JUNIPER NETWORKS | Engineering Simplicity

Reference Architecture for Contrail Cloud 13.1 and 13.2

Published
2023-07-14

RELEASE
Release 13.1 and 13.2

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Reference Architecture for Contrail Cloud 13.1 and 13.2
Release 13.1 and 13.2

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at https://support.juniper.net/support/eula/. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

CHAPTER 1

Reference Architecture for Contrail Cloud 13.1 and 13.2

Introduction

IN THIS SECTION

This section provides an introduction to Contrail Cloud and the Contrail Cloud reference architecture.

About This Reference Architecture

This reference architecture provides networking professionals with the concepts and tools needed to optimally architect a Contrail Cloud environment. The intended audience for this guide includes system integrators, infrastructure professionals, partners, and customers that are currently deploying or are considering deploying Contrail Cloud.

The architecture presented in this guide is intended for implementation with Contrail Cloud 13.1 and 13.2 environments.

Contrail Cloud Overview

Juniper Networks Contrail Cloud provides cloud service providers with a bundled solution to build cloud platform infrastructures. The Contrail Cloud bundle simplifies your cloud building journey by integrating multiple software components—Red Hat Openstack orchestration, Contrail Networking, Red Hat Ceph Storage, and Appformix—into a bundle with pre-configured files that is collectively installed on multiple devices using a single installation procedure.

The bundled Contrail Cloud package includes:

- Red Hat OpenStack
- Red Hat Ceph Storage

- Juniper Networks Contrail Networking
- Juniper Networks Contrail Command
- Juniper Networks AppFormix
- Pre-configured YAML files to simplify the initial configuration procedures, as well as script files to implement configuration changes.

All Contrail Cloud software components are downloaded from a Juniper Satellite site. You must send an email request to mailto:contrail_cloud_subscriptions@juniper.net to obtain the credentials to access the Juniper Satellite site. See Deploying Contrail Cloud.

The remainder of this reference architecture assumes that the reader has a strong working knowledge of Contrail Cloud. For additional information on Contrail Cloud fundamentals, see the Contrail Cloud Product Information page and the Contrail Cloud TechLibrary page.

Reference Architecture Overview

This document provides a reference architecture for a Contrail Cloud 13.1 or 13.2 deployment. The reference architecture has been tested by Juniper Networks and is optimized for performance, scale, and resilience. It provides sample server and network device configurations as well as design guidance to provide in-depth insight on the configuration and design decisions made within Contrail Cloud architectures.

The main design characteristics of this reference architecture:

- Three controller nodes that are installed in different racks for high availability.
- A subnetting scheme that provides separate subnets per rack for tenant data, internal API, and storage traffic.
- An IP Fabric that includes leaf devices dual-homed to spine devices while connecting to separate SDN gateway routers. The IP Fabric runs EVPN-VXLAN.
- Separate NICs for tenant data and storage traffic.

This reference architecture includes sections that describe supported variations to the architecture. The supported variations include environments with a reduced number of physical interfaces and simplified networking setups. See "Enhanced Resilience and Scale Variations" on page 87 and "Reference Architecture Variations" on page 91.

The following deployment scenarios are covered in this reference architecture:

- large scale, high performance cloud deployments.

- Deployments with compute nodes in any combination of kernel, DPDK, and SR-IOV modes.
- Dual-homed compute nodes with separate and combined management, control/data, and storage networks.

In this document, configuration file fragments are provided that show how to configure the components of a Contrail Cloud environment to meet the specifications of the deployment architecture.

This document is intended to be used in conjunction with the Contrail Cloud Deployment Guide, which provides Contrail Cloud installation procedures.

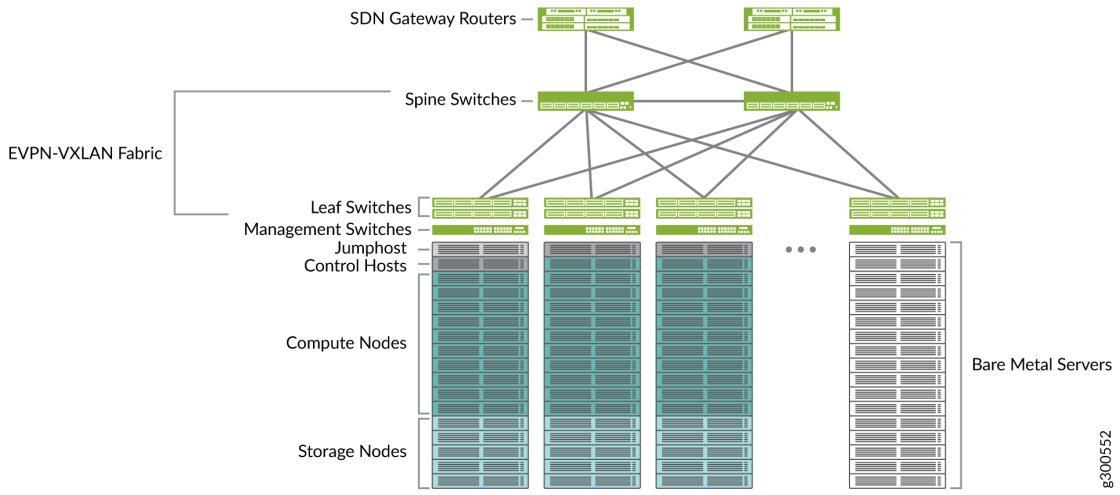
Reference Architecture Scope

Contrail Cloud is a software package that arrives with detailed descriptions of how networking should be established on servers running in the topology and the options available for configuring the software running on the servers.

This reference architecture document provides details and recommendations for configuring Contrail Cloud beyond the initial server setup. This reference architecture also provides walkthroughs and advice related to optimizing the networking between servers and the fabric in a Contrail Cloud environment.

Figure 1 on page 4 visually provides the high-level overview of a typical cloud environment networking architecture. The figure includes the devices in the EVPN-VXLAN IP Fabric as well as the SDN gateway routers, whose configuration is beyond the scope of this reference architecture.

Figure 1: Reference Architecture High-level Overview



The following information is provided in this document:

- Server hardware recommendations for each type of node to achieve optimal scale and performance.
- NIC configuration and network connectivity for each node type.
- Recommended BIOS settings.
- Network addressing scheme for servers.
- Detailed specification of capabilities needed in the networking infrastructure.
- Recommendations for Juniper device models for each network role.
- Configuration examples from each network device type showing how the various networking requirements can be met by configuring an IP Fabric using Juniper devices.

NOTE: This reference architecture provides some information related to connecting nodes to the end fabric, but does not provide in-depth coverage of fabric provisioning and configuration. See Data Center Fabric Architecture Guide.

RELATED DOCUMENTATION

Contrail Cloud Deployment Guide

Contrail Cloud TechLibrary page

Contrail Cloud Software

IN THIS SECTION

This section provides detailed information about Contrail Cloud software components.

Contrail Cloud Software Overview

Juniper Networks Contrail Cloud is an integrated cloud platform that uses the Red Hat OpenStack Platform for orchestration, Juniper Contrail Networking for network automation, AppFormix for monitoring and analytics, and Red Hat Ceph Storage for storage. All Contrail Cloud software packages are deployed using a single installer. Contrail Cloud is commonly deployed in Service Provider and Enterprise environments.

Software Versions

This reference architecture supports Contrail Cloud 13.1 and 13.2.

The following software runs on the servers in this reference architecture.

Table 1: Contrail Cloud Release 13.2 Software Components

Software Components	Contrail Cloud Release 13.2 Version
Contrail Networking	Contrail Networking Release 1910
AppFormix	AppFormix Release 3.1.6
OpenStack	Red Hat OpenStack 13 (z9)–OpenStack Queens Version (Red Hat CDN sync 1-October-2019)
Operating System and Kernel Versions	Red Hat Enterprise Linux 7.7 (RHEL 7.7), Kernel Version 3.10.0-1062.1.2.el7.x86_64 (Red Hat CDN sync 1-October-2019)
CEPH Storage	3.2 (Red Hat CDN sync 1-October-2019)

Table 2: Contrail Cloud Release 13.1 Software Components

Software Components	Contrail Cloud Release 13.1 Version
Contrail Networking	Contrail Networking Release 1908

AppFormix	AppFormix Release 3.1
OpenStack	Red Hat OpenStack 13 - OpenStack Queens
Operating System and Kernel Versions	Red Hat Enterprise Linux 7.6 (RHEL 7.6), Kernel 3.10.0-957.5.1 (Red Hat OpenStack 13)
CEPH Storage	3.2

The software versions are automatically downloaded by the Contrail Cloud deployment scripts during installation. The software versions are fixed Contrail and Red Hat Openstack packages and containers downloaded from a Juniper Satellite.

You must send an email request to mailto:contrail_cloud_subscriptions@juniper.net to obtain the credentials to access the Juniper Satellite. See Deploying Contrail Cloud.

See Release Notes: Contrail Cloud for additional information regarding Contrail Cloud component versioning.

Node Types

Nodes in Contrail Cloud are software packages that run in servers and in virtual machines. Figure 2 on page 8 illustrates the nodes that are deployed in Contrail Cloud 13.1.

Figure 2: Contrail Cloud Software Components

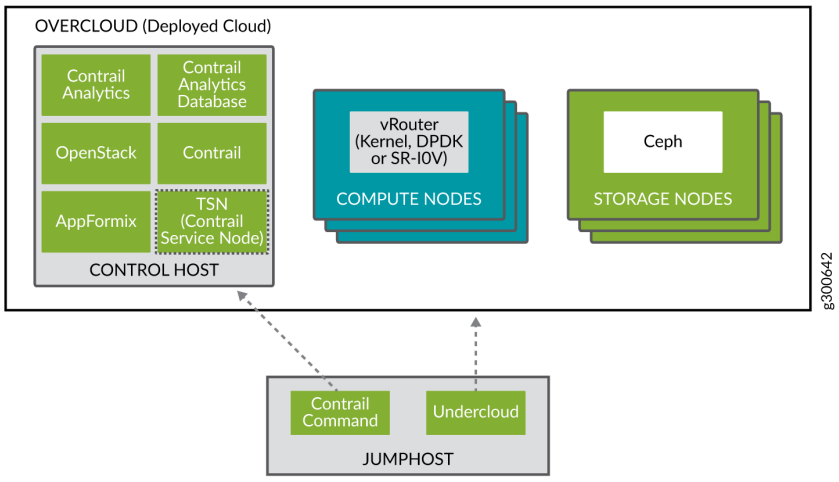


Table 3 on page 8 provides a summary description of each node.

Table 3: Node Summary

Node	Description
Compute node	Server provisioned to support virtualization; hosts virtual machines. Networking of the VMs on a compute node is provided by the Contrail Networking vRouter.
Control Host	Server that hosts controller nodes. Controller nodes are VMs responsible for controlling Contrail Cloud functions. For additional information on control hosts and controller nodes, see "Contrail Cloud Configuration" on page 30.

Jumphost	A server that performs the following functions in Contrail Cloud: • hosts the undercloud. The undercloud runs as a VM on the jumphost and is responsible for provisioning and managing all controls hosts, compute nodes, and storage nodes. • hosts Contrail Cloud automation. All Contrail Cloud scripts and YAML configuration files are stored and instantiated on the jumphost. • hosts the Contrail Command web user interface virtual machine. • serves as the entry point for access to the undercloud or overcloud. You can "jump" between the overcloud and the undercloud on the jumphost.
Storage node	Server whose purpose is storing data. Storage nodes run Red Hat Ceph storage software in Contrail Cloud.

NOTE: Storage nodes must be separate from compute nodes in a Contrail Cloud environment. Hyperconverged nodes are not supported.

NOTE: TSN stands for ToR Service Node. A TSN is an optional node used when Contrail Networking is managing bare metal servers. In the current version of Contrail Networking, the TSN name has been changed to Contrail Service Node (CSN). The Red Hat OOO Heat templates, however, still contain the term TSN. Both terms are used in this guide.

Table 4 on page 9 illustrates which nodes are deployed as virtual machines and which nodes must run in the operating system of a server.

Table 4: Virtual Machine Summary

Function	Deployed as VM
Jumphost	No

	Undercloud	Yes
	Contrail Command	Yes
Controller Hosts	OpenStack	Yes
	Contrail Controller	Yes
	Contrail Analytics	Yes
	Contrail Analytics DB	Yes
	Appformix	Yes
	Contrail Service Node (TOR Service Node)	Yes
Compute Nodes	vRouter in kernel mode	No
	vRouter using DPDK	No
	vRouter using SR-IOV	No
Storage Nodes (Ceph Software)		No

All hosts in a Contrail Cloud deployment must run on servers dedicated to the Contrail Cloud deployment since Contrail Cloud provisions the servers with a new operating system. The jumphost can only run the undercloud and Contrail Command VMs.

The hardware management for nodes deployed in bare-metal servers is typically done using IPMI while VMs are typically deployed using VirtualBMC. Note that a Red Hat support exception has been made to use VirtualBMC in Contrail Cloud production environments.

In the most common deployment architectures, an instance of each controller function runs in each of three controller node servers. This style of deployment is described in detail in this reference architecture. See "Contrail Cloud Configuration" on page 30.

Additional descriptions on how to configure variable deployments—such as splitting OpenStack, Contrail Controller, and Contrail Analytics onto separate sets of servers for improved performance and reliability—are also provided in this reference architecture. See "Reference Architecture Variations" on page 91.

Server Hardware

The servers used for the jumphost, controller nodes, compute nodes, and storage nodes have different physical server requirements.

Physical server requirements can vary between environments. The following table provides the recommended minimum physical server specifications for each node type.

Table 5: Physical Server Minimum Specifications

Role	Cores	Memory	Disk	Network
Jumphost	20 (2x10)	128 GB	2 x 1 TB HDD (SW RAID) — Host OS **NOTE**: HW Raid recommended if RAID controller is available in your environment. 3 x 1 TB HDD — Data	1G — IPMI 1G/10G — Provisioning 1G/10G — Intranet
Control Host	36 (2x18)	256 GB	1 x 1 TB SSD (Cassandra Journaling) **NOTE**: 1x1TB is the recommended disk memory option for control hosts. 2 x 1 TB HDD (HW RAID) —Host OS 3 x 2 TB HDD (HW RAID) —Data	1 x 1G — IPMI 1x1G/10G — Provisioning 1x1G/10G — Management (optional) 4 x 10G/25G/40G ports on two Intel Fortville (XL710/X710/XXV710) family NICs. Both high-speed NICs must be on NUMA 0.

Table 5: Physical Server Minimum Specifications *(Continued)*

Role	Cores	Memory	Disk	Network
Compute Node	28 (2x14)	256 GB	2 x 1 TB HDD (HW RAID) — Host OS 4 x 1 TB HDD — Data (only needed if Ceph storage not used to provide storage)	1 x 1G — IPMI 1x1G/10G — Provisioning 1x1G/10G — Management (optional) 4 x 10G/25G/40G ports on two Intel Fortville (XL710/X710/XXV710) family NICs. Both high-speed NICs must be on NUMA 0.
Storage Node	16 (2x8)	128 GB	2 x 1 TB HDD (HW RAID) — Host OS 2 x 480 GB SSDs — Ceph Journaling 10 x 1 TB HDDs — Ceph OSDs	1 x 1G — IPMI 1x1G/10G — Provisioning 1x1G/10G — Management (optional) 4 x 10G/25G/40G ports on two Intel Fortville (XL710/X710/XXV710) family NICs. Both high-speed NICs must be on NUMA 0.

NOTE: Virtual machines and processes located on NUMA 1 communicate with NICs via the QPI bus, which is less costly than balancing traffic across NICs in both NUMAs.

NOTE: For improved controller performance, you can use 6 SSDs. Configure 5 drives as RAID 5 and maintain a spare drive when using 6 SSDs.

NOTE: The number of drives and capacity in storage nodes can be increased as needed. The Red Hat recommendation of a 1:5 ratio of SSD:HDD drives should be followed. For additional information, see the *Throughput-optimized Solutions* section in the Red Hat Storage Hardware Selection Guide).

The sizing and configuration recommendations provided in this reference architecture are intended to enable a Contrail Cloud environment to operate at the highest possible performance. The configuration

file examples provided in this reference architecture assume these server configuration recommendations are followed. Other server configurations are supported; see "Reference Architecture Variations" on page 91.

Networking for Contrail Cloud 13.1 and 13.2

Traffic for different purposes is segregated onto different physical logical interfaces in Contrail Cloud 13.1 and 13.2. The configuration of server-facing ports in the IP Fabric must match the configuration of the corresponding server ports as specified in the Contrail Cloud configuration files.

Table 6 on page 13 lists the server-layer networks used in Contrail Cloud 13.1 and 13.2.

Table 6: Networking in Contrail Cloud Summary

Network	Purpose
IPMI	Provides hardware management of servers. IPMI services are mostly used by the Openstack Ironic service and are established outside of the Contrail Cloud installation. IPMI services can be used by the server nodes in Contrail Cloud.
Intranet	Provides user access to the jumphost. Provides access to satellite repositories for the jumphost. Provides access to the Contrail Command web user interface. Provides external network access via SNAT from the control plane network.
Provisioning or Control Plane	Deploys new nodes using PXE booting and to install software packages on overcloud bare metal servers. The provisioning/control plane network is used by Red Hat Director and is predefined before the creation of the undercloud.
Internal API	Provides communication with the OpenStack and Contrail Networking services, including Keystone, Nova, and Neutron using API communication, RPC messages, and database communication.

External	Provides tenant administrators access to the OpenStack Dashboard (Horizon) graphic interface, the public APIs for Openstack services, public Contrail APIs, and the Appformix WebUI. Commonly used as a path to the intranet.
Tenant	Supports overlay data-plane traffic—VXLAN and MPLS over UDP—and Contrail Controller to Contrail vRouter control plane traffic.
Storage	Supports storage data traffic for Ceph, block storage, NFS, iSCSI, and any other storage types.
Storage Management	Provides Ceph control, management, and replication traffic.
Management	(Optional) Provides direct access to compute nodes without having to send traffic through the jumphost. Can be used for DNS and NTP services.

NOTE: Depending on the context, Red Hat documentation uses the names Provisioning and Control Plane for the same network. In this document, the Provisioning network name is used to refer to the network in the undercloud and the Control Plane network name is used to refer to the network in the overcloud. The Control Plane network name, notably, is used in configuration files.

All networks, except IPMI and Intranet, are configured on nodes by Contrail Cloud 13.1 and 13.2 using information contained in YAML configuration files. Table 7 on page 14 lists the networking requirements by role in Contrail Cloud 13.1 and 13.2.

Table 7: Device & VM Network Summary

Role	Intranet	Provisioning	Internal API	External	Tenant	Storage	Storage Mgmt	Mgmt
Jumphost (Undercloud VM)	✓	✓						

Openstack Controller		✓	✓	✓	Optional	✓	✓	Optional
Contrail Controller		✓	✓	Optional	✓			Optional
Contrail Analytics		✓	✓	Optional	✓			Optional
Contrail Analytics DB		✓	✓	Optional	✓			Optional
Contrail Service Node (ToR Service Node)		✓	✓	Optional	✓			Optional
Appformix Controller		✓	✓	✓	Optional		✓	Optional
Control Host (Running all Controller VMs)		✓						Optional
Compute Node		✓	✓		✓	✓		Optional
Storage Node		✓				✓	✓	Optional

Table 8 on page 16 provides the networking requirements for each host type in a Contrail Cloud. The table assumes that each listed host is running all applicable controller functions.

Table 8: Host Network Summary

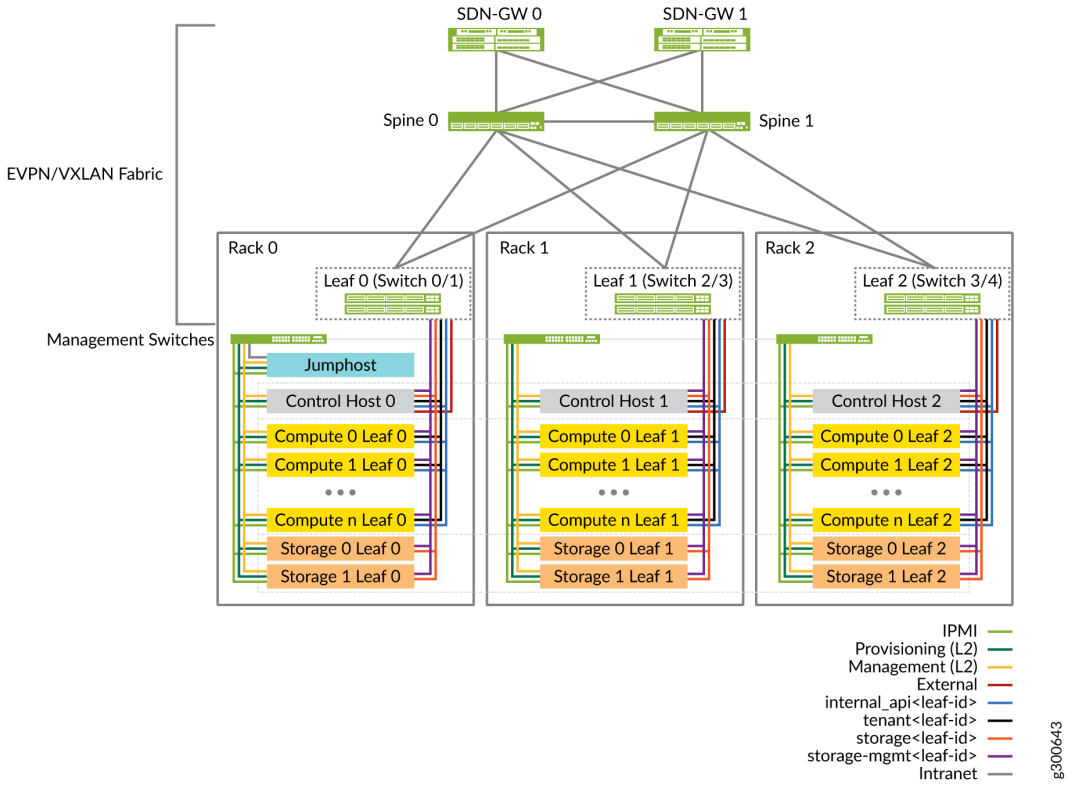
Host	Intranet	Provisioning	Internal API	External	Tenant	Storage	Storage Mgmt	Mgmt
Jumphost	✓	✓						
Control Host		✓	✓	✓	✓	✓	✓	Optional
Compute Node		✓	✓		✓	✓		Optional
Storage Node		✓				✓	✓	Optional

The server port assignments depend on the available NICs in each server.

The use of a management network to enable direct access to nodes without going through the jumphost is optional. It is included in this reference architecture for convenience.

The diagram, below, shows the layout for a Contrail Cloud environment in three racks, with controller nodes that are running all controller functions.

Figure 3: High-level Architecture Overview—Rack View



NOTE: Red Hat configuration files use the term leaf to refer to a group of servers—generally in the same rack—which share switches for network connectivity. In this reference architecture, connectivity for servers in the same Red Hat leaf is implemented by a pair of switches with leaf roles in the IP Fabric and by a management switch.

NOTE: Some Contrail Cloud documentation shows the controller nodes all placed in the same rack. The nodes are placed in separate racks for resilience in this reference architecture. The architecture shown in the other Contrail documentation is a supported variance for this reference architecture. See "Reference Architecture Variations" on page 91.

The IPMI, Management, and Provisioning networks are connected via management switches and the networks are stretched between switches using trunk connections for VLANs, or by using a separate VXLAN for each network.

The *external* network contains the externally-routable API and UI IP addresses for the various controller functions, and generally these addresses reside in the same subnet. The VLAN for the *external* network

is configured to be in a VXLAN, which terminates in a VRF which is configured to route between the network where tenant user packets arrive from the *external* network. The network naming convention for these rack-specific subnets is *network_name*(*leaf #*), where the leaf number is appended to the network name. The name of the network without an appended leaf number is used to identify the subnets that are used by control hosts.

The rest of the networks are connected into the leaf switches within the IP Fabric, where EVPN-VXLAN is used to connect switches between racks. The diagram shows connectivity for these networks within supernets that span racks, but these supernets are subdivided on a per rack basis using the leaf-id as part of each subnet identifier. This setup is described in more detail later in this document.

The other networks connected to the IP Fabric have a separate subnet for each rack. Routing is used to connect the subnets together using VXLAN. The routing can be configured in leaf devices (edge routing) or spine devices (central routing). Edge routing is used in this reference architecture.

Table 9 on page 18 summarizes recommended network configurations options.

Table 9: Recommended Network Subnets and Implementations

Network	Subnet Description	Implementation
IPMI	IPMI is typically an external service that can be used by devices that can route to it. IPMI is not established by Contrail Cloud.	IPMI is typically not spanned over racks. It can be spanned over racks using VXLAN.
Intranet	Administrative access to jumphost.	Specific port configuration
Provisioning	Layer 2 single subnet	Span over racks using VLAN or VXLAN
Internal API	Layer 2 single subnet	Span over racks using VLAN or VXLAN
Internal API *leaf-number*	One subnet per rack *leaf-number* is the Red Hat Openstack leaf number.	Layer 3 routing between racks
External	Layer 2 single subnet	Span over racks using VLAN or VXLAN
Tenant	Layer 2 single subnet	Span over racks using VLAN or VXLAN

Tenant *leaf-number*	One subnet per rack *leaf-number* is the Red Hat Openstack leaf number.	Layer 3 routing between racks
Storage	Layer 2 single subnet	Span over racks using VLAN or VXLAN
Storage *leaf-number*	One subnet per rack *leaf-number* is the Red Hat Openstack leaf number.	Layer 3 routing between racks
Storage Management	Layer 2 single subnet	Span over racks using VLAN or VXLAN
Storage Management *leaf-number*	One subnet per rack *leaf-number* is the Red Hat Openstack leaf number.	Layer 3 routing between racks
Management	Layer 2 single subnet	Span over racks using VLAN or VXLAN

When a network is composed of multiple subnets—one subnet per rack—the subnet that contains all the rack subnets is termed the supernet. The subnet address of the supernet is used to configure static routes on servers to ensure proper traffic flow. It is important that each supernet address range is large enough to accommodate future growth in the number of racks.

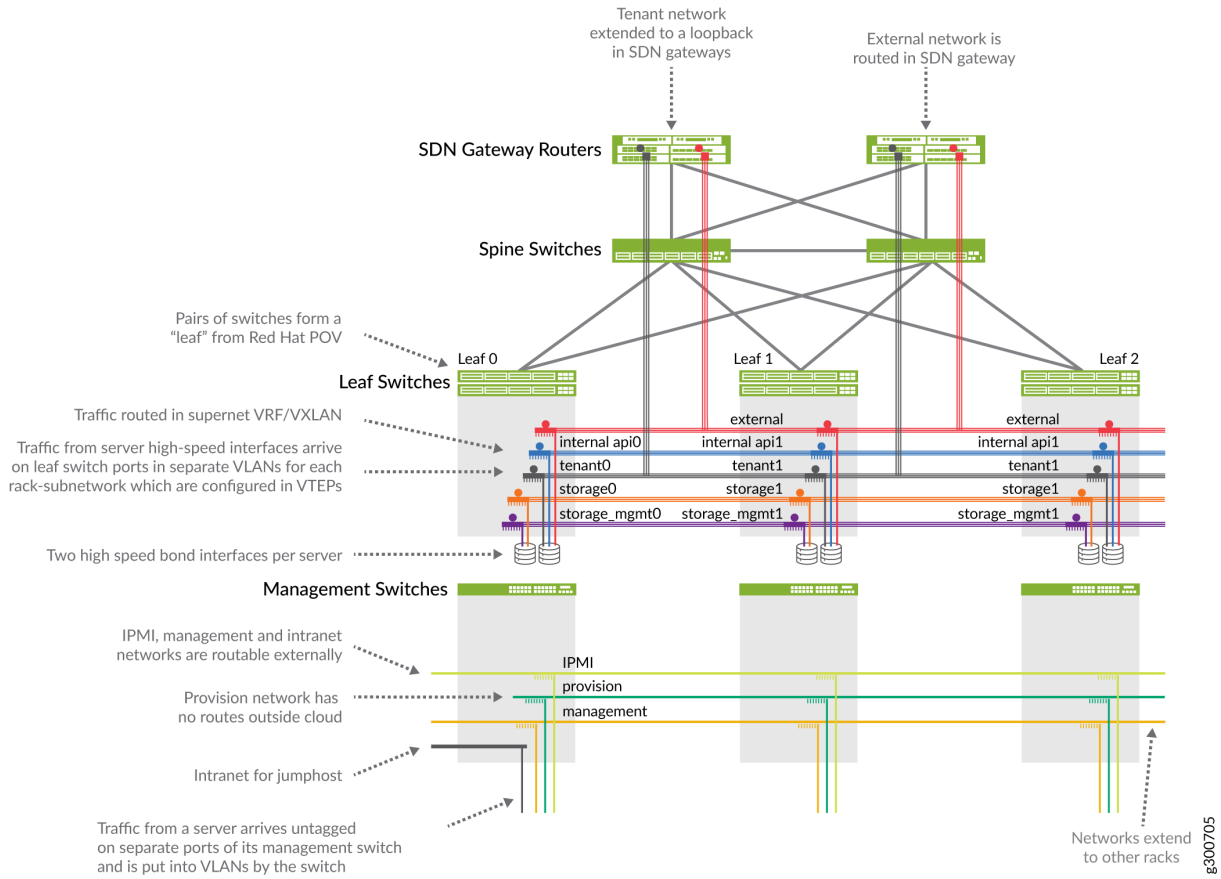
For networks that have a subnet per rack, the base subnet that is used for control hosts is stretched across the racks containing such nodes. A naming convention is used in configuration files where the name of the subnet in each rack has a suffix of the number of the rack (leaf in Red Hat terminology) and the base subnet has no suffix. For instance, the tenant network is stretched across the racks containing control hosts, and tenant0, tenant1, and is used for compute and storage nodes in each rack. This is explained in detail in Internal API, Tenant, Storage, and Storage Management Network sections.

The network configuration described in the table is designed to provide maximum robustness and performance. Variations are supported where networks span racks instead.

The upcoming diagrams illustrate how VLANs, VTEPs, IRBs, and VRFs are configured in gateway, spine, and leaf devices. Illustrations of these components in edge-routed bridging (ERB) environments—where routing is done on the leaf devices—and centrally-routed bridging (CRB)—where routing is done on the spine devices—are provided.

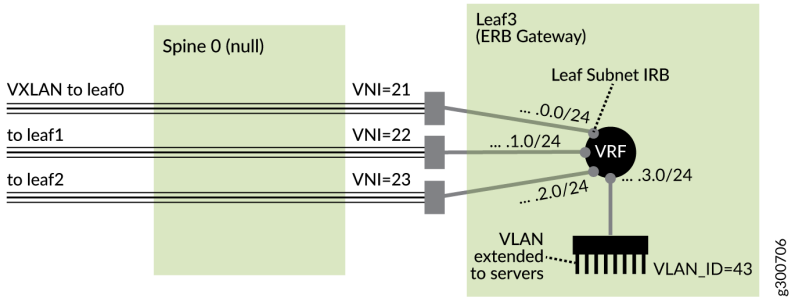
For additional information on ERB and CRB architectures, see Centrally-Routed Bridging Overlay Design and Implementation and Edge-Routed Bridging Overlay Design and Implementation in the Cloud Data Center Architecture Guide.

Figure 4: Network Connectivity Details



NOTE: The diagram shows a logical view of the networks. All traffic between leaf switches passes through a spine switch and not directly between leaf switches.

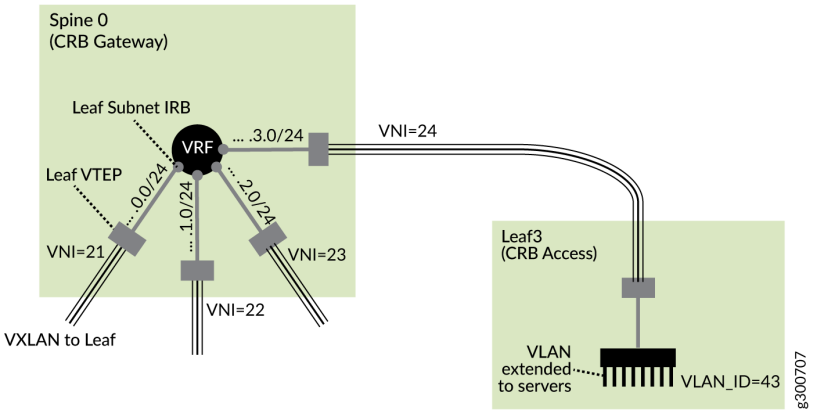
Figure 5: VXLAN Connectivity for a Leaf Switch in an ERB Environment



The Virtual Routing and Forwarding (VRF) instances are created on the spine devices and provide VXLAN tunnels to each leaf device when centralized routing and bridging (CRB) is used.

Figure 6 on page 21 illustrates VXLAN connectivity for a leaf device in a CRB environment.

Figure 6: VXLAN Connectivity for a Leaf Switch in a CRB Environment



The main features of this network design:

Table 10: VLAN Summaries

Device	VLAN Summary

Management Switch	The IPMI, Management, Provisioning, and Intranet networks all connect the management switch to a server on different ports. The port traffic for each network arrives on the management switch untagged and is configured into the same VLAN. The VLAN is extended between the management switches on different racks.
Leaf Switch (EVPN-VXLAN Fabric)	Traffic from the external, internal API, tenant, storage, and storage management networks arrives on the leaf switch's high-speed interfaces from the servers in different VLANs. The VLANs are configured in VTEPs. Leaf ports are configured with logical interfaces that have VLANs for the networks that will be attached by servers to that port. Each VLAN on each switch is configured into a VXLAN VTEP and EVPN advertises host routes for each connected server to the spines. The VLANs used for each network are specified in the file overcloud-nics.yml.
Spine Switch (EVPN-VXLAN Fabric)	The spine switches are configured with a VTEP for each of the Internal API, Tenant, Storage and Storage-Mgt networks, and each of these are connected to an IRB interface whose IP address is that of the supernet. Each spine switch has a VRF that receives routes to each host from the leaf switches.
SDN Gateway Router	The SDN gateways are configured with a VTEP and VRF for the *external* network. Each SDN gateway will advertise a route for the *external* network to peers outside the Layer 2 network.

This reference architecture does not provide fabric configuration options and details.

To configure an EVPN-VXLAN fabric that can be used by the nodes in a Contrail Cloud to transport Layer 2 and Layer 3 traffic, see Data Center Fabric Architecture Guide.

Interface Addressing and Settings

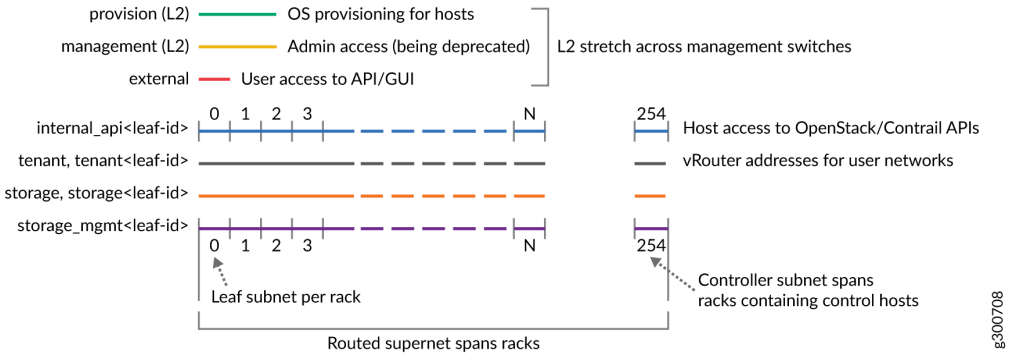
This section covers interface addressing and other interface settings.

Leaf Subnets and Supernets

Leaf subnet addresses are contained within a supernet address range. The supernet range should be large enough to contain the largest possible number of leaf subnet addresses—which is the number of racks— that could be deployed in your Contrail Cloud environment. Set your supernet address ranges thoughtfully, as you cannot change the values of supernet address ranges after the first servers are deployed.

The spans of the various networks are shown below (assuming that /24 networks are used for leaf subnets, and a /16 network is used for each supernet.

Figure 7: Leaf Subnets Within Supernets



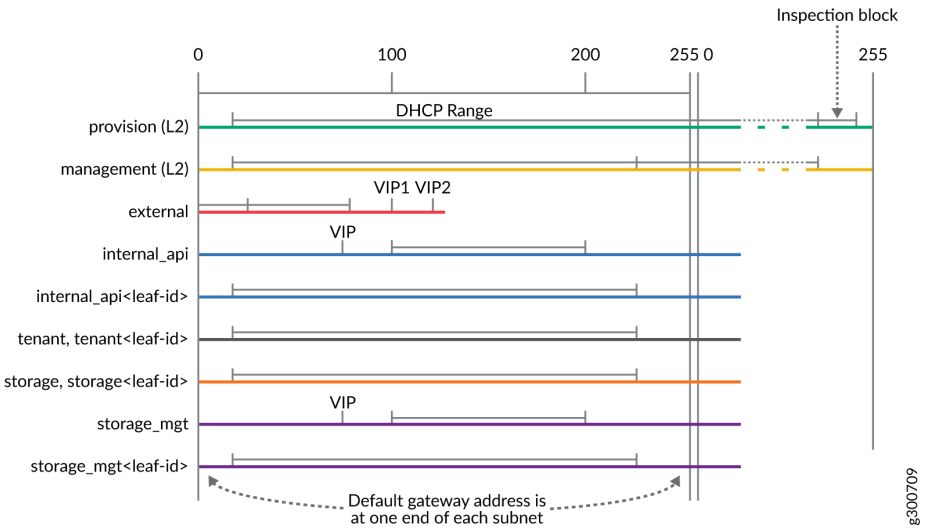
As an example, if the supernet for the internal_api network is 172.18.0.0/16 then the leaf subnets could be 172.18.0.0/24 (Leaf0), 172.18.1.0/24 (Leaf1), and 172.18.2.0/24 (Leaf2).

The subnet addresses used by control hosts are put at the high end of the supernet range in order to allow synchronization of the leaf devices with their subnet addresses.

Address Allocation Within Networks

Contrail Cloud networks are subdivided into groups of addresses for different purposes. Only part of the subnet is used. See Figure 8 on page 24.

Figure 8: Subnet Address Spaces by Network



The DHCP range for the provision and management networks is large enough—it uses a /23 CIDR—to supply addresses to all of the controller VMs and all of the compute and storage nodes. in the Contrail Cloud environment. The Internal API network needs to supply addresses only to control hosts and the VMs running on them, so its DHCP range is small. The external subnet is typically allocated by a datacenter administrator and only has to accommodate addressing for the controller functions that have external access. The VIPs that will be the public access IP addresses for GUIs and APIs, and AppFormix VM addresses are statically configured outside the DHCP range of the external network. The base subnets for the internal_api and storage_mgmt networks have a small DHCP range and a VIP is allocated outside that range. The leaf-specific subnets for internal_api and storage_mgmt networks and all the subnets for tenant and storage networks simply have a DHCP range allocated that is large enough to accommodate all the servers that can be placed in one rack.

The address of the default gateway is usually the first or last address of a subnet. The exception is the external network, whose default gateway is determined by the data center architecture.

Use of IPv6 in Contrail Cloud networks

Red Hat supports IPv6 in all networks. Contrail Cloud does not support IPv6 in the *provision*, *internalAPI*, and *tenant* networks, however, because the Contrail Cloud vRouter does not support IPv6 for compute node addresses and some Contrail API components.

MTU settings

Packets passing between VMs are encapsulated in MPLSoUDP, MPLSoGRE, or VXLAN headers. These headers increase packet sizes. MTU settings in configuration files must account for these packet size increases.

Table 11 on page 25 provides the MTU setting requirements when the switching environment supports jumbo frames of 9000 bytes.

Table 11: MTU Setting Requirements

Entity	MTU setting
Switch ports	9216
Interfaces on bare metals servers Controllers VMs defined by overcloud networks	9100
Tenant VM interface	1500 or 9000

The difference in MTU between the various entities is more than actually needed to avoid fragmentation. These MTU settings can be tuned more precisely, if needed for your environment.

If jumbo frames cannot be configured in the switch ports, then the MTU on server ports and tenant VM interfaces should be reduced to allow passage of VXLAN and Ethernet headers without fragmentation (typically, 1400 bytes for server ports and 1300 bytes for tenant VM interfaces).

NOTE: Due to limitations in the Linux ip netns feature, if SNAT or LBaaS are used with DPDK, the MTU must be set to less than 1500 bytes. This limitation is documented in the Contrail 1910 release notes as issue JCB-177787.

Interface Naming Conventions

The examples in this reference architecture use a numeric convention instead of named interfaces to identify interfaces. This convention allows the same configuration file snippets to manage servers of different types in scenarios where the interface names seen in BIOS and OS differ.

Table 12 on page 26 provides a naming schema to use if interface names are used in place of numeric conventions.

For example, a server with 6 interfaces— 2 1G interfaces and 4 10G interfaces—could use the following names and corresponding “numeric” names.

Table 12: Interface Naming Convention Schema Example—2 1G and 4 10G interfaces.

Interface Name	Numeric Name
eno1	nic1
eno2	nic2
ens7f0	nic3
ens7f1	nic4
ens7f2	nic5
ens7f3	nic6

NOTE: The number of NICs correspond to the physical order of NICs in a server (PCI Address).

NOTE: Red Hat numeric naming uses the term nicN for each interface. Physical NICs usually contain two or four interfaces per NIC.

After the introspection phase (running ./inventory-assign.sh), the names and order of interfaces on a server can be found by logging into the undercloud VM and running the **openstack baremetal introspection interface list *node-name*** command:

```
openstack baremetal introspection interface list 5a6s1-node5
+---------+-----------------+----------------------+-----------------+-------+
 Interface  MAC               Switch Port            Switch            Switch
            Address           VLAN IDs               Chassis ID        Port ID
+---------+-----------------+----------------------+-----------------+-------+
| eno1    | 0c:c4:7a:81:a5:92| [300,301,302,303,304] | 40:71:83:56:c2:80| 654   |
| eno2    | 0c:c4:7a:81:a5:93| [301,302,303,304,1008]| 40:71:83:56:c2:80| 503   |
| ens7f0  | 0c:c4:7a:b7:26:7c| [305,306,307,308,309] | 40:71:83:31:67:40| 532   |
| ens7f1  | 0c:c4:7a:b7:26:7d| [305,306,307,308,309] | 40:71:83:31:67:40| 617   |
```

```
| ens7f2  | 0c:c4:7a:b7:26:8a| [305,306,307,308,309] | 40:71:83:31:67:40| 618   |
| ens7f3  | 0c:c4:7a:b7:26:8b| [305,306,307,308,309] | 40:71:83:31:67:40| 619   |
+---------+------------------+----------------------+------------------+-------+
```

RELATED DOCUMENTATION

- Contrail Cloud Deployment Guide
- Contrail Cloud TechLibrary page

Pre-Deployment Checklist

IN THIS SECTION

This section covers tasks that must be completed before deploying Contrail Cloud 13.1 or 13.2.

Shared Configuration Between Network and Servers

The following information must be consistent between the Contrail Cloud configuration and the network devices that are connected to the servers:

- VLAN IDs for the fast-interface networks.
- ESI must match on the pairs of switch ports for the same bond connections.
- VRFs for routing between rack subnets must be configured on each leaf switch.
- External network must be reachable by users outside the cloud environment.

- MTU must be correctly configured in the fabric, physical servers, and overcloud roles.

Networking Infrastructure

The network infrastructure must meet the following requirements:

- Every server must have access to the Contrail Cloud repository satellite or to an instance of Capsule Server that is connected to it. The satellite is used to distribute packages and control software versions.
- The jumphost must have access to the Intelligent Platform Management Interface (IPMI) or to the vendor-specific equivalent of every managed server.
- The jumphost must have an interface in the same broadcast domain as each managed server's provisioning interface to allow PXE booting (*provision* network). This setup can be created by stretching a Layer 2 network across the interfaces using technologies like VXLAN or VLAN.

  DHCP Relay in the network fabric is not supported.

  To configure an IP Fabric with EVPN-VXLAN in a Contrail Cloud environment, see the Data Center Fabric Architecture Guide.
- The networking for management switches and IP Fabric has been configured as described in the previous sections of this document.
- The undercloud running on the jumphost must be the only DHCP server in the network.
- Switch ports on the provisioning network should be configured to skip STP negotiation.
- If SR-IOV will be used on a compute node, the feature needs to be enabled in the BIOS of all compute nodes using SR-IOV.

Jumphost

The undercloud is deployed as a virtual machine on a Linux kernel-based virtual machine (KVM) on the jumphost.

You must ensure the jumphost:

- Runs Red Hat Enterprise Linux (RHEL) 7.6 installed with the base packages only.

- Does not run virtual machines other than the undercloud and Contrail Command. The Contrail Cloud provisioning scripts often delete other virtual machines running on the jumphost during the Contrail Cloud installation.
- Has a network connection that can reach the Contrail Cloud Satellite or an instance of Capsule proxy, and also has IPMI access to physical hardware.
- Has a network connection that can be used for provisioning the control hosts and VMs, compute nodes, and storage nodes.
- Meets the memory requirements—RAM, vCPUs and disks—as specified in this document. See "Contrail Cloud Software" on page 5.
- Supports users, including a root user who has password-free sudo access and ssh access to the localhost.
- Resolves Internet and satellite site addresses with DNS.
- Is time synchronized with an NTP source.

Overcloud Hosts

The overcloud hosts—storage nodes, compute nodes, and control hosts—must meet the following requirements:

- Hardware RAID for OS partitions must be configured.
- The boot order must be configured to boot from PXE first and to boot from disk second.
- PXE on interface used for provisioning must be enabled. You do not need to enable PXE on all interfaces.
- The latest BIOS version must be running.
- The latest NIC firmware must be running.
- Verify BIOS boot mode (legacy or UEFI). If the boot mode is UEFI, enable the UEFI boot mode in the *capabilities: boot_mode:* hierarchy in the **inventory.yml** file.

RELATED DOCUMENTATION

Contrail Cloud Deployment Guide

Contrail Cloud TechLibrary page

Contrail Cloud Configuration

IN THIS SECTION

This chapter covers Contrail Cloud configuration.

Contrail Cloud Configuration File Structure Overview

Contrail Cloud is configured using YAML files. A series of pre-configured YAML file templates are provided as part of a Contrail Cloud installation. These user-configurable YAML files are downloaded onto the jumphost server during the initial phase of the Contrail Cloud installation. The YAML files can be accessed and edited by users from within the jumphost.

Contrail Cloud configuration changes are applied using Ansible playbook scripts, which are also provided as part of the Contrail Cloud bundle. The Ansible playbooks read the configuration provided in the YAML files. The Ansible playbook scripts populate parameters from the YAML files into a second set of configuration files that are used by RedHat Openstack Director to provision servers and configure the components of Contrail Cloud.

See Deploying Contrail Cloud for additional information on YAML file locations and configuration updating procedures.

Table 13 on page 31 lists commonly-used YAML file parameters in Contrail Cloud and provides a summary of the purpose of each parameter.

Table 13: YAML File Parameters

YAML File Parameter	*Purpose*
site.yml	
global:	DNS, NTP, domain name, time zone, satellite URL, and proxy configuration for the deployment environment.
jumphost:	Provision NIC name definition and PXE boot interface for the jumphost.
control_hosts:	Control host parameters. Includes disk mappings for bare metal servers and control plane VM sizing per role for functions like analytics.
compute_hosts:	Parameters for SR-IOV, DPDK, and TSN in compute nodes. Root disk configuration per hardware profile.
storage_hosts:	Ceph and block storage profiles definition for storage nodes.
undercloud:	Nova flavors for roles. Applicable when using additional hardware profiles.
overcloud:	Hardware profile and leaf number-based: • disk mappings • network definitions—names, subnets, VLANs, DHCP pools, and roles for network. Other network definitions like TLS cert, keystone LDAP backend enablement, post deployment extra actions, tripleO extra configurations
ceph:	Ceph enablement and disk assignments (pools, OSDs) on storage nodes.
ceph_external:	Externally deployed Ceph integration parameters.
appformix:	Enable HA, VIP IPs, and network devices monitoring for Appformix.

inventory.yml	
inventory_nodes:	Name, IPMI IP, Ironic driver used for LCM, root disk, and other related functions for all Contrail cluster nodes.
control-host-nodes.yml	
control_host_nodes:	Internal IP and DNS (per control node) for control hosts and the control plane.. Statically added IPs for controllers need to be outside of DHCP pools for networks that use them.
control_host_nodes_network_ config:	Bridges, bonds, DHCP/IP, and MTU for control hosts.
control_hosts:	VM interface to bridge on control-host mapping.
overcloud-nics.yml	
contrail_network_config: controller_network_config: appformixController_network_config: computeKernel_network_config: compute_dpdk_network_config: cephStorage_network_config:	Interface to network mapping, routes, DHCP-IP allocation, bonds, VLAN to interface maps, and bond options for control, storage, and compute nodes.
compute-nodes.yml	
compute_nodes_kernel: compute_nodes_dpdk: compute_nodes_sriov:	Mapping hosts from inventory to compute roles and profiles for compute nodes.
storage-nodes.yml	
storage_nodes:	Names of storage nodes.
vault-data.yml	
global:	satellite key and contrail user password for the Red Hat Open Stack Vault function.

undercloud: overcloud: control_hosts:	VM & Bare metal server (BMS) passwords for Contrail Cluster nodes and the undercloud when using the Red Hat Open Stack Vault function.
appformix:	MySQL and RabbitMQ passwords for Appformix when using the Red Hat Open Stack Vault function.
ceph_external:	Client key used by Ceph External with the Red Hat Open Stack Vault function..
inventory_nodes:	IPMI credentials for Contrail cluster nodes when using the Red Hat Open Stack Vault function.

The Ansible playbooks initially read variables from the **default.yml** file. Configuration files are then read in the order presented in Table 13 on page 31. Variables are stored and updated in the **default.yml** file as the script runs. If the same variable has different configuration settings in different YAML files, the setting in the YAML file that is read later in the configuration file processing order is implemented.

Sample YAML files should be copied from the **/var/lib/contrail_cloud/samples** directory to the **/var/lib/contrail_cloud/config** directory and updated according to the requirements of the current deployment. Parameter values in the files in the **/var/lib/contrail_cloud/samples** directory can be used as default values in most cases where guidance for setting values is not given in this document. See Contrail Cloud Deployment Guide.

Hardware Profiles

A hardware profile allows administrators to apply the same configuration to a group of servers acting as compute or storage nodes.

As new servers are added to a deployment, each server might have different disk and network hardware. The method that networks and storage are configured may differ between servers.

Servers are associated to hardware profiles in the **compute-nodes.yml** file. The leaf number is also set in this file. This sample configuration snippet from the **compute-nodes.yml** file shows a hardware profile configuration:

```
compute_nodes_kernel:
  - name: compute-1-rack0 #Compute name
    leaf: '0'    #Leaf number
    profile: hw0  #Server Hardware profile tag
  - name: compute-1-rack1
```

```
    leaf: ‘1'
    profile: hw1
  - name: compute-1-rack2
    leaf: '2'
    profile: hw1
  - name: compute-2-rack2
    leaf: '2'
    profile: hw2
```

The hardware profile configurations are applied to servers using the **site.yml** and **overcloud-nics.yml** files, using the sections of the files that use the following semantic:

```
[role][leaf number][hardware profile tag]
```

where:

- The role is one of the following values: ComputeKernel, ComputeDPDK, ComputeSriov, CephStorage
- The leaf number is the number of the leaf. For instance, **0**.
- The Hardware profile tag is any alphanumeric string starting with a capital letter. We strongly recommend using the **Hw*[number]*** convention.

For example, the following sample names could be used for compute nodes of different types in leaf 0:

- **ComputeKernel0Hw1** for kernel-mode
- **ComputeDpdk0Hw1** for DPDK
- **ComputeSriov0Hw1** for SR-IOV

Two sample hardware profile configurations within the **site.yml** file are provided below. The example allocates different SCSI disks for local VM ephemeral storage using variable HCTL.

```
overcloud:
  # Contains a list of label to disk mappings for roles
  disk_mapping:
     ComputeKernel1Hw0: #compute in leaf 1 with hardware profile label ‘hw0’
      - label: ephemeral-0
        hctl: '7:0:0:0'
      - label: ephemeral-1
        hctl: '8:0:0:0'
    ComputeKernel2Hw1: #compute in leaf 2 with hardware profile label ‘hw1’
```

```
      - label: ephemeral-0
        hctl: '5:0:0:0'
      - label: ephemeral-1
        hctl: '6:0:0:0'
```

```
overcloud:
[...]
  network:
[...]
   internal_api0:
       heat_name: InternalApi0
       cidr: "172.16.1.0/24"
       default_route: 172.16.1.1
       role:
        - ComputeDpdk0Hw1
        - ComputeSriov0Hw1
        - ComputeKernelHw1
[...]
```

Example of hardware profiles in the **overcloud-nics.yml** file are provided later in this reference architecture.

Network Configuration

This section describes how the networks are configured in Contrail Cloud. All properties of the networks—with the exceptions of the IPMI and Provision networks—are specified in the *network:* section within the **site.yml** file.

IPMI Network

The IPMI network is generally a set of network addresses reserved for hardware management within a data center. The management switches must be configured with access to the default gateway for the IPMI network. Servers in the environment must have IP addresses statically allocated for IPMI or sent via DHCP using the MAC address as the key for address allocation.

Provision Network

The properties of the provision network are configured in the *undercloud:* section of the **site.yml** file.

A sample configuration snippet from the **site.yml** file:

```
undercloud:
  vm:
    network:
      provision:
        #undercloud VM ip
        ip: 192.168.212.1
        cidr: "192.168.212.0/23"
        gateway: 192.168.212.1
        #undercloud_dhcp_start (from this range hosts will have IPs in provision network
        dhcp:
          start: 192.168.212.20
          end: 192.168.213.200
      inspection:
        #undercloud_inspection_ip_range
        ip_range:
          start: 192.168.213.201
          end: 192.168.213.253

overcloud:
  network:
    control:
      # In Contrail Cloud and recent tripleO the network is called provisioning
      heat_name: ControlPlane
      default_route: 192.168.212.1
      cidr: "192.168.212.0/23"
      mtu: 9100
```

NOTE: The control network in the overcloud is the same as the provision network in the undercloud.

NOTE: The example configuration provides for up to two hundred compute nodes and fifteen IP address for control and storage nodes.

NOTE: The inspection block specifies a range of IP addresses that the installer introspection service uses during the PXE boot and provisioning process. Use the *ip_range* variable to define the start and end values within this range. When batch provisioning is used—which is recommended in this reference architecture—only a small number of these addresses are in use at the time. The range, therefore, can be much smaller than the DHCP range.

A server PXE boots from the provisioning network and receives an IP address via DHCP when it is provisioned. The boot preference is then changed to disk, the IP address is configured by the Ironic service into the operating system, and the server boots from the disk with the same IP address.

External Network

The External network is used for cloud users to access the public API addresses of the control hosts. A VIP IP address as well as a pool of IP addresses that can be used by DHCP is specified.

The External network parameters are set in the **site.yml** file.

A sample External network configuration snippet from the **site.yml** file:

```
network:
   external:
     cidr: "192.168.176.0/25"
     vlan: 305
     vip: "192.168.176.100"
     pool:
       start: "192.168.176.10"
       end: "192.168.176.99"
     mtu: 9100
```

There are only a limited number of control hosts. The external network, therefore, can be a small subnet of IP addresses.

The External network is associated with an interface on control hosts in the **control-host-nodes.yml** files.

Management

The properties of the management network are configured in the *network:* section of the **site.yml** file.

A sample Management network configuration snippet from the **site.yml** file:

```
overcloud:
  network:
[...]
    management:
      heat_name: Management      # Network name used by TripleO Heat Templates
      cidr: "192.168.0.0/23"
      mtu: 9100
         start: 192.168.0.5
        # Range end for the DHCP pool
        end: 192.168.1.220
[...]
```

Internal API, Tenant, Storage, and Storage Management Networks

Red Hat Openstack 13 (RHOSP 13) supports the capability to use separate subnets per rack. This feature is used for the networks that are connected to the IP Fabric in this reference architecture. These are the internal API, tenant, storage and storage management networks. Each of these networks is assigned a supernet IP address range (/16), which includes all the rack subnets (/24) for that network.

The concept of spine-leaf networking in TripleO is described in

Red Hat OpenStack Platform 13: Spine Leaf Networking

TripleO uses the term leaf to group servers that have shared connectivity. In this Contrail Cloud reference architecture, leaves in the TripleO context refer to grouped servers in the same rack. In this reference architecture, a Red Hat leaf is implemented by a management switch and a pair of top of rack switches (ToR switches).

The names of networks in configuration files follow the convention established in the compute and storage nodes for a rack, or a leaf. The names are a concatenation of base network name and leaf number. Leaf numbers are assigned to nodes in the **compute_nodes.yml** file. Base network names include InternalApi, Storage, StorageMgmt, and Tenant networks.

Compute nodes in Leaf 0 should use networks:

- InternalApi0
- Storage0
- Tenant0

Compute nodes in Leaf 1 should have defined networks:

- InternalApi1
- Storage1
- Tenant1

Compute nodes in Leaf N should have defined networks:

- InternalApiN
- StorageN
- TenantN

For an example of splitting a supernet, see the example below for the Internal API and Tenant networks. The same procedure must be performed for the remaining networks used for compute and storage nodes (Storage, Storage Mgmt).

In the following configuration snippet, these parameters are set for all networks:

- supernet - supernet definition.
- cidr - subnet configuration for a leaf with the first subnet used for the controllers.
- default_route - static route definition pointing to a “supernet” via a given operating system interface, such as bond1, vhost0, or other interfaces.
- vrouter_gateway - default route definition for vrouter encapsulated traffic in the overlay network. This variable is defined as a gateway parameter in the **/etc/contrail/contrail-vrouter-agent.conf** file. This gateway IP address is used to reach DC gateways (MX routers and other vRouters to setup MPLSoUDP or other overlays).
- role - the role or hardware profile that assigns the subnet. The first subnet is always for controllers and the remaining subnets are assigned to compute and storage nodes based on the leaf identifier.

When a network is specified to share an interface in the ***role*_network_config.yml** file, the network is assigned a VLAN. The VLAN can be identical in all racks when VXLAN is used, which is how the VLANs are labelled in this reference architecture.

```
overcloud:
 network:
  internal_api:                   # This is the network for control nodes (no suffix)
      heat_name: InternalApi
      supernet: "172.16.0.0/16"
      cidr: "172.16.0.0/24"
      default_route: 172.16.0.1
```

```
    vlan: 100
    mtu: 9100
    pool:
      start: 172.16.0.100
      end: 172.16.0.199
    vip: 172.16.0.90
    role:
      - ContrailController
      - ContrailAnalytics
      - ContrailAnalyticsDatabase
      - ContrailTsn
internal_api0:                  # Network for nodes attached to leaf 0
    heat_name: InternalApi0
    cidr: "172.16.1.0/24"       # Leaf subnet
    default_route: 172.16.1.1
     vlan: 100
     mtu: 9100
     vip: false
    pool:
      start: 172.16.1.100
      end: 172.16.1.200
     role:
      - ComputeDpdk1Hw2
      - ComputeSriov1Hw4
      - ComputeKernel1Hw1
      - ComputeKernel1Hw0
internal_api1:                  # Network for nodes attached to leaf 1
     heat_name: InternalApi1
     cidr: "172.16.2.0/24"
     default_route: 172.168.2.1
     vlan: 100
     mtu: 9100
     vip: false
     pool:
       start: 172.16.2.100
       end: 172.16.2.199
     role:
      - ComputeDpdk1Hw3        # Different hardware profiles for this leaf
      - ComputeSriov1Hw5
      - ComputeKernel1Hw1
      - ComputeKernel1Hw0
internal_api2:                  # Network for nodes attached to leaf 2
     heat_name: InternalApi2
```

```
        cidr: "172.16.3.0/24"
        default_route: 172.16.3.1
        vlan: 100
        mtu: 9100
        vip: false
        pool:
          start: 172.16.3.100
          end: 172.16.3.199
        role:
         - ComputeDpdk2Hw3
         - ComputeSriov2Hw5
         - ComputeKernel2Hw1
         - ComputeKernel2Hw0

  internal_api3:
[...]

  tenant:                           # This is the network for control nodes (no suffix)
      heat_name: Tenant
      supernet: "172.18.0.0/16"
      cidr: "172.18.0.0/24"
      default_route: 172.18.0.1        # passed to host OS
      vrouter_gateway: 172.18.0.1      # passed to vRouter for encapsulated traffic
      vlan: 200
      mtu: 9100
      vip: false
      pool:
         start: 172.18.0.100
         end: 172.18.0.199
      role:
        - ContrailController
        - ContrailAnalytics
        - ContrailAnalyticsDatabase
        - ContrailTsn
  tenant0:
      heat_name: Tenant1
      cidr: "172.18.1.0/24"
      default_route: 172.18.1.1        # passed to host OS
      vrouter_gateway: 172.18.1.1      # passed to vRouter for encapsulated traffic
      vlan: 200
      mtu: 9100
      vip: false
      pool:
```

```
        start: 172.18.1.100
        end: 172.18.1.199
    role:
      - ComputeDpdk0Hw2
      - ComputeSriov0Ww4
      - ComputeKernel0Hw0
      - ComputeKernel0Hw1
tenant1:
    heat_name: Tenant1
    cidr: "172.18.2.0/24"
    default_route: 172.18.2.1
    vrouter_gateway: 172.18.2.1
    [...}
tenant2:
    heat_name: Tenant2
    cidr: "172.18.3.0/24"
    default_route: 172.18.3.1
    vrouter_gateway: 172.18.3.1
    [...]
tenant3:
    heat_name: Tenant3
    cidr: "172.18.4.0/24"
    default_route: 172.18.4.1
    vrouter_gateway: 172.18.4.1
    [...]
storage                             # Storage and storage_mgt have same format
[...]
storage_mgmt
[...]
```

Example Networks Used in this Reference Architecture

Table 14 on page 42 presents a sample addressing scheme in a Contrail Cloud environment with four racks. This addressing scheme is used in the configuration file examples in this reference architecture.

Table 14: Addressing Scheme

Network	Supernet	Subnet

Provision		192.168.212.0/23
External		10.10.10.0/25
internal_api	172.16.0.0/16	172.16.0.0/24
internal_api[0-3]		172.16.1-4.0/24
management		192.168.0.0/23
Storage	172.19.0.0/16	172.19.0.0/24
storage[0-3]		172.19.1-4.0/24
storage_mgmt	172.20.0.0/16	172.20.0.0/24
storage_mgmt[0-3]		172.20.1-4.0/24
Tenant	172.18.0.0/16	172.18.0.0/24
tenant[0-3]		172.18.1-4./24

NOTE: In the **site.yml** file, the provision network is subdivided into an inspection block with addresses that are used during PXE booting. These addresses are configured onto servers during provisioning.

Supernet addresses are specified for networks that contain a rack with a separate subnet. The supernet address is used in a static route on servers to send inter-rack traffic through the correct interface and corresponding VLAN.

Batch Deployment Configuration

Juniper Networks recommends running compute and storage node deployments in batches of 5 to 10 nodes. We make this recommendation based on the potential for timeouts during the Triple0 Heat automation process during larger deployments.

Batch deployments are configured in the **site.yml** file. A sample batch deployment configuration snippet from the **site.yml** file:

```
# To use batch deployment, you will need to run openstack-deploy.sh script multiple times (each
run will deploy new nodes).

overcloud:
  batch_deployment:
    CephStorage: 5
    ComputeDpdk: 5
    ComputeKernel: 5
```

In this configuration, 5 CephStorage, 5 ComputeDPDK, and 5 ComputeSriov nodes are deployed when the openstack-deploy.sh script is run. The script should be run repeatedly until it reports that there are no more nodes to be deployed.

Jumphost

The jumphost is the host from which an administrator initiates provisioning of a Contrail Cloud environment. This section covers jumphost configuration options.

Jumphost Overview

The jumphost:

- hosts the undercloud. The undercloud is a VM responsible for provisioning and managing all control hosts, storage nodes, and compute nodes in a Contrail Cloud. All Contrail-related setup and configuration is performed through the undercloud in a Contrail Cloud.
- stores Contrail Cloud configuration-related files. The YAML files that configure Contrail Cloud are stored on the jumphost. The Ansible scripts that apply the configurations made in the YAML files to the Contrail Cloud nodes are also stored on the jumphost.

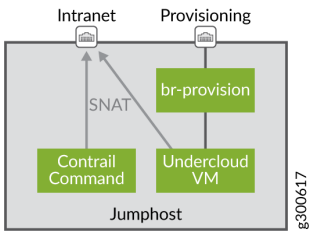
  The Contrail Cloud scripts are stored in the /var/lib/contrail_cloud directory.

- hosts the Contrail Command web user interface virtual machine.
- runs Red Hat Enterprise Linux with only base packages installed.
- provides SNAT for traffic from the Provisioning network to intranet and external networks.
- provides access to the servers of a Contrail Cloud environment if a management network is not present

A jumphost must be operational as a prerequisite for a Contrail Cloud installation. The jumphost should not run any virtual machines besides the undercloud and Contrail Command VMs.

Figure 9 on page 45 illustrates the jumphost network connections.

Figure 9: Jumphost Network Connections



The Intranet network is the network configured manually before the Contrail Cloud 13 packages are downloaded. This network is used by Contrail Cloud to download packages and to provide outside connectivity for the nodes in a Contrail Cloud environment to reach outside networks. The Intranet network IP address can be the IP address from the External network.

The jumphost is configured during the installation process to use an IP masquerade for SNAT of outbound traffic from hosts connected to the provisioning network. The “public” IP address on the jumphost—the IP address of the “Intranet” port—is used as a source address for traffic exiting the cloud during provisioning. This IP address should be permitted in firewalls to allow access to public repositories, Juniper Satellite servers, and Red Hat subscription managers. If external access is not permitted from the jumphost for security purposes, a Capsule proxy server can be used as described in Capsule Configuration. See "Miscellaneous" on page 104.

You can also configure a proxy, in which case the “Intranet” port should have access to the proxy. Note that the externally accessible VIP addresses for Openstack, Appformix, and Contrail APIs should be excluded from proxying (to avoid issues during provisioning).

A sample proxy configuration snippet from the **site.yml** file:

```
global:
  proxy:
    enabled: true
    port: 443
    host:  10.10.10.10
    exclude: '127.0.0.1,localhost, 192.168.212.1, 192.168.212.2, 10.10.10.50,10.10.10.51,
10.10.10.52'
```

Where 10.10.100.50, 10.10.10.51, 10.10.10.52 are VIPs in the External network.

Adding an IP Address for the Jumphost to the Provision Network

The jumphost should be allocated an IP address in the Provision network to enable SSH to the other hosts in the Contrail Cloud environment. This IP address allocation is a convenience to enable troubleshooting from the jumphost.

This IP address is added to the jumphost in the **site.yml** file as shown in this configuration snippet:

```
jump host:
  network:
    provision:
      # jump host nic IP to be used for provisioning (PXE booting) servers
      ip: "192.168.212.2"
      prefix: 23
```

Contrail Command Configuration

Contrail Command is a standalone solution based on two containers: contrail_command and ccontrail_psql. Contrail Command has no HA capabilities and only new UI services are provided. The Contrail Command VM is created on the jumphost.

The Contrail Command web UI can be reached in a Contrail Cloud environment by entering this URL in a web browser:
https://[*jumphost-IP-address*]:9091

Contrail Command can be accessed after a Contrail Cloud deployment without user configuration. Contrail Cloud configuration parameters are updated in the **site.yml** file.

A sample **site.yml** file configuration snippet where the Contrail Command parameters are updated:

```
command:
  vm:
   [user and password section will be used from credentials provided in vault-data.yml]
    #command_vm_cpu_count
    cpu: 16
    #command_vm_memory_size
    memory: 32
    #command_vm_disk_size
    disk: 100
    network:
      provision:
        #command_ip
        ip: 192.168.213.3
        #command_prefix
        cidr: "192.168.213.0/24"
        #command_gateway
        gateway: 192.168.213.1
```

Authentication details for Contrail Command are provided in the **vault-data.yml** file.

A sample **vault-data.yml** file configuration snippet with modified Contrail Command attributes:

```
  command:
    vm:
      # command user name
      user: "contrail"
      # password for the command vm user
      password: "c0ntrail123"
      # root password for the command VM
      root_password: "c0ntrail123"
      # backend database
      database_password: "c0ntrail123"
      # keystone admin password
      admin_password: "c0ntrail123"
      # Passphrase used to encrypt ssh key of command user.
      # If not defined ssh private key will not be encrypted.
      # ssh_key_passphrase: "c0ntrail123"
      vnc:
```

```
        # VNC console password for the command VM
        password: "contrail123"
```

Controller Node Configuration

This section describes the controller node configuration in Contrail Cloud. It also includes sections related to VM resources and networking for controller nodes.

Control Node VM Resources

This reference architecture is designed to support a large-scale Contrail Cloud environment. The following memory resources should be allocated to controller VMs to support this architecture.

Table 15: Controller VMs Requirements

Role	vCPU (Threads)	Memory (GB)	Disk (GB)
Undercloud VM	28	128	500
OpenStack Controller VM*	8	48	500
Contrail Analytics DB VM	12	48	500 & 1000
Contrail Analytics VM	12	48	250
Contrail Controller VM	16	64	250
AppFormix VM	16	32	500
TSN (Contrail Service Node)	4	8	100

Table 15: Controller VMs Requirements *(Continued)*

Role	vCPU (Threads)	Memory (GB)	Disk (GB)
Control Host OS	4	8	100

* The Openstack Controller VM size is significantly smaller than Red Hat's recommended Openstack Controller VM size. The VM uses less resources in Contrail Cloud because several network functions are handled by Contrail Networking and telemetry functions are performed by Appformix. See Red Hat OpenStack Platform 13: Recommendations for Large Deployments to see the recommended VM sizes.

NOTE: Operating system resources need to be reserved, not overtaken by resources allocated to controller VMs (assuming no oversubscription on controller hosts). There are no configuration file options to configure resources for operating systems.

The following configuration snippet from the *control_hosts:vm:* section of the **site.yml** files configures control host options:

```
control_hosts:
  vm:
    control:
      cpu: 8
      memory: 48
      disk:
        vda:
          size: 500
          pool: ssd_storage
      hv:
        - rack0-node1
        - rack1-node1
        - rack2-node1
    contrail-controller:
      cpu: 16
      memory: 64
      disk:
        vda:
          size: 250
          pool: ssd_storage
```

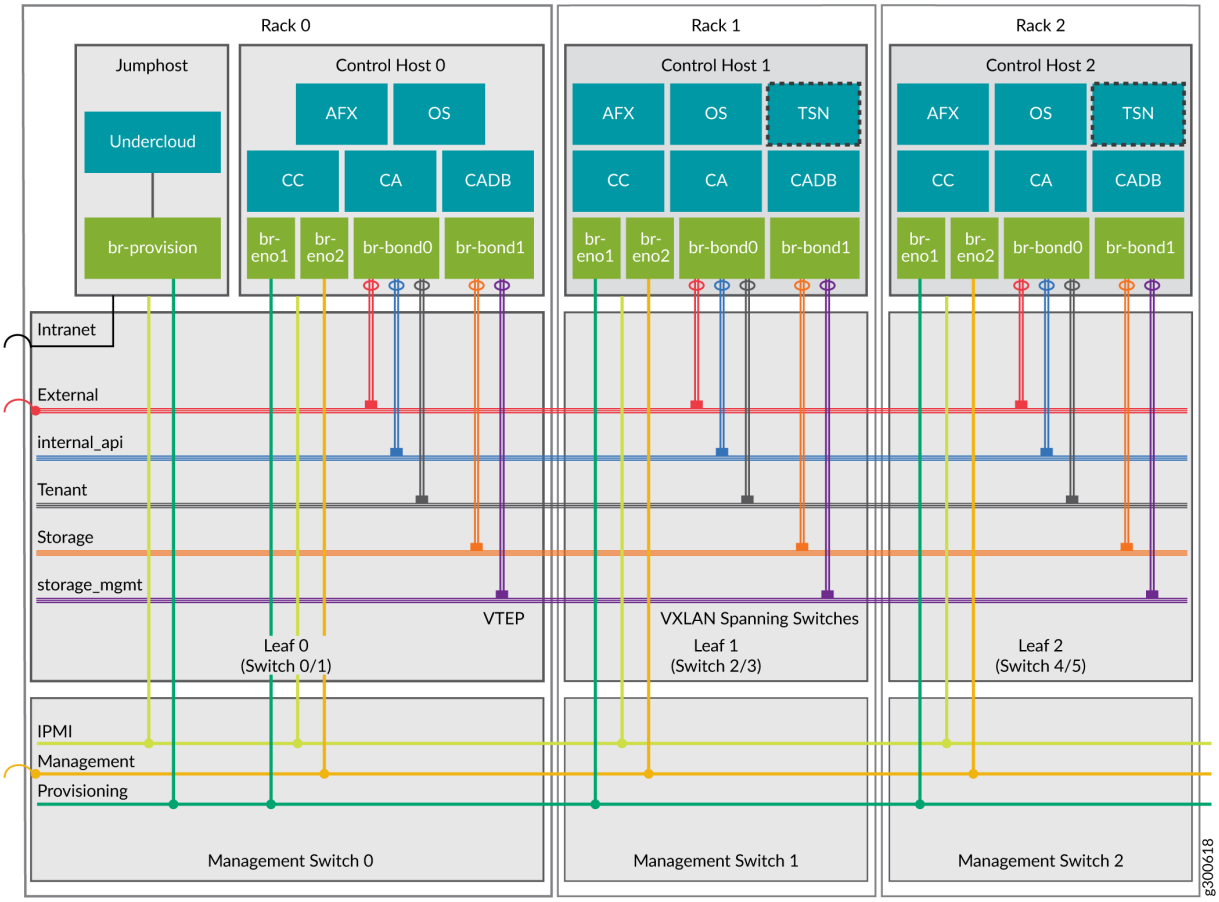
```
    hv:
      - rack0-node1
      - rack1-node1
      - rack2-node1
  contrail-analytics:
    cpu: 12
    memory: 48
    disk:
      vda:
        size: 250
        pool: ssd_storage
    hv:
      - rack0-node1
      - rack1-node1
      - rack2-node1
  contrail-analytics-database:
    cpu: 12
    memory: 48
    disk:
      vda:
        size: 500
        pool: ssd_storage
      vdb:
        size: 1000
        pool: spinning_storage
    hv:
      - rack0-node1
      - rack1-node1
      - rack2-node1
  appformix-controller:
    cpu: 16
    memory: 32
    disk:
      vda:
        size: 500
        pool: ssd_storage
    hv:
      - rack0-node1
      - rack1-node1
      - rack2-node1
  contrail-tsn:
    cpu: 4
    memory: 8
```

```
      vda:
        size: 100
        pool: default_dir_pool
    hv:
      - rack0-node1
      - rack1-node1
```

Controller Host Network Configuration

Corresponding interfaces of controller hosts are in the same L2 network, and are preferably deployed in different racks and connected to leaf devices in the EVPN-VXLAN IP Fabric. Figure 10 on page 51 illustrates these connections.

Figure 10: Control Host Network Connections



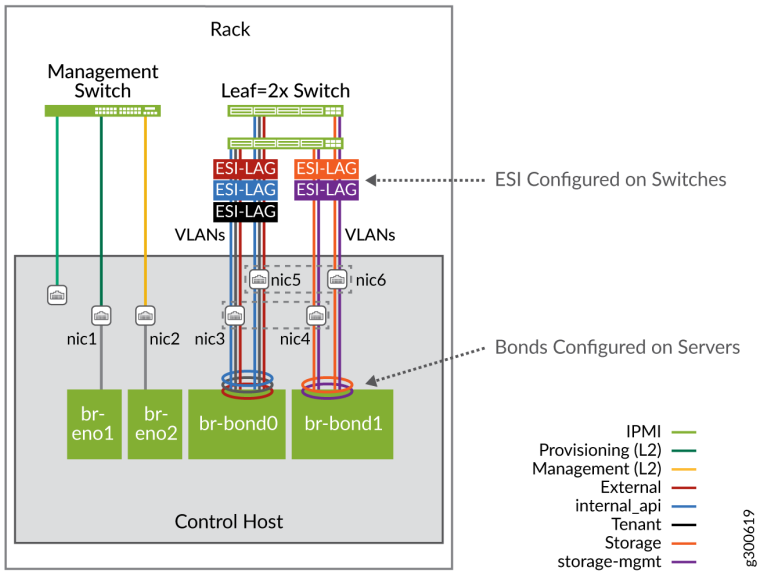
The following VMs are running on the control hosts.

- OS - Openstack Controller
- CC - Contrail Controller
- CA - Contrail Analytics
- CADB - Contrail Analytics DB
- AFX - Appformix
- TSN - ToR Service Node (optional)

  A TSN is enabled in the *control_hosts: vm: contrail-tsn:* hierarchy within the **site.yml** file. The function has been renamed CSN (Contrail Service Node) in Contrail Networking but tripleO manifests continue to use the TSN term.

Figure 11 on page 52 illustrates how the physical and logical interfaces on a control host connect to it's management switch and leaf switches.

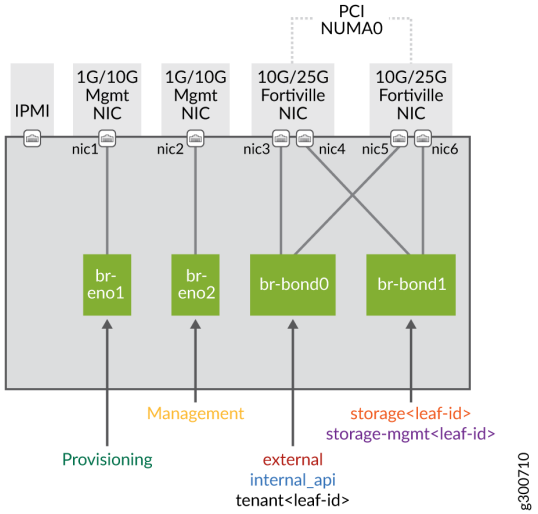
Figure 11: Control Host—Physical and Logical Interfaces



NOTE: Physical NICs typically contain multiple physical interfaces. In Red Hat configuration files, however, the naming convention is to use **nic*N*** to indicate the Nth physical interface on a server. For information on finding the order of interfaces using an introspection command, see "Miscellaneous" on page 104.

Figure 12 on page 53 illustrates how networks and ports are assigned to bridges on a controller node.

Figure 12: Control Host—Physical Interface to Bridge Allocation



Physical NICs typically contain two ports which are named nic3 and nic4 in the first physical NIC.

Table 16 on page 53 shows which files are used for configuring each network.

Table 16: Files for Configuring Control Host Networks

Port/NIC	Configuration
IPMI	Address is entered as an input to the **inventory.yml** file.
br-eno1	1 x 1G/10G NIC - untagged interface (e.g. built-in copper interface) with addresses generated automatically from the provisioning network pool.
br-eno2	1 x 1G/10G NIC - Management network, untagged interface with address defined in the *overcloud: network:* section of the **site.yml** file.
br-bond0	2 x 10G/25G/40G NICs made of first ports from both NICs. Tagged interface with networks: Tenant, Internal API and External networks. Bond physical allocation is defined in the **control-host-nodes.yml** file. Addressing is set in the *overcloud: network:* section of the **site.yml** file.

br-bond1	2 x 10G/25G/40G NICs made of second ports from both NICs. Tagged interface with networks: Storage and Storage Mgmt networks. Bond physical allocation is defined in the **control-host-nodes.yml** file. Addressing is set in the *overcloud: network:* section of the **site.yml** file.

The configuration for bond interfaces is performed in the *control_host_nodes_network_config:* hierarchy of the **control-hosts-nodes.yml** file. Bond interfaces should be configured with the following parameters:

- Linux bond - mode 4/LACP (802.3ad)
- Hash policy - layer3+4
- Use ovs-bridge and linux_bond. An OVS-bridge Linux bridge is configurable, but is not recommended by Red Hat

This configuration snippet from the **control-hosts.nodes.yml** file shows various control host configurations, including bond interface configurations.

```
control_host_nodes_network_config:
  - type: ovs_bridge
    name: br-eno1
    addresses:
    -
      ip_netmask: "{{ host.control_ip_netmask }}"
    dns_servers:
      - "{{ host.dns_server1 }}"
      - "{{ host.dns_server2 }}"
    routes:
      -
        ip_netmask: "{{ host.control_ip_netmask }}"
        next_hop: "{{ host.control_gateway }}"
        default: true
    use_dhcp: false
    mtu: "{{ overcloud['network']['control']['mtu'] }}"
    members:
      - type: interface
        name: nic1
        mtu: "{{ overcloud['network']['control']['mtu'] }}"
  - type: ovs_bridge
    name: br-eno2
```

```
    use_dhcp: false
    mtu: "{{ overcloud['network']['management']['mtu'] }}"
    members:
      -
        type: interface
        name: nic2
  - type: ovs_bridge
    name: br-bond0
    use_dhcp: false
    mtu: 9100
    members:
      - type: linux_bond
        name: bond0
        use_dhcp: false
        mtu: 9100
        bonding_options: "mode=802.3ad xmit_hash_policy=layer3+4 lacp_rate=fast miimon=100"
        members:
          - type: interface
            name: nic3
            primary: true
            mtu: 9100
          - type: interface
            name: nic4
            mtu: 9100
  - type: ovs_bridge
    name: br-bond1
    use_dhcp: false
    mtu: 9100
    members:
      - type: linux_bond
        name: bond1
        use_dhcp: false
        mtu: 9100
        bonding_options: "mode=802.3ad xmit_hash_policy=layer3+4 lacp_rate=fast miimon=100"
        members:
          - type: interface
            name: nic5
            primary: true
            mtu: 9100
          - type: interface
            name: nic6
            mtu: 9100
```

NOTE: Use validation tools to check numbered to named NICs allocations. See "Miscellaneous" on page 104.

Mapping Controller VM Interfaces to Host Bridges

The control host VM interface connections to the bridges configured in "Controller Host Network Configuration" on page 51 is done in the *control_hosts:* hierarchy of the **control-host-nodes.yml** file.

A sample configuration snippet:

```
control_hosts:
  vm_interfaces:
    - interface: eth0
      bridge: br-eno1
    - interface: eth1
      bridge: br-eno2
    - interface: eth2
      bridge: br-bond0
    - interface: eth3
      bridge: br-bond1
```

The first interface—eth0—must connect to the bridge for the provision network to allow the VM to PXE boot. The other interface names must be sequential, which matches the sample configuration snippet. You should configure one interface for each bridge.

Appformix VM Configuration

AppFormix is provided with the Contrail Cloud bundle. AppFormix provides monitoring and troubleshooting for the networking and server infrastructure of Contrail Cloud. Appformix provides the same services for the workloads running in Contrail Cloud. For additional information on Appformix, see the Appformix TechLibrary page.

Appformix provides a WebUI as well as a REST API. The WebUI and the REST API are exposed to the Internal API and External networks.

The recommended Appformix deployment—which is also the default deployment—is deployed in a 3-node configuration for high availability.

Appformix node configuration for Contrail Cloud is defined in the **site.yml** file. A sample configuration snippet:

```
appformix:
  # Set to true if you have multiple control hosts which allows Appformix to run in HA mode
  enable_ha: true
  # Floating virtual IP for the Appformix APIs on the external network, used and required by HA
mode.
  vip: "192.168.176.101"
  secondary_vip: "172.16.0.176"
  keepalived:
    # Set which interface will be used for vrrp
    vrrp_interface: "vlan305"
    # vrrp interface for secondary vip
    secondary_vrrp_interface: "vlan100"
```

The network configuration of Appformix is defined in the **overcloud-nics.yml** file.

A sample configuration snippet:

```
AppformixController_network_config:
  - type: interface
    name: nic1
    dns_servers:
      get_param: DnsServers
    use_dhcp: false
    mtu:
      get_param: ControlPlaneNetworkMtu
    addresses:
    - ip_netmask:
        list_join:
          - '/'
          - - get_param: ControlPlaneIp
            - get_param: ControlPlaneSubnetCidr
    routes:
    -
      ip_netmask: 169.254.169.254/32
      next_hop:
        get_param: EC2MetadataIp
    -
```

```
      next_hop:
        get_param: ControlPlaneDefaultRoute
    - type: vlan
      device: nic2
      vlan_id:
        get_param: InternalApiNetworkVlanID
      mtu:
        get_param: InternalApiNetworkMtu
      addresses:
      - ip_netmask:
          get_param: InternalApiIpSubnet
    - type: interface
      name: nic2
      mtu:
        get_param: ExternalNetworkMtu
      addresses:
      - ip_netmask:
          get_param: ExternalIpSubnet
      routes:
      -
        default: True
        next_hop:
          get_param: ExternalInterfaceDefaultRoute
```

The following resources are automatically monitored when AppFormix is installed in Contrail Cloud.

- Openstack API endpoints and processes
- Contrail API endpoint and processes
- Openstack MySQL
- Rabbit cluster status
- Compute nodes including vRouter, Nova compute, operating system health and metrics

Appformix is not automatically configured to monitor the physical networking infrastructure, but adapters (without any configuration) for monitoring network devices can be installed during a Contrail Cloud deployment. AppFormix must be configured manually to monitor network devices after deployment. See the Network Devices section of the Appformix User Guide.

Network-related adapters can be installed for Appformix during a Contrail Cloud deployment in the **site.yml** file.

A sample configuration snippet:

```
appformix:
  network_device_monitoring:
    appformix_install_snmp_dependencies: true
    appformix_install_jti_dependencies: true
    network_device_discovery_enabled: true
    appformix_install_ipmi_dependencies: true
```

Custom Appformix plugins can also be installed during a Contrail Cloud deploying in the **site.yml** file.

A sample configuration snippet:

```
appformix:
  enable_copy_user_defined_plugins: true
  user_defined_plugins_config: |
    - { plugin_info: 'user_defined_plugins/plugin_1.json', plugin_file: 'user_defined_plugins/
check_1.py'}
    - { plugin_info: 'user_defined_plugins/plugin_2.json', plugin_file: 'user_defined_plugins/
check_2.py'}
    - { plugin_info: 'user_defined_plugins/plugin_3.json', plugin_file: 'user_defined_plugins/
check_3.py'}
    - { plugin_info: 'user_defined_plugins/plugin_4.json', plugin_file: 'user_defined_plugins/
check_4.py'}
```

For more information, see Extensibility Using Plug-Ins in the Appformix User Guide.

Contrail Service Node (CSN) - Optional

The Contrail Service Node (CSN) provides DHCP, ARP, and multicast services when Contrail is managing the full lifecycle of bare-metal servers, including provisioning the OS. CSNs are not needed in Contrail Cloud deployments that aren't using bare-metal servers.

The triple-O templates for Contrail Cloud 13 release continue to use the Red Hat term ToR Service Node(TSN) to refer to the CSN function in Contrail Cloud. This reference architecture, therefore, uses both terms.

To enable CSN support in Contrail Cloud, edit the *compute_hosts:* hierarchy in the **site.yml** file.

A sample configuration snippet:

```
# to enable tsn support you need to set
compute_hosts:
  tsn:
    enabled: true
control_hosts:
  vm:
    contrail-tsn:
      hv:
        - control-host2
        - control-host3
```

TSN VMs are created on all controller hosts by default. We recommend running TSNs on two control hosts in Contrail Cloud environments that include at least one bare-metal server (BMS). TSN VMs should not be run in Contrail Cloud environments that are not using a BMS. You can change the number of TSN instances in the *control_hosts:* hierarchy in the **site.yml** file.

Compute Node Configuration

Compute nodes in Contrail Cloud are installed in racks and connected to a pair of top-of-rack (ToR) switches and a management switch. The ToR switches are the leaf nodes in the EVPN-VXLAN IP Fabric.

The networking for compute nodes is configured to place compute nodes in racks that compose separate Layer 3 subnets. Each rack is it's own separate layer 3 for the networks that are used by tenant workloads. Figure 13 on page 61 illustrates this compute node networking structure.

Figure 13: Compute Node Network Connections

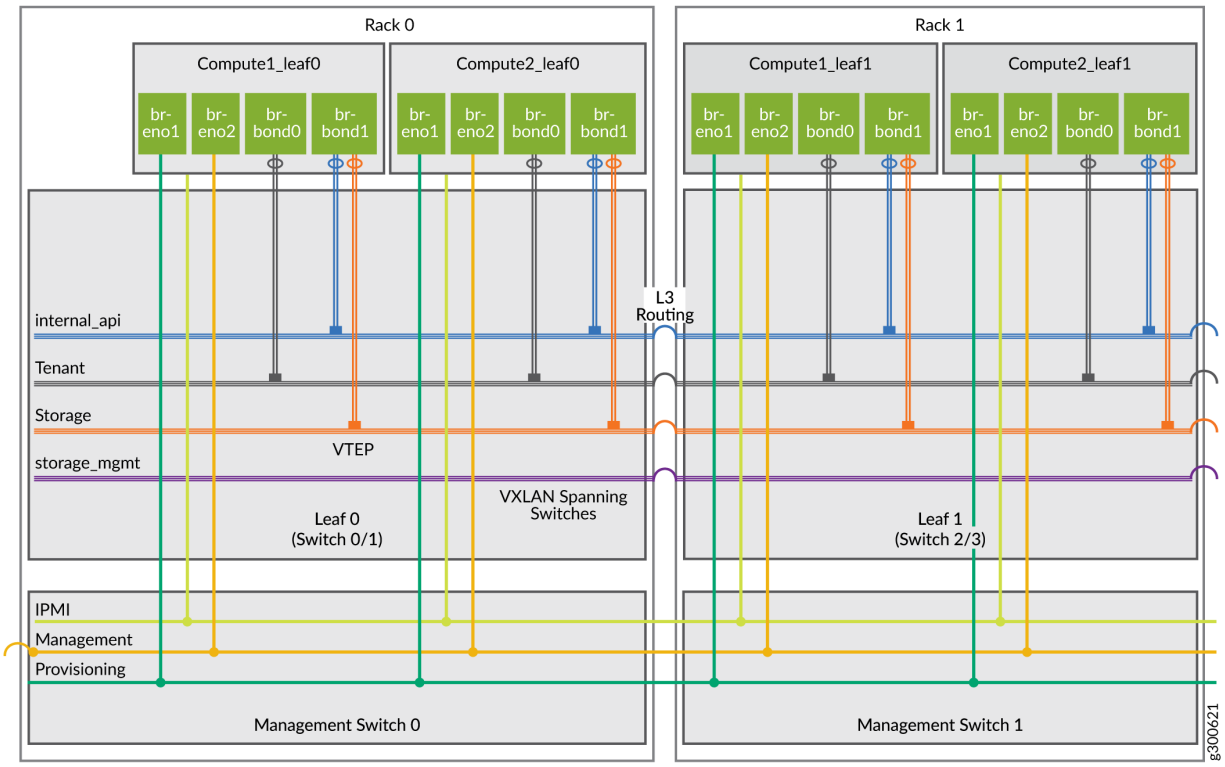


Figure 14 on page 62 illustrates how the physical and logical interfaces of a compute node connect it to a management switch and the IP Fabric leaf switches.

Figure 14: Compute Node Network Interfaces

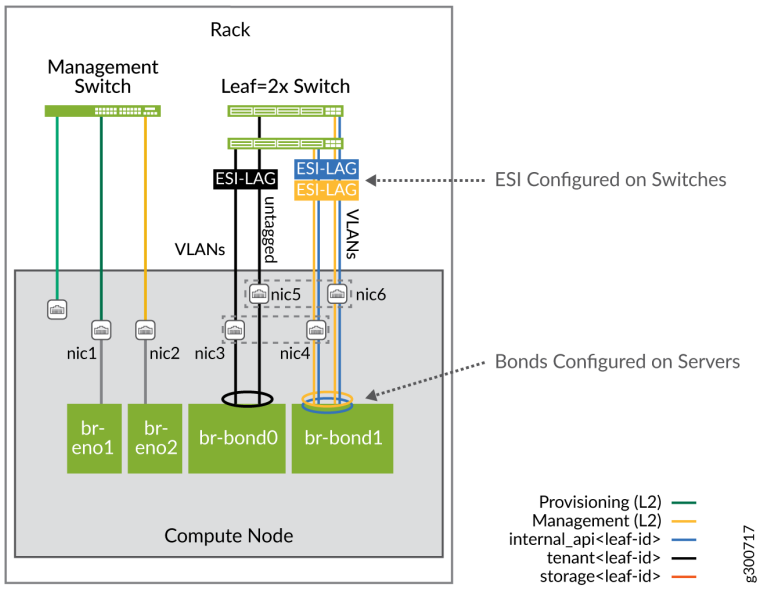


Figure 15 on page 62 illustrates networks and port assignments to bridges on a compute node.

Figure 15: Compute Node to Bridge Connections

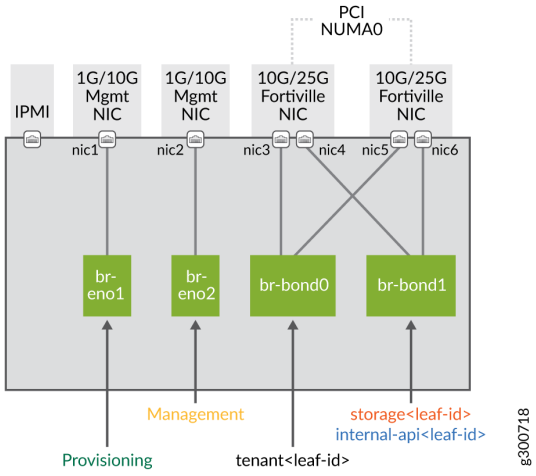


Table 17 on page 63 shows which files are used for configuring each network connection on a compute node.

Table 17: Files for Configuring Networks for a Compute Node

Port/NIC	Configuration
IPMI	Address is entered an input to the **inventory.yml** file.
br-eno1	1 x 1G/10G NIC - untagged interface (e.g. built-in copper interface) with addresses generated automatically from the provisioning network pool.
br-eno2	1 x 1G/10G NIC - Optional Management network, untagged interface with address defined in the *overcloud: network:* hierarchy in the **site.yml** file.
br-bond0	2 x 10G/25G NICs made of first ports from both NICs. Untagged interface with Tenant network. Bond physical allocation is defined in the **overcloud-nics.yml** file and addressing is set in the *overcloud: network:* hierarchy within the **site.yml** file.
br-bond1	2 x 10G/25G NICs made of second ports from both NICs. Tagged interface with Internal-API and Storage networks for the leaf. Bond physical allocation is defined in the **overcloud-nics.yml** file and addressing is set in the *overcloud: network:* hierarchy within the **site.yml** file.

Provisioning and optional Management networks are connected via out-of-band management switches and must be configured as a Layer 2 stretch across the management switches.

Compute Node Networking

The data plane interfaces of compute nodes can be configured to support the following forwarding methods.

Table 18: Compute Node Forwarding Methods

Kernel-mode	The vRouter forwarding function is performed in the Linux kernel by replacing the default Linux bridge code with Contrail Networking code.
DPDK	vRouter runs in a user space in a specified number of cores.
SR-IOV	VM or container interface connects directly to the NIC, bypassing the vRouter.

Your traffic forwarding method choice depends on the traffic profile expectations for each individual compute node. A Contrail Cloud environment can have different compute nodes configured with different interface types, and workloads can be placed on the most appropriate compute node using various technologies, such as OpenStack availability zones.

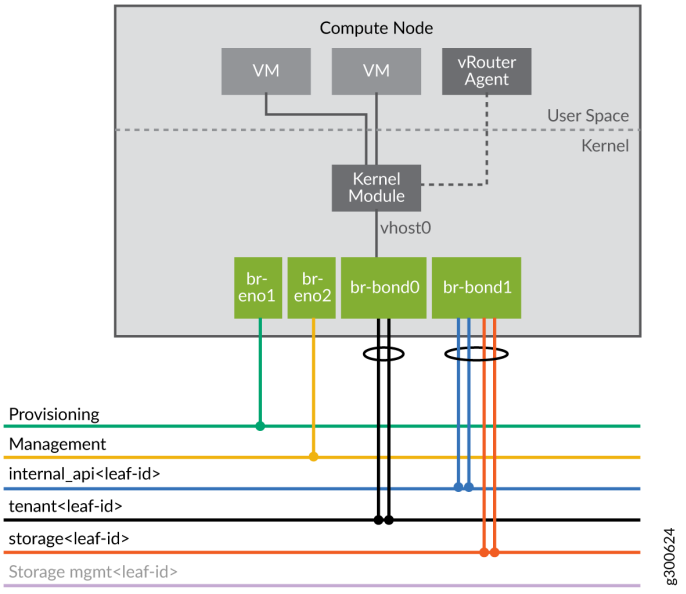
The IP address of the management interface and the addressing and configuration of bond interfaces are configured in the **overcloud-nics.yml** file.

Kernel-Mode vRouter Configuration

The vRouter vhost0 interface is connected into the br-bond0 bridge in compute nodes which run the vRouter in kernel-mode. The br-bond0 bridge is connected via a bond interface to the Tenant network.

Figure 16 on page 64 illustrates these connections.

Figure 16: Kernel Mode vRouter



Compute Resources

The following guidelines should be following to optimize vRouter performance.

- Guarantee minimum 4 cores for host operating system and minimum 2 of them can be used by vRouter kernel module. There is no mechanism to allocate cores but we assuming that Host OS

processes will consume no more than 2 cores and remaining 2 a kernel scheduler will allocate for vRouter.

- Guarantee minimum 8GB RAM for host operating system where 4GB will be used for vRouter.

vRouters running in kernel mode should achieve up to 500kpps per vRouter (<1000 flows in a table) when these guidelines are followed.

Bond Interface Configuration

The following options should be set for bond interfaces when kernel mode is used:

- bond_mode: 4 (IEEE 802.3ad)
- bond_policy: layer3+4

The bond configuration is defined in profile definitions in the **overcloud-nics.yml** file:

```
ComputeKernel0Hw0_network_config:
 [...] #name, DHCP, MTU etc settings
 - type: linux_bond
    name: bond0
    use_dhcp: false
    bonding_options: "mode=802.3ad xmit_hash_policy=layer3+4 lacp_rate=fast updelay=1000
miimon=100"
```

Optimizations

We recommend increasing the maximum number of flows per vRouter from the default value of 500,000 to 2 million flows to increase performance and scaling. We also recommend allocating 4 threads for flow processing.

These configuration parameters are specified in the **site.yml** file. A configuration snippet:

```
overcloud:
  extra_config:
    ContrailVrouterModuleOptions: "vr_flow_entries=2000000"
  contrail:
    vrouter:
     contrail_settings:
```

```
      default:
        VROUTER_AGENT__FLOWS__thread_count: "4"
```

Adding Kernel-Mode Compute Nodes

A compute node operates in kernel mode when it is added into the *compute_nodes_kernel:* hierarchy in the **compute-nodes.yml** file.

A sample configuration snippet:

```
compute_nodes_kernel:
  - name: compute1
    leaf: '0'
    profile: hw0
  - name: compute2
    leaf: '1'
    profile: hw1
```

Complete Leaf/Profile Configuration Snippet

This section provides a full configuration network snippet from the **overcloud-nics.yml** file for a compute node.

Profiles for network configuration in the **overcloud-nics.yml** file are written in this format:

```
[role][leaf number][hardware profile tag]_network_config
```

where:

- role is one of the following options:
  - ComputeKernel - for vRouter in kernel mode
  - ComputeDpdk
  - ComputeSriov
  - CephStorage
- leaf number - number or name of the leaf device. For instance, ”0” or “leaf0”

- hardware profile tag - any name to define a profile tag. We strongly recommend using the **Hw*[number]*** format for your hardware profile tags.

Examples:

Compute node DPDK in leaf “0” with hardware profile “hw1”

```
ComputeDpdk0hw1_network_config
```

Compute SR-IOV in leaf “0” with hardware profile “hw1”

```
ComputeSriov0hw1_network_config
```

Compute kernel mode in leaf “0” with hardware profile “hw1”

```
ComputeKernel0hw1_network_network_config
```

The sample **compute-nodes.yml** file:

```
#[role][leaf number][hardware profile tag]_network_config
# e.g.
ComputeDpdk0hw1_network_config
# Provisioning interface definition
  - type: interface
     name: nic1             # br-eno1
     dns_servers:
       get_param: DnsServers
     use_dhcp: false
     mtu:
       get_param: ControlPlaneNetworkMtu
     addresses:
     - ip_netmask:
         list_join:
           - '/'
           - - get_param: ControlPlaneIp
             - get_param: ControlPlaneSubnetCidr
     routes:
     -
       ip_netmask: 169.254.169.254/32
       next_hop:
```

```
          get_param: EC2MetadataIp
      -
        default: True         #Default route via provisioning, e.g. to access Satellite
        next_hop:
          get_param: ControlPlaneDefaultRoute

# Management interface definition
 - type: interface
    name: nic2                # br-eno2
    mtu:
      get_param: ManagementNetworkMtu
    addresses:
     - ip_netmask:
         get_param: ManagementIpSubnet
     routes:
     -
       ip_netmask: 10.0.0.0/8  #Address pool of corporate network that has access to
                               #servers via management network
       next_hop: 192.168.0.1


# br-bond0 interface definition (for vRouter overlay)
   - type: linux_bond
     name: bond0                # br-bond0
     use_dhcp: false
     bonding_options: "mode=802.3ad xmit_hash_policy=layer3+4 lacp_rate=fast updelay=1000
miimon=100"
     members:
     - type: interface
       name: nic3
       mtu:
        get_param: Tenant0NetworkMtu
       primary: true
     - type: interface
       name: nic4
       mtu:
        get_param: Tenant0NetworkMtu
   - type: vlan
     vlan_id:
       get_param: Tenant0NetworkVlanID
     device: bond0
   - type: contrail_vrouter
     name: vhost0
```

```
      use_dhcp: false
      members:
        -
          type: interface
          name:
            str_replace:
              template: vlanVLANID
              params:
                VLANID: {get_param: Tenant0NetworkVlanID}
          use_dhcp: false
      addresses:
      - ip_netmask:
          get_param: Tenant0IpSubnet
      mtu:
        get_param: Tenant0NetworkMtu
      routes:
      -
        ip_netmask:
          get_param: TenantSupernet
        next_hop:
          get_param: Tenant0InterfaceDefaultRoute

# br-bond1 interface definition (for Storage and Internal API networks)
 - type: linux_bond
   name: bond1                  # br-bond1
   use_dhcp: false
   bonding_options: "mode=802.3ad xmit_hash_policy=layer3+4 lacp_rate=fast updelay=1000
miimon=100"
   members:
   - type: interface
     name: nic5
     primary: true
   - type: interface
     name: nic6
   - type: vlan
     device: bond1
     vlan_id:
       get_param: Storage0NetworkVlanID
     mtu:
       get_param: Storage0NetworkMtu
     addresses:
     - ip_netmask:
         get_param: Storage0IpSubnet
```

```
    routes:
    -
      ip_netmask:
        get_param: StorageSupernet
      next_hop:
        get_param: Storage0InterfaceDefaultRoute
  - type: vlan
    device: bond1
    vlan_id:
      get_param: InternalApi0NetworkVlanID
    mtu:
      get_param: InternalApi0NetworkMtu
    addresses:
    - ip_netmask:
        get_param: InternalApi0IpSubnet
    routes:
    -
      ip_netmask:
        get_param: InternalApiSupernet
      next_hop:
        get_param: InternalApi0InterfaceDefaultRoute
```

NOTE: MTU settings are inherited from network settings in the **site.yml** file. The name is in “heat” notation (camel case) with added MTU value at the end.

NOTE: Use validation tools to check numbered to named NICs allocation. See "Miscellaneous" on page 104.

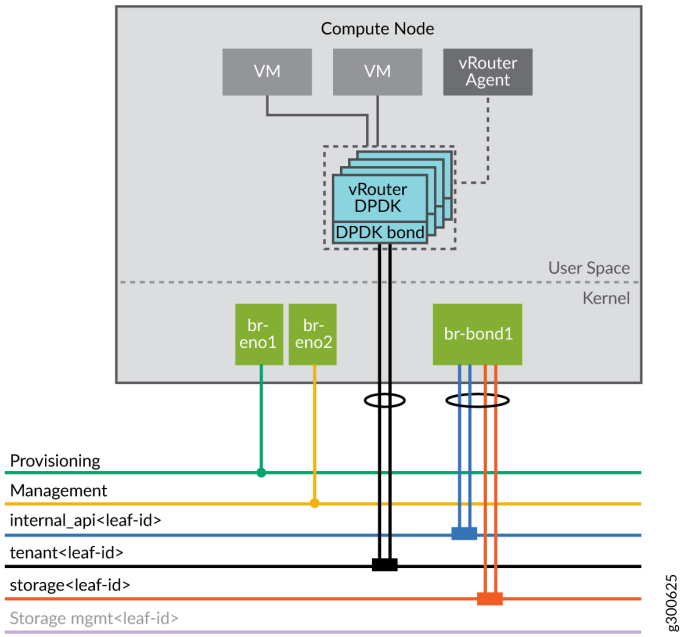
In the example above, the default route is configured using the management network, assuming it provides access for administrators to external resources. If the management network is not present, the default route should be via the provision network, for which SNAT to the Intranet is configured on the jumphost. In this scenario, the default: true statement should be in the *provision:* section of the **compute-nodes.yml** file. It is also possible to put the default route on the tenant network, which is useful if SNAT is used in tenant networks.

DPDK-mode vRouter Configuration

A vRouter in DPDK mode runs in a user space on the compute node. Network traffic is handled by a special DPDK dedicated interface or interfaces that handle VLANs and bonds. A specified number of cores is assigned to perform the vRouter forwarding function.

Figure 17 on page 71 illustrates a vRouter in DPDK mode.

Figure 17: DPDK vRouter



DPDK vRouters provide higher throughput than kernel vRouters. See Configuring the Data Plane Development Kit (DPDK) Integrated with Contrail vRouter for additional information on DPDK in Contrail networking.

DPDK Bond Interface Configuration

LACP bonding is under DPDK control when a vRouter is in DPDK mode. There is no Linux bond.

LACP bonding is defined in the **overcloud-nics.yml** file and configured using these options:

- bond_mode: 4 (IEEE 802.3ad)
- bond_policy: layer3+4

A sample **overcloud-nics.yml** file for the bond configuration:

```
ComputeDpdk0hw2_network_config:
  [...] #name, DHCP, MTU etc settings
  - type: contrail_vrouter_dpdk
    name: vhost0
    driver: vfio-pci
    bond_mode: 4
    bond_policy: layer3+4
```

The LACP rate with DPDK is taken during the LACP negotiation process between switches acting as LACP partners and applied to what is configured on the switches. Most Juniper switches are configured in fast LACP mode by default, and the vRouter applies that setting.

If you want to force LACP into fast LACP mode, set the *LACP_RATE:* field to 1 in the **site.yml** file:

```
contrail:
    vrouter:
      contrail_settings:
        default:
          LACP_RATE: 1
```

Performance Tuning for DPDK

To maximize throughput for a vRouter in DPDK mode, set the following parameters:

- Allocate 4 threads for flow processing in the vRouter agent.
- Allocate huge pages.
- Allocate CPUs for DPDK, Host OS, and Nova.
- Increase the flow table size to 2 million from the 500,000 default setting.
- Increase the buffer sizes to 2048 to reduce packet drops from microbursts.
- Double the vrouter memory pool size to 131072.
- Set the CPU scaling governor into performance mode.

Enabling CPU performance mode

CPU frequency scaling enables the operating system to scale the CPU frequency to save power. CPU frequencies can be scaled automatically depending on the system load, in response to ACPI events, or manually by user space programs. To run the CPU at the maximum frequency, set the scaling_governor parameter to performance.

In the BIOS all power saving options should be disabled, including power performance tuning, CPU P-State, CPU C3 Report and CPU C6 Report. Select **Performance** as the *CPU Power and Performance* policy.

The configuration can be defined in the **site.yml** file as an *extra_action* (post deployment action) parameter.

A sample configuration snippet from the **site.yml** file:

```
   post_deployment:
      shell:
        CPUperf: |
          if [[ "$role" =~ "ComputeDpdk" ]]; then
          sudo cat  > /usr/bin/after_reboot.sh <<'CPUPERF_EOF'
          #!/bin/bash
          for f in /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor ; do echo performance >
${f} ; done
          CPUPERF_EOF
          chmod +x /usr/bin/after_reboot.sh
          echo -e "$(sudo crontab -u root -l)\n#contrail cloud: setting performance
\n@reboot /usr/bin/after_reboot.sh" | sudo crontab -u root -
          fi
```

Configuring Flow Threads and Huge Pages

Memory that does not need to be used by the operating system should be segmented into huge pages to maximize efficiency. For instance, suppose a server with 256GB RAM needs 8GB for the OS (including the vRouter) and therefore has 248GB remaining for other functions. This server should allocate the remaining memory into a 1GB huge page to maximize memory usage.

Additionally, some 2MB huge pages should be configured to maximize memory usage for the vRouter.

The number of threads used for flow processing and huge page allocations are configured in the **site.yml** file.

A sample configuration snippet from the **site.yml** file:

```
overcloud:
  contrail:
    vrouter:
      contrail_settings:
       default:
        VROUTER_AGENT__FLOWS__thread_count: “4”
      dpdk:
        driver: vfio-pci #must be for Intel Fortville NICs
        huge_pages:
          two_mb: 9196
          one_gb: 224 # depends on how many VMs (amount of memory) is needed including 2G for
vRouter
```

CPU Allocation

CPU partitioning must be properly defined and configured to optimize performance. CPU partitioning issues can cause transient packet drops in moderate and high throughput environments.

Consider the following parameters when planning physical CPU core assignments:

- Numa topology
- Usage of hyperthreading (HT)
- Number of cores assigned for vrouter DPDK
- Number of cores allocated to VMs
- Number of cores left for system processes (include vRouter agent)

The following core mapping illustrates a NUMA topology for a CPU with 2 NUMAs. Each NUMA has 18 physical cores and supports hyper-threading.

```
NUMA node0:
  Physical cores:       0  2  4  6  8  10 12 14 16 18 20 22 24 26 28 30 32 34
  Hyperthreading cores: 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70
NUMA node1:
  Physical cores:       1  3  5  7  9  11 13 15 17 19 21 23 25 27 29 31 33 35
  Hyperthreading cores: 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71
```

Output key:

- Blue: allocated to DPDK
- Red: allocated to Nova for VMs
- Green: should not be allocated.
- Black: remainder used for operating system

> **NOTE**: Cores 0 and 1 with their corresponding HT siblings must not be allocated for either DPDK or Nova.

Six cores, without hyperthreading, are allocated for vRouter and are shown in blue in the example snippet. Our test results found that six cores is the maximum number that should be allocated in this environment, since multi-queue virtio does not handle larger core numbers effectively. The number of cores delivering maximum throughput may vary with different hardware specifications. Use of hyperthreading for DPDK has been shown to cause reduced throughput.

If more cores are used for DPDK, then the vr_mempool__sz parameter should be modified from the value suggested below, according to the formula:

*vr_mempool_sz = 2 * (dpdk_rxd_sz + dpdk_txd_sz) * (num_cores) * (num_ports)*

where:

- *num_cores* is the number of cores allocated for DPDK (including HT siblings, which are not recommended)
- *num_ports* is the number of physical ports in the DPDK bond interface
- *dpdk_rxd_sz* is the number of receive buffer descriptors
- *dpdk_txd_sz* is the number of transmit buffer descriptors

The default values for *dpdk_txd_sz* and *dpdk_rxd_sz* are set at 128 descriptors, but Intel recommends setting these values to 2048 descriptors to handle microbursts. We recommend not exceeding the 2048 descriptors setting as larger settings can cause unexpected latencies.

The cores that are used for virtual machine workloads are set using Nova CPU pinning, shown in red in the example snippet.

The operating system is allocated 4 physical cores with hyper-threading in this setup, as shown in black in the example snippet. The first cores on each NUMA must be allocated for the operating system. These operating system core reservations are not explicitly set by the user; the reservations are implicitly specified when the NUMAs are not allocated for DPDK or Nova.

The DPDK cores are all on NUMA0 in this architecture, which is where the corresponding NICs are located. It is good practice to place as many OS cores on NUMA1 as possible to create an environment where a higher proportion of VM workloads run on NUMA0 where network performance is maximized.

DPDK core allocations are defined in the **overcloud-nics.yml** file using hardware profiles.

A sample configuration snippet from the **overcloud-nics.yml** file:

```
ComputeDpdk0hw2_network_config:
  [...] #name, DHCP, MTU etc settings
  - type: contrail_vrouter_dpdk
    name: vhost0
   [...] #members, address, driver etc
  cpu_list: "2,4,6,8,10,12"
```

DPDK parameters—including the maximum number of flows and buffer sizes as well as parameters related to Nova pinning and general Nova functions—are defined in the *extra-config:* hierarchy in the **site.yml** file.

A sample configuration snippet from the **site.yml** file:

```
overcloud:
  extra_config:
    ComputeDpdkOptions: "--vr_flow_entries=2000000 --vr_mempool_sz 98304 --dpdk_txd_sz 2048 --
dpdk_rxd_sz 2048"
    ComputeDpdkParameters:
       TunedProfileName: "cpu-partitioning"
       IsolCpusList: "2,4,6-35,38,40,42-45,47-71"
    NovaVcpuPinSet: [‘7’,’9’,’11’,’13-35’,’43’,’45’,’47-71’]
    NovaSchedulerDefaultFilters:
      - RetryFilter
      - AvailabilityZoneFilter
      - RamFilter
      - DiskFilter
      - ComputeFilter
      - ComputeCapabilitiesFilter
      - ImagePropertiesFilter
      - ServerGroupAntiAffinityFilter
      - ServerGroupAffinityFilter
      - AggregateInstanceExtraSpecsFilter
      - NUMATopologyFilter
```

```
    NovaComputeExtraConfig:
      nova::cpu_allocation_ratio: 1.0
      nova::ram_allocation_ratio: 1.0
      nova::disk_allocation_ratio: 1.0
    ControllerExtraConfig:
      nova::config::nova_config:
          filter_scheduler/build_failure_weight_multiplier:
                value: 100.0
```

NOTE: The tuning profile *cpu-partitioning* causes the *cpu_affinity* value in the **tuned.conf** file to be set to the set of cores that are not in the *IsolCpusList*.

Adding DPDK Compute Nodes

Compute nodes that run in DPDK mode are identified in the compute_nodes_dpdk hierarchy of the **compute-nodes.yml** file.

A sample configuration snippet from the **compute-nodes.yml** file:

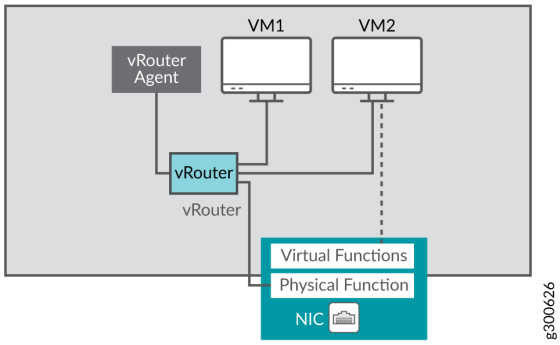
```
compute_nodes_dpdk:
  - name: ComputeDpdk1
    leaf: '0'
    profile: hw2
  - name: ComputeDpdk2
    leaf: '1'
    profile: hw3
```

SR-IOV Mode Compute Nodes

A compute node in SR-IOV mode provides direct access from the NIC to a VM. Because network traffic bypasses the vRouter in SR-IOV mode, no network policy or flow management is performed for traffic. See Configuring Single Root I/O Virtualization (SR-IOV) for additional information on SR-IOV in Contrail networking.

Figure 18 on page 78 illustrates the VM connections in computes nodes using SR-IOV mode.

Figure 18: Compute Node—VM and vRouter Connections in SR-IOV Mode



In SR-IOV terminology, a NIC is represented by Physical Function (PF) and the VMs, which are viewed as virtual versions of the NIC, are referred to as Virtual Functions (VFs). A VM with multiple interfaces can connect using overlay networks on some interfaces and SR-IOV on other interfaces.

VMs can use the overlay network or go directly to an underlay network through VFs using SR-IOV. This configuration is performed in the *compute_nodes_sriov:* hierarchy within the **compute-nodes.yml** file.

A configuration snippet from the **compute-nodes.yml** file.

```
compute_nodes_sriov:
  - name: ComputeSriov1
    leaf: '0'
    profile: hw4
  - name: ComputeSriov2
    leaf: '1'
    profile: hw5
```

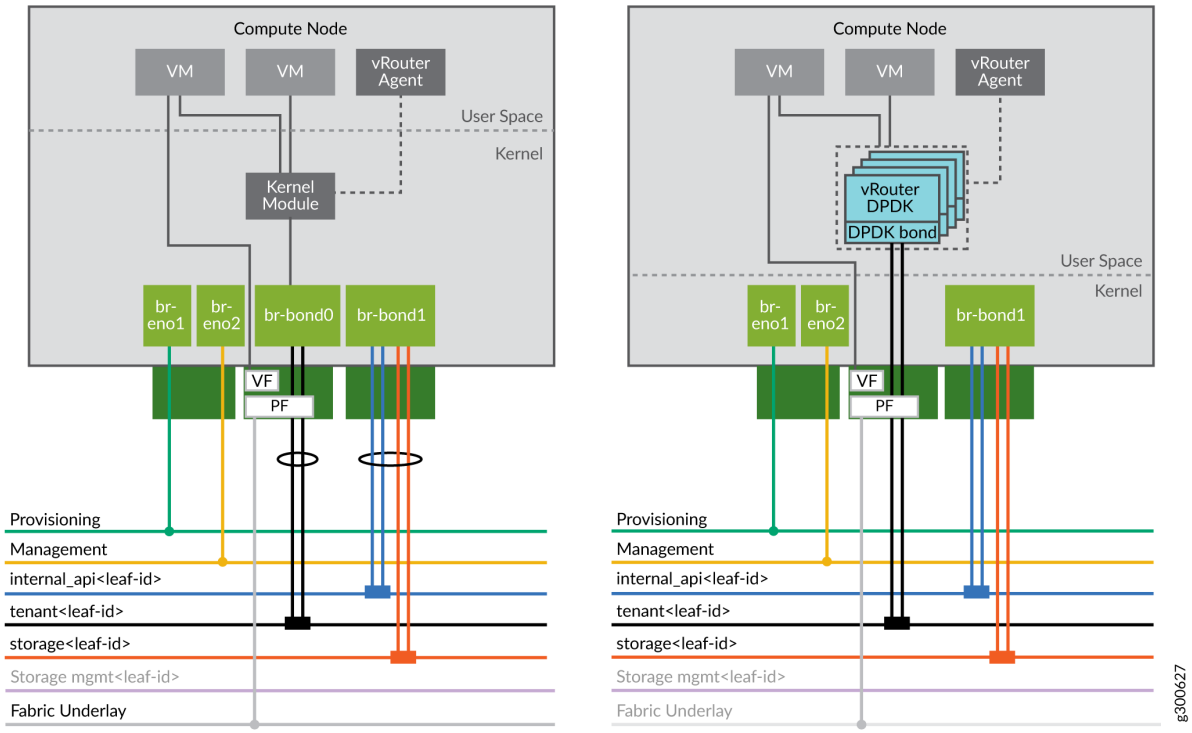
NOTE: SR-IOV must be enabled in BIOS.

The two types of vRouter deployments for SR-IOV:

- vRouter in Kernel mode on top of SR-IOV
- vRouter in DPDK mode on top of SR-IOV

Figure 19 on page 79 illustrates the vrouter in both modes.

Figure 19: SR-IOV—Kernel Mode and DPDK Mode



When SR-IOV is enabled on compute nodes, each vRouter is attached to a bond interface configured from the SR-IOV Physical Functions (PF). A VM interface that is in a network that has SR-IOV enabled is connected to a NIC Virtual Function (VF) and exposed to the fabric underlay network by Nova PCI passthrough.

You can select SR-IOV-mode kernel or dpdk mode in the **site.yml** file.

A configuration snippet from the **site.yml** file:

```
compute_hosts:
  sriov:
    enabled: true
    mode: kernel|dpdk
    #Sriov NumVFs separated by comma
    num_vf:
      - "ens7f1:7"
      - "ens7f2:7"
    #NovaPCIPassthrough settings
    pci_passthrough:
      - devname: "ens7f1"
```

```
        physical_network: "sriov1"
      - devname: "ens7f2"
        physical_network: "sriov2"
```

NOTE: Interface names must be used in the *sriov:* hierarchy in the **site.yml** file. See Interface naming conventions for server naming details.

The bond needs to be configured in the **overcloud-nics.yml** file.

A sample configuration snippet from the **overcloud-nic.yml** file:

```
[...]
   - type: linux_bond
     name: bond0                # br-bond0
     use_dhcp: false
     bonding_options: "mode=802.3ad xmit_hash_policy=layer3+4 lacp_rate=fast updelay=1000
miimon=100"
     members:
     - type: interface
       name: nic3   #ens7f1
       mtu:
        get_param: Tenant0NetworkMtu
       primary: true
     - type: interface
       name: nic4 #ens7f1
       mtu:
        get_param: Tenant0NetworkMtu
   - type: vlan
     vlan_id:
       get_param: Tenant0NetworkVlanID
     device: bond0
   - type: contrail_vrouter
     name: vhost0
     use_dhcp: false
     members:
       -
         type: interface
         name:
           str_replace:
```

```
            template: vlanVLANID
            params:
              VLANID: {get_param: Tenant0NetworkVlanID}
        use_dhcp: false
    addresses:
    - ip_netmask:
        get_param: Tenant0IpSubnet
    mtu:
      get_param: Tenant0NetworkMtu
    routes:
    -
      ip_netmask:
        get_param: TenantSupernet
      next_hop:
        get_param: Tenant0InterfaceDefaultRoute
```

In the *compute_hosts:* section of the **site.yml** file, SR-IOV must be enabled and the mode must be set to kernel or dpdk.

In the num_vf section of the **site.yml** file, set the number of virtual functions (VFs) that are allocated for NIC interfaces by the operating system. In this sample configuration snippet, 7 VFs are allocated. VF0 from ens7f1 and ens7ens2 are allocated for Nova PCI passthrough - provider networks with names: sriov1 and sriov2. The interface names are those seen when logging into a server using IPMI. The network names are arbitrary and only used inside the SR-IOV configuration.

The following snippet displays the **ip link** command output from a host when SR-IOV is enabled.

```
ip link
[...]
3: ens7f0: (BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP) mtu 9100 qdisc mq master bond0 state UP mode
DEFAULT group default qlen 1000
    link/ether 0c:c4:7a:b7:2d:6a brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off, query_rss off
    vf 1 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off, query_rss off
    vf 2 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off, query_rss off
    vf 3 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off, query_rss off
    vf 4 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off, query_rss off
    vf 5 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off, query_rss off
    vf 6 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off, query_rss off

5: ens7f1: (BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP) mtu 9100 qdisc mq master bond0 state UP mode
```

```
DEFAULT group default qlen 1000
    link/ether 0c:c4:7a:b7:2d:6a brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off, query_rss off
    vf 1 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off, query_rss off
    vf 2 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off, query_rss off
    vf 3 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off, query_rss off
    vf 4 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off, query_rss off
    vf 5 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off, query_rss off
    vf 6 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off, query_rss off
```

Storage Nodes (Ceph OSD)

The storage nodes run Ceph software in Contrail Cloud. For additional information on using Red Hat Ceph storage software, see Product Documentation for Red Hat Ceph Storage.

We recommend following these guidelines to optimize Ceph software performance in your Contrail Cloud deployment:

- Storage nodes must be separate from compute nodes in a Contrail Cloud environment. Hyperconverged nodes are not supported.
- Ceph requires a minimum of three storage nodes to operate.

If storage other than Ceph configured through Contrail Cloud is needed for your deployment, your deployment design might require additional support and engagement to ensure contractual support. Send an email to mailto:sre@juniper.net before moving forward with non-Ceph storage providers to ensure your deployment remains in compliance with your support contract.

Ceph Configuration

Ceph storage configuration is defined in the **site.yml** file.

Disk mapping for Ceph Storage nodes need to be defined on overcloud: disk_mapping: hierarchy for each storage node. The recommended disk settings are 4 OSD disks + 1 SSD journal disk. This configuration can be scaled appropriately; for instance, 8 OSD disks + 2 SSD journal disks for a larger environment. CPU, RAM, and storage traffic needs to be considered when using multiple 4 OSDs+1 journal bundles.

The mapping section of the **site.yml** file allows disks to be referenced by name rather than by device id.

A sample **site.yml** configuration file snippet to configure disk mappings:

```
overcloud:
  # Contains a list of label to disk mappings for roles
  disk_mapping:
    CephStorage0:
      # Mapping of labels to disk devices. The label is assigned to the disk
      # device so that the disk can be referenced by the alias in other
      # configurations. for example /dev/disk/by-alias/(label)
      # Each list element contains:       #   label: label to assign       #   hctl: disk device
path H:C:T:L. see lsscsi       - label: osd-0         hctl: '4:0:0:0'       - label:
osd-1         hctl: '5:0:0:0'       - label: osd-2         hctl: '6:0:0:0'       - label:
osd-3         hctl: '7:0:0:0'       - label: osd-4         hctl: '8:0:0:0'       - label:
journal-0         hctl: '9:0:0:0'
```

This sample **site.yml** configuration file snippet configures Ceph storage for a 4 OSD disks + 1 SSD journal bundle:

```
ceph:
  # Choice to enable Ceph storage in the overcloud.
  # "true" means that Ceph will be deployed as the backend for Cinder and Glance services.
  # "false" false means that Ceph will not be deployed.
  enabled: true
  # Ceph OSD disk configuration
  osd:
    # Update the Ceph crush map when OSDs are started
    crush_update_on_start: true
    # Size for OSD journal files.
    journal_size: 2048
    # Ceph OSD disk assignments. The named disks will be exclusively used by Ceph for
persistence.
    # For each disk, a "journal" can be configured.  journals can be shared between OSDs.
    disk:
      default:
       '/dev/sdb':
         journal: '/dev/disk/by-alias/journal-0'
       '/dev/sdc':
         journal: '/dev/disk/by-alias/journal-0'
       '/dev/sdd':
```

```
      journal: '/dev/disk/by-alias/journal-0'
    '/dev/sde':
      journal: '/dev/disk/by-alias/journal-0'
    '/dev/sdf':
      journal: '/dev/disk/by-alias/journal-0'
  CephStorageHw8:
    '/dev/sdb':
      journal: '/dev/disk/by-alias/journal-0'
    '/dev/sdc':
      journal: '/dev/disk/by-alias/journal-0'
    '/dev/sdd':
      journal: '/dev/disk/by-alias/journal-0'
    '/dev/sdf':
      journal: '/dev/disk/by-alias/journal-0'
  CephStorageHw7:
    '/dev/sdb':
      journal: '/dev/disk/by-alias/journal-0'
    '/dev/sdc':
      journal: '/dev/disk/by-alias/journal-0'
    '/dev/sdd':
      journal: '/dev/disk/by-alias/journal-0'
```

The disk layouts, notably, are per hardware profile.

Ceph service configuration hides the complexity of computing placement groups, by default. This configuration can be manually provided in the **site.yml** file if required, such as in cases where the environment has a small number of OSDs.

Storage Node Network Configuration

In this reference architecture, separate bonds are used on storage nodes for the Storage Mgmt (replication) and Storage (user access) networks.

Figure 20 on page 85 illustrates these connections.

Figure 20: Storage Node Networks

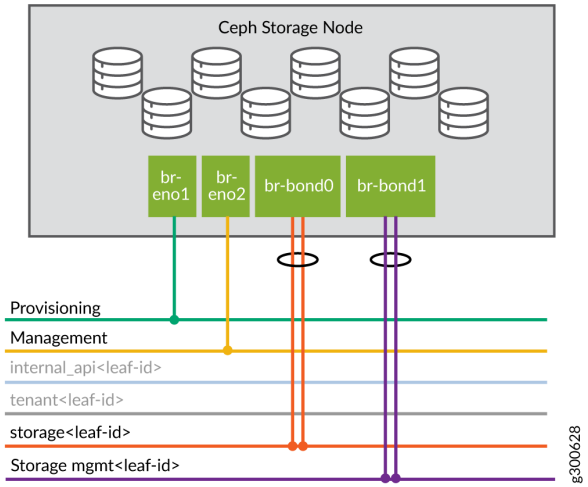


Figure 21 on page 85 illustrates how the physical and logical interfaces of a storage node connect to a management switch and leaf switches.

Figure 21: Storage Node—Physical and Logical Interfaces

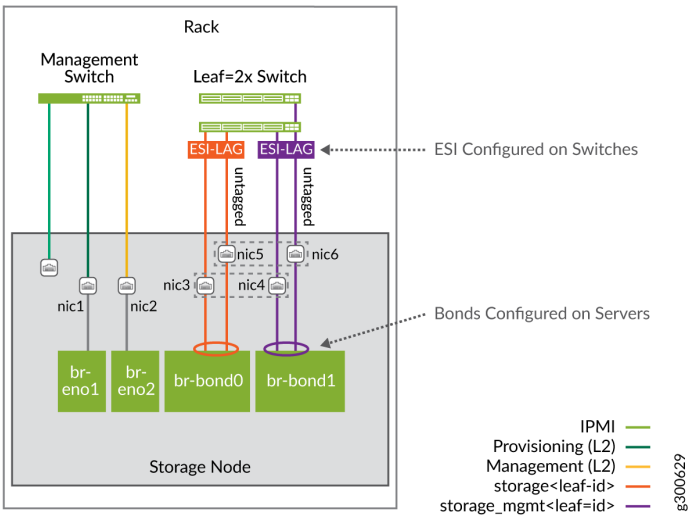


Figure 22 on page 86 illustrates how the physical interfaces of a storage host connect to the bridges configured on it.

Figure 22: Storage Node—Interfaces

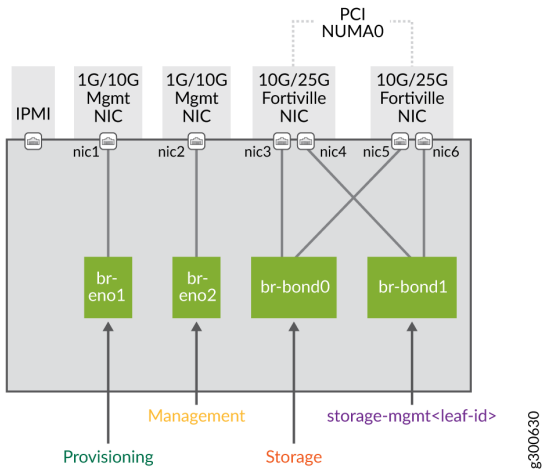


Table 19 on page 86 shows which files are used for configuring each network connection on a storage node.

The IP address of the management interface and addressing and configuration of bond interfaces on a storage node are configured according to the configuration files that are used by the Contrail Cloud provisioning system. The ports on each storage host are configured as follows.

Table 19: Storage Node Interface Naming Configuration Files

Port/NIC	Configuration
IPMI	Address is entered as an input to the **inventory.yml** file.
br-eno1	1 x 1G/10G NIC - untagged interface (e.g. built-in copper interface) with addresses generated automatically from the provisioning network pool.
br-eno2	1 x 1G/10G NIC - Optionally Management network, untagged interface with address defined in the *overcloud: network:* section of the **site.yml** file.
br-bond0	2 x 10G/25G NICs made of first ports from both NICs. Untagged interface with Storage network. Bonds physical allocation defined in the **overcloud-nics.yml** file. Addressing is set in the *overcloud: network:* section of the **site.yml** file.

br-bond1	2 x 10G/25G NICs made of second ports from both NICs. Untagged interface with Storage Mgmt network. Bonds physical allocation is defined in the **overcloud-nics.yml** file. Addressing is set in the *overcloud: network:* section of the **site.yml** file.

Linux bond configuration is defined per storage host in the **overcloud-nics.yml** file,

A sample configuration snippet from the **overcloud-nics.yml** file:

```
CephStorage0Hw1_network_config:   #0 means leaf number
 [...] #name, DHCP, MTU etc settings
 - type: ovs_bond
    name: bond0
    use_dhcp: false
    bonding_options: "mode=802.3ad xmit_hash_policy=layer3+4 lacp_rate=fast updelay=1000
miimon=100"
```

Where:

- NIC-0 - onboard copper NIC 1/10G
- NIC-1 and NIC-2 - 2 ports 10G/25G/40G Intel Fortville family NICs in PCI slot connected to NUMA0

RELATED DOCUMENTATION

Contrail Cloud Deployment Guide

Contrail Cloud TechLibrary page

Enhanced Resilience and Scale Variations

IN THIS SECTION

This section covers Contrail Cloud architecture options to support enhanced resilience and scale variations.

Enhanced Resilience and Scale Variations Overview

You can add controller nodes when increased resilience and scale are required in your Contrail Cloud environment. Two options for adding controller nodes are described in this section.

Five Controller Deployments

Enhanced resiliency and performance for Contrail Networking and Openstack Red Hat can be achieved by running controller VMs on additional servers and providing Layer 2 connectivity between the controller VMs.

In this use case, enhanced resiliency is achieved by running controller VMs on five servers with Layer 2 connectivity.

The setup includes:

- 5 Contrail Controllers
- 5 Openstack Controllers
- 3 Analytic nodes
- 3 Analytics DB nodes
- 3 Appformix nodes

The enhanced resiliency provided by this architecture ensures the Contrail Cloud environments remains operational in cases where two controller nodes fail.

Controller VM placement in this architecture is configured in the site.yml file.

A configuration snippet from the site.yml file illustrating this controller VM placement:

```
 control_hosts:
vm:
  control:   #openstack controller
    [...] CPU, Memory and disk size definitions
    hv:
      - controller-host1
```

```
      - controller-host2
      - controller-host3
      - controller-host4
      - controller-host5
  contrail-controller:
    [...] CPU, Memory and disk size definitions
    hv:
      - controller-host1
      - controller-host2
      - controller-host3
      - controller-host4
      - controller-host5
  contrail-analytics:
   [...] CPU, Memory and disk size definitions
    hv:
      - controller-host1
      - controller-host2
      - controller-host3
  contrail-analytics-database:
   [...] CPU, Memory and disk size definitions
    hv:
      - controller-host3
      - controller-host4
      - controller-host5
  appformix-controller:
   [...] CPU, Memory and disk size definitions
    hv:
      - controller-host1
      - controller-host4
      - controller-host5
```

NOTE: CPU, memory, and disk allocations should not change between a base three controller architecture and this five controller architecture.

Separated Controller Functions

In environments with large amounts of control plane or analytics traffic, you might want to increase scalability by splitting the OpenStack, Contrail Controller, and Contrail Analytics (with AppFormix) functions onto separate servers. This setup leads to base environments where 9 controller nodes are located in 3 racks.

Table 20 on page 90 lists the roles for each controller node in separated controller environments.

Table 20: Roles in a Separated Controller Functions Environment

Role	Network							
	Intranet	Provisioning	Internal API	External	Tenant	Storage	Storage Mgmt	Mgmt
Jumphost	✓	✓						
Undercloud	✓	✓						
Contrail Command	✓	✓						
Openstack Controller		✓	✓	✓	✓	✓	✓	Optional
Contrail Controller		✓	✓	Optional	✓		Optional	Optional
Contrail Analytics		✓	✓	Optional	✓		Optional	Optional
Contrail Analytics DB		✓	✓	Optional	✓		Optional	Optional
Contrail TSN		✓	✓	Optional	✓		Optional	Optional
AppFormix		✓	✓		✓		✓	Optional
Compute Node		✓	✓		✓	✓		Optional
Storage Node		✓				✓	✓	Optional

When groups of software components are allocated to different control hosts, the networking requirement of each control host is the union of the requirements of the software roles running on the

host. Specialized configurations can be configured using a different hardware profile for each combination of software components.

RELATED DOCUMENTATION

- Contrail Cloud Deployment Guide
- Contrail Cloud TechLibrary page

Reference Architecture Variations

IN THIS SECTION

This section provides a walkthrough of variations to the Contrail Cloud reference architecture.

Contrail Cloud Reference Architecture Variations

Contrail Cloud can be deployed with simpler server and network configurations in environments where performance and resilience requirements are more relaxed.

This section provides information about these architectural variations.

Supported Reference Architecture Variations Summary

Table 21 on page 91 lists supported variations for this reference architecture:

Table 21: Supported Reference Architecture Variations

Architectural Variation	Comment

Controller hosts in same rack	The controller networks don't need to be stretched between racks, but there is an increased risk of outage. No change to the configuration files is necessary for this variation.
Separate OpenStack and Contrail controller hosts	Use this variation in environments where you want to reduce the impact of a node failure.
Separate Controller and Analytics, AppFormix hosts	To increase performance in this variation: • Place the Openstack and Contrail Controllers on 3 hosts • Place the Contrail Analytics, Analytics DB, and Appformix controllers on additional 3 hosts
Use of NICs on NUMA 0 and NUMA 1	Intel architectures have become more flexible for cross-NUMA traffic from DPDK core to NIC, but these configurations have less throughput than the recommendation of both NICs and DPDK cores on NUMA 0.
Single bond interface on servers	Use in cases where there are no separate storage nodes, or where network traffic is light and there is a low risk of contention causing packet drops. Note that DPDK for the Tenant network cannot share an interface, so DPDK mode cannot be used in this configuration.
Single subnet across racks for Tenant, Storage, Storage Mgt, and Internal API traffic	Use in smaller environments where per-rack addressing is not a requirement
Use the same network for External, Management, and Intranet traffic	Network sharing can be used in non-production networks like labs and POCs, but this variation is not recommended in production environments.

Single Bond Interface in Variation Architectures

When servers with a single bond interface are to be used, each of the networks in the **overcloud-nics.yml** file is specified to be present on the same bond. The configuration is performed in the *controller_network_config* hierarchy and in each of the *compute[leaf][h/w profile]* and *storage[leaf][h/w profile]* hierarchies.

Leaf switch ports must be configured as follows for connections to the bond interfaces of each node type:

Table 22: Leaf Switch Port VLAN Summary

Connected node	VLANs
Controller	Tenant Storage Internal API External
Compute	Tenant Storage Internal API
Storage	Storage Storage Mgmt

The following diagrams illustrate connectivity for this architecture.

Figure 23: Control Host Networking—Single Bond Interface

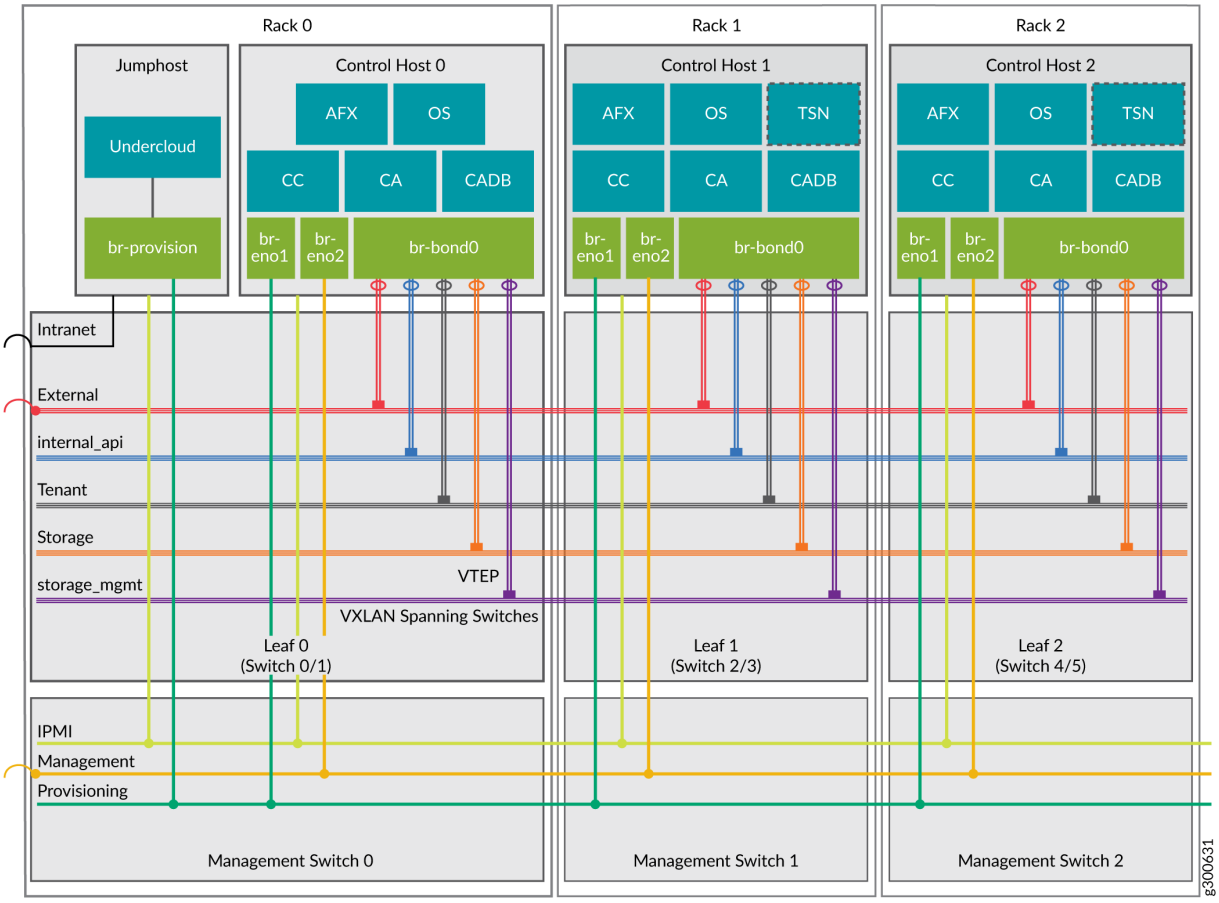
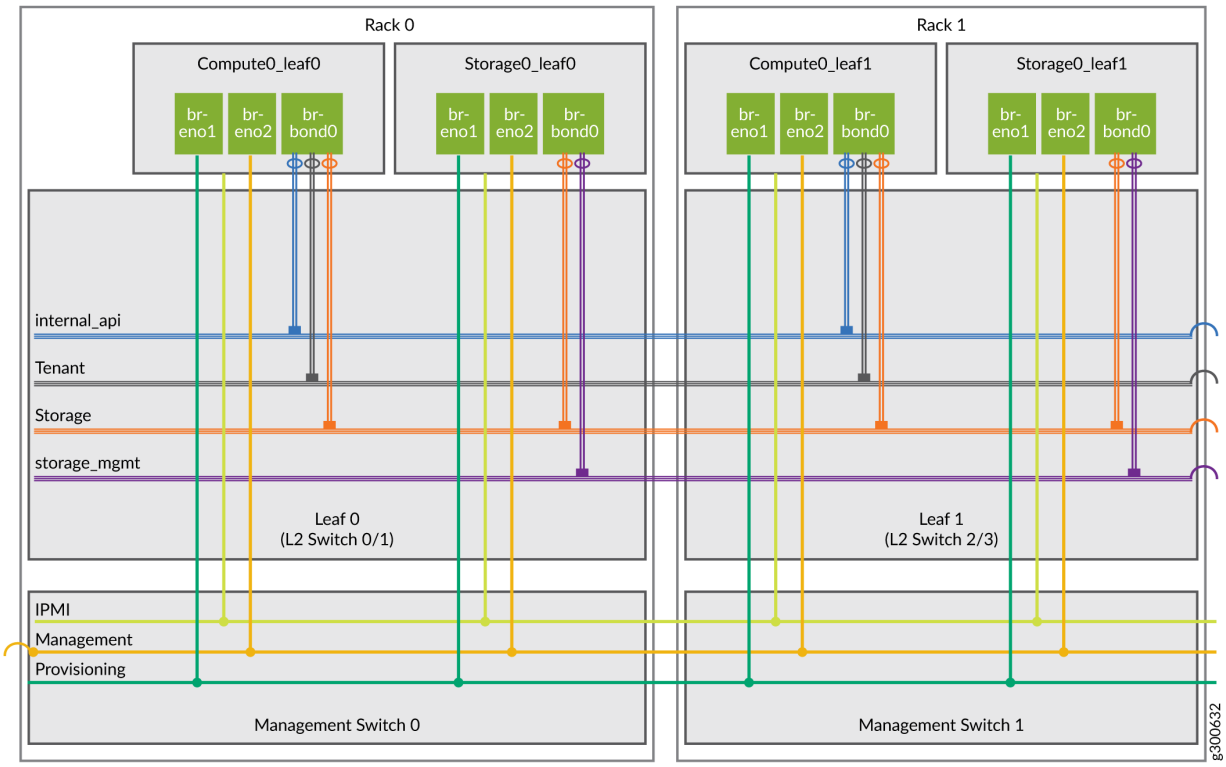


Figure 24: Compute and Storage Node Networking—Single Bond Interface



The following is a full configuration network snippet for a compute node in the **overcloud-nics.yml** file.

```
#[role][leaf number][hardware profile tag]_network_config
# e.g.
ComputeDpdk0Hw1_network_config
# Provisioning interface definition
  - type: interface
     name: nic1             # br-eno1
     dns_servers:
       get_param: DnsServers
     use_dhcp: false
     mtu:
       get_param: ControlPlaneNetworkMtu
     addresses:
     - ip_netmask:
         list_join:
           - '/'
           - - get_param: ControlPlaneIp
```

```
              - get_param: ControlPlaneSubnetCidr
     routes:
     -
       ip_netmask: 169.254.169.254/32
       next_hop:
         get_param: EC2MetadataIp
     -
       default: True         #Default route via provisioning, e.g. to access Satellite
       next_hop:
         get_param: ControlPlaneDefaultRoute

# Management interface definition
 - type: interface
    name: nic2               # br-eno2
    mtu:
      get_param: ManagementNetworkMtu
    addresses:
     - ip_netmask:
         get_param: ManagementIpSubnet
     routes:
     -
       ip_netmask: 10.0.0.0/8  #Address pool of corporate network that have access to
                               #servers via management network
       next_hop: 192.168.0.1



# br-bond0 interface definition (for all networks except provision and management
tagged)
   - type: linux_bond
     name: bond0               # br-bond0
     use_dhcp: false
     bonding_options: "mode=802.3ad xmit_hash_policy=layer3+4 lacp_rate=fast updelay=1000
miimon=100"
     members:
     - type: interface
       name: nic3
       mtu:
        get_param: Tenant0NetworkMtu
       primary: true
     - type: interface
       name: nic4
       mtu:
```

```
      get_param: Tenant0NetworkMtu
  - type: vlan
    vlan_id:
      get_param: Tenant0NetworkVlanID
    device: bond0
  - type: contrail_vrouter
    name: vhost0
    use_dhcp: false
    members:
      -
        type: interface
        name:
          str_replace:
            template: vlanVLANID
            params:
              VLANID: {get_param: Tenant0NetworkVlanID}
        use_dhcp: false
    addresses:
    - ip_netmask:
        get_param: Tenant0IpSubnet
    mtu:
      get_param: Tenant0NetworkMtu
    routes:
    -
      ip_netmask:
        get_param: TenantSupernet
      next_hop:
        get_param: Tenant0InterfaceDefaultRoute
  - type: vlan
    device: bond0
    vlan_id:
      get_param: Storage0NetworkVlanID
    mtu:
      get_param: Storage0NetworkMtu
    addresses:
    - ip_netmask:
        get_param: Storage0IpSubnet
    routes:
    -
      ip_netmask:
        get_param: StorageSupernet
      next_hop:
        get_param: Storage0InterfaceDefaultRoute
```

```
  - type: vlan
    device: bond0
    vlan_id:
      get_param: InternalApi0NetworkVlanID
    mtu:
      get_param: InternalApi0NetworkMtu
    addresses:
    - ip_netmask:
        get_param: InternalApi0IpSubnet
    routes:
    -
      ip_netmask:
        get_param: InternalApiSupernet
      next_hop:
        get_param: InternalApi0InterfaceDefaultRoute
```

Layer 2 Networks Between Racks

The same subnet address can be used across racks in small Contrail Cloud deployments.

Figure 25 on page 99 illustrates networking for control hosts using layer 2 to stretch across racks in a Contrail Cloud deployment. Figure 26 on page 100 illustrates networking for compute and storage nodes using layer 2 to stretch across racks.

Figure 25: Control Hosts—Networking with Stretched Layer 2 Networks

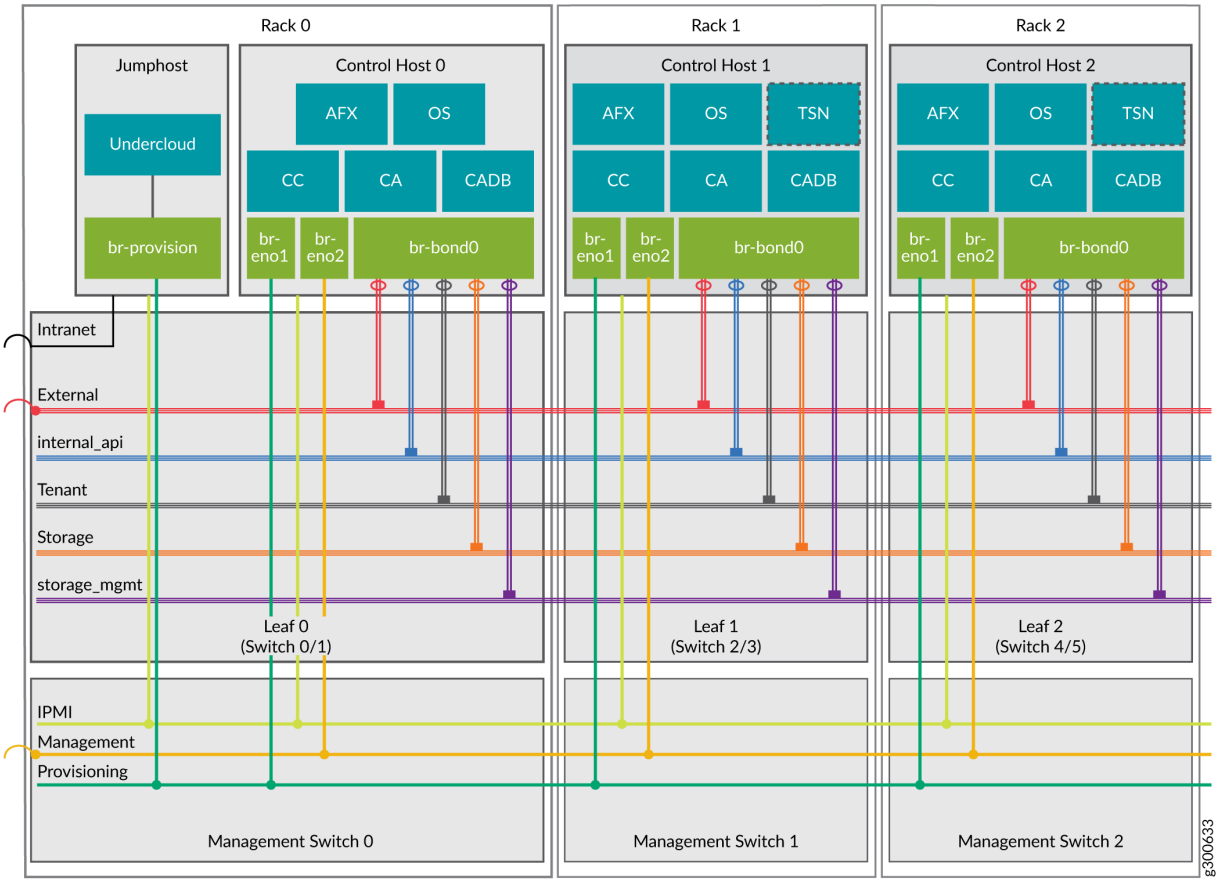
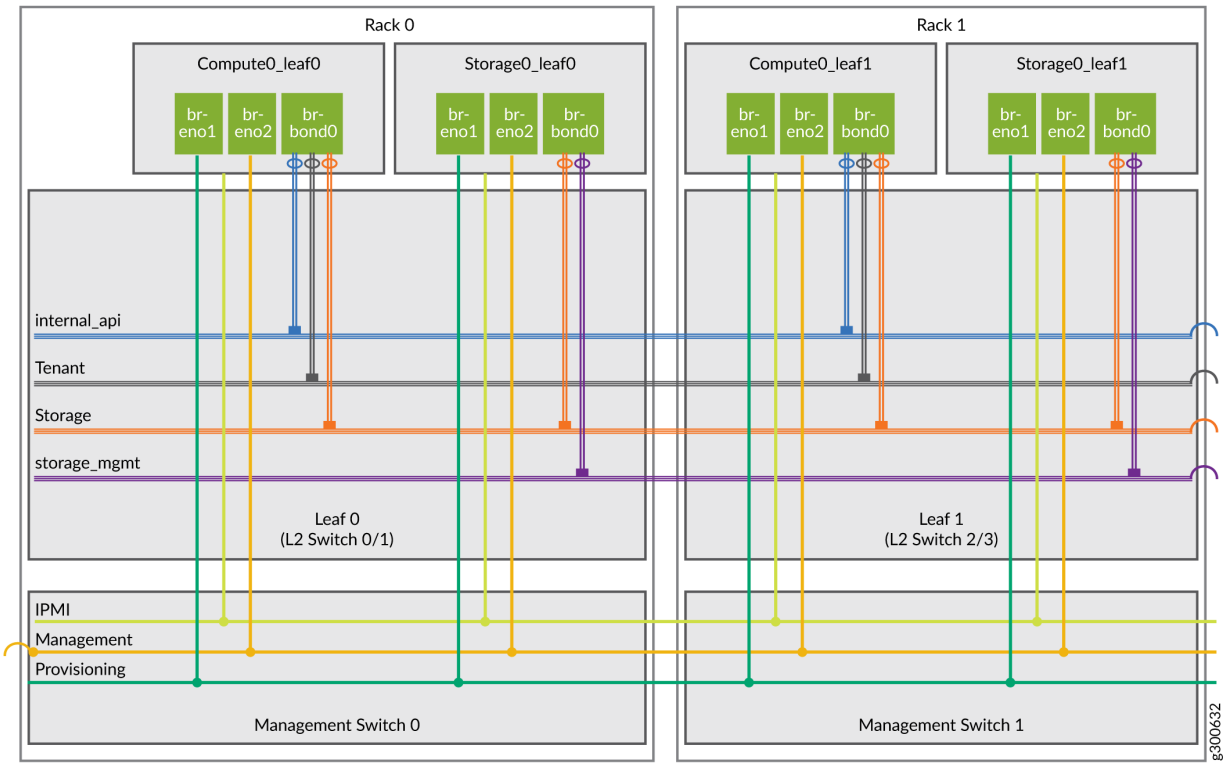


Figure 26: Compute and Storage Nodes—Networking with Stretched Layer 2 Networks



This Layer 2 addressing scheme is not recommended for environments with a large number of devices. Layer 2 stretch can be achieved using trunking between switches, or VXLAN if additional scalability is needed. Use of a separate management switch is optional.

For this type of deployment, a single network of each type is defined and no supernet is specified.

The following configuration snippet from the **site.yml** file illustrates this deployment.

```
network:
  external:
    cidr: "192.168.176.0/21"
    vlan: 305
    vip: "192.168.183.200"
    pool:
      start: "192.168.176.11"
      end: "192.168.183.199"
    mtu: 9100
  internal_api:
```

```
      vlan: 304
      cidr: "192.168.168.0/21"
      vip: "192.168.175.200"
      pool:
        start: "192.168.168.11"
        end: "192.168.175.199"
      mtu: 9100
    tenant:
      vlan: 301
      cidr: "192.168.144.0/21"
      pool:
        start: "192.168.144.11"
        end: "192.168.151.199"
      mtu: 9100
    storage:
      vlan: 303
      cidr: "192.168.160.0/21"
      pool:
        start: "192.168.160.11"
        end: "192.168.167.199"
      mtu: 9100
    storage_management:
      vlan: 302
      cidr: "192.168.152.0/21"
      pool:
        start: "192.168.152.11"
        end: "192.168.159.199"
      mtu: 9100
    management:
      vlan: "0"
      cidr: "192.168.72.0/21"
      default_route: "192.168.79.254"
      pool:
        start: "192.168.72.11"
        end: "192.168.79.199"
      mtu: 9100
```

Leaf switch ports are configured with VLANs in the same way as described in the previous section.

Single Controller Node

Small scale experimental and lab deployments of Contrail Cloud can have a single controller host. This is configured by having a single entry in the *control_host_nodes:* hierarchy within the **control-host-nodes.yml** file.

Proof of Concept Environments with High Availability

For proof-of-concept trial environments, the following is the minimum Contrail Cloud environment that can be configured with High Availability support:

- Jumphost
- 3 controls hosts
- 2 compute nodes, which can be used to validate routing and tunnels
- 3 storage nodes (optional)

Simplified networking can be implemented with the following components:

- IPMI connectivity from the jumphost
- Single network connection from each server to a switch
- Provision network configured as untagged on the interface
- Other networks configured with VLANs on the interface
- VLANs configured in switch to span between servers

This setup supports testing of most Contrail Networking features.

Single Controller Node

Small-scale Contrail Cloud environments—including experimental or controlled lab deployments—can be established with a jumphost, a single control host, and one or more compute nodes.

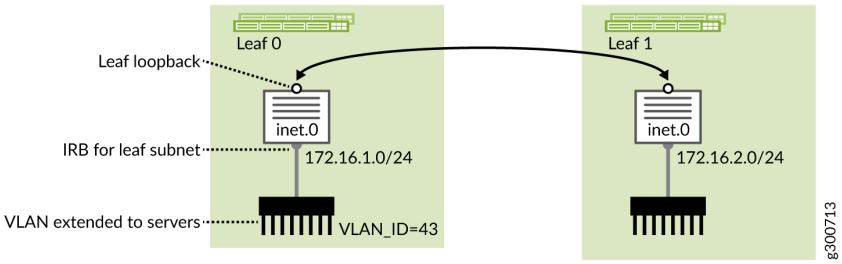
To configure this type of small-scale environment, include a single entry in the *control_host_nodes:* hierarchy in the **control-host-nodes.yml** file.

Underlay Routing Between Leaf Switches

You can configure routing between leaf switches in the underlay fabric network to simplify leaf switch configuration.

Figure 27 on page 103 illustrates leaf switch routing in the underlay network.

Figure 27: Underlay Routing Between Leaf Switches in the Fabric



The IRB interfaces for the leaf device subnets are configured but are not placed in VRF instances. Traffic, therefore, is routed using routes in the inet.0 global routing table on each switch. A route to each IRB interface is advertised between the leaf switches using iBGP.

Supported Variations Requiring Additional Approval

The following variations can be supported in production environments, but the variations must be explicitly approved by Juniper Networks to receive full customer support. Email mailto:sre@juniper.net or contact your Juniper representative before deploying these variations to ensure your Contrail Cloud environment remains in compliance with your support agreement.

Engagement with the Juniper Networks professional services team is typically required to deploy these variations.

Variations that Require Approval Overview

The following variations can be supported in production environments. Email mailto:sre@juniper.net or contact your Juniper Networks representative before deploying these variations to ensure your Contrail Cloud environment remains in compliance with your support agreement.

Table 23 on page 103 lists these variations.

Table 23: Architecture Variations that require Approval

Variation	Explanation
Use of VLANs instead of EVPN VXLAN, including the use of MC-LAG for server connectivity	Use in labs, POCs and smaller production environments where VLAN configuration on switches is manageable and the limitations of STP are not impactful.

Collapsed spine/gateway	Configuring SDN gateway function in spine switches is possible providing the spine supports the required functionality and scale (number of externally connected VRFs)
Single leaf switch per rack	For truly cloud-native applications which are resilient to infrastructure failures
Non-IP CLOS connectivity	No management switches - for lab environments only
Single controller node	Use for labs and training for feature testing. Not supported for production environments

NOTE: Contrail Cloud 13 releases do not support all-in-one deployments where a single node supports both controller and compute functions. Storage nodes also need to be separate devices.

The following sections provide information on how the configuration of Contrail Cloud can be modified to support these architectural variations.

RELATED DOCUMENTATION

- Contrail Cloud Deployment Guide
- Contrail Cloud TechLibrary page

Miscellaneous

IN THIS SECTION

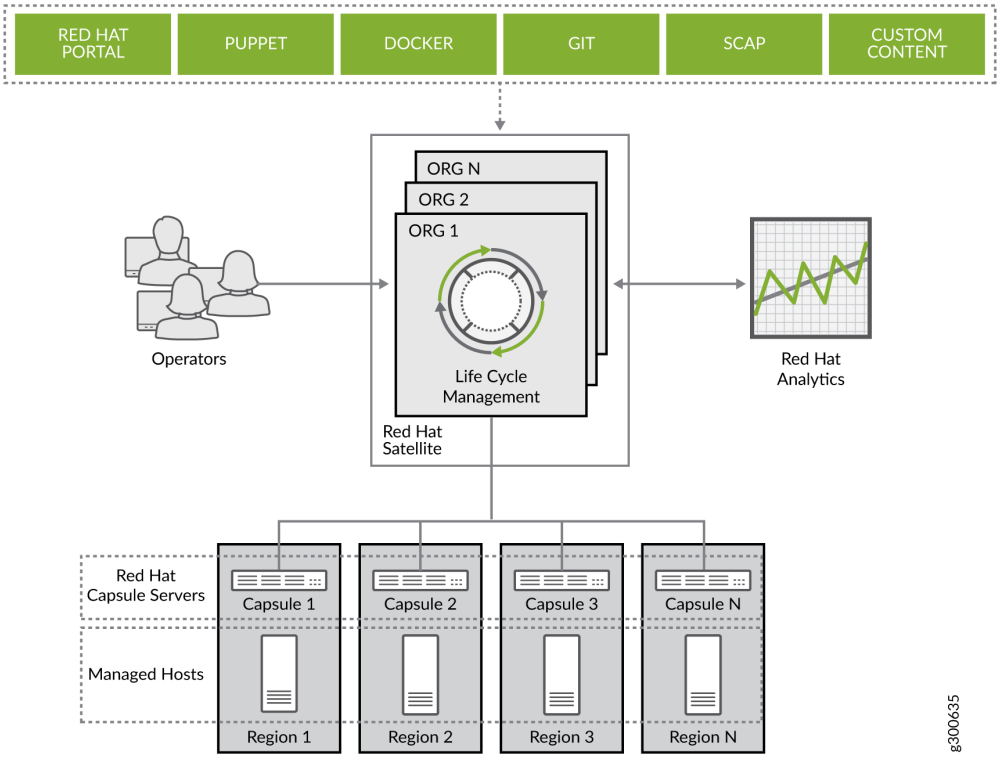
This section of the guide covers options available in Contrail Cloud.

Capsule Server Configuration

Capsule servers mirror content from a Satellite Server to establish content sources in various geographical locations. This enables host systems to pull content and configuration from the capsule servers in their location and not from the central Satellite Server. For additional information on capsule servers, see What Satellite Server and Capsule Server do from the Red Hat Satellite Installation Guide.

Figure 28 on page 105 illustrates a capsule server topology.

Figure 28: Capsule Server Topology



The Capsule VMs in this topology must have access to the Internet. Internet access is required to connect with Juniper Satellite servers for Contrail Cloud and to connect with RedHat subscription managers servers. For a detailed capsule server installation procedure, see Installing Capsule Server from the Red Hat Satellite Installation Guide.

The procedure for installing capsule server in Contrail Cloud:

1. Create a capsule VM using Red Hat Enterprise Linux 7.6.

2. Install the capsule keys for Red Hat Enterprise Linux from the Contrail Cloud Satellite device. The capsule keys are used to encrypt communication between the Contrail Cloud satellite and the capsule. The keys also allow Red Hat Enterprise Linux to register the operating system and are required to enable Red Hat Subscription Manager.

```
yum install ntp
ntpdate ntp_server_address
systemctl start ntpd
systemctl enable ntpd
rpm -Uvh https://contrail-cloud-satellite.juniper.net/pub/katello-ca-consumer-
latest.noarch.rpm
service goferd start
systemctl start goferd
```

3. Register the Contrail Cloud satellite device with the Red Hat Openstack subscription manager:

```
subscription-manager register --activationkey=[satellite_key] --org=[contrail] -force
```

4. From the Red Hat Subscription Manager, get the ID of the pool with RedHat Satellite and capsule server.

```
subscription-manager repos --disable "*"
subscription-manager release --unset
yum clean all
POOL=$(subscription-manager list --available --matches 'Red Hat Satellite' --pool-only | tail
-1)
subscription-manager attach --pool=$POOL
```

5. In Red Hat Subscription Manager, enable repositories.

```
subscription-manager repos --enable=rhel-7-server-rpms --enable=rhel-server-rhscl-7-rpms --
enable=rhel-7-server-satellite-6.2-rpms --enable=rhel-7-server-satellite-6.2-puppet4-rpms
```

6. In Red Hat Subscription Manager, refresh the subscription manager.

```
subscription-manager refresh
yum -y update
```

7. In Red Hat Subscription Manager, install the capsule server.

```
yum -y install satellite-capsule
```

8. Register your Capsule Server to the Contrail Cloud Satellite Server:

   - Create the certificates archive on the Contrail Cloud Satellite Server. This task requires access to the Contrail Cloud server and often has to be performed by a site's Contrail Cloud administrator.

     The certificates are delivered as a tar file.

     A configuration snippet of the command that is executed by the Capsule Server administrator.

```
capsule-certs-generate \
--capsule-fqdn "mycapsule.example.com" \
--certs-tar "~/mycapsule.example.com-certs.tar"
```

     NOTE: The provided FQDN must be resolvable from the satellite server to the VM IP address with capsule server.

   - Run the capsule installation script with the provided certificates:

```
satellite-installer --scenario capsule \
--foreman-proxy-content-parent-fqdn "contrail-cloud-satellite.juniper.net" \
--foreman-proxy-register-in-foreman "true"\
--foreman-proxy-foreman-base-url "https://contrail-cloud-satellite.juniper.net" \
--foreman-proxy-trusted-hosts "contrail-cloud-satellite.juniper.net" \
--foreman-proxy-oauth-consumer-key "[foreman-proxy-oauth-consumer-key]" \
--foreman-proxy-oauth-consumer-secret "[foreman-proxy-oauth-consumer-secret]" \
--foreman-proxy-content-certs-tar "[local a path to a tar file with certificates]"\
--puppet-server-foreman-url "https://contrail-cloud-satellite.juniper.net
```

Deployment Validation Tools

The Contrail Cloud 13.1 software bundle includes scripts that assist with detecting connectivity and configuration issues.

This section provides information on these scripts.

Check Server Hardware Specifications Script

The inventory-assigne.sh script—which is stored in the /var/lib/contrail_cloud/introspection/ directory on the jumphost—can be used to generate a JSON file with full hardware specifications for each server in your Contrail Cloud environment. The script uses the Red Hat Ironic service to help generate the JSON file.

The node-configuration.py python script—which is stored in the /var/lib/contrail_cloud/scripts/ directory on the jumphost—can be used to simplify the process of gathering server hardware specification information from the generated JSON files.

Check Disk allocation, CPU architecture and memory

In the following example, the node-configuration.py script is run after the inventory-assigne.sh script to gather information about check disk allocations, CPU architecture, and memory information:

```
/var/lib/contrail_cloud/scripts/node-configuration.py show -i /var/lib/contrail_cloud/
introspection/5a6s1-node5.introspection
INFO: SYS 5a6s1-node5 manufacturer:Supermicro
            cpu:Intel(R) Xeon(R) CPU E5-2630L v4 @ 1.80GHz core:20 core_ht:40
            memory:94GB
            boot_mode:bios
            Memory per numa:
            NUMA0: 47GB
            NUMA1: 48GB
            CPU per numa:
            NUMA0: CPU: 0 TH: [0, 20], CPU: 1 TH: [1, 21], CPU: 2 TH: [2, 22], CPU: 3 TH: [3,
23], CPU: 4 TH: [4, 24], CPU: 8 TH: [5, 25], CPU: 9 TH: [6, 26], CPU: 10 TH: [7, 27], CPU: 11
TH: [8, 28], CPU: 12 TH: [9, 29]
            NUMA1: CPU: 0 TH: [10, 30], CPU: 1 TH: [11, 31], CPU: 2 TH: [12, 32], CPU: 3 TH:
[13, 33], CPU: 4 TH: [14, 34], CPU: 8 TH: [15, 35], CPU: 9 TH: [16, 36], CPU: 10 TH: [17, 37],
CPU: 11 TH: [18, 38], CPU: 12 TH: [19, 39]
[...] # NICs informations
INFO: Disk /dev/sda root_disk:True, size:931GB
```

```
              serial:S4704LHK, model:ST1000NX0313
              by_path:/dev/disk/by-path/pci-0000:00:1f.2-ata-1.0, hctl:4:0:0:0
INFO: Disk /dev/sdb root_disk:False, size:931GB
              serial:S470BVD4, model:ST1000NX0313
              by_path:/dev/disk/by-path/pci-0000:00:1f.2-ata-2.0, hctl:5:0:0:0
INFO: Disk /dev/sdc root_disk:False, size:931GB
              serial:S470BTJZ, model:ST1000NX0313
              by_path:/dev/disk/by-path/pci-0000:00:1f.2-ata-3.0, hctl:6:0:0:0
INFO: Disk /dev/sdd root_disk:False, size:931GB
              serial:S470BVGP, model:ST1000NX0313
              by_path:/dev/disk/by-path/pci-0000:00:1f.2-ata-4.0, hctl:7:0:0:0
INFO: Disk /dev/sde root_disk:False, size:931GB
              serial:S470AQ3P, model:ST1000NX0313
              by_path:/dev/disk/by-path/pci-0000:00:1f.2-ata-5.0, hctl:8:0:0:0
INFO: Disk /dev/sdf root_disk:False, size:223GB
              serial:PHWA605401H8240AGN, model:INTEL SSDSC2BB24
              by_path:/dev/disk/by-path/pci-0000:00:1f.2-ata-6.0, hctl:9:0:0:0
```

Check NUMA and numbers of NICs

In this example, you check the NUMA and NIC numbers after the inventory-assigne.sh script has been.

```
 openstack baremetal introspection data save 5a6s1-node5 |   jq .| grep -i nic -A 100 | grep -e
name -e numa_node
        "name": "ens7f1",
        "numa_node": 0
        "name": "ens7f0",
        "numa_node": 0
        "name": "eno2",
        "numa_node": 0
        "name": "eno1",
        "numa_node": 0
```

Additional information related to NICs and NUMAs can be gathered by running the node-configuration.py script from the jumphost:

```
/var/lib/contrail_cloud/scripts/node-configuration.py show -i /var/lib/contrail_cloud/
introspection/5a6s1-node5.introspection
```

```
[...] #CPU information
INFO: NIC eno1 (nic1) NUMA:0 MAC:0c:c4:7a:81:a5:92, has_carrier:True PXE:True
              tagged_vlans:[300, 301, 302, 303, 304], native_vlan:300
              link_aggregation:False, link_aggregation_id:0
              port_desription:ge-1/0/0.0, switch_name:5a6-ex1
INFO: NIC eno2 (nic2) NUMA:0 MAC:0c:c4:7a:81:a5:93, has_carrier:True PXE:False
              tagged_vlans:[301, 302, 303, 304, 1008], native_vlan:1008
              link_aggregation:False, link_aggregation_id:0
              port_desription:ge-0/0/0.0, switch_name:5a6-ex1
INFO: NIC ens7f0 (nic3) NUMA:0 MAC:0c:c4:7a:b7:26:7c, has_carrier:True PXE:False
              tagged_vlans:[305, 306, 307, 308, 309], native_vlan:309
              link_aggregation:True, link_aggregation_id:736
              port_desription:xe-0/0/0, switch_name:5a6-qfx-1
INFO: NIC ens7f1 (nic4) NUMA:0 MAC:0c:c4:7a:b7:26:7d, has_carrier:True PXE:False
              tagged_vlans:[305, 306, 307, 308, 309], native_vlan:309
              link_aggregation:True, link_aggregation_id:736
              port_desription:xe-1/0/0, switch_name:5a6-qfx-1
INFO: NIC ens7f2 (nic5) NUMA:0 MAC:0c:c4:7a:b7:26:8a, has_carrier:True PXE:False
              tagged_vlans:[305, 306, 307, 308, 309], native_vlan:309
              link_aggregation:True, link_aggregation_id:737
port_desription:xe-0/0/1, switch_name:5a6-qfx-1
INFO: NIC ens7f3 (nic6) NUMA:0 MAC:0c:c4:7a:b7:26:8b, has_carrier:True PXE:False
              tagged_vlans:[305, 306, 307, 308, 309], native_vlan:309
              link_aggregation:True, link_aggregation_id:737
              port_desription:xe-1/0/1, switch_name:5a6-qfx-1
[...] #disk information
```

Pre-Deployment Check

The Contrail Cloud software bundle includes a script to check a server deployment before installing the Red Hat OpenStack platform. We suggest running the script before deploying the OpenStack cluster. The process is outlined in the Contrail Cloud Deployment Guide.

The validation-node.sh script—which is stored in the /var/lib/contrail_cloud/scripts directory on the jumphost—should be run before the openstack-deploy.sh script. The validation-node.sh script validates the configuration YAML files for the following components:

- Network for Openstack Controllers, Contrail Controller, Contrail Analytics and Analytics DB.
- Networking for controller hosts and compute hosts.
- Disk validation (comparing actual available disks with configured disks).

The following configuration snippet displays the process when the validation-node.sh script is run.

```
./validate-node.sh

[...]


Friday 29 March 2019  10:20:10 +0100 (0:00:06.661)       0:03:02.041 **********
===============================================================================
Reprocess introspection data ------------------------------------------- 48.56s
validate network for contrail-analytics -------------------------------- 6.85s
validate network for control ------------------------------------------- 6.80s
validate network for contrail-analytics-database ----------------------- 6.66s
validate network for contrail-controller ------------------------------- 6.59s
validate network for baremetal ----------------------------------------- 6.16s
Gathering Facts -------------------------------------------------------- 4.89s
validate network for computes ------------------------------------------ 4.67s
Get all nodes for introspection reprocess ------------------------------ 2.75s
ironic-node : Save data to file ---------------------------------------- 2.57s
ironic-node : Save data to file ---------------------------------------- 2.55s
ironic-node : Save data to file ---------------------------------------- 2.50s
ironic-node : Save data to file ---------------------------------------- 2.47s
ironic-node : Save data to file ---------------------------------------- 2.45s
ironic-node : Save data to file ---------------------------------------- 2.45s
ironic-node : Save data to file ---------------------------------------- 2.44s
ironic-node : Save data to file ---------------------------------------- 2.44s
ironic-node : Save data to file ---------------------------------------- 2.41s
ironic-node : Save data to file ---------------------------------------- 2.40s
ironic-node : Save data to file ---------------------------------------- 2.36s
Playbook run took 0 days, 0 hours, 3 minutes, 2 seconds
```

NOTE: The script verifies configuration parameters but cannot guarantee the success of the deployment.

Post-Deployment Check

The Contrail Cloud software bundle includes a set of Tempest test packages to check the health of the Contrail Cloud environment.

The Tempest test packages are launched using the overcloud-validation.sh script. The script runs the following Tempest test packages:

- openstack-tempest
- python2-cinder-tests-tempest
- python2-horizon-tests-tempest
- python2-keystone-tests-tempest
- python2-neutron-tests-tempest
- Contrail tempest checks including following scenarios: contrail, lbaasv2, pagination, sorting, security-group, ipam, port-security, binding, provider, agent, quotas, route-table, standard-attr-description, external-net, policy, router, allowed-address-pairs, extra_dhcp_opt, project-id, extra_lbaas_opts

Fabric Design

This reference architecture does not cover fabric design.

To configure an EVPN-VXLAN fabric that can be used by the nodes in a Contrail Cloud to transport Layer 2 and Layer 3 traffic, see Data Center Fabric Architecture Guide.

RELATED DOCUMENTATION

- Contrail Cloud Deployment Guide
- Contrail Cloud TechLibrary page