability zone name>:<resource pool id>:<datastore id>:<Edge HA enabled true/false>:<HA datastore id> where:

- resource pool id is the vSphere resource pool created on the ESXi Edge cluster (From vCenter Home Host and Cluster Resource pool ID embedded in the URL)
- datastore id is the vSphere datastore ID (From vCenter Home Storage Storage Device ID embedded in the URL)
- HA datastore id is optional, and relevant only if the parameter edge_ha is set to True.

The following example shows one possible configuration:

[nsxv]

availability_zones = zone1:resgroup-1:datastore-1:true:datastore-2,zone2:resgroup-2:datastore-3:false

OpenStack Cloud admin defined defined 2 neutron availability zones: zone1 & zone2. When choosing zone1, the NSX edge appliance will be deployed on vSphere resource pool 'resgoup-1', and datastores 'datastore-1' & 'datastore-2' (because the edge_ha parameter is true). When choosing zone2, the edge will be deployed on vSphere resource pool 'resgoup-2' and datastore 'datastore-3' without ha datastore since edge_ha is false.

In VIO 4.0 and later, this configuration is made simpler. More parameters are added per availability zone to support a wider array of use cases. The availability_zones parameter contains only names of the neutron AZ, not resources.

[nsxv]

availability_zones = zone,zone2zone3

For each of these availability zones, there is a new dynamic section that contains all the relevant parameters for the zone. Most of the parameters are optional, and when not stated, the global value is used.

```
[az:zone1]
 resource_pool_id = resgroup-## <mandatory>
datastore_id = datastore-## <mandatory> edge_ha =
True/False <default False>
 ha_datastore_id = datastore-## <mandatory if the edge-ha of this az is True> ha_placement_random =
True/False <optional or global value will be used>
backup_edge_pool = service:compact:#:#,vdr:compact:#:# <optional or global value will be used>
mgt_net_moid = network-## <optional or global value will be used>
mgt_net_proxy_ips = x.x.x.x <list of IPs. optional or global value will be used> mgt_net_proxy_netmask =
255.0.0.0 <or a different net mask. optional or global value will be used> mgt_net_default_gateway = x.x.x.x
<ip. optional or global value will be used>
external_network = network-## <optional or global value will be used>
vdn_scope_id = vdnscope-## <optional or global value will be used> dvs_id =
dvs-## <optional or global value will be used> datacenter_moid = datacenter-#
<optional or global value will be used>

[az:zone2]
 …
[az:zone3]

...
```

For backward compatibility, earlier convention defined in VIO 3.x still works. Below is a list of Neutron Client and API

Support for availability zone:

- View all the availability zones for networks and for routers:
  neutron availability-zone-list
- Create a network with availability zone
  neutron net-create --availability-zone-hint zone2 net2
- Create a router with availability zone
  neutron router-create --availability-zone-hint zone1 retry

The default neutron availability zone is created automatically. All routers and networks without a hint are created on the default zone, which is mapped to the global AZ setting as configured in the nsxv.ini file. Until a NSX edge is actually deployed, the router-show and net-show commands do not show its availability zones, and display only the availability zone hints.

7.3 Example of Availability Zone Implementation

The following diagram (Figure 5.2) is an example of multi availability zone design. In this example, a customer has two data centers, DC1 and DC2. Each data center has a dedicated set of ESXi clusters for Cluster compute and edge. The Management Cluster is located in DC1, but spans across both DC1 and DC2 if sufficient network bandwidth is available, and the network latency is low.

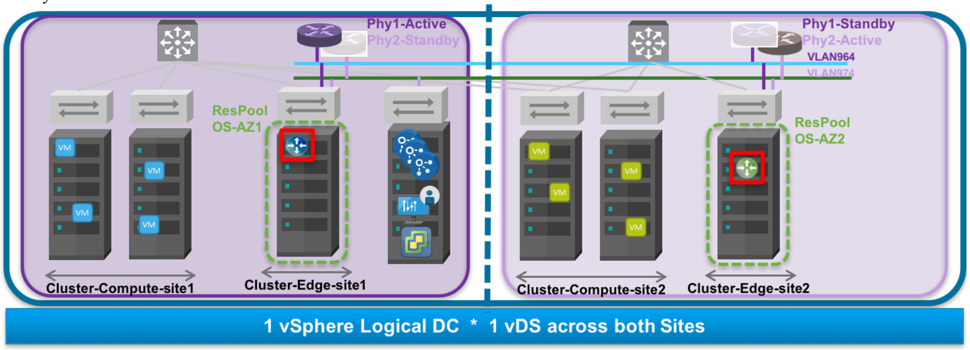


Figure 7.2: Multi Availability Zone Design

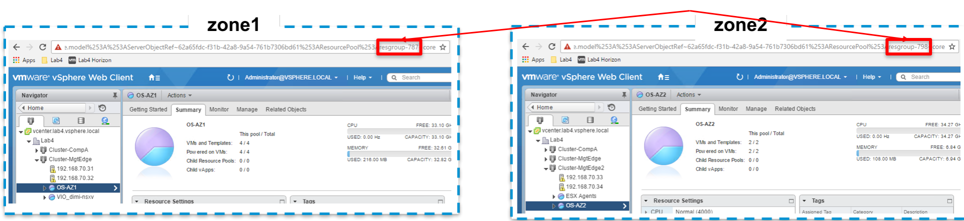
Two nova compute availability zones (Table 7.1) are created, zone-DC1 and zone-DC2.

Host Aggregates

Name	Availability Zone	Hosts
cluster-compute-site1	zone-DC1	compute01
cluster-compute-site2	zone-DC2	compute02

Table 7.1: Host Aggregates

The Neutron availability zone information is updated in the OMS server and pushed to VIO controllers. The resource group and datastore IDs are obtained from the vCenter Server URL.



Capture 7.4: Host aggregates in availability zone

To configure the Availability Zones information on the VIO-Mgr, edit the Ansible template file

/var/lib/vio/ansible/roles/neutron-server/templates/etc/neutron/plugins/vmware/nsxv.ini

[nsxv]

availability_zones=zone-DC1:resgroup-787:datastore-34,true,datastore-333,zone-DC2:resgroup-798:datastore-799

As a side note, if we keep storage isolated to an single ESXi host within Cluster-MgmtEdge:

- 192.168.70.31 datastore-34
- 192.168.70.32 datastore-333

storage availability will force NSX edge active/standby to be placed on different ESXi hosts.

Using OpenStack neutron and nova commands, you can create the following logical topology (Figure 7.3):

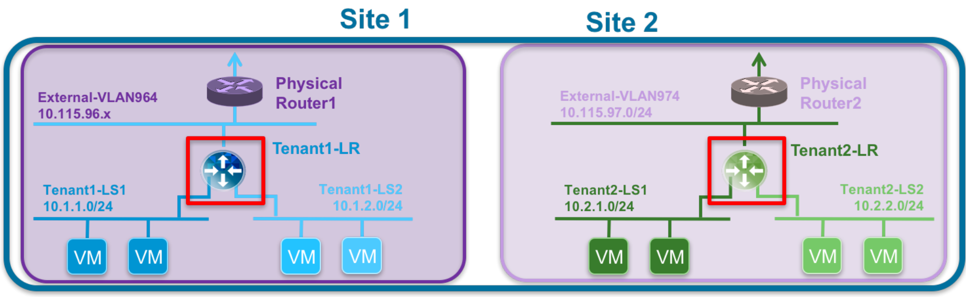


Figure 7.3: Availability Zone Logical Topology

```
# neutron router-create --router_type=exclusive --availability-zone-hint zone-DC1 Tenant1-LR
# neutron router-create --router_type=exclusive --availability-zone-hint zone-DC2 Tenant2-LR
# neutron net-create --availability-zone-hint zone-DC1 Tenant1-LSx
# neutron subnet-create --name Tenant1-LSx Tenant1-LSx 10.1.1.0/24
# neutron net-create --availability-zone-hint zone-DC2 Tenant2-LSx
# neutron subnet-create --name Tenanty-LSx Tenant2-LSx 10.2.x.0/24 where x is
an integer
nova boot --flavor m1.small --image ubuntu-14.04-server-amd64 --availability-zone zone-DC1 --nic net-id=UUID Tenant1-LSx VM1 nova boot
--flavor m1.small --image ubuntu-14.04-server-amd64 --availability-zone zone-DC2 --nic net-id=UUID Tenant2-LSx VM2
```

If you create separate external networks for each tenant (Figure 7.4), you can also work with Network Admins to ensure that default GW for each external network are always local to the physical router within the AZ.
This ensures that traffic in or out of DC is always localized to the physical router within the same data center.

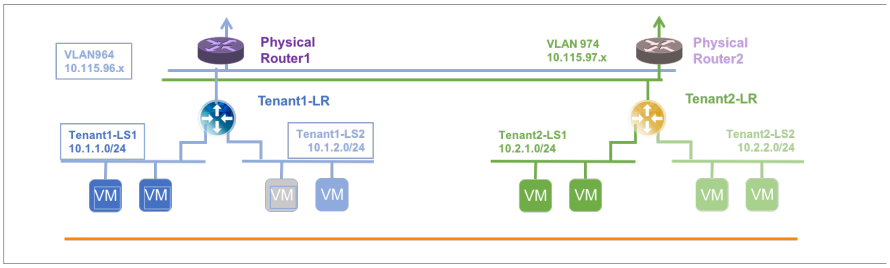


Figure 7.4: Multi Availability Zone Design with separate external networks for each tennant

You can also extend L2 between DC by splitting Edge cluster across both DC (Figure 7.5):

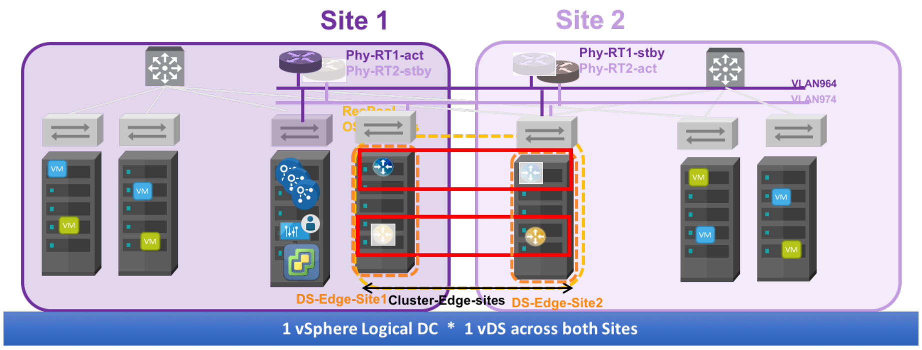


Figure 7.5: Multi Availability Zone Design by splitting edge clusters

Section 8: VIO Automation and Orchestration

Infrastructure-as-code is one of the many reasons customers adopt OpenStack, Several popular tools are available for enabling IAC:

- Packer
- Ansible
- Terraform
- Heat
- There's also Chef, Puppet, and SaltStack.

Differences between each tool can be generalized to following:

- Configuration Management vs Orchestration - Ansible, Chef and SaltStack are examples of Configuration management tool. Configuration management tool are designed to install and manage packages on a server. While Heat and Terraform are examples of Orchestration tool.
- Mutable vs Immutable Infrastructure - Mutable infrastructure are infrastructure that relies on Configuration management tool to update when requirements change. Immutable infrastructure assumes infrastructure rebuild when Infrastructure updates are required.

Packer is used to create virtual machine images, similar to Docker with container images, which can be consumed by Orchestration tools such as HEAT or Terraform to build the infrastructure. Ansible (along with Chef, Puppet, and SatStack) is a Configuration Management tool that can be used to deploy software packages and updates. Ansible modules have idempotence baked in. Depending on the combination of tools used, resulting infrastructure is either mutable or immutable. If Packer builds Golden images (Image inclusive of all application templates), Terraform to deploy all Golden images, resulting infrastructure can be considered immutable. When images are updated,

Terraform is used to redeploy based on the new VM image. If a customer relies on baseline images (created by Packer, Image builder, or some other tool), Terraform or Heat to bring up the infrastructure, leverages Ansible to deploy packages on top of the baseline images. The resulting infrastructure is generally considered mutable. When package updates are required, instead of updating the VM image and redeploying, the Ansible playbook is used to upgrade the infrastructure. Configuration drift is the most common concern with a mutable infrastructure. Immutable infrastructures require a much more disciplined automation and CICD process.

All of these tools can be used to manage infrastructure as code. All of them are open source, used by many of VIO customers, and work with various cloud providers. All of them are well documented and supported by a community of users.

We're going to provide an overview into each tool and provide working examples of how you can leverage these tools to consume VIO Infrastructure.

8.1 VIO Heat Orchestration

Heat is the orchestration piece of OpenStack. It allows users to describe deployments of cloud applications in YAML text files called templates. These templates are then parsed and executed by the Heat engine. Heat accepts AWS CloudFormation templates and its own set of compatible syntaxes. Like all OpenStack services, Heat can be consumed through direct API but for simplicity, most OpenStack users consume Heat using a standalone command line client or the Horizon web-based client.

A typical Heat Orchestration Template has following components:

- Version. A mandatory field used to specify the version of the template syntax that is used.
- Description. An optional component, used to describe the template.
- Parameters. A list of inputs typically pre-created static resources outside of HEAT template.
- Resources. The actions to be taken in the template, this is where the different OpenStack operations are defined. Infrastructure resources. These include servers, floating IPs, volumes, security groups, users, and so on.
- Outputs. Defines what attributes to export after successful deployment.

- Conditions - Includes statements which can be used to restrict when a resource is created or when a property is defined. They can be associated with resources and resource properties in the resources section, also can be associated with outputs in the outputs sections of a template (Newton Feature).

The following is an example of a simple heat template Example 1

```
heat_template_version: 2013-05-23

description: >
  demo template for setting up public and private network under nsxv env.
parameters:
  public_net:
    type: string
    default: 42f8b1b5-0caa-44bd-ad7e-0fce64d8e94a
  private_cidr:
    type: string
    default: "10.0.30.0/24" description:
  cidr of private subnet image:
    type: string
    default:      ubuntu-14.04-server-amd64
  description:   Image   used   for   servers
  flavor:
    type: string
    default: m1.small
    description: flavor used by the web servers
  key_name:
    type: string
    default: demo-keypair

resources:
   router:
    type: OS::Neutron::Router
    properties:
      name: nsxv_router_demo
      value_specs: {"router_type": "exclusive"}


  private_network:
    type: OS::Neutron::Net
    properties:
      name: "vxlan_net"
```

```
  private_subnet:
    type: OS::Neutron::Subnet
    properties:
      name: "private_subnet"
      network_id: { get_resource: private_network}
      ip_version : 4
      cidr: {get_param: private_cidr} dns_nameservers:
      ["10.20.20.1"]
  router_interface:
    type: OS::Neutron::RouterInterface
    properties:
      router_id: {get_resource: router} subnet_id:
      {get_resource: private_subnet}
  router_gateway:
    type: OS::Neutron::RouterGateway
    properties:
      router_id: {get_resource: router}
      network_id: {get_param: public_net}

outputs:
  server_networks:
    description: The networks of the deployed server value: {
    get_resource: private_network }
```

This particular example contains three top-level sections:

heat_template_version. 2013-05-23 is the first release. A later version can be 2017-02-14 or later description. This particular templates creates neutron tenant network and router resources. The number of Neutron resources are used in this example:

- type: OS::Neutron::Router
- type: OS::Neutron::Net
- type: OS::Neutron::Subnet
- type: OS::Neutron::RouterInterface
- type: OS::Neutron::RouterGateway

Each resource is associated with a set of properties. Properties are basically variables. Following are various common ways to retrieve and set properties:

- Parameters (retrieved using get_param). Each parameter has an associated type. you can optionally assign a default value to a parameter if applicable.
- Resources created in previous steps (retrieved using get_resource) Hardcoded string, list or integer

As deployments gets more complex, timing and ordering of resource creation becomes extremely important. Heat has an understanding of implicit dependency between resources. Environmental differences can have huge influences on how quickly a resource can be created, depend_on clause is used to define explicit dependency. With the depend_on clause, Heat operations can be delayed until required resources are available. The following is an example of the depend_on clause:

Example 2:

```
lb:
  type: OS::Neutron::LBaaS::LoadBalancer properties:
    admin_state_up: true
    description: VIO LB
    name: VIO
    vip_subnet: {get_resource: private_subnet}
  depends_on: [ router_interface ]

lb_listener:
  type: OS::Neutron::LBaaS::Listener
  properties:
    loadbalancer: {get_resource: lb} name:
    web_port
    protocol: HTTP
    protocol_port: 80

lb_pool:
  type: OS::Neutron::LBaaS::Pool
  properties:
    admin_state_up: true
    description: LB pool
    listener: {get_resource: lb_listener}
    lb_algorithm: ROUND_ROBIN protocol:
    HTTP

lb_vip_floating_ip:
  type: OS::Neutron::FloatingIP
  properties:
    floating_network_id: { get_param: public_net }
    port_id: { get_attr: [lb, vip_port_id] }
```

Example 2 is a continuation of Example 1. The VIO load balancer is not created until the required subnets are attached to an exclusive router. This is due to load-balancer dependency to router, before a load balancer can be created, a router must exist. The example uses the following statement to force or delay load balancer

```
depends_on: [ router_interface ]
```

creation until the association of the IP subnet to the router is complete.

Finally, HEAT output can be any HEAT resource attribute that you want to display. Example 1 displayed private network.

```
outputs:
  server_networks:
    description: The networks of the deployed server
    value: { get_resource: private_network }
```

You can also display VM GuestOS if VMs have been deployed.

```
outputs:
  server_image:
    description: The GuestOS of the deployed server
    value: { get_attr: [server, image] }
```

or web hooks required to trigger scale up/down:

```
outputs:
  scale_up_url:
  description: >
      This URL is the webhook to scale up the autoscaling group. You can
      invoke the scale-up operation by doing an HTTP POST to this URL; no
      body nor extra headers are needed.
    value: {get_attr: [web_server_scaleup_policy, alarm_url]}

  scale_down_url:
    description: >
      This URL is the webhook to scale down the autoscaling group. You can
      invoke the scale-up operation by doing an HTTP POST to this URL; no body
      nor extra headers are needed.
    value: {get_attr: [web_server_scaledown_policy, alarm_url]}
```

A complete working example of a HOT template can be accessed from git@gitlab.com:xiaog/heat-auto-scale.git

8.2 VIO Ceilometer

Ceilometer is frequently deployed in an OpenStack environment to accomplish following three objectives:

- Metering - Collecting information about Nova, Cinder, Neutron, Glance, and Swift Objects
- Alarming - Trigger actions based on policies for auto-scaling based upon data collected
- Billing - based on per user/tenant usage report, assemble billable items into a single customer bill.

Ceilometer supports two methods of data collection (Figure 8.1):

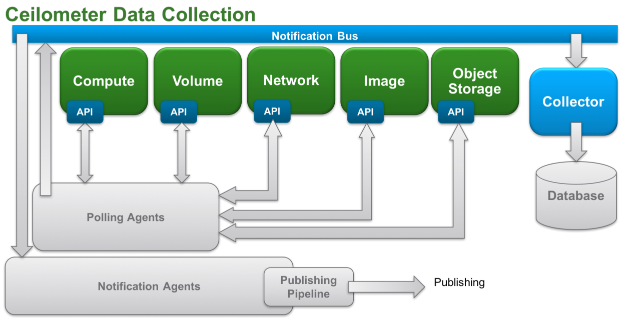


Figure 8.1 Ceilometer Data Collection flow

- Bus listener agent which takes events generated on the notification bus and transforms them into Ceilometer samples. This is the preferred method of data collection. Ceilometer-notification agent monitors the message queues for notifications. The notification daemon loads one or more listener plugins, using the namespace ceilometer.notification. Each plugin can listen to any topics, but by default it will listen to notifications.info. The listeners grab messages off the defined topics and redistributes them to the appropriate plugins(endpoints) to be processed into Events and Samples.
- Polling agents, which is the less preferred method, will poll some API (public REST APIs exposed by services and host-level SNMP/IPMI daemons) or other tool to collect information at a regular interval. Polling for compute resources is handled by a polling agent running on the compute node (where communication with the hypervisor is more efficient), often referred to as the compute-agent. Polling via service APIs for non-compute resources is handled by an agent running on a cloud controller node, often referred to the central-agent. The polling agent daemon is configured to run one or more pollster plugins using either the ceilometer.poll.compute and/or ceilometer.poll.central namespaces.The agents periodically ask each pollster for instances of Sample objects. The frequency of polling is controlled via the pipeline configuration. The agent framework then passes the samples to the notification agent for processing.

The collector daemon gathers the processed event and metering data captured by the notification and polling agents. It validates the incoming data and (if the signature is valid) then writes the messages to Mongo DB. Overall Ceilometer sequence of events can be summarized as below:

1. Listen to events from Ceilometer Agent.
2. Listen to notifications from Nova, Cinder, Glance , Neutron, Swift
3. Store to database. Default is MongoDB for VIO
4. REST API is available to access the collected data

In conjunction with Heat, Ceilometer can be configured to provide threshold alarms. Once alarm crosses preconfigured threshold, notify the Heat engine to scale up or down. This is commonly referred to as Heat Auto Scaling. Below resources are required to enable Heat Auto Scale:

- OS::Heat::AutoScalingGroup - An autoscaling group that can scale arbitrary resources
- OS::Heat::ScalingPolicy - A resource to manage scaling of OS::Heat::AutoScalingGroup
- OS::Ceilometer::Alarm - The resource for defining a Ceilometer alarm
- OS::stack_id - identifier used to tie an OS::Ceilometer::Alarm to an OS::Heat::AutoScalingGroup

refer to Heat AutoScaling with Ceilometer section for details.

8.2.1 Heat AutoScaling with Ceilometer

AutoScale is the ability to provision workloads on demand based on load. OpenStack provides autoscaling feature through Heat and Ceilometer. Ceilometer Alarm is defined using heat Scaling Policy resource to provide alarming (i.e. set and monitor thresholds) and to report back to Heat Engine. Upscaling and Downscaling scheduling groups are created based on threshold alarms. Autoscaling is desirable as it reduces the need to pre-create Virtual Machines in anticipation of the demand. To enable

AutoScaling, both Ceilometer and Heat is required. You can use Heat resources to detect when a Ceilometer alarm triggers and provision or de-provision a new VM depending on the trigger. Heat "stack_id" is used as glue to tie an OS::Ceilometer::Alarm to an OS::Heat::AutoScalingGroup. AutoScaling typically works in conjunction with a Load balancer which distributes excess load between VMs on the autoscaling group. Below is a high-level workflow (Figure 8.2)

:

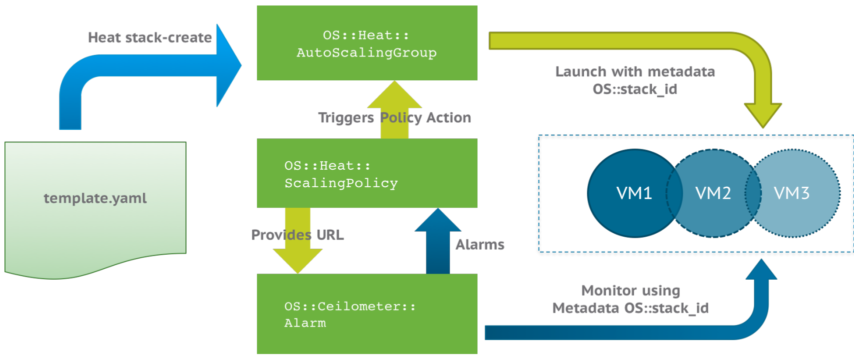


Figure 8.2 Autoscaling Workflow

Autoscaling starts with defining a server group. Within the server group resource, information about VM limit, image flavor, network information and Load balancer pool are mapped to group properties. Below is an example of a heat autoscale group. Max_size parameter is an extremely important setting used to control / limit maximum size of the scaling group. This number should be set based on amount of available capacity. If max is set very high, one badly behaving app can potentially utilize all available capacity. Maximum of 3 VMs is used in our example.

Instead of pointing to OS::Nova::Server to create Virtual Machines, it is possible to point to a different heat template. This reference Heat Template can contain server information regarding post creation jobs, LB membership information, and VM naming convention (passed as properties).

Example 1:

```
web_server_group:
 type: OS::Heat::AutoScalingGroup properties:
   min_size: 1
   max_size: 3
   resource:
     type: web-node-demo.yaml
     properties:
       flavor: {get_param: flavor}
       image: {get_param: image}
       key_name: {get_param: key_name} private_net_id:
       {get_resource: private_network} pool: {get_resource:
       lb_pool}
       public_net_id: {get_param: public_net} subnet_id:
       {get_resource: private_subnet}
       metadata: {"metering.stack": {get_param: "OS::stack_id"}}
```

Once autoscale group is defined, scale up and scale down policies are needed. Many folks place majority of their autoscale effort on capacity expansion, but never properly plan out when to remove excess capacity. Benefits of HEAT autoscale is marginalized when scale down policy isn't set based on actual application profile. To fully realize the benefit of autoscale, it's important to define a scale down policy that:

- is not reactive to smallest change in workloads
- does not break application consistency

Scale down adjustment - defines how many servers to bring up/down during autoscale event.

Example 2:

```
web_server_scaleup_policy: type:
    OS::Heat::ScalingPolicy
    properties:
      adjustment_type: change_in_capacity auto_scaling_group_id:
      {get_resource: web_server_group} cooldown: 30
      scaling_adjustment: 1
  web_server_scaledown_policy:
    type:     OS::Heat::ScalingPolicy
    properties:
      adjustment_type: change_in_capacity auto_scaling_group_id:
      {get_resource: web_server_group} cooldown: 60
      scaling_adjustment: -1
```

Ceilometer alarms are also created using heat. Threshold and period/sample interval are the most important setting to ensure HEAT does not react to smallest change in workloads. With low period/interval value, HEAT will react to any spike in application workload. With large period/interval value, application may run in degraded state up to 2 X sample

```
cpu_alarm_high:
    type: OS::Ceilometer::Alarm
    properties:
      description: Scale-up if the average CPU 50% for 1 minute meter_name:
      cpu_util
      statistic: avg
      period: 60
      evaluation_periods: 1
      threshold: 50
      alarm_actions:
        - {get_attr: [web_server_scaleup_policy, alarm_url]}
      matching_metadata: {'metadata.user_metadata.stack': {get_param: "OS::stack_id"}}
      comparison_operator: gt
```

interval before any action will be taken by HEAT.

Similar to ScaleUp/Down policies, it is critical to define when a threshold alarm should be removed. if ceilometer low threshold alarm are not set, customers may as well not run HEAT autoscale. It will be operationally simpler to always pre-provision capacity based on worst case demand.

```
cpu_alarm_low:
   type: OS::Ceilometer::Alarm
   properties:
     description: Scale-down if the average CPU < 15% for 10 minutes meter_name:
     cpu_util
     statistic: avg
     period: 600
     evaluation_periods: 1
     threshold: 15
     alarm_actions:
       - {get_attr: [web_server_scaledown_policy, alarm_url]}
     matching_metadata: {'metadata.user_metadata.stack': {get_param: "OS::stack_id"}}
     comparison_operator: lt
```

It is possible to integrate heat autoscale up / down with an external monitoring system by exposing the web hook for scale up and down.

URL for web hook can be displayed as part of HOT template output. With URL exposed, customers can use curl or favorite REST client to create/ remove virtual machines from the application stack.

```
outputs:
  scale_up_url:
  description: >
      This URL is the webhook to scale up the autoscaling group. You can
      invoke the scale-up operation by doing an HTTP POST to this URL; no
      body nor extra headers are needed.
    value: {get_attr: [web_server_scaleup_policy, alarm_url]}

  scale_down_url:
    description: >
      This URL is the webhook to scale down the autoscaling group. You can
      invoke the scale-up operation by doing an HTTP POST to this URL; no body
      nor extra headers are needed.
    value: {get_attr: [web_server_scaledown_policy, alarm_url]}
```

Below is an example of manually Scale down:

```
bash-3.2$ curl -X POST -i "http://10.28.228.61:8000/v1/signal/arn%3Aopenstack%3Aheat%3A%3A171fbc0def834875897d4d701e49cd54
LLnYbrdpgoN1kR4zIrT0k%3D"
```

Example to manually scale up:

```
bash-3.2$ curl -X POST -i "http://10.28.228.61:8000/v1/signal/arn%3Aopenstack%3Aheat%3A%3A171fbc0def834875897d4d701e49cd54
RkjIHv4JpSYKLhFaO4H4%3D"
```

8.3 Terraform

Terraform is a tool for building, changing, and versioning infrastructure safely and efficiently. The key features of Terraform are:

- Execution Plans. Terraform has a planning step where it generates an execution plan. The execution plan displays action to be taken after Terraform apply is executed.
- Resource Graph. Terraform builds a graph of all your resource dependencies. Based on the dependency graph, Terraform will parallelizes the creation and modification of resources.
- Change Automation. Terraform is aware of any state it created in the past. .tfstate is used to track details about the newly created resource. When complex changeset is required, Terraform gets the ID from the .tfstate file and compare that to what already exists. Terraform then figures out what changed, and the order in which changes are applied will be based on the .tfstate file.

Using combinations of the Execution Plan, Resource Graph, and Change Automation, operators know what will happen before any action takes place.

Terraform is a cloud orchestration tool, and not a configuration management tool. It is possible to integrate Terraform with a configuration management tool such as Ansible. Using Terraform provisioners, an operator can execute scripts or Configuration Management playbooks on a local or remote machine as part of resource creation or deletion.

Unlike Cloud SDKs such as Libcloud, Jcloud, Fog, and so on, Terraform is not intended to give low-level programmatic access to providers. Terraforms uses a plug-in based model to support various cloud providers using higher level abstraction to describe resource and service stitching. Terraform provides a declarative approach to Infrastructure as Code, where code specifies the desired end state, and Terraform figures out how to achieve that state using resource graphs and execution plans.

8.3.1 Terraform Code Structure

Terraform code template includes following components:

- Resource. A component that must exist in the infrastructure, such as a compute instance, tenant network, router, or load balancer.
- Provider: The infrastructure provider, such as OpenStack or AWS.
- Provisioner: After the resource is created successfully at the provider's end, a provisioner can execute a local command, invoke Chef or Ansible to execute a remote command.

A typical OpenStack provider includes following sets of attributes:

```
provider "openstack" {
  user_name = "${var.openstack_user_name}"
  tenant_name = "${var.openstack_tenant_name}"
  password = "${var.openstack_password}" auth_url
  = "${var.openstack_auth_url}" insecure =
  "${var.insecure}"
}
```

Terraform will concatenate all the .tf files in the folder together. One thing to bear in mind a variable can only be defined once in the folder that is being worked on.   Terraform will error if a variable is defined in multiple files in the worked on folder.

So if you had something like this:

```
terraform
├── provider.tf
├── main.tf
├── deploy.tf
└── variables.tf
```

If both variable.tf and deploy.tf contained

```
variable "foo" { default = "foo" }
```

This would cause Terraform to error as variable foo is defined twice. It is recommended to group all variables into a single TF variable file.

For larger deployments, with multiple sites, it's recommended to separate each location into different working folders and leverage Terraform modules to avoid code duplication. By separating into different folders, Terraform creates a separate tfstate file for each part of the infrastructure. Modules in Terraform are folders with Terraform files used to modularize and encapsulate groups of resources in your infrastructure. The name of the module is simply a reference, you may name modules whatever you'd like. The only required key is source, which tells Terraform where this module can be downloaded from. Valid source values are

- Local file paths
- GitHub
- Bitbucket
- Generic Git
- Mercurial repositories
- HTTP URLs
- S3 buckets

Terraform module section covers each source in detail.

The following is an example of 3 deployment sites, Dev, NonProd, and Prod, each module simply points to the root module to create and control resources.

```
bash-3.2$ ls
dev nonprod prod terraform
bash-3.2$ ls -lt
drwxr-xr-x  16 xgao  staff  544 Jun 25 23:56 terraform
drwxr-xr-x   7 xgao  staff  238 May 22 00:44 prod
drwxr-xr-x   4 xgao  staff  136 May 22 00:32 dev
drwxr-xr-x   4 xgao  staff  136 May 22 00:32 nonprod

bash-3.2$ cd dev
bash-3.2$ more deploy.tf
module "dev" {
  source = "../terraform"
}
bash-3.2$ cd nonprod
bash-3.2$ more deploy.tf
module "nonprod" {
source = "../terraform"
}
bash-3.2$ cd prod
bash-3.2$ more deploy.tf
module "prod" {
  source = "../terraform"
}
bash-3.2$
```

Not required in our example, if Dev, NonProd, and Prod each have different SLA or flavor requirements, we can use inputs or variables to control virtual machine CPU or Memory consumption.

```
bash-3.2$ more deploy.tf

module "prod" {

  source = "../terraform"

  count = 10

}


bash-3.2$ more deploy.tf

module "nonprod" { source

= "../terraform" count = 4

}

bash-3.2$

bash-3.2$ more deploy.tf

module "dev" {

  source = "../terraform"

}
```

"Terraform get" will download the module from source into a local .terraform folder. This folder should not be committed to version control.
The .terraform folder is created relative to your current working directory regardless of the dir argument given to this command. If a module is already downloaded and the -update flag is not set, Terraform will do nothing. As a result, it is safe (and fast) to run this command multiple times.

```
bash-3.2$ terraform get

Get: file:///Users/xgao/terraform/terraform

bash-3.2$ terraform apply

module.prod.openstack_compute_floatingip_v2.fip.1: Creating...
```

Using the provisioner module, Terraform can execute a set of system or shell commands against the provisioned resource. Usually, developers want to execute a set of configuration management playbooks against newly created VMs.

While it is feasible to configure the provisioner to call the playbook directly in the following manner:

```
provisioner "local-exec" {
            command = "ansible-playbook -i '${openstack_compute_floatingip_v2.fip.*.address},' --private-key
${var.private_key_path}                   -e ansible_role_path='${var.role_path}' -s ${var.playbook_path} -T 300"
    }
```

The problem you might encounter is that any error in your configuration management playbook will result in a failed Terraform state, and might require manual intervention for correction. The recommendation is to use the provisioner to build the configuration management inventory and launch the playbook against the inventory post infrastructure creation.

```
  provisioner "local-exec" {
    command =  "echo \"\n[web]\" > tmp/inventory"
  }
  provisioner "local-exec" {
    command = "echo \"${join("\n",formatlist("%s ansible_ssh_host=%s", openstack_compute_instance_v2.web.*.name,
openstack_compute_floatingip_v2.fip.*.address))}\" >> tmp/inventory"
  }
  provisioner "local-exec" {
    command =  "echo \"\n[db]\" >> tmp/inventory"
  }
  provisioner "local-exec" {
    command = "echo ${openstack_compute_instance_v2.db.name} ansible_ssh_host=${openstack_compute_floatingip_v2.fipdb.address} >>
tmp/inventory"
  }
  provisioner "local-exec" {
    command =  "echo \"\n[3tier-cluster:children]\nweb\ndb\" >> tmp/inventory"
  }
}
ansible-playbook --key-file ~xiaog/.ssh/id_rsa -u ubuntu -i tmp/inventory ansible/web.yml
```

You can obtain a working example of Terrafrom code from git@gitlab.com:xiaog/terraform.git

8.4 Packer

Packer is an open source tool for creating identical machine images for multiple platforms from a single source configuration. When building images, Packer is able to use basic shell commands or tools like Ansible or Puppet to install software onto the image. Packer supported platforms include AWS EC2, Docker, GCE, OpenStack, VirtualBox and VMware. The outputs produced by Packer are called artifacts.

Packer can be used to address following Operating System Image requirements:

- Building base images for your application infrastructure. You can use Packer to create an image that contains all the dependencies, monitoring software, and security patches required to run one or all your applications. Then, you can push the image out to your infrastructure and run a configuration management system on top for use case specific tweaking.
- Golden images. A golden image is an immutable image tied to a specific software version and may be reused across multiple applications without much customization.

In order to build a machine image, the following two files are common:

- JSON template file. A Packer specific file that defines an image build
- Provisioning scripts: A folder with the provisioning scripts used to customize the image

Because Packer needs administrator-level credentials, it's generally a good idea to store all credential info outside of JSON template or provisioning scripts. In case of VIO, you can maintain a local copy of the OpenStack.rc file, and source the file prior to any packer operations.

JSON template file consists of two sections:

- Builders
- Provisioners

8.4.1 Builders

The builders section specifies the format and the instructions on how to build an image across different platforms. Packer supports a number of builders for different target platforms including OpenStack, Amazon EC2 AMI images, VirtualBox, and VMware. You can include multiple types of providers in a single Packer builder, allowing a Cloud Admin to simultaneously update an image across all environments, reducing the likelihood of image snowflake.

```
{
      "type": "virtualbox-iso",
      "name": "jessie-vboxiso",
      "headless": "false",
      .....
   },

   {
      "type": "vmware-iso",
      "name": "jessie-vmwareiso",
      "headless": "false",
      ......
   },

   {
      "type": "openstack",
      "name": "jessie-awsebs",
      "image_name": "jessie-openstack",
      "source_image": "ac6903d0-f3bd-4d2a-bed6-f49f6e3d3046",
      .....
   }
 ],
```

Specific to OpenStack, the builder:

- Takes a source image
- Creates a temporary keypair that provide temporary access to the server while the image is being created
- Boots a VM using the source image and temporary keypair

- Runs any provisioning necessary on the Virtual Machine after launching it
- Creates a new reusable image, and load into OpenStack Glance.

This reusable image can then be used as the foundation of new servers that are launched within OpenStack. The Packer does *not* manage images lifecycle. It's up to the Cloud administrators to maintain revision control and track all image updates using tools such as GIT or similar.

The following is an example of an OpenStack builder:

```
"builders": [{
   "type": "openstack",
   "image_name": "ubuntu-14.04-server-amd64-version50",
   "source_image": "ac6903d0-f3bd-4d2a-bed6-f49f6e3d3046",
   "flavor": "2",
   "networks": ["0d443c38-f2b8-4596-97ef-fe1a6a6c9f81"],
   "security_groups": ["default"],
   "insecure": "True", "ssh_username":
   "ubuntu", "floating_ip_pool":
   "provider-vlan", "domain_name":
   "Default"
  }
```

8.4.2 Provisioners

Provisioners provide a way to configure a base image such that a new custom image can be created. Provisioners prepare the system for use, so common use cases for provisioners include the following:

- Installing packages
- Patching the kernel
- Creating users
- Downloading application code

Many provisioners are available, including shell provisoners and provisioners that use DevOps tools such as Ansible or Puppet. The following are examples of shell and Ansible provisioner

```
"provisioners": [{ "type": "ansible-local", "playbook_file": "configure-ami.yml" }]
```

```
"provisioners": [{
  "type": "shell",
  "inline": [
  "sleep 30",
    "sudo apt-get update",
    "sudo apt-get install -y nginx"
  ]
 }]
```

Packer also supports Windows images. Because we are working with Windows images, we need to ensure tools and connection details align with the Windows ecosystem. Windows Remote Management is most frequently used to access Windows servers. It uses WS-Manageement Protocol, a standard Simple Object Access Protocol (SOAP) protocol to access and exchange management information across an IT infrastructure. The communicator type winrm corresponds to the Windows Remote Management feature.

The following is an example of the windows communicator settings.

```
"builders": [{ ... "communicator": "winrm", "winrm_username": "Administrator", "winrm_password": <password>, ... }],
```

WinRM access session information is required in the form of user data script, which is a Windows PowerShell script that configures various settings required to allow remote connectivity through winRM. Settings included in the user data script includes users, passwords, and an execution policy so that Packer is able to connect to the instance after it is created. The following is an example of user data script taken from her e:

```
#ps1_sysnative wmic UserAccount set PasswordExpires=False net user Administrator uUteQ419EPFUMoE4zaTE cmd /C netsh advfirewall set allprofiles state off winrm quickconfig -q winrm set winrm/config/winrs '@{MaxMemoryPerShellMB="500"}' winrm set winrm/config '@{MaxTimeoutms="1800000"}' winrm set winrm/config/service '@{AllowUnencrypted="true"}' winrm set winrm/config/client/auth '@{Basic="true"}' winrm set winrm/config/service/auth '@{Basic="true"}' net stop winrm net start winrm Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope LocalMachine -Force
```

The reference to the userdata script needs to be added to the builders section:

```
"builders": [{ ... "user_data_file": "./userdata_setup.ps1", ... }],
```

For a working example of Packer Windows workflow, go here. For a complete working example of building a Debian image using multiple providers, go to GitHub.

Section 9: VIO Operational Maintenance

To ensure continuous operation of your VMware Integrated OpenStack, it is critical to understand and leverage tools available to monitor and troubleshoot your OpenStack Environment. VIO Operational Maintenance section will discuss following topics:

- VIO Maintenance and VIOCLI Commands
- VIO Monitoring and Logging
- VIO Backup and Restore

9.1 VIO Backup and Restore

Backup of all the VMware Integrated Openstack components is critically necessary for a restore of the deployment to its working state in the event of a failure. The VMware Integrated Openstack management server database backup and openstack_db backup contain configuration and other critical file backups required for recovery. The OMS is taken offline during the backup. The VMware Integrated Openstack, openstack_db backup, contains cinder, glance, heat, keystone, neutron, nova details.Your backup frequency and schedule might vary based on your business needs and operational procedures. We recommend taking VMware Integrated Openstack backups frequently during times of frequent configuration changes. VMware Integrated Openstack backups can be taken on demand or on an hourly, daily, or weekly basis.
It is recommended taking backups in the following scenarios:

- Before a Patch or Upgrade
- After a Patch or Upgrade
- After Day Zero Deployment and initial configuration
- Before Infrastructure changes
- After Infrastructure changes
- Before Topology changes After Topology changes
- After any Major Day 2 change

For more information, see the Configure the Backup Service for Block Storage section in the VMware Integrated OpenStack Administrator Guide.

9.1.1 Restore VMware Integrated Openstack Database from Backup

In the event VMware Integrated Openstack is corrupted due to database failure, you can restore your VMware Integrated OpenStack management server and OpenStack database from backup. Restore process is documented in the *Restore VMware Integrated OpenStack from a Backup* section of the VMware Integrated OpenStack Administrator Guide.

9.1.2 Failure Recovery

In the event of a disk failure or another critical issue, a VIO node is lost and can not be restored. You can recover the individual nodes in your VMware Integrated OpenStack deployment using the CLI. Recovery will restore the nodes and apply the configuration if you have customization in custom.yml or certificate changes, with the exception of database nodes. To recover a database node, you must also reference a backup file and NFS volume where backup file is stored.
For more information, see the *Failure Recovery* section in the VMware Integrated OpenStack Administrator Guide.

9.1.3 NSX Backup and Restore

Proper backup of all NSX components is crucial to restore the system to its working state in the event of a failure. The NSX Manager backup contains all of the NSX configuration, including controllers, logical switching and routing entities, security, firewall rules, and everything else that you configure within the NSX Manager UI or API. The vCenter database and related elements like the virtual switches need to be backed up separately.

At a minimum, we recommend taking regular backups of NSX Manager and vCenter. Your backup frequency and schedule might vary based on your business needs and operational procedures. We recommend taking NSX backups frequently during times of frequent configuration changes. NSX Manager backups can be taken on demand or on an hourly, daily, or weekly basis.

We recommend taking backups in the following scenarios:

- Before an NSX or vCenter upgrade.

- After an NSX or vCenter upgrade.
- After Day Zero deployment and initial configuration of NSX components, such as after the creation of NSX Controllers, logical switches, logical routers, edge services gateways, security, and firewall policies.
- After infrastructure or topology changes. After any major Day 2 change.

To provide an entire system state at a given time to roll back to, we recommend synchronizing NSX component backups (such as NSX Manager) with your backup schedule for other interacting components, such as vCenter, cloud management systems, operational tools, and so on.

9.1.4 Back Up NSX Manager Data

For more information, see the Back Up NSX Manager Data section in the NSX Upgrade Guide.

9.1.5 NSX Manager Data Restore

For more information, see the Restore an NSX Manager Backup section in the NSX Upgrade Guide.
https://pubs.vmware.com/NSX-62/index.jsp#com.vmware.nsx.upgrade.doc/GUID-B22A6600-0E65-4765-AC4E-A9D20FC57D1D.html

9.1.6 Backup/Restore NSX Edges

All NSX Edge configurations (logical routers and edge services gateways) are backed up as part of NSX Manager data backup. If you have an intact NSX Manager configuration, you can recreate an inaccessible or failed Edge appliance VM by redeploying the NSX Edge (click the Redeploy NSX Edge icon in the vSphere Web Client). Taking individual NSX Edge backups is not supported.

9.1.7 Backup/Restore vSphere Distributed Switches

For more information, see Exporting/importing/restoring Distributed Switch configs using vSphere Web Client (2034602).

9.1.8 Backup/Restore vCenter

For more information, see the *Back Up vCenter* section in the NSX Upgrade Guide.

9.2 VIO Maintenance and VIOCLI Commands

viocli is a command line tool that enable VIO administrators to perform specific tasks on the VIO cloud It allows a cloud Admin to perform a set of maintenance action using command line instead of the VIO WebClient Plug. Tasks such as DB backup/restore, deployment update, services start/stop and storage volume operators are operations supported by viocli command. Most of viocli commands are available via WebClient. If a corresponding action is available on the VIO WebClient Plugin, we recommend to use the VIO WebClient. Below is a quick summary of available commands, commands color coded in red are those we recommend to perform from the web or only when recommended by VMware GSS team.

viocli Best Practice

viocli backup	OK
viocli restore	OK
viocli recover	OK
viocli show	OK
viocli upgrade	do not use if possible
viocli rollback	do not use if possible
viocli deployment start	do not use if possible
viocli deployment stop	do not use if possible
viocli deployment pause	do not use if possible
viocli deployment configure	use only when supported by official documentation
viocli deployment cert-req-update	do not use if possible
viocli deployment cert-update	do not use if possible
viocli deployment getlogs	OK

viocli services start	do not use if possible
viocli services stop	do not use if possible
viocli hyperic install	OK
viocli hyperic config	OK
viocli hyperic uninstall	OK
viocli hyperic stop	OK
viocli hyperic start	OK
viocli dbverify	OK
viocli deploy	do not use if possible
viocli ds-migrate-prep	OK
viocli volume-migrate	OK
viocli lbaasv2-enable	OK
viocli inventory-admin	OK

Table 9.1 viocli best practice

One of the most important VICLI command is to back, recover, as well as restore VIO database. Use the viocli backup command to create a

backup of either manager server data or the OpenStack database. We strongly recommend database backup to be performed at least once per hour. viocli backup command requires an NFS server to be available for the VMware Integrated OpenStack CL to mount.

9.2.1 OpenStack Management Server Update and Best Practices

Prior to upgrading OMS some steps need to be done to help ensure success.

a) snapshot the OMS VM. That way if anything happens, we have a recovery path.
b) verify how much free space we have on the OMS.
c) rule of thumb is 2x the size of the upgrade/update .deb file
d) run a dpkg command against the .deb file to see space required - (dpkg --info <name of package>)
e) free up space if necessary. Use "du -a <directory> | sort -n -r | head -n 10" to find top 10 files in size.

9.2.2. VIO Maria DB Restart Process

VIO uses MariaDB in a Galara Cluster. This section we describe few scenarios and how to bring back the database VMs to normal functional state in the rare event of a DB failure.

If only one database VM has been restarted, and other 2 nodes are functioning properly: Only required action is to run "service mysql start" as root on the restarted DB node. This must be done if other 2 database nodes are functional, because if other nodes were not running, starting mysql on single node in normal mode would corrupt internal state. This should work regardless of whether that single node was stopped gracefully or crashed.
When all database nodes have been stopped gracefully: Use viocli deployment start command to bring back the database service.

If all nodes are down, and at least one of them has crashed:

a) Run #sudo mysqld_safe --wsrep-recover command on each of the database nodes.
b) Register the output of the above command. One of the lines will have text: "Recovered position 5b821365-fafb-11e4-a3d3-471472ccc7b3:38719". Note the number after the colon
c) Choose the db VM where the above number is greatest. If that number is equal on all nodes, choose database01 VM (the first one).
d) Run #sudo service mysql start --wsrep-new-cluster on the selected node.
e) Run #sudo service mysql start on the rest of the nodes.
f) Once db cluster is started, you can run #vioconfig start on management server (OMS) to start the rest of the cluster components if needed.

9.2.3 VIO UI Status Update

During rare occasions when VIO runs into an error state, Cloud Admins fixes the error using CLI utilities. Depending on which CLI utilities used, VIO UI are not always getting updated. If you recover VIO using CLI and are sure that VIO is fully functional, then you can use the below commands to update the status.

Step-by-step guide

1. On management server, run # /opt/vmware/vpostgres/current/bin/psql -U omsdb
2. Execute query: # update cluster set status='RUNNING';
3. Refresh UI

Do your due diligence on all components to make sure they are up and running before manually updating the status.

9.2.4 RabbitMQ

During rare occasions when underline network infrastructure is not stable, RabbitMQ cluster may enter partitioned status. Some of the symptoms of degraded RabbitMQ cluster includes but not limited to:

- intermittent failure when creating / deleting cinder volume
- intermittent failure when creating / deleting VM
- intermittent failure updating security groups
- intermittent failure creating neutron segments

In most situations RabbitMQ cluster will auto-heal, however there are scenarios manual intervention is required. To manually

repair a partitioned cluster:

- verify the cluster cluster: rabbitmqctl cluster_status

```
root@database02:~# rabbitmqctl cluster_status Cluster status of node rabbit@database02 ...
    [{nodes,[{disc,[rabbit@database01,rabbit@database02,rabbit@database03]}]},
    {running_nodes,[rabbit@database01,rabbit@database02]},
    {cluster_name,<<"rabbit@database03">>},
    {partitions,[]}]                         <<<<<<<This should be empty when there is no partitioning
```

- Log in to each of the database nodes, and look for connection issues identified in the `/var/log/rabbitmq` log files
- On the controller nodes, check RabbitMQ message queues that are growing without being consumed which will indicate which OpenStack service is affected. Restart the affected OpenStack service.
- On the compute nodes, check RabbitMQ message queues that are growing without being consumed which will indicate which OpenStack services are affected. Restart the affected OpenStack services.
- Restart the RabbitMQ service on the database nodes may also be required to repair a partitioned cluster.

```
# service rabbitmq-server stop
# service rabbitmq-server start
```

9.3 VIO Monitoring and Logging

OpenStack is capable of generating a significant amount of syslog messages to assist you with troubleshooting. This section will review where and how OpenStack logs are stored and used.

9.3.1 Log Insight

OpenStack logs are scattered across multiple files and those files are spread across multiple VMs. The best starting point is to point all logs to a central Logging source.

The Openstack management pack for Log Insight collects machine generated data from the various different Openstack components and graphically represents this data with action oriented dashboards. The Openstack content pack helps rationalize, structure and analyze logs uniformly irrespective of whether the logs are coming from Cinder, Nova, Neutron, Keystone and/or other components. Log files are filtered via Openstack regular expressions, are stored and structured in a human readable database like repository, are analyzed based on high, medium, low log activities, and are graphically displayed with charts, graphs, and trend reports. Usage trends, API response times, error rates, log in activities, VCPU and memory consumption, images active versus inactive are all activities reported and displayed with the OpenStack Content Pack. Openstack administrators can quickly see which components are failing, where there are abnormal activities, and via alerting can be notified of high severity level, and/or outage conditions.

If there's a need to view individual logs, following refer to following tables to figure out which logs are enabled for which service:

Openstack Identity Service:

Service	Service Name	Log path	Description
Identity Service	keystone	**{controller node}**/var/log/keystone/key stone.log	Tracking users and their permissions. Providing a catalog of available services with their API endpoints.
Apache web server	apache2	**{controller node}**/var/log/apache2/erro r.log	serving Keystone API's through Apache

Table 9.2: Openstack identity service

Openstack Image Service :

Service	Service Name	Log path	Description
Image Service API server	glance-api	**{controller node}**/var/log/glance/glan ce-api.log	Accepts Image API calls for image discovery, retrieval, and storage.
Image Service Registry server	glance-registry	**{controller node}**/var/log/glance/glan ce-registry.log	Stores, processes, and retrieves metadata about images. Metadata includes items such as size and type.

Table 9.3: Openstack image service

Openstack Compute Service:

Service	Service Name	Log Path	Description
Compute API service	nova-api	**{controller node}**/var/log/nova/no va-api.log	Accepts and responds to end user compute API calls. The service supports the OpenStack Compute API, the Amazon EC2 API, and a special Admin API for privileged users to perform administrative actions. It enforces some policies and initiates most orchestration activities, such as running an instance.
Compute Conductor service	nova-conductor	**{controller node}**/var/log/nova/no va-conductor.log	Mediates interactions between the `nova-compute` service and the database. It eliminates direct accesses to the cloud database made by the `nova-compute` servi ce. The `nova-conductor` module scales horizontally
Compute VNC console authentication server	nova-consoleauth	**{controller node}**/var/log/nova/no va-consoleauth.log	Authorizes tokens for users that console proxies provide.
Compute HTML5 console driver (VMware)	nova-mksproxy	**{controller node}**/var/log/nova/no va-mksproxy.log	The VMware compute driver supports Native HTML5 consoles as an alternative to VNC.
Compute NoVNC Proxy service	nova-novncproxy	**{controller node}**/var/log/nova//no va-novncproxy.log	Provides a proxy for accessing running instances through a VNC connection. Supports browser-based novnc clients..
Compute Scheduler service	nova-scheduler	**{controller node}**/var/log/nova/no va-scheduler.log	Takes a virtual machine instance request from the queue and determines on which compute server host it runs.
Compute Service	nova-compute	**{compute node}**/var/log/nova/nov a-compute.log	A worker daemon that creates and terminates virtual machine instances through APIs. The compute driver talks to vCenter and then vCenter communicates with the hypervisor.

Table 9.4: Openstack compute service

Openstack Networking Service:

Service	Service Name	Log Path	Description

Networking Service	neutron-server	**{controller node}**/var/log/neutron/server.log	Accepts and routes API requests to the appropriate OpenStack Networking plug-in for action.

Table 9.5: Openstack networking service

OpenStack Block Storage Service:

Service	Service Name	Log Path	Description (Configuration: (Ansible managed file, do not edit directly) /etc/cinder/cinder.conf)
Block Storage API	cinder-api	**{controller node}**/var/log/cinder/cinder-api.log	Accepts API requests, and routes them to the cinder-volume for action.
Block Storage Scheduler	cinder-scheduler	**{controller node}**/var/log/cinder/cinder-scheduler.log	Selects the optimal storage provider node on which to create the volume. A similar component to the nova-scheduler.
Block Storage Volume	cinder-volume	**{controller node}**/var/log/cinder/cinder-volume.log	Interacts directly with the Block Storage service, and processes such as the cinder-scheduler. It also interacts with these processes through a message queue. The cinder-volume service responds to read and write requests sent to the Block Storage service to maintain state. It can interact with a variety of storage providers through a driver architecture.
		{controller node}/var/log/cinder/cinder-volume-<AZ>:<VC FQDN or IP>.log	Two cinder volume service processes per VC e.g. cinder-volume-nova:vxlan-vm-111-31.nimbus-tb.eng.vmware.com.log
		{controller node}/etc/cinder/cinder-<AZ>:<VC FQDN or IP>.conf	Each cinder volume service per VC has its own configuration file e.g. /etc/cinder/cinder-nova:vxlan-vm-111-31.nimbus-tb.eng.vmware.com.conf

Table 9.6: Openstack block storage service

Openstack Dashboard:

Service	Service Name	Log Path	Description
Apache web server	apache2	**{controller node}**/var/log/apache2/error.log	The dashboard is served to users through the Apache web server. Logs all unsuccessful attempts to access the web server, along with the reason that each attempt failed.
		{controller node}/var/log/apache2/access.log	Logs all attempts to access the web server.

Table 9.7: Openstack dashboard

Memory cache servers: Memcached

Service	Service Name	Log Path	Description Configuration: (Ansible managed file, do not edit directly) /etc/memcached.conf
Memory Cache	memcached	**{controller node}**/var/log/memcached.log **{controller node}**/var/log/syslog	Memory cache demon that can be used by most OpenStack services to store ephemeral data, such as tokens.

Table 9.8: Openstack memory cache servers:

•

Orchestration (heat):

Service	Services Name	Log Path	Description

Heat API Service	heat-api	**{controller node}**/var/log/heat/heat-api.log	An OpenStack-native REST API that processes API requests by sending them to the heat-engine over Remote Procedure Call (RPC).
	heat-api-cfn	**{controller node}**/var/log/heat/heat- api-cfn.log	An AWS Query API that is compatible with AWS CloudFormation. It processes API requests by sending them to the heat-engine over RPC.
	heat-api-cloudwatch	**{controller node}**/var/log/heat/heat- api-cloudwatch.log	Heat/Using-CloudWatch Wiki
Heat Engine Service	heat-engine	**{controller node}**/var/log/heat/heat-engine.log	Orchestrates the launching of templates and provides events back to the API consumer.

Table 9.9: Openstack orchestration

Ceilometer

Service	Service Name	Log Path	Description
ceilometer-agent-central	ceilometer-polling	**{ceilometer node}**/var/log/ceilometer/ceilometer-agent-central.l og	Telemetry service central agent
ceilometer-agent-notification	ceilometer-agent-notification	**{ceilometer node}**/var/log/ceilometer/ceilometer-agent-notificat ion.log	Telemetry service notification agent
ceilometer-api	ceilometer-api	**{ceilometer node}**/var/log/ceilometer/ceilometer-api.log	Telemetry service API
ceilometer-collector	celiometer-collector	**{ceilometer node}**/var/log/ceilometer/ceilometer-collector.log	Telemetry service collection
ceilometer-dbsync	MongoDB integration	**{ceilometer node}**/var/log/ceilometer	Informational messages

Table 9.10: Ceilometer

Management Server:

Service	Service Name	Log Path	Description
OpenStack Management Service (VMware)	oms	**{OMS Server}**/var/log/oms/om s.log	Provisioning issues. Typically logs an exception to describe the failure. The specified Openstack configuration is dumped (look for ClusterDeployment in the log. **createClusterPlanStep**: given the parameters from the User, the OMS creates the resource plan for the VIO Management Cluster VMs **createVMStep**: the OMS goes to vCenter. VIO Management Cluster VMs are created and connected to the networks **softwareCreateClusterStep**: OMS calls Jarvis and Ansible starts configuring the VMs based on teh role assigned (Controllers, etc) **autoConfigureOsvmwStep**: the UI is updted with the information on the deployment
OpenStack WebClient Plugin Service (VMware)	osvmw	**{OMS Server}**/var/log/oms/reg ister-plugin.log	Issues with plugin registration.

		{OMS Server}/var/log/jarvis/ansible.log	A free-software platform for configuring and managing computers which combines multi-node software deployment, ad hoc task execution, and configuration management.
Jarvis	jarvis	**{OMS Server}**/var/log/jarvis/jarvis.log	A REST API on top of Ansible for deploying OpenStack and doing subsequent day 2 tasks . The OMS makes REST calls to it and it responds by running Ansible plays. It is the engine responsible for running tasks like stopping/starting a deployment, adding ceilometer to a deployment, patching, etc. (VMware)
		{OMS Server}/var/log/jarvis/pecan.log	A lightweight python web framework that allows REST API ability to Jarvis
viocli	none	**{OMS Server}**/var/log/viocli/viocli.log	viocli utility logs its output on a file, including the Ansible operations output, the full command and parameters executed and debugging information. The maximum file number for the viocli logs is 7: the logs are rotated each time the file reaches 100MB size.
Ansible Inventory File			In the inventory file we can see all the variables that Ansible has used to configure the VIO Management Cluster and will use every time it will be asked to reconfigure the environment > **sudo viocli show -p**

Table 9.11: Management server

Database Servers (MariaDB):

Service	Service Name	Log Path	Description **Configuration: (Ansible managed file, do not edit directly)** /etc/mysql/my.cnf
MySQL	mysql	{**database node**}/var/log/syslog	
		{**database node**}/var/log/mysql.err	NOT USED; See syslog
		{**database node**}/var/log/mysql.log	NOT USED, See syslog
		{**database node**}/var/log/mysql/maria db-bin.{log number}	**(**The binary log contains a record of all changes to the databases, both data and structure.It consists of a set of binary log files and an index**)**
		{**database node**}/var/log/mysql/maria db-bin.index	**(**The binary log contains a record of all changes to the databases, both data and structure. It consists of a set of binary log files and an index**)**

Table 9.12: Database servers

RabbitMQ servers: **RabbitMQ File Locations** | **RabbitMQ Admin Guide** | **RabbitMQ Troubleshooting**

Service	Service Name	Log Path	Description
RabbitMQ	rabbitmq-server	**{database node}**/var/log/rabbitmq/rabbit@{RABBITMQ_NODENAME}.log	(RabbitMQ server's Erlang log file) **NOTE:** Erlang nodes use a cookie to determine whether they are allowed to communicate with each other - for two nodes to be able to communicate they must have the same cookie.
		{database node}/var/log/rabbitmq/rabbit@{RABBITMQ_NODENAME}-sasl.log	(RabbitMQ server's Erlang SASL (System Application Support Libraries) log file)
		{database node}/var/log/rabbitmq/shutdown_lo g	
		{database node}/var/log/rabbitmq/shutdown_er r	
		{database node}/var/log/rabbitmq/startup_log	
		{database node}/var/log/rabbitmq/startup_err	

Table 9.13: RabbitMQ servers

Load Balancers (HAProxy): **HAProxy**

Service	Service Name	Log Path	Description

HAProxy	haproxy	{**loadbalancer node**}/var/log /haproxy/haproxy.log	HAProxy provides a fast and reliable HTTP reverse proxy and load balancer for TCP or HTTP applications. It is particularly suited for web crawling under very high loads while needing persistence or Layer 7processing. Configuration: (Ansible managed file, do not edit directly) /etc/haproxy/haproxy.cfg
Keepalive	keepalived	{**loadbalancer node**}/var/log /syslog {**loadbalancer node**}/var/log /haproxy/haproxy.log	Keepalived implements a set of checkers to dynamically and adaptively maintain and manage loadbalanced server pool according their health. Configuration: (Ansible managed file, do not edit directly) /etc/keepalived/keepalived.conf

Table 9.14: Load Balancers

Miscellaneous (version 2.5 and above)

Service	Service Name	Log Path	Description	Notes
viomon	none	**{OMS Server}**/var/log/vio mon/viomon.log	built-in monitoring system, viomon, that regularly polls the deployment to get information regarding its health and status.	viocli utility has been updated with a tool that interrogates this data and summarizes it for the user. (viocli deployment status --help)
viocli	none	**{OMS Server}**/var/log/vioc li/viocli.log	viocli utility logs its output on a file, including the Ansible operations output, the full command and parameters executed and debugging information.	The maximum file number for the viocli logs is 7: the logs are rotated each time the file reaches 100MB size.

Table 9.15: Miscellaneous (version 2.5 and above)

Import VM Troubleshooting (VIO 3.0 and above)

		Log Path	Description	Notes
vapi	vapi	**{OMS Server}**/var/log/vmware/vapi/vapi.lo g	API logs	Import VM tool is implemented using vAPI Provider Development Kit (PDK) • Standardize the experience with VMware APIs, CLIs across products • REST, DCLI, vROPs, PowerCLI cmdlets are deliverables from vAPI framework
dcli		**{OMS Server}**/var/log/vapi/dcli.log	dcli logs	
vioshim	vioshim			**Configuration:** /opt/vmware/vapi/vioshim/conf vioshim service is restricted only to OMS only in VIO 3.0

Table 9.16: Import VM Troubleshooting

Upgrade/Update

Log Path	Description
{OMS Server}/var/log/dpkg.log	shows details of the packages being installed
{OMS Server}/var/log/syslog.log	shows details of the ansible scripts being run
{OMS Server}/var/log/viopatch/	shows the output of the viopatch command

Table 9.17: Upgrade/Update

9.3.2 Log Capture and Aggregation

When working with VMware support, it's often necessary to capture all logs and upload to VMware for review. OpenStack Management Server has build commands to simplify the log capture process. To capture and upload logs to VMware support:

1. SSH to the OpenStack Management Server (OMS)
2. Check that there is enough space on the OMS
3. Free space if needed or mount a NFS datastore to the OMS server for additional space.
4. Start gathering logs by the following command.

```
viocli deployment
```

5. Once the bundle is collected the bundle can be ether directly FTP to VMware. Following KB articles has additional details:
   a) Uploading diagnostic information for VMware using FTP (2070100)
   b) Uploading diagnostic information for VMware through the Secure FTP portal (2069559)

9.3.3 Log collection for connected products

vCenter

"Collecting diagnostic information for VMware vCenter Server 4.x, 5.x and 6.0 (1011641)" https://ikb.vmware.com/kb/1011641

ESXi

"Collecting diagnostic information using the vm-support command in VMware ESX/ESXi (1010705)" https://ikb.vmware.com/kb/1010705

NSX

"Collecting diagnostic information for VMware NSX for vSphere 6.x (2074678)" https://ikb.vmware.com/kb/2074678

Other

"Collecting diagnostic information for VMware products (1008524)" https://ikb.vmware.com/kb/1008524

Section 10 Conclusion

VMware Integrated OpenStack is a VMware-supported OpenStack distribution that makes it easy for IT to run an enterprise-grade OpenStack cloud on top of VMware virtualization technologies. It allows organizations to boost developer productivity by providing developers with simple, standard and vendor-neutral OpenStack API access to VMware infrastructure.

Below are four use cases that VMware Integrated OpenStack enables customers to accelerate their journey on digital transformation:

- **Developer Cloud**- Increase developer productivity by providing self-service and programmable provisioning of infrastructure resources through standard OpenStack APIs. With VMware Integrated OpenStack developers get the best of both words - public cloud-like user experience with the APIs they want on the most proven infrastructure.

- **Production Ready Container Management** - Run containers on top of a production ready OpenStack platform. VMware Integrated OpenStack provides a complete solution for organizations who want to run containerized applications in production, making it easier for IT with advanced security, high availability, multi-tenancy, troubleshooting and so on.

- **Advanced Network Virtualization with VMware NSX -** Deploy VMware Integrated OpenStack with VMware NSX for advanced security and network virtualization capabilities such as firewalling and micro-segmentation for your business-critical workloads.

- **NFV cloud for CSP** - Build Network Virtualization Functions (NFV). VMware Integrated OpenStack is the ideal platform for Communication Service Providers (CSPs) looking to build an NFV cloud with specific capabilities and key functionality required for NFV customers.

VIO offers the following key features:

- **Integrated Container Orchestration and Management Support -** Run containerized applications in production on top of OpenStack with multi-tenancy and persistent volumes.

- **Streamlined OpenStack Deployment -** Deploy a standard OpenStack cloud in 20 minutes on top of VMware SDDC. VMware Integrated OpenStack is easily deployed with an OVA file using the VMware vSphere Web Client. Perform patching and upgrades with minimal disruption.

- **Best-of-Breed Infrastructure -** Leverage VMware SDCC to take advantage of advanced enterprise and production-ready features and capabilities delivered by VMware vSphere, VMware NSX and VMware vSAN. Experience benefits such as improved security, high availability, simplified maintenance and disaster recovery.

- **Integrated Operations and Management -** Out-of-the-box vRealize Operations, vRealize Log Insight and vRealize Automation integrations for streamlined operations such as health checks, troubleshooting and capacity management, as well as governance and control with user management, role-based access control (RBAC), quotas and more.

- **Advanced NFV Capabilities -** Advanced features and capabilities targeted at communications service providers looking to build an NFV cloud based on Integrated OpenStack Carrier Edition.

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com