

Contrail™

---

# Contrail Feature Guide

Published  
2023-11-02

RELEASE  
4.1

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

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. in the United States and other countries. All other trademarks, service marks, registered marks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

*Contrail™ Contrail Feature Guide*

4.1

Copyright © 2023 Juniper Networks, Inc. All rights reserved.

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

1

## Overview

**Understanding Contrail Controller | 2**

Contrail Overview | 2

Contrail Description | 3

Contrail Installation Overview | 4

2

## Installing and Upgrading Contrail

**Supported Platforms and Server Requirements | 8**

Supported Platforms Contrail 4.1 | 8

Server Requirements | 12

Contrail Node Roles, Processes, and Ports | 13

**Installing Contrail and Provisioning Roles | 22**

Introduction to Containerized Contrail Modules | 22

Downloading Installation Software | 26

Installing the Operating System and Contrail Packages | 26

Installing Containerized Contrail Clusters Using Server Manager | 28

Installing Containerized Contrail Using Server Manager Lite (SM-Lite) | 32

Supporting Multiple Interfaces on Servers and Nodes | 35

Configuring the Control Node with BGP | 39

Adding a New Node to an Existing Containerized Contrail Cluster | 44

Using `contrailctl` to Configure Services Within Containers | 47

Contrail Global Controller | 50

Role and Resource-Based Access Control | 52

**Installation and Configuration Scenarios | 63**

Setting Up and Using a Simple Virtual Gateway with Contrail 4.0 | 63

|  |            |
|--|------------|
| Introduction to the Simple Gateway   | 64         |
| How the Simple Gateway Works   | 64         |
| Setup Without Simple Gateway   | 64         |
| Setup With a Simple Gateway  | 65         |
| Simple Gateway Configuration Features  | 66         |
| Packet Flows with the Simple Gateway   | 67         |
| Packet Flow Process From the Virtual Network to the Public Network                 | 68         |
| Packet Flow Process From the Public Network to the Virtual Network                 | 68         |
| Methods for Configuring the Simple Gateway   | 69         |
| Using the vRouter Configuration File to Configure the Simple Gateway               | 69         |
| Using Thrift Messages to Dynamically Configure the Simple Gateway                  | 69         |
| How to Dynamically Create a Virtual Gateway  | 70         |
| How to Dynamically Delete a Virtual Gateway  | 71         |
| Using Devstack to Configure the Simple Gateway                                     | 72         |
| Common Issues with Simple Gateway Configuration                                    | 72         |
| <br>   |            |
| Simple Underlay Connectivity without Gateway                                       | 73         |
| <br>   |            |
| Configuring MD5 Authentication for BGP Sessions                                    | 76         |
| <br>   |            |
| Configuring the Data Plane Development Kit (DPDK) Integrated with Contrail vRouter | 78         |
| <br>   |            |
| Configuring Single Root I/O Virtualization (SR-IOV)                                | 81         |
| <br>   |            |
| Provisioning DPDK SRIOV with Server Manager  | 86         |
| <br>   |            |
| Configuring Virtual Networks for Hub-and-Spoke Topology                            | 89         |
| Route Targets for Virtual Networks in Hub-and-Spoke Topology                       | 90         |
| Example: Configuring Hub-and-Spoke Virtual Networks                                | 90         |
| Troubleshooting Hub-and-Spoke Topology   | 91         |
| <br>   |            |
| Configuring Transport Layer Security-Based XMPP in Contrail                        | 96         |
| <br>   |            |
| Configuring Graceful Restart and Long-lived Graceful Restart                       | 99         |
| <br>   |            |
| <b>Using Contrail with Kubernetes  </b>  | <b>105</b> |
| Contrail Integration with Kubernetes   | 105        |
| <br>   |            |
| Installing and Provisioning Containerized Contrail Controller for Kubernetes       | 112        |
| <br>   |            |
| Viewing Configuration for CNI for Kubernetes                                       | 123        |
| View Pod Name and IP Address   | 124        |



- Verify Reachability of Pods | **124**
- Verify If Isolated Namespace-Pods Are Not Reachable | **124**
- Verify If Non-Isolated Namespace-Pods Are Reachable | **125**
- Verify If a Namespace is Isolated | **126**

#### Provisioning Contrail CNI for Kubernetes | **126**

- Requirements | **127**
- Overview | **130**
- Configuration | **130**
- Troubleshooting | **132**

#### Using Kubernetes Helm to Provision Contrail | **134**

- Requirements | **134**
- Overview | **137**
- Configuration | **137**
- Troubleshooting | **139**

### Using VMware vCenter with Containerized Contrail, Release 4.0.1 and Greater | **142**

#### Installing and Provisioning VMware vCenter with Containerized Contrail | **142**

- Overview: Integrating Contrail 4.0.1 and Greater with vCenter Server | **143**
- Different Modes of vCenter Integration with Contrail | **143**
- vCenter-Only Mode | **143**
- vCenter-as-Compute Mode | **144**
- Preparing the Installation Environment | **145**
- Installation for vCenter-Only Mode | **145**
- Installing the vCenter-Only Components | **146**
- Installation of vCenter-as-Compute Mode | **147**
- Installing the vCenter-as-Compute Components | **148**
- Verification | **148**
- Adding Hosts or Nodes | **148**
- Adding an ESXi Host to an Existing vCenter Cluster | **148**
- Adding a vCenter Cluster to vCenter-as-Compute | **148**

#### Underlay Network Configuration for Containerized ContrailVM | **149**

- Standard Switch Setup | **149**
- Distributed Switch Setup | **151**
- PCI Pass-Through Setup | **152**

|   |            |
|---|------------|
| SR-IOV Setup  | 155        |
| Sample JSON Configuration Files for vCenter with Containerized Contrail 4.0.1 and Greater | 159        |
| Sample Image JSON for vCenter-Only Mode   | 160        |
| Sample Cluster JSON for vCenter-Only Mode   | 160        |
| Sample Server JSON for vCenter-Only Mode  | 162        |
| Sample JSON for vCenter-only with High Availability                                       | 164        |
| Sample Image JSON for vCenter-as-Compute Mode   | 169        |
| Sample Cluster JSON for vCenter-as-Compute Mode   | 170        |
| Sample Server JSON for vCenter-as-Compute Mode  | 171        |
| Using the Contrail and VMWare vCenter User Interfaces to Manage the Network               | 175        |
| <b>Using Contrail with Red Hat  </b>  | <b>196</b> |
| Deploying Contrail with Red Hat OpenStack Platform Director 10                            | 196        |
| Installing Red Hat OpenShift Container Platform with Contrail Networking                  | 247        |
| Launch Instances (Azure, AWS, or Baremetal)   | 247        |
| Host Registration   | 248        |
| Install Base Packages   | 248        |
| Install OpenShift with Contrail Networking  | 250        |
| Installing a Contrail System on an Existing OpenShift Setup                               | 252        |
| Upgrade Procedure for RHOSP-based Contrail 3.2.x to Contrail 4.1                          | 254        |
| Upgrade Procedure for RHOSP-based Contrail 4.1.1 to Contrail 4.1.2                        | 268        |
| Upgrade Procedure for RHOSP-based Contrail 4.1.2 to Contrail 4.1.3                        | 279        |
| Upgrade Procedure for RHOSP-based Contrail 4.1.3 to Contrail 4.1.4                        | 292        |
| Restoring Contrail Nodes in a RHOSP-based Environment                                     | 308        |
| Prerequisites   | 308        |
| Verify the Controller Node Status and Rebuild the Node                                    | 308        |
| Finish Rebuilding One or Two Contrail Controller Nodes                                    | 310        |
| Finish the Rebuilding of all Contrail Controller Nodes                                    | 311        |
| Rebuilding Contrail Analytics And Analytics Database Nodes                                | 312        |
| Finish Rebuilding the Analytics Nodes   | 314        |
| <b>Using Server Manager to Automate Provisioning  </b>                                    | <b>316</b> |
| Installing Server Manager   | 316        |

|   |     |
|---|-----|
| Using Server Manager to Automate Provisioning               | 323 |
| Overview of Server Manager                                  | 323 |
| Server Manager Requirements and Assumptions                 | 324 |
| Server Manager Component Interactions                       | 325 |
| Configuring Server Manager                                  | 326 |
| Configuring the Cobbler DHCP Template                       | 328 |
| User-Defined Tags for Server Manager                        | 329 |
| Server Manager Client Configuration File                    | 329 |
| Restart Services  | 330 |
| Accessing Server Manager                                    | 330 |
| Communicating with the Server Manager Client                | 331 |
| Server Manager Commands for Configuring Servers             | 332 |
| Server Manager Commands Common Options                      | 332 |
| Add New Servers or Update Existing Servers                  | 333 |
| Delete Servers  | 334 |
| Display Server Configuration                                | 335 |
| Server Manager Commands for Managing Clusters               | 336 |
| Server Manager Commands for Managing Tags                   | 338 |
| Server Manager Commands for Managing Images                 | 340 |
| Server Manager Operational Commands for Managing Servers    | 344 |
| Reimaging Server(s)   | 344 |
| Provisioning and Configuring Roles on Servers               | 346 |
| Restarting Server(s)  | 347 |
| Show Status of Server(s)                                    | 348 |
| Show Status of Provision                                    | 349 |
| Server Manager REST API Calls                               | 349 |
| REST APIs for Server Manager Configuration Database Entries | 350 |
| API: Add a Server   | 350 |
| API: Delete Servers   | 350 |
| API: Retrieve Server Configuration                          | 351 |
| API: Add an Image   | 351 |
| API: Upload an Image  | 352 |
| API: Get Image Information                                  | 352 |
| API: Delete an Image  | 352 |
| API: Add or Modify a Cluster                                | 353 |
| API: Delete a Cluster                                       | 353 |

- API: Get Cluster Configuration | 353
- API: Get All Server Manager Configurations | 354
- API: Reimage Servers | 354
- API: Provision Servers | 354
- API: Restart Servers | 355

Example: Reimaging and Provisioning a Server | 355

Using the Server Manager Web User Interface | 357

- Log In to Server Manager | 357
- Create a Cluster for Server Manager | 358
- Edit a Cluster through Edit JSON | 369
- Working with Servers in the Server Manager User Interface | 369
- Add a Server | 370
- Edit Tags for Servers | 373
- Using the Edit Config Option for Multiple Servers | 373
- Edit a Server through Server Manager, Edit JSON | 374
- Filter Servers by Tag | 375
- Viewing Server Details | 375
- Configuring Images and Packages | 378
- Add New Image or Package | 379
- Selecting Server Manager Actions for Clusters | 379
- Reimage a Cluster | 380
- Provision a Cluster | 380

Installing and Using Server Manager Lite | 381

**Extending Contrail to Physical Routers, Bare Metal Servers, Switches, and Interfaces | 384**

Using ToR Switches and OVSDB to Extend the Contrail Cluster to Other Instances | 384

Configuring High Availability for the Contrail OVSDB ToR Agent | 397

Using Device Manager to Manage Physical Routers | 404

SR-IOV VF as the Physical Interface of vRouter | 436

Using Gateway Mode to Support Remote Instances | 438

REST APIs for Extending the Contrail Cluster to Physical Routers, and Physical and Logical Interfaces | 440

**Installing and Using Contrail Storage | 447**

Installing and Using Contrail Storage | 447

Overview of the Contrail Storage Solution | 447

Basic Storage Functionality with Contrail | 448

Ceph Block and Object Storage Functionality | 448

Using the Contrail Storage User Interface | 449

Hardware Specifications | 450

Contrail Storage Provisioning | 450

**Upgrading Contrail Software | 453**

Upgrading Contrail 4.0 to 4.1 | 453

Upgrade Procedure for RHOSP-based Contrail 4.1.2 to Contrail 4.1.3 | 455

Upgrade Procedure for Ubuntu-based Contrail 4.1.3 to Contrail 4.1.4 Using Juju with Netronome SmartNIC | 468

Upgrade Procedure for RHOSP-based Contrail 4.1.3 to Contrail 4.1.4 | 477

Dynamic Kernel Module Support (DKMS) for vRouter | 492

Backup and Restore Contrail Configuration Database | 493

Backup config database | 494

Restore config database | 497

3

## Configuring Contrail

**Configuring Virtual Networks | 508**

Creating Projects in OpenStack for Configuring Tenants in Contrail | 508

Creating a Virtual Network with Juniper Networks Contrail | 510

Creating a Virtual Network with OpenStack Contrail | 514

Creating an Image for a Project in OpenStack Contrail | 516

Creating a Floating IP Address Pool | 520

Using Security Groups with Virtual Machines (Instances) | 522

Security Groups Overview | 522

Creating Security Groups and Adding Rules | 522

Security Policy Enhancements | 526

Support for IPv6 Networks in Contrail | 545

## Configuring EVPN and VXLAN | 549

- Configuring the VXLAN Identifier Mode | 551
- Configuring Forwarding | 553
- Configuring the VXLAN Identifier | 554
- Configuring Encapsulation Methods | 555

## Example of Deploying a Multi-Tier Web Application Using Contrail | 559

### Example: Deploying a Multi-Tier Web Application | 559

- Multi-Tier Web Application Overview | 559
- Example: Setting Up Virtual Networks for a Simple Tiered Web Application | 560
- Verifying the Multi-Tier Web Application | 563
- Sample Addressing Scheme for Simple Tiered Web Application | 563
- Sample Physical Topology for Simple Tiered Web Application | 564
- Sample Physical Topology Addressing | 565

### Sample Network Configuration for Devices for Simple Tiered Web Application | 567

## Configuring Services | 574

### Configuring DNS Servers | 574

- DNS Overview | 574
- Defining Multiple Virtual Domain Name Servers | 575
- IPAM and Virtual DNS | 576
- DNS Record Types | 576
- Configuring DNS Using the Interface | 577
- Configuring DNS Using Scripts | 585

### Distributed Service Resource Allocation with Containerized Contrail | 586

### Support for Multicast | 597

- Subnet Broadcast | 598
- All-Broadcast/Limited-Broadcast and Link-Local Multicast | 598
- Host Broadcast | 599

### Using Static Routes with Services | 600

- Static Routes for Service Instances | 600
- Configuring Static Routes on a Service Instance | 601
- Configuring Static Routes on Service Instance Interfaces | 602
- Configuring Static Routes as Host Routes | 603

Configuring Metadata Service | 604

## Configuring Service Chaining | 606

Service Chaining | 606

- Service Chaining Basics | 606

- Service Chaining Configuration Elements | 608

Service Chaining MX Series Configuration | 611

ECMP Load Balancing in the Service Chain | 613

Customized Hash Field Selection for ECMP Load Balancing | 614

Service Chain Version 2 with Port Tuple | 619

Using the Contrail Heat Template | 623

Service Chain Route Reorigination | 628

Service Instance Health Checks | 650

- Health Check Object | 650

- Bidirectional Forwarding and Detection Health Check over Virtual Machine Interfaces | 655

- Bidirectional Forwarding and Detection Health Check for BGPaaS | 655

- Health Check of Transparent Service Chain | 656

- Service Instance Fate Sharing | 656

## Examples: Configuring Service Chaining | 658

Example: Creating an In-Network Service Chain | 658

- Hardware and Software Requirements | 658

- Overview | 659

- Configuration | 659

Example: Creating an In-Network-NAT Service Chain | 672

- Hardware and Software Requirements | 672

- Overview | 672

- Configuration | 673

Example: Creating a Transparent Service Chain | 686

- Hardware and Software Requirements | 686

- Overview | 686

- Configuration | 686

## **Adding Physical Network Functions in Service Chains | 701**

Using Physical Network Functions in Contrail Service Chains | 701

PNF Service Chaining Objects | 701

Prerequisites and Assumptions | 702

Example: Adding a Physical Network Function Device to a Service Chain | 703

## **Configuring High Availability | 711**

Juniper OpenStack High Availability | 711

Introduction | 712

Contrail High Availability | 712

OpenStack High Availability | 712

Supported Platforms | 712

Juniper OpenStack High Availability Architecture | 713

Juniper OpenStack Objectives | 713

Limitations | 714

Solution Components | 714

Virtual IP with Load Balancing | 714

Failure Handling | 715

Deployment | 716

Minimum Hardware Requirement | 716

Compute | 716

Network | 716

Installation | 717

High Availability Support Options | 719

High Availability for Containerized Contrail | 723

## **QoS Support in Contrail | 726**

Quality of Service in Contrail | 726

Configuring Network QoS Parameters | 735

Overview | 735

QoS Configuration Examples | 735

Limitations | 737

BGP as a Service | 737



## Load Balancers | 743

Using Load Balancers in Contrail | 743

Support for OpenStack LBaaS Version 2.0 APIs | 758

Configuring Load Balancing as a Service in Contrail | 760

Overview: Load Balancing as a Service | 761

Contrail LBaaS Implementation | 762

Configuring LBaaS Using CLI | 763

## Optimizing Contrail | 766

Route Target Filtering | 766

Introduction | 766

Debugging and Troubleshooting Route Target Filtering | 767

RTF Limitations in Contrail 1.10 | 768

Source Network Address Translation (SNAT) | 769

Overview | 769

Neutron APIs for Routers | 770

Network Namespace | 770

Using the Web UI to Configure Routers with SNAT | 771

Multiqueue Virtio Interfaces in Virtual Machines | 772

vRouter Command Line Utilities | 774

Overview | 774

vif Command | 775

flow Command | 779

vrfstats Command | 781

rt Command | 782

dropstats Command | 783

mpls Command | 787

mirror Command | 789

vxlan Command | 791

nh Command | 793

## Monitoring and Troubleshooting Contrail

Configuring Traffic Mirroring to Monitor Network Traffic | 798

Configuring Traffic Analyzers and Packet Capture for Mirroring | 798

- Traffic Analyzer Images | 799
- Configuring Traffic Analyzers | 799
- Setting Up Traffic Mirroring Using Monitor > Debug > Packet Capture | 799
- Setting Up Traffic Mirroring Using Configure > Networking > Services | 804

Configuring Interface Monitoring and Mirroring | 811

Analyzer Service Virtual Machine | 812

Mirroring Enhancements | 816

Mapping VLAN Tags from a Physical NIC to a VMI (NIC-Assisted Mirroring) | 817

## **Understanding Contrail Analytics | 819**

Understanding Contrail Analytics | 819

Contrail Alerts | 820

Underlay Overlay Mapping in Contrail | 824

- Overview: Underlay Overlay Mapping using Contrail Analytics | 825
- Underlay Overlay Analytics Available in Contrail | 825
- Architecture and Data Collection | 826
- New Processes/Services for Underlay Overlay Mapping | 826
- External Interfaces Configuration for Underlay Overlay Mapping | 827
- Physical Topology | 827
- SNMP Configuration | 828
- Link Layer Discovery Protocol (LLDP) Configuration | 828
- IPFIX and sFlow Configuration | 828
- Sending pRouter Information to the SNMP Collector in Contrail | 831
- pRouter UVEs | 831
- Contrail User Interface for Underlay Overlay Analytics | 833
- Enabling Physical Topology on the Web UI | 834
- Viewing Topology to the Virtual Machine Level | 834
- Viewing the Traffic of any Link | 834
- Trace Flows | 835
- Search Flows and Map Flows | 836
- Overlay to Underlay Flow Map Schemas | 837
- Module Operations for Overlay Underlay Mapping | 840
- SNMP Collector Operation | 840
- Topology Module Operation | 842

IPFIX and sFlow Collector Operation | 843

Troubleshooting Underlay Overlay Mapping | 844

Script to add pRouter Objects | 844

## **Configuring Contrail Analytics | 847**

Analytics Scalability | 847

High Availability for Analytics | 849

Role-Based Access Control for Analytics | 849

System Log Receiver in Contrail Analytics | 851

Overview | 851

Redirecting System Logs to Contrail Collector | 851

Exporting Logs from Contrail Analytics | 851

Sending Flow Messages to the Contrail System Log | 852

More Efficient Flow Queries | 853

Ceilometer Support in a Contrail Cloud | 853

Overview | 854

Ceilometer Details | 854

Verification of Ceilometer Operation | 855

Contrail Ceilometer Plugin | 857

Ceilometer Installation and Provisioning | 860

User Configuration for Analytics Alarms and Log Statistics | 860

Configuring Alarms Based on User-Visible Entities Data | 861

Examples: Detecting Anomalies | 863

Configuring the User-Defined Log Statistic | 864

Implementing the User-Defined Log Statistic | 867

Alarms History | 870

Node Memory and CPU Information | 872

Role- and Resource-Based Access Control for the Contrail Analytics API | 873

Configuring Analytics as a Standalone Solution | 874

Configuring Secure Sandesh and Introspect for Contrail Analytics | 877

**Using Contrail Analytics to Monitor and Troubleshoot the Network | 880**

Monitoring the System | **880**

Debugging Processes Using the Contrail Introspect Feature | **884**

Monitor > Infrastructure > Dashboard | **889**

Monitor Dashboard | **890**

Monitor Individual Details from the Dashboard | **890**

Using Bubble Charts | **891**

Color-Coding of Bubble Charts | **892**

Monitor > Infrastructure > Control Nodes | **893**

Monitor Control Nodes Summary | **893**

Monitor Individual Control Node Details | **894**

Monitor Individual Control Node Console | **896**

Monitor Individual Control Node Peers | **899**

Monitor Individual Control Node Routes | **901**

Monitor > Infrastructure > Virtual Routers | **904**

Monitor vRouters Summary | **904**

Monitor Individual vRouters Tabs | **906**

Monitor Individual vRouter Details Tab | **906**

Monitor Individual vRouters Interfaces Tab | **908**

Monitor Individual vRouters Networks Tab | **910**

Monitor Individual vRouters ACL Tab | **911**

Monitor Individual vRouters Flows Tab | **913**

Monitor Individual vRouters Routes Tab | **914**

Monitor Individual vRouter Console Tab | **915**

Monitor > Infrastructure > Analytics Nodes | **918**

Monitor Analytics Nodes | **918**

Monitor Analytics Individual Node Details Tab | **920**

Monitor Analytics Individual Node Generators Tab | **921**

Monitor Analytics Individual Node QE Queries Tab | **922**

Monitor Analytics Individual Node Console Tab | **923**

Monitor > Infrastructure > Config Nodes | **926**

Monitor Config Nodes | **926**

Monitor Individual Config Node Details | **927**

Monitor Individual Config Node Console | **928**

**Monitor > Networking | 930**

- Monitor > Networking Menu Options | 930
- Monitor -> Networking -> Dashboard | 931
- Monitor > Networking > Projects | 933
- Monitor Projects Detail | 934
- Monitor > Networking > Networks | 937

**Query > Flows | 942**

- Query > Flows > Flow Series | 943
- Example: Query Flow Series | 946
- Query > Flow Records | 948
- Query > Flows > Query Queue | 951

**Query > Logs | 952**

- Query > Logs Menu Options | 953
- Query > Logs > System Logs | 953
- Sample Query for System Logs | 955
- Query > Logs > Object Logs | 957

**Understanding Flow Sampling | 959****Example: Debugging Connectivity Using Monitoring for Troubleshooting | 962**

- Using Monitoring to Debug Connectivity | 963

**Common Support Answers | 970****Debugging Ping Failures for Policy-Connected Networks | 970****Debugging BGP Peering and Route Exchange in Contrail | 978**

- Example Cluster | 978
- Verifying the BGP Routers | 978
- Verifying the Route Exchange | 982
- Debugging Route Exchange with Policies | 984
- Debugging Peering with an MX Series Router | 986
- Debugging a BGP Peer Down Error with Incorrect Family | 988
- Configuring MX Peering (iBGP) | 990
- Checking Route Exchange with an MX Series Peer | 992
- Checking the Route in the MX Series Router | 994

**Troubleshooting the Floating IP Address Pool in Contrail | 996**

- Example Cluster | **997**
- Example | **998**
- Example: MX80 Configuration for the Gateway | **999**
- Ping the Floating IP from the Public Network | **1002**
- Troubleshooting Details | **1003**
- Get the UUID of the Virtual Network | **1003**
- View the Floating IP Object in the API Server | **1004**
- View floating-ips in floating-ip-pools in the API Server | **1008**
- Check Floating IP Objects in the Virtual Machine Interface | **1011**
- View the BGP Peer Status on the Control Node | **1015**
- Querying Routes in the Public Virtual Network | **1016**
- Verification from the MX80 Gateway | **1018**
- Viewing the Compute Node Vnsw Agent | **1020**
- Advanced Troubleshooting | **1022**

Removing Stale Virtual Machines and Virtual Machine Interfaces | **1025**

- Problem Example | **1025**
- Show Virtual Machines | **1027**
- Show Virtual Machines Using Python API | **1028**
- Delete Methods | **1030**

Troubleshooting Link-Local Services in Contrail | **1030**

- Overview of Link-Local Services | **1030**
- Troubleshooting Procedure for Link-Local Services | **1031**
- Metadata Service | **1032**
- Troubleshooting Procedure for Link-Local Metadata Service | **1032**

## 5

### Contrail Commands and APIs

**Contrail Commands | 1036**

Getting Contrail Node Status | **1036**

- Overview | **1036**
- UVE for NodeStatus | **1037**
- Node Status Features | **1037**
- Using Introspect to Get Process Status | **1044**
- contrail-status script | **1046**

contrail-logs (Accessing Log File Messages) | **1048**

contrail-status (Viewing Node Status) | **1051**

contrail-version (Viewing Version Information) | **1053**

service (Managing Services) | **1056**

Backing Up Contrail Databases Using JSON Format | **1058**

## **Contrail Application Programming Interfaces (APIs) | 1066**

Contrail Analytics Application Programming Interfaces (APIs) and User-Visible Entities (UVEs) | **1066**

User-Visible Entities | **1067**

Common UVEs in Contrail | **1068**

Virtual Network UVE | **1068**

Virtual Machine UVE | **1069**

vRouter UVE | **1069**

UVEs for Contrail Nodes | **1070**

Wild Card Query of UVEs | **1070**

Filtering UVE Information | **1070**

Log and Flow Information APIs | **1080**

HTTP GET APIs | **1080**

HTTP POST API | **1081**

POST Data Format Example | **1081**

Query Types | **1083**

Examining Query Status | **1083**

Examining Query Chunks | **1084**

Example Queries for Log and Flow Data | **1084**

Working with Neutron | **1088**

Data Structure | **1088**

Network Sharing in Neutron | **1089**

Commands for Neutron Network Sharing | **1090**

Support for Neutron APIs | **1090**

Contrail Neutron Plugin | **1091**

DHCP Options | **1091**

Incompatibilities | **1092**

Support for Amazon VPC APIs on Contrail OpenStack | **1092**

Overview of Amazon Virtual Private Cloud | **1093**

Mapping Amazon VPC Features to OpenStack Contrail Features | **1093**

VPC and Subnets Example | **1094**

Euca2ools CLI for VPC and Subnets | **1095**

Security in VPC: Network ACLs Example | **1095**

Euca2ools CLI for Network ACLs | **1097**

Security in VPC: Security Groups Example | **1097**

Euca2ools CLI for Security Groups | **1098**

Elastic IPs in VPC | **1099**

Euca2ools CLI for Elastic IPs | **1099**

Euca2ools CLI for Route Tables | **1100**

Supported Next Hops | **1100**

Internet Gateway Next Hop Euca2ools CLI | **1101**

NAT Instance Next Hop Euca2ools CLI | **1101**

Example: Creating a NAT Instance with Euca2ools CLI | **1101**



# 1

PART

## Overview

---

[Understanding Contrail Controller | 2](#)

---

# Understanding Contrail Controller

## IN THIS CHAPTER

- [Contrail Overview | 2](#)
- [Contrail Description | 3](#)
- [Contrail Installation Overview | 4](#)

## Contrail Overview

Juniper Networks Contrail is an open, standards-based software solution that delivers network virtualization and service automation for federated cloud networks. It provides self-service provisioning, improves network troubleshooting and diagnostics, and enables service chaining for dynamic application environments across enterprise virtual private cloud (VPC), managed Infrastructure as a Service (IaaS), and Networks Functions Virtualization use cases.

Contrail simplifies the creation and management of virtual networks to enable policy-based automation, greatly reducing the need for physical and operational infrastructure typically required to support network management. In addition, it uses mature technologies to address key challenges of large-scale managed environments, including multitenancy, network segmentation, network access control, and IP service enablement. These challenges are particularly difficult in evolving dynamic application environments such as the Web, gaming, big data, cloud, and the like.

Contrail allows a tenant or a cloud service provider to abstract virtual networks at a higher layer to eliminate device-level configuration and easily control and manage policies for tenant virtual networks. A browser-based user interface enables users to define virtual network and network service policies, then configure and interconnect networks simply by attaching policies. Contrail also extends native IP capabilities to the hosts (compute nodes) in the data center to address the scale, resiliency, and service enablement challenges of traditional orchestration platforms.

Using Contrail, a tenant can define, manage, and control the connectivity, services, and security policies of the virtual network. The tenant or other users can use the self-service graphical user interface to easily create virtual network nodes, add and remove IP services (such as firewall, load balancing, DNS, and the like) to their virtual networks, then connect the networks using traffic policies that are simple to

create and apply. Once created, policies can be applied across multiple network nodes, changed, added, and deleted, all from a simple browser-based interface.

Contrail can be used with open cloud orchestration systems such as OpenStack. It can also interact with other systems and applications based on Operations Support System (OSS) and Business Support Systems (BSS), using northbound APIs. Contrail allows customers to build elastic architectures that leverage the benefits of cloud computing – agility, self-service, efficiency, and flexibility – while providing an interoperable, scale-out control plane for network services within and across network domains.

## RELATED DOCUMENTATION

| [Contrail Description](#)

## Contrail Description

### IN THIS SECTION

- [Contrail Major Components | 3](#)
- [Contrail Solution | 4](#)

## Contrail Major Components

The following are the major components of Contrail.

### Contrail Control Nodes

- Responsible for the routing control plane, configuration management, analytics, and the user interface.
- Provide APIs to integrate with an orchestration system or a custom user interface.
- Horizontally scalable, can run on multiple servers.

### Contrail Compute Nodes – XMPP Agent and vRouter

- Responsible for managing the data plane.

- Functionality can reside on a host OS.

## Contrail Solution

Contrail architecture takes advantage of the economics of cloud computing and simplifies the physical network (IP fabric) with a software virtual network overlay that delivers service orchestration, automation, and intercloud federation for public and hybrid clouds.

Similar to the native Layer 3 designs of web-scale players in the market and public cloud providers, the Contrail solution leverages IP as the abstraction between dynamic applications and networks, ensuring smooth migration from existing technologies, as well as support of emerging dynamic applications.

The Contrail solution is software running on x86 Linux servers, focused on enabling multitenancy for enterprise Information Technology as a Service (ITaaS). Multitenancy is enabled by the creation of multiple distinct Layer 3-enabled virtual networks with traffic isolation, routing between tenant groups, and network-based access control for each user group. To extend the IP network edge to the hosts and accommodate virtual machine workload mobility while simplifying and automating network (re)configuration, Contrail maintains a real-time state across dynamic virtual networks, exposes the network-as-a-service to cloud users, and enables deep network diagnostics and analytics down to the host.

In this paradigm, users of cloud-based services can take advantage of services and applications and assume that pooled, elastic resources are orchestrated, automated, and optimized across compute, storage, and network nodes in a converged architecture that is application-aware and independent of underlying hardware and software technologies.

### RELATED DOCUMENTATION

[Contrail Overview](#)

[Contrail Roles Overview](#)

## Contrail Installation Overview

### IN THIS SECTION

- [Installing Contrail on Different Operating Systems | 5](#)

Contrail is validated on several operating systems and orchestration systems. Installation procedures vary, depending on your environment. Additionally, API tools are available to customize your system.

This section provides links to the installation procedures for different validated environments.

## Installing Contrail on Different Operating Systems

To get anticipated results, be sure to check the validated supported operating system for your version of Contrail and make sure you have the correct kernel version:

*Supported Platforms Contrail 4.1*

or refer to the Release Notes for your version of Contrail.

You should start your validated installation by referring to the documentation section that corresponds to your operating environment, including:

### Juniper OpenStack Ubuntu Installation

Refer to the following topics when you are installing Juniper OpenStack Contrail on Ubuntu.

- *Introduction to Containerized Contrail Modules*
- *Downloading Installation Software*
- *Installing the Operating System and Contrail Packages*
- *Installing Containerized Contrail Clusters Using Server Manager*
- *Installing Containerized Contrail Using Server Manager Lite (SM-Lite)*

### Using VMware vCenter with Containerized Contrail, Release 4.0.1 and Greater

Refer to the following topics when you are installing containerized Contrail, Release 4.0.1 and greater, on VMware vCenter.

- [Installing and Provisioning VMware vCenter with Containerized Contrail](#)
- [Underlay Network Configuration for Containerized ContrailVM](#)
- [Sample JSON Configuration Files for vCenter with Containerized Contrail 4.0.1 and Greater](#)

### Using VMware vCenter with Contrail, through Release 4.0

Refer to the following topics when you are installing Contrail through Release 4.0 on VMware vCenter.

- [Installing and Provisioning VMware vCenter with Contrail](#)
- [Underlay Network Configuration for ContrailVM](#)
- [Sample Testbed.py Files for Contrail vCenter](#)

### **Using Red Hat with Contrail**

Refer to the following topics when you are installing Contrail with Red Hat.

- [Installing Red Hat OpenShift Container Platform with Contrail Networking](#)

### **Using Contrail with Kubernetes Automation Platform**

Refer to the following topics when you are installing containerized Contrail integrated with the Kubernetes automation platform.

- ["Contrail Integration with Kubernetes" on page 105](#)
- [Installing and Provisioning Containerized Contrail Controller for Kubernetes](#)
- ["Verifying Configuration for CNI for Kubernetes " on page 123](#)
- [Using Kubernetes Helm to Provision Contrail](#)

### **Using APIs with Contrail**

Additionally, Contrail can interact with other systems and applications using northbound APIs, enabling customization of your system. An index to current APIs is available in your installed version of Contrail at: <http://<your-server-IP>:8082/documentation/index.html>, or you can refer to:

[Juniper Contrail Configuration API Reference](#)

# 2

PART

## Installing and Upgrading Contrail

---

[Supported Platforms and Server Requirements](#) | 8

[Installing Contrail and Provisioning Roles](#) | 22

[Installation and Configuration Scenarios](#) | 63

[Using Contrail with Kubernetes](#) | 105

[Using VMware vCenter with Containerized Contrail, Release 4.0.1 and Greater](#) | 142

[Using Contrail with Red Hat](#) | 196

[Using Server Manager to Automate Provisioning](#) | 316

[Extending Contrail to Physical Routers, Bare Metal Servers, Switches, and Interfaces](#) | 384

[Installing and Using Contrail Storage](#) | 447

[Upgrading Contrail Software](#) | 453

---

# Supported Platforms and Server Requirements

## IN THIS CHAPTER

- Supported Platforms Contrail 4.1 | 8
- Server Requirements | 12
- Contrail Node Roles, Processes, and Ports | 13

## Supported Platforms Contrail 4.1

Table 1 on page 8 lists the operating system versions and the corresponding Linux or Ubuntu kernel versions supported by Contrail Release 4.1.

**Table 1: Supported Platforms**

| Contrail Release       | Orchestrator Release | Operating System and Kernel Versions   |
|------------------------|----------------------|--|
| Contrail Release 4.1.5 | OpenStack Newton     | <ul style="list-style-type: none"> <li>● RHEL7.5—Linux Kernel Version 3.10.0-862.14.4 (RHOSP 10.0)<br/>[Satellite content synced on Oct 29, 2018]</li> <li>● RHEL7.7—Linux Kernel Version 3.10.0-1062.12.1 (RHOSP 10.0.14)<br/>[Satellite content synced on May 20, 2020]</li> </ul> |
|                        | OpenStack Ocata      | <ul style="list-style-type: none"> <li>● Ubuntu 16.04.6 - Linux Kernel Version 4.15.0-112-generic</li> </ul>   |



**Table 1: Supported Platforms (Continued)**

| Contrail Release         | Orchestrator Release | Operating System and Kernel Versions   |
|--------------------------|----------------------|--|
| Contrail Release 4.1.4.1 | OpenStack Newton     | <ul style="list-style-type: none"> <li>• RHEL7.5—Linux Kernel Version 3.10.0-862.14.4 (RHOSP 10.0)</li> <li>• RHEL7.7—Linux Kernel Version 3.10.0-1062.9.1 (RHOSP 10.0.14)</li> </ul>  |
| Contrail Release 4.1.4   | OpenStack Ocata      | <ul style="list-style-type: none"> <li>• Ubuntu 16.04.2—Linux kernel version 4.4.0-165-generic</li> <li>• VMware vCenter 6.0, 6.5—Ubuntu 16.04.2 kernel version 4.4.0-62-generic</li> </ul>  |
|                          | OpenStack Newton     | <ul style="list-style-type: none"> <li>• RHEL7.7—Linux Kernel Version 3.10.0-1062.1.2 (RHOSP 10.0.12)</li> <li>• Ubuntu 16.04.2—Linux kernel version 4.4.0-165-generic</li> </ul>  |
|                          | OpenStack Mitaka     | <ul style="list-style-type: none"> <li>• Ubuntu 14.04.5—Linux kernel versions 3.13.0-171-generic</li> </ul>  |
| Contrail Release 4.1.3   | OpenStack Ocata      | <ul style="list-style-type: none"> <li>• RHEL 7.5—Linux kernel version 3.10.0-862.11.6 and Linux kernel version 3.10.0-957 (RHOSP11)</li> <li>• Ubuntu 16.04.2—Linux kernel version 4.4.0-116-generic</li> <li>• VMware vCenter 6.0, 6.5—Ubuntu 16.04.2 kernel version 4.4.0-62-generic</li> </ul> |
|                          | OpenStack Newton     | <ul style="list-style-type: none"> <li>• RHEL 7.5—Linux kernel version 3.10.0-862.11.6</li> <li>• RHEL 7.6—Linux kernel version 3.10.0-957 (RHOSP10)</li> <li>• Ubuntu 16.04.2—Linux kernel version 4.4.0-116-generic</li> </ul>   |

**Table 1: Supported Platforms (Continued)**

| Contrail Release       | Orchestrator Release | Operating System and Kernel Versions   |
|------------------------|----------------------|--|
|                        | OpenStack Mitaka     | <ul style="list-style-type: none"> <li>• Ubuntu 14.04.5—Linux kernel versions 3.13.0-142-generic and 4.4.0-116-generic</li> </ul>  |
| Contrail Release 4.1.2 | Kubernetes 1.7.5     | <ul style="list-style-type: none"> <li>• Ubuntu 16.04.2—Linux kernel version 4.4.0-116-generic</li> </ul>  |
|                        | OpenStack Ocata      | <ul style="list-style-type: none"> <li>• RHEL 7.5—Linux kernel version 3.10.0-862.11.6 and Linux kernel version 3.10.0-957 (RHOSP11)</li> <li>• Ubuntu 16.04.2—Linux kernel version 4.4.0-116-generic</li> <li>• VMware vCenter 6.0, 6.5—Ubuntu 16.04.2 kernel version 4.4.0-62-generic</li> </ul> |
|                        | OpenStack Newton     | <ul style="list-style-type: none"> <li>• RHEL 7.5—Linux kernel version 3.10.0-862.11.6</li> <li>• Ubuntu 16.04.2—Linux kernel version 4.4.0-116-generic</li> </ul>   |
|                        | OpenStack Mitaka     | <ul style="list-style-type: none"> <li>• Ubuntu 14.04.5—Linux kernel versions 3.13.0-142-generic and 4.4.0-116-generic</li> </ul>  |
| Contrail Release 4.1.1 | Kubernetes 1.7.5     | <ul style="list-style-type: none"> <li>• Ubuntu 16.04.2—Linux kernel version 4.4.0-116-generic</li> </ul>  |
|                        | Openshift 3.6        | <ul style="list-style-type: none"> <li>• RHEL 7.5—Linux kernel version 3.10.0-862.3.2</li> </ul>   |

**Table 1: Supported Platforms (Continued)**

| Conrail Release     | Orchestrator Release | Operating System and Kernel Versions  |
|---------------------|----------------------|---|
|                     | OpenStack Ocata      | <ul style="list-style-type: none"> <li>• RHEL 7.5—Linux kernel version 3.10.0-862.3.2 (RHOSP11)</li> <li>• Ubuntu 16.04.2—Linux kernel version 4.4.0-116-generic</li> <li>• VMware vCenter 6.0, 6.5—Ubuntu 16.04.2 kernel version 4.4.0-62-generic</li> </ul> |
|                     | OpenStack Newton     | <ul style="list-style-type: none"> <li>• RHEL 7.5—Linux kernel version 3.10.0-862.3.2 (RHOSP10)</li> <li>• Ubuntu 16.04.2—Linux kernel version 4.4.0-116-generic</li> </ul>   |
|                     | OpenStack Mitaka     | <ul style="list-style-type: none"> <li>• Ubuntu 14.04.5—Linux kernel versions 3.13.0-142-generic and 4.4.0-116-generic</li> </ul>   |
| Conrail Release 4.1 | Kubernetes 1.7.5     | <ul style="list-style-type: none"> <li>• Ubuntu 16.04.2—Linux kernel version 4.4.0-62-generic</li> </ul>  |
|                     | OpenShift 3.6        | <ul style="list-style-type: none"> <li>• RHEL 7.4—Linux kernel version 3.10.0-693</li> </ul>  |
|                     | OpenStack Ocata      | <ul style="list-style-type: none"> <li>• RHEL 7.4—Linux kernel version 3.10.0-693 (RHOSP11)</li> <li>• Ubuntu 16.04.2—Linux kernel version 4.4.0-62-generic</li> <li>• VMware vCenter 6.0, 6.5—Ubuntu 16.04.2 kernel version 4.4.0-62-generic</li> </ul>      |

**Table 1: Supported Platforms (Continued)**

| Contrail Release | Orchestrator Release | Operating System and Kernel Versions   |
|------------------|----------------------|--|
|                  | OpenStack Newton     | <ul style="list-style-type: none"> <li>• RHEL 7.4—Linux kernel version 3.10.0-693 (RHOSP10)</li> <li>• Ubuntu 16.04.2—Linux kernel version 4.4.0-62-generic</li> </ul> |
|                  | OpenStack Mitaka     | <ul style="list-style-type: none"> <li>• Ubuntu 14.04.5—Linux kernel versions 3.13.0-110-generic and 4.4.0-34-generic</li> </ul>                                       |

**NOTE:** In Contrail Release 4.0 and later, if the stock kernel version of your Ubuntu system is other than the required version, you can upgrade the kernel for all nodes in the cluster by using the following parameter in `cluster.json` for Server Manager or SM-Lite provisioning or `testbed.py`.

```
{
  "cluster" : [{
    "parameters" : {
      "provisioning" : {
        "contrail" : {
          "kernel_upgrade" : true
        }
      }
    }
  }]
}
```

## Server Requirements

The minimum requirement for a proof-of-concept (POC) system is 3 servers, either physical or virtual machines. All non-compute roles can be configured in each controller node. For scalability and availability reasons, it is highly recommended to use physical servers.

Each server must have a minimum of:

- 64 GB memory
- 300 GB hard drive
- 4 CPU cores
- At least one Ethernet port

## RELATED DOCUMENTATION

[Contrail Roles Overview](#)

*Downloading Installation Software*

## Contrail Node Roles, Processes, and Ports

### IN THIS SECTION

- [Processes in Logical Roles and Nodes | 13](#)
- [Roles and Ports | 15](#)

This section describes processes that run on various Contrail nodes and the ports used by those processes.

### Processes in Logical Roles and Nodes

The following are the processes in each of the logical roles or nodes in Contrail.

| Node           | Processes  |
|----------------|--|
| OpenStack node | This node runs non-Neutron OpenStack services, such as nova-api, nova-scheduler, nova-conductor, glance-api, and the like. |

*(Continued)*

| Node          | Processes  |
|---------------|--|
| Database node | Every instance of a database node runs the following processes: <ul style="list-style-type: none"> <li>• Cassandra</li> <li>• Zookeeper</li> </ul>   |
| Config node   | Every instance of a config node runs the following processes: <ul style="list-style-type: none"> <li>• neutron-server</li> <li>• contrail-api</li> <li>• ifmap</li> <li>• contrail-schema</li> <li>• contrail-svc-monitor</li> <li>• rabbitmq-server (optionally, this can be located on an external sever)</li> <li>• contrail-vcenter-plugin (optional)</li> </ul> |
| Control node  | Every instance of a control node runs the following processes: <ul style="list-style-type: none"> <li>• control-node</li> <li>• contrail-dns</li> <li>• contrail-named</li> </ul>  |
| Compute node  | Every instance of a compute node runs the following processes: <ul style="list-style-type: none"> <li>• nova-compute</li> <li>• contrail-vrouter-agent</li> <li>• contrail-tor-agent instances are run on a need basis, one instance per ToR, to manage TORs via OVS.</li> </ul>   |

*(Continued)*

| Node           | Processes   |
|----------------|---|
| Analytics node | Every instance of an analytics node runs the following processes: <ul style="list-style-type: none"> <li>• contrail-collector</li> <li>• contrail-analytics-api</li> <li>• contrail-query-engine .</li> <li>• contrail-snmp-collector</li> <li>• contrail-topology</li> </ul> |
| Web UI node    | Every instance of a Web UI node runs the following processes: <ul style="list-style-type: none"> <li>• contrail-webui</li> <li>• contrail-webui-middleware</li> </ul>   |

## Roles and Ports

This section presents notes about processes and ports by roles and nodes.

### OpenStack Role

#### Nova

##### API

- Ports
  - 8773 - EC2 API
  - 8774 - OpenStack API
  - 8775 - Metadata port

##### nova-novncproxy

- Ports
  - 5999

## Cinder

- Ports
  - 8776 - Cinder API

## Glance

- API
  - Ports
    - 9292 Glance API
- Registry
  - Ports
    - 9191 - Glance API

## Keystone

- Ports
  - 5000 - tenant port
  - 35357 - administrative port

## RabbitMQ

- Ports
  - 5672

## MySQL

- Ports
  - 3306



## Configuration Node

### Neutron Server

The neutron-server process serves the Openstack Networking (Neutron) API. It connects to contrail-api server and uses that as the backend for serving neutron API. It works in Active/Active mode in multi-node deployments.

- Service Name - neutron-server
  - Ports
    - 9696 - public port (HAproxy front end port)
    - 9697 - HAproxy back end port

### API Server

The contrail-api process exposes a REST-based interface for Contrail API, and:

- connects to Cassandra on database node(s) for persistence.
- publishes to IFMAP server for consumption of config information by other nodes.
- uses AMQP/rabbitmq to communicate with other API servers in multi-node deployment.
- Service Name - contrail-api

#### Ports

- 8082 - public port, accessible with credentials in auth mode
- 8095 - debug port, accessible only on localhost in auth mode
- 8084 - HAproxy front end port
- 9100 - HAproxy back end port

### IFMAP Server

The IFMAP server process acts as a bulletin board for pubsub (publish-subscribe) purposes. The contrail-control, contrail-schema, and contrail-svc-monitor connect to IFMAP server and act upon configuration change published from the API server.

- Ports
  - 8443 - basic authentication port

## Schema Transformer

The contrail-schema process listens to configuration changes done by user at higher-level constructs such as VirtualNetwork, and generates appropriate system configuration objects, lower-level construct such as RoutingInstance. The contrail-schema uses the REST interface of the API server to manipulate the system objects, and in multi-node deployments it works in Active/Backup mode.

- Service Name - contrail-schema
  - Ports
    - 8087 - Introspect port

## Service Monitor

The contrail-service-monitor process listens to configuration changes in service-template and service-instance, and in response, spawns and monitors virtual machines for services such as firewall, analyzer, and the like, using Nova API to spawn the virtual machines. In multi-node deployments it works in Active/Backup mode. The contrail-service-monitor process also syncs the Keystone tenants to Contrail, and the tenants appear in <http://<controller-ip>:8082/projects>.

- Service Name - contrail-svc-monitor
  - Ports
    - 8088 - Introspect port

## vCenter Plugin

The contrail-vcenter-plugin process integrates the Contrail controller with VMware vCenter as the orchestrator.

- Service Name - contrail-vcenter-plugin
  - Ports
    - 8234 - HTTP Introspect port

## Analytics Node

### Analytics REST API Server

- Service name - contrail-analytics-api
- Ports

- 8081 - public port

### **Collector**

- Service name - contrail-collector
- Ports
  - 8086 - public port

### **Query Engine**

- Service name - contrail-query-engine

### **Contrail SNMP Collector**

- Service name - contrail-snmp-collector

### **Contrail Topology**

- Service name - contrail-topology

### **Redis Server**

- Service name - redis

### **Web UI Role**

#### **webui**

- Service name - contrail-webui
- Ports
  - 8080 - http port redirects to https
  - 8143 - https port

#### **webui middleware**

- Service name - contrail-webui-middleware

## Redis Server

- Service name - redis

## Database Role

### Cassandra

- Service name - contrail-database
- Ports
  - 9160 - thrift port
  - 9042 - Cassandra Query Language (CQL) native port

### Zookeeper

- Service name - zookeeper
- Ports
  - 2181 - listen port

## Control Role

### control-node

- Service name - contrail-control
- Ports
  - 8083 - Introspect
  - 5269 - XMPP port
  - 5222 - TLS-based XMPP port
- Service name - contrail-dns
- Ports
  - 8092 - Introspect for DNS
  - 8093 - XMPP for DNS

## vRouter Role

### vRouter Agent

- Service name - contrail-vrouter
- Ports
  - 8085
  - 9090 - port for communication between vRouter agent and nova-compute

### Contrail ToR Agent

The contrail-tor-agent process runs the OVSDB protocol between a ToR and Contrail. Each contrail-tor-agent manages a single ToR, and multiple contrail-tor-agents can run on a single node. Each contrail-tor-agent service name is suffixed by a unique instance-id.

By default, the introspect service for the contrail-tor-agent is run on port 9009 + instance-id. The default port can be overridden in the fab file.

- Service name - contrail-tor-agent
- Ports
  - 9009 + <instance id>

# Installing Contrail and Provisioning Roles

## IN THIS CHAPTER

- Introduction to Containerized Contrail Modules | 22
- Downloading Installation Software | 26
- Installing the Operating System and Contrail Packages | 26
- Installing Containerized Contrail Clusters Using Server Manager | 28
- Installing Containerized Contrail Using Server Manager Lite (SM-Lite) | 32
- Supporting Multiple Interfaces on Servers and Nodes | 35
- Configuring the Control Node with BGP | 39
- Adding a New Node to an Existing Containerized Contrail Cluster | 44
- Using `contrailctl` to Configure Services Within Containers | 47
- Contrail Global Controller | 50
- Role and Resource-Based Access Control | 52

## Introduction to Containerized Contrail Modules

### IN THIS SECTION

- Why Use Containers? | 23
- Overview of Contrail Containers | 23
- Contrail 4.0 Containers | 24
- Summary of Container Design, Configuration Management, and Orchestration | 25

Starting with Contrail 4.0, some subsystems of Contrail are delivered as Docker containers.

## Why Use Containers?

Contrail software releases are distributed as sets of packages for each of the subsystem modules of a Contrail system. The Contrail modules depend on numerous open source packages and provisioning tools and are validated on specific Linux distributions. Each module has its own dependency chains and its own configuration parameters.

These dependencies lead to complexities of deployment, including:

- The Linux version of the target system must match exactly to the version upon which Contrail is qualified, or the installation might fail.
- A deployment that succeeds despite an operating system mismatch could pull dependent packages from a customer mirror site that don't match the dependencies with which the Contrail system was qualified, creating potential for failure.
- Change in any package on the target system creates a risk of failure of dependencies in the Contrail software, creating a need for requalification upon any system change.
- Currently, provisioning tools such as Fuel, Juju, Puppet, and the like interact directly with Contrail services. Over time, these tools become more complex, requiring interaction with the lowest level of details of Contrail service parameters.

Containerizing some Contrail subsystems reduces the complexity of deploying Contrail and provides a straightforward, simple way to deploy and operate Contrail.

## Overview of Contrail Containers

Starting with Contrail 4.0, some of the Contrail subsystems are delivered as Docker containers that group together related functional components. Each container file includes an INI-based configuration file for configuring the services within the container. The purpose of the INI is to provide enough high-level configuration entries to configure all services within the container, while masking the complexity of the internal service configuration. The container configuration files are available on the host system and mounted within specific containers.

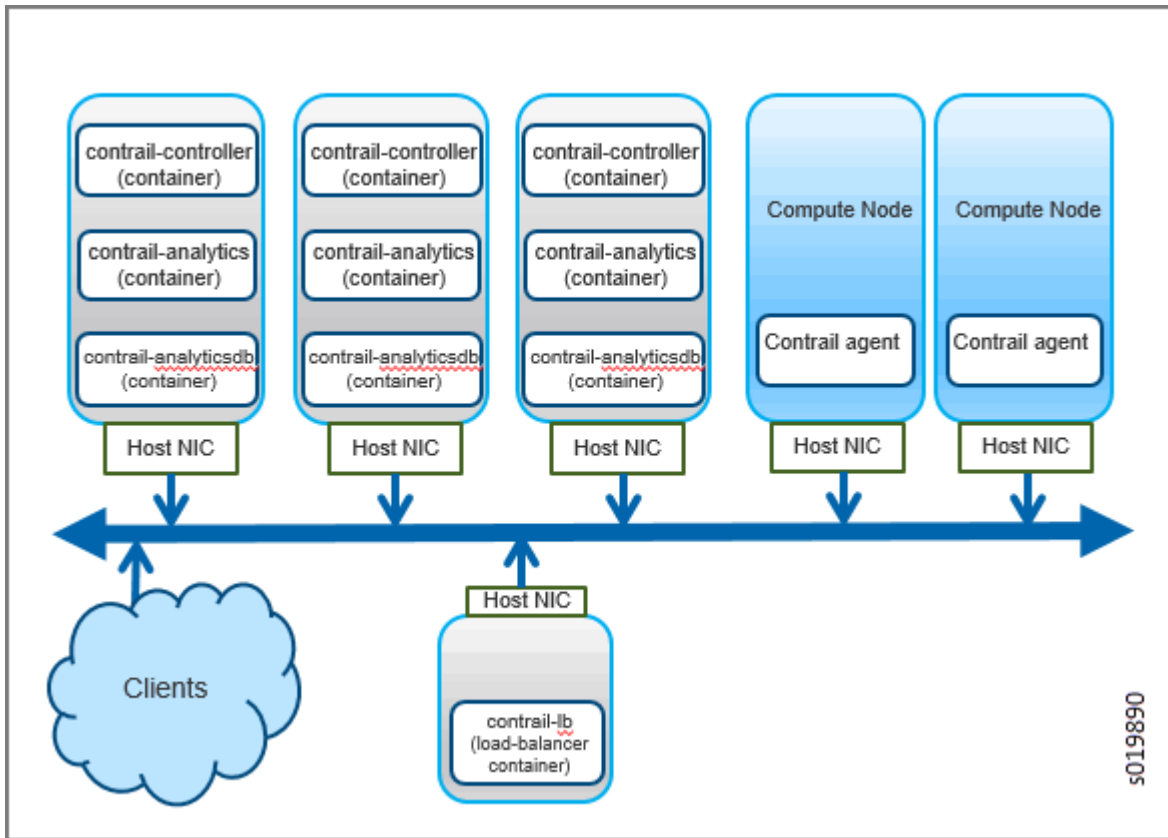
In Contrail 4.0, the containerized components include Contrail controller, analytics, and load-balancer applications. Contrail OpenStack components are not containerized at this time.

In Contrail 4.0.1, the containerized components include OpenStack Ocata services. Only OpenStack Ocata services are containerized. Mitaka and Newton SKUs of OpenStack are still provisioned as non-containerized host services.

All Contrail containers run with the host network, without using a Docker bridge, however, all services within the container listen on the host network interface. Some services, such as RabbitMQ, require extra parameters, such as a host-based PID namespace.

The intention is to build a composable Contrail core system of containers that can be used with differing cloud and container orchestration systems, such as OpenStack, Kubernetes, Mesos, and the like.

**Figure 1: Sample Configuration Containerized Contrail**



## Contrail 4.0 Containers

This section describes the containers in Contrail 4.0 and their contents.

### contrail-controller

The `contrail-controller` container includes all Contrail applications that make up a Contrail controller, including:

- All configuration services, such as `contrail api`, `config-nodemgr`, `device-manager`, `schema`, `svc-monitor`, and `CONFIGDB`.
- All control services, such as `contrail-control`, `control-nodemgr`, `contrail-dns`, and `contrail-named`.
- All Web UI services, such as `contrail-webui` and `contrail-webui-middleware`.



- Configuration database (Cassandra)
- Zookeeper
- RabbitMQ
- Redis for Web Ui

### **contrail-analytics**

The `contrail-analytics` container includes all Contrail analytics services, including:

- `alarm-gen`
- `analytics-api`
- `analytics-nodemgr`
- `contrail-collector`
- `query-engine`
- `snmp-collector`
- `contrail-topology`

### **contrail-analyticsdb**

The `contrail-analyticsdb` container has Cassandra for the analytics database and Kafka for streaming data.

### **contrail-lb**

The `contrail-lb` loadbalancer container includes all components that provide load-balancing and high availability to the system, such as HAproxy, keepalive, and the like.

In previous releases of Contrail, HAproxy and keepalive were included in most services to load-balance Contrail service endpoints. Starting with Contrail 4.0, the load-balancers are taken out of the individual services and held instead in a dedicated `loadbalancer` container. An exception is HAproxy as part of the vrouter agent, which can be used to implement Load-Balancing as a Service (LBaaS).

The `loadbalancer` container is an optional container, and customers can choose to use their own load-balancing system.

## **Summary of Container Design, Configuration Management, and Orchestration**

The following are key features of the new architecture of Contrail containers.

- All of the Contrail containers are multiprocess Docker containers.
- Each container has an INI-based configuration file that has the configurations for all of the applications running in that container.
- The user toolset `contrailctl` is used to manage the container configuration files.
- Each container is self-contained, with minimal external orchestration needs.
- A single tool, Ansible, is used for all levels of building, deploying, and provisioning the containers. The Ansible code for the Contrail system is named `contrail-ansible` and kept in a separate repository. The Contrail Ansible code is responsible for all aspects of Contrail container build, deployment, and basic container orchestration.

## RELATED DOCUMENTATION

| *Using `contrailctl` to Configure Services Within Containers*

## Downloading Installation Software

All components necessary for installing the Contrail Controller are available for each Contrail release, for the supported Linux operating systems and versions, and for the supported versions of OpenStack.

All installation images can be downloaded from <https://www.juniper.net/support/downloads/?p=contrail#sw>.

The Contrail image includes the following software:

- All dependent software packages needed to support installation and operation of OpenStack and Contrail
- Contrail Controller software – all components
- OpenStack release currently in use for Contrail

## Installing the Operating System and Contrail Packages

Install the stock CentOS or Ubuntu operating system image appropriate for your version of Contrail onto the server. See [Supported Platforms Contrail 4.0.x](#) or [Supported Platforms Contrail 4.1](#). Then install Contrail packages separately.

The following are general guidelines for installing the operating system and preparing to install Contrail.

1. Install a CentOS or Ubuntu minimal distribution as desired on all servers. Typically, for CentOS this is a basic ISO install; for Ubuntu, use a core server install, with only OpenSSH and no other packages. Follow the published operating system installation procedure for the selected operating system; refer to the website for the operating system.
2. Install Contrail Server Manager, see [Installing Server Manager](#).
3. Create an image.json with the Ubuntu or CentOS image to be used to reimage the target server.

Sample JSON Snippet

```
{
  "image": [
    {
      "category": "image",
      "id": "ubuntu-14.04.04",
      "parameters": {
        "kickseed": "/etc/contrail_smgr/kickstarts/contrail-ubuntu_trusty.seed",
        "kickstart": "/etc/contrail_smgr/kickstarts/contrail-ubuntu_trusty.ks"
      },
      "path": "/path/to/ubuntu-image.iso",
      "type": "ubuntu",
      "version": "14.04.04"
    }
  ]
}
```

4. Use Server Manager to add the image.json, to be used for reimagining.

```
server-manager add image -f image.json
```

For full installation information, see [Installing Containerized Contrail Clusters Using Server Manager](#) and [Installing Containerized Contrail for Single- and Multi-Node Systems Using Server Manager Lite](#)

## RELATED DOCUMENTATION

[Introduction to Containerized Contrail Modules](#)

[Contrail Roles Overview](#)

[Installing Containerized Contrail Clusters Using Server Manager](#)

[Installing Containerized Contrail Using Server Manager Lite \(SM-Lite\)](#)

[Upgrading Contrail 3.2 to 4.0](#)[Download Software](#)

## Installing Containerized Contrail Clusters Using Server Manager

### IN THIS SECTION

- [Installing Server Manager | 28](#)
- [Creating Objects with Server Manager and JSONs | 29](#)
- [Preparing the Target System for Provisioning | 30](#)
- [Provisioning the System | 31](#)

This topic presents the steps needed to install containerized Contrail Release 4.0 in a single- or multi-node configuration.

You can use Contrail Server Manager or Server Manager Lite (SM-Lite) to provision containerized Contrail.

This is the procedure for using Server Manager. SM-Lite is typically used for Contrail networking, only.

The installation is completed using the following major activities:

### Installing Server Manager

Before installing Contrail Release 4.0, you must install Contrail Server Manager on a server running Ubuntu.

1. Install the Server Manager wrapper package:

```
dpkg -i contrail-server-manager-installer_[version~sku].deb
```

2. Install Server Manager and its dependent packages, including docker-engine and Cobbler:

```
cd /opt/contrail/contrail_server_manager/; ./setup.sh --all --hostip=[IP address of SM]
```

**NOTE:** The `setup.sh` script could fail to start the Docker registry if you are installing over an existing version of Server Manager.

If you encounter the Docker registry failure to start error, use the following workaround:

- a. In the `setup.sh` script, comment out the line containing the `docker run` command.
  - b. `dpkg --purge contrail-server-manager`
  - c. `setup.sh --all --hostip=[IP address of SM]`
3. When the Server Manager install completes with no errors, modify the DHCP template at `/etc/cobbler/dhcp.template` to include the details of the subnet being reimaged or provisioned. Be sure to include DNS details.

**NOTE:** Container hosts require Internet connectivity at this point to launch the containers.

4. Start the Server Manager process:

```
service contrail-server-manager start
```

For more details about the Server Manager installation process, refer to *Installing Server Manager*.

## Creating Objects with Server Manager and JSONs

Once Server Manager is installed, use Server Manager commands with a JSON file to create Contrail objects.

Configure an appropriate JSON file with the IP addresses, interface names, and password strings specific to your system.

Select a sample JSON from the following and update it to match your system:

- Sample JSONs for an All-In-One-Node Cluster:  
[Sample JSONs for an all-in-one, single node with roles](#)
- Sample JSONs for a Multinode Cluster with Two Nodes:  
[Sample JSONs for a Multinode Cluster](#)
- Sample JSONs for a Multinode Cluster with 7 Nodes and High Availability:  
[Sample JSONs for a Multinode Cluster with High Availability:](#)

The following procedure helps you create a target system that includes the components for OpenStack, Contrail controller, analytics, analytics database, and agent. The controller, analytics, and analytics

database services are provisioned using Contrail containers, however, the agent service is configured on the bare-metal target host.

### 1. Configure the images needed for reimagining and provisioning.

#### a. Add the Ubuntu image from JSON (used for reimagining)

```
server-manager add image -f image-ubuntu-14.04.04.json
```

#### b. Add the Contrail Debian image and containers from JSON (used for provisioning)

```
server-manager add image -f contrail_image.json
```

**NOTE:** Wait for this command to complete, it operates in the background and can take as long as 5 minutes to complete.

Before proceeding, check for a log message: Image add/Modify success, in **/var/log/contrail-server-manager/debug.log**.

### 2. Configure the cluster(s).

For an all-in-one, single-node demo system:

```
server-manager add cluster -f <all_ins_one_cluster>.json
```

For a multi-node system:

```
server-manager add cluster -f <multi_node_cluster>.json
```

If a Keystone admin password is generated, be sure to write it down.

**NOTE:** During installation, if a password is provided, no other passwords are generated. If a password is *NOT* provided, all needed passwords are generated.

### 3. Configure the server.

```
server-manager add server -f contrail_server.json
```

Repeat this step for every server in the system, using the correct server.json file, based on the number of servers or type of your system.

## Preparing the Target System for Provisioning

To prepare the target system for provisioning, reimage the target system(s), including the Contrail server and the OpenStack server.

- For an all-in-one, single-node demo system:

```
server-manager reimage --server_id <server_id> <ubuntu_image>
```

- For a multi-node system:

```
server-manager reimage --cluster_id <multi_node> <ubuntu_image>
```

## Provisioning the System

Launch the system provisioning.

- For an all-in-one, single-node demo system:

```
server-manager provision --cluster_id <all_in_one_cluster> combined_image_mainline
```

- For a multi-node system:

```
server-manager provision --cluster_id <multi_node> combined_image_mainline
```

The `server-manager provision` command first provisions the OpenStack role, which includes using Puppet manifests. Next, the command provisions Contrail Docker containers and compute nodes.

You can monitor progress of the provisioning by observing log entries:

```
/var/log/contrail-server-manager/debug.log
```

When provisioning is complete, confirm successful installation by creating a virtual network and launching virtual machines from the OpenStack node.

## RELATED DOCUMENTATION

[Sample JSONs for an all-in-one, single node with roles](#)

[Sample JSONs for a Multinode Cluster](#)

[Sample JSONs for a Multinode Cluster with High Availability](#)

*Introduction to Containerized Contrail Modules*

[Contrail Roles Overview](#)

*Installing the Operating System and Contrail Packages*

*Installing Containerized Contrail Using Server Manager Lite (SM-Lite)*

[Upgrading Contrail 3.2 to 4.0](#)

## Installing Containerized Contrail Using Server Manager Lite (SM-Lite)

### IN THIS SECTION

- [Preparing for SM-Lite Installation | 32](#)
- [Installing SM-Lite | 33](#)
- [Provisioning Contrail Using SM-Lite | 34](#)
- [Sample JSONs and Testbed.py | 34](#)

Server Manager Lite (SM-Lite) is a streamlined version of Server Manager. SM-Lite has functionality similar to Server Manager, except it does not perform reimaging. SM-Lite is typically used for Contrail networking only.

You can use Contrail Server Manager or Server Manager Lite (SM-Lite) to provision containerized Contrail. To use SM-Lite for provisioning, you install regular Server Manager, then use SM-Lite commands for provisioning.

This topic is the procedure for installing and provisioning Contrail 4.0 and later using SM-Lite.

The SM-Lite installation of containerized Contrail is completed using the following major activities:

### Preparing for SM-Lite Installation

For Contrail 4.0, SM-Lite install is only supported on Ubuntu 14.04.5. Contrail 4.1 adds support for Ubuntu 16.04.2.

Before installing containerized Contrail, you must install Server Manager SM-Lite on a server running a supported version of Ubuntu.

You can install SM-Lite on any server or node, and you can run it using multiple options:

- Provision a single node or VM for Contrail, then install SM-Lite on the same node and use it to perform Contrail provisioning.
- Use a separate node or VM to install SM-Lite, and provision Contrail with the rest of the nodes.
- Use a node or VM that has Contrail roles (typically a config node) to install SM-Lite.

To specify servers and associated Contrail roles and cluster details, you can use either a `testbed.py` or JSONs based on the sample JSONs used with regular Server Manager. The image details come from the image JSON.



## Prerequisites

Before installing the SM-Lite package, ensure the following cautions have been met:

- Ensure that the `sources.list` is present and empty.
- Ensure that `/etc/apt/sources.list.d/` is not pointing to any external or local repositories.

If you are installing SM-Lite on a VM spawned from OpenStack Horizon or from an Ubuntu cloud image:

- Verify that the VM is set up correctly with hostname and domain details:
  - The hostname and domain name are present in `/etc/hosts` as follows:
 

```
<Host non mgmt IP> <server hostname>.<domain_name> <server hostname>
```
  - The domain name is present in `/etc/resolv.conf` as follows:
 

```
search <domain_name>
```
- When correctly set up, the command `"hostname -f"` will return `< hostname >.< domain_name >`

## Installing SM-Lite

1. Install the regular Server Manager wrapper package (Debian). (An example package is: `contrail-server-manager-installer_2.22~juno_all.deb`.)

```
dpkg -i </github-build/mainline/<build_number> /ubuntu-14-04/mitaka/artifacts/ contrail-server-manager-installer_4.0.0.0-<build-number>-mitaka_all.deb>
```

2. Now you can use the SM-Lite `provision_containers` command to provision Contrail.

The full syntax and available options of the `provision_containers.sh` script:

```
Help:
`/opt/contrail/contrail_server_manager/provision_containers.sh -h`
`-h --help`
`-cj <cluster json path>`
`-sj <server json path>`
`-ij <image json path>`
`-t|--testbed <testbed.py path>`
`-c <contrail cloud docker package path>`
`-cid|--cluster-id <cluster-id>`
`-ni|--no-install-sm-lite`
```

The `-ni` option is used to reprovision an existing cluster, create a new cluster, or upgrade an existing cluster with a different version.

For more details about SM-Lite, refer to [Installing and Using Server Manager Lite](#).

## Provisioning Contrail Using SM-Lite

To activate SM-Lite and provision the target systems, use `provision_containers.sh` along with system-specific configuration information.

Provision Contrail with system-specific configuration information using one of the following options:

- Using JSONs

```
/opt/contrail/contrail_server_manager/provision_containers.sh -cj <cluster json path> -sj <server json path> -ij <image json path> --cluster-id <Cluster ID>
```

- Using `testbed.py` and `contrail-docker-cloud.tgz`

```
/opt/contrail/contrail_server_manager/provision_containers.sh -t <testbed.py path> -c <contrail-cloud-docker tgz path> --cluster-id <Cluster ID>
```

The SM-Lite provisioning logs can be viewed at `/var/log/contrail-server-manager/debug.log`.

Running the `provision_containers.sh` script does the following:

1. Installs SM-Lite components: sm client, sm webui, sm monitoring/inventory, and the like.
2. Prepares the targets for provisioning by running the `preconfig.py` script.
3. Adds Server Manager objects for cluster, server, and image from the JSONs or the `testbed.py` as provided.
4. Loads Docker containers and pushes them to the registry in the background.
5. Launches the Contrail provisioning, using the Server Manager client CLI.

## Sample JSONs and Testbed.py

Use the SM-Lite command `provision_containers.sh` with a JSON file or a `testbed.py` to provision Contrail objects.

Configure an appropriate JSON file or `testbed.py` with the IP addresses, interface names, and password strings specific to your system, then identify its path when you use the SM-Lite `provision_containers.sh` command.

Select a sample JSON or `testbed.py` from the following and update it to match your system:

- [Sample testbed.py for Provisioning Containers with SM-Lite](#)
- [Sample combined JSON for provisioning Contrail 4.1 and Openstack Ocata with SM Lite \(all in one node & single interface\)](#)
- [Sample JSONs for a Multinode Cluster with Two Nodes](#)
- [Sample JSONs for a Multinode Cluster with 7 Nodes and High Availability](#)

## RELATED DOCUMENTATION

[Sample JSONs for an All-In-One-Node Cluster \(for demo\)](#)

[Sample JSONs for a Multinode Cluster with Two Nodes](#)

[Sample JSONs for a Multinode Cluster with 7 Nodes and High Availability](#)

[Sample testbed.py for Provisioning Containers with SM-Lite](#)

*[Introduction to Containerized Contrail Modules](#)*

[Contrail Roles Overview](#)

*[Installing the Operating System and Contrail Packages](#)*

*[Installing Containerized Contrail Clusters Using Server Manager](#)*

[Upgrading Contrail 3.2 to 4.0](#)

## Supporting Multiple Interfaces on Servers and Nodes

### IN THIS SECTION

- [Support for Multiple Interfaces | 36](#)
- [Server Interface Examples | 38](#)
- [Interface Naming and Configuration Management | 38](#)

This section describes how to set up and manage multiple interfaces.

## Support for Multiple Interfaces

Servers and nodes with multiple interfaces should be deployed with exclusive management and control and data networks. In the case of multiple interfaces per server, the expectation is that the management network provides only management connectivity to the cluster, and the control and data network carries the control plane information and the guest traffic data.

Examples of control traffic include the following:

- XMPP traffic between the control nodes and the compute nodes.
- BGP protocol messages across the control nodes.
- Statistics, monitoring, and health check data collected by the analytics engine from different parts of the system.

In Contrail, control and data must share the same interface, configured in the `testbed.py` file in a section named `control_data`.

## Number of cfm Nodes Supported

The Contrail system can have any number of `cfm` nodes.

## Uneven Number of Database Nodes Required

In Contrail, Apache ZooKeeper resides on the database node. Because a ZooKeeper ensemble operates most effectively with an odd number of nodes, it is required to have an odd number (3, 5, 7, and so on) of database nodes in a Contrail system.

## Support for VLAN Interfaces

A VLAN ID can also be specified in the `server.json` file under the `network, interfaces` section, similar to the following example:

```
“network”: {
  “interfaces”: [
    {
      “name”: “vlan2003”,
      “type” : “vlan”,
      “vlan”: “2003”,
      “parent_interface”: “bond0”,
      “ip_address”: “10.224.11.10/24”,
      “default_gateway”: “10.224.12.1”
    }
  ]
}
```

```

    }
  ]
}

```

## Support for Bonding Options

Contrail provides support for bond interface options.

The default bond interface options are:

```
miimon=100, mode=802.3ad(lacp), xmit_hash_policy=layer3+4
```

For Contrail 4.0 and later, in the provisioning file bond section, anything other than name and member are treated as a bond interface option, and provisioned as such. The following is an example:

```

"network": {
  "interfaces": [
    name: "bond0",
    "type" : "bond",
    "bond_options" : {"miimon": "100", "mode": "802.3ad", "xmit_hash_policy":
"layer3+4"},
    "member_interfaces": ["p20p1", "p20p2"]
  ],
},
],

```

## Support for Static Route Options

Contrail provides support for adding static routes on target systems. This option is ideal for use cases in which a system has servers with multiple interfaces and has control data or management connections that span multiple networks.

The following shows static routes added in the server.json under the 'network' section.

```

"network": {
  "routes": [
    {
      "gateway": "3.3.2.254",
      "interface": "enp129s0f0",
      "netmask": "255.255.255.0",
      "network": "3.3.4.0"
    }
  ],
},

```

```

    {
      "gateway": "3.3.3.254",
      "interface": "enp129s0f1",
      "netmask": "255.255.255.0",
      "network": "3.3.5.0"
    }
  ]
}
```

## Server Interface Examples

In Contrail Release 1.10 and later, control and data are required to share the same interface. A set of servers can be deployed in any of the following combinations for management, control, and data:

- `mgmt=control=data` -- Single interface use case
- `mgmt, control=data` -- Exclusive management access, with control and data sharing a single network.

In Contrail, the following server interface combinations are not allowed:

- `mgmt=control, data`--Dual interfaces in Layer 3 mode, management and control shared on a single network
- `mgmt, control, data`—Complete exclusivity across management, control, and data traffic.

## Interface Naming and Configuration Management

On a standard Linux installation there is no guarantee that a physical interface will come up with the same name after a system reboot. Linux NetworkManager tries to accommodate this behavior by linking the interface configurations to the hardware addresses of the physical ports. However, Contrail avoids using hardware-based configuration files because this type of solution cannot scale when using remote provisioning and management techniques.

The Contrail alternative is a threefold interface-naming scheme based on *<bus, device, port (or function)>*. As an example, on a server operating system that typically assigns interface names such as `p4p0` and `p4p1` for onboard interfaces, the Contrail system assigns `p4p0p0` and `p4p0p1`, when using the optional `contrail-interface-name` package.

When the `contrail-interface-name` package is installed, it uses the threefold naming scheme to provide consistent interface naming after reboots. The `contrail-interface-name` package is installed by default when a Contrail ISO image is installed. If you are using an RPM-based installation, you should install the `contrail-interface-name` package before doing any network configuration.

If your system already has another mechanism for getting consistent interface names after a reboot, it is not necessary to install the **contrail-interface-name** package.

## Configuring the Control Node with BGP

An important task after a successful installation is to configure the control node with BGP. This procedure shows how to configure basic BGP peering between one or more virtual network controller control nodes and any external BGP speakers. External BGP speakers, such as Juniper Networks MX80 routers, are needed for connectivity to instances on the virtual network from an external infrastructure or a public network.

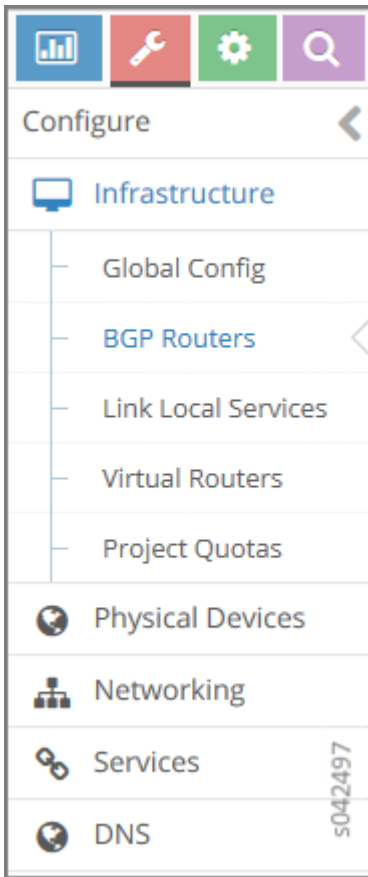
Before you begin, ensure that the following tasks are completed:

- The Contrail Controller base system image has been installed on all servers.
- The role-based services have been assigned and provisioned.
- IP connectivity has been verified between all nodes of the Contrail Controller.
- You can access the Contrail user interface at <http://nn.nn.nn.nn:8080>, where *nn.nn.nn.nn* is the IP address of the configuration node server that is running the **contrail-webui** service.

To configure BGP peering in the control node:

1. From the Contrail Controller module control node (<http://nn.nn.nn.nn:8080>), select **Configure > Infrastructure > BGP Routers**; see [Figure 2 on page 40](#).

Figure 2: Configure > Infrastructure > BGP Routers



A summary screen of the control nodes and BGP routers is displayed; see [Figure 3 on page 40](#).

Figure 3: BGP Routers Summary

Configure > Infrastructure > BGP Routers

BGP Routers + 🗑️ ⬇️ 🔍 ^

| <input type="checkbox"/> IP Address   | Type         | Vendor   | HostName  |  |
|---------------------------------------|--------------|----------|-----------|--|
| <input type="checkbox"/> 10.84.25.31  | Control Node | contrail | b5s31     |  |
| <input type="checkbox"/> 10.84.11.252 | BGP Router   | mx       | a3-mx80-1 |  |
| <input type="checkbox"/> 10.84.25.30  | Control Node | contrail | b5s30     |  |
| <input type="checkbox"/> 10.84.25.29  | Control Node | contrail | b5s29     |  |
| <input type="checkbox"/> 10.84.25.28  | Control Node | contrail | b5s28     |  |
| <input type="checkbox"/> 10.84.25.27  | Control Node | contrail | b5s27     |  |
| <input type="checkbox"/> 10.84.11.253 | BGP Router   | mx       | mx1       |  |

Total: 7 records | 50 Records ▾ ⏪ Page 1 of 1 ⏩

s042498



2. (Optional) The global AS number is 64512 by default. To change the AS number, on the **BGP Router** summary screen click the gear wheel and select **Edit**. In the Edit BGP Router window enter the new number.
3. To create control nodes and BGP routers, on the **BGP Routers** summary screen, click the **+** icon. The **Create BGP Router** window is displayed; see [Figure 4 on page 41](#).

**Figure 4: Create BGP Router**

4. In the **Create BGP Router** window, click **BGP Router** to add a new BGP router or click **Control Node** to add control nodes.  
For each node you want to add, populate the fields with values for your system. See [Table 2 on page 42](#).

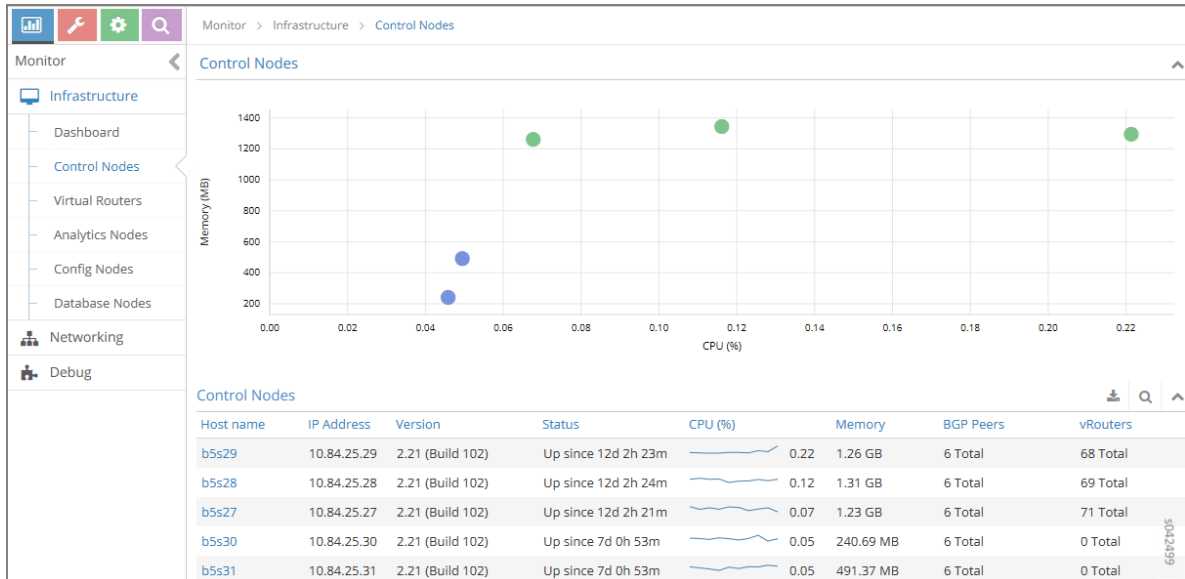
Table 2: Create BGP Router Fields

| Field                      | Description  |
|----------------------------|--|
| <b>Hostname</b>            | Enter a name for the node being added.   |
| <b>Vendor ID</b>           | Required for external peers. Populate with a text identifier, for example, "MX-0". (BGP peer only) |
| <b>IP Address</b>          | The IP address of the node.  |
| <b>Router ID</b>           | Enter the router ID.   |
| <b>Autonomous System</b>   | Enter the AS number for the node. (BGP peer only)  |
| <b>Address Families</b>    | Enter the address family, for example, <b>inet-vpn</b>   |
| <b>Hold Time</b>           | BGP session hold time. The default is 90 seconds; change if needed.                                |
| <b>BGP Port</b>            | The default is 179; change if needed.  |
| <b>Authentication Mode</b> | Enable MD5 authentication if desired.  |
| <b>Authentication key</b>  | Enter the Authentication Key value.  |
| <b>Physical Router</b>     | The type of the physical router.   |
| <b>Available Peers</b>     | Displays peers currently available.  |
| <b>Configured Peers</b>    | Displays peers currently configured.   |

5. Click **Save** to add each node that you create.
6. To configure an existing node as a peer, select it from the list in the **Available Peers** box, then click **>>** to move it into the **Configured Peers** box.  
Click **<<** to remove a node from the **Configured Peers** box.

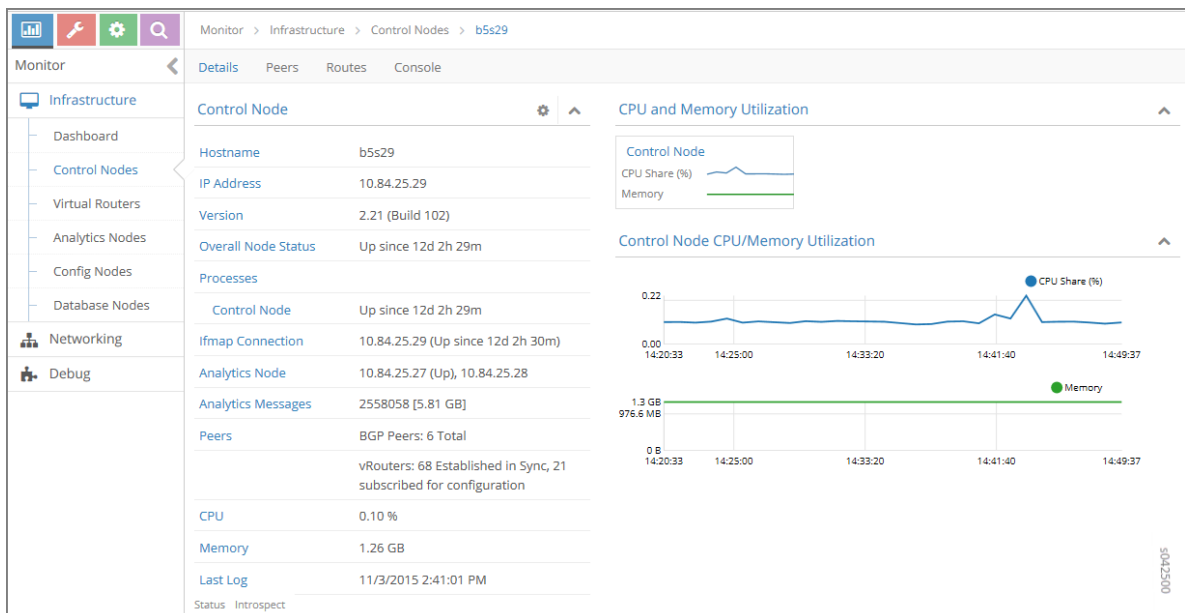
- You can check for peers by selecting **Monitor > Infrastructure > Control Nodes**; see [Figure 5](#) on page 43.

**Figure 5: Control Nodes**



In the **Control Nodes** window, click any hostname in the memory map to view its details; see [Figure 6](#) on page 43.

**Figure 6: Control Node Details**



8. Click the **Peers** tab to view the peers of a control node; see [Figure 7 on page 44](#).

**Figure 7: Control Node Peers Tab**

| Peer          | Peer Type | Peer ASN | Status               | Last flap | Messages (Recv/Sent) |
|---------------|-----------|----------|----------------------|-----------|----------------------|
| ▶ 10.84.21.1  | XMPP      | -        | Established, in sync | -         | 35497 / 138229       |
| ▶ 10.84.21.10 | XMPP      | -        | Established, in sync | -         | 35511 / 137011       |
| ▶ 10.84.21.11 | XMPP      | -        | Established, in sync | -         | 37045 / 141735       |
| ▶ 10.84.21.12 | XMPP      | -        | Established, in sync | -         | 37493 / 140054       |
| ▶ 10.84.21.13 | XMPP      | -        | Established, in sync | -         | 35540 / 137864       |
| ▶ 10.84.21.14 | XMPP      | -        | Established, in sync | -         | 40098 / 112770       |
| ▼ 10.84.21.15 | XMPP      | -        | Established, in sync | -         | 35450 / 137599       |

```

Details:
- {
  name: "b5s29:10.84.21.15",
  value: - {
    xmppPeerInfoData: - {
      state_info: - {
        last_state: "Active",
        state: "Established",

```

## RELATED DOCUMENTATION

*[Creating a Virtual Network with Juniper Networks Contrail](#)*

*[Creating a Virtual Network with OpenStack Contrail](#)*

## Adding a New Node to an Existing Containerized Contrail Cluster

### IN THIS SECTION

- [Controller Configuration | 45](#)

This is the initial process for adding a new node to an existing cluster in containerized Contrail.

## Controller Configuration

1. Create contrailctl configuration and start a controller container on a new node.

- Configure contrailctl configurations in `/etc/contrailctl/controller.conf` .

See examples on github at:

[contrail-docker/tools/python-contrailctl/examples/configs/controller.conf](https://github.com/contrail-docker/tools/python-contrailctl/examples/configs/controller.conf)

- Start the controller container. For more information, see [How to run Contrail Docker containers](#).
- Wait for the new containers to come up completely.

2. Configure the existing cluster nodes with new nodes.

The purpose of this step is to reconfigure the existing cluster application configurations to include newly added servers, then restart to accommodate the configuration changes.

You can do this by using one of two methods described below:

### Using contrailctl to add node configuration on existing containers

You can use contrailctl to add the node configuration on existing containers by running the following steps on all existing containers on all cluster nodes.

**NOTE:** Run this step first on all zookeeper *follower* nodes, then run on the leader node.

1. Determine which node is the leader node.

To determine which node is the leader and which are followers in a zookeeper cluster, run the following commands against your zookeeper cluster nodes.

```
$ echo stat | nc 192.168.0.102 2181 | grep Mode Mode: leader
```

```
$ echo stat | nc 192.168.0.100 2181 | grep Mode Mode: follower
```

2. Run contrailctl on all the existing containers in all cluster nodes, follower nodes first and leader node last.

```
$ contrailctl config node add -h
usage: contrailctl config node add [-h] -t {controller,analyticsdb,analytics}
      -n NODE_ADDRESSES [-s SEED_LIST]
      [-f CONFIG_FILE] -c
```

```

{controller,analyticsdb,analytics,agent,lb,kubemanager,mesosmanager}
    [--config-list CONFIG_LIST]

optional arguments:
  -h, --help            show this help message and exit
  -t {controller,analyticsdb,analytics}, --type {controller,analyticsdb,analytics}
                        Type of node
  -n NODE_ADDRESSES, --node-addresses NODE_ADDRESSES
                        Comma separated list of node addresses
  -s SEED_LIST, --seed-list SEED_LIST
                        Comma separated list of seed nodes to be used
  -f CONFIG_FILE, --config-file CONFIG_FILE
                        Master config file path
  -c {controller,analyticsdb,analytics,agent,lb,kubemanager,mesosmanager}, --component
{controller,analyticsdb,analytics,agent,lb,kubemanager,mesosmanager}
                        contrail role to be configured
  --config-list CONFIG_LIST
                        comma separated list of config nodes. Optional it is
                        needed only when the new controller nodes added are
                        config service disabled

# Add new controllers in analytics container
$ contrailctl config node add -t controller -n 192.168.0.10,192.168.0.11 -s
192.168.0.102,192.168.0.99 -c analytics

# Add new controllers in analyticsdb container
$ contrailctl config node add -t controller -n 192.168.0.10,192.168.0.11 -s
192.168.0.102,192.168.0.99 -c analyticsdb

# Add new controllers in other controller containers
$ contrailctl config node add -t controller -n 192.168.0.10,192.168.0.11 -s
192.168.0.102,192.168.0.99 -c controller

```

## Manually configure contrailctl on all containers and sync the configs

### 1. Determine which node is the leader node.

To determine which node is the leader and which are followers in a zookeeper cluster, run the following commands against your zookeeper cluster nodes.

```
$ echo stat | nc 192.168.0.102 2181 | grep Mode Mode: leader
```

```
$ echo stat | nc 192.168.0.100 2181 | grep Mode Mode: follower
```

2. Manually configure `/etc/contrailctl/controller.conf` with new nodes for various `*_list` configurations and `config_seed_list`. See examples at: <https://github.com/Juniper/contrail-docker/blob/master/tools/python-contrailctl/examples/configs/controller.conf>

3. Run `contrailctl` within the containers.

```
$ docker exec <container name> contrailctl config sync -c <component name>
```

```
$ docker exec controller contrailctl config sync -c controller
```

## Removing Nodes in an Existing Containerized Cluster

For the first version of containerized Contrail, there is no script available for removing a node from an existing cluster. If it is necessary to remove a node from an existing containerized Contrail cluster, please contact Juniper Networks JTAC for assistance.

## Using `contrailctl` to Configure Services Within Containers

### IN THIS SECTION

- [What is `contrailctl`? | 47](#)
- [Command Operations | 48](#)

Starting with Contrail 4.0, some subsystems of Contrail are delivered as Docker containers. The `contrailctl` tool is a set of commands that enable a user to make some changes to the configuration file within a Contrail container.

### What is `contrailctl`?

Starting with Contrail 4.0, some modules of the Contrail architecture have been grouped by function into Docker containers. Each container has an INI-based configuration file to maintain the specific configuration for that container. The `contrailctl` is a tool within the container that provides the user a simple command structure for provisioning and operating the Contrail services packaged in the container.

Because it is complex to provision and manage the various services within Contrail containers, the `contrailctl` tool helps configure the services in the container to be in sync with the container-specific configuration files.

The `contrailctl` tool is driven by the single INI-based configuration file per container, for example, the `/etc/contrailctl/controller.conf` for the controller container. Any state changes of the services within the container must be made according to the configuration in the `contrailctl` configuration file for that container. The `contrailctl` configuration files are available on each node at a default location of `/etc/contrailctl/*.conf`.

Any changes made to the configuration files in the node are available within the container.

Each Contrail container has a separate `contrailctl` configuration file, currently:

- `contrail-controller`—`/etc/contrailctl/controller.conf`
- `contrail-analytics`—`/etc/contrailctl/analytics.conf`
- `contrail-analyticsdb`—`/etc/contrailctl/analyticsdb.conf`

Sample container configuration files can be seen at

<https://github.com/Juniper/contrail-docker/tree/master/tools/python-contrailctl/examples/configs>

## Command Operations

The `contrailctl` is used within the node that holds a container. It is used at startup to configure and start the services within the container. The user must connect to the container to run `contrailctl`, or use the following command syntax to run `contrailctl`:

```
docker exec <container name or id> contrailctl <arguments>
```

Example:

```
docker exec controller contrailctl config sync -c controller -Fv
```

The main function of the `contrailctl` is to ensure that the desired configurations for the services within a container are in sync with the `contrailctl` master configuration file within the container.

## Command Syntax and Options

```
$ contrailctl config sync -h
usage: contrailctl config sync [-h] [-f CONFIG_FILE] -c
{controller,analyticsdb,analytics,agent,lb,kubemanager,mesosmanager}
[-F] [-t TAGS]

optional arguments:
  -h, --help            show this help message and exit
  -f CONFIG_FILE, --config-file CONFIG_FILE
```



```

                                Master config file path
-c {controller,analyticsdb,analytics,agent,lb,kubemanager,mesosmanager}, --component
{controller,analyticsdb,analytics,agent,lb,kubemanager,mesosmanager}
                                Component[s] to be configured
-F, --force                       Whether to apply config forcibly
-t TAGS, --tags TAGS             comma separated list of tags to runspecific set of
                                ansible code

```

## Updating and Syncing Service Configurations Within the Container

You can update service configurations by editing the appropriate container configuration file and then syncing.

While starting the container, the `contrailctl` configurations are provided under `/etc/contrailctl`. During startup, `contrailctl config sync` runs to synchronize the configurations to the internal services.

If a user wants to add or change configurations, the user can edit the appropriate configuration file in `/etc/contrailctl/` and then manually run `contrailctl config sync` on that specific container.

Using `contrailctl config sync` synchronizes the entire configuration from the master configuration file in `/etc/contrailctl` to the service configurations within the container.

### Syntax and Usage: `config sync`

```
contrailctl config sync [section] [param] [-f|--force]
```

- Use the options `section` and `parameter` to restrict the data to be synced to a specific section and parameter.
- Use the optional `force` to perform an Ansible run, even if there is no configuration change to be synced.

### Example: `config sync`

In this example, the user wants to add a configuration `"foo=bar"` to the controller container after the container is started.

The following example shows the procedure to sync a configuration change within the controller container.

1. The user edits the `/etc/contrailctl/controller.conf` to add the desired configuration changes within the node that holds the container.
2. The user syncs the change to the services running within the container.

```
$ docker exec <my controller> config sync -c controller -v
```

## RELATED DOCUMENTATION

| [Introduction to Containerized Contrail Modules](#)

# Contrail Global Controller

## IN THIS SECTION

- [Resource Identifier Management | 50](#)
- [Multiple Location Resource Provisioning | 51](#)

Starting with Release 3.1, Contrail provides support for a global controller. The global controller feature provides a seamless controller experience across multiple regions in a cloud environment by helping manage multiple OpenStack installations, each having its own Keystone, Neutron, Nova and so on. High availability is provided by using separate failure domains by region.

To handle the resource burdens when connecting and configuring servers and virtual machines over multiple, different regions, the global controller has the following main responsibilities:

## Resource Identifier Management

The global controller uses centralized resource ID management to manage multiple types of identifiers (IDs), identifying such things as route targets, virtual networks, security groups, and so on.

The Contrail global controller can interconnect virtual networks (VNs) residing in different data centers using BGP VPN technology. BGP VPN recognizes virtual private networks (VPNs) by using route target identifiers. A virtual network ID is used to identify the same virtual networks in different data centers, to prevent looping in service chains. Security group IDs identify the same security group over multiple data centers, so that the same security group policies can be used. It is important to use the same security group over multiple regions to allow traffic from all routes in the same virtual networks.

The global controller needs to manage all of the identifiers when interconnecting multiple data centers.

## Multiple Location Resource Provisioning

There are many cases in which the same resource, such as policy or services, needs to exist in multiple data centers. For example, there might be a security policy to apply a firewall for any traffic for an application server network that exists in multiple locations. Each location needs to have the same virtual network, network policy, and firewalls. The Contrail global controller automates this process.

### Requirements, Assumptions, and Constraints

The following are requirements, assumptions, and constraints for implementing the Contrail global controller:

- Each data center has different regions with OpenStack with Contrail.
- Each region that is managed under the same OpenStack Keystone or Keystone data must be replicated with multiple data centers.
- The global controller has a secure API connection for each OpenStack with Contrail region.
- Each Contrail controller needs peering by eBGP or iBGP; eBGP is recommended.
- Each OpenStack Keystone has an administrator account for the global controller. The account must be authorized to manage resources in each region.

### Platform Support

The following are the platform requirements for the Contrail global controller:

- OpenStack Liberty
- Ubuntu 14.04.4
- Contrail Release 3.1 or greater

### Installation

The global controller is a new feature starting with Contrail Release 3.1. The installation instructions can be found in the following location:

<https://nati.gitbooks.io/contrail-global-controller/content/doc/installation.html>

## Role and Resource-Based Access Control

### IN THIS SECTION

- [Contrail Role and Resource-Based Access \(RBAC\) Overview | 52](#)
- [API-Level Access Control | 52](#)
- [Object Level Access Control | 54](#)
- [Configuration | 54](#)
- [Utilities | 56](#)
- [Upgrading from Previous Releases | 58](#)
- [Configuring RBAC Using the Contrail User Interface | 58](#)
- [RBAC Resources | 61](#)

### Contrail Role and Resource-Based Access (RBAC) Overview

Contrail Release 3.0 and later provides role and resource-based access control (RBAC) with API operation-level access control.

The RBAC implementation relies on user credentials obtained from Keystone from a token present in an API request. Credentials include user, role, tenant, and domain information.

API-level access is controlled by a list of rules. The attachment points for the rules include `global-system-config`, `domain`, and `project`. Resource-level access is controlled by permissions embedded in the object.

### API-Level Access Control

If the RBAC feature is enabled, the API server requires a valid token to be present in the `X-Auth-Token` of any incoming request. The API server trades the token for user credentials (role, domain, project, and so on) from Keystone.

If a token is missing or is invalid, an HTTP error 401 is returned.

The `api-access-list` object holds access rules of the following form:

```
<object, field> => list of <role:CRUD>
```

Where:

**object** An API resource such as network or subnet.

**field** Any property or reference within the resource. The `field` option can be multilevel, for example, `network.ipam.host-routes` can be used to identify multiple levels. The `field` is optional, so in its absence, the create, read, update, and delete (CRUD) operation refers to the entire resource.

**role** The Keystone role name.

Each rule also specifies the list of roles and their corresponding permissions as a subset of the CRUD operations.

### Example: ACL RBAC Object

The following is an example access control list (ACL) object for a project in which the admin and any users with the `Development` role can perform CRUD operations on the network in a project. However, only the `admin` role can perform CRUD operations for policy and IP address management (IPAM) inside a network.

```
<virtual-network, network-policy> => admin:CRUD

<virtual-network, network-ipam> => admin:CRUD

<virtual-network, *> => admin:CRUD, Development:CRUD
```

### Rule Sets and ACL Objects

The following are the features of rule sets for access control objects in Contrail.

- The rule set for validation is the union of rules from the ACL attached to:
  - User project
  - User domain
  - Default domain

It is possible for the project or domain access object to be empty.

- Access is only granted if a rule in the combined rule set allows access.
- There is no explicit deny rule.
- An ACL object can be shared within a domain. Therefore, multiple projects can point to the same ACL object. You can make an ACL object the default.

## Object Level Access Control

The `perms2` permission property of an object allows fine-grained access control per resource.

The `perms2` property has the following fields:

- owner** This field is populated at the time of creation with the tenant UUID value extracted from the token.
- share list** The share list gets built when the object is selected for sharing with other users. It is a list of tuples with which the object is shared.

The `permission` field has the following options:

- R—Read object
- W—Create or update object
- X—Link (refer to) object

Access is allowed as follows:

- If the user is the owner and permissions allow (`rwX`)
- Or if the user tenant is in a shared list and permissions allow
- Or if world access is allowed

## Configuration

This section describes the parameters used in Contrail RBAC.

### Parameter: `aaa-mode`

RBAC is controlled by a parameter named `aaa-mode`. This parameter is used in place of the multi-tenancy parameter of previous releases.

The `aaa-mode` can be set to the following values:

- `no-auth`—No authentication is performed and full access is granted to all.
- `cloud-admin`—Authentication is performed and only the admin role has access.
- `rbac`—Authentication is performed and access is granted based on role.

**NOTE:** The `multi_tenancy` parameter is deprecated, starting with Contrail 3.0. The parameter should be removed from the configuration. Instead, use the `aaa_mode` parameter for RBAC to take effect.

If the `multi_tenancy` parameter is not removed, the `aaa-mode` setting is ignored.

## Parameter: `cloud_admin_role`

A user who is assigned the `cloud_admin_role` has full access to everything.

This role name is configured with the `cloud_admin_role` parameter in the API server. The default setting for the parameter is `admin`. This role must be configured in Keystone to change the default value.

If a user has the `cloud_admin_role` in one tenant, and the user has a role in other tenants, then the `cloud_admin_role` role must be included in the other tenants. A user with the `cloud_admin_role` doesn't need to have a role in all tenants, however, if that user has any role in another tenant, that tenant must include the `cloud_admin_role`.

## Configuration Files with Cloud Admin Credentials

The following configuration files contain `cloud_admin_role` credentials:

- `/etc/contrail/contrail-keystone-auth.conf`
- `/etc/neutron/plugins/opencontrail/ContrailPlugin.ini`
- `/etc/contrail/contrail-webui-userauth.js`

## Changing Cloud Admin Configuration Files

Modify the cloud admin credential files if the `cloud_admin_role` role is changed.

1. Change the configuration files with the new information.

2. Restart the following:

- API server  
`service supervisor-config restart`
- Neutron server  
`service neutron-server restart`
- WebUI

```
service supervisor-webui restart
```

## Global Read-Only Role

You can configure a global read-only role (`global_read_only_role`).

A `global_read_only_role` allows read-only access to all Contrail resources. The `global_read_only_role` must be configured in Keystone. The default `global_read_only_role` is not set to any value.

A `global_read_only_role` user can use the Contrail Web Ui to view the global configuration of Contrail default settings.

## Setting the Global Read-Only Role

To set the global read-only role:

1. The `cloud_admin` user sets the `global_read_only_role` in the Contrail API:

```
/etc/contrail/contrail-api.conf

global_read_only_role = <new-admin-read-role>
```

2. Restart the `contrail-api` service:

```
service contrail-api restart
```

## Parameter Changes in `/etc/neutron/api-paste.ini`

Contrail RBAC operation is based upon a user token received in the `X-Auth-Token` header in API requests. The following change must be made in `/etc/neutron/api-paste.ini` to force Neutron to pass the user token in requests to the Contrail API server:

```
keystone = user_token request_id catch_errors ....
...
...
[filter:user_token]
paste.filter_factory =
neutron_plugin_contrail.plugins.opencontrail.neutron_middleware:token_factory
```

## Utilities

This section describes the utilities available for Contrail RBAC.



## Utility: rbacutil.py

Use `rbacutil.py` to manage `api-access-list` rules. It allows adding, removing, and viewing of rules.

### Read RBAC rule-set using UUID or FQN

To read an RBAC rule-set using FQN domain/project:

```
python /opt/contrail/utils/rbacutil.py --uuid '$ABC123' --op read
python /opt/contrail/utils/rbacutil.py --name 'default-domain:default-api-access-list' --op read
```

### Create RBAC rule-set using FQN domain/project

To create the RBAC rule-set, using UUID or FQN:

```
python /opt/contrail/utils/rbacutil.py --fq_name 'default-domain:api-access-list' --op create
```

### Delete RBAC group using FQN or UUID

To delete an RBAC group using FQN or UUID:

```
python /opt/contrail/utils/rbacutil.py --name 'default-domain:api-access-list' --op delete
python /opt/contrail/utils/rbacutil.py --uuid $ABC123 --op delete
```

### Add rule to existing RBAC group

To add a rule to an existing RBAC group:

```
python /opt/contrail/utils/rbacutil.py --uuid <uuid> --rule "* Member:R" --op add-rule
python /opt/contrail/utils/rbacutil.py --uuid <uuid> --rule "useragent-kv *:CRUD" --op add-rule
```

## Delete rule from RBAC group - specify rule number or exact rule

To delete a rule from an RBAC group, and specify a rule number or exact rule:

```
python /opt/contrail/utils/rbacutil.py --uuid <uuid> --rule 2 --op del-rule
python /opt/contrail/utils/rbacutil.py --uuid <uuid> --rule "useragent-kv *:CRUD" --op del-rule
```

### Utility: chmod2.py

The utility `chmod2.py` enables updating object permissions, including:

- Ownership—Specify a new owner tenant UUID.
- Enable/disable sharing with other tenants—Specify the tenants.
- Enable/disable sharing with world—Specify permissions.

## Upgrading from Previous Releases

The `multi_tenancy` parameter is deprecated, starting with Contrail 3.1. The parameter should be removed from the configuration. Instead, use the `aaa_mode` parameter for RBAC to take effect.

If the `multi_tenancy` parameter is not removed, the `aaa-mode` setting is ignored.

## Configuring RBAC Using the Contrail User Interface

To use the Contrail UI with RBAC:

1. Set the `aaa_mode` to `no_auth`.

```
/etc/contrail/contrail-analytics-api.conf
```

```
aaa_mode = no-auth
```

2. Restart the `analytics-api` service.

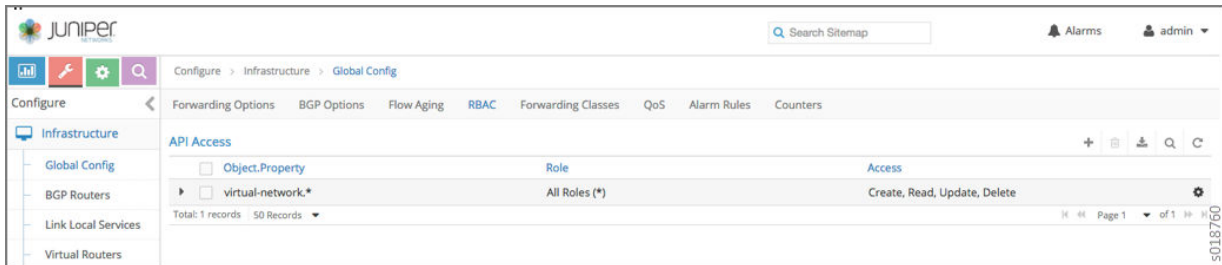
```
service contrail-analytics-api restart
```

You can use the Contrail UI to configure RBAC at both the API level and the object level. API level access control can be configured at the global, domain, and project levels. Object level access is available from most of the create or edit screens in the Contrail UI.

## Configuring RBAC at the Global Level

To configure RBAC at the global level, navigate to **Configure > Infrastructure > Global Config > RBAC**, see [Figure 8 on page 59](#).

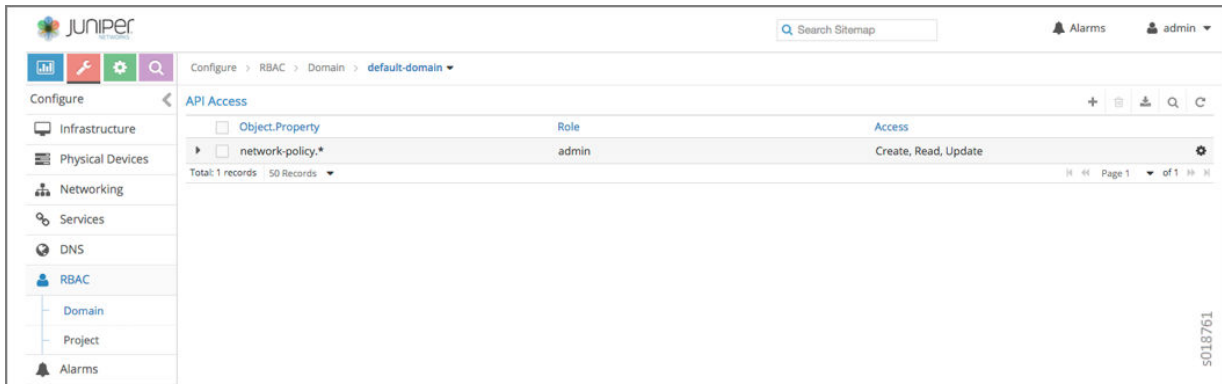
**Figure 8: RBAC Global Level**



## Configuring RBAC at the Domain Level

To configure RBAC at the domain level, navigate to **Configure > RBAC > Domain**, see [Figure 9 on page 59](#).

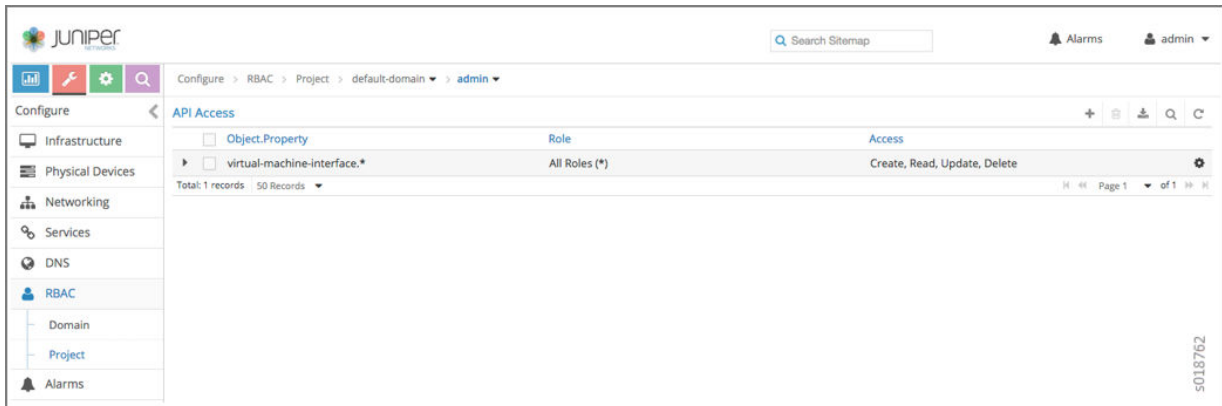
**Figure 9: RBAC Domain Level**



## Configuring RBAC at the Project Level

To configure RBAC at the project level, navigate to **Configure > RBAC > Project**, see [Figure 10 on page 60](#).

Figure 10: RBAC Project Level



## Configuring RBAC Details

Configuring RBAC is similar at all of the levels. To add or edit an API access list, navigate to the global, domain, or project page, then click the plus (+) icon to add a list, or click the gear icon to select from Edit, Insert After, or Delete, see [Figure 11 on page 60](#).

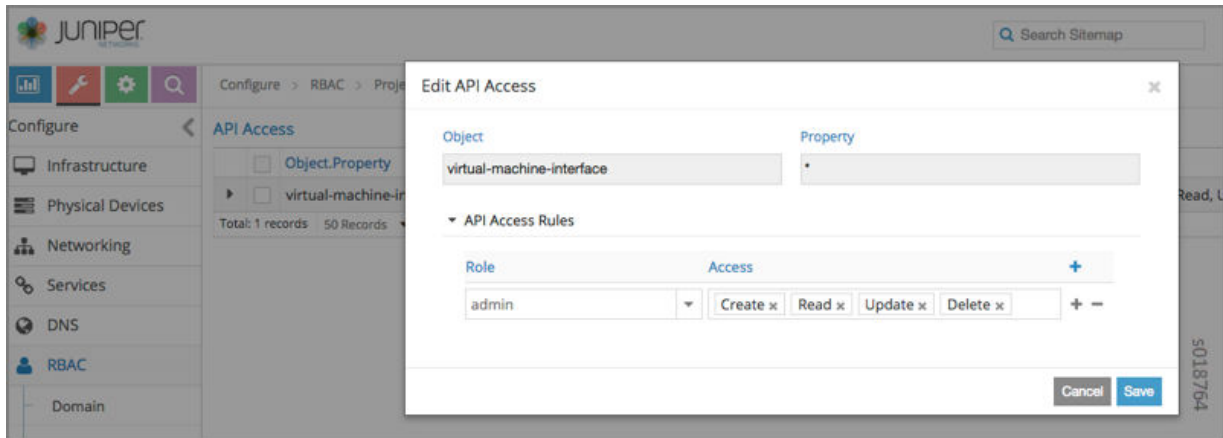
Figure 11: RBAC Details API Access



## Creating or Editing API Level Access

Clicking create, edit, or insert after activates the Edit API Access popup window, where you enter the details for the API Access Rules. Enter the user type in the Role field, and use the + icon in the Access field to enter the types of access allowed for the role, including, Create, Read, Update, Delete, and so on, see [Figure 12 on page 61](#).

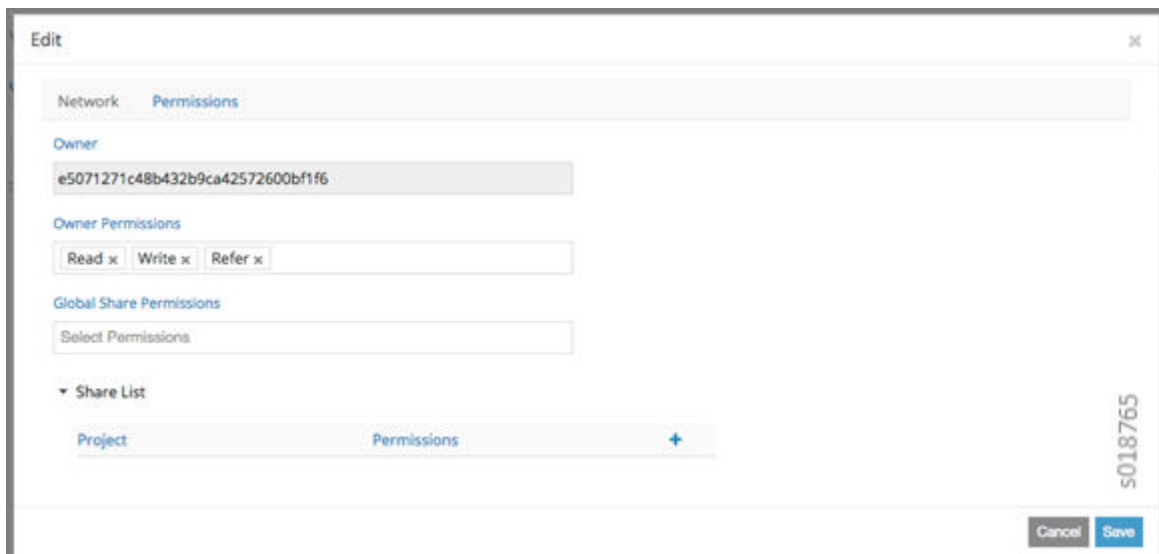
Figure 12: Edit API Access



### Creating or Editing Object Level Access

You can configure fine-grained access control by resource. A **Permissions** tab is available on all create or edit popups for resources. Use the **Permissions** popup to configure owner permissions and global share permissions. You can also share the resource to other tenants by configuring it in the **Share List**, see [Figure 13 on page 61](#).

Figure 13: Edit Object Level Access



### RBAC Resources

Refer to the *OpenStack Administrator Guide* for additional information about RBAC:

- Identity API protection with role-based access control (RBAC)

# Installation and Configuration Scenarios

## IN THIS CHAPTER

- [Setting Up and Using a Simple Virtual Gateway with Contrail 4.0 | 63](#)
- [Simple Underlay Connectivity without Gateway | 73](#)
- [Configuring MD5 Authentication for BGP Sessions | 76](#)
- [Configuring the Data Plane Development Kit \(DPDK\) Integrated with Contrail vRouter | 78](#)
- [Configuring Single Root I/O Virtualization \(SR-IOV\) | 81](#)
- [Provisioning DPDK SRIOV with Server Manager | 86](#)
- [Configuring Virtual Networks for Hub-and-Spoke Topology | 89](#)
- [Configuring Transport Layer Security-Based XMPP in Contrail | 96](#)
- [Configuring Graceful Restart and Long-lived Graceful Restart | 99](#)

## Setting Up and Using a Simple Virtual Gateway with Contrail 4.0

### IN THIS SECTION

- [Introduction to the Simple Gateway | 64](#)
- [How the Simple Gateway Works | 64](#)
- [Setup Without Simple Gateway | 64](#)
- [Setup With a Simple Gateway | 65](#)
- [Simple Gateway Configuration Features | 66](#)
- [Packet Flows with the Simple Gateway | 67](#)
- [Packet Flow Process From the Virtual Network to the Public Network | 68](#)
- [Packet Flow Process From the Public Network to the Virtual Network | 68](#)
- [Methods for Configuring the Simple Gateway | 69](#)
- [Using the vRouter Configuration File to Configure the Simple Gateway | 69](#)

- [Using Thrift Messages to Dynamically Configure the Simple Gateway | 69](#)
- [Common Issues with Simple Gateway Configuration | 72](#)

## Introduction to the Simple Gateway

Every virtual network has a routing instance associated with it. The routing instance defines the network connectivity for the virtual machines in the virtual network. By default, the routing instance contains routes only for virtual machines spawned within the virtual network. Connectivity between virtual networks is controlled by defining network policies.

The public network is the IP fabric or the external networks across the IP fabric. The virtual networks do not have access to the public network, and a gateway is used to provide connectivity to the public network from a virtual network. In traditional deployments, a routing device such as a Juniper Networks MX Series router can act as a gateway.

The simple virtual gateway for Contrail is a restricted implementation of a gateway that can be used for experimental purposes, only. The simple gateway provides the Contrail virtual networks with access to the public network, and is represented as `vgw`.

The simple gateway is valid **ONLY** for a kernel vrouter, and *cannot* be used with any other flavor of vrouter, such as DPDK, SR-IOV, or the like. The simple gateway *cannot* be used in a production environment, it is for experimental uses only.

## How the Simple Gateway Works

The following sections illustrate how the simple gateway works, first, by showing a virtual network setup with no simple gateway, then illustrating the same setup with a simple gateway configured.

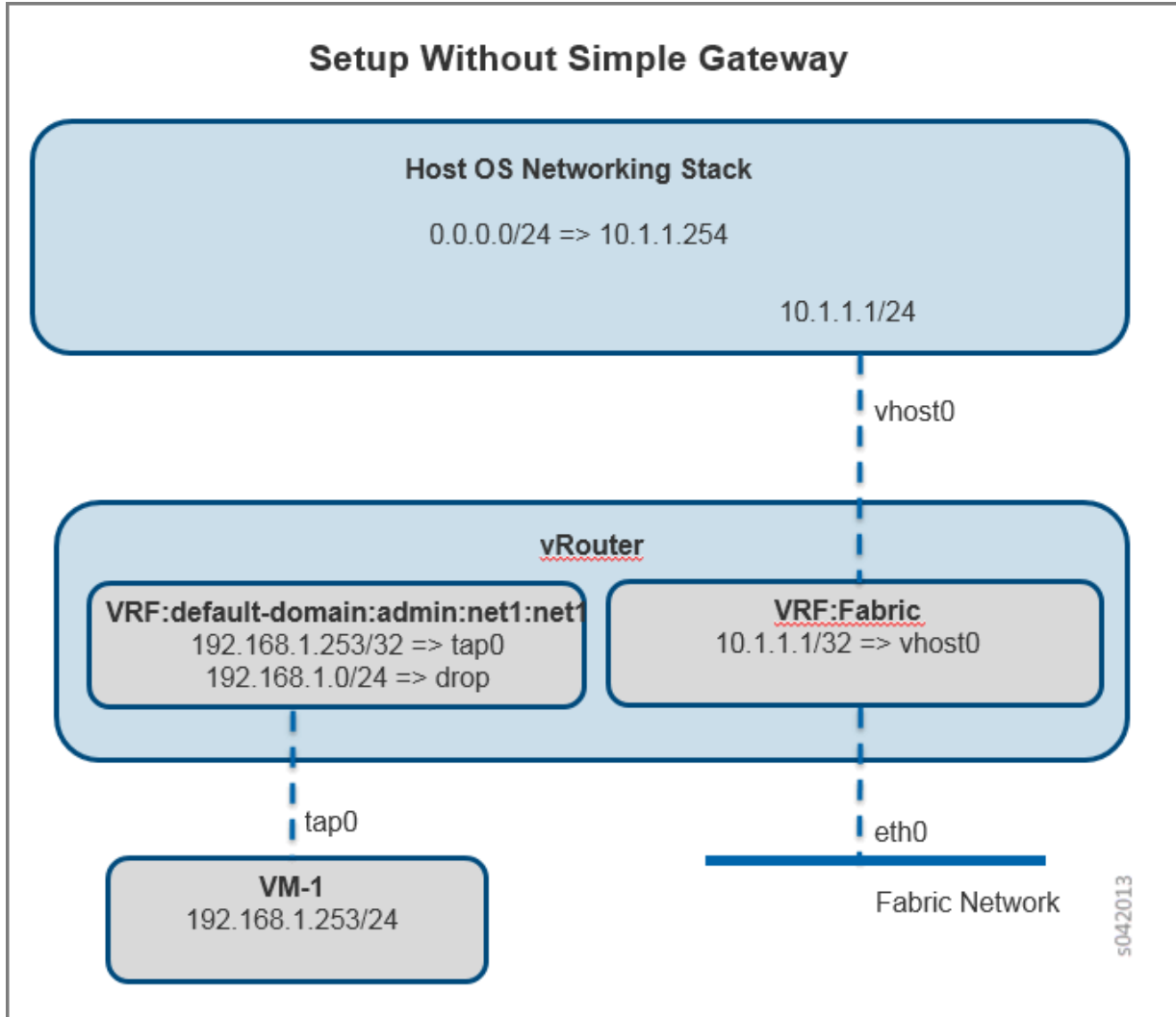
### Setup Without Simple Gateway

The following shows a virtual network setup when the simple gateway is not configured.

- A virtual network, `default-domain:admin:net1`, is configured with the subnet `192.168.1.0/24`.
- The routing instance `default-domain:admin:net1:net1` is associated with the `default-domain:admin:net1` virtual network .
- A virtual machine with the `192.168.1.253` IP address is spawned in `net1`.
- A virtual machine is spawned on compute server 1.
- An interface, `vhost0`, is in the host OS of server 1 and is assigned the `10.1.1.1/24` IP address.
- The `vhost0` interface is added to the vRouter in the routing instance fabric.



- The simple gateway is not configured.



### Setup With a Simple Gateway

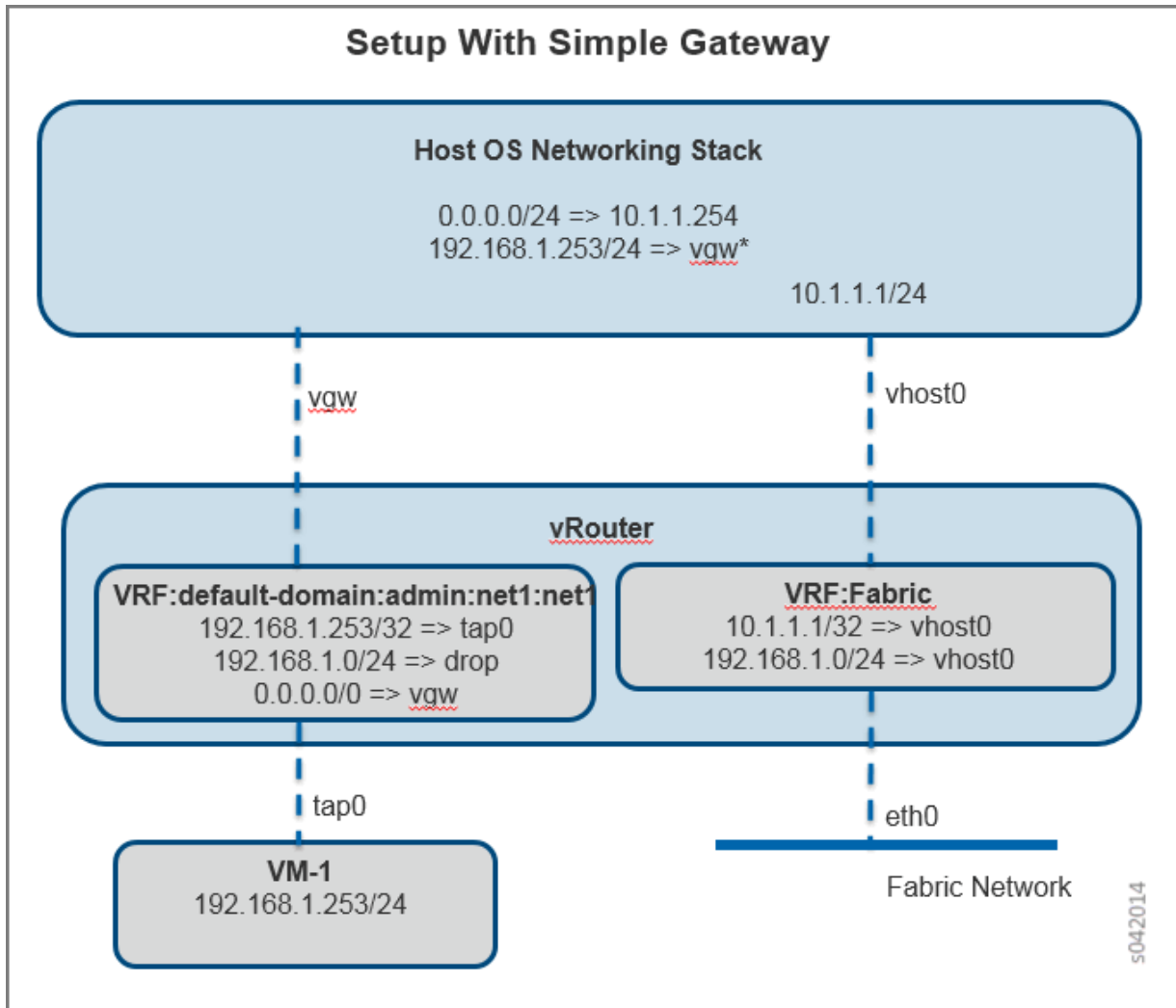
Figure 14 on page 66 shows a virtual network setup with the simple gateway configured for the `default-domain:admin:net1` virtual network.

The simple gateway configuration uses a gateway interface (`vgw`) to provide connectivity between the fabric routing instance and the `default-domain:admin:net1` virtual network.

Figure 14 on page 66 shows the packet flows between the fabric VRF and the `default-domain:admin:net1` virtual network.

In the diagram, routes marked with (\*) are added by the simple gateway feature.

Figure 14: Virtual Network Setup With a Simple Gateway



### Simple Gateway Configuration Features

The simple gateway configuration has the following features.

- The simple gateway is configured for the default-domain:admin:net1 virtual network.
  - The `vgw` gateway interface provides connectivity between the routing instance default-domain:admin:net1:net1 and the fabric.
  - An IP address is not configured for the `vgw` gateway interface.
- The host OS is configured with the following:
  - Two INET interfaces are added to the host OS: `vgw` and `vhost0`

- The host OS is not aware of the routing instances, so the `vgw` and `vhost0` interfaces are part of the same routing instance in the host OS.
- The simple gateway adds the `192.168.1.0/24` route, pointing to the `vgw` interface, and that setup is added to the host OS. This route ensures that any packet destined to the virtual machine is sent to the vRouter on the `vgw` interface.
- The vRouter is configured with the following:
  - The routing instance named `Fabric` is created for the fabric network.
  - The interface `vhost0` is added to the routing instance `Fabric`.
  - The interface `eth0`, which is connected to the fabric network, is added to the routing instance named `Fabric`.
  - The simple gateway adds the `192.168.1.0/24` route to the `vhost0` interface. Consequently, packets destined to the `default-domain:admin:net1` virtual network are sent to the host OS.
- The `default-domain:admin:net1:net1` routing instance is created for the `default-domain:admin:net1` virtual network.
  - The `vgw` interface is added to the `default-domain:admin:net1:net1` routing instance.
  - The simple gateway adds a default route (`0.0.0.0/0`) that points to the `vgw` interface. Packets in the `default-domain:admin:net1:net` routing instance that match this route are sent to the host OS on the `vgw` interface. The host OS routes the packets to the fabric network over the `vhost0` interface.

### Simple Gateway Restrictions

The following are restrictions of the simple gateway:

- A single compute node can have the simple gateway configured for multiple virtual networks, however, there cannot be overlapping subnets. The host OS does not support routing instances. Therefore, all gateway interfaces in the host OS are in the same routing instance and the subnets in the virtual networks must not overlap.
- Each virtual network can have a single simple gateway interface. ECMP is not supported.

### Packet Flows with the Simple Gateway

The following sections describe the packet flow process when the simple gateway is configured on a Contrail system.

First, the packet flow process from the virtual network to the public network is described. Next, the packet flow process from the public network to the virtual network is described.

## Packet Flow Process From the Virtual Network to the Public Network

The following describes the procedure used to move a packet from the virtual network (net1) to the public network.

1. A packet with a source IP address of 192.168.1.253 and a destination IP address of 10.1.1.253 comes from a virtual machine and is received by the vRouter on the tap0 interface.
2. The tap0 interface is in the default-domain:admin:net1:net1 routing instance.
3. The route lookup for 10.1.1.253 in the default-domain:admin:net1:net1 routing instance finds the default route pointing to the tap interface named vgw.
4. The vRouter transmits the packet toward the vgw interface and it is received by the networking stack of the host OS.
5. The host OS performs forwarding based on its routing table and forwards the packet on the vhost0 interface.
6. Packets transmitted on the vhost0 interface are received by the vRouter.
7. The vhost0 interface is added to the Fabric routing instance.
8. The routing table for 10.1.1.253 in the Fabric routing instance indicates that the packet is to be transmitted on the eth0 interface.
9. The vRouter transmits the packet on the eth0 interface.
10. The 10.1.1.253 host on the Fabric routing instance receives the packet.

## Packet Flow Process From the Public Network to the Virtual Network

The following describes the procedure used to move a packet from the public network to the virtual network (net1).

1. A packet with a source IP address of 10.1.1.253 and a destination IP address of 192.168.1.253 coming from the public network is received on the eth0 interface.
2. The tap0 interface is in the default-domain:admin:net1:net1 routing instance.
3. The vRouter receives the packet from the eth0 interface in the Fabric routing instance.
4. The route lookup for 192.168.1.253 in the Fabric routing instance points to the interface vhost0.
5. The vRouter transmits the packet on the vhost0 interface and it is received by the networking stack of the host OS.
6. The host OS performs forwarding according to its routing table and forwards the packet on the vgw interface.
7. The vRouter receives the packet on the vgw interface into the routing instance default-domain:admin:net1:net1.
8. The route lookup for 192.168.1.253 in the default-domain:admin:net1:net1 routing instance points to the tap0 interface.

9. The vRouter transmits the packet on the tap0 interface.
10. The virtual machine receives the packet destined to 192.168.1.253.

## Methods for Configuring the Simple Gateway

There are different methods that can be used to configure the simple gateway. Each of the methods is described in the following sections.

### Using the vRouter Configuration File to Configure the Simple Gateway

Another way to enable a simple gateway is to configure one or more vgw interfaces within the `contrail-vrouter-agent.conf` file.

Any changes made in this file for simple gateway configuration are implemented upon the next restart of the vRouter agent. To configure the simple gateway in the `contrail-vrouter-agent.conf` file, each simple gateway interface uses the following parameters:

- `interface=vgwxx`— Simple gateway interface name.
- `routing_instance=default-domain:admin:public xx:public xx`— Name of the routing instance for which the simple gateway is being configured.
- `ip_block=1.1.1.0/24`— List of the subnet addresses allocated for the virtual network. Routes within this subnet are added to both the host OS and routing instance for the fabric instance. Represent multiple subnets in the list by separating each with a space.
- `routes=10.10.10.1/24 11.11.11.1/24`— List of subnets in the public network that are reachable from the virtual network. Routes within this subnet are added to the routing instance configured for the vgw interface. Represent multiple subnets in the list by separating each with a space.

### Using Thrift Messages to Dynamically Configure the Simple Gateway

#### IN THIS SECTION

- [How to Dynamically Create a Virtual Gateway | 70](#)
- [How to Dynamically Delete a Virtual Gateway | 71](#)
- [Using Devstack to Configure the Simple Gateway | 72](#)

Another way to configure the simple gateway is to dynamically send create and delete thrift messages to the vrouter agent.

Starting with Contrail Release 1.10 and greater, the following thrift messages are available:

- `AddVirtualGateway`—add a virtual gateway
- `DeleteVirtualGateway` —delete a virtual gateway
- `ConnectForVirtualGateway` —allows audit of the virtual gateway configuration by stateful clients. Upon a new `ConnectForVirtualGateway` request, one minute is allowed for the configuration to be redone. Any older virtual gateway configuration remaining after this time is deleted.

### How to Dynamically Create a Virtual Gateway

To dynamically create a simple virtual gateway, you run a script on the compute node where the virtual gateway is being created.

When run, the script does the following:

1. Enables forwarding on the node.
2. Creates the required interface.
3. Adds the interface to the vRouter.
4. Adds required routes to the host OS.
5. Sends the `AddVirtualGateway` thrift message to the vRouter agent telling it to create the virtual gateway.

### Example: Dynamically Create a Virtual Gateway

The following procedure dynamically creates the `vgw1` interface, with `20.30.40.0/24` and `30.40.50.0/24` subnets in the `default-domain:admin:vn1:vn1` VRF.

1. Set the `PYTHONPATH` variable to the location of the `InstanceService.py` and `types.pyfiles`, for example:

```
export PYTHONPATH=/usr/lib/python2.7/dist-packages/nova_contrail_vif/gen_py/instance_service
```

```
export PYTHONPATH=/usr/lib/python2.6/site-packages/contrail_vrouter_api/gen_py/instance_service
```

2. Run the virtual gateway provision command with the `oper create` option.

Use the `subnets` option to specify the subnets defined for virtual network `vn1`.

Use the `routes` option to specify the routes in the public network that are injected into `vn1`.

In the following example, the virtual machines in `vn1` can access subnets `8.8.8.0/24` and `9.9.9.0/24` in the public network:

```
python /opt/contrail/utils/provision_vgw_interface.py --oper create --interface vgw1 --subnets 20.30.40.0/24
30.40.50.0/24 --routes 8.8.8.0/24 9.9.9.0/24 --vrf default-domain:admin:vn1:vn1
```

## How to Dynamically Delete a Virtual Gateway

To dynamically delete a virtual gateway, run a script on the compute node where the virtual gateway is.

When run, the script does the following:

1. Sends the `DeleteVirtualGateway` thrift message to the vRouter agent. Tell it to delete the virtual gateway.
2. Deletes the virtual gateway interface from the vRouter.
3. Deletes the virtual gateway routes that were added in the host OS when the virtual gateway was created.

## Example: Dynamically Create a Virtual Gateway

The following procedure dynamically deletes the `vgw1` interface. It also deletes the `20.30.40.0/24` and `30.40.50.0/24` subnets in the `default-domain:admin:vn1:vn1` VRF .

1. Set the `PYTHONPATH` variable to the location of the `InstanceService.py` and `types.py` files, for example:

```
export PYTHONPATH=/usr/lib/python2.7/dist-packages/nova_contrail_vif/gen_py/instance_service
export PYTHONPATH=/usr/lib/python2.6/site-packages/contrail_vrouter_api/gen_py/instance_service
```

2. Run the virtual gateway provision command with the `oper delete` option.

```
python /opt/contrail/utils/provision_vgw_interface.py --oper delete --interface vgw1 --subnets 20.30.40.0/24
30.40.50.0/24 --routes 8.8.8.0/24 9.9.9.0/24
```

3. (optional) If you are using a stateful client, send the `ConnectForVirtualGateway` thrift message to the vRouter agent when the client starts.

**NOTE:** If the vRouter agent restarts or if the compute node reboots, it is expected that the client reconfigures again.

## Using Devstack to Configure the Simple Gateway

Another way to configure the simple gateway is to set configuration parameters in the devstack localrc file.

The following parameters are available:

- `CONTRAIL_VGW_PUBLIC_NETWORK` — The name of the routing instance for which the simple gateway is being configured.
- `CONTRAIL_VGW_PUBLIC_SUBNET` — A list of subnet addresses allocated for the virtual network. Routes containing these addresses are added to both the host OS and the routing instance for the fabric. List multiple subnets by separating each with a space.
- `CONTRAIL_VGW_INTERFACE` — A list of subnets in the public network that are reachable from the virtual network. Routes containing these subnets are added to the routing instance configured for the simple gateway. List multiple subnets by separating each with a space.

This method can only add the default route `0.0.0.0/0` into the routing instance specified in the `CONTRAIL_VGW_PUBLIC_NETWORK` option.

### Example: Devstack Configuration for Simple Gateway

Add the following lines in the localrc file for stack.sh:

```
CONTRAIL_VGW_INTERFACE=vgw1

CONTRAIL_VGW_PUBLIC_SUBNET=192.168.1.0/24

CONTRAIL_VGW_PUBLIC_NETWORK=default-domain:admin:net1:net1
```

**NOTE:** This method can only add the `0.0.0.0/0` default route into the routing instance specified in the `CONTRAIL_VGW_PUBLIC_NETWORK` option.

## Common Issues with Simple Gateway Configuration

The following are common problems you might encounter when configuring a simple gateway.

- Packets from the external network are not reaching the compute node.

The devices in the fabric network must be configured with static routes for the IP addresses defined in the public subnet (192.168.1.0/24 in the example) to reach the compute node that is running as a simple gateway.



- Packets are reaching the compute node, but are not routed from the host OS to the virtual machine.

Check to see if the `firewall_driver` in the `/etc/nova/nova.conf` file is set to `nova.virt.libvirt.firewall.IptablesFirewallDriver`, which enables IPTables. IPTables can discard packets.

Resolutions include disabling IPTables during runtime or setting the `firewall_driver` in the `localrc` file: `LIBVIRT_FIREWALL_DRIVER=nova.virt.firewall.NoopFirewallDriver`

## Simple Underlay Connectivity without Gateway

### IN THIS SECTION

- [Simple Routing of Packets Without a Gateway | 73](#)
- [Supported Use Cases | 74](#)
- [Implementation: Routing Instances | 74](#)
- [Implementation | 76](#)

### Simple Routing of Packets Without a Gateway

For simple enterprise use cases and public cloud environments, it is possible to directly route packets using the IP fabric network without using an SDN gateway.

The primary use for Contrail in this mode is to manage distributed security policy for workloads or bare metal servers.

The following features can be enabled when using this method:

- Network policy support for IP fabric
- Security groups for VMs and containers on IP fabric
- Security groups for vhost0 interface, to protect compute node or bare metal server applications
- Support for service chaining, if policy dictates that traffic goes through a service chain.

## Supported Use Cases

Starting with Contrail 4.1, the IP fabric network present in the default project can be marked for IP fabric based forwarding without tunneling. When two virtual networks with this type of configuration communicate, traffic will be forwarded directly using the underlay.

The following use cases for no SDN gateway are supported.

- Virtual networks with an IP subnet that is a subset of the IP fabric network or another subnet, and are using the IP fabric network as the provider network.

VMs and containers from this type of VNs communicate within their VNs, with IP fabric VN, and with other VNs that also have IP fabric as their provider network based on configured policy, using only the underlay, with no tunneling.

- Virtual networks with IP fabric VN as their provider network, communicating with other VNs that do not have any provider network based on policy configured, using overlay with tunneling.
- Vhost communication , with other compute vhosts and with VMs and containers in the IP fabric network or other VNs with IP fabric network as the provider network based on policy configured, using underlay and no tunneling.
- Vhost communication with VMs in any virtual network based on policy configuration, using overlay with tunneling.

## Implementation: Routing Instances

To implement the simple underlay connectivity with no SDN gateway, the IP fabric network has two routing instance associations:

- A default routing instance, `ip-fabric:default`, which is used for all forwarding decisions by the data path.
- A new routing instance, `ip-fabric:ip-fabric`, to carry L3VPN routes for endpoints in IP fabric. Network policy and security groups are applied based on these entries.

The IP fabric network can be associated with an IPAM and have its subnets. The IPAM for IP fabric will always use a flat subnet mode, whereby the same subnet can be shared with multiple virtual networks. The IP fabric IPAM has the overall subnet, with other virtual networks using blocks from this subnet.

## IPAM Addressing Schemes

Two IPAM addressing schemes are supported for IP fabric:

- Common subnet mode with a set of subnet prefixes.

- Prefix per vrouter mode. To scale up underlay routing, block allocation per vrouter is supported, whereby address blocks are advertised instead of individual addresses. Every vrouter and compute node gets its own prefix. IP address-to-VMI allocation occurs after the scheduling decision is made for the VM or container. This scheme is supported for K8S and Mesos without restrictions. However, OpenStack requires the address before the scheduling decision, so in this scheme, the user must assign an address and dictate the scheduling decision to use OpenStack.

## Operation

When a VMI is created in the IP fabric network, the vrouter exports an L3VPN route for the VMI in the ip-fabric:ip-fabric routing instance, with the vrouter as the next hop (along with the MPLS label, policy tags, security group tags, and so on). An Inet route is exported in the ip-fabric:default routing instance, with the vrouter as next hop.

Vrouters use the ip-fabric routing instance to apply policy and the default routing instance is used to forward traffic. The control node peers with ToRs and publishes the routes of the vrouter nexthops of the TOR.

It is expected that the ToR propagates these routes to the rest of the underlay network. When using the prefix per vrouter mode, the ToR might also be configured with static routes pointing to the compute nodes, instead of peering with the control node.

Vhost interface is also added in the default routing instance. Policy and security groups can be applied on this interface as well, so that traffic from the applications and services running on the host can be subjected to all policy decisions possible in Contrail.

The IP fabric network is a Layer 3-only network and the vrouter only looks at the routing table for all forwarding decisions.

## ARP Handling

ARP requests in the IP fabric network and in VNs with the IP fabric network as the provider network are handled in the following ways:

- VM-to-VM communication, on the same compute or on different compute nodes— Respective vrouters respond to ARP requests from the VMs with the vrouter's MAC. Agent resolves the ARP for other compute nodes to fill the next hop corresponding to remote VMs.
- Vhost connectivity to VM on the same compute node—Vrouter responds with vhost MAC (its own MAC) for ARP requests from vhost. ARP requests from the VM will be responded with vrouter's MAC.

Each subnet in the networks, IP fabric network or other VNs using IP fabric as the provider network, has a subnet route in the compute host pointing to the vhost interface. There is a Layer 3 route in the fabric

default VRF for each VM, with the next hop pointing to its VMI. Traffic is forwarded to the VM based on this route. The next hop is a Layer 3 interface next hop with the source MAC being the vrouter's MAC.

When the vhost and the VN are using different subnets, an ARP request from the vhost has the VM's IP as the destination IP and the vhost's IP as the source IP. Vrouter responds to an ARP request with the vhost's MAC.

- Vhost connectivity to VM on a different compute node—ARP requests for VMs on a different compute node are flooded on the fabric interface. The compute node hosting the VM has a Layer 3 route for the VM, with the next hop pointing to its VMI. The vrouter on that node responds to the ARP request with its vhost MAC address. The VM's ARP request is always responded to by with vrouter's MAC.
- Vhost connectivity to another compute node—As in the previous example, the ARP request is transmitted on the fabric interface. Other vrouters cross connect the ARP request to their vhost interface because there is not any Layer 3 route pointing to the VMI. The host responds to the ARP request.

## Broadcast and Multicast Traffic

In Contrail 4.1, broadcast or multicast traffic from VMs in the IP fabric network and from VNs having IP fabric network as the provider network is handled in the normal way, using the native routing instance of the interface from which it originates.. DHCP requests from these VMs are served by the vrouter agent.

## Implementation

A virtual network can have a provider network configured using a link from the VN to the IP fabric VN.

A vrouter-specific IP allocation pool can be created. If an instance IP is created with a link to a vrouter and the vrouter is linked with a flat subnet IPAM, then the instance IP is allocated an address from the vrouter-specific allocation pool.

Provisioning will create VMI for vhost interface. Creation of virtual networks with IP fabric forwarding, policy / security group configurations for vhost interface can now be done.

## Configuring MD5 Authentication for BGP Sessions

Contrail supports MD5 authentication for BGP peering based on RFC 2385.

This option allows BGP to protect itself against the introduction of spoofed TCP segments into the connection stream. Both of the BGP peers must be configured with the same MD5 key. Once

configured, each BGP peer adds a 16-byte MD5 digest to the TCP header of every segment that it sends. This digest is produced by applying the MD5 algorithm on various parts of the TCP segment. Upon receiving a signed segment, the receiver validates it by calculating its own digest from the same data (using its own key) and compares the two digests. For valid segments, the comparison is successful since both sides know the key.

The following are ways to enable BGP MD5 authentication and set the keys on the Contrail node.

1. If the md5 key is not included in the provisioning, and the node is already provisioned, you can run the following script with an argument for md5:

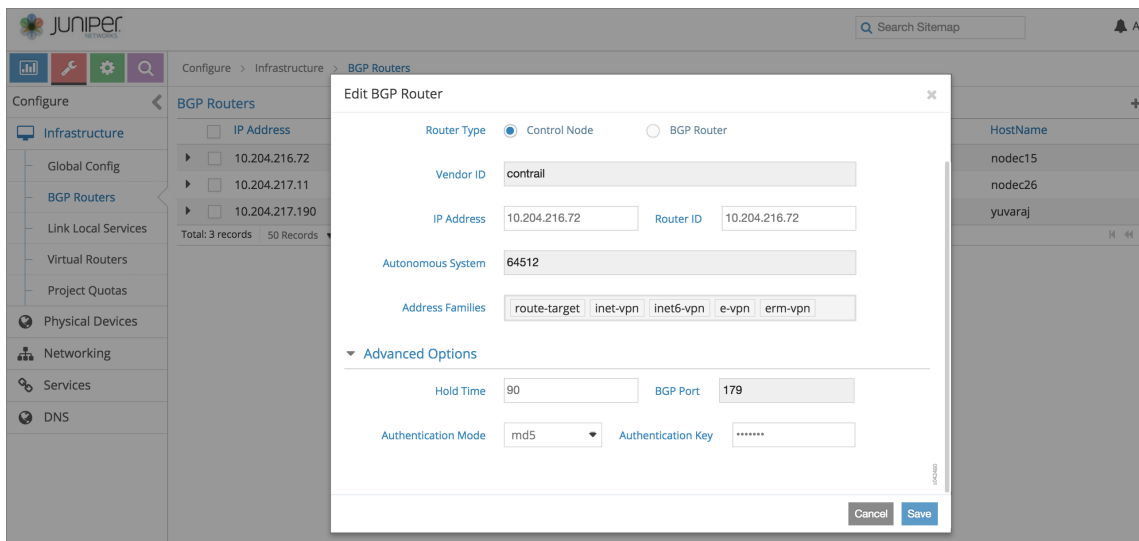
```
contrail-controller/src/config/utils/provision_control.py
```

```
host@<your_node>:/opt/contrail/utils# python provision_control.py --host_name <host_name> --
host_ip <host_ip> --router_asn <asn> --api_server_ip <api_ip> --api_server_port <api_port> --
oper add --md5 "juniper" --admin_user admin --admin_password <password> --admin_tenant_name
admin
```

2. You can also use the web user interface to configure MD5.

- Connect to the node's IP address at port 8080 (<node\_ip>:8080) and select **Configure->Infrastructure->BGP Routers**. As shown in [Figure 15 on page 77](#), a list of BGP peers is displayed.

**Figure 15: Edit BGP Router Window**



- For a BGP peer, click on the gear icon on the right hand side of the peer entry. Then click **Edit**. This displays the Edit BGP Router dialog box.
- Scroll down the window and select **Advanced Options**.
- Configure the MD5 authentication by selecting **Authentication Mode>MD5** and entering the **Authentication Key** value.

## RELATED DOCUMENTATION

*[Creating a Virtual Network with Juniper Networks Contrail](#)*

*[Creating a Virtual Network with OpenStack Contrail](#)*

## Configuring the Data Plane Development Kit (DPDK) Integrated with Contrail vRouter

### IN THIS SECTION

- [DPDK Support in Contrail | 78](#)
- [Preparing the Environment File for Provisioning a Cluster Node with DPDK | 79](#)
- [Creating a Flavor for DPDK in OpenStack Kilo | 80](#)

### DPDK Support in Contrail

Contrail 3.0 and later supports the Data Plane Development Kit (DPDK).

DPDK is an open source set of libraries and drivers for fast packet processing. DPDK enables fast packet processing by allowing network interface cards (NICs) to send direct memory access (DMA) packets directly into an application's address space, allowing the application to poll for packets, and thereby avoiding the overhead of interrupts from the NIC.

Integrating with DPDK allows a Contrail vRouter to process more packets per second than is possible when running as a kernel module.

When using DPDK with Contrail 4.0, one or more Contrail compute nodes are provisioned with DPDK during installation. An entry in the server configuration file specifies which nodes are to be configured to

use the DPDK vRouter mode instead of the regular kernel mode. This allows for a mixed setup, where different nodes use different modes of the vRouter.

Upon installation, when a Contrail compute node is provisioned with DPDK, the server configuration file specifies which physical interface(s) to use, how many CPU cores to use for forwarding packets, and the number of huge pages to allocate for DPDK.

## Preparing the Environment File for Provisioning a Cluster Node with DPDK

The environment file is used at provisioning to specify all of the options necessary for the installation of a Contrail cluster, including whether any node should be configured to use DPDK.

Each node to be configured with the DPDK vRouter must be listed in the provisioning file with a dictionary entry, along with the percentage of memory for DPDK huge pages and the CPUs to be used.

The following are descriptions of the required entries for the server configuration. :

- `huge_pages`—Specify the percentage of host memory to be reserved for the DPDK huge pages. The reserved memory will be used by the vRouter and the Quick Emulator (QEMU) for allocating memory resources for the virtual machines (VMs) spawned on that host.

**NOTE:** The percentage allocated to `huge_pages` should not be too high, because the host Linux kernel also requires memory.

- `coremask`—Specify a CPU affinity mask with which vRouter will run. vRouter will use only the CPUs specified for its threads of execution. These CPU cores will be constantly polling for packets, and they will be displayed as 100% busy in the output of “top”.

Supported formats include:

- Hexadecimal (for example, 0xf)
- Comma-separated list of CPUs (1,2,4...)
- Dash-separated range of CPUs (for example, 1-4)

The server configuration file is configured prior to the installation of Contrail.

Use the standard Contrail installation procedure, and upon completion, your cluster with specified nodes using the DPDK vRouter implementation is ready to use.

## Support for Multiple UIO Drivers

Support is available for optionally specifying the userspace IO (UIO) driver to use in a DPDK-enabled compute node.

Specify UIO in the `dpdk` section of the server configuration file, by using the optional attribute `uio_driver`:

```
{
  "server" : [{
    "parameters" : {
      "provision": {
        "contrail_4": {
          "core_mask": "0x3f",
          "huge_pages": "50",
          "uio_driver" : "igb_uio"
        }
      }
    }
  }
}]
}
```

The supported values for `uio_driver` include:

- `igb_uio`—specify that the `igb_uio` module from the DPDK library should be used.
- `vfiopci`—specify that the `vfiopci` module in the Linux kernel should be used instead of `uio`, which protects memory accesses using the IOMMU when a SR-IOV virtual function is used as the physical interface of vrouter.
- `uio_pci_generic`—specify that the UIO driver that is built into the Linux kernel should be used. This option does not support the use of SR-IOV VFs.

If the `uio_driver` is not specified in the server configuration file, `igb_uio` is used by default.

## Creating a Flavor for DPDK in OpenStack Kilo

OpenStack Kilo has a feature called flavors, which are virtual hardware templates that define sizes for RAM, disk, and so on. Contrail 3.0 and later supports the OpenStack Kilo flavor that specifies that a VM should use huge pages. The use of huge pages is a requirement for using a DPDK vRouter.

Use the following command to add the flavor, where `m1.large` is the name of the flavor. When a VM is created using this flavor, OpenStack ensures that the VM will only be spawned on a compute node that has huge pages enabled.

```
$nova flavor-key m1.large set hw:mem_page_size=large
```

Huge pages are enabled for compute nodes where vRouter is provisioned with DPDK.



If a VM is spawned with a flavor that does not have huge pages enabled, the VM should not be created on a compute node on which vRouter is provisioned with DPDK.

You can use OpenStack availability zones or host aggregates to exclude the hosts where vRouter is provisioned with DPDK.

## RELATED DOCUMENTATION

[Configuring Single Root I/O Virtualization \(SR-IOV\) | 81](#)

<http://www.dpdk.org>

## Configuring Single Root I/O Virtualization (SR-IOV)

### IN THIS SECTION

- [Overview: Configuring SR-IOV | 81](#)
- [Configuring SR-IOV Using JSON | 82](#)
- [Preparing the testbed.py File for Provisioning a Contrail Cluster with SR-IOV | 82](#)
- [Enabling ASPM in BIOS | 83](#)
- [Configuring SR-IOV Features Without env.sriov in testbed.py | 83](#)
- [Launching SR-IOV Virtual Machines | 84](#)

### Overview: Configuring SR-IOV

Contrail 3.0 and later supports single root I/O virtualization (SR-IOV) on Ubuntu systems only.

SR-IOV is an interface extension of the PCI Express (PCIe) specification. SR-IOV allows a device, such as a network adapter, to have separate access to its resources among various hardware functions.

As an example, the Data Plane Development Kit (DPDK) library has drivers that run in user space for several network interface cards (NICs). However, if the application runs inside a virtual machine (VM), it does not see the physical NIC unless SR-IOV is enabled on the NIC.

This topic shows how to configure SR-IOV with your Contrail system.

## Configuring SR-IOV Using JSON

If you are provisioning your system by using JSON, you can configure SR-IOV within the `server.json` file.

```

"parameters" : {
  "provision": {
    "contrail_4": {
      "sriov": {
        "p5p1": {
          "VF": 7,
          "physnets": [
            "physnet1"
          ]
        }
      }
    },
  },
}

```

## Preparing the `testbed.py` File for Provisioning a Contrail Cluster with SR-IOV

The `testbed.py` file is a Python file that is configured to specify all of the options necessary for the installation of a Contrail cluster. The `testbed.py` file can only be used with Contrail releases through 3.x.x, or with SM-Lite provisioning with Contrail 4.0.

An `env.sriov` entry in the `testbed.py` file is used to specify which NIC interface will be used to launch SR-IOV virtual machines (VMs).

Each VM to be configured as SR-IOV must be listed in the `env.sriov` entry, along with the number of virtual functions to be used and the physical network details, as in the following example:

```

env.sriov = {
    b7s36 :[ {'interface' : 'p6p1', 'VF' : 7, 'physnets' : ['physnet1', 'physnet2']}],
    b7s37 :[ {'interface' : 'p6p1', 'VF' : 7, 'physnets' : ['physnet1', 'physnet3']}],
}

```

The following are required entries for each VM specified in the `env.sriov`:

- `interface`—Specify the server NIC where SR-IOV VMs will be launched.
- `VF`—Specify the number of virtual functions to be configured.
- `physnets`—Specify the name of the physical networks to be used by each VM.

The `testbed.py` file is configured prior to the installation of Contrail.

Use the standard Contrail installation procedure with `fab` tools, and upon completion, your cluster is ready to be enabled to launch SR-IOV VMs with specified NICs.

## Enabling ASPM in BIOS

To use SR-IOV, it must have Active State Power Management (ASPM) enabled for PCI Express (PCIe) devices. Enable ASPM in the system BIOS.

**NOTE:** The BIOS of your system might need to be upgraded to a version that can enable ASPM.

## Configuring SR-IOV Features Without `env.sriov` in `testbed.py`

If you are enabling SR-IOV on a system where `testbed.py` with `env.sriov` was NOT used to install, you must also complete the following:

1. Enable the Intel Input-Output Memory Management Unit (IOMMU) on Linux, by doing the following:

- a. Make sure the IOMMU is turned on, using the following option line in `/etc/default/grub`:

```
GRUB_CMDLINE_LINUX_DEFAULT="nomdmonddf nomdmonisw intel_iommu=on"
```

- b. Enter the command `update-grub/sys`.

- c. Reboot the compute node.

2. Enable the required number of VFs on the selected NIC.

The following example enables seven VFs on the `eth0` interface. Check the configuration by using the command `lspci -nn` or `ip link`.

```
echo '7' > /sys/class/net/eth0/device/sriov_numvfs
```

3. In the Nova config file on the controller, configure the names of the physical networks whose VMs can use interface VFs. The following example enables the VMs attached to "physnet1" to use the VFs of "eth0":

```
/etc/nova/nova.conf
[default]
pci_passthrough_whitelist = { "devname": "eth0", "physical_network": "physnet1"}
```

4. Reboot Nova compute.

```
service nova-compute restart
```

5. In the Nova config file, configure a Nova Scheduler filter based on the new PCI configuration, as in the following example:

```
/etc/nova/nova.conf
[default]
scheduler_default_filters = PciPassthroughFilter
scheduler_available_filters = nova.scheduler.filters.all_filters
scheduler_available_filters =
nova.scheduler.filters.pci_passthrough_filter.PciPassthroughFilter
```

6. Restart Nova scheduler.

```
service nova-scheduler restart
```

## Launching SR-IOV Virtual Machines

After ensuring that SR-IOV features are enabled on your system, use one of the following procedures to create a virtual network from which to launch an SR-IOV VM, either by using the Contrail UI or the CLI. Both methods are included.

### Using the Contrail UI to Enable and Launch an SR-IOV Virtual Machine

To use the Contrail UI to enable and launch an SR-IOV VM:

1. At **Configure > Networking > Networks**, create a virtual network with SR-IOV enabled. Ensure the virtual network is created with a subnet attached. In the Advanced section, select the **Provider Network** check box, and specify the physical network already enabled for SR-IOV (in `testbed.py` or `nova.conf`) and its VLAN ID. See [Figure 16 on page 84](#).

**Figure 16: Edit Network**

Provider Network

Physical Network  
Network Name

VLAN  
0 - 4094

2. On the virtual network, create a Neutron port (**Configure > Networking > Ports**), and in the **Port Binding** section, define a **Key** value of SR-IOV and a **Value** of direct. See [Figure 17 on page 85](#).

Figure 17: Create Port

ECMP Hashing Fields

Select ECMP Hashing Fields

Device Owner

None

Port Binding(s)

| Key                       | Value  |
|---------------------------|--------|
| SR-IOV (vnic_type:direct) | direct |

Disable Policy

Sub Interface

Mirroring

Cancel Save

S018551

- Using the UUID of the Neutron port you created, use the `nova boot` command to launch the VM from that port.

```
nova boot --flavor m1.large --image <image name> --nic port-id=<uuid of above port> <vm name>
```

## Using the CLI to Enable and Launch SR-IOV Virtual Machines

To use CLI to enable and launch an SR-IOV VM:

- Create a virtual network with SR-IOV enabled. Specify the physical network already enabled for SR-IOV (in `testbed.py` or `nova.conf`) and its VLAN ID.

The following example creates `vn1` with a VLAN ID of 100 and is part of `physnet1`:

```
neutron net-create --provider:physical_network=physnet1 --provider:segmentation_id=100 vn1
```

- Create a subnet in `vn1`.

```
neutron subnet-create vn1 a.b.c.0/24
```

- On the virtual network, create a Neutron port on the subnet, with a binding type of `direct`.

```
neutron port-create --fixed-ip subnet_id=<subnet uuid>,ip_address=<IP address from above subnet> --name <name of port> <vn uuid> --binding:vnic_type direct
```

- Using the UUID of the Neutron port created, use the nova boot command to launch the VM from that port.

```
nova boot --flavor m1.large --image <image name> --nic port-id=<uuid of above port> <vm name>
```

- Log in to the VM and verify that the Ethernet controller is VF by using the lspci command to list the PCI buses.

The VF that gets configured with the VLAN can be observed using the ip link command.

## RELATED DOCUMENTATION

[Configuring the Data Plane Development Kit \(DPDK\) Integrated with Contrail vRouter](#) | 78

## Provisioning DPDK SRIOV with Server Manager

If you are using Server Manager to provision or reimage target servers, you can also use it to provision and enable DPDK and SRIOV.

The following is an example JSON for bringing up the DPDK vrouter as a virtual function.

In the example, p3p1 and p3p2 are the physical interfaces on the host. The server JSON has these defined with the proper MAC. The p3p1\_0 and p3p2\_0 are the virtual functions given to vrouter.

```
"parameters": {
  "provision": {
    "contrail": {
    },
    "contrail_4": {
      "ctrl_data_gw": "192.xxx.2.1",
      "coremask": "0xf",
      "huge_pages": "70",
      "sriov": {
        "p3p1": {
          "VF": "31",
          "physnets": [
            "physnet1"
```



```

    },
    {
      "name": "p3p1_0"
    },
    {
      "name": "p3p2_0"
    },
    {
      "bond_options": {
        "mode": "balance-rr",
        "miimon": "100",
        "xmit_hash_policy": "layer3+4"
      },
      "member_interfaces": [
        "p3p1_0",
        "p3p2_0"
      ],
      "name": "bond0",
      "type": "bond"
    },
    {
      "name": "vlan300",
      "dhcp": false,
      "ip_address": "192.xxx.2.17/27",
      "parent_interface": "bond0",
      "type": "vlan",
      "vlan": "300"
    }
  ],
  "management_interface": "em1",
  "provisioning": "kickstart"
},
"routes": [{
  "gateway": "192.xxx.2.1",
  "interface": "vlan300",
  "netmask": "255.255.255.224",
  "network": "192.xxx.2.32"
}],
"parameters": {
  "provision": {
    "contrail": {
      },
    },
  },
}

```



```

    "contrail_4": {
      "ctrl_data_gw": "192.xxx.2.1",
      "coremask": "0xf",
      "huge_pages": "70",
      "sriov": {
        "p3p1": {
          "VF": "31",
          "physnets": [
            "physnet1"
          ]
        },
        "p3p2": {
          "VF": "31",
          "physnets": [
            "physnet2"
          ]
        }
      }
    }
  },
  "password": "<password>",
  "roles": [
    "contrail-compute"
  ]
}

```

## Configuring Virtual Networks for Hub-and-Spoke Topology

### IN THIS SECTION

- [Route Targets for Virtual Networks in Hub-and-Spoke Topology | 90](#)
- [Example: Configuring Hub-and-Spoke Virtual Networks | 90](#)
- [Troubleshooting Hub-and-Spoke Topology | 91](#)

As of Contrail Release 3.0, hub-and-spoke topology can be used to ensure that virtual machines (VMs) don't communicate with each other directly; their communication is only allowed indirectly by means of a designated hub virtual network.

## Route Targets for Virtual Networks in Hub-and-Spoke Topology

Hub-and-spoke topology can be used to ensure that virtual machines (VMs) don't communicate with each other directly; their communication is only allowed indirectly by means of a designated hub virtual network (VN). The VMs are configured in spoke VNs.

This is useful for enabling VMs in a spoke VN to communicate by means of a policy or firewall, where the firewall exists in a hub site.

hub-and-spoke topology is implemented using two route targets (`hub-rt` and `spoke-rt`), as follows:

- Hub route target (`hub-rt`):
  - The hub VN *exports* all routes tagged with `hub-rt`.
  - The spoke VN *imports* routes tagged with `hub-rt`, ensuring that the spoke VN has only routes exported by the hub VN.
  - To attract spoke traffic, the hub VN readvertises the spoke routes or advertises the default route.
- Spoke route target (`spoke-rt`):
  - All spoke VNs export routes with route target `spoke-rt`.
  - The hub VN imports all spoke routes, ensuring that hub VN has all spoke routes.

**NOTE:** The hub VN or VRF can reside in an external gateway, such as an MX Series router, while the spoke VN resides in the Contrail controller.

## Example: Configuring Hub-and-Spoke Virtual Networks

The following example uses a script to configure the hub-and-spoke virtual networks.

In the example, the “`hub-vn`” is configured as a hub virtual network, with the import route target of “`target:1:1`” and the export route target of “`target:1:2`”. The “`spoke-vn*`” is configured as a spoke virtual network, with the import route target of “`target:1:2`” and the export route target of “`target:1:1`”.

The `spoke-rt` is “`target:1:1`” and the `hub-rt` is “`target:1:2`”, consequently, the “`hub-vn`” imports “`spoke-rt`” and exports “`hub-rt`”, and the `spoke-vn` imports “`hub-rt`” and exports “`spoke-rt`”.

## Using vnc-api to Configure Hub-and-Spoke Topology Example

```

from vnc_api.vnc_api import *
lib = VncApi("admin", "<password>", "admin", "<ip address>", "8082")
vn=lib.virtual_network_read(fq_name=["default-domain", "admin", "hub-vn"])
vn.set_import_route_target_list(RouteTargetList(["target:1:1"]))
vn.set_export_route_target_list(RouteTargetList(["target:1:2"]))
lib.virtual_network_update(vn)

vn=lib.virtual_network_read(fq_name=["default-domain", "admin", "spoke-vn1"])
vn.set_import_route_target_list(RouteTargetList(["target:1:2"]))
vn.set_export_route_target_list(RouteTargetList(["target:1:1"]))
lib.virtual_network_update(vn)

vn=lib.virtual_network_read(fq_name=["default-domain", "admin", "spoke-vn2"])
vn.set_import_route_target_list(RouteTargetList(["target:1:2"]))
vn.set_export_route_target_list(RouteTargetList(["target:1:1"]))
lib.virtual_network_update(vn)

vn=lib.virtual_network_read(fq_name=["default-domain", "admin", "spoke-vn3"])
vn.set_import_route_target_list(RouteTargetList(["target:1:2"]))
vn.set_export_route_target_list(RouteTargetList(["target:1:1"]))
lib.virtual_network_update(vn)

vn=lib.virtual_network_read(fq_name=["default-domain", "admin", "spoke-vn4"])
vn.set_import_route_target_list(RouteTargetList(["target:1:2"]))
vn.set_export_route_target_list(RouteTargetList(["target:1:1"]))
lib.virtual_network_update(vn)

```

## Troubleshooting Hub-and-Spoke Topology

The following examples provide methods to help you troubleshoot hub-and-spoke configurations.

### Example: Validating the Configuration on the Virtual Network

The following example uses the api-server HTTP get request to validate the configuration on the virtual network.

Hub VN configuration:

```
curl -u admin:<password> http://<host ip>/virtual-network/<hub-vn-uuid>| python -m json.tool
```

```
{
  "virtual-network": {
    "display_name": "hub-vn",
    "fq_name": [
      "default-domain",
      "admin",
      "hub-vn"
    ],
    "export_route_target_list": {
      "route_target": [
        "target:1:2"
      ]
    },
    "import_route_target_list": {
      "route_target": [
        "target:1:1"
      ]
    },
  }
}
```

Spoke VN configuration:

```
curl -u admin:<password> http://<host ip>:8095/virtual-network/<spoke-vn-uuid> | python -m json.tool
```

```
{
  {
    "virtual-network": {
      "display_name": "spoke-vn1",
      "fq_name": [
        "default-domain",
        "admin",
        "spoke-vn1"
      ],
      "export_route_target_list": {
        "route_target": [
          "target:1:1"
        ]
      },
    },
  }
```

```

    "import_route_target_list": {
      "route_target": [
        "target:1:2"
      ]
    },
  }
}

```

### Example: Validate the Configuration on the Routing Instance

The following example uses `api-server HTTP get` request to validate the configuration on the routing instance.

Spoke VRF configuration (with a system-created VRF by schema transformer):

```

user@node:/opt/contrail/utils# curl -u admin:<password> http://<host ip>:8095/routing-instance/<spoke-vrf-uuid>|
python -m json.tool

```

```

{
  "routing-instance": {
    "display_name": "spoke-vn1",
    "fq_name": [
      "default-domain",
      "admin",
      "spoke-vn1",
      "spoke-vn1"
    ],
    "route_target_refs": [
      {
        "attr": {
          "import_export": "export"
        },
        "href": "http://<host ip>:8095/route-target/446a3bbe-f263-4b58-
a537-8333878dd7c3",
        "to": [
          "target:1:1"
        ],
        "uuid": "446a3bbe-f263-4b58-a537-8333878dd7c3"
      },
      {
        "attr": {
          "import_export": null
        }
      }
    ]
  }
}

```

```

    },
    "href": "http://<host ip>:8095/route-target/7668088d-
e403-414f-8f5d-649ed80e0689",
    "to": [
        "target:64512:8000012"
    ],
    "uuid": "7668088d-e403-414f-8f5d-649ed80e0689"
},
{
    "attr": {
        "import_export": "import"
    },
    "href": "http://<host ip>:8095/route-target/8f216064-8488-4486-8fce-
b4afb87266bb",
    "to": [
        "target:1:2"
    ],
    "uuid": "8f216064-8488-4486-8fce-b4afb87266bb"
}
],
"routing_instance_is_default": true,
}
}

```

#### Hub VRF configuration:

```
curl -u admin:<password> http://<host ip>:8095/routing-instance/<hub-vrf-uuid> | python -m json.tool
```

```

{
  "routing-instance": {
    "display_name": "hub-vn",
    "fq_name": [
      "default-domain",
      "admin",
      "hub-vn",
      "hub-vn"
    ],
    "route_target_refs": [
      {
        "attr": {
          "import_export": "import"
        }
      }
    ]
  }
}

```

```

    },
    "href": "http://<host ip>:8095/route-target/446a3bbe-f263-4b58-
a537-8333878dd7c3",
    "to": [
        "target:1:1"
    ],
    "uuid": "446a3bbe-f263-4b58-a537-8333878dd7c3"
},
{
    "attr": {
        "import_export": "export"
    },
    "href": "http://<host ip>:8095/route-target/8f216064-8488-4486-8fce-
b4afb87266bb",
    "to": [
        "target:1:2"
    ],
    "uuid": "8f216064-8488-4486-8fce-b4afb87266bb"
},
{
    "attr": {
        "import_export": null
    },
    "href": "http://<host ip>:8095/route-target/a85fec19-eed2-430c-
af23-9919aca1dd12",
    "to": [
        "target:64512:8000016"
    ],
    "uuid": "a85fec19-eed2-430c-af23-9919aca1dd12"
}
],
"routing_instance_is_default": true,
}
}

```

### Example: Using Contrail Control Introspect

Figure 18 on page 96 shows the import and export targets for hub-vn and spoke-vns, by invoking `contrail-control-introspect`.

Figure 18: Contrail Introspect

The screenshot shows the Contrail Introspect interface. The browser address bar indicates the URL: `nodec13.englab.juniper.net:8083/5nh_ShowRoutingInstanceSummaryReq?search_string=`. The page title is "ShowRoutingInstanceSummaryResp". The main content is a table with the following columns: "name", "virtual\_network", "vn\_index", "vxlan\_id", "import\_target", and "export\_target". The table contains five rows of data representing different virtual networks and their associated targets.

| name                                     | virtual_network                | vn_index | vxlan_id | import_target                                       | export_target                                       |
|--|--------------------------------|----------|----------|---|---|
| default-domain:admin:hub-vn:hub-vn       | default-domain:admin:hub-vn    | 15       | 0        | import_target<br>target:1:1<br>target:64512:8000018 | export_target<br>target:1:2<br>target:64512:8000018 |
| default-domain:admin:spoke-vn1:spoke-vn1 | default-domain:admin:spoke-vn1 | 11       | 0        | import_target<br>target:1:2<br>target:64512:8000012 | export_target<br>target:1:1<br>target:64512:8000012 |
| default-domain:admin:spoke-vn2:spoke-vn2 | default-domain:admin:spoke-vn2 | 12       | 0        | import_target<br>target:1:2<br>target:64512:8000013 | export_target<br>target:1:1<br>target:64512:8000013 |
| default-domain:admin:spoke-vn3:spoke-vn3 | default-domain:admin:spoke-vn3 | 14       | 0        | import_target<br>target:1:2<br>target:64512:8000015 | export_target<br>target:1:1<br>target:64512:8000015 |
| default-domain:admin:spoke-vn4:spoke-vn4 | default-domain:admin:spoke-vn4 | 13       | 0        | import_target<br>target:1:2<br>target:64512:8000014 | export_target<br>target:1:1<br>target:64512:8000014 |

## Configuring Transport Layer Security-Based XMPP in Contrail

### IN THIS SECTION

- [Overview: TLS-Based XMPP | 96](#)
- [Configuring XMPP Client and Server in Contrail | 97](#)

### Overview: TLS-Based XMPP

Starting with Contrail 3.0, Transport Layer Security (TLS)-based XMPP can be used to secure all Extensible Messaging and Presence Protocol (XMPP)-based communication that occurs in the Contrail environment.

Secure XMPP is based on *RFC 6120, Extensible Messaging and Presence Protocol (XMPP): Core*.



## TLS XMPP in Contrail

In the Contrail environment, the Transport Layer Security (TLS) protocol is used for certificate exchange, mutual authentication, and negotiating ciphers to secure the stream from potential tampering and eavesdropping.

The RFC 6120 highlights a basic stream message exchange format for TLS negotiation between an XMPP server and an XMPP client.

**NOTE:** Simple Authentication and Security Layer (SASL) authentication is not supported in the Contrail environment.

## Configuring XMPP Client and Server in Contrail

In the Contrail environment, XMPP based communications are used in client and server exchanges, between the compute node (as the XMPP client), and:

- the control node (as the XMPP server)
- the DNS server (as the XMPP server)

## Configuring Control Node for XMPP Server

To enable secure XMPP, the following parameters are configured at the XMPP server.

On the control node, enable the parameters in the configuration file:

`/etc/contrail/contrail-control.conf`.

| Parameter                     | Description                            | Default   |
|-------------------------------|--|---|
| <code>xmpp_server_cert</code> | Path to the node's public certificate  | <code>/etc/contrail/ssl/certs/server.pem</code>           |
| <code>xmpp_server_key</code>  | Path to server's or node's private key | <code>/etc/contrail/ssl/private/server-privkey.pem</code> |
| <code>xmpp_ca_cert</code>     | Path to CA certificate                 | <code>/etc/contrail/ssl/certs/ca-cert.pem</code>          |

*(Continued)*

| Parameter             | Description            | Default  |
|-----------------------|------------------------|--|
| xmpp_auth_enable=true | Enables SSL based XMPP | Default is set to false, XMPP is disabled.<br><br><b>NOTE:</b> The keyword true is case sensitive. |

## Configuring DNS Server for XMPP Server

To enable secure XMPP, the following parameters are configured at the XMPP DNS server.

On the DNS server control node, enable the parameters in the configuration file:

`/etc/contrail/contrail-control.conf`

| Parameter                 | Description                           | Default  |
|---------------------------|---------------------------------------|--|
| xmpp_server_cert          | Path to the node's public certificate | <code>/etc/contrail/ssl/certs/server.pem</code>  |
| xmpp_server_key           | Path to server's/node's private key   | <code>/etc/contrail/ssl/certs/server-privkey.pem</code>  |
| xmpp_ca_cert              | Path to CA certificate                | <code>/etc/contrail/ssl/certs/ca-cert.pem</code>   |
| xmpp_dns_auth_enable=true | Enables SSL based XMPP                | Default is set to false, XMPP is disabled.<br><br><b>NOTE:</b> The keyword true is case sensitive. |

## Configuring Control Node for XMPP Client

To enable secure XMPP, the following parameters are configured at the XMPP client.

On the compute node, enable the parameters in the configuration file:

`/etc/contrail/contrail-vrouter-agent.conf`

| Parameter  | Description                           | Default  |
|--|---------------------------------------|--|
| xmpp_server_cert                                   | Path to the node's public certificate | /etc/contrail/ssl/certs/server.pem   |
| xmpp_server_key                                    | Path to server's/node's private key   | /etc/contrail/ssl/private/server-privkey.pem   |
| xmpp_ca_cert                                       | Path to CA certificate                | /etc/contrail/ssl/certs/ca-cert.pem  |
| xmpp_auth_enable=true<br>xmpp_dns_auth_enable=true | Enables SSL based XMPP                | Default is set to false, XMPP is disabled.<br><b>NOTE:</b> The keyword true is case sensitive. |

## Configuring Graceful Restart and Long-lived Graceful Restart

### IN THIS SECTION

- [Application of Graceful Restart and Long-lived Graceful Restart | 100](#)
- [BGP Graceful Restart Helper Mode | 100](#)
- [Feature Highlights | 100](#)
- [XMPP Helper Mode | 101](#)
- [Configuration Parameters | 101](#)
- [Cautions for Graceful Restart | 103](#)
- [Configuring Graceful Restart with the Contrail User Interface | 103](#)

Starting with Contrail Release 3.2, graceful restart and long-lived graceful restart BGP helper modes are supported for the Contrail control node. Release 4.1 introduces support for the XMPP helper mode as well.

## Application of Graceful Restart and Long-lived Graceful Restart

Whenever a BGP peer session is detected as down, all routes learned from the peer are deleted and immediately withdrawn from advertised peers. This causes instantaneous disruption to traffic flowing end-to-end, even when routes kept in the vrouter kernel in the data plane remain intact.

Graceful restart and long-lived graceful restart features can be used to alleviate traffic disruption caused by downs.

When configured, graceful restart features enable existing network traffic to be unaffected if Contrail controller processes go down. The Contrail implementation ensures that if a Contrail control module restarts, it can use graceful restart functionality provided by its BGP peers. Or when the BGP peers restart, Contrail provides a graceful restart helper mode to minimize the impact to the network. The graceful restart features can be used to ensure that traffic is not affected by temporary outage of processes.

Graceful restart is not enabled by default.

With graceful restart features enabled, learned routes are not deleted when sessions go down, and the routes are not withdrawn from the advertised peers. Instead, the routes are kept and marked as 'stale'. Consequently, if sessions come back up and routes are relearned, the overall impact to the network is minimized.

After a certain duration, if a downed session does not come back up, all remaining stale routes are deleted and withdrawn from advertised peers.

The graceful restart and long-lived graceful restart features can be enabled only for BGP peers in Contrail 3.2. Future releases will provide support for XMPP-based peering sessions (agents).

### BGP Graceful Restart Helper Mode

The BGP helper mode can be used to minimize routing churn whenever a BGP session flaps. This is especially helpful if the SDN gateway router goes down gracefully, as in an rpd crash or restart on an MX Series Junos device. In that case, the contrail-control can act as a graceful restart helper to the gateway, by retaining the routes learned from the gateway and advertising them to the rest of the network as applicable. In order for this to work, the restarting router (the SDN gateway in this case) must support and be configured with graceful restart for all of the address families used.

The graceful restart helper mode is also supported for BGP-as-a-Service (BGPaaS) clients. When configured, contrail-control can provide a graceful restart or long-lived graceful restart helper mode to a restarting BGPaaS client.

### Feature Highlights

The following are highlights of the graceful restart and long-lived graceful restart features.

- Configuring a non-zero restart time enables the ability to advertise graceful restart and long-lived graceful restart capabilities in BGP.
- Configuring helper mode enables the ability for graceful restart and long-lived graceful restart helper modes to retain routes even after sessions go down.
- With graceful restart configured, whenever a session down event is detected and a closing process is triggered, all routes, across all address families, are marked stale. The stale routes are eligible for best-path election for the configured graceful restart time duration.
- When long-lived graceful restart is in effect, stale routes can be retained for a much longer time than that allowed by graceful restart alone. With long-lived graceful restart, route preference is retained and best paths are recomputed. The community marked LLGR\_STALE is tagged for stale paths and re-advertised. However, if no long-lived graceful restart community is associated with any received stale route, those routes are not kept, instead, they are deleted.
- After a certain time, if a session comes back up, any remaining stale routes are deleted. If the session does not come back up, all retained stale routes are permanently deleted and withdrawn from the advertised peer.

## XMPP Helper Mode

Contrail release 4.1 introduces support for long-lived graceful restart (LLGR) with XMPP helper mode. Graceful restart and long lived graceful restart can be enabled using the Contrail web UI or by using the `provision_control` script.

The helper modes can also be enabled via schema, and can be disabled selectively in a `contrail-control` node for BGP or XMPP sessions by configuring `gr_helper_disable` in the `/etc/contrail/contrail-control.conf` configuration file.

## Configuration Parameters

Graceful restart parameters are configured in the `global-system-config` of the schema. They can be configured by means of a provisioning script or by using the Contrail Web UI.

Configure a non-zero restart time to advertise for graceful restart and long-lived graceful restart capabilities from peers.

Configure helper mode for graceful restart and long-lived graceful restart to retain routes even after sessions go down.

Configuration parameters include:

- `enable` or `disable` for all graceful restart parameters:
  - `restart-time`

- long-lived-restart-time
- end-of-rib-timeout
- bgp-helper-enable to enable graceful restart helper mode for BGP peers in contrail-control
- xmpp-helper-enable to enable graceful restart helper mode for XMPP peers (agents) in contrail-control

The following shows configuration by a provision script.

```
/opt/contrail/utils/provision_control.py
  --api_server_ip 10.xx.xx.20
  --api_server_port 8082
  --router_asn 64512
  --admin_user admin
  --admin_password <password>
  --admin_tenant_name admin
  --set_graceful_restart_parameters
  --graceful_restart_time 60
  --long_lived_graceful_restart_time 300
  --end_of_rib_timeout 30
  --graceful_restart_enable
  --graceful_restart_bgp_helper_enable
```

The following are sample parameters:

```
-set_graceful_restart_parameters
  --graceful_restart_time 300
  --long_lived_graceful_restart_time 60000
  --end_of_rib_timeout 30
  --graceful_restart_enable
  --graceful_restart_bgp_helper_enable
```

When BGP peering with Juniper Networks devices, Junos must also be explicitly configured for graceful restart/long-lived graceful restart, as shown in the following example:

```
set routing-options graceful-restart
set protocols bgp group <a1234> type internal
set protocols bgp group <a1234> local-address 10.xx.xxx.181
set protocols bgp group <a1234> keep all
set protocols bgp group <a1234> family inet-vpn unicast graceful-restart long-lived restarter
stale-time 20
```

```

set protocols bgp group <a1234> family route-target graceful-restart long-lived restarter stale-
time 20
set protocols bgp group <a1234> graceful-restart restart-time 600
set protocols bgp group <a1234> neighbor 10.xx.xx.20 peer-as 64512

```

The graceful restart helper modes can be enabled in the schema. The helper modes can be disabled selectively in the `contrail-control.conf` for BGP sessions by configuring `gr_helper_disable` in the `/etc/contrail/contrail-control.conf` file.

The following are examples:

```
/usr/bin/openstack-config /etc/contrail/contrail-control.conf DEFAULT gr_helper_bgp_disable 1
```

```
/usr/bin/openstack-config /etc/contrail/contrail-control.conf DEFAULT gr_helper_xmpp_disable 1
```

```
service contrail-control restart
```

For more details about graceful restart configuration, see <https://github.com/Juniper/contrail-controller/wiki/Graceful-Restart>.

## Cautions for Graceful Restart

Be aware of the following caveats when configuring and using graceful restart.

- Using the graceful restart/long-lived graceful restart feature with a peer is effective either to all negotiated address families or to none. If a peer signals support for graceful restart/long-lived graceful restart for only a subset of the negotiated address families, the graceful restart helper mode does not come into effect for any family in the set of negotiated address families.
- Because graceful restart is not yet supported for `contrail-vrouter-agent`, the parameter should *not* be set for `graceful_restart_xmpp_helper_enable`. If the vrouter agent restarts, the data plane is reset and the routes and flows are reprogrammed anew, which typically results in traffic loss for several seconds for new and /existing flows.
- Graceful restart/long-lived graceful restart is not supported for multicast routes.
- Graceful restart/long-lived graceful restart helper mode may not work correctly for EVPN routes, if the restarting node does not preserve forwarding state for EVPN routes.

## Configuring Graceful Restart with the Contrail User Interface

To configure graceful restart in the Contrail UI, go to **Configure > Infrastructure > Global Config**, then select the **BGP Options** tab. The **Edit BGP Options** window opens. Click the box for **Graceful Restart** to enable graceful restart, and enter a non-zero value for the **Restart Time**. Click the helper boxes as

needed for BGP Helper and XMPP Helper. You can also enter values for the long-lived graceful restart time in seconds, and for the end of RIB in seconds. See [Figure 19 on page 104](#).

**Figure 19: Configuring Graceful Restart**

The screenshot displays the 'Edit BGP Options' configuration window. The interface includes a sidebar on the left with navigation options: Infrastructure, Global Config, BGP Routers, Link Local Services, RBAC, Nodes, Project Settings, Service Appliance Sets, Service Appliances, Tags, Security, Physical Devices, and Networking. The main content area shows the following settings:

- Autonomous System:** 64512
- BGP as a Service Port Range (Start Port - End Port):** 50000 - 50512
- Graceful Restart:**
- BGP Helper:**
- Restart Time (secs):** 60
- End of RIB (secs):** 30
- iBGP Auto Mesh:**  Enable,  Disable
- Always Compare MED:**  Enable,  Disable
- XMPP Helper:**
- LLGR Time (secs):** 300

At the bottom of the window, there is a 'Cancel' button and a 'Save' button. A vertical label 's019909' is visible on the right side of the window.



# Using Contrail with Kubernetes

## IN THIS CHAPTER

- [Contrail Integration with Kubernetes | 105](#)
- [Installing and Provisioning Containerized Contrail Controller for Kubernetes | 112](#)
- [Viewing Configuration for CNI for Kubernetes | 123](#)
- [Provisioning Contrail CNI for Kubernetes | 126](#)
- [Using Kubernetes Helm to Provision Contrail | 134](#)

## Contrail Integration with Kubernetes

### IN THIS SECTION

- [What is Kubernetes? | 105](#)
- [Configuration Modes for Contrail Integrated with Kubernetes | 106](#)
- [Kubernetes Services | 109](#)
- [Ingress | 110](#)
- [Contrail Kubernetes Solution | 110](#)

Contrail Release 4.0 supports the Container Network Interface (CNI) for integrating Contrail with the Kubernetes automation platform.

### What is Kubernetes?

Kubernetes, also called K8s, is an open source platform for automating deployment, scaling, and operations of application containers across clusters of hosts, providing container-centric infrastructure. It provides a portable platform across public and private clouds. Kubernetes supports deployment, scaling, and auto-healing of applications.

Kubernetes supports a pluggable framework called Container Network Interface (CNI) for most of the basic network connectivity, including container pod addressing, network isolation, policy-based security, a gateway, SNAT, load-balancer, and service chaining capability for Kubernetes orchestration. Contrail Release 4.0 provides support for CNI for Kubernetes.

Kubernetes provides a flat networking model in which all container pods can talk to each other. Network policy is added to provide security between the pods. Contrail integrated with Kubernetes adds additional networking functionality, including multi-tenancy, network isolation, micro-segmentation with network policies, load-balancing, and more.

[Table 3 on page 106](#) lists the mapping between Kubernetes concepts and OpenContrail resources.

**Table 3: Kubernetes to OpenContrail Mapping**

| Kubernetes     | OpenContrail Resources                              |
|----------------|---|
| Namespace      | Shared or single project                            |
| Pod            | Virtual-machine, Interface, Instance-ip             |
| Service        | ECMP-based native Loadbalancer                      |
| Ingress        | HAProxy-based L7 Loadbalancer for URL routing       |
| Network policy | Security group based on namespace and pod selectors |

### What is a Kubernetes Pod?

A Kubernetes pod is a group of one or more containers (such as Docker containers), the shared storage for those containers, and options on how to run the containers. Pods are always co-located and co-scheduled, and run in a shared context. The shared context of a pod is a set of Linux namespaces, cgroups, and other facets of isolation. Within the context of a pod, individual applications might have further sub-isolations applied.

You can find more information about Kubernetes at: <http://kubernetes.io/docs/whatisk8s/>.

### Configuration Modes for Contrail Integrated with Kubernetes

Contrail can be configured in several different modes in Kubernetes. This section describes the various configuration modes.

## Default Mode

In Kubernetes, all pods can communicate with all other pods without using network address translation (NAT). This is the default mode of Contrail Kubernetes cluster. In the default mode, Contrail creates a virtual-network that is shared by all namespaces, from which service and pod IP addresses are allocated.

All pods in all namespaces that are spawned in the Kubernetes cluster are able to communicate with one another. The IP addresses for all of the pods are allocated from a pod subnet that is configured in the Contrail Kubernetes manager.

**NOTE:** System pods that are spawned in the kube-system namespace are not run in the Kubernetes cluster; they run in the underlay, and networking for these pods is not handled by Contrail.

## Namespace Isolation Mode

In addition to the default networking model mandated by Kubernetes, Contrail supports additional custom networking models that make available the many rich features of Contrail to the users of the Kubernetes cluster. One such feature is network isolation for Kubernetes namespaces.

For namespace isolation mode, the cluster administrator can configure a namespace annotation to turn on isolation. As a result, services in that namespace are not accessible from other namespaces, unless security groups or network policies are explicitly defined to allow access.

A Kubernetes namespace can be configured as isolated by annotating the Kubernetes namespace metadata:

```
opencontrail.org/isolation : true
```

Namespace isolation provides network isolation to pods, because the pods in isolated namespaces are not reachable to pods in other namespaces in the cluster.

Namespace isolation also provides service isolation to pods. If any Kubernetes service is implemented by pods in an isolated namespace, those pods are reachable only to pods in the same namespace through the Kubernetes service-ip.

To make services remain reachable to other namespaces, service isolation can be disabled by the following additional annotation on the namespace:

```
opencontrail.org/isolation.service : false
```

Disabling service isolation makes the services reachable to pods in other namespaces, however pods in isolated namespaces still remain unreachable to pods in other namespaces.

A namespace annotated as “isolated” for both pod and service isolation has the following network behavior:

- All pods created in an isolated namespace have network reachability with each other.
- Pods in other namespaces in the Kubernetes cluster *cannot* reach pods in the isolated namespace.
- Pods created in isolated namespaces *can* reach pods in non-isolated namespaces.
- Pods in isolated namespaces *can* reach non-isolated services in any namespace in the Kubernetes cluster.
- Pods from other namespaces *cannot* reach services in isolated namespaces.

A namespace annotated as “isolated”, with service-isolation disabled and only pod isolation enabled, has the following network behavior:

- All pods created in an isolated namespace have network reachability with each other.
- Pods in other namespaces in the Kubernetes cluster *cannot* reach pods in the isolated namespace.
- Pods created in isolated namespaces *can* reach pods in other namespaces.
- Pods in isolated namespaces *can* reach non-isolated services in any namespace in the Kubernetes cluster.
- Pods from other namespaces *can* reach services in isolated namespaces.

### Custom Isolation Mode

Administrators and application developers can add annotations to specify the virtual network in which a pod or all pods in a namespace are to be provisioned. The annotation to specify this custom virtual network is:

```
"opencontrail.org/network: <fq_network_name>"
```

If this annotation is configured on a pod spec then the pod is launched in that network. If the annotation is configured in the namespace spec then all the pods in the namespace are launched in the provided network.

**NOTE:** The virtual network must be created using Contrail VNC APIs or Contrail-UI prior to configuring it in the pod or namespace spec.

## Nested Mode

Contrail supports the provisioning of Kubernetes cluster inside an OpenStack cluster. While this nesting of clusters by itself is not unique, Contrail provides a *collapsed* control and data plane in which a single Contrail control plane and a single network stack manage and service both the OpenStack and Kubernetes clusters. With unified control and data planes, interworking and configuring these clusters is seamless, and the lack of replication and duplicity makes this a very efficient option.

In nested mode, a Kubernetes cluster is provisioned in the virtual machine of an OpenStack cluster. The CNI-plugin and the Contrail-kubernetes manager of the Kubernetes cluster interface directly with Contrail components that manage the OpenStack cluster.

In a nested-mode deployment, all Kubernetes features, functions, and specifications are supported as is. Nested deployment stretches the boundaries and limits of Kubernetes by allowing it to operate on the same plane as underlying OpenStack cluster.

## Kubernetes Services

A Kubernetes service is an abstraction that defines a logical set of pods and the policy used to access the pods. The set of pods implementing a service are selected based on the **LabelSelector** field in the service definition. In Contrail, a Kubernetes service is implemented as an ECMP-native load-balancer.

The Contrail Kubernetes integration supports the following **ServiceTypes**:

- **`clusterIP`**: This is the default mode. Choosing this **ServiceType** makes the service reachable through the cluster network.
- **`LoadBalancer`**: Designating a **ServiceType** as **`LoadBalancer`** enables the service to be accessed externally. The **`LoadBalancer`** **\_Service\_** is assigned both ClusterIP and ExternalIP addresses. This **ServiceType** assumes that the user has configured the public network with a floating-ip pool.

Contrail Kubernetes Service-integration supports TCP and UDP for protocols. Also, Service can expose more than one port where port and targetPort are different. For example:

```
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - name: http
      protocol: TCP
```

```

port: 80
targetPort: 9376
- name: https
  protocol: TCP
  port: 443
  targetPort: 9377

```

Kubernetes users can specify `spec.clusterIP` and `spec.externalIPs` for both **LoadBalancer** and **clusterIP ServiceTypes**.

If **ServiceType** is **LoadBalancer** and no `spec.externalIP` is specified by the user, then `contrail-kube-manager` allocates a floating-ip from the public pool and associates it to the `ExternalIP` address.

## Ingress

Kubernetes services can be exposed externally or exposed outside of the cluster in many ways. See <https://kubernetes.io/docs/concepts/services-networking/ingress/#alternatives> for a list of all methods of exposing Kubernetes services externally. Ingress is one such method. Ingress provides Layer 7 load-balancing whereas the other methods provide Layer 4 load-balancing. Contrail supports http-based single-service ingress, simple-fanout ingress, and name-based virtual hosting ingress.

## Contrail Kubernetes Solution

Contrail Kubernetes solution includes the following elements.

### Contrail Kubernetes Manager

The Contrail Kubernetes implementation requires listening to the Kubernetes API messages and creating corresponding resources in the Contrail API database.

A new module, `contrail-kube-manager`, runs in a Docker container to listen to the messages from the Kubernetes API server.

### ECMP Load-Balancers for Kubernetes Services

Each service in Kubernetes is represented by a load-balancer object. The service IP allocated by Kubernetes is used as the VIP for the load-balancer. Listeners are created for the port on which the service is listening. Each pod is added as a member of the listener pool. The `contrail-kube-manager` listens for any changes based on service labels or pod labels, and updates the member pool list with any added, updated, or deleted pods.

Load-balancing for services is Layer 4 native, non-proxy load-balancing based on ECMP. The instance-ip (service-ip) is linked to the ports of each of the pods in the service. This creates an ECMP next-hop in Contrail and traffic is load-balanced directly from the source pod.

## HAProxy Loadbalancer for Kubernetes Ingress

Kubernetes Ingress is implemented through the HAProxy load-balancer feature in Contrail. Whenever ingress is configured in Kubernetes, contrail-kube-manager creates the load-balancer object in contrail-controller. The Contrail service monitor listens for the load-balancer objects and launches the HAProxy with appropriate configuration, based on the ingress specification rules in active-standby mode.

See "[Using Load Balancers in Contrail](#)" on page 743 for more information on load balancers.

## Security Groups for Kubernetes Network Policy

Kubernetes network policy is a specification of how groups of pods are allowed to communicate with each other and other network endpoints. **NetworkPolicy** resources use labels to select pods and define white list rules which allow traffic to the selected pods in addition to what is allowed by the isolation policy for a given namespace.

For more information about Kubernetes network policies, see <https://kubernetes.io/docs/concepts/services-networking/networkpolicies/>.

The contrail-kube-manager listens to the Kubernetes network policy events for create, update, and delete, and translates the Kubernetes network policy to Contrail security group objects applied to virtual machine interfaces (VMIs). The VMIs are dynamically updated as pods and labels are added and deleted.

## Domain Name Server (DNS)

Kubernetes implements DNS using SkyDNS, a small DNS application that responds to DNS requests for service name resolution from pods. SkyDNS runs as a pod in Kubernetes.

## RELATED DOCUMENTATION

[Installing and Provisioning Containerized Contrail Controller for Kubernetes](#) | 112

[Viewing Configuration for CNI for Kubernetes](#) | 123

## Installing and Provisioning Containerized Contrail Controller for Kubernetes

### IN THIS SECTION

- [Installing and Provisioning Containerized Contrail Controller for Kubernetes | 114](#)
- [inventory/my-inventory/hosts Inventory File | 115](#)
- [inventory/my-inventory/group\\_vars/all.yml Inventory File | 118](#)
- [Example: inventory/my-inventory/group\\_vars/all.yml File in a Stand-alone Contrail Kubernetes Cluster | 120](#)
- [Example: inventory/my-inventory/group\\_vars/all.yml File in a Nested Contrail Kubernetes Cluster | 121](#)

This section describes the steps required to install and provision containerized Contrail Controller for Kubernetes.

Ensure the following prerequisites are met for successful provisioning of a Contrail Kubernetes cluster.

- An installed and running Kubernetes cluster is available.

You can choose the installation method for Kubernetes.

- Kubernetes cluster must have at least one worker node.

The Kubernetes cluster consists of one master node and at least one worker node. Kubernetes “tainted” master, a mode in which worker pods are scheduled on Kubernetes master node, is not supported.

- Ensure that Kubelet running on the Kubernetes master node does *not* have network plugin options.

If Kubelet is configured with a network plugin option:

1. Disable or comment out the KUBELET\_NETWORK\_ARGS option in the configuration file.

```
/etc/systemd/system/kubelet.service.d/10-kubeadm.conf
```

2. Restart the Kubelet service.

```
systemctl daemon-reload; systemctl restart kubelet.service
```

- Get a service account token that has a cluster-admin cluster role.

You can configure this token in `contrail-ansible` during provisioning of the Contrail Kubernetes cluster.

For more information, refer to the `kubernetes_access_token` variable in the `all.yml` in `contrail-ansible`.



1. Create a service account and bind it to the cluster-admin cluster role.

```
kubectl create clusterrolebinding <role-binding-name> --clusterrole=cluster-admin --
serviceaccount=<service-account-name>
```

Alternatively, you can bind the cluster-admin role to an existing service account.

**Example:** Bind a cluster-admin role to a service account named default.

```
kubectl create clusterrolebinding contrail-kube-manager --clusterrole=cluster-admin --
serviceaccount=default:default
```

2. Get the secret associated with the service account.

```
kubectl describe sa <service-account-name>
```

Example:

```
> kubectl describe sa default
Name:          default
Namespace:    default
Labels:       <none>
Annotations:  <none>
Tokens:       default-token-r353k <-----
Image pull secrets: <none>
Mountable secrets: default-token-r353k
```

3. Get the token associated with the secret.

```
kubectl describe secret <name>
```

Example:

```
> kubectl describe secret default-token-r353k
Name:          default-token-r353k
Namespace:    default
Labels:       <none>
Annotations:  kubernetes.io/service-account.name=default
              kubernetes.io/service-account.uid=4fbcc5cf-3fed-11e7-acf4-0271c93f63d6
Type:         kubernetes.io/service-account-token
Data
====
ca.crt:      1025 bytes
```

```
namespace: 7 bytes
token:     $123ABC
```

## Installing and Provisioning Containerized Contrail Controller for Kubernetes

Perform the following steps to install containerized Contrail controller for Kubernetes:

1. Set up password-free access to all hosts from Ansible host.

```
ssh-keygen -t rsa
ssh <user>@<host-ip> mkdir -p .ssh
ssh <user>@<host-ip> chmod 700 .ssh
cat .ssh/id_rsa.pub | ssh <user>@<host-ip> 'cat >> .ssh/authorized_keys'
```

Ensure `ssh <user>@<host-ip>` works fine.

2. Install Ansible on your Mac OS X or any other machine. Version must be = 2.2.0.

```
sudo easy_install pip
sudo pip install ansible==2.2.0
```

3. Download `contrail-kubernetes-docker_<release>_<os-name>.tgz` package and extract it. The extracted package contains `contrail-networking-tools_<release>.tgz` and `contrail-kubernetes-docker-images_<release>.tgz` packages.

The `contrail-networking-tools_<release>.tgz` contains the `contrail-ansible` package while the `contrail-kubernetes-docker-images_<release>.tgz` contains Contrail container images.

4. Extract the `contrail-ansible` package from the `contrail-networking-tools_<release>.tgz` package.

`contrail-ansible` is used to provision a Contrail Kubernetes cluster. The `contrail-ansible` repo contains a `site.yml` playbook that has the requisite roles and tasks to provision a fully-functional Contrail Kubernetes cluster. The inventory files in the repo expose all the parameters required by the playbook to provision the cluster. The `contrail-ansible` directory-based inventory file mechanism is recommended for provisioning.

**NOTE:** The scope of `contrail-ansible` is to provision only the Contrail part of the Kubernetes solution. The Kubernetes cluster should be provisioned independently using recommended Kubernetes guidelines.

Contrail Kubernetes clusters can be provisioned in the following modes:

- **Stand-alone Contrail Kubernetes cluster**

In this mode, Contrail provides networking to a stand-alone Kubernetes cluster. Contrail components are provisioned and dedicated to the management of this cluster.

- **Nested Contrail Kubernetes cluster**

In this mode, Contrail provides networking for a Kubernetes cluster that is provisioned on a Contrail OpenStack cluster. Contrail components are shared between the two clusters. Ansible provisions only the Contrail components that directly interface with the Kubernetes API server. All other Contrail components are shared between OpenStack and Kubernetes clusters.

5. Create a folder called `container_images` inside `contrail-ansible/playbook`. Copy container images to this folder by extracting `contrail-kubernetes-docker-images_<release>.tgz`.

6. Update the inventory file.

The inventory files in directory-based provisioning are as following:

- `inventory/my-inventory/hosts`. See "[inventory/my-inventory/hosts Inventory File](#)" on page 115 for more information.
- `inventory/my-inventory/group_vars/all.yml`. See "[inventory/my-inventory/group\\_vars/all.yml Inventory File](#)" on page 118 for more information.

7. Run the Ansible playbook from `contrail-ansible/playbook`.

```
ansible-playbook -i inventory/my-inventory site.yml
```

## inventory/my-inventory/hosts Inventory File

This section describes the parameters and provides examples of the `inventory/my-inventory/hosts` inventory file in stand-alone and nested Contrail Kubernetes clusters.

[Table 4 on page 116](#) lists the parameters used in the `inventory/my-inventory/hosts` inventory file.

In [Table 4 on page 116](#), **Cluster Mode** is one of the following:

- Stand-alone –Applicable only to a stand-alone cluster.
- Nested –Applicable only to nested cluster.
- Both–Applicable to both stand-alone and nested clusters.

**Table 4: Parameters in inventory/my-inventory/hosts**

| Parameter                       | Cluster Mode | Description   |
|---------------------------------|--------------|---|
| contrail-repo                   | Nested       | List of hosts where contrail apt or yum repo container will be started. This repo will be used by other nodes on installing any packages in the node. Setting up contrail-cni needs this repo enabled |
| contrail-controllers            | Stand-alone  | List of hosts where contrail-controller container or processes are to be provisioned. .   |
| contrail-analyticsdb            | Stand-alone  | List of hosts where contrail-analyticsdb container or process is to be provisioned.   |
| contrail-analytics              | Stand-alone  | List of hosts where contrail-analytics container or process is to be provisioned.   |
| contrail-kubernetes             | Both         | Node where contrail-kube-manager container or process is to be run.   |
| contrail-compute                | Both         | List of hosts which are to be provisioned as kubernetes compute/ minion nodes. Contrail vRouter or vrouter-agent or CNI will be provisioned on these nodes.   |
| kubernetes-contrail-controllers | Nested       | List of nodes with pre-existing contrail-controller container or processes to which contrail-kube-manager should connect to.  |
| kubernetes-contrail-analytics   | Nested       | List of nodes with pre-existing contrail-analytics container or processes to which contrail-kube-manager should connect to.   |

### Example: inventory/my-inventory/hosts File in a Stand-alone Contrail Kubernetes Cluster

The following is an example of the `inventory/my-inventory/hosts` file in a stand-alone Contrail Kubernetes cluster:

```
[contrail-controllers]
10.xx.27.16

[contrail-analyticsdb]
10.xx.27.16

[contrail-analytics]
10.xx.27.16

[contrail-kubernetes]
10.xx.27.16

[contrail-compute]
10.xx.23.37
```

### Example: Nested inventory/my-inventory/hosts File in a Nested Contrail Kubernetes Cluster

The following is an example of the `inventory/my-inventory/hosts` file in a nested Contrail Kubernetes cluster:

```
[contrail-repo]
10.xx.31.71

[contrail-kubernetes]
10.xx.31.71

[contrail-compute]
10.xx.31.72

[kubernetes-contrail-controllers]
10.xx.29.27

[kubernetes-contrail-analytics]
10.xx.29.27
```

## inventory/my-inventory/group\_vars/all.yml Inventory File

This section describes the parameters and provides examples of the **inventory/my-inventory/group\_vars/all.yml** inventory file in stand-alone and nested Contrail Kubernetes clusters.

[Table 5 on page 118](#) describes the configuration parameters used in the **inventory/my-inventory/group\_vars/all.yml** inventory file.

In [Table 5 on page 118](#), **Cluster Mode** is one of the following:

- Stand-alone –Applicable only to a stand-alone cluster.
- Nested –Applicable only to nested cluster.
- Both–Applicable to both stand-alone and nested clusters.

**Table 5: Parameters in inventory/my-inventory/group\_vars/all.yml**

| Parameter                      | Value   | Default    | Cluster Mode | Description  |
|--------------------------------|---|------------|--------------|--|
| cloud_orchestrator             | Kubernetes  | None       | Both         | Specifies orchestrator type.   |
| contrail_compute_mode          | container   | bare_metal | Both         | Specifies if the Contrail components must be run as containers or as processes on a stand-alone server.  |
| keystone_config                | {ip: <ip>,<br>admin_password:<br><passwd>,<br>admin_user:<br><username>,<br>admin_tenant:<br><tenant-name>} | None       | Nested       | Keystone authentication information.   |
| nested_cluster_private_network | "<cluster-private-CIDR>"  | None       | Nested       | The IP subnet reserved for use by Kubernetes for internal cluster management and housekeeping. The Ansible user is responsible to make sure this CIDR does not collide with existing CIDRs in the virtual-network. |

Table 5: Parameters in inventory/my-inventory/group\_vars/all.yml (Continued)

| Parameter                          | Value  | Default                                    | Cluster Mode | Description   |
|------------------------------------|--|--|--------------|---|
| kubernetes_cluster_name            | <cluster-name>   | k8s-default                                | Both         | Name of the Kubernetes cluster being provisioned.   |
| nested_cluster_network             | {domain: <name>, project: <name>, name: <name>}          | None                                       | Nested       | Virtual Network in which the Kubernetes cluster must be provisioned. This network must be the same network to which the virtual machines that host the Kubernetes cluster belong. |
| kubernetes_access_token            | < token >  | None                                       | Both         | RBAC token to connect to Kubernetes API server.   |
| nested_mode                        | true   | None                                       | Nested       | Parameter to enable nested provisioning of a Kubernetes cluster.  |
| kubernetes_public_floating_ip_pool | {domain: <id>, project: <id>, network: <id>, name: <id>} | None                                       | Both         | Kubernetes FloatingIpPool to be used for service or ingress.  |
| kubernetes_cluster_project         | {domain: <id>, project: <id>}                            | {domain: default-domain, project: default} | Both         | Fq-name of Contrail project within which Kubernetes cluster must be provisioned.  |
| kubernetes_pod_subnet              | <CIDR>   | 10.32.0.0/12                               | Both         | Pod subnet used by Kubernetes cluster.  |
| kubernetes_service_subnet          | <CIDR>   | 10.96.0.0/12                               | Both         | Service subnet used by Kubernetes cluster.  |

**Table 5: Parameters in inventory/my-inventory/group\_vars/all.yml (Continued)**

| Parameter             | Value | Default                  | Cluster Mode | Description                                     |
|-----------------------|-------|--------------------------|--------------|---|
| kubernetes_api_server | <IP>  | Contrail Control Node IP | Both         | Node on which kubernetes-api server is running. |

### Example: inventory/my-inventory/group\_vars/all.yml File in a Stand-alone Contrail Kubernetes Cluster

The following is an example of the `inventory/my-inventory/group_vars/all.yml` file in a stand-alone Contrail Kubernetes cluster:

```

docker_install_method: package
docker_py_pkg_install_method: pip

# ansible connection details
ansible_user: root
ansible_become: true
ansible_ssh_private_key_file: ~/.ssh/id_rsa

contrail_compute_mode: container

os_release: ubuntu14.04

# contrail version
contrail_version: 4.0.0.0-16

cloud_orchestrator: kubernetes

# vrouter physical interface
vrouter_physical_interface: enp6s0f0

# global_config:

analytics_api_config: {aaa_mode: no-auth}

# To configure custom webui http port
webui_config: {http_listen_port: 8085}

```







```
57eEFTstCxcpR4itqxsRi7jc0nrrcbDkv10kDhA93ID4ChPwE2PcsAf_LV9ds-g
SzuyPIQt0qdxnQvI262AjgeNowbQhkYguoqZWJIE--AwpGSE0NiNpjcxixUx1HC2
uaRSP3g9mMr2g4YQHRjxJwuz3fUkaSRNZyQEpyE5G5WKXTefc7h52R5Kph
n2nT9gg6x175mrrnNQ

# Kubernetes cluster is nested within an Openstack cluster.
nested_mode: true

# Kubernetes API server IP.
kubernetes_api_server: 10.14.27.16
```

## RELATED DOCUMENTATION

[Contrail Integration with Kubernetes | 105](#)

[Viewing Configuration for CNI for Kubernetes | 123](#)

## Viewing Configuration for CNI for Kubernetes

### IN THIS SECTION

- [View Pod Name and IP Address | 124](#)
- [Verify Reachability of Pods | 124](#)
- [Verify If Isolated Namespace-Pods Are Not Reachable | 124](#)
- [Verify If Non-Isolated Namespace-Pods Are Reachable | 125](#)
- [Verify If a Namespace is Isolated | 126](#)

Use the verification steps in this topic to view and verify your configuration of Contrail Container Network Interface (CNI) for Kubernetes.

## View Pod Name and IP Address

Use the following command to view the IP address allocated to a pod.

```
[root@device ~]# kubectl get pods --all-namespaces -o wide
```

| NAMESPACE | NAME         | READY          | STATUS  | RESTARTS | AGE | IP | NODE |
|-----------|--------------|----------------|---------|----------|-----|----|------|
| default   | client-1     | 1/1            | Running | 0        |     |    |      |
| 19d       | 10.47.25.247 | k8s-minion-1-3 |         |          |     |    |      |
| default   | client-2     | 1/1            | Running | 0        |     |    |      |
| 19d       | 10.47.25.246 | k8s-minion-1-1 |         |          |     |    |      |
| default   | client-x     | 1/1            | Running | 0        |     |    |      |
| 19d       | 10.84.21.272 | k8s-minion-1-1 |         |          |     |    |      |

## Verify Reachability of Pods

Perform the following steps to verify if the pods are reachable to each other.

1. Determine the IP address and name of the pod.

```
[root@device ~]# kubectl get pods --all-namespaces -o wide
```

| NAME           | READY | STATUS  | RESTARTS | AGE | IP           | NODE  |
|----------------|-------|---------|----------|-----|--------------|-------|
| example1-36xpr | 1/1   | Running | 0        | 43s | 10.47.25.251 | b3s37 |
| example2-pldp1 | 1/1   | Running | 0        | 39s | 10.47.25.250 | b3s37 |

2. Ping the destination pod from the source pod to verify if the pod is reachable.

```
root@device ~]# kubectl exec -it example1-36xpr ping 10.47.25.250
```

```
PING 10.47.25.250 (10.47.25.250): 56 data bytes
```

```
64 bytes from 10.47.25.250: icmp_seq=0 ttl=63 time=1.510 ms
```

```
64 bytes from 10.47.25.250: icmp_seq=1 ttl=63 time=0.094 ms
```

## Verify If Isolated Namespace-Pods Are Not Reachable

Perform the following steps to verify if pods in isolated namespaces cannot be reached by pods in non-isolated namespaces.

1. Determine the IP address and name of a pod in an isolated namespace.

```
[root@device ~]# kubectl get pod -n test-isolated-ns -o wide
```

| NAME           | READY | STATUS  | RESTARTS | AGE | IP           | NODE  |
|----------------|-------|---------|----------|-----|--------------|-------|
| example3-bvqx5 | 1/1   | Running | 0        | 1h  | 10.47.25.249 | b3s37 |

2. Determine the IP address of a pod in a non-isolated namespace.

```
[root@device ~]# kubectl get pods
```

| NAME           | READY | STATUS  | RESTARTS | AGE |
|----------------|-------|---------|----------|-----|
| example1-36xpr | 1/1   | Running | 0        | 15h |
| example2-pldp1 | 1/1   | Running | 0        | 15h |

3. Ping the IP address of the pod in the isolated namespace from the pod in the non-isolated namespace.

```
[root@device ~]# kubectl exec -it example1-36xpr ping 10.47.25.249
--- 10.47.255.249 ping statistics ---
2 packets transmitted, 0 packets received, 100% packet loss
```

## Verify If Non-Isolated Namespace-Pods Are Reachable

Perform the following steps to verify if pods in non-isolated namespaces can be reached by pods in isolated namespaces.

1. Determine the IP address of a pod in a non-isolated namespace.

```
[root@device ~]# kubectl get pods -o wide
```

| NAME           | READY | STATUS  | RESTARTS | AGE | IP           | NODE  |
|----------------|-------|---------|----------|-----|--------------|-------|
| example1-36xpr | 1/1   | Running | 0        | 15h | 10.47.25.251 | b3s37 |
| example2-pldp1 | 1/1   | Running | 0        | 15h | 10.47.25.250 | b3s37 |

2. Determine the IP address and name of a pod in an isolated namespace.

```
[root@device ~]# kubectl get pod -n test-isolated-ns -o wide
```

| NAME           | READY | STATUS  | RESTARTS | AGE | IP           | NODE  |
|----------------|-------|---------|----------|-----|--------------|-------|
| example3-bvqx5 | 1/1   | Running | 0        | 1h  | 10.47.25.249 | b3s37 |

3. Ping the IP address of the pod in the non-isolated namespace from a pod in the isolated namespace.

```
[root@device ~]# kubectl exec -it example3-bvqx5 -n test-isolated-ns ping 10.47.25.251
PING 10.47.25.251 (10.47.25.251): 56 data bytes
64 bytes from 10.47.25.251: icmp_seq=0 ttl=63 time=1.467 ms
64 bytes from 10.47.25.251: icmp_seq=1 ttl=63 time=0.137 ms
^C--- 10.47.25.251 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.137/0.802/1.467/0.665 ms
```

## Verify If a Namespace is Isolated

Namespace annotations are used to turn on isolation in a Kubernetes namespace. In isolated Kubernetes namespaces, the namespace metadata is annotated with the `opencontrail.org/isolation : true` annotation.

Use the following command to view annotations on a namespace.

```
[root@a7s16 ~]#
kubectl describe namespace test-isolated-ns
Name:          test-isolated-ns
Labels:        <none>
Annotations:   opencontrail.org/isolation : true    Namespace is isolated
Status:        Active
```

## RELATED DOCUMENTATION

[Contrail Integration with Kubernetes | 105](#)

[Installing and Provisioning Containerized Contrail Controller for Kubernetes | 112](#)

## Provisioning Contrail CNI for Kubernetes

### IN THIS SECTION

● [Requirements | 127](#)

● [Overview | 130](#)

- Configuration | 130
- Troubleshooting | 132

You can use the following procedure to provision Contrail Container Network Interface (CNI) for Kubernetes.

## Requirements

This procedure requires the following minimum virtual machine and host specifications:

- 32 GB RAM
- Eight vCPUs
- 150 GB disk space

The supported software versions for provisioning Contrail CNI for Kubernetes are:

- Kubernetes 1.6
- Docker Engine versions 1.11.0 to 1.13.0
- Ubuntu 16.04.2 or CentOS 7 operating systems

## Preparing for Installation

Before provisioning Contrail CNI for Kubernetes, ensure the following prerequisites are met:

1. Stop the firewall service and delete all the iptable rules.

- For Ubuntu 16.04 host OS, use the following commands:

```
sudo service ufw stop
sudo iptables -F
```

- For CentOS 7 host OS, use the following commands:

```
sudo service firewalld stop
sudo iptables -F
```

2. Ensure that the Kubernetes cluster is running. You can choose any method to install Kubernetes. For quick installation steps, use the following commands:

- Commands for Ubuntu 16.04 host OS:

```

sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl -y

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
sudo bash -c 'cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb http://apt.kubernetes.io/ kubernetes-xenial main
EOF'

sudo apt-get update -y
sudo apt-get install -y kubect1
sudo apt-get install -y kubelet
sudo apt-get install -y kubeadm
sudo apt-get install -y docker-engine

sudo kubeadm init

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

# If you have multi-node cluster then please copy paste the kubeadm join line
# on all the slave nodes from the stdout of sudo kubeadm init command.
# It looks like below
sudo kubeadm join --token fd554a.97d239c2234d0de352 192.0.2.0:6443

```

- Commands for CentOS 7 host OS:

```

sudo bash -c 'cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
      https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
EOF'

```



```

sudo setenforce 0 || true

yum install -y kubelet kubeadm kubectl docker
systemctl enable docker && systemctl start docker
systemctl enable kubelet && systemctl start kubelet

echo 1 > /proc/sys/net/bridge/bridge-nf-call-iptables

sudo kubeadm init --kubernetes-version v1.7.4

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

# If you have multi-node cluster then please copy paste the kubeadm join line
# on all the slave nodes from the stdout of sudo kubeadm init command.
# It looks like below
sudo kubeadm join --token fd554a.97d239c2234d0de352 192.0.2.01:6443

```

**3. Patch LivenessProbe and ReadinessProbe of kube-dns deployment using the following commands:**

```

kubectl patch deploy/kube-dns --type json -p='[{"op": "replace", "path": "/spec/template/spec/containers/0/readinessProbe", "value": {"exec": {"command": ["wget", "-O", "-", "http://192.0.2.1:8081/readiness"]}}}]' -n kube-system
kubectl patch deploy/kube-dns --type json -p='[{"op": "replace", "path": "/spec/template/spec/containers/0/livenessProbe", "value": {"exec": {"command": ["wget", "-O", "-", "http://192.0.2.1:10054/healthcheck/kubedns"]}}}]' -n kube-system && kubectl patch deploy/kube-dns --type json -p='[{"op": "replace", "path": "/spec/template/spec/containers/1/livenessProbe", "value": {"exec": {"command": ["wget", "-O", "-", "http://192.0.2.1:10054/healthcheck/dnsmasq"]}}}]' -n kube-system && kubectl patch deploy/kube-dns --type json -p='[{"op": "replace", "path": "/spec/template/spec/containers/2/livenessProbe", "value": {"exec": {"command": ["wget", "-O", "-", "http://192.0.2.1:10054/metrics"]}}}]' -n kube-system

```

**4. (Optional) If you are re-provisioning Contrail on the same setup, ensure that you delete the configdb and analyticsdb data from the previous installation.**

```

sudo rm -rf /var/lib/contrail*
sudo rm -rf /var/lib/configdb*
sudo rm -rf /var/lib/analyticsdb*

```

## Overview

Kubernetes is an open source platform for automating deployment, scaling, and operations of application containers across clusters of hosts, providing container-centric infrastructure. Kubernetes supports a pluggable framework called CNI for most of the basic network connectivity, including container pod addressing, network isolation, policy-based security, a gateway, SNAT, load-balancer, and service chaining capability for Kubernetes orchestration. Contrail supports CNI for integrating Contrail with the Kubernetes automation platform.

## Configuration

### IN THIS SECTION

- [Procedure | 130](#)

### Procedure

#### Step-by-Step Procedure

To provision Contrail CNI for Kubernetes, perform the following steps:

1. Git clone the contrail-docker repository.

```
git clone https://github.com/Juniper/contrail-docker.git -b R4.0
```

2. Change directory to **contrail-docker/kubernetes/manifests/**.

```
cd contrail-docker/kubernetes/manifests/
```

3. Edit the single yaml file. You can use any editor to edit the file.

- For Ubuntu 16.04 host OS:

```
vim contrail-host-ubuntu.yaml
```

- For CentOS 7 host OS:

```
vim contrail-host-centos.yaml
```

4. Edit the following variables in configmap and change it according to your setup. Mandatory variables that must be changed are `config_nodes`, `controller_nodes`, `analytics_nodes`, `analyticsdb_nodes`, and `api_server`. Refer to [Definable Input Variables While Provisioning Contrail for Kubernetes](#) for more information.

```
data:
global-config: |-
  [GLOBAL]
  cloud_orchestrator = kubernetes
  sandesh_ssl_enable = False
  enable_config_service = True
  enable_control_service = True
  enable_webui_service = True
  introspect_ssl_enable = False
  config_nodes = "192.0.2.2"
  controller_nodes = 192.0.2.2
  analytics_nodes = 192.0.2.2
  analyticsdb_nodes = 192.0.2.2
agent-config: |-
  [AGENT]
  compile_vrouter_module = True
  # Optional ctrl_data_network, if different from management
  # ctrl_data_network = "192.0.2.3/24"
kubemanager-config: |-
  [KUBERNETES]
  cluster_name = k8s-default
  cluster_project = {'domain': 'default-domain', 'project': 'default'}
  cluster_network = {}
  service_subnets = 192.0.2.4/12
  pod_subnets = 192.0.2.5/12
  api_server = 192.0.2.2
kubernetes-agent-config: |-
  [AGENT]
```

5. (Optional) If the setup is a single node setup then uncomment the following lines in `contrail-agent daemonset`.

```
#tolerations:
#- key: node-role.kubernetes.io/master
# operator: Exists
# effect: NoSchedule
```

6. Deploy Contrail using the following command.

```
kubectl apply -f contrail-host-ubuntu.yaml
```

7. Verify Contrail status on all Contrail containers. Look-up Contrail pod names using the following command.

```
kubectl get pods -n kube-system
contrail-analytics-9m545      1/1      Running   1      23h
contrail-analyticsdb-cpdjn    1/1      Running   1      23h
contrail-controller-gd5vl     1/1      Running   1      23h
contrail-kube-manager-82fcq   1/1      Running   1      23h
contrail-vrouter-agent-vwmbk  1/1      Running   1      23h
```

8. Check the `contrail-status` for all the pods, using the following command.

```
kubectl exec -it <contrail-pod-name> -n kube-system -- contrail-status
```

## Troubleshooting

### IN THIS SECTION

- | 133
- | 133
- | 133
- | 134

## Problem

To check if Contrail pods are running.

## Solution

Use the following command to list the Contrail pods:

```
kubectl get pods -n kube-system -o wide | grep contrail
```

## Problem

To ensure that contrail-agent pod is displayed.

## Solution

If the setup is a single node setup then uncomment the following lines in contrail-agent daemonset.

```
#tolerations:  
#- key: node-role.kubernetes.io/master  
# operator: Exists  
# effect: NoSchedule
```

## Problem

To bring up Contrail control plane pods on nodes other than the Kubernetes master.

## Solution

Label the node as `opencontrail.org/controller=true` using the following command:

```
kubectl label node <node-name> opencontrail.org/controller=true
```

## Problem

To delete the Contrail stack.

## Solution

Use the following command to delete the Contrail stack:

```
kubectl delete -f contrail-host-ubuntu.yaml
```

## RELATED DOCUMENTATION

| [Installing and Provisioning Containerized Contrail Controller for Kubernetes](#) | 112

## Using Kubernetes Helm to Provision Contrail

### IN THIS SECTION

- [Requirements](#) | 134
- [Overview](#) | 137
- [Configuration](#) | 137
- [Troubleshooting](#) | 139

Starting with Contrail Release 4.0.1, you can use Kubernetes Helm to provision Contrail.

## Requirements

This procedure requires the following minimum virtual machine and host specifications:

- 32 GB RAM
- Eight vCPUs
- 150 GB disk space

The supported software versions for using Kubernetes Helm for provisioning Contrail are:

- Kubernetes 1.6
- Docker Engine versions 1.11.0 to 1.13.0
- Helm 2.4 and greater
- Contrail Release 4.0.1

### Preparing for Installation

Before using Helm to provision Contrail with Kubernetes, ensure the following prerequisites are met:

#### 1. Stop the firewall service and delete all the iptable rules.

- For Ubuntu 16.04 host OS, use the following commands:

```
sudo service ufw stop
sudo iptables -F
```

- For CentOS 7 host OS, use the following commands:

```
sudo service firewalld stop
sudo iptables -F
```

#### 2. Ensure that the Kubernetes cluster is running. You can choose any method to install Kubernetes. For quick installation steps, use the following commands:

- Commands for Ubuntu 16.04 host OS:

```
sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl -y

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
sudo bash -c 'cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb http://apt.kubernetes.io/ kubernetes-xenial main
EOF'
sudo apt-get update -y
sudo apt-get install -y kubect1
sudo apt-get install -y kubelet
```

```

sudo apt-get install -y kubeadm
sudo apt-get install -y docker-engine

sudo kubeadm init

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

# If you have multi-node cluster then please copy paste the kubeadm join line
# on all the slave nodes from the stdout of sudo kubeadm init command.
# It looks like below
sudo kubeadm join --token fd554a.97d239c2234d0de352 192.0.2.0:6443

```

- Commands for CentOS7 host OS:

```

sudo bash -c 'cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
      https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
EOF'

sudo setenforce 0 || true

yum install -y kubelet kubeadm kubectl docker
systemctl enable docker && systemctl start docker
systemctl enable kubelet && systemctl start kubelet

echo 1 > /proc/sys/net/bridge/bridge-nf-call-iptables

sudo kubeadm init --kubernetes-version v1.7.4

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

# If you have multi-node cluster then please copy paste the kubeadm join line

```



```
# on all the slave nodes from the stdout of sudo kubeadm init command.
# It looks like below
sudo kubeadm join --token fd554a.97d239c2234d0de352 192.0.2.0:6443
```

### 3. Patch LivenessProbe and ReadinessProbe of kube-dns deployment using the following commands:

```
kubectl patch deploy/kube-dns --type json -p='[{"op": "replace", "path": "/spec/template/spec/containers/0/readinessProbe", "value": {"exec": {"command": ["wget", "-O", "-", "http://192.0.2.1:8081/readiness"]}}}]' -n kube-system
kubectl patch deploy/kube-dns --type json -p='[{"op": "replace", "path": "/spec/template/spec/containers/0/livenessProbe", "value": {"exec": {"command": ["wget", "-O", "-", "http://192.0.2.1:10054/healthcheck/kubedns"]}}}]' -n kube-system && kubectl patch deploy/kube-dns --type json -p='[{"op": "replace", "path": "/spec/template/spec/containers/1/livenessProbe", "value": {"exec": {"command": ["wget", "-O", "-", "http://192.0.2.1:10054/healthcheck/dnsmasq"]}}}]' -n kube-system && kubectl patch deploy/kube-dns --type json -p='[{"op": "replace", "path": "/spec/template/spec/containers/2/livenessProbe", "value": {"exec": {"command": ["wget", "-O", "-", "http://192.0.2.1:10054/metrics"]}}}]' -n kube-system
```

### 4. (Optional) If you are re-provisioning Contrail on the same Kubernetes cluster, ensure the following:

- Delete the configdb and analyticsdb data from the previous installation.

```
sudo rm -rf /var/lib/contrail*
sudo rm -rf /var/lib/configdb*
sudo rm -rf /var/lib/analyticsdb*
```

- Remove the vhost0 interface from all the nodes. Reboot the cluster to remove the vhost0 interface.

## Overview

Helm is a tool that helps package, install, and manage Kubernetes applications. Starting with Contrail 4.0.1, you can use Kubernetes Helm to provision Contrail.

## Configuration

### IN THIS SECTION

- [Procedure | 138](#)

## Procedure

### Step-by-Step Procedure

1. Download and install Kubernetes Helm.

```
export HELM_VERSION=v2.5.1
export TMP_DIR=$(mktemp -d)
curl -sSL https://storage.googleapis.com/kubernetes-helm/helm-${HELM_VERSION}-linux-
amd64.tar.gz | tar -zxv --strip-components=1 -C ${TMP_DIR}
sudo mv ${TMP_DIR}/helm /usr/local/bin/helm
rm -rf ${TMP_DIR}
```

2. Download Contrail-related charts and manifests. Ensure that the correct version of contrail-docker tag is checked out.

```
git clone https://github.com/Juniper/contrail-docker.git -b R4.0
```

3. Install Kubernetes Helm's tiller pods with the correct tiller version.

```
cd contrail-docker/kubernetes/manifests
kubectrl create -f tiller.yaml
kubectrl patch ds/tiller-ds --type json -p='[{"op": "replace", "path": "/spec/
updateStrategy/type", "value": "RollingUpdate"}]' -n kube-system && kubectrl set image ds/
tiller-ds tiller=gcr.io/kubernetes-helm/tiller:${HELM_VERSION} -n kube-system
```

4. Initialize the Kubernetes Helm client by using the following command

```
helm init --client-only
```

5. Edit the **values.yaml** file. Refer to [Definable Input Variables While Provisioning Contrail for Kubernetes](#) for more information.

```
cd ../helm
vi contrail/values.yaml
```

6. Install Contrail charts using the following command.

```
helm install --name deployment name path_to_chart
```

7. Verify Contrail status on all Contrail containers.

- Check the Contrail pod names using the following command.

```
kubectl get pods -n kube-system
contrail-analytics-9m545          1/1      Running   1          23h
contrail-analyticsdb-cpdjn       1/1      Running   1          23h
contrail-controller-gd5vl        1/1      Running   1          23h
contrail-kube-manager-82fcq      1/1      Running   1          23h
contrail-vrouter-agent-vwmbk     1/1      Running   1          23h
```

- Check the contrail-status for all the pods, using the following command.

```
kubectl exec -it contrail-pod-name -n kube-system -- contrail-status
```

## Troubleshooting

### IN THIS SECTION

- | [139](#)
- | [140](#)
- | [140](#)
- | [141](#)

### Problem

To check if Contrail pods are running.

## Solution

Use the following command to list the Contrail pods:

```
kubectl get pods -n kube-system -o wide | grep contrail
```

## Problem

To bring up Contrail control plane pods on nodes other than the Kubernetes master.

## Solution

Label the node as `opencontrail.org/controller=true` using the following command:

```
kubectl label node node-name opencontrail.org/controller=true
```

## Problem

To install Helm on locally loaded Contrail containers.

## Solution

1. Look-up the Contrail-image name and tags using the following command:

```
sudo docker images | grep contrail-controller
contrail-controller-ubuntu16.04          4.0.1.0-31
1cbed50707a7      3 days ago      1.614 GB
```

2. Edit the `contrail/values.yaml` file and change the respective image name under images using the `image-name:tag` format. For example:

```
images:
  controller: "docker.io/opencontrail/contrail-controller-ubuntu16.04:4.0.1."
```

Edit the file before installing Helm.

## Problem

To delete the Contrail stack.

## Solution

Use the following command to delete the Contrail stack:

```
kubectl delete -f contrail-host-ubuntu.yaml
```

## RELATED DOCUMENTATION

[Installing and Provisioning Containerized Contrail Controller for Kubernetes | 112](#)

[Provisioning Contrail CNI for Kubernetes | 126](#)

# Using VMware vCenter with Containerized Contrail, Release 4.0.1 and Greater

## IN THIS CHAPTER

- [Installing and Provisioning VMware vCenter with Containerized Contrail | 142](#)
- [Underlay Network Configuration for Containerized ContrailVM | 149](#)
- [Sample JSON Configuration Files for vCenter with Containerized Contrail 4.0.1 and Greater | 159](#)
- [Using the Contrail and VMWare vCenter User Interfaces to Manage the Network | 175](#)

## Installing and Provisioning VMware vCenter with Containerized Contrail

### IN THIS SECTION

- [Overview: Integrating Contrail 4.0.1 and Greater with vCenter Server | 143](#)
- [Different Modes of vCenter Integration with Contrail | 143](#)
- [vCenter-Only Mode | 143](#)
- [vCenter-as-Compute Mode | 144](#)
- [Preparing the Installation Environment | 145](#)
- [Installation for vCenter-Only Mode | 145](#)
- [Installing the vCenter-Only Components | 146](#)
- [Installation of vCenter-as-Compute Mode | 147](#)
- [Installing the vCenter-as-Compute Components | 148](#)
- [Verification | 148](#)
- [Adding Hosts or Nodes | 148](#)
- [Adding an ESXi Host to an Existing vCenter Cluster | 148](#)
- [Adding a vCenter Cluster to vCenter-as-Compute | 148](#)

## Overview: Integrating Contrail 4.0.1 and Greater with vCenter Server

This topic describes how to install and provision Contrail Release 4.0.1 and later.

The Contrail VMware vCenter solution has the following main components:

- Control and management that runs the following components as needed per Contrail system:
  - a. A VMware vCenter Server independent installation that is not managed by Juniper Networks Contrail. The Contrail software provisions vCenter with Contrail components and creates entities required to run Contrail.
  - b. The Contrail controller, including the configuration nodes, control nodes, analytics, database, and Web UI, which are installed, provisioned, and managed by Contrail software.
  - c. A VMware vCenter plugin provided with Contrail. Starting with Contrail Release 4.0.1 release, the `contrail-vcenter-plugin` runs in Docker container and provisioning of additional `contrail-vcenter-plugin` and `contrail-vcenter-compute` server roles are added
- VMware ESXi virtualization platforms forming the compute cluster, with Contrail data plane (vRouter) components running inside an Ubuntu-based virtual machine. The virtual machine, named *ContrailVM*, forms the compute personality while performing Contrail installs. The ContrailVM is set up and provisioned by Contrail. There is one ContrailVM running on each ESXi host.

## Different Modes of vCenter Integration with Contrail

The vCenter integrated Contrail solution has the following modes:

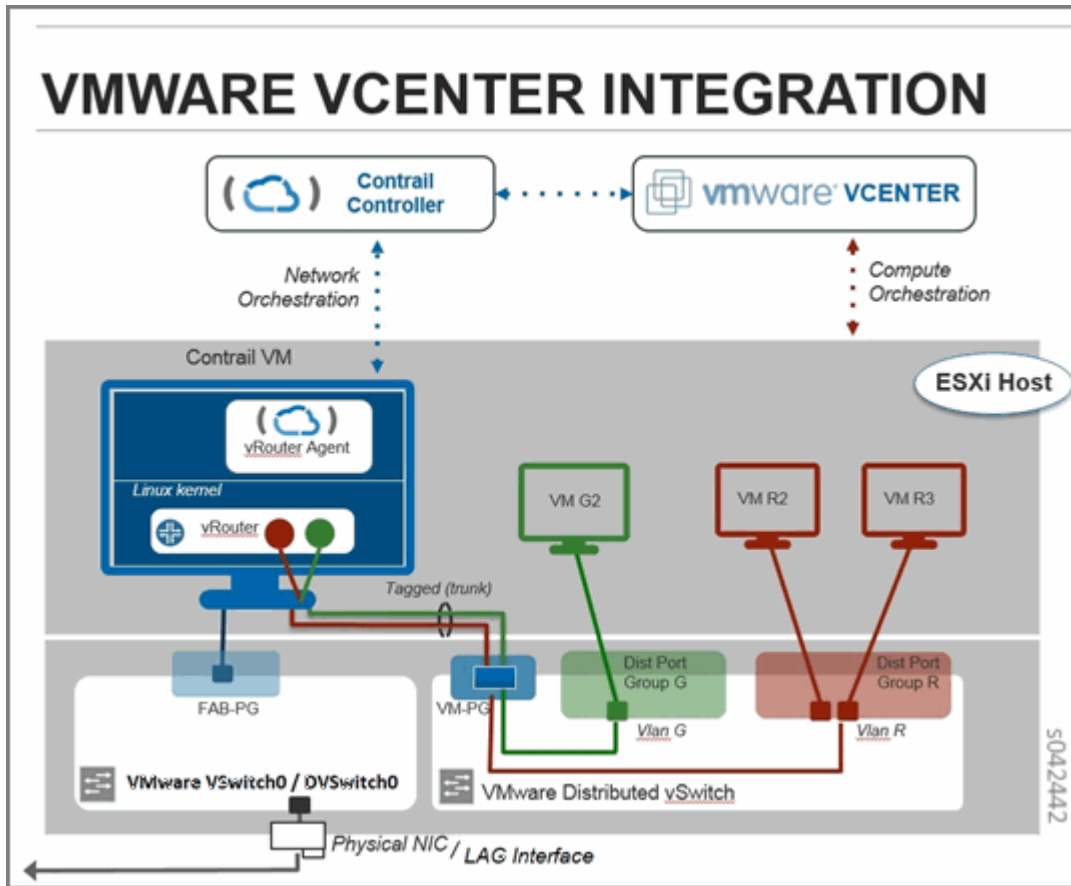
- vCenter-only
- vCenter-as-compute

### vCenter-Only Mode

In the vCenter-only mode, vCenter is the main orchestrator, and Contrail is integrated with vCenter for the virtual networking.

[Figure 20 on page 144](#) shows the Contrail vCenter-only solution.

Figure 20: Contrail vCenter-Only Solution



### vCenter-as-Compute Mode

In the vCenter-as-compute mode, OpenStack is the main orchestrator, and the vCenter cluster, along with the managed ESXi hosts, act as a Nova compute node to the OpenStack orchestrator.

Figure 21 on page 145 shows the Contrail vCenter-as-compute solution.





- A cluster of ESXi hosts with VMware
- The following software installation packages:
  - The `contrail-vcenter-docker_x.x.x.x-x_trusty.tgz` for Ubuntu 14.04
  - The `contrail-vcenter-docker_4.0.1.0-40_xenial.tgz` for Ubuntu 16.04.
- Tar file of the OVF image of ContrailVM
- Because a Contrail vRouter runs as a virtual machine on each ESXi host, it needs an IP address assigned from the same underlay network as the host, all of which must be specified appropriately in the server JSON configuration. Refer to the section "[Underlay Network Configuration for Containerized ContrailVM](#)" on page 149 for ContrailVM IP fabric connectivity.

## Installing the vCenter-Only Components

Follow the steps in this section to install the Contrail for vCenter-only components. See sample image, server, and cluster JSON configuration files for Contrail vCenter in "[Sample JSON Configuration Files for vCenter with Containerized Contrail 4.0.1 and Greater](#)" on page 159 for specific examples.

ContrailVM IP fabric connectivity can be configured in various ways. See "[Underlay Network Configuration for Containerized ContrailVM](#)" on page 149 for more information. The ContrailVM is created as part of Ansible-provisioning triggered from Server Manager. The IP address or MAC address for the ContrailVM is specified in the server JSON. Configure the DHCP server, allocating IP addresses to the cluster nodes, with static mapping of the MAC to IP address in DHCP server. The `server_manager` or `smlite` version of the Server Manager can be used for provisioning the Contrail cluster with a `vcenter-as-orchestrator` node. Ensure that the image is added to the Server Manager, and the servers and cluster configurations are added to the Server Manager. See *Installing Containerized Contrail Using Server Manager Lite (SM-Lite)*.

1. Ensure that `openstack_sku` is configured as "vcenter" in the image JSON. See "[Sample JSON Configuration Files for vCenter with Containerized Contrail 4.0.1 and Greater](#)" on page 159.
2. Ensure that `orchestrator` is set to "vcenter" in the cluster JSON. See "[Sample JSON Configuration Files for vCenter with Containerized Contrail 4.0.1 and Greater](#)" on page 159.
3. Add the image to Server Manager:
 

```
server-manager add image -f <path_to_image_json>
```
4. Add the cluster configuration to the Server Manager:
 

```
server-manager add cluster -f <path_to_cluster_json>
```
5. Add the server configuration to the Server Manager:
 

```
server-manager add server -f <path_to_server_json>
```
6. Use one of the following commands to provision Contrail clusters.

- To provision a Contrail cluster using server-manager:

```
server-manager provision -cluster_id <cluster_id> <contrail_image>
```

- To provision a Contrail cluster using SMLite:

```
Cd /opt/contrail/contrail_server_manager
```

```
./provision_containers.sh -j <json file path having image/cluster/server params>
```

When using SMLite installation, the single JSON file must include image, cluster, and server configurations. See *Installing Containerized Contrail Using Server Manager Lite (SM-Lite)*.

## Installation of vCenter-as-Compute Mode

This section lists the basic installation procedure and the assumptions and prerequisites necessary before starting the installation of any VMware vCenter-as-compute Contrail integration.

**NOTE:** To ensure you are using the correct versions of all software for your specific system, see the Supported Platforms section in the Release Notes for your release of Contrail .

### Installation: Assumptions and Prerequisites

The following assumptions and prerequisites are required for a successful installation of a VMware vCenter containerized Contrail integrated system:

- VMware vCenter Server 6.0 or 6.5
- A cluster of ESXi hosts with VMware
- The following software installation packages:
  - The `contrail-cloud-docker_x.x.x.x-x-mitaka_trusty.tgz` for Ubuntu14.04
  - The `contrail-cloud-docker_x.x.x.x-x-newton_xenial.tgz` for Ubuntu 16.04
  - Tar file of the OVF image of ContrailVM
- Because a Contrail vRouter runs as a virtual machine on each ESXi host, it needs an IP address assigned from the same underlay network as the host, all of which must be specified appropriately in the **server JSON** file. Refer to "[Underlay Network Configuration for Containerized ContrailVM](#)" on [page 149](#) for ContrailVM IP fabric connectivity.

For the vCenter-as-compute mode, an additional role of 'contrail-vcenter-compute' is required, specified as ['contrail-vcenter\_compute'] in the server JSON configuration in Server Manager. Nodes configured as contrail-vcenter\_compute act as the nova-compute nodes in this mode.

## Installing the vCenter-as-Compute Components

Ensure that the contrail-vcenter-compute role is defined in the server JSON. The installation or provisioning of the vcenter-as-compute cluster is the same as specified in the vcenter-as-orchestrator using Server Manager. Refer to the sample JSON files in "[Sample JSON Configuration Files for vCenter with Containerized Contrail 4.0.1 and Greater](#)" on page 159.

## Verification

When the provisioning step completes, run the contrail-status command on all containers to view a health check of the Contrail configuration and control components.

## Adding Hosts or Nodes

You can add some vCenter features to existing installations, including:

- Adding an ESXi host
- Adding a vCenter cluster

## Adding an ESXi Host to an Existing vCenter Cluster

You can provision and add an ESXi host to an existing vCenter cluster.

To add an ESXi host, add the server JSON configuration for the new contrail-compute server role in the Server Manager and run the following server-manager provision command for the cluster:

```
server-manager provision -cluster_id <cluster_id> <image_id>
```

**NOTE:** The server-manager provision command also works for server-manager smlite version.

## Adding a vCenter Cluster to vCenter-as-Compute

Use this procedure to add a vCenter cluster to a vCenter-as-compute system to an existing cluster. Ensure that you have provisioned and added all the ESXi hosts, as described in the procedure *Adding an ESXi Host to an Existing vCenter Cluster* procedure.

To set up and add a vCenter compute node, add the server JSON configuration for the new server with the `contrail-vcenter-compute` role in Server Manager and run the following `server-manager provision` command for the cluster:

```
server-manager provision -cluster_id <cluster_id> <image_id>
```

**NOTE:** The `server-manager provision` command also works for the `server-manager smlite` version.

## RELATED DOCUMENTATION

[Underlay Network Configuration for Containerized ContrailVM | 149](#)

[Sample JSON Configuration Files for vCenter with Containerized Contrail 4.0.1 and Greater | 159](#)

[Using the Contrail and VMWare vCenter User Interfaces to Manage the Network | 175](#)

## Underlay Network Configuration for Containerized ContrailVM

### IN THIS SECTION

- [Standard Switch Setup | 149](#)
- [Distributed Switch Setup | 151](#)
- [PCI Pass-Through Setup | 152](#)
- [SR-IOV Setup | 155](#)

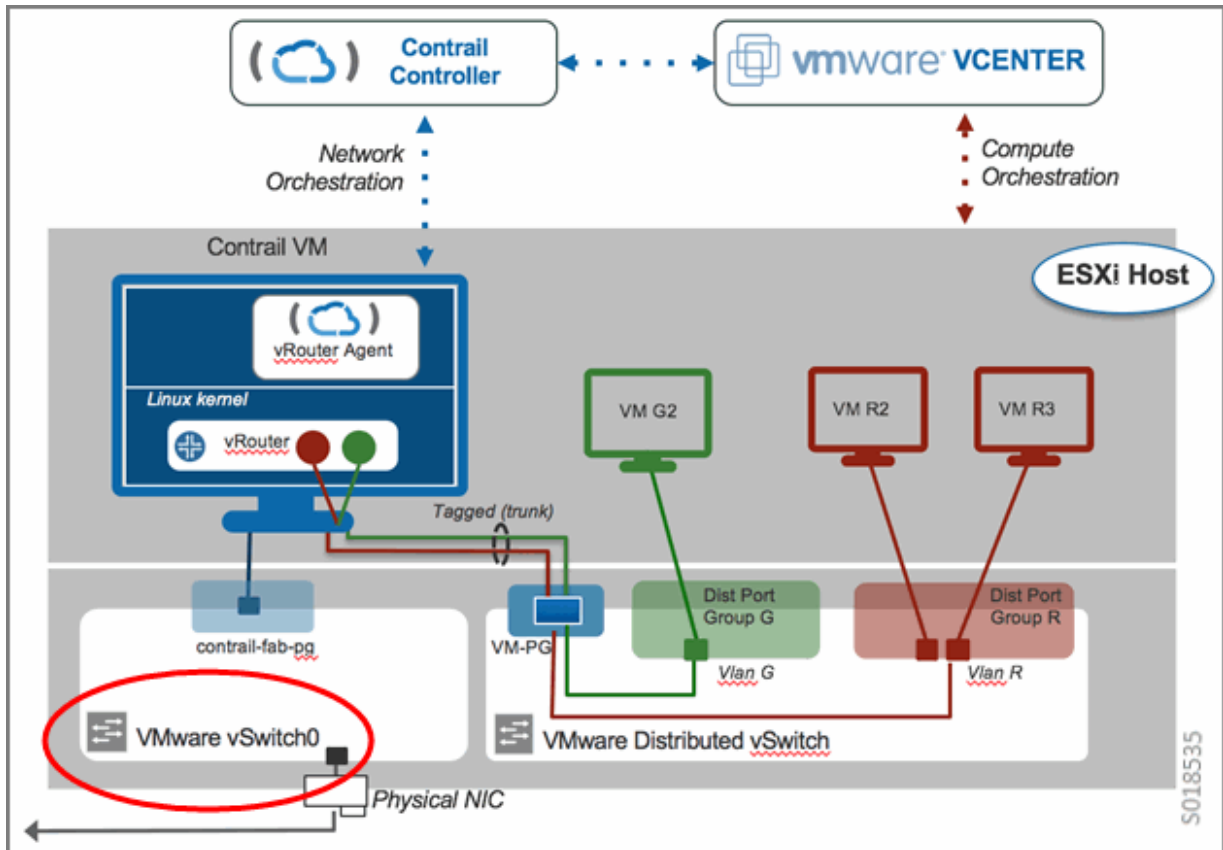
Starting with Contrail Release 4.0.1 and greater, vCenter can be used with containerized Contrail.

When using vCenter-as-compute with containerized Contrail, the ContrailVM can be configured in several different ways for the underlay (ip-fabric) connectivity:

### Standard Switch Setup

In the standard switch setup, the ContrailVM is provided an interface through the standard switch port group that is used for management and control data, see [Figure 22 on page 150](#).

Figure 22: Standard Switch Setup



To set up the ContrailVM in this mode, the standard switch and port group must be configured in the server JSON configuration.

If switch name is not configured, the default values of vSwitch0 are used for the standard switch.

The ContrailVM supports multiple NICs for management and control\_data interfaces. The management interface must have the DHCP flag as true and the control\_data interface can have DHCP set as false. When DHCP is set to false, the interface script is updated with the IP address as specified in the server JSON. Additional configuration such as static routes and bond interface can be configured in the server JSON in Server Manager.

The following is an example of server configuration in Server Manager with standard switch.

```
"contrail_vm": {
  "mgmt_switch": "vSwitch0",
  "mgmt_pg": "mgmt.-pg"
  "control_data_switch": "vSwitch1",
  "control_data_pg": "control-pg",
```



**NOTE:** The uplink can be a link aggregation group (LAG). If you use LAG, then DVS and LAG need to be preconfigured.

The following is an example distributed switch configuration in cluster JSON.

```

"vcenter_servers": [
  {
    "server1": {
      "datacenters": {
        "i27_datacenter11": {
          "dv_switch_control_data": {
            "dv_port_group_control_data": {
              "dv_portgroup_name": "pg_name",
              "number_of_ports": "3",
              "uplink": "vmnic1"
            },
            "dv_switch_name": "dvs_name"
          },
          "dv_switch_mgmt": {
            "dv_port_group_mgmt": {
              "dv_portgroup_name": "",
              "number_of_ports": "",
              "uplink": ""
            },
            "dv_switch_name": ""
          }
        }
      }
    }
  }
]

```

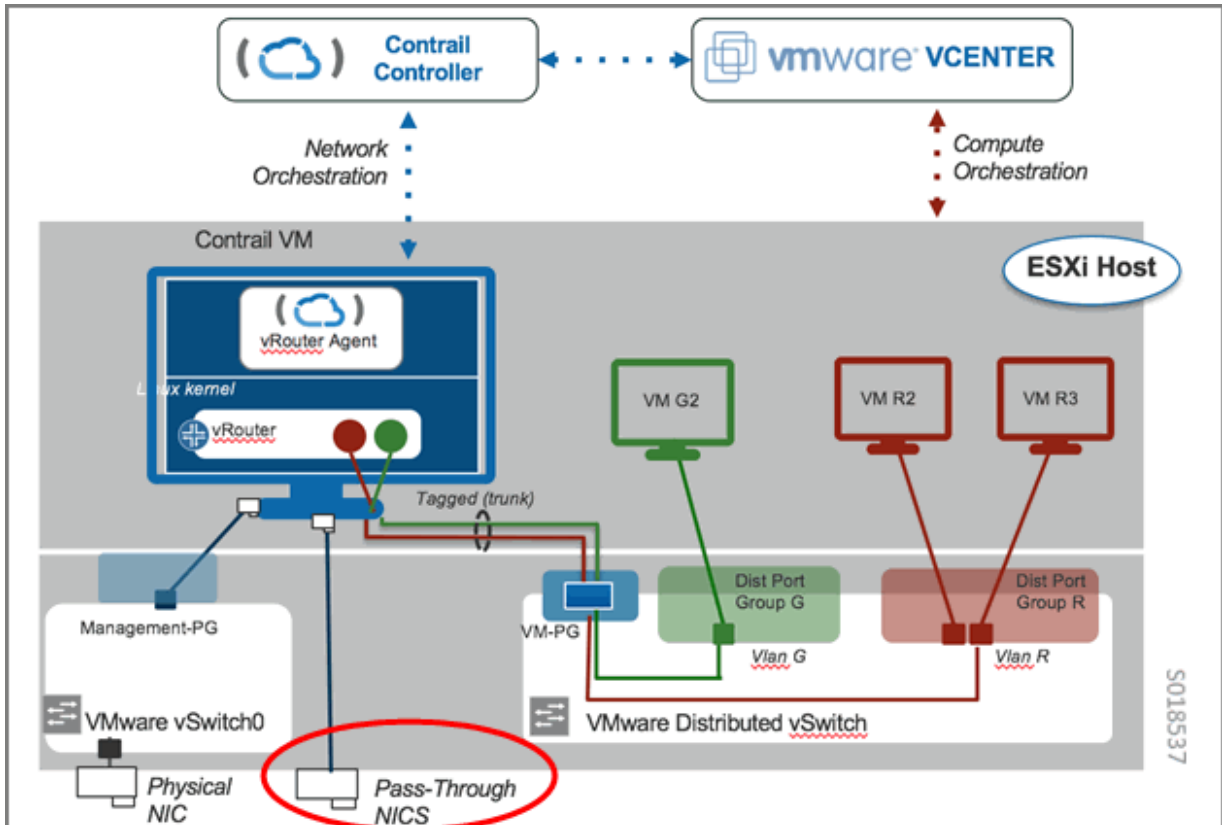
## PCI Pass-Through Setup

PCI pass-through is a virtualization technique in which a physical Peripheral Component Interconnect (PCI) device is directly connected to a virtual machine, bypassing the hypervisor. Drivers in the VM can directly access the PCI device, resulting in a high rate of data transfer.



In the pass-through setup, the ContrailVM is provided management and control data interfaces. Pass-through interfaces are used for control data. [Figure 24 on page 153](#) shows a PCI pass-through setup with a single control\_data interface.

**Figure 24: PCI Pass-Through with Single Control Data Interface**



When setting up the ContrailVM with pass-through interfaces, upon provisioning ESXi hosts in the installation process, the PCI pass-through interfaces are exposed as Ethernet interfaces in the ContrailVM, and are identified in the control\_data device field.

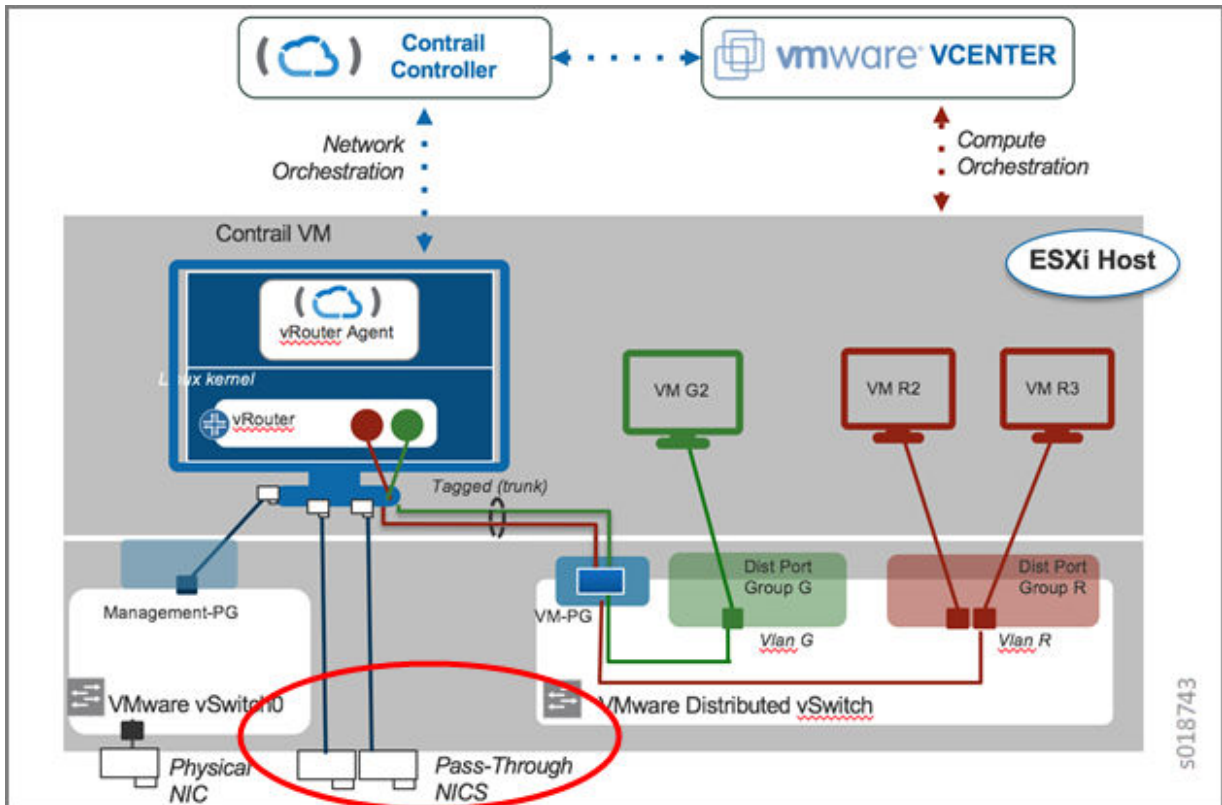
The following is an example PCI pass-through configuration with a single control\_data interface:

```
'contrail_vm': {
  "pci_devices": {
    "nics": ["04:00.0"],
  },
  "vmdk": "/root/vmdk_new/vmdk.tar"
```

}

Figure 25 on page 154 shows a PCI pass-through setup with a bond\_control data interface, which has multiple pass-through NICs.

Figure 25: PCI Pass-Through Setup with Bond Control Interface



Update the ContrailVM section in server JSON configuration with pci\_devices as shown in the following example. Refer to the Server Manager documentation for bond interface-configuration in server JSON configuration.

```
"contrail_vm": {
  "pci_devices": {
    "nics": ["04:00.0", "04:00.1"]
  }
}
```

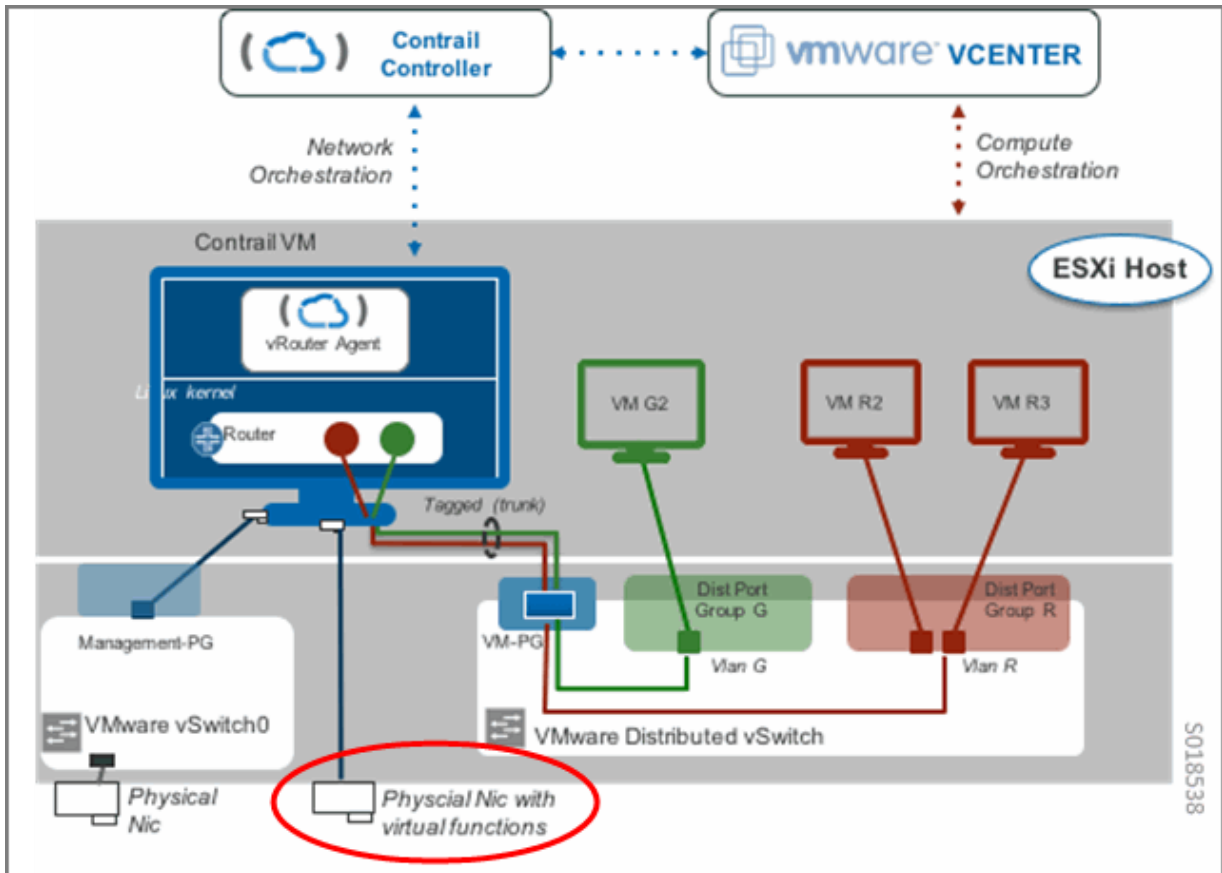
```
"vmdk": "/root/vmdk_new/vmdk.tar"
},
```

## SR-IOV Setup

A single root I/O virtualization (SR-IOV) interface allows a network adapter device to separate access to its resources among various hardware functions.

In the SR-IOV setup, the ContrailVM is provided management and control data interfaces. SR-IOV interfaces are used for control data. See [Figure 26 on page 155](#).

Figure 26: SR-IOV Setup



In VMware, the port-group is mandatory for SR-IOV interfaces because the ability to configure the networks is based on the active policies for the port holding the virtual machines. For more information, refer to VMware's [SR-IOV Component Architecture and Interaction](#).

The port-group is created as part of provisioning; however, before the provisioning, the distributed virtual switch (DVS) for the port-group should be created by the user.



The cluster configuration:

```

"vcenter_servers": [
  {
    "server1": {
      "datacenters": {
        "i27_datacenter11": {
          "dv_switch_sr_iov": {
            "dv_port_group_sr_iov": {
              "dv_portgroup_name": "",
              "number_of_ports": "",
              "uplink": ""
            },
            "dv_switch_name": ""
          }
        }
      }
    }
  }
]

```

The server configuration:

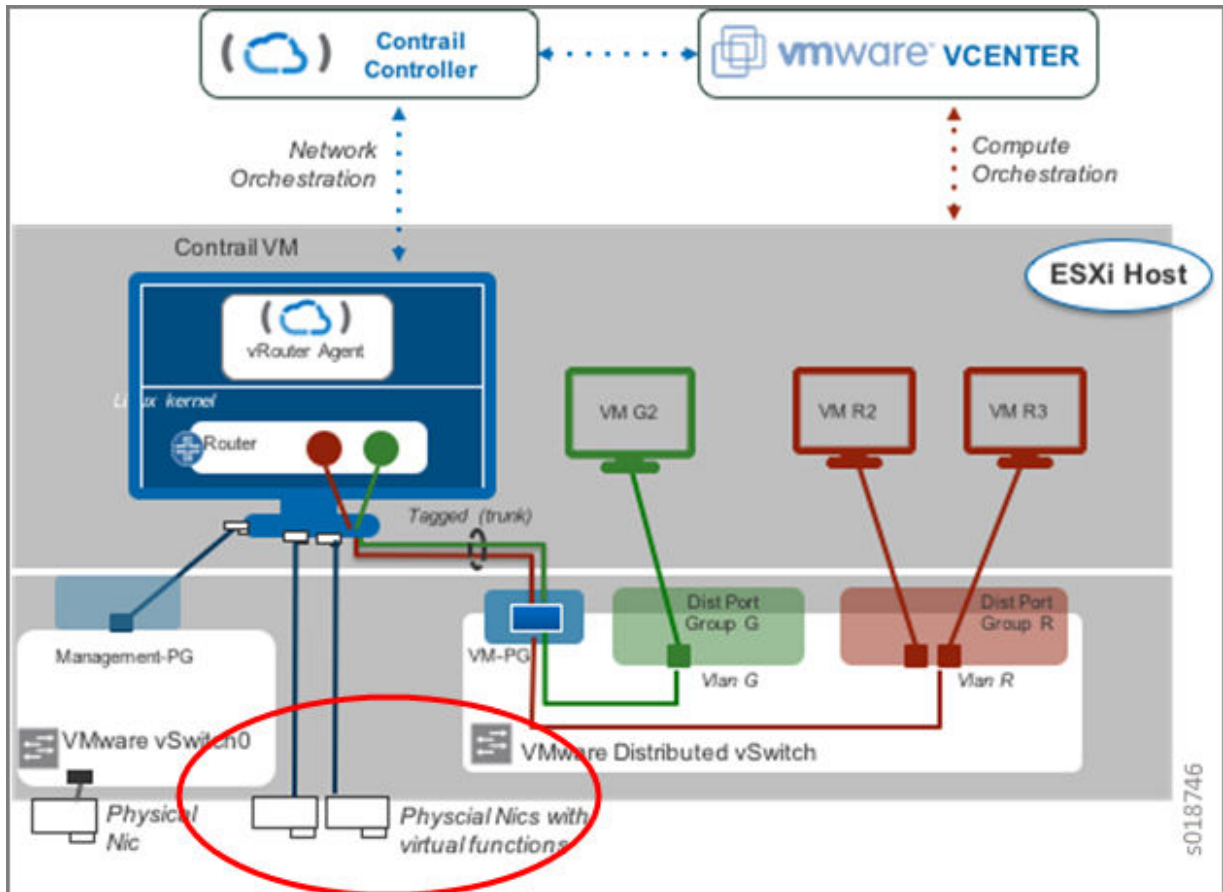
```

"contrail_vm": {
  "sr_iov_nics": {
    "nics": ["vmmnic0"]
  }
  "vmdk": "/root/vmdk_new/vmdk.tar"
},

```

Figure 28 on page 158 shows an SR-IOV configuration with a bond control\_data interface, which has multiple SR-IOV NICs.

Figure 28: SR-IOV With Bond Control Data Interface



For Bond interface-configuration specify multiple NICs in `sr_iov_nics`, and add required configuration for multi-interface and bond configuration in server JSON configuration as specified in Server Manager documentation.

The cluster configuration:

```
"vcenter_servers": [
  {
    "server1": {
      "datacenters": {
        "i27_datacenter11": {
          "dv_switch_sr_iov": {
            "dv_port_group_sr_iov": {
              "dv_portgroup_name": "",
              "number_of_ports": "",
              "uplink": ""
            }
          }
        }
      }
    }
  }
]
```

```

    },
    "dv_switch_name": ""
  }
}
}
}
]

```

The server configuration:

```

"contrail_vm": {
  "sr_iov_nics": {
    "nics": ["vmnic0", "vmnic1"]
  }
  "vmdk": "/root/vmdk_new/vmdk.tar"
},

```

## RELATED DOCUMENTATION

[Installing and Provisioning VMware vCenter with Containerized Contrail | 142](#)

[Sample JSON Configuration Files for vCenter with Containerized Contrail 4.0.1 and Greater | 159](#)

[Using the Contrail and VMware vCenter User Interfaces to Manage the Network | 175](#)

## Sample JSON Configuration Files for vCenter with Containerized Contrail 4.0.1 and Greater

### IN THIS SECTION

- [Sample Image JSON for vCenter-Only Mode | 160](#)
- [Sample Cluster JSON for vCenter-Only Mode | 160](#)
- [Sample Server JSON for vCenter-Only Mode | 162](#)
- [Sample JSON for vCenter-only with High Availability | 164](#)
- [Sample Image JSON for vCenter-as-Compute Mode | 169](#)

- [Sample Cluster JSON for vCenter-as-Compute Mode | 170](#)
- [Sample Server JSON for vCenter-as-Compute Mode | 171](#)

Starting with Contrail Release 4.0.1, vCenter can be used with containerized Contrail. This section presents sample JSON files that can be used to provision containerized Contrail using Server Manager. Be sure to replace parameters in the samples with values specific to your system.

### Sample Image JSON for vCenter-Only Mode

```
{
  "image": [
    {
      "category": "package",
      "id": "contrail_vc_orch",
      "path": "/root/contrail-vcenter-docker_4.0.1.0-50_trusty.tgz",
      "type": "contrail-ubuntu-package",
      "parameters": {"openstack_sku": "vcenter"},
      "version": "4.0.1.0-50"
    }
  ]
}
```

### Sample Cluster JSON for vCenter-Only Mode

```
{
  "cluster": [
    {
      "id": "cluster-esxi-new",
      "parameters": {
        "provision": {
          "contrail": {
          },
          "contrail_4": {
            "cloud_orchestrator": "vcenter",
            "vcenter_servers": [
              {
                "server1": {

```





```

    }
  }
]
}

```

## Sample Server JSON for vCenter-Only Mode

```

{
  "server": [
    {
      "cluster_id": "cluster-esxi-new",
      "contrail": {
        "control_data_interface": "eth1"
      },
      "domain": "contrail.juniper.net",
      "host_name": "controller1-b7s28",
      "id": "controller1-b7s28",
      "network": {
        "interfaces": [
          {
            "dhcp": true,
            "ip_address": "10.xx.xx.59",
            "mac_address": "00:50:56:a6:47:72",
            "default_gateway": "10.xx.xx.254",
            "name": "eth0"
          },
          {
            "dhcp": false,
            "ip_address": "192.xxx.xxx.101/24",
            "mac_address": "00:50:56:a6:4d:38",
            "name": "eth1"
          }
        ],
        "management_interface": "eth0"
      },
      "password": "<password>",
      "roles": [
        "contrail-controller",
        "contrail-analytics",
        "contrail-analyticsdb",
        "contrail-vcenter-plugin"
      ]
    }
  ]
}

```

```

    ]
  },
  {
    "cluster_id": "cluster-esxi-new",
    "contrail": {
      "control_data_interface": "eth1"
    },
    "domain": "contrail.juniper.net",
    "host_name": "computevm-b7s28",
    "id": "computevm-b7s28",
    "ip_address": "10.xx.xx.54",
    "network": {
      "interfaces": [
        {
          "dhcp": true,
          "ip_address": "10.xx.xx.54",
          "mac_address": "00:50:56:05:ba:ba",
          "default_gateway": "10.xx.x.254",
          "name": "eth0"
        },
        {
          "dhcp": false,
          "ip_address": "192.xxx.xxx.28/24",
          "mac_address": "00:50:56:05:bb:bb",
          "name": "eth1"
        }
      ],
      "management_interface": "eth0"
    },
    "parameters": {
      "esxi_parameters": {
        "cluster": "kp_cluster11",
        "contrail_vm": {
          "control_data_pg": "control_data",
          "control_data_switch": "vSwitch1",
          "mgmt_pg": "mgmt-pg",
          "vmdk": "/root/vmdk/vmdk.tar"
        },
        "datacenter": "kp_datacenter11",
        "datastore": "datastore1",
        "name": "10.xx.xx.28",
        "password": "<password>",
        "username": "root",

```

```

        "validate_certs": false,
        "vcenter_server": "server1"
    }
},
"password": "<password>",
"roles": [
    "contrail-compute"
]
}
]
}

```

### Sample JSON for vCenter-only with High Availability

```

{
  "cluster": [{
    "id": "vcenter_cluster",
    "parameters": {
      "provision": {
        "contrail_4": {
          "cloud_orchestrator": "vcenter",
          "vcenter_servers": [
            {
              "<your.server.company.net>": {
                "datacenters": {
                  "Cluster": {
                    "datacenter_mtu": 1500,
                    "dv_switches": [
                      {
                        "clusternames": [
                          "contrail-csg2"
                        ],
                        "dv_port_group": {
                          "dv_portgroup_name": "dv-portgroup",
                          "number_of_ports": "3"
                        },
                        "dv_switch_name": "dvs-switch"
                      }
                    ]
                  }
                }
              }
            }
          ]
        }
      }
    }
  ]
},

```

```

        "hostname": "10.xx.xx.xxx",
        "password": "<password>!",
        "username": "administrator@vsphere.local",
        "validate_certs": false
    }
  ]],
  "ha": {
    "contrail_external_vip": "10.12.34.1",
    "contrail_internal_vip": "10.12.34.1"
  }
}
}
],
"server": [{
  "cluster_id": "vcenter_cluster",
  "contrail": {},
  "domain": "<domain-name.company.net>",
  "host_name": "<hostname>",
  "id": "<hostname>",
  "network": {
    "interfaces": [{
      "default_gateway": "10.xx.xx.254",
      "dhcp": true,
      "ip_address": "10.xx.xx.152/24",
      "mac_address": "00:0c:29:99:77:45",
      "name": "eth0"
    }],
    "management_interface": "eth0"
  },
  "parameters": {},
  "password": "<password>",
  "roles": [
    "contrail-controller",
    "contrail-analytics",
    "contrail-analyticsdb",
    "contrail-vcenter-plugin"
  ]
},
{
  "cluster_id": "vcenter_cluster",

```

```

"contrail": {},
"domain": "<domain-name.company.net>",
"host_name": "<hostname>",
"id": "<hostname>",
"network": {
  "interfaces": [{
    "default_gateway": "10.xx.xx.254",
    "dhcp": true,
    "ip_address": "10.xx.xx.155/24",
    "mac_address": "00:0c:29:38:6d:5d",
    "name": "eth0"
  }],
  "management_interface": "eth0"
},
"parameters": {},
"password": "<password>",
"roles": [
  "contrail-controller",
  "contrail-analytics",
  "contrail-analyticsdb",
  "contrail-vcenter-plugin"
]
},
{
  "cluster_id": "vcenter_cluster",
  "contrail": {},
  "domain": "<domain-name.company.net>",
  "host_name": "<hostname>",
  "id": "<id-name>",
  "network": {
    "interfaces": [{
      "default_gateway": "10.xx.xx.254",
      "dhcp": true,
      "ip_address": "10.xx.xx.156/24",
      "mac_address": "00:0c:29:ea:37:33",
      "name": "eth0"
    }],
    "management_interface": "eth0"
  },
  "parameters": {},
  "password": "<password>",
  "roles": [
    "contrail-controller",

```

```

        "contrail-analytics",
        "contrail-analyticsdb",
        "contrail-vcenter-plugin"
    ]
},
{
    "cluster_id": "vcenter_cluster",
    "contrail": {},
    "domain": "<domain-name.company.net>",
    "host_name": "<hostname>",
    "id": "<id-name>",
    "network": {
        "interfaces": [{
            "default_gateway": "10.xx.xx.254",
            "dhcp": true,
            "ip_address": "10.xx.xx.1/24",
            "mac_address": "08:9e:01:93:cb:e0",
            "name": "em1"
        }],
        "management_interface": "em1"
    },
    "parameters": {},
    "password": "<password>",
    "roles": [
        "contrail-lb"
    ]
},
{
    "cluster_id": "vcenter_cluster",
    "contrail": {},
    "domain": "<domain-name.company.net>",
    "host_name": "<hostname>",
    "id": "<id-name>",
    "network": {
        "interfaces": [{
            "default_gateway": "10.xx.xx.254",
            "dhcp": true,
            "ip_address": "10.xx.xx.153/24",
            "mac_address": "00:50:56:05:ba:b5",
            "name": "eth0"
        }],
        "management_interface": "eth0"
    },
},

```

```

"parameters": {
  "esxi_parameters": {
    "cluster": "contrail-csg2",
    "contrail_vm": {
      "mgmt_pg": "mgmt-pg",
      "vmdk": "/var/tmp/ContrailVM1604-ovf.tar"
    },
    "datacenter": "Cluster",
    "datastore": "datastore1",
    "name": "10.xx.xx.41",
    "password": "<password>",
    "username": "root",
    "validate_certs": false,
    "vcenter_server": "<hostname.domain-name.company.net>"
  }
},
"password": "<password>",
"roles": [
  "contrail-compute"
]
},
{
  "cluster_id": "vcenter_cluster",
  "contrail": {},
  "domain": "<domain-name.company.net>",
  "host_name": "<hostname>",
  "id": "<id-name>",
  "network": {
    "interfaces": [{
      "default_gateway": "10.xx.xx.254",
      "dhcp": true,
      "ip_address": "10.xx.xx.154/24",
      "mac_address": "00:50:56:05:ce:c5",
      "name": "eth0"
    }],
    "management_interface": "eth0"
  },
  "parameters": {
    "esxi_parameters": {
      "cluster": "contrail-csg2",
      "contrail_vm": {
        "mgmt_pg": "mgmt-pg",
        "vmdk": "/var/tmp/ContrailVM1604-ovf.tar"
      }
    }
  }
}

```



```

        },
        "datacenter": "Cluster",
        "datastore": "datastore2",
        "name": "10.xx.xx.42",
        "password": "<password>",
        "username": "root",
        "validate_certs": false,
        "vcenter_server": "<hostname.domain-name.company.net>"
    }
},
"password": "<password>",
"roles": [
    "contrail-compute"
]
}
],
"image": [{
    "category": "package",
    "id": "contrail_vc_orch",
    "path": "/var/tmp/contrail-vcenter-docker_4.1.0.0-33_xenial.tgz",
    "type": "contrail-ubuntu-package",
    "parameters": {
        "openstack_sku": "vcenter"
    },
    "version": "4.1.0.0-33"
}]
}

```

### Sample Image JSON for vCenter-as-Compute Mode

```

{
  "image": [
    {
      "category": "package",
      "id": "contrail_vc_compute",
      "path": "/root/contrail-cloud-docker_4.0.1.0-39-mitaka_trusty.tgz",
      "type": "contrail-ubuntu-package",
      "version": "4.0.1.0-39"
    }
  ]
}

```

## Sample Cluster JSON for vCenter-as-Compute Mode

```
{
  "cluster": [
    {
      "id": "vcenter_five_node",
      "parameters": {
        "provision": {
          "contrail": {
          },
          "contrail_4": {
            "vcenter_servers": [
              {
                "server1": {
                  "datacenters": {
                    "vc_datacenter1": {
                      "datacenter_mtu": 1500,
                      "dv_switch_control_data": {
                        "dv_port_group_control_data": {
                          "dv_portgroup_name": "",
                          "number_of_ports": "",
                          "uplink": ""
                        },
                      },
                      "dv_switch_name": ""
                    },
                  },
                  "dv_switch_mgmt": {
                    "dv_port_group_mgmt": {
                      "dv_portgroup_name": "",
                      "number_of_ports": "",
                      "uplink": ""
                    },
                  },
                  "dv_switch_name": ""
                },
              },
            ],
            "dv_switches": [
              {
                "clusternames": [
                  "vcenter1"
                ],
                "dv_port_group": {
                  "dv_portgroup_name": "guest_dvs_pg",
                  "number_of_ports": "3"
                }
              },
            ],
          },
        },
      },
    },
  ],
}
```



```

    ],
  },
  "password": "<password>",
  "roles": [
    "contrail-controller",
    "contrail-analytics",
    "contrail-analyticsdb"
  ]
},
{
  "cluster_id": "vcenter_five_node",
  "contrail": {},
  "domain": "contrail.juniper.net",
  "host_name": "nk-vm2",
  "id": "nk-vm2",
  "network": {
    "interfaces": [
      {
        "default_gateway": "10.xx.x.254",
        "ip_address": "10.xx.x.203/24",
        "mac_address": "52:54:DE:AD:BD:A2",
        "name": "eth1"
      }
    ]
  },
  "password": "<password>",
  "roles": [
    "openstack"
  ]
},
{
  "cluster_id": "vcenter_five_node",
  "contrail": {},
  "domain": "contrail.juniper.net",
  "host_name": "nk-vm3",
  "id": "nk-vm3",
  "network": {
    "interfaces": [
      {
        "default_gateway": "10.xx.x.254",
        "ip_address": "10.xx.x.204/24",
        "mac_address": "52:54:DE:AD:BD:A3",
        "name": "eth1"
      }
    ]
  }
}

```

```

        }
    ]
},
"password": "<password>",
"roles": [
    "contrail-vcenter-plugin"
]
},
{
    "cluster_id": "vcenter_five_node",
    "contrail": {},
    "domain": "contrail.juniper.net",
    "host_name": "nk-vm4",
    "id": "nk-vm4",
    "network": {
        "interfaces": [
            {
                "default_gateway": "10.xx.x.254",
                "ip_address": "10.xx.x.205/24",
                "mac_address": "52:54:DE:AD:BD:A4",
                "name": "eth1"
            }
        ]
    },
    "password": "<password>",
    "roles": [
        "contrail-vcenter-compute"
    ]
},
{
    "cluster_id": "vcenter_five_node",
    "contrail": {},
    "domain": "contrail.juniper.net",
    "host_name": "computevm-b7s27",
    "id": "computevm-b7s27",
    "network": {
        "interfaces": [
            {
                "default_gateway": "10.xx.xx.254",
                "dhcp": true,
                "ip_address": "10.xx.xx.57/24",
                "mac_address": "00:50:56:AD:BD:A5",
                "name": "eth1"
            }
        ]
    }
}

```

```

    }
  ]
},
"parameters": {
  "esxi_parameters": {
    "cluster": "vcenter1",
    "contrail_vm": {
      "mgmt_pg": "mgmt-pg",
      "mode": "vcenter",
      "vmdk": "/root/vmdk/vmdk.tar"
    },
    "datacenter": "vc_datacenter1",
    "datastore": "datastore1",
    "name": "10.xx.xx.27",
    "password": "<password>",
    "username": "root",
    "validate_certs": false,
    "vcenter_server": "server1"
  },
  "password": "<password>",
  "roles": [
    "contrail-compute"
  ]
}
]
}

```

## RELATED DOCUMENTATION

[Installing and Provisioning VMware vCenter with Containerized Contrail](#) | 142

[Underlay Network Configuration for Containerized ContrailVM](#) | 149

## Using the Contrail and VMWare vCenter User Interfaces to Manage the Network

### IN THIS SECTION

- [Overview: User Interfaces for Contrail Integration with VMware vCenter | 175](#)
- [Feature Configuration for Contrail vCenter | 176](#)
- [Creating a Virtual Machine | 185](#)
- [Configuring the vCenter Network in Contrail UI | 195](#)

You can install Contrail to work with the VMware vCenter Server in various vSphere environments and use the Contrail user interface and the vCenter user interface to configure and manage the integrated Contrail system.

### Overview: User Interfaces for Contrail Integration with VMware vCenter

This topic shows how to use the Contrail user interface and the vCenter user interface to configure and manage features of a Contrail VMware integrated system.

The two user interfaces are available after installing the integrated Contrail system, see [Installing and Provisioning VMware vCenter with Contrail](#) .

When Contrail is integrated with VMware vCenter, the following two user interfaces are used to manage and configure features of the system.

### Contrail Administration User Interface

The Contrail UI is an administrator's user interface. It provides a view of all components managed by the Contrail controller.

To log in to the Contrail UI, use your Contrail server main IP address URL as follows:

```
https://<Contrail IP>:8143
```

Then log in using your registered Contrail account administrator credentials.

## Contrail vCenter User Interface

The Contrail vCenter user interface (vCenter UI) is a subset of the Contrail administration UI. The Contrail vCenter UI provides a view of all of the virtual components within a Contrail vCenter project.

**NOTE:** This is applicable only to the vCenter-only mode.

To access the login page for the Contrail vCenter UI, use your Contrail IP address URL as follows:

`https://<Contrail URL>:8143/vcenter`

Then use the vCenter registered account log in name and password to access the Contrail vCenter UI.

Upon successful login, the Contrail vCenter user interface is displayed, as in the following example.



## Feature Configuration for Contrail vCenter

This section shows how to use the Contrail UI and the Contrail vCenter UI to configure features for the Contrail vCenter integrated system.

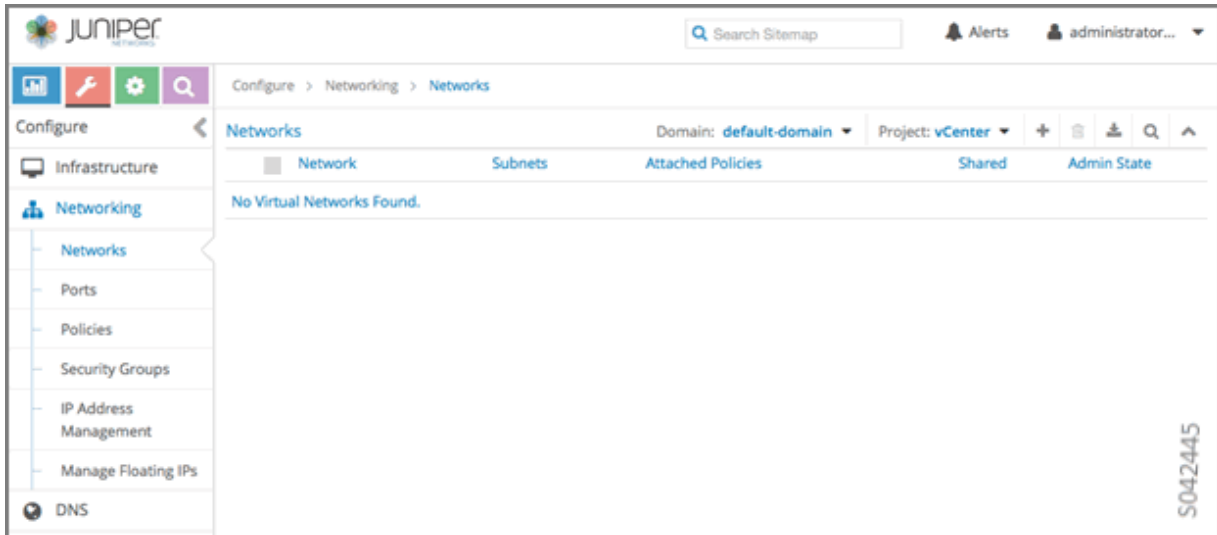
### Creating a Virtual Network

This section describes how to create a virtual network using the Contrail UI and the Contrail vCenter UI.

#### Create Virtual Network – Contrail UI

After logging in to the Contrail UI, select **Configure > Networking > Networks** to access the Networks window.



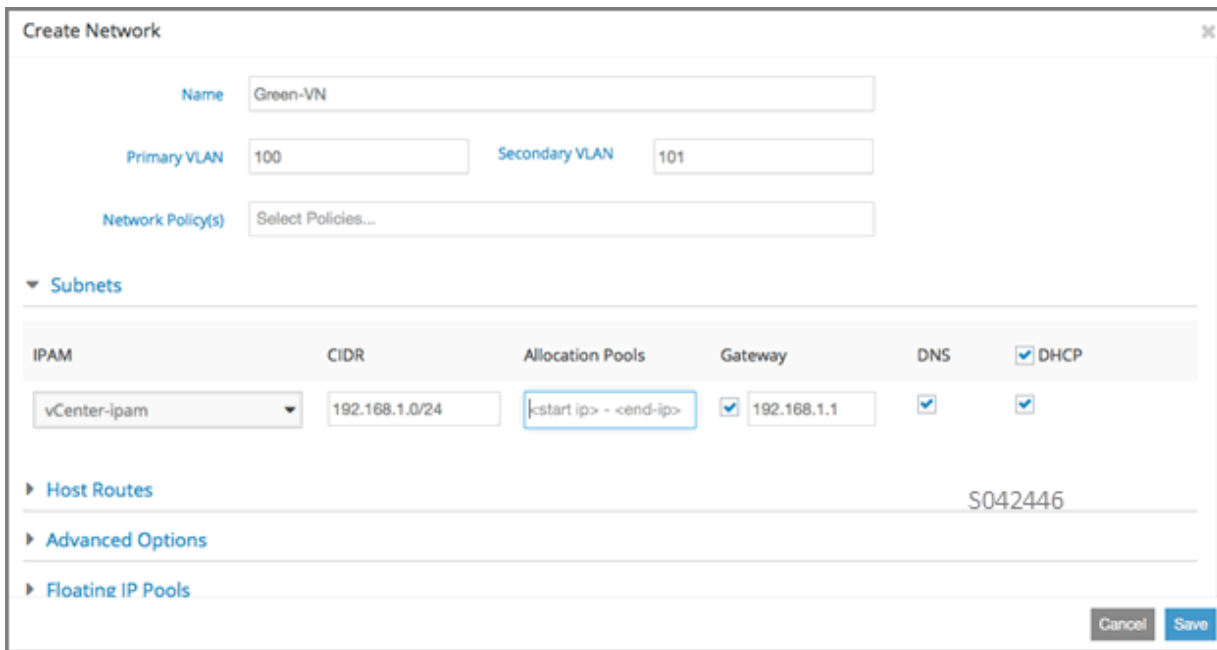


At Networks, click the plus icon ( +)to access the **Create Network** window.

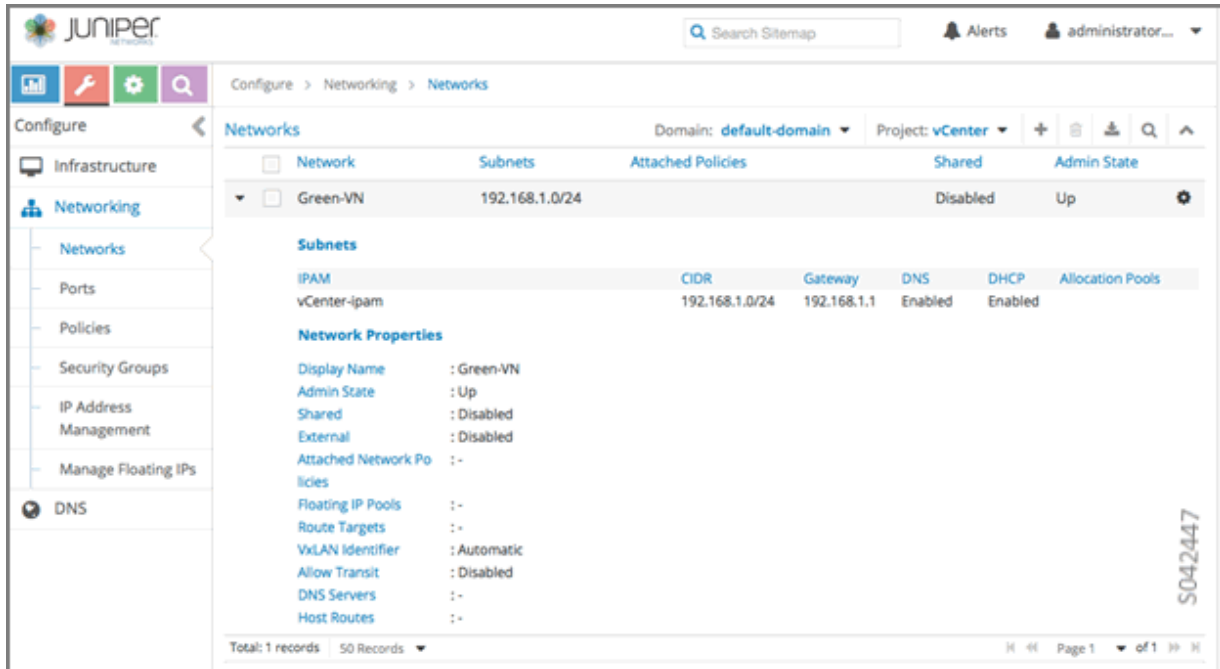
Complete the fields in the **Create Network** window. Provide a **Primary VLAN** value and a **Secondary VLAN** value as part of a **Private VLAN** configuration. **Private VLAN** pairs are configured on a Distributed Virtual Switch. Select the values for the Primary and Secondary VLANs from one of the configured, isolated, private-vlan pairs.

The following figure shows the creation of a virtual network named Green-VN.

Click **Save** to create the virtual network.



The virtual network just created (Green-VN) is displayed with its details, as in the following figure.



## Create Virtual Networks – Contrail vCenter UI

You can also create a virtual network in the vCenter UI, and view and manage it from either the vCenter UI or the Contrail UI.

**NOTE:** This is applicable only to the vCenter-only mode.

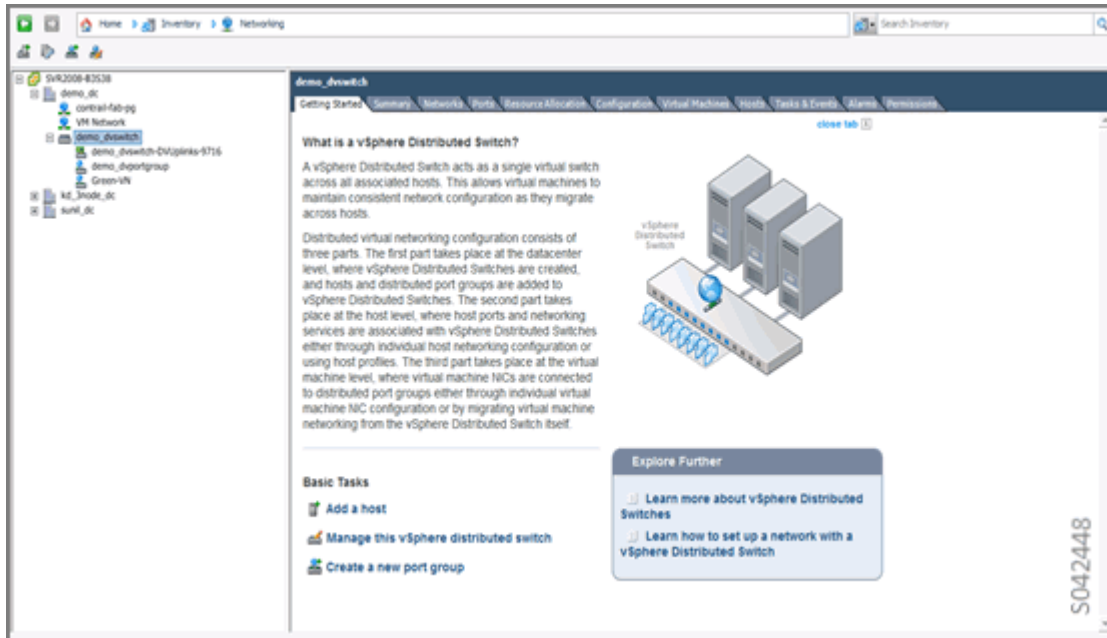
In vCenter, a virtual network is called a port group, which is part of a distributed switch.

Log in to the vCenter client UI (<https://<Contrail URL>:9443/vsphere-client>).

To start creating a virtual network (distributed port group), click the distributed virtual switch (**dvswitch**) on the left panel.

The following figure shows the *demo\_dvswitch* has been selected for this example.

To create a virtual network (vCenter port group), at the bottom of the window, click **Create a new port group**.

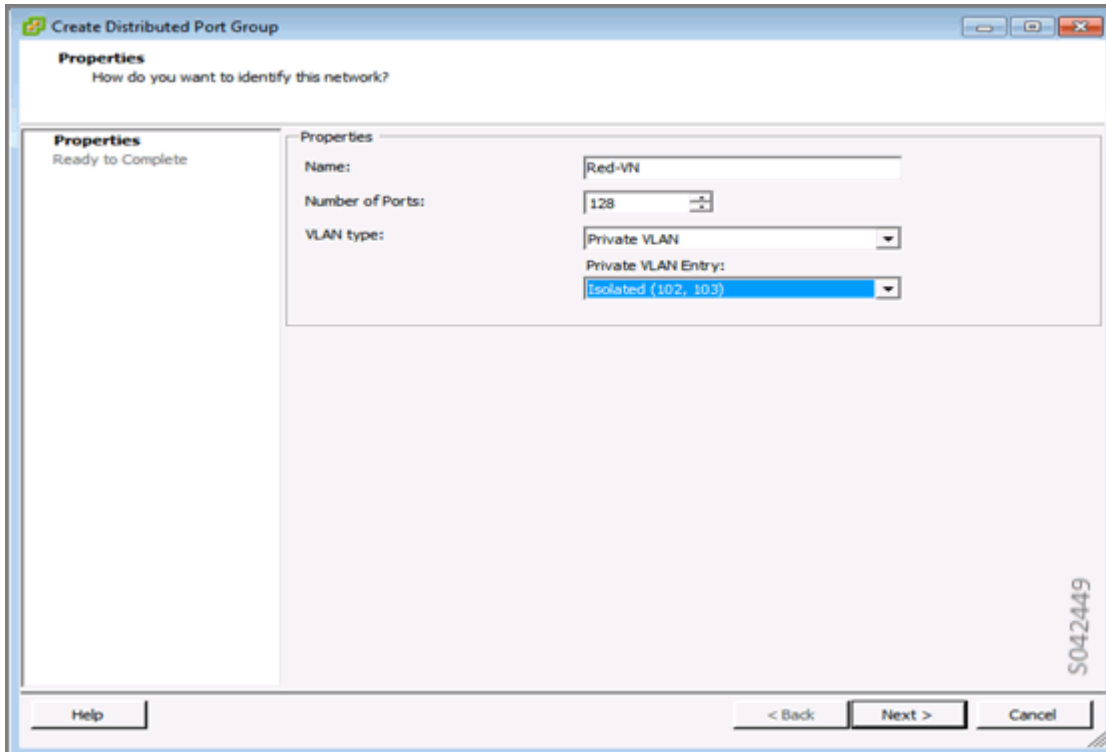


When you click **Create a new port group**, the **Create Distributed Port Group** window is displayed, as in the following figure.

Enter the name of the virtual network. Select the **VLAN type**, then select other details for the selected VLAN type.

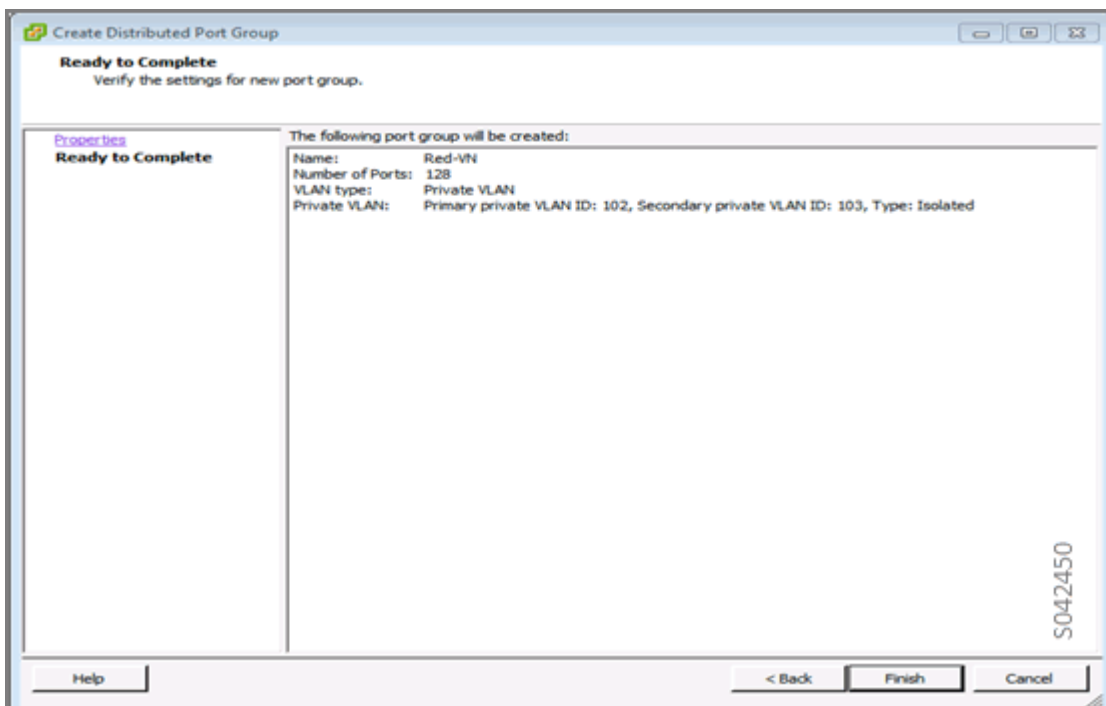
The following figure shows the **Create Distributed Port Group** window with the example creation of a virtual network named Red-VN, with a Private VLAN and isolated private VLAN ports 102, 103.

When you are finished, click **Next**.



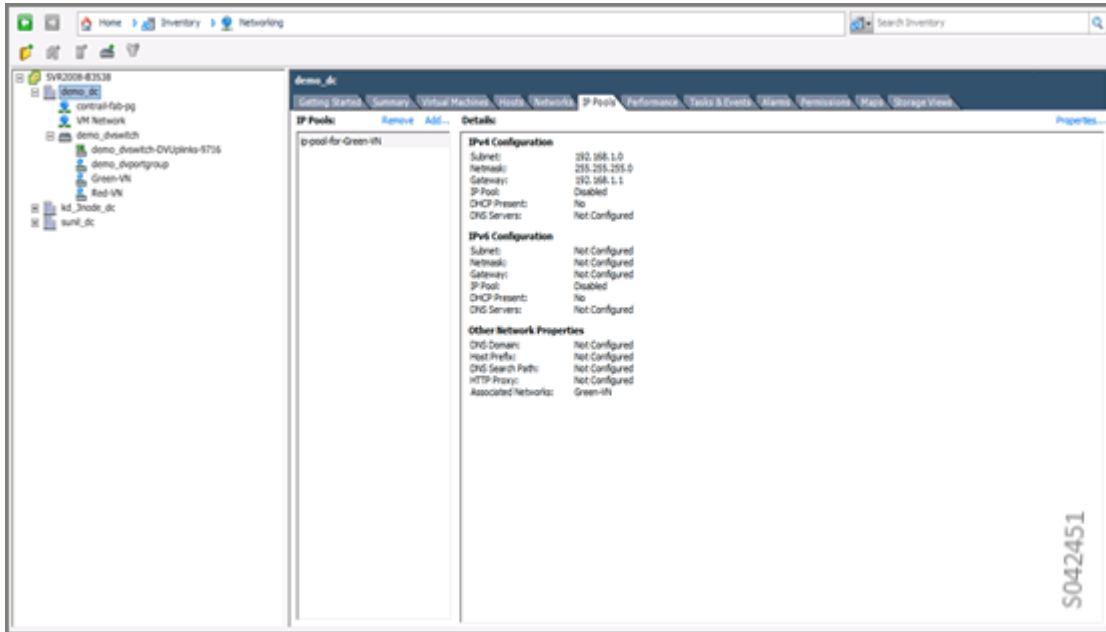
The **Ready to Complete** window is displayed, see the following figure. It shows the details entered for the virtual network (distributed port group).

If changes are needed, click **Back**. If the details are correct, click **Finish** to verify the port group details and complete its creation.

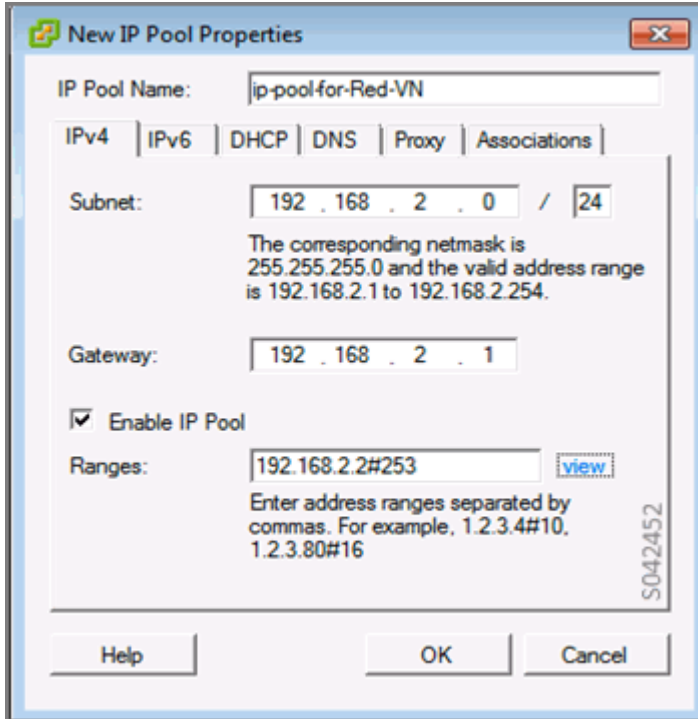


Next, create IP pools for the virtual network port group. Select the datacenter name in the left side panel, then click the **IP Pools** tab.

The following figure shows the **IP Pools** tab for the datacenter named demo\_dc.



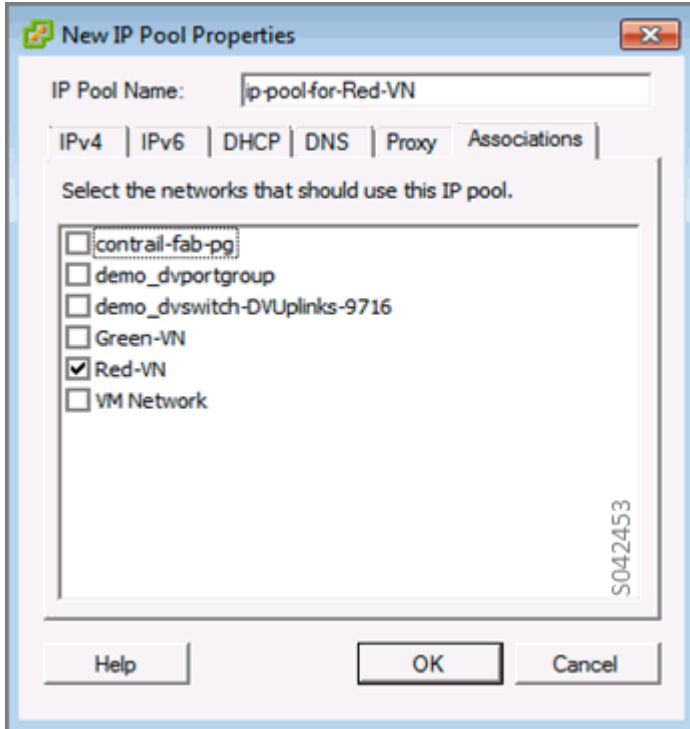
Near the top of the **IP Tools** window, click **Add** to open the **New IP Pool Properties** window, as in the following figure. The **IP Pool Properties** window has several tabs across the upper area. Ensure the **IPv4** tab is selected, and enter a name for the IP pool. Then enter the IP pool IPv4 details, including subnet, gateway, and IP address ranges. To enable IP address pools, select **Enable IP Pool**.



In the **New IP Pool Properties** window, click the **Associations** tab to select the networks that should use the IP address pool you are creating. This tab enables you to associate the IP pool with the port group.

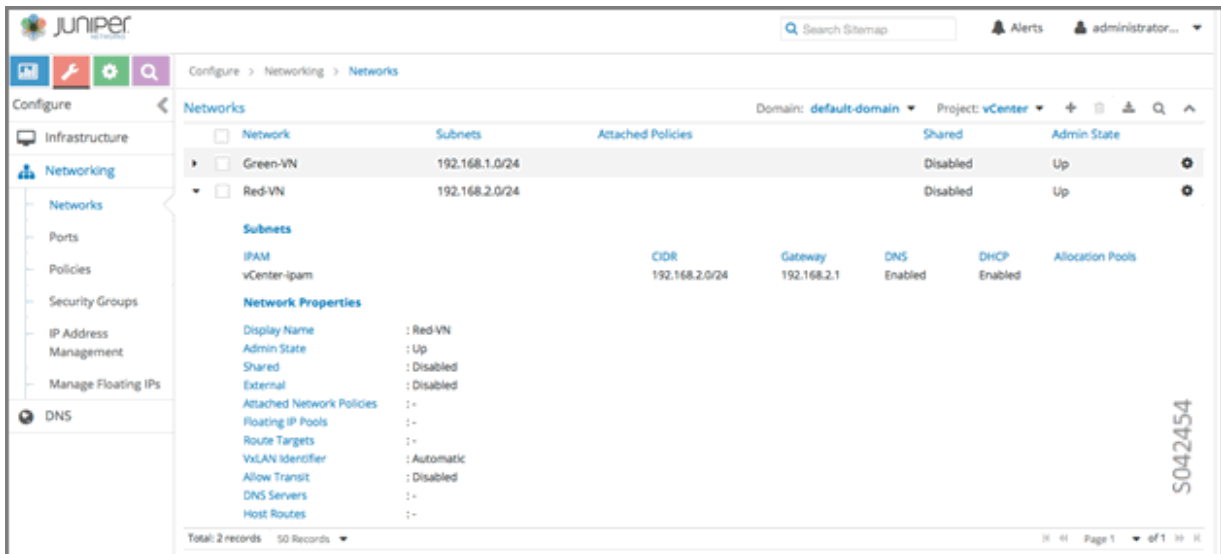
The following figure of the **Associations** tab shows that the IP pool being created should be associated with the virtual network port group named Red-VN.

When you are finished, click **OK**.



To verify that the virtual network is created and visible to Contrail, in the Contrail UI, select **Configure > Networking > Networks** to display Contrail network information.

The virtual network just created (Red-VN in this example) is displayed in the **Networks** window, see the following.

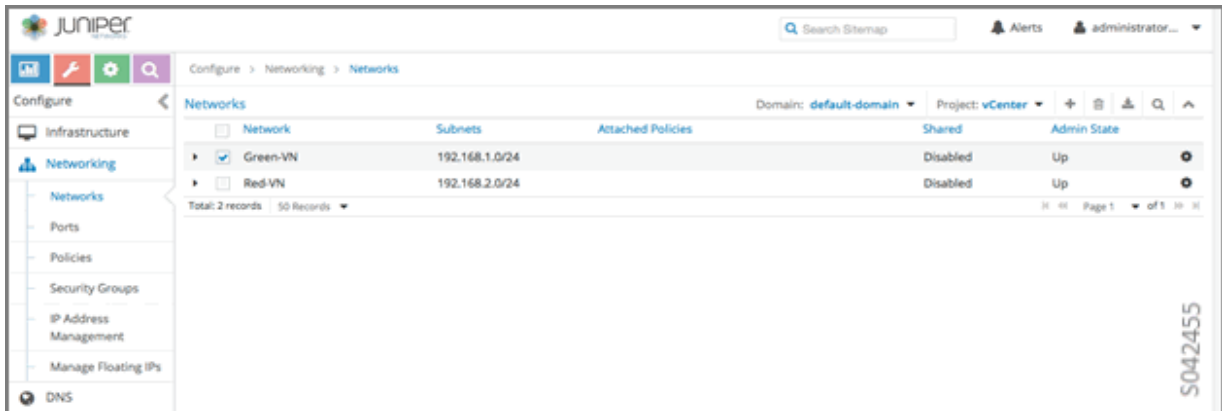


## Delete Virtual Network – Contrail UI

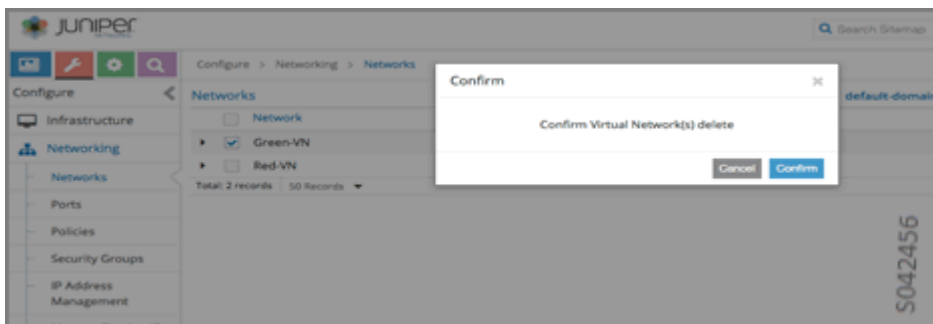
You can delete a virtual network in either the Contrail UI or in the vCenter UI. This section shows you how to delete a virtual network in the Contrail UI.

In the Contrail UI, select **Configure > Networking > Networks** to display Contrail network information.

Select the network you want to delete, then click the trashcan icon.



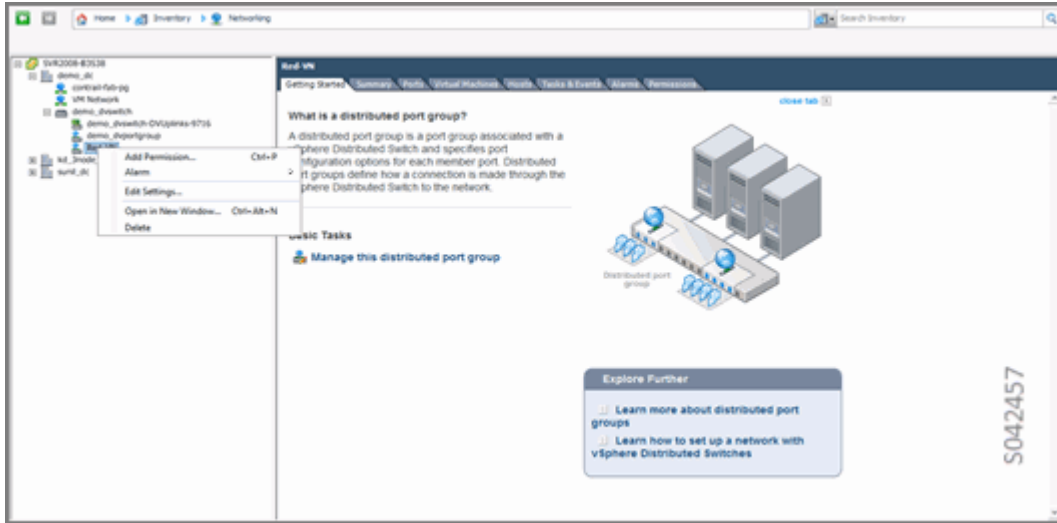
A Confirm window is displayed. Click **Confirm** to delete the selected network.



## Delete Virtual Networks – vCenter UI

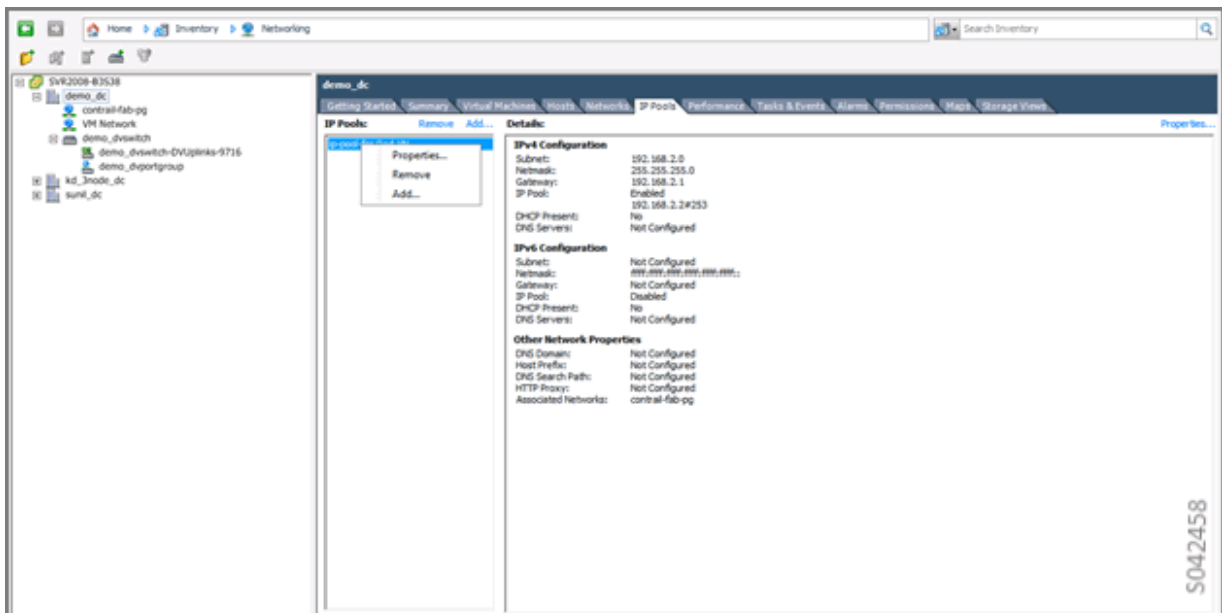
You can also delete a virtual network from the vCenter UI. From the vCenter UI, in the left side panel, right-click the port-group (virtual-network) you want to delete. In the menu, select **Delete** to delete the selected port group. An example is shown in the following.





When deleting a port group (virtual network) using the vCenter UI, you must also delete the IP pool associated with the port group. Select the **IP Pools** tab, and right click the name of the IP pool associated with the port group being deleted. From the menu, select **Remove** to delete the IP pool.

The following shows the deletion of the IP pool associated with the Red-VN from the vCenter UI.



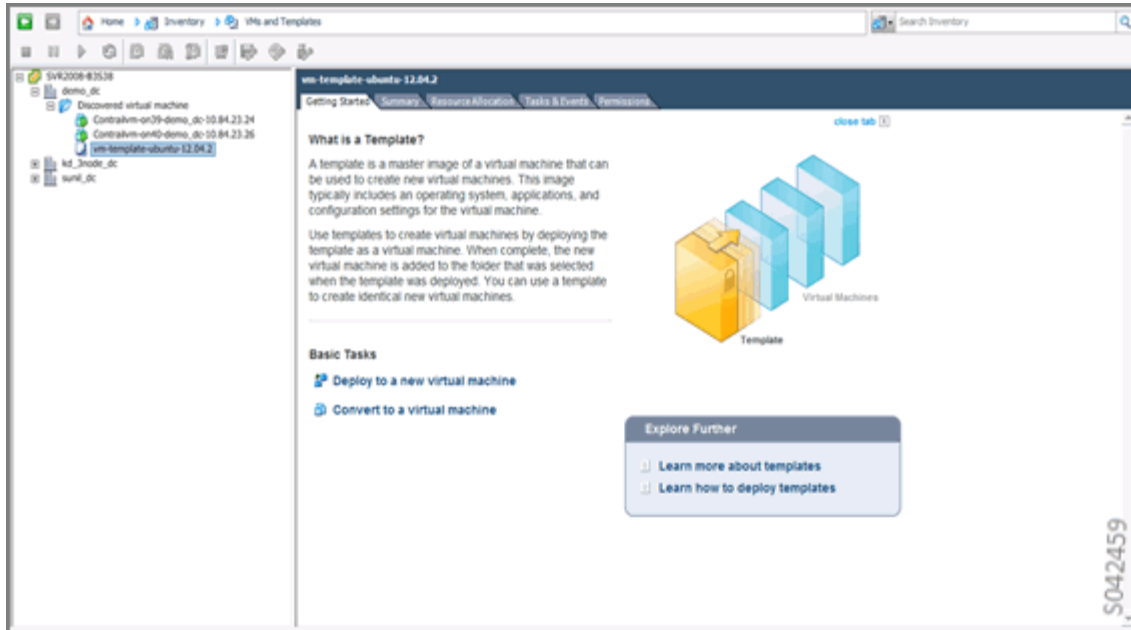
## Creating a Virtual Machine

Use the vCenter client interface to create a virtual machine for your VMware vCenter Contrail integrated system. This section describes how to create a virtual machine using a virtual machine template from the vCenter client interface.

## Create a Virtual Machine – vCenter UI

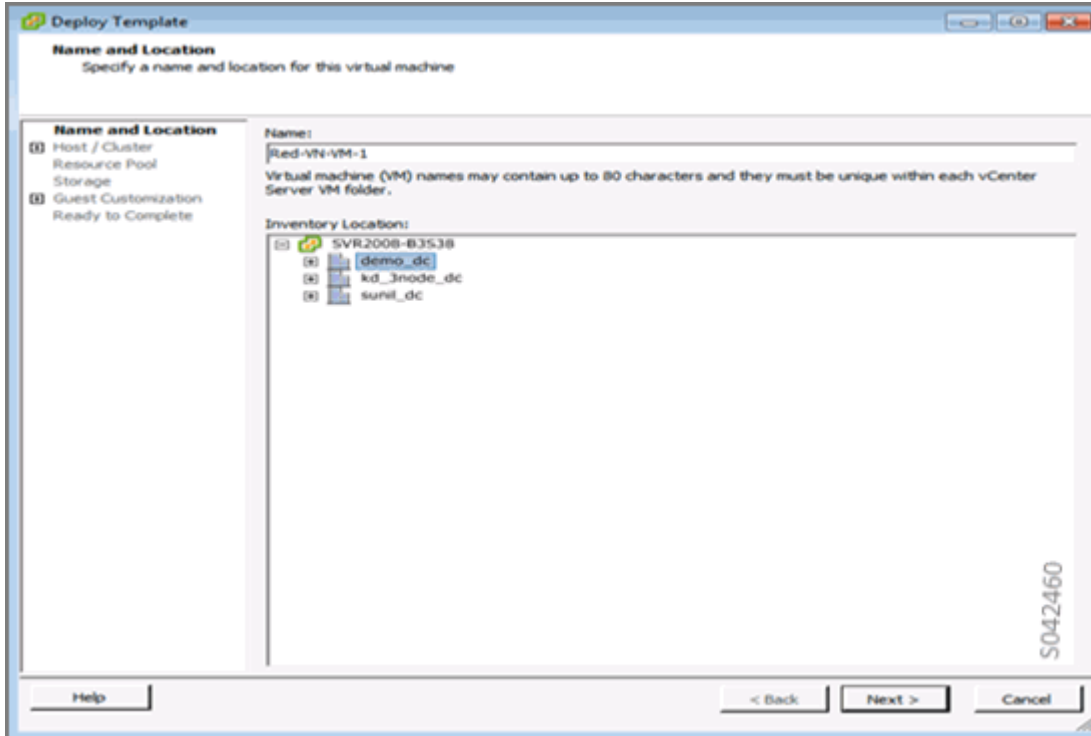
From the vCenter UI, select the virtual machine template from the left side panel. At the bottom of the right side pane, click **Deploy** to deploy a new virtual machine.

The following figure shows the **vm-template-ubuntu-12.04.2** virtual machine selected.



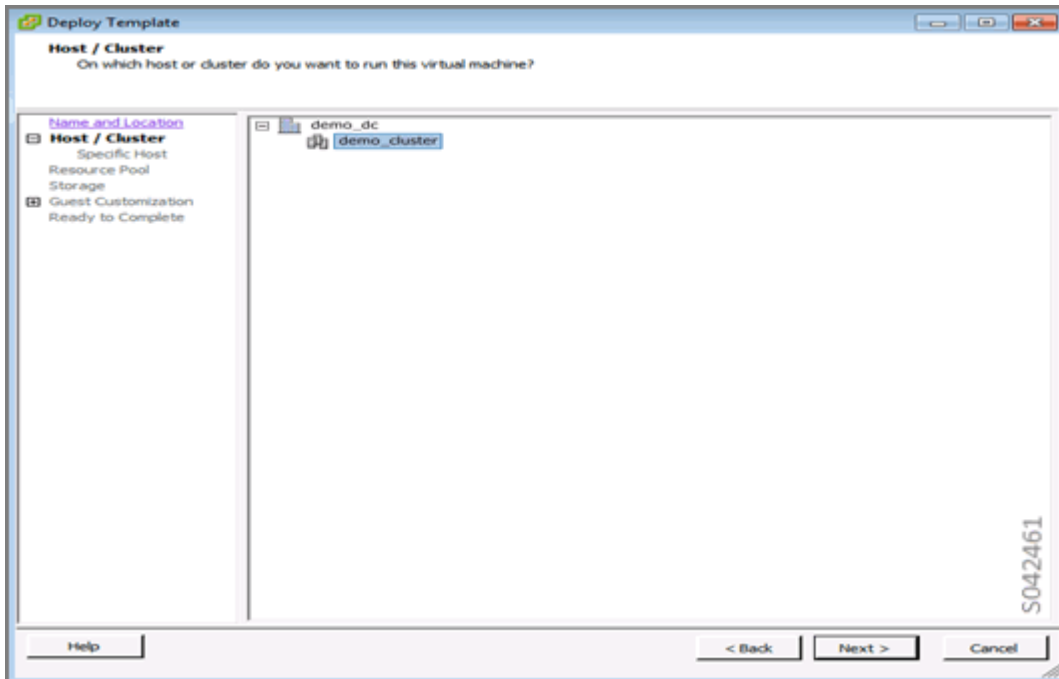
The **Deploy Template Name and Location** window is displayed, as in the following. Specify a name for the virtual machine and select the datacenter on which the virtual machine is to be spawned.

When you are finished, click **Next**.



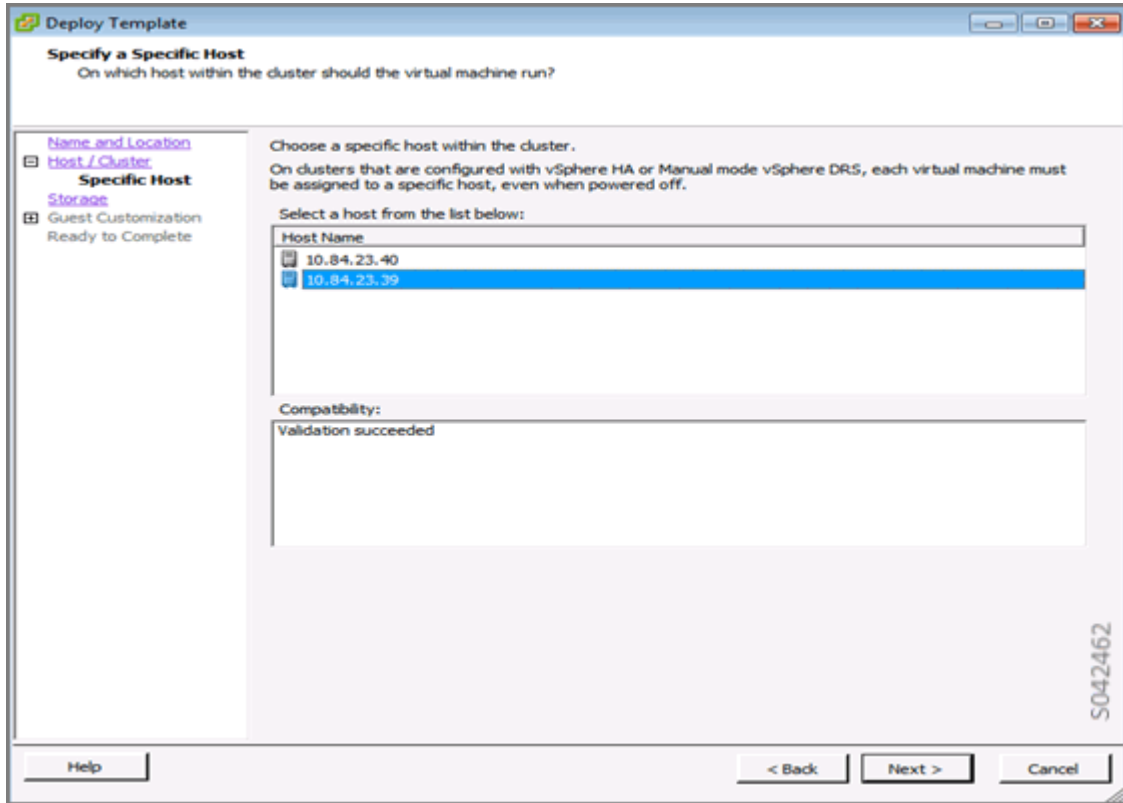
The **Host/Cluster** window is displayed, as in the following. Select the cluster on which to spawn the virtual machine.

When you are finished, click **Next**.



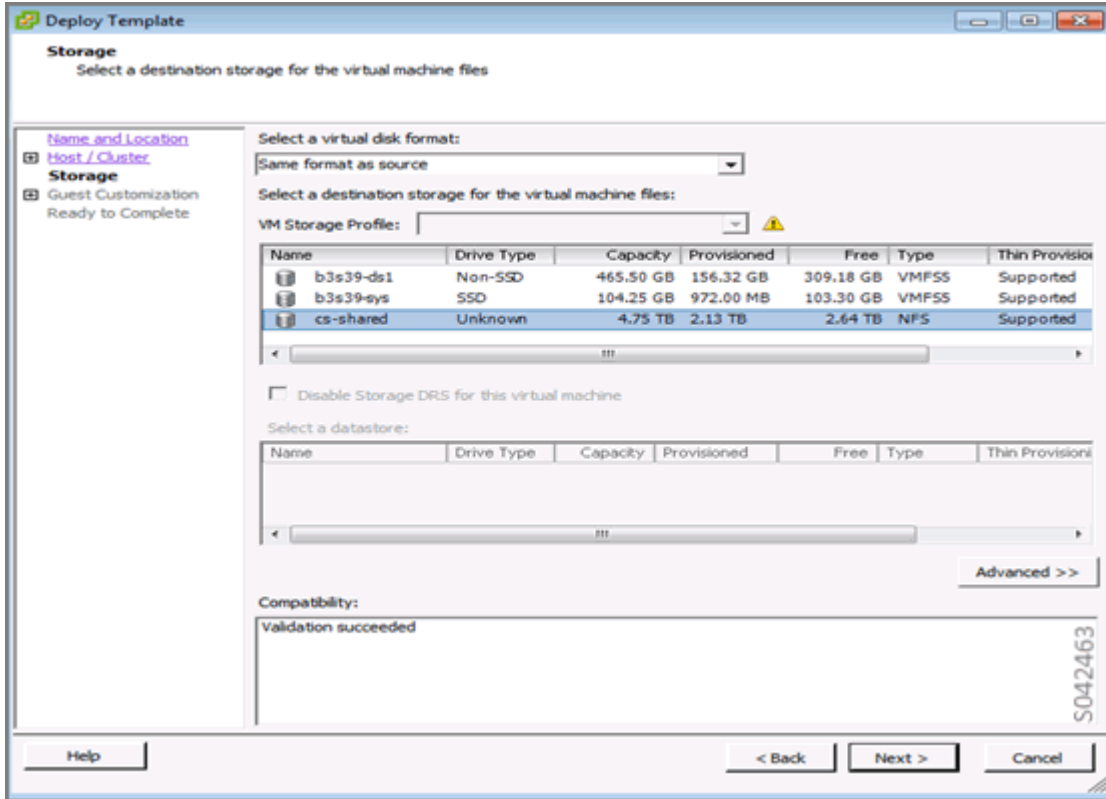
The **Specify a Specific Host** window is displayed, as in the following. Select the ESXi host on which to spawn the virtual machine.

When you are finished, click **Next**.



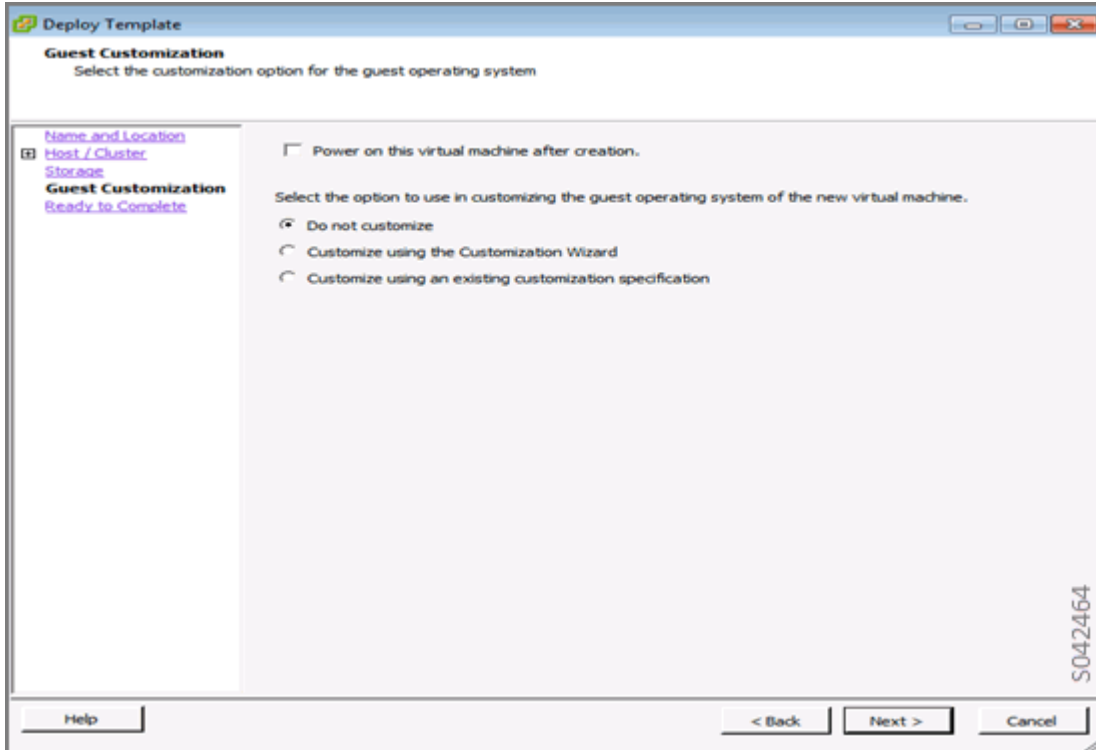
In the **Storage** window, select the destination storage location for the virtual machine.

When you are finished, click **Next**.



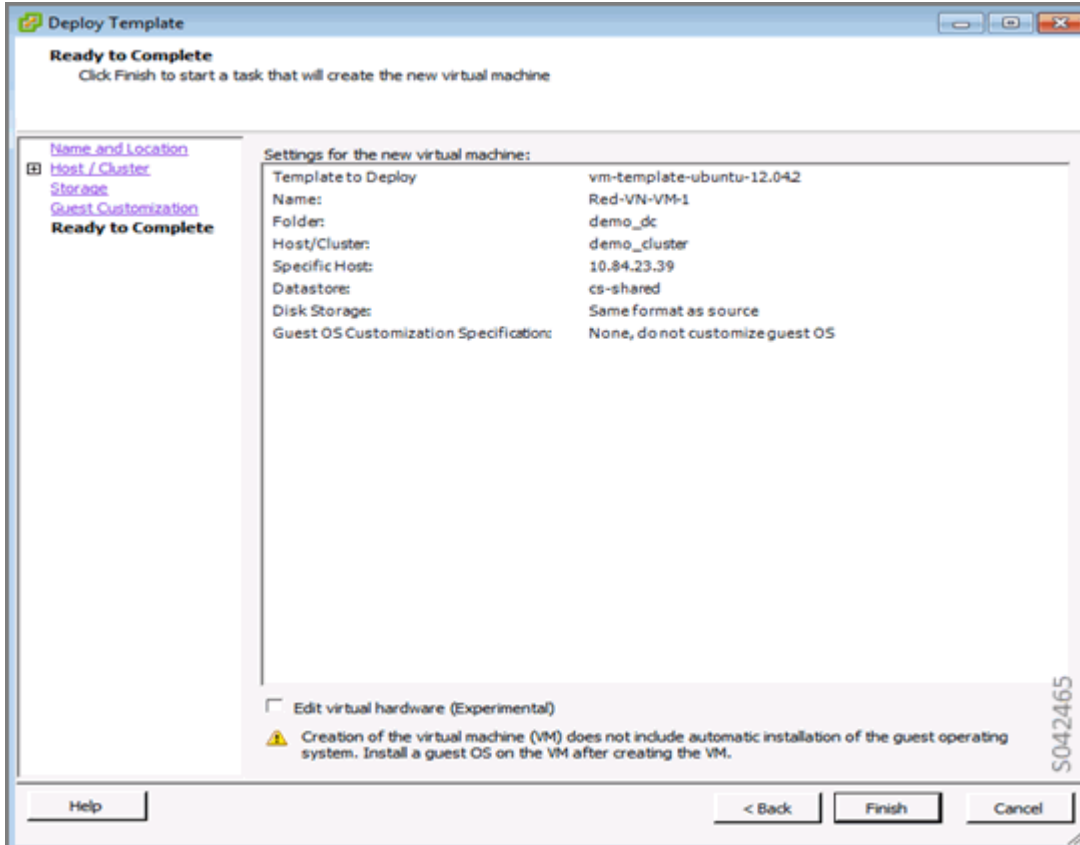
On the **Guest Customization** window, the typical selection is Do not customize. Select **Do not customize**.

When you are finished, click **Next**.



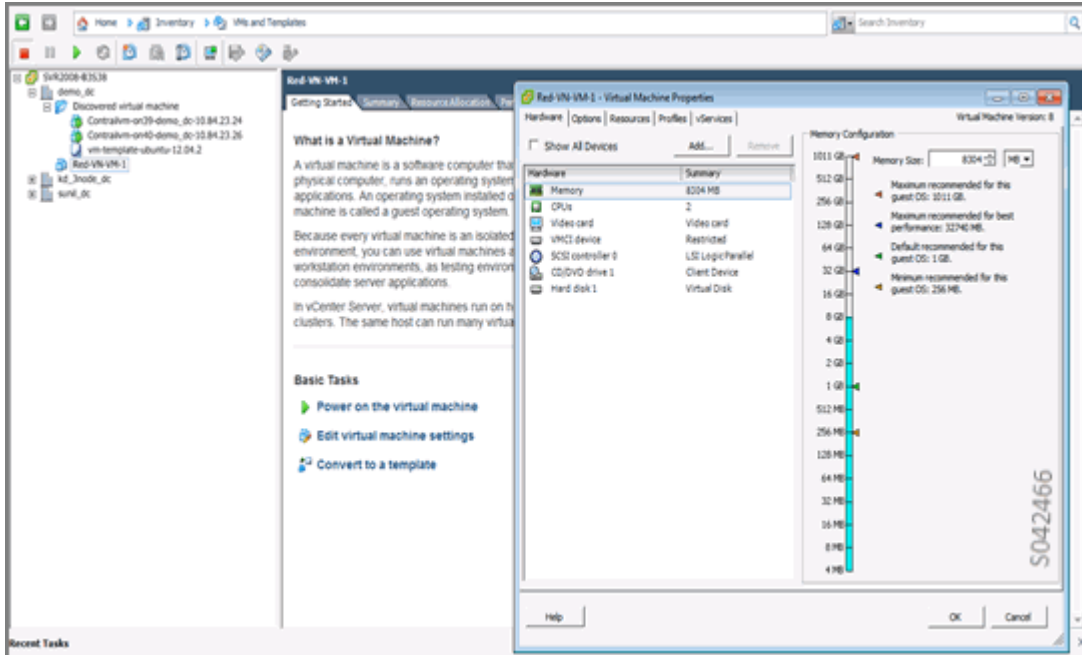
On the **Ready to Complete** window, review all of the virtual machine definitions that you have selected for the template.

If all the selections are correct, click **Finish**. This spawns the virtual machine.



To complete the settings for the virtual machine, select the virtual machine to be edited in the left column of the main window of the vCenter UI. Then click **Edit virtual machine settings**.

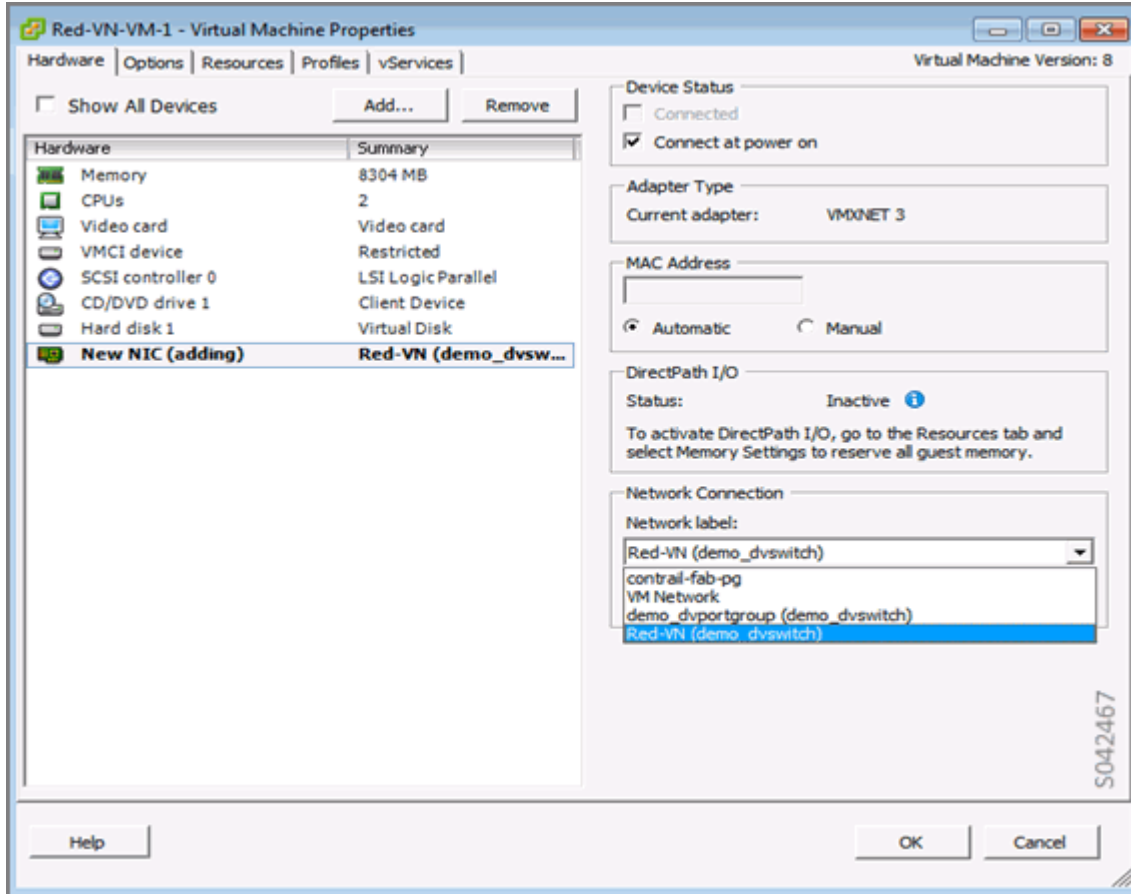
The **Virtual Machine Properties** window is displayed, as in the following. From here you can update the virtual machine properties.



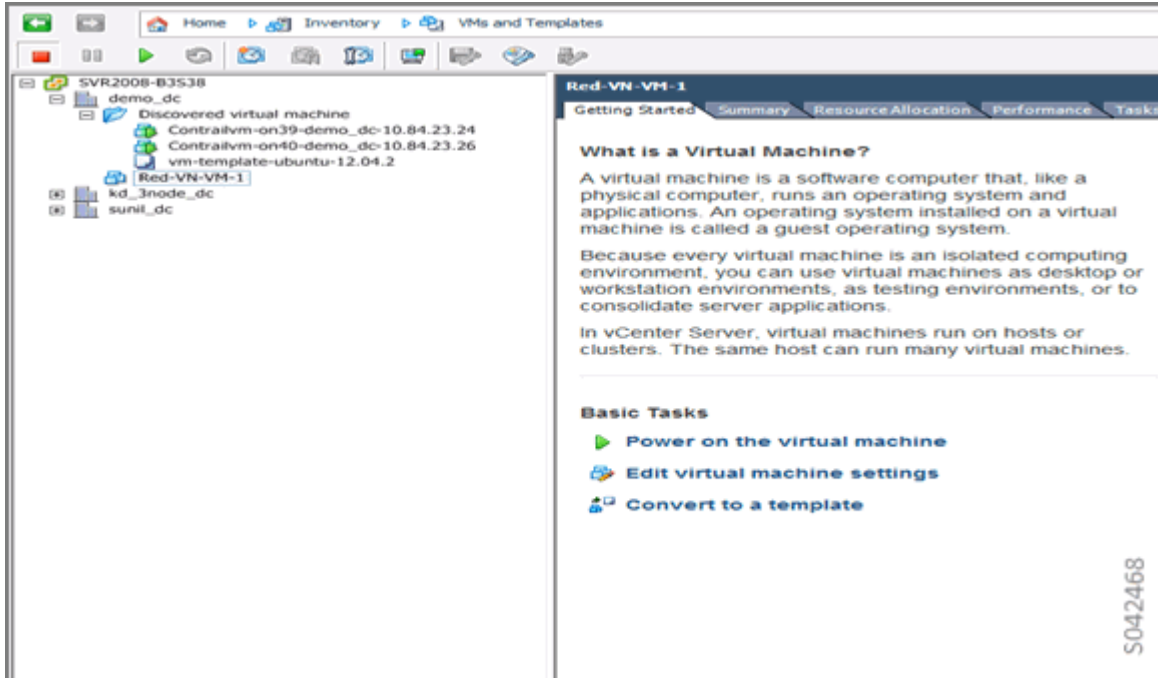
Click the **Hardware** tab in the **Virtual Machine Properties** window. Next, click **Add** to add a NIC and select the appropriate network. Select **Connect at power on**, as in the following.

When you are finished, click **OK**.





You are returned to the main vCenter UI window. Select the **Getting Started** tab. Select **Power on the virtual machine**. The virtual machine launches.



Once the virtual machine is launched, you can view it from the Contrail UI. Select **Monitor > Networking > Instances**. The virtual machines are displayed in the **Instances Summary** window, as in the following.

| Instance    | UUID                                 | Virtual Network  | Interfa... | vRouter    | IP Address                | Floating IPs (In/Out) | Traffic (In/Out) (Last 1 hr) |
|-------------|--------------------------------------|------------------|------------|------------|---------------------------|-----------------------|------------------------------|
| Red-VN-VM-1 | 5039e121-36b2-4ff4-9c38-2b251fc4ab13 | Red-VN (vCenter) | 1          | ContrailVM | IPv4: 192.168.2.254-23-24 |                       | 2.72 KB / 2.04               |

S042469

You can also see real-time running information for the virtual machine in the vCenter UI. Select the virtual machine and the **Console** tab. Real-time information is displayed, including ping statistics, as in the following.

```

UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
RX packets:11 errors:0 dropped:0 overruns:0 frame:0
TX packets:21 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:1750 (1.7 KB)  TX bytes:2088 (2.0 KB)

lo
  Link encap:Local Loopback
  inet addr:127.0.0.1  Mask:255.0.0.0
  inet6 addr: ::1/128 Scope:Host
  UP LOOPBACK RUNNING  MTU:16436  Metric:1
  RX packets:0 errors:0 dropped:0 overruns:0 frame:0
  TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
  collisions:0 txqueuelen:0
  RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@ubuntu-server-12:~# ping 192.168.2.1
PING 192.168.2.1 (192.168.2.1) 56(84) bytes of data:
64 bytes from 192.168.2.1: icmp_req=1 ttl=64 time=0.523 ms
64 bytes from 192.168.2.1: icmp_req=2 ttl=64 time=0.413 ms
64 bytes from 192.168.2.1: icmp_req=3 ttl=64 time=0.296 ms
^C
--- 192.168.2.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.296/0.410/0.523/0.095 ms
root@ubuntu-server-12:~# _

```

## Configuring the vCenter Network in Contrail UI

The following items can be configured for the vCenter network by using the Contrail UI.

- Network policy is configured by using the Contrail UI.
- Security policy is configured by using the Contrail UI.
- Public networks, floating IP address pools, and floating IP addresses are configured using the Contrail Administrator UI.

When you configure a virtual network using the administrator UI, the network is a Contrail-only network. No resources are consumed on vCenter to implement this type of network. You can configure a floating IP address pool on the network, allocate floating IP addresses, and associate floating IP addresses to virtual machine interfaces (ports).

## RELATED DOCUMENTATION

[Installing and Provisioning VMware vCenter with Contrail](#)

# Using Contrail with Red Hat

## IN THIS CHAPTER

- [Deploying Contrail with Red Hat OpenStack Platform Director 10 | 196](#)
- [Installing Red Hat OpenShift Container Platform with Contrail Networking | 247](#)
- [Upgrade Procedure for RHOSP-based Contrail 3.2.x to Contrail 4.1 | 254](#)
- [Upgrade Procedure for RHOSP-based Contrail 4.1.1 to Contrail 4.1.2 | 268](#)
- [Upgrade Procedure for RHOSP-based Contrail 4.1.2 to Contrail 4.1.3 | 279](#)
- [Upgrade Procedure for RHOSP-based Contrail 4.1.3 to Contrail 4.1.4 | 292](#)
- [Restoring Contrail Nodes in a RHOSP-based Environment | 308](#)

## Deploying Contrail with Red Hat OpenStack Platform Director 10

### IN THIS SECTION

- [Overview | 197](#)
- [TripleO Features | 197](#)
- [Deployment Tools | 199](#)
- [Preparing the Environment for Deployment | 199](#)
- [Deploying an OSPD-10 Overcloud | 203](#)
- [Configuring the Overcloud | 206](#)
- [Sample Heat Templates for NICs | 219](#)
- [What are NIC Templates? | 230](#)
- [How NIC Templates Work | 230](#)
- [Contrail NIC Templates | 232](#)
- [NIC Templates for Compute Nodes | 237](#)
- [NIC Templates for DPDK Compute Nodes | 242](#)

This document explains how to integrate a Contrail 3.2 through Contrail 4.1 installation with RedHat OpenStack Platform Director 10.

## Overview

RedHat OpenStack Platform provides an installer named Director (RHOSPD). The Red Hat Director installer is based on the OpenStack project TripleO (OOO, OpenStack on OpenStack). TripleO is an open source project that uses features of OpenStack to deploy a fully functional, tenant-facing OpenStack environment.

You can use TripleO and Director to deploy a Red Hat cloud environment integrated with Contrail.

**NOTE:** For Contrail Release 4.1.1, you must ensure that the OpenJDK version is java-1.8.0-openjdk-1.8.0.151-5.b12.el7\_4.x86\_64. This is because of a compatibility issue that the Cassandra 3.0 package has with the latest version of Open JDK provided in RHEL 7.5.

## TripleO Features

TripleO uses the concepts of undercloud and overcloud. TripleO sets up an undercloud, an operator-facing deployment cloud that contains the OpenStack components needed to deploy and manage an overcloud, a tenant-facing cloud that hosts user workloads.

The overcloud is the deployed solution that can represent a cloud for any purpose, such as production, staging, test, and so on. The operator can select to deploy to their environment any of the available overcloud roles, such as controller, compute, and the like.

TripleO leverages existing core components of OpenStack including Nova, Ironic, Neutron, Heat, Glance, and Ceilometer to deploy OpenStack on bare metal hardware.

- Nova and Ironic are used in the undercloud to manage the bare metal instances that comprise the infrastructure for the overcloud.
- Neutron is used to provide a networking environment in which to deploy the overcloud.
- Glance stores machine images.
- Ceilometer collects metrics about the overcloud.

For more information about TripleO architecture, see

<https://docs.openstack.org/developer/tripleo-docs/introduction/architecture.html>

## Composable Roles

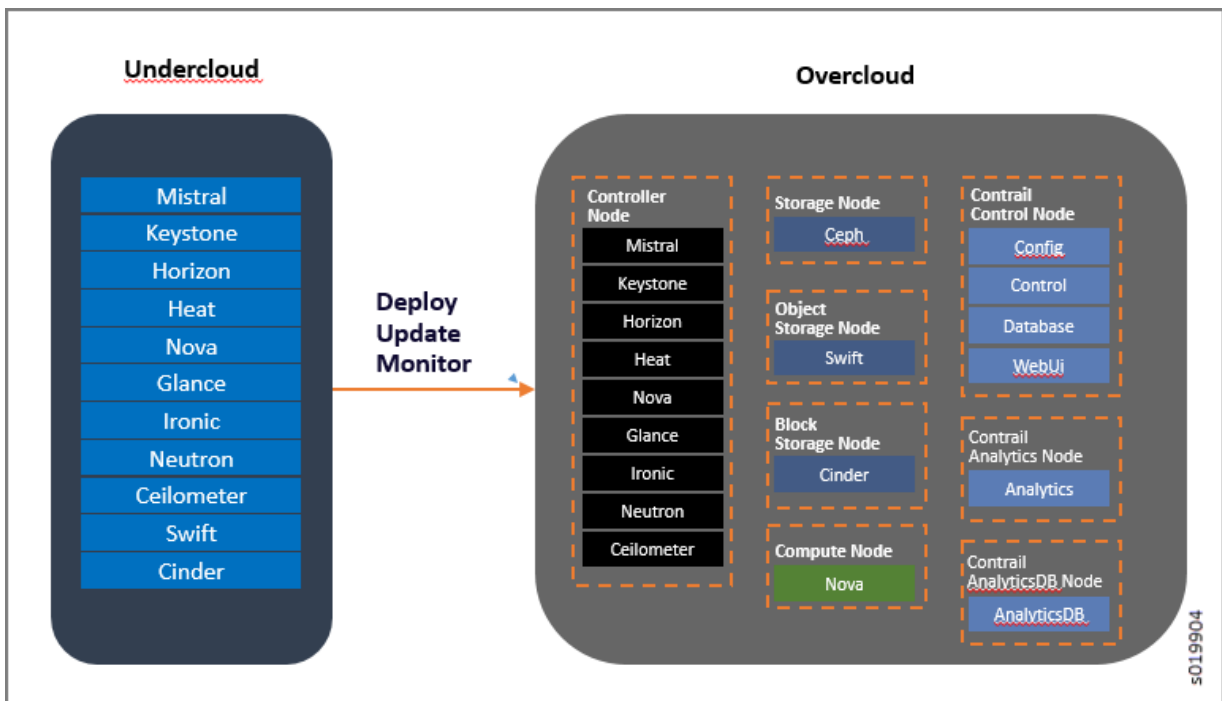
TripleO enables composable roles. Each role is a group of services that are defined in Heat templates. Composable roles gives the operator the flexibility to add and modify roles as needed.

The following are the Contrail roles used for integrating Contrail to the overcloud environment:

- Contrail Controller
- Contrail Analytics
- Contrail Analytics Database
- Contrail-TSN
- Contrail-DPDK

Figure 29 on page 198 shows the relationship and components of an undercloud and overcloud architecture for Contrail.

Figure 29: Undercloud and Overcloud with Roles



## Deployment Tools

Deployment to physical servers or virtual machines occurs by means of collaboration between Heat, Nova, Neutron, Glance, and Ironic.

One nested Heat stack is deployed from the undercloud. The Heat stack has various Nova instances. The Nova instances are the overcloud roles. The definitions for all roles are provided in the Heat templates.

To deploy the stack, Heat makes successive calls to Nova, OpenStack's compute service controller. Nova depends on Ironic, which, by this stage in the process, has acquired an inventory of introspected hardware.

If configured, Nova flavors can act as a constraint, influencing the range of machines that can be selected for deployment by the Nova scheduler. For each request to deploy a new node with a specific role, Nova filters the list of available nodes, ensuring that the selected nodes meet the hardware requirements.

When the target node has been selected, Ironic performs the provisioning of the node: Ironic retrieves from Glance the OS image associated with the role, causes the node to boot a deployment ramdisk, and, in a typical case, exports the node's local disk over iSCSI so that the disk can be partitioned and have the OS image written onto it by the Ironic conductor.

## Preparing the Environment for Deployment

The overcloud roles can be deployed to bare metal servers or to virtual machines (VMs). The compute nodes must be deployed to bare metal systems.

Ensure your environment is prepared for the Red Hat deployment. Refer to Red Hat documentation:

[https://access.redhat.com/documentation/en-us/red\\_hat\\_openstack\\_platform/10/html-single/director\\_installation\\_and\\_usage](https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/10/html-single/director_installation_and_usage)

## Preparing for the Contrail Roles

Ensure the following requirements are met for the Contrail nodes per role.

- Non-high availability: A minimum of four overcloud nodes are needed for control plane roles for a non-high availability deployment:
  - 1x contrail-config (includes Contrail control)
  - 1x contrail-analytics
  - 1x contrail-analytics-database

- 1x OpenStack controller
- High availability: A minimum of 12 overcloud nodes are needed for control plane roles for a high availability deployment:
  - 3x contrail-config (includes Contrail control)
  - 3x contrail-analytics
  - 3x contrail-analytics-database
  - 3x OpenStack controller
  - If the control plane roles will be deployed to VMs, use 3 separate physical servers and deploy one role of each kind to each physical server.

RHOSP Director expects the nodes to be provided by the administrator, for example, if you are deploying to VMs, the administrator must create the VMs before starting with deployment.

### Preparing for the Underlay Network

Refer to Red Hat documentation for planning and implementing underlay networking, including the kinds of networks used and the purpose of each:

- [https://access.redhat.com/documentation/en-us/red\\_hat\\_openstack\\_platform/10/html-single/director\\_installation\\_and\\_usage/#sect-Planning\\_Networks](https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/10/html-single/director_installation_and_usage/#sect-Planning_Networks)
- [https://access.redhat.com/documentation/en-us/red\\_hat\\_openstack\\_platform/10/html-single/director\\_installation\\_and\\_usage/#sect-Networking\\_Requirements](https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/10/html-single/director_installation_and_usage/#sect-Networking_Requirements)

At a high level, every overcloud node must support IPMI.

Refer to “Requirements for Deploying to VMs” in this document if you are deploying to VMs.

### Preparing for the Provisioning Network

Ensure the following requirements are met for the provisioning network.

- One NIC from every machine must be in the same broadcast domain of the provisioning network, and it should be the same NIC on each of the overcloud machines. For example, if you use the second NIC on the first overcloud machine, you should use the second NIC on each additional overcloud machine.

During installation, these NICs will be referenced by a single name across all overcloud machines.

- The provisioning network NIC should not be the same NIC that you are using for remote connectivity to the undercloud machine. During the undercloud installation, an Open vsSwitch bridge will be



created for Neutron and the provisioning NIC will be bridged to the Open vSwitch bridge. Consequently, connectivity would be lost if the provisioning NIC was also used for remote connectivity to the undercloud machine.

- The provisioning NIC on the overcloud nodes must be untagged.
- You must have the MAC address of the NIC that will PXE boot the IPMI information for the machine on the provisioning network. The IPMI information will include such things as the IP address of the IPMI NIC and the IPMI username and password.
- All of the networks must be available to all of the Contrail roles and computes.

## Network Isolation

TripleO enables configuration of isolated overcloud networks. Using this approach, it is possible to host traffic in isolated networks for specific types of network traffic, such as tenants, storage, API, and the like. This enables assigning network traffic to specific network interfaces or bonds.

When isolated networks are configured, the OpenStack services are configured to use the isolated networks. If no isolated networks are configured, all services run on the provisioning network.

The following networks are typically used when using network isolation topology:

- Provisioning---for the undercloud control plane
- Internal API--- for OpenStack internal APIs
- Tenant
- Storage
- Storage Management
- External
  - Floating IP---Can either be merged with external or can be a separate network.
- Management

## Templates for Network Isolation

Use the following template files to enable network isolation:

- `environments/network-isolation.yaml`

Contains the path of templates that need to be included to create various Neutron networks and ports

- `environments/contrail/contrail-net.yaml`

Contains the subnet/mask, allocation pool, default gateway IP information. Make changes to this file to configure the subnets for your setup.

- `environments/contrail/contrail-nic-config.yaml`

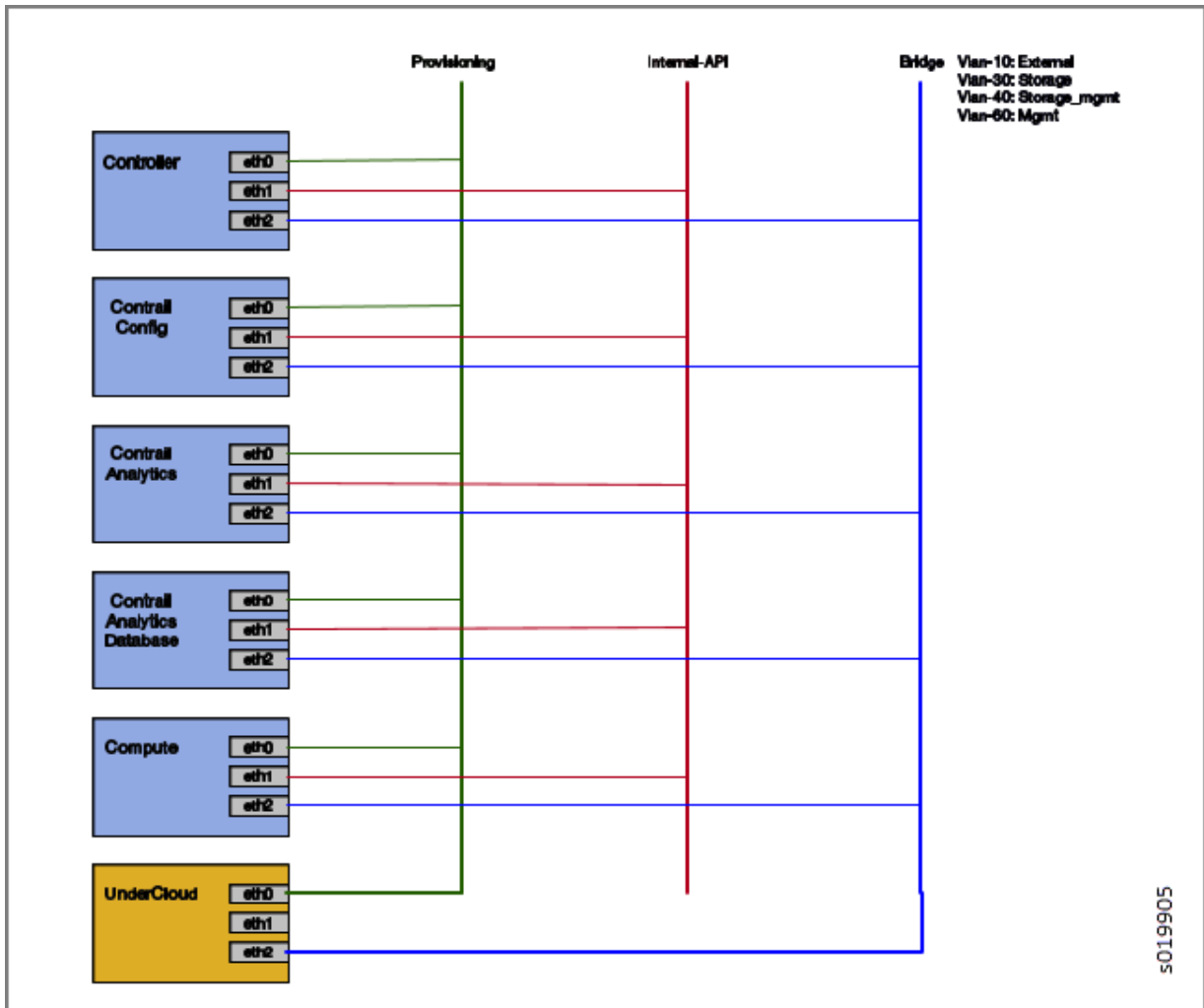
Defines the NICs that the overcloud VMs will use for each of the networks. Change the contents of this template as needed for your environment.

Features of the default configuration include:

- The first NIC is used for the control plane provisioning network.
- The second NIC is used for the internal API network.
- The third NIC uses multiple VLANs to provide for the rest of the networks:
  - VLAN-10: External network
  - VLAN-30: Storage network
  - VLAN-40: Storage management network
  - VLAN-50: Tenant network
  - VLAN-60: Management network
  - VLAN-XXX: Floating network (if separate from external network)

[Figure 30 on page 203](#) shows the network connectivity for the overcloud roles when you use the default Heat templates. Fig . In [Figure 30 on page 203](#), the vertical lines depict the underlay, which could be a switch. The underlay connectivity must be prepared before starting the deployment. The undercloud must have reachability in the provisioning network and the external networks.

Figure 30: Network Isolation Model



## Deploying an OSPD-10 Overcloud

When the requirements for the environment are met, you are ready to start deploying.

### Installing the Undercloud

Use the Red Hat OS Director to install the undercloud after the environment has been prepared. You'll need Red Hat credentials, such as account, password, pool, and the like, to register the undercloud and overcloud nodes.

Follow procedures in the Red Hat documentation to install an undercloud:

[https://access.redhat.com/documentation/en-us/red\\_hat\\_openstack\\_platform/10/html-single/director\\_installation\\_and\\_usage/#chap-Installing\\_the\\_Undercloud](https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/10/html-single/director_installation_and_usage/#chap-Installing_the_Undercloud)

## Configuring Undercloud and Overcloud

After the undercloud is installed, you can use the following procedures to change parameters in the `undercloud.conf` file to match your local deployment.

### 1. Configure the undercloud.

```
cp /usr/share/instack-undercloud/undercloud.conf.sample ~/undercloud.conf
vi ~/undercloud.conf
```

### 2. Install the undercloud OpenStack.

```
openstack undercloud install
```

### 3. Source the undercloud credentials.

```
source ~/stackrc
```

### 4. Get overcloud images.

```
sudo yum install rhosp-director-images rhosp-director-images-ipa
mkdir ~/images
cd ~/images
```

### 5. Upload overcloud images.

```
for i in /usr/share/rhosp-director-images/overcloud-full-latest-10.0.tar /usr/share/rhosp-director-images/
ironic-python-agent-latest-10.0.tar; do tar -xvf $i; done
openstack overcloud image upload --image-path /home/stack/images/
cd ~
```

## Defining Nodes with Ironic

The properties of the overcloud nodes and VMs are in the `instackenv.json` file, which is imported to Ironic.

This procedure shows how to define nodes with Ironic.

### 1. Define nodes in `instackenv.json`.

```
vi ~/instackenv.json
```

- A password-less SSH must be enabled on all hosts on which overcloud VMs will be spawned.

- If you need definitive node placement, assign the appropriate capabilities in the node definition in `instackenv.json`.

For more information about using `instackenv.json`, see Red Hat documentation:

- [https://access.redhat.com/documentation/en-us/red\\_hat\\_openstack\\_platform/10/html/director\\_installation\\_and\\_usage/chap-configuring\\_basic\\_overcloud\\_requirements\\_with\\_the\\_cli\\_tools](https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/10/html/director_installation_and_usage/chap-configuring_basic_overcloud_requirements_with_the_cli_tools)
- [https://access.redhat.com/documentation/en-us/red\\_hat\\_openstack\\_platform/10/html-single/director\\_installation\\_and\\_usage/#sect-Registering\\_Nodes\\_for\\_the\\_Overcloud](https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/10/html-single/director_installation_and_usage/#sect-Registering_Nodes_for_the_Overcloud)

A sample `instackenv.json` is included later in this topic.

## 2. Import nodes to Ironic.

```
openstack baremetal import --json ~/instackenv.json
```

## 3. Activate node introspection.

Introspection boots each Ironic node over the PXE network and is used to collect hardware data for the nodes. The capabilities and profile of each node is determined at this step. Because this step includes pushing an image to each of the overcloud roles, successful completion of Ironic introspection also means that the underlay configuration is valid on the provisioning network.

**NOTE:** Make sure that the maximum transmission unit (MTU) is consistent across all of the networks to prevent any issues.

```
for node in $(openstack baremetal node list -c UUID -f value) ; do openstack baremetal node manage $node ; done
```

```
openstack overcloud node introspect --all-manageable --provide
```

## 4. Perform node profiling.

If you provided capabilities for overcloud nodes, create the corresponding flavors at this time. Each overcloud role can be assigned a certain Nova-flavor in the Heat templates. You can provide details such as memory, disk-size, number of CPUs, and so on. At the time of deploying a role, Director tries to find an Ironic node that has the capabilities listed in the flavor. This is the way in which you can control node placement.

```
openstack flavor create <flavor-name> --ram <RAM> --vcpus <CPUs> --disk <disk-size>
```

```
openstack flavor set --property "capabilities:boot_option"="local" --property "capabilities:profile"="<capability-name>" <flavor-name>
```

## Configuring the Overcloud

### Get Conrail Components

This procedure provides the components needed to integrate Conrail with Director, including adding a repo that hosts Conrail packages and providing Heat templates and corresponding Puppet modules.

#### 1. Create a Conrail repo.

A Conrail repo is needed to make sure that the overcloud Conrail roles can install the Conrail packages. The Conrail repo can be hosted on the undercloud or on any machine that is accessible from the overcloud nodes on the provisioning network.

```
sudo mkdir /var/www/html/conrail sudo tar zxvf ~/conrail-install-packages_<package>.tgz -C /var/www/html/
conrail/
```

#### 2. Upload Puppet modules to Swift.

Install the RPMs for Puppet modules in the directory: **home/stack//usr/share/openstack-puppet/modules/**.

This folder must contain the Puppet modules necessary to successfully install and start Conrail services in the overcloud roles.

Use the command `upload-swift-artifacts` to make sure that these modules get uploaded on the overcloud nodes during deployment. All of the commands are executed as user **stack**.

```
cd /var/www/html/conrail

yum localinstall conrail-tripleo-puppet-<version>.el7.noarch.rpm puppet-conrail-<version>.el7.noarch.rpm

mkdir -p ~/usr/share/openstack-puppet/modules/conrail

cp -R /usr/share/openstack-puppet/modules/conrail/* ~/usr/share/openstack-puppet/modules/conrail/

mkdir -p ~/usr/share/openstack-puppet/modules/tripleo

cp -R /usr/share/conrail-tripleo-puppet/* ~/usr/share/openstack-puppet/modules/tripleo

cd ~

tar czvf puppet-modules.tgz ~/usr/

upload-swift-artifacts -f puppet-modules.tgz
```

#### 3. Get TripleO Heat templates.

```
cp -r /usr/share/openstack-tripleo-heat-templates/ ~/tripleo-heat-templates
```

```
cd /var/www/html/contrail

yum localinstall contrail-tripleo-heat-templates-<version>.el7.noarch.rpm

cp -r /usr/share/contrail-tripleo-heat-templates/environments/contrail ~/tripleo-heat-templates/environments

cp -r /usr/share/contrail-tripleo-heat-templates/puppet/services/network/* ~/tripleo-heat-templates/puppet/
services/network
```

#### 4. Update the contrail-services.yaml.

The `contrail-services.yaml` is the main administrator-facing Heat template. Provide the correct URL for the Contrail repo that you created, the flavor for overcloud roles, the count for overcloud roles, and other various environment-specific parameters such as DNS-server, NTP server, and the like.

```
vi ~/tripleo-heat-templates/environments/contrail/contrail-services.yaml
```

You must set the value for `ContrailVersion` to 5.

## Configure NICs for Overcloud Networking

Use this information to configure the NICs for your system.

### Overcloud Networking—Multiple NICs

```
vi ~/tripleo-heat-templates/environments/contrail/contrail-net.yaml
vi ~/tripleo-heat-templates/environments/contrail/contrail-nic-config-compute.yaml
vi ~/tripleo-heat-templates/environments/contrail/contrail-nic-config.yaml
```

### Overcloud Networking—Multiple NICs with Bond and VLAN

```
vi ~/tripleo-heat-templates/environments/contrail/contrail-net-bond-vlan.yaml
vi ~/tripleo-heat-templates/environments/contrail/contrail-nic-config-compute-bond-vlan.yaml
vi ~/tripleo-heat-templates/environments/contrail/contrail-nic-config-vlan.yaml
```

### Overcloud Networking—Single NIC

```
vi ~/tripleo-heat-templates/environments/contrail/contrail-net-single.yaml
vi ~/tripleo-heat-templates/environments/contrail/contrail-nic-config-compute-single.yaml
vi ~/tripleo-heat-templates/environments/contrail/contrail-nic-config-single.yaml
```

## Assign Addresses and Credentials

### 1. Assign static IP addresses.

Use the template `ips-from-pool-all.yaml` to provide static IP addresses for the overcloud nodes.

```
vi ~/tripleo-heat-templates/environments/contrail/ips-from-pool-all.yaml
```

### 2. Provide subscription manager credentials.

Use the template `environment-rhel-registration.yaml` to provide subscription manager credentials, including `rhel_reg_password`, `rhel_reg_pool_id`, `rhel_reg_repos`, `rhel_reg_user`, and `method`.

```
vi ~/tripleo-heat-templates/extraconfig/pre_deploy/ rhel-registration/environment-rhel-registration.yaml
```

The following is a sample `environment-rhel-registration.yaml` file for deployment.

**NOTE:** The repos enabled are required to enable deployment for Contrail 3.2 with Director 10 and OpenStack Newton.

```
[stack@instack ~]$ cat environment-rhel-registration.yaml
# Note this can be specified either in the call
# to heat stack-create via an additional -e option
# or via the global environment on the seed in
# /etc/heat/environment.d/default.yaml
parameter_defaults:
  rhel_reg_activation_key: ""
  rhel_reg_auto_attach: "true"
  rhel_reg_base_url: ""
  rhel_reg_environment: ""
  rhel_reg_force: ""
  rhel_reg_machine_name: ""
  rhel_reg_org: ""
  rhel_reg_password: ""
  rhel_reg_pool_id: ""
  rhel_reg_release: ""
  rhel_reg_repos: "rhel-7-server-rpms rhel-7-server-extras-rpms rhel-7-server-rh-common-rpms
rhel-ha-for-rhel-7-server-rpms rhel-7-server-openstack-10-rpms rhel-7-server-openstack-10-
devtools-rpms"
  rhel_reg_sat_url: ""
  rhel_reg_server_url: ""
  rhel_reg_service_level: ""
```



```

rhel_reg_user: ""
rhel_reg_type: ""
rhel_reg_method: "portal"
rhel_reg_sat_repo: "rhel-7-server-satellite-tools-6.1-rpms"

```

### 3. Set the overcloud nameserver.

```
neutron subnet-show neutron subnet-update <SUBNET-UUID> --dns-nameserver NAMESERVER_IP
```

## Deploying the Overcloud

When you perform the overcloud installation, the overcloud is generated with the definitions you provide in the Heat templates.

The `openstack overcloud deploy` command creates a nested stack with all the resources needed to deploy the overcloud roles, networks, services, and so on.

- The stack can be updated if you wish to make changes to the overcloud.
- To redeploy the overcloud with a fresh installation, you delete the existing stack, make appropriate changes to the Heat templates, and then redeploy the stack.

### Deploy Overcloud with a Single NIC

```

openstack overcloud deploy --templates tripleo-heat-templates/ \
  --roles-file tripleo-heat-templates/environments/contrail/roles_data.yaml \
  -e tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/ environment-rhel-
  registration.yaml \
  -e tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/ rhel-registration-resource-
  registry.yaml \
  -e tripleo-heat-templates/environments/contrail/contrail-services.yaml \
  -e tripleo-heat-templates/environments/contrail/contrail-net-single.yaml

```

### Deploy Overcloud with Multiple NICs

```

openstack overcloud deploy --templates tripleo-heat-templates/ \
  --roles-file tripleo-heat-templates/environments/contrail/roles_data.yaml \
  -e tripleo-heat-templates/environments/puppet-pacemaker.yaml \
  -e tripleo-heat-templates/environments/contrail/contrail-services.yaml \
  -e tripleo-heat-templates/environments/contrail/network-isolation.yaml \

```

```

-e tripleo-heat-templates/environments/contrail/contrail-net.yaml \
-e tripleo-heat-templates/environments/contrail/ips-from-pool-all.yaml \
-e tripleo-heat-templates/environments/network-management.yaml \
-e tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/ environment-rhel-
registration.yaml \
-e tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/ rhel-registration-resource-
registry.yaml

```

## Deploy Overcloud with Multiple NICs with Bond and VLAN

```

openstack overcloud deploy --templates tripleo-heat-templates/ \
--roles-file tripleo-heat-templates/environments/contrail/roles_data.yaml \
-e tripleo-heat-templates/environments/puppet-pacemaker.yaml \
-e tripleo-heat-templates/environments/contrail/contrail-services.yaml \
-e tripleo-heat-templates/environments/contrail/network-isolation.yaml \
-e tripleo-heat-templates/environments/contrail/contrail-net-bond-vlan.yaml \
-e tripleo-heat-templates/environments/contrail/ips-from-pool-all.yaml \
-e tripleo-heat-templates/environments/network-management.yaml \
-e tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/ environment-rhel-
registration.yaml \
-e tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/ rhel-registration-resource-
registry.yaml

```

## Sample instackenv.json

This section has a sample `instackenv.json`, with OpenStack and Contrail controller on separate physical machines. This sample imports VMs to Ironic.

The sample `instackenv.json` is from a working environment that includes:

- 3x KVM hosts: 10.xx.xx.22, 10.xx.xx.24, 10.xx.xx.25 2.
- The following overcloud VMs on each KVM host:
  - openstack-controller
  - contrail-controller
  - contrail-analytics
  - contrail-analytics database
  - compute

- This sample imports VMs to Ironic.

Mandatory parameters for importing VMs to Ironic include:

|                    |   |
|--------------------|---|
| <b>Pm_addr</b>     | the IP address of the host on which the target VM is spawned.   |
| <b>Pm_user</b>     | Preferably the root user, or any other user with required permissions for accessing libvirtd.   |
| <b>Pm_password</b> | the public SSH key of the host on which the target VM is spawned. Make sure that the line breaks are replaced with '\n'. You can use a simple program such as 'awk '{printf "%s\n", \$0}' ~/.ssh/id_rsa' to achieve this. |
| <b>MAC</b>         | the MAC address of the target VM's NIC that is connected to the provisioning control plane network.   |
| <b>Pm_type</b>     | pxe_ssh, the driver needed to provision VMs.  |

#### Sample instackenv.json

```
{
  "arch": "x86_64",
  "host-ip": "192.168.122.1",
  "power_manager": "nova.virt.baremetal.virtual_power_driver.VirtualPowerManager",
  "seed-ip": "",
  "ssh-key": "-----BEGIN RSA PRIVATE KEY-----
    $ABC123
  -----END RSA PRIVATE KEY-----\n",
  "ssh-user": "root",
  "nodes": [
    {
      "mac": [
        "52:54:00:d7:e4:87"
      ],
      "name": "control_1_at_5b5s36",
      "capabilities" : "profile:control",
      "cpu": "4",
      "memory": "16384",
      "disk": "50",
      "arch": "x86_64",
      "pm_user": "root",
      "pm_addr": "10.xx.xx.24",
      "pm_password": "-----BEGIN RSA PRIVATE KEY-----"
```

```

$ABC123
-----END RSA PRIVATE KEY-----",
  "pm_type": "pxe_ssh"
}
,
{
  "mac": [
    "52:54:00:82:0d:9e"
  ],
  "name": "compute_1_at_5b5s36",
  "capabilities" : "profile:compute",
  "cpu": "4",
  "memory": "16384",
  "disk": "50",
  "arch": "x86_64",
  "pm_user": "root",
  "pm_addr": "10.xx.xx.24",
  "pm_password": "-----BEGIN RSA PRIVATE KEY-----
$ABC123
-----END RSA PRIVATE KEY-----",
  "pm_type": "pxe_ssh"
}
,
{
  "mac": [
    "52:54:00:a2:ff:7a"
  ],
  "name": "contrail-controller_1_at_5b5s36",
  "capabilities" : "profile:contrail-controller",
  "cpu": "4",
  "memory": "16384",
  "disk": "50",
  "arch": "x86_64",
  "pm_user": "root",
  "pm_addr": "10.xx.xx.24",
  "pm_password": "-----BEGIN RSA PRIVATE KEY-----
$ABC123
-----END RSA PRIVATE KEY-----",
  "pm_type": "pxe_ssh"
}
,
{
  "mac": [

```

```

    "52:54:00:51:35:bd"
  ],
  "name": "contrail-analytics_1_at_5b5s36",
  "capabilities" : "profile:contrail-analytics",
  "cpu": "4",
  "memory": "16384",
  "disk": "50",
  "arch": "x86_64",
  "pm_user": "root",
  "pm_addr": "10.xx.xx.24",
  "pm_password": "-----BEGIN RSA PRIVATE KEY-----
$ABC123
-----END RSA PRIVATE KEY-----",
  "pm_type": "pxe_ssh"
}
,
{
  "mac": [
    "52:54:00:a1:ae:4d"
  ],
  "name": "contrail-analytics-database_1_at_5b5s36",
  "capabilities" : "profile:contrail-analytics-database",
  "cpu": "4",
  "memory": "16384",
  "disk": "50",
  "arch": "x86_64",
  "pm_user": "root",
  "pm_addr": "10.xx.xx.24",
  "pm_password": "-----BEGIN RSA PRIVATE KEY-----
$ABC123
-----END RSA PRIVATE KEY-----",
  "pm_type": "pxe_ssh"
}
,
{
  "mac": [
    "52:54:00:8b:0e:b8"
  ],
  "name": "control_1_at_5b5s34",
  "capabilities" : "profile:control",
  "cpu": "4",
  "memory": "16384",
  "disk": "50",

```

```

    "arch": "x86_64",
    "pm_user": "root",
    "pm_addr": "10.xx.xx.22",
    "pm_password": "-----BEGIN RSA PRIVATE KEY-----
$ABC123
-----END RSA PRIVATE KEY-----",
    "pm_type": "pxe_ssh"
}
,
{
    "mac": [
        "52:54:00:c5:ba:b0"
    ],
    "name": "compute_1_at_5b5s34",
    "capabilities" : "profile:compute",
    "cpu": "4",
    "memory": "16384",
    "disk": "50",
    "arch": "x86_64",
    "pm_user": "root",
    "pm_addr": "10.xx.xx.22",
    "pm_password": "-----BEGIN RSA PRIVATE KEY-----
$ABC123
-----END RSA PRIVATE KEY-----",
    "pm_type": "pxe_ssh"
}
,
{
    "mac": [
        "52:54:00:b8:5b:aa"
    ],
    "name": "contrail-controller_1_at_5b5s34",
    "capabilities" : "profile:contrail-controller",
    "cpu": "4",
    "memory": "16384",
    "disk": "50",
    "arch": "x86_64",
    "pm_user": "root",
    "pm_addr": "10.xx.xx.22",
    "pm_password": "-----BEGIN RSA PRIVATE KEY-----
$ABC123
-----END RSA PRIVATE KEY-----",
    "pm_type": "pxe_ssh"
}

```

```

}
,
{
  "mac": [
    "52:54:00:2a:38:f1"
  ],
  "name": "contrail-analytics_1_at_5b5s34",
  "capabilities" : "profile:contrail-analytics",
  "cpu": "4",
  "memory": "16384",
  "disk": "50",
  "arch": "x86_64",
  "pm_user": "root",
  "pm_addr": "10.xx.xx.22",
  "pm_password": "-----BEGIN RSA PRIVATE KEY-----
$ABC123
-----END RSA PRIVATE KEY-----",
  "pm_type": "pxe_ssh"
}
,
{
  "mac": [
    "52:54:00:fc:b7:67"
  ],
  "name": "contrail-analytics-database_1_at_5b5s34",
  "capabilities" : "profile:contrail-analytics-database",
  "cpu": "4",
  "memory": "16384",
  "disk": "50",
  "arch": "x86_64",
  "pm_user": "root",
  "pm_addr": "10.xx.xx.22",
  "pm_password": "-----BEGIN RSA PRIVATE KEY-----
$ABC123
-----END RSA PRIVATE KEY-----",
  "pm_type": "pxe_ssh"
}
,
{
  "mac": [
    "52:54:00:48:b0:9b"
  ],
  "name": "control_1_at_5b5s37",

```

```

    "capabilities" : "profile:control",
    "cpu": "4",
    "memory": "16384",
    "disk": "50",
    "arch": "x86_64",
    "pm_user": "root",
    "pm_addr": "10.xx.xx.25",
    "pm_password": "-----BEGIN RSA PRIVATE KEY-----
$ABC123
-----END RSA PRIVATE KEY-----",
    "pm_type": "pxe_ssh"
}
,
{
    "mac": [
        "52:54:00:b3:01:b8"
    ],
    "name": "compute_1_at_5b5s37",
    "capabilities" : "profile:compute",
    "cpu": "4",
    "memory": "16384",
    "disk": "50",
    "arch": "x86_64",
    "pm_user": "root",
    "pm_addr": "10.xx.xx.25",
    "pm_password": "-----BEGIN RSA PRIVATE KEY-----
$ABC123
-----END RSA PRIVATE KEY-----",
    "pm_type": "pxe_ssh"
}
,
{
    "mac": [
        "52:54:00:9a:8c:f8"
    ],
    "name": "contrail-controller_1_at_5b5s37",
    "capabilities" : "profile:contrail-controller",
    "cpu": "4",
    "memory": "16384",
    "disk": "50",
    "arch": "x86_64",
    "pm_user": "root",
    "pm_addr": "10.xx.xx.25",

```



```

    "pm_password": "-----BEGIN RSA PRIVATE KEY-----
    $ABC123
-----END RSA PRIVATE KEY-----",
    "pm_type": "pxe_ssh"
  }
  ,
  {
    "mac": [
      "52:54:00:8d:3d:d9"
    ],
    "name": "contrail-analytics_1_at_5b5s37",
    "capabilities" : "profile:contrail-analytics",
    "cpu": "4",
    "memory": "16384",
    "disk": "50",
    "arch": "x86_64",
    "pm_user": "root",
    "pm_addr": "10.xx.xx.25",
    "pm_password": "-----BEGIN RSA PRIVATE KEY-----
    $ABC123
-----END RSA PRIVATE KEY-----",
    "pm_type": "pxe_ssh"
  }
  ,
  {
    "mac": [
      "52:54:00:9d:9e:57"
    ],
    "name": "contrail-analytics-database_1_at_5b5s37",
    "capabilities" : "profile:contrail-analytics-database",
    "cpu": "4",
    "memory": "16384",
    "disk": "50",
    "arch": "x86_64",
    "pm_user": "root",
    "pm_addr": "10.xx.xx.25",
    "pm_password": "-----BEGIN RSA PRIVATE KEY-----
    $ABC123
-----END RSA PRIVATE KEY-----",
    "pm_type": "pxe_ssh"
  }
]

```

```
}

```

## Adding a New Physical Compute Node

The following is a sample instackenv.json for adding a new physical compute node, by importing the physical compute or bare metal server to Ironic.

```
{
  "nodes": [
    {
      "mac": [
        "00:1b:21:99:ce:94"
      ],
      "name": "physical-compute_5b5s35",
      "capabilities": "profile:compute",
      "pm_user": "ADMIN",
      "pm_addr": "10.xx.xxx.206",
      "pm_password": "ADMIN",
      "pm_type": "pxe_ipmitool"
    }
  ]
}
```

The following are the mandatory parameters to import a physical compute or bare metal server to Ironic.

|                    |   |
|--------------------|---|
| <b>Pm_addr</b>     | Server's IPMI   |
| <b>Pm_user</b>     | IPMI user name  |
| <b>Pm_password</b> | IPMI password   |
| <b>MAC</b>         | MAC address of the server's NIC that is connected to the provisioning/control-plane network |
| <b>Pm_type</b>     | pxe_ipmitoo<br>Specify this driver to provision physical servers.                           |

## Requirements for Deploying to VMs

The following are required for deploying to VMs.

- Password-less SSH must be set up from the undercloud to all servers that will host overcloud VMs, for the user 'root'.
- Libvirtd on KVM hosts must be configured to allow TCP sessions without requiring Transport Layer Security (TLS).

## Sample Heat Templates for NICs

This section provides sample Heat templates for different configurations for NICs.

### Example 1: NIC-1 to control plane; NIC-2 bridged interface

This sample has the following topology:

- NIC-1 is connected to the control plane provisioning network  
Connected to an access port on the underlay switch
- NIC-2 is a bridged interface, and has a unique VLAN tag for each of the other overlay networks.

Underlying switch configuration:

- NIC-1 is connected to the control plane provisioning VLAN access-ports of a switch.
- NIC-2 is connected to trunk ports on the switch. The trunk ports will carry multiple VLAN tags, one each for the following networks:

VLAN-10: External VLAN

VLAN-20: Internal API VLAN

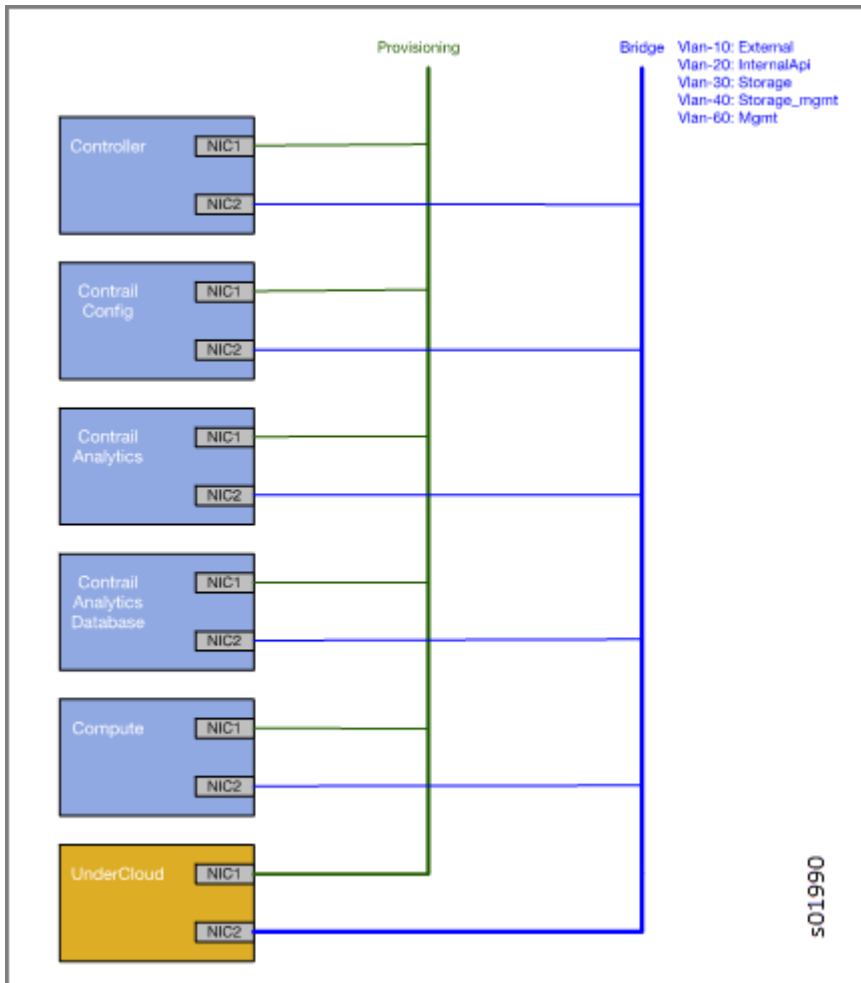
VLAN-30: Storage VLAN

VLAN-40: Storage-MGMT VLAN

VLAN-60: Management VLAN

[Figure 31 on page 220](#) shows the server NIC configuration for this example.

Figure 31: Server NIC Configuration



### NIC Template

The following is the NIC template to configure the setup in this example.

Note: For this setup, the default route is reachable by means of the InternalAPI network.

```
heat_template_version: 2015-04-30
```

```
description: >
```

```
Software Config to drive os-net-config to configure multiple interfaces
for the compute role.
```

```
parameters:
```

```
ControlPlaneIp:
```

```
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal API network
    type: string
InternalApiDefaultRoute: # Not used by default in this template
    default: '10.0.0.1'
    description: The default route of the internal api network.
    type: string
StorageIpSubnet:
    default: ''
    description: IP address/subnet on the storage network
    type: string
StorageMgmtIpSubnet:
    default: ''
    description: IP address/subnet on the storage mgmt network
    type: string
TenantIpSubnet:
    default: ''
    description: IP address/subnet on the tenant network
    type: string
ManagementIpSubnet: # Only populated when including environments/network-management.yaml
    default: ''
    description: IP address/subnet on the management network
    type: string
ExternalNetworkVlanID:
    default: 10
    description: Vlan ID for the external network traffic.
    type: number
InternalApiNetworkVlanID:
    default: 20
    description: Vlan ID for the internal_api network traffic.
    type: number
StorageNetworkVlanID:
    default: 30
    description: Vlan ID for the storage network traffic.
    type: number
```

```

StorageMgmtNetworkVlanID:
  default: 40
  description: Vlan ID for the storage mgmt network traffic.
  type: number
TenantNetworkVlanID:
  default: 50
  description: Vlan ID for the tenant network traffic.
  type: number
ManagementNetworkVlanID:
  default: 60
  description: Vlan ID for the management network traffic.
  type: number
ControlPlaneSubnetCidr: # Override this via parameter_defaults
  default: '24'
  description: The subnet CIDR of the control plane network.
  type: string
ControlPlaneDefaultRoute: # Override this via parameter_defaults
  description: The default route of the control plane network.
  type: string
ExternalInterfaceDefaultRoute: # Not used by default in this template
  default: '10.0.0.1'
  description: The default route of the external network.
  type: string
ManagementInterfaceDefaultRoute: # Commented out by default in this template
  default: unset
  description: The default route of the management network.
  type: string
DnsServers: # Override this via parameter_defaults
  default: []
  description: A list of DNS servers (2 max for some implementations) that will be added to
  resolv.conf.
  type: comma_delimited_list
EC2MetadataIp: # Override this via parameter_defaults
  description: The IP address of the EC2 metadata server.
  type: string
resources:
  OsNetConfigImpl:
    type: OS::Heat::StructuredConfig
    properties:
      group: os-apply-config
      config:
        os_net_config:
          network_config:

```

```

-
  type: interface
  name: nic1
  use_dhcp: false
  dns_servers: {get_param: DnsServers}
  addresses:
    -
      ip_netmask:
        list_join:
          - '/'
          - - {get_param: ControlPlaneIp}
            - {get_param: ControlPlaneSubnetCidr}
      routes:
        -
          ip_netmask: 1xx.254.1xx.254/32
          next_hop: {get_param: EC2MetadataIp}
-
  type: vlan
  use_dhcp: false
  vlan_id: {get_param: InternalApiNetworkVlanID}
  device: nic2
  addresses:
    -
      ip_netmask: {get_param: InternalApiIpSubnet}
  routes:
    -
      default: true
      next_hop: {get_param: InternalApiDefaultRoute}
-
  type: vlan
  vlan_id: {get_param: ManagementNetworkVlanID}
  device: nic2
  addresses:
    -
      ip_netmask: {get_param: ManagementIpSubnet}
-
  type: vlan
  vlan_id: {get_param: ExternalNetworkVlanID}
  device: nic2
  addresses:
    -
      ip_netmask: {get_param: ExternalIpSubnet}
-

```

```

    type: vlan
    vlan_id: {get_param: StorageNetworkVlanID}
    device: nic2
    addresses:
      -
        ip_netmask: {get_param: StorageIpSubnet}
      -
    type: vlan
    vlan_id: {get_param: StorageMgmtNetworkVlanID}
    device: nic2
    addresses:
      -
        ip_netmask: {get_param: StorageMgmtIpSubnet}

outputs:
  OS::stack_id:
    description: The OsNetConfigImpl resource.
    value: {get_resource: OsNetConfigImpl}

```

NIC definitions of the corresponding compute file are the following.

```

network_config:
  -
    type: interface
    name: nic1
    use_dhcp: false
    dns_servers: {get_param: DnsServers}
    addresses:
      -
        ip_netmask:
          list_join:
            - '/'
            - - {get_param: ControlPlaneIp}
              - {get_param: ControlPlaneSubnetCidr}
    routes:
      -
        ip_netmask: 169.xxx.xxx.254/32
        next_hop: {get_param: EC2MetadataIp}
      -
    type: interface
    name: vhost0
    use_dhcp: false

```



```

addresses:
  -
    ip_netmask: {get_param: InternalApiIpSubnet}
routes:
  -
    default: true
    next_hop: {get_param: InternalApiDefaultRoute}
-
type: vlan
vlan_id: {get_param: ExternalNetworkVlanID}
device: nic2
addresses:
  -
    ip_netmask: {get_param: ExternalIpSubnet}
-
type: vlan
vlan_id: {get_param: StorageNetworkVlanID}
device: nic2
addresses:
  -
    ip_netmask: {get_param: StorageIpSubnet}
-
type: vlan
vlan_id: {get_param: StorageMgmtNetworkVlanID}
device: nic2
addresses:
  -
    ip_netmask: {get_param: StorageMgmtIpSubnet}

```

### Example 2: NIC-1 to control plane; NIC-2 and NIC-3 bond interface; NIC-4 other networks

This sample has the following topology:

- NIC-1 is connected to the control plane provisioning network  
Connected to an access port on the underlay switch
- NIC-2 and NIC-3 are connected to the InternalAPI network.  
These two NICs are part of a bond interface.
- NIC-4 has a unique VLAN tag for each of the other overlay networks. It carries the rest of the networks.

Underlying switch configuration:

- NIC-1 is connected to the control plane provisioning VLAN access-ports of a switch.
- NIC-2 and NIC-3 connected to access ports on the switch in the InternalAPI VLAN. These switch ports are bundled together as a LAG
- NIC-4 is connected to trunk ports on the switch. The trunk ports will carry multiple VLAN tags, one each for the following networks:

VLAN-10: External VLAN

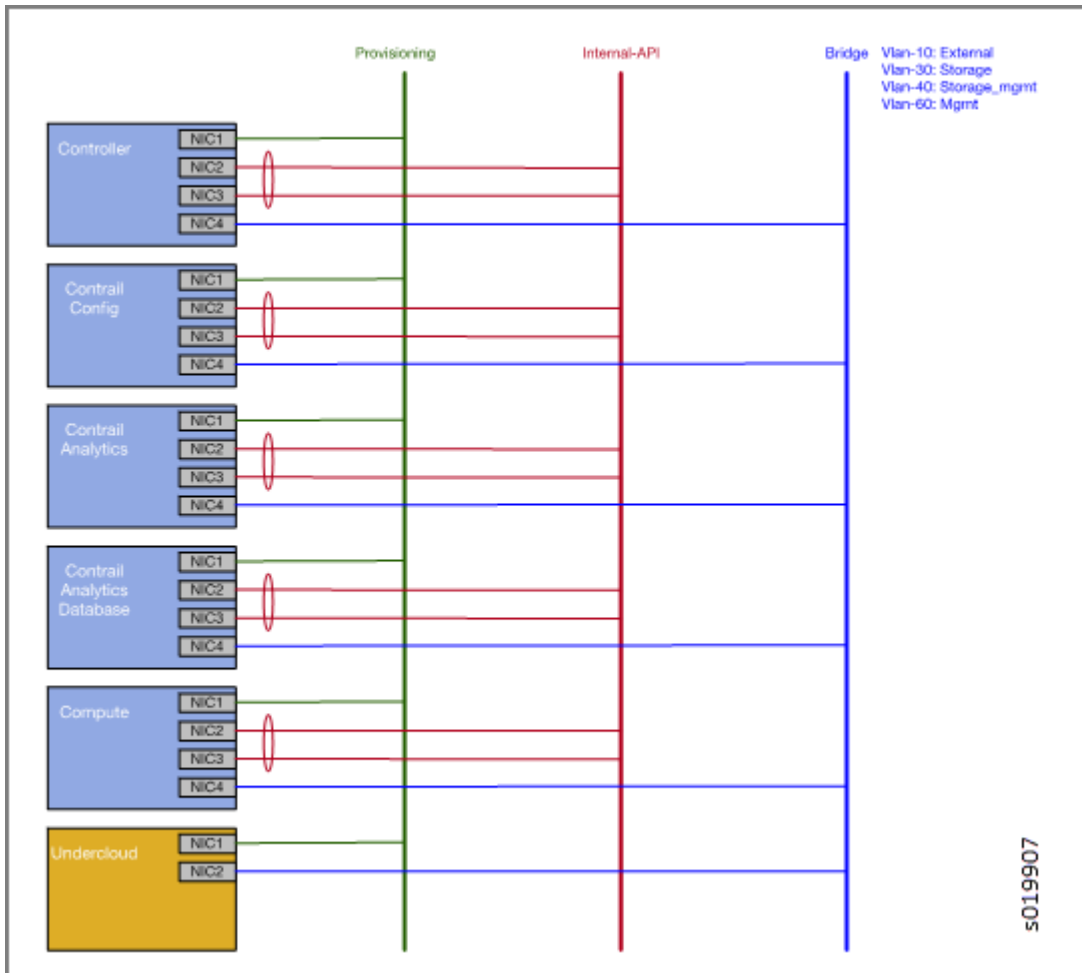
VLAN-30: Storage VLAN

VLAN-40: Storage-MGMT VLAN

VLAN-60: Management VLAN

[Figure 32 on page 227](#) shows the server NIC configuration for this example.

Figure 32: Server NIC Configuration



### NIC Template

The following is a snippet of the corresponding NIC template to configure the setup in this example.

Note: For this setup, the default route is reachable by means of the InternalAPI network.

```
resources:
  OsNetConfigImpl:
    type: OS::Heat::StructuredConfig
    properties:
      group: os-apply-config
      config:
        os_net_config:
          network_config:
```

```

-
  type: interface
  name: nic1
  use_dhcp: false
  dns_servers: {get_param: DnsServers}
  addresses:
    -
      ip_netmask:
        list_join:
          - '/'
          - - {get_param: ControlPlaneIp}
            - {get_param: ControlPlaneSubnetCidr}
  routes:
    -
      ip_netmask: 169.xxx.xxx.254/32
      next_hop: {get_param: EC2MetadataIp}
-
  type: linux_bond
  name: bond0
  use_dhcp: false
  addresses:
    -
      ip_netmask: {get_param: InternalApiIpSubnet}
  routes:
    -
      default: true
      next_hop: {get_param: InternalApiDefaultRoute}
  bonding_options: "mode=active-active"
  members:
    -
      type: interface
      name: nic2
    -
      type: interface
      name: nic3
-
-
  type: vlan
  vlan_id: {get_param: ManagementNetworkVlanID}
  device: nic4
  addresses:
    -
      ip_netmask: {get_param: ManagementIpSubnet}
-

```

```

    type: vlan
    vlan_id: {get_param: ExternalNetworkVlanID}
    device: nic4
    addresses:
    -
      ip_netmask: {get_param: ExternalIpSubnet}
-
  type: vlan
  vlan_id: {get_param: StorageNetworkVlanID}
  device: nic4
  addresses:
  -
    ip_netmask: {get_param: StorageIpSubnet}
-
  type: vlan
  vlan_id: {get_param: StorageMgmtNetworkVlanID}
  device: nic4
  addresses:
  -
    ip_netmask: {get_param: StorageMgmtIpSubnet}

outputs:
  OS::stack_id:
    description: The OsNetConfigImpl resource.
    value: {get_resource: OsNetConfigImpl}

```

## More Template Examples

More template examples are available in the directory:

```
/home/stack/tripleo-heat-templates/environments/contrail
```

There are separate templates for control-plane and compute. You can modify the example templates to match your topology.

```

[stack@instack contrail]$ pwd
/home/stack/tripleo-heat-templates/environments/contrail
[stack@instack contrail]$ ls -lrt | grep nic | grep compute
-rw-rw-r--. 1 stack stack 6136 May 31 15:07 contrail-nic-config-compute-bond-vlan.yaml
-rw-rw-r--. 1 stack stack 5839 May 31 15:07 contrail-nic-config-compute-bond-vlan-dpdk.yaml
-rw-rw-r--. 1 stack stack 5669 May 31 15:07 contrail-nic-config-compute-storage-mgmt.yaml
-rw-rw-r--. 1 stack stack 3864 May 31 15:07 contrail-nic-config-compute-single.yaml

```

```

-rw-rw-r--. 1 stack stack 5422 May 31 15:07 contrail-nic-config-compute-dpdk.yaml
-rw-rw-r--. 1 stack stack 5643 Jun  1 11:56 contrail-nic-config-compute-dpdk-bond-vlan.yaml
-rw-rw-r--. 1 stack stack 5661 Jun  2 12:43 contrail-nic-config-compute.yaml
[stack@instack contrail]$
[stack@instack contrail]$
[stack@instack contrail]$ ls -lrt | grep nic | grep -v compute
-rw-rw-r--. 1 stack stack 5568 May 31 15:07 contrail-nic-config-storage-mgmt.yaml
-rw-rw-r--. 1 stack stack 3861 May 31 15:07 contrail-nic-config-single.yaml
-rw-rw-r--. 1 stack stack 6688 May 31 15:07 contrail-nic-config-ovs-bond.yaml
-rw-rw-r--. 1 stack stack 5793 Jun  1 11:46 contrail-nic-config-vlan.yaml
-rw-rw-r--. 1 stack stack 5793 Jun  2 11:54 contrail-nic-config.yaml

```

## What are NIC Templates?

TripleO (OpenStack On OpenStack) provides the flexibility to have different NIC templates for different overcloud roles. For example, there might be differences between the NIC and networking layout for the overcloud-compute-nodes and the overcloud-contrail-controller-nodes.

## How NIC Templates Work

The NIC templates provide data to the backend scripts that take care of provisioning the network on the overcloud nodes. The templates are written in standard JSON formats.

The resources section within each template contains all of the networking information for the corresponding overcloud role, including:

- Number of NICs
- Network associated with each NIC
- Static routes associated with each NIC
- Any VLAN configuration which is tied to a particular NIC
  - Network associated with each VLAN interface
  - Static routes associated with each VLAN

For more information on what each of these sections looks like, see Red Hat documentation: [https://access.redhat.com/documentation/en-us/red\\_hat\\_openstack\\_platform/10/html/advanced\\_overcloud\\_customization/sect-isolating\\_networks](https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/10/html/advanced_overcloud_customization/sect-isolating_networks)

The Red Hat documentation has many examples of how to define a NIC within the template, and some of that information is used in the examples in this topic.

A limitation in Red Hat Director 10 is that all of the overcloud networks must be stretched at Layer 2 to all of the overcloud nodes. If the overcloud nodes are physical servers that are present in different racks or subnets of an IP fabric, then you'll have to first stretch all the overcloud networks to the physical servers. One way to do this is to use EVPN. If you have a traditional datacenter topology (non-IP fabric), then you can extend VLANs across the physical computes to extend all the overcloud networks.

Deploying an overcloud using TripleO and Director across multiple subnets is an upstream feature and a work-in-progress at this time. Upstream developers (mostly from Red Hat) are driving this effort. To check the status of this feature, see:

## Common Topologies

One of the most common topologies for a TripleO deployment consists of 3 NICs:

- NIC-1: Carries these networks:
  - Provisioning: Untagged
  - Management: Tagged
  - External: Tagged
- NIC-2: Carries internal-API network
- NIC-3: Carries tagged storage related networks (storage and storage management)

## Conventions in this Document

Examples are provided in this document.

- The topology used in the examples has the following constraints:
  - The first NIC must be connected to the ControlPlane network.
  - The second NIC must have separate VLAN interfaces for every other network.
- With the above limitations, 'eth1' is specified as the VlanParentInterface.
- Note that 'nic-2' is specified as the interface with multiple VLAN sub-interfaces in the NIC definition template.
- In the current version of RHEL 7.3/7.4, the NICs manifest as eth0, eth1, and so on. Because of this, NIC-2 translates to eth-1.

There are several NIC templates within Contrail that are available to users. These templates are named according to the topology that they're trying to solve, and are available in the `environments/contrail/`

directory. Please modify these templates according to your topology before deploying Contrail with TripleO/Red Hat Director.

## Contrail NIC Templates

As part of deployment, a network (net) template must be provided. The net template files are all available at the same location:

### Sample Net Templates

```
[stack@undercloud contrail]$ ls -lrt | grep contrail-net
-rw-rw-r--. 1 stack stack 1866 Sep 19 17:10 contrail-net-storage-mgmt.yaml
-rw-rw-r--. 1 stack stack  894 Sep 19 17:10 contrail-net-single.yaml
-rw-rw-r--. 1 stack stack 1528 Sep 19 17:10 contrail-net-dpdk.yaml
-rw-rw-r--. 1 stack stack 1504 Sep 19 17:10 contrail-net-bond-vlan.yaml
-rw-rw-r--. 1 stack stack 1450 Sep 19 17:12 contrail-net.yaml
```

The template files are prepopulated examples that are included with a Contrail package. The file names match the use case that each is trying to solve. For example, use the `contrail-net-dpdk.yaml` file if your use case includes a DPDK compute. Similarly, use the `contrail-net-bond-vlan.yaml` file if your topology uses bond interfaces and VLAN subinterfaces that need to be created on top of the bond interfaces.

Please note that these are example files, and you'll need to modify them to match your topology.

### Resource Registry Example

The `resource_registry` section of the net template file specifies which NIC template must be used for each role:

### Sample Resource Registry of Net Template

```
[stack@undercloud contrail]$ cat contrail-net.yaml
resource_registry:
  OS::TripleO::Compute::Net::SoftwareConfig: contrail-nic-config-compute.yaml
  OS::TripleO::ContrailDpdk::Net::SoftwareConfig: contrail-nic-config-compute-dpdk-bond-vlan.yaml
  OS::TripleO::Controller::Net::SoftwareConfig: contrail-nic-config.yaml
  OS::TripleO::ContrailController::Net::SoftwareConfig: contrail-nic-config.yaml
  OS::TripleO::ContrailAnalytics::Net::SoftwareConfig: contrail-nic-config.yaml
  OS::TripleO::ContrailAnalyticsDatabase::Net::SoftwareConfig: contrail-nic-config.yaml
  OS::TripleO::ContrailTsn::Net::SoftwareConfig: contrail-nic-config-compute.yaml

parameter_defaults:
```



```

ControlPlaneSubnetCidr: '24'
InternalApiNetCidr: 10.0.0.0/24
InternalApiAllocationPools: [{'start': '10.0.0.10', 'end': '10.0.0.200'}]
InternalApiDefaultRoute: 10.0.0.1
ManagementNetCidr: 10.1.0.0/24
ManagementAllocationPools: [{'start': '10.1.0.10', 'end': '10.1.0.200'}]
ManagementInterfaceDefaultRoute: 10.1.0.1
ExternalNetCidr: 10.2.0.0/24
ExternalAllocationPools: [{'start': '10.2.0.10', 'end': '10.2.0.200'}]
EC2MetadataIp: 192.0.2.1 # Generally the IP of the Undercloud
DnsServers: ["8.8.8.8", "8.8.4.4"]
VrouterPhysicalInterface: vlan20
VrouterGateway: 10.0.0.1
VrouterNetmask: 255.255.255.0
ControlVirtualInterface: eth0
PublicVirtualInterface: vlan10
VlanParentInterface: eth1 # If VrouterPhysicalInterface is a vlan interface using vlanX
notation

```

## NIC Templates for Control Nodes

In this example, all of the OpenStack controller and the Contrail control plane roles use the NIC template named `contrail-nic-config.yaml`. Note that the compute roles and the DPDK roles use different NIC templates.

The NIC template files can be accessed at this location:

### Sample NIC Templates

```

[stack@undercloud contrail]$ ls -lrt | grep contrail-nic-config-
-rw-rw-r--. 1 stack stack 5615 Sep 19 17:10 contrail-nic-config-vlan.yaml
-rw-rw-r--. 1 stack stack 5568 Sep 19 17:10 contrail-nic-config-storage-mgmt.yaml
-rw-rw-r--. 1 stack stack 3861 Sep 19 17:10 contrail-nic-config-single.yaml
-rw-rw-r--. 1 stack stack 5669 Sep 19 17:10 contrail-nic-config-compute-storage-mgmt.yaml
-rw-rw-r--. 1 stack stack 3864 Sep 19 17:10 contrail-nic-config-compute-single.yaml
-rw-rw-r--. 1 stack stack 5385 Sep 19 17:10 contrail-nic-config-compute-dpdk.yaml
-rw-rw-r--. 1 stack stack 5839 Sep 19 17:10 contrail-nic-config-compute-bond-vlan.yaml
-rw-rw-r--. 1 stack stack 5666 Sep 19 17:10 contrail-nic-config-compute-bond-vlan-dpdk.yaml
-rw-rw-r--. 1 stack stack 5538 Sep 19 17:10 contrail-nic-config-compute-bond-dpdk.yaml
-rw-rw-r--. 1 stack stack 5132 Sep 19 17:13 contrail-nic-config-compute.yaml
-rw-r--r--. 1 stack stack 5503 Sep 19 17:13 contrail-nic-config-compute-dpdk-bond-vlan.yaml

```

Just like the network template files, these NIC template files are examples which are included with the Contrail package. These files also have their names matching the use case that they're trying to solve.

Note that these NIC template files are examples, and you may have to modify these according to your cluster's topology.

Also, these examples call out NIC names in the format of `nic1`, `nic2`, `nic3`, and so on (`nic.<number>`). Think of these as variables, and Director's backend scripts translate these NIC numbers into actual interface names based on the interface boot order. So if you specify `nic1`, `nic2`, and `nic3` in the template and the boot order of interfaces is `eth0`, `eth1`, and `eth2`, then the mapping of these `nic` variables to actual interfaces would look like:

- Nic1 mapped to eth0
- Nic2 mapped to eth1
- Nic3 mapped to eth2

TripleO also provides the flexibility to use actual NIC names (`eth0`, `em1`, `ens2f`, and so on) in the NIC templates instead of using `nic1`, `nic2`, `nic3`, and the like.

**NOTE:** A common mistake while defining NIC templates is that the boot order of NICs is not set correctly. Because of this, your deployment might progress beyond the network configuration stage, but there might be connectivity issues because the IP/Subnet/route information might not be configured correctly for the NICs of overcloud nodes.

This section takes a zoom-in look at the `network_config` section of the NIC template used by the controllers: `contrail-nic-config.yaml`.

### Sample Network Config for Control Nodes

```
network_config:
-
  type: interface
  name: nic1
  use_dhcp: false
  dns_servers: {get_param: DnsServers}
  addresses:
  -
    ip_netmask:
      list_join:
      - '/'
      - - {get_param: ControlPlaneIp}
```

```

        - {get_param: ControlPlaneSubnetCidr}
    routes:
    -
        ip_netmask: 169.254.169.254/32
        next_hop: {get_param: EC2MetadataIp}
    -
        type: vlan
        use_dhcp: false
        vlan_id: {get_param: InternalApiNetworkVlanID}
        device: nic2
        addresses:
        -
            ip_netmask: {get_param: InternalApiIpSubnet}
    routes:
    -
        default: true
        next_hop: {get_param: InternalApiDefaultRoute}
    -
        type: vlan
        vlan_id: {get_param: ManagementNetworkVlanID}
        device: nic2
        addresses:
        -
            ip_netmask: {get_param: ManagementIpSubnet}
    -
        type: vlan
        vlan_id: {get_param: ExternalNetworkVlanID}
        device: nic2
        addresses:
        -
            ip_netmask: {get_param: ExternalIpSubnet}
    -
        type: vlan
        vlan_id: {get_param: StorageNetworkVlanID}
        device: nic2
        addresses:
        -
            ip_netmask: {get_param: StorageIpSubnet}
    -
        type: vlan
        vlan_id: {get_param: StorageMgmtNetworkVlanID}
        device: nic2
        addresses:

```

```
-
  ip_netmask: {get_param: StorageMgmtIpSubnet}
```

## NIC Control Node Template Subsection: Definition for NIC1

The subsection of the template for NIC1 includes the following.

- The definition for an interface called 'nic1'
- The DNS server is defined. Make sure that this parameter has a valid value. Most commonly, this variable is assigned a value in the `contrail-services.yaml` file.
- An IP and subnet is provided under the 'addresses' section. Note that these values are also variables, and the format is: `$(Network_Name)IP` and `$(Network_Name)SubnetCidr`.
  - This means that this particular NIC is on the ControlPlane network. In the background, this NIC might be connected to an access port on a switch for the ControlPlane VLAN.
- In the 'routes' section, there's a /32 route out of this NIC. At the time of planning the networking for your cluster, you may need to provision static routes on the overcloud roles. Use the format mentioned under the 'routes' section to specify any such static routes.

### Sample NIC1

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

## NIC Template Subsection: Definition for NIC2

The subsection of the template for NIC2 includes the following.

- The NIC2 has multiple VLANs defined on it.
  - In the background, NIC2 might be connected to a switch's trunk port, and all of the corresponding VLANs must be allowed on the trunk.
  - Because Director-based deployments need the administrator to use a number of networks, it's a very common requirement or design to use VLAN interfaces on the overcloud nodes. Consequently, the administrators do not have to be concerned about having 6-7 physical NICs on each overcloud node.
- For each VLAN interface, the `vlan_id` is defined. Note that the `vlan_id` points to a variable. As with the example for NIC1, these variables can be assigned values in the `contrail-net.yaml`.
- Another important observation is the setting of the default route. In this example, the default route was provisioned on the VLAN interface in the InternalAPI network. Note that the next hop points to a variable. As with other variables, this variable can be set in the `contrail-net.yaml` file. The following snippet shows the default route information.

### Sample Default Route Information

```
-
  type: vlan
  use_dhcp: false
  vlan_id: {get_param: InternalApiNetworkVlanID}
  device: nic2
  addresses:
    -
      ip_netmask: {get_param: InternalApiIpSubnet}
  routes:
    -
      default: true
      next_hop: {get_param: InternalApiDefaultRoute}
```

## NIC Templates for Compute Nodes

The NIC definitions for compute roles are slightly different from the definitions for control nodes. This is because Contrail provisions a logical interface called 'vhost0' on all compute nodes, and this interface must be provided in the NIC definition file for a compute node. Vhost0 is the logical interface that gets attached to the control data network (or the InternalAPI network in TripleO-based installation).

In the `contrail-net.yaml` example provided in the beginning of this topic, the NIC template used for the compute nodes is `contrail-nic-config-compute.yaml`. The following is the 'resources' section of the `contrail-nic-config-compute.yaml` file:

### Sample Resources for Compute Nodes

```
resources:
  OsNetConfigImpl:
    type: OS::Heat::StructuredConfig
    properties:
      group: os-apply-config
      config:
        os_net_config:
          network_config:
            -
              type: interface
              name: nic1
              use_dhcp: false
              dns_servers: {get_param: DnsServers}
              addresses:
                -
                  ip_netmask:
                    list_join:
                      - '/'
                      - {get_param: ControlPlaneIp}
                      - {get_param: ControlPlaneSubnetCidr}
              routes:
                -
                  ip_netmask: 169.254.169.254/32
                  next_hop: {get_param: EC2MetadataIp}
            -
              type: vlan
              vlan_id: {get_param: InternalApiNetworkVlanID}
              device: nic2
            -
              type: interface
              name: vhost0
              use_dhcp: false
              addresses:
                -
                  ip_netmask: {get_param: InternalApiIpSubnet}
              routes:
```

```

-
  default: true
  next_hop: {get_param: InternalApiDefaultRoute}
-
  type: vlan
  vlan_id: {get_param: ManagementNetworkVlanID}
  device: nic2
  addresses:
  -
    ip_netmask: {get_param: ManagementIpSubnet}
-
  type: vlan
  vlan_id: {get_param: ExternalNetworkVlanID}
  device: nic2
  addresses:
  -
    ip_netmask: {get_param: ExternalIpSubnet}
-
  type: vlan
  vlan_id: {get_param: StorageNetworkVlanID}
  device: nic2
  addresses:
  -
    ip_netmask: {get_param: StorageIpSubnet}
-
  type: vlan
  vlan_id: {get_param: StorageMgmtNetworkVlanID}
  device: nic2
  addresses:
  -
    ip_netmask: {get_param: StorageMgmtIpSubnet}

```

### NIC Compute Node Template Subsection: Definition for NIC1

This section is very similar to the NIC1 definition template for the control nodes. In this example topology, the first NIC for all the compute nodes is connected to the ControlPlane network. Note that this is untagged, so this NIC might be connected to an access port on the underlay switch.

#### Sample NIC1 for Compute Node

```

-
  type: interface

```

```

name: nic1
use_dhcp: false
dns_servers: {get_param: DnsServers}
addresses:
  -
    ip_netmask:
      list_join:
        - '/'
        - - {get_param: ControlPlaneIp}
          - {get_param: ControlPlaneSubnetCidr}
routes:
  -
    ip_netmask: 169.254.169.254/32
    next_hop: {get_param: EC2MetadataIp}

```

### NIC Compute Node Template Subsection: Definition for NIC2

This section is very similar to the NIC2 definition template for the control nodes, however there are two major differences:

- The VLAN subinterface for `InternalApiNetwork` does not have an IP address.
- The `Vhost0` interface holds the IP address for `InternalApiNetwork`.
  - If you're using stock TripleO-based installation, then the IP address for the `InternalApiNetwork` will always be configured on the `vhost0` interface.

### Sample NIC2 for Compute Node

```

-
  type: interface
  name: vhost0
  use_dhcp: false
  addresses:
    -
      ip_netmask: {get_param: InternalApiIpSubnet}
  routes:
    -
      default: true
      next_hop: {get_param: InternalApiDefaultRoute}
-
  type: vlan
  vlan_id: {get_param: InternalApiNetworkVlanID}

```



```

device: nic2
-
  type: vlan
  vlan_id: {get_param: ManagementNetworkVlanID}
  device: nic2
  addresses:
    -
      ip_netmask: {get_param: ManagementIpSubnet}
-
  type: vlan
  vlan_id: {get_param: ExternalNetworkVlanID}
  device: nic2
  addresses:
    -
      ip_netmask: {get_param: ExternalIpSubnet}
-
  type: vlan
  vlan_id: {get_param: StorageNetworkVlanID}
  device: nic2
  addresses:
    -
      ip_netmask: {get_param: StorageIpSubnet}
-
  type: vlan
  vlan_id: {get_param: StorageMgmtNetworkVlanID}
  device: nic2
  addresses:
    -
      ip_netmask: {get_param: StorageMgmtIpSubnet}

```

The following are additional parameters that are required to successfully provision compute nodes. The parameters are handled as variables and are normally specified in the `contrail-net.yaml` file.

- Network-related parameters:
  - Subnet CIDR: You can set the subnet mask of each overcloud network in this file.
  - Allocation Pool Range: If set, then the overcloud nodes are allocated IP addresses from the specified range
  - Default Route: Set the next hop for the default route in the specified format. In this example, the default route is set for InternalApi network and the next hop is set as 10.0.0.1

- `VrouterPhysicalInterface`: This is the interface on which `vhost0` interface gets attached. This may be a physical NIC (e.g. `eth2` or `enps0f0`), or a VLAN interface (e.g. `Vlan20`)
- `VrouterGateway`: This is the IP address of the SDN gateway. In a lot of deployments, this might be the IP address of the MX router's IP address. This IP must be reachable via the InternalAPI network
- `VrouterNetmask`: subnet mask for the `vhost0` interface (this is provisioned in the compute nodes' config files).
- `VlanParentInterface`: This is optional, and needed only if `vhost0` needs to be attached to a VLAN interface.

### Sample NIC2 Additional Parameters for Compute Node

```
parameter_defaults:
  ControlPlaneSubnetCidr: '24'
  InternalApiNetCidr: 10.0.0.0/24
  InternalApiAllocationPools: [{'start': '10.0.0.10', 'end': '10.0.0.200'}]
  InternalApiDefaultRoute: 10.0.0.1
  ManagementNetCidr: 10.1.0.0/24
  ManagementAllocationPools: [{'start': '10.1.0.10', 'end': '10.1.0.200'}]
  ManagementInterfaceDefaultRoute: 10.1.0.1
  ExternalNetCidr: 10.2.0.0/24
  ExternalAllocationPools: [{'start': '10.2.0.10', 'end': '10.2.0.200'}]
  EC2MetadataIp: 192.0.2.1 # Generally the IP of the Undercloud
  DnsServers: ["8.8.8.8", "8.8.4.4"]
  VrouterPhysicalInterface: vlan20
  VrouterGateway: 10.0.0.1
  VrouterNetmask: 255.255.255.0
  ControlVirtualInterface: eth0
  PublicVirtualInterface: vlan10
```

### NIC Templates for DPDK Compute Nodes

You can either use a separate YAML template for DPDK compute nodes or use the `contrai-net.yaml` template file. If you use the `contrai-net.yaml` template file, you must add the following additional parameters:

```
VrouterDpdkPhysicalInterface: bond0
  ContrailVrouterDpdkPhysicalInterface: bond0
  BondDpdkInterface: bond0
  ContrailBondDpdkInterface: bond0
```

```
BondDpdkInterfaceMembers: 'ens7f0'
BondInterfaceMembers: 'ens7f0'
```

### Sample Config for DPDK Compute Nodes

```
heat_template_version: 2015-04-30

description: >
  Software Config to drive os-net-config to configure multiple interfaces
  for the compute role.

parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
  InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal API network
    type: string
  InternalApiDefaultRoute: # Not used by default in this template
    default: '10.88.0.1'
    description: The default route of the internal api network.
    type: string
  StorageIpSubnet:
    default: ''
    description: IP address/subnet on the storage network
    type: string
  StorageMgmtIpSubnet:
    default: ''
    description: IP address/subnet on the storage mgmt network
    type: string
  TenantIpSubnet:
    default: ''
    description: IP address/subnet on the tenant network
    type: string
  ManagementIpSubnet: # Only populated when including environments/network-management.yaml
    default: ''
```

```
description: IP address/subnet on the management network
type: string
ExternalNetworkVlanID:
  default: 10
  description: Vlan ID for the external network traffic.
  type: number
InternalApiNetworkVlanID:
  default: 20
  description: Vlan ID for the internal_api network traffic.
  type: number
StorageNetworkVlanID:
  default: 30
  description: Vlan ID for the storage network traffic.
  type: number
StorageMgmtNetworkVlanID:
  default: 40
  description: Vlan ID for the storage mgmt network traffic.
  type: number
TenantNetworkVlanID:
  default: 50
  description: Vlan ID for the tenant network traffic.
  type: number
ManagementNetworkVlanID:
  default: 60
  description: Vlan ID for the management network traffic.
  type: number
ControlPlaneSubnetCidr: # Override this via parameter_defaults
  default: '24'
  description: The subnet CIDR of the control plane network.
  type: string
ControlPlaneDefaultRoute: # Override this via parameter_defaults
  description: The default route of the control plane network.
  type: string
ExternalInterfaceDefaultRoute: # Not used by default in this template
  default: '10.88.0.1'
  description: The default route of the external network.
  type: string
ManagementInterfaceDefaultRoute: # Commented out by default in this template
  default: unset
  description: The default route of the management network.
  type: string
DnsServers: # Override this via parameter_defaults
  default: []
```

```

description: A list of DNS servers (2 max for some implementations) that will be added to
resolv.conf.

```

```

type: comma_delimited_list

```

```

EC2MetadataIp: # Override this via parameter_defaults

```

```

description: The IP address of the EC2 metadata server.

```

```

type: string

```

```

resources:

```

```

  OsNetConfigImpl:

```

```

    type: OS::Heat::StructuredConfig

```

```

    properties:

```

```

      group: os-apply-config

```

```

      config:

```

```

        os_net_config:

```

```

          network_config:

```

```

            -

```

```

              type: interface

```

```

              name: nic1

```

```

              use_dhcp: false

```

```

              dns_servers: {get_param: DnsServers}

```

```

              addresses:

```

```

                -

```

```

                  ip_netmask:

```

```

                    list_join:

```

```

                      - '/'

```

```

                      - - {get_param: ControlPlaneIp}

```

```

                      - {get_param: ControlPlaneSubnetCidr}

```

```

              routes:

```

```

                -

```

```

                  ip_netmask: 169.254.169.254/32

```

```

                  next_hop: {get_param: EC2MetadataIp}

```

```

            -

```

```

              type: linux_bond

```

```

              name: bond0

```

```

              use_dhcp: false

```

```

              bonding_options: "mode=802.3ad lacp_rate=fast updelay=1000 miimon=100"

```

```

            -

```

```

              type: interface

```

```

              name: vhost0

```

```

              use_dhcp: false

```

```

              addresses:

```

```

                -

```

```

                  ip_netmask: {get_param: InternalApiIpSubnet}

```

```

routes:
  -
    default: true
    next_hop: {get_param: InternalApiDefaultRoute}
  -
type: linux_bridge
name: br0
use_dhcp: false
members:
  -
    type: interface
    name: nic2
    # force the MAC address of the bridge to this interface
    #
    primary: true
    #
  -
type: vlan
vlan_id: {get_param: ManagementNetworkVlanID}
device: br0
addresses:
  -
    ip_netmask: {get_param: ManagementIpSubnet}
  -
type: vlan
vlan_id: {get_param: ExternalNetworkVlanID}
device: br0
addresses:
  -
    ip_netmask: {get_param: ExternalIpSubnet}
  -
type: vlan
vlan_id: {get_param: StorageNetworkVlanID}
device: br0
addresses:
  -
    ip_netmask: {get_param: StorageIpSubnet}
  -
type: vlan
vlan_id: {get_param: StorageMgmtNetworkVlanID}
device: br0
addresses:
  -
    ip_netmask: {get_param: StorageMgmtIpSubnet}

```

```
outputs:  
  OS::stack_id:  
    description: The OsNetConfigImpl resource.  
    value: {get_resource: OsNetConfigImpl}
```

## Installing Red Hat OpenShift Container Platform with Contrail Networking

### IN THIS SECTION

- [Launch Instances \(Azure, AWS, or Baremetal\) | 247](#)
- [Host Registration | 248](#)
- [Install Base Packages | 248](#)
- [Install OpenShift with Contrail Networking | 250](#)
- [Installing a Contrail System on an Existing OpenShift Setup | 252](#)

Perform the following steps to install Red Hat OpenShift Container Platform version 3.7 with Juniper Networks Contrail Networking Release 4.1. These instructions are valid for systems with Microsoft Azure, Amazon Web Services (AWS), or baremetal systems (BMS).

### Launch Instances (Azure, AWS, or Baremetal)

Launch instances in the same subnet, using the following minimum configuration guidelines.

- **Master Node** (x1 or x3 for high availability)
  - Image: RHEL 7.3 or 7.4
  - CPU/RAM: 4 CPU, 32 GB RAM
  - Disk: 250 GB
  - Security Group: Allow all traffic from everywhere
- **Slave Node** (x*n*)
  - Image: RHEL 7.3 or 7.4

- CPU/RAM: 8 CPU, 64 GB RAM
- Disk: 250 G
- Security Group: Allow all traffic from everywhere
- **Load Balancer Node** (x1, only when using high availability. Not needed for single master node installation)
  - Image: RHEL 7.3 or 7.4
  - CPU/RAM: 2 CPU, 16 GB RAM
  - Disk: 100 G
  - Security Group: Allow all traffic from everywhere

## Host Registration

Use the following procedure to register all nodes in the cluster.

1. Register all nodes in cluster using Red Hat Subscription Manager (RHSM).

```
(all-nodes)# subscription-manager register --username <username> --password <password> --force
```

2. List the available subscriptions.

```
(all-nodes)# subscription-manager list --available --matches '*OpenShift*'
```

3. From the list of available subscriptions, find the pool ID for the OpenShift Container Platform subscription and attach it.

```
(all-nodes)# subscription-manager attach --pool=<pool-ID>
```

4. Disable all yum repositories.

```
(all-nodes)# subscription-manager repos --disable="*"
```

5. Enable only the repositories required by OpenShift Container Platform 3.7.

```
(all-nodes)# subscription-manager repos \
    --enable="rhel-7-server-rpms" \
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-ose-3.7-rpms" \
    --enable="rhel-7-fast-datapath-rpms"
```

## Install Base Packages

1. Install Extra Packages for Enterprise Linux (EPEL).



```
(all-nodes)# yum install wget -y && wget -O /tmp/epel-release-latest-7.noarch.rpm https://
dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm && rpm -ivh /tmp/epel-release-
latest-7.noarch.rpm
```

**2. Update the system to use the latest packages.**

```
(all-nodes)# yum update -y
```

**3. Install the atomic package that provides the OpenShift Container Platform utilities.**

```
(all-nodes)# yum install atomic-openshift-excluder atomic-openshift-utils git -y
```

**4. Use unexclude to remove the atomic-openshift packages from the list for the duration of the installation.**

```
(all-nodes)# atomic-openshift-excluder unexclude -y
```

**5. Enable SSH access for the root user.**

```
(all-nodes)# sudo su
(all-nodes)# passwd
(all-nodes)# sed -i -e 's/#PermitRootLogin yes/PermitRootLogin yes/g' -e 's/
PasswordAuthentication no/PasswordAuthentication yes/g' /etc/ssh/sshd_config
(all-nodes)# service sshd restart
(all-nodes)# logout
```

After logout, log in as root user.

**6. Enforce the SELinux security policy.**

```
(all-nodes)# vi /etc/selinux/config

SELINUX=enforcing
```

**7. Add a static entry for master and slaves in `/etc/hosts` and test with ping. Be sure to use the output of “`hostname -f`” to populate the file.**

```
(all-nodes)# vi /etc/hosts

10.xx.vv.1 master.test.net master
10.xx.xx.2 slave.test.net slave

(all-nodes)# ping master
(all-nodes)# ping slave
```

8. Enable passwordless SSH access.

```
(ansible-node)# ssh-keygen -t rsa
(ansible-node)# ssh-copy-id root@<master>

(ansible-node)# ssh-copy-id root@<slave>
```

9. Sync NTP.

```
(all-nodes)# service ntpd stop
(all-nodes)# ntpdate -s time.nist.gov
(all-nodes)# service ntpd start
```

## Install OpenShift with Contrail Networking

1. Download the Contrail Docker images from the Juniper software download site: <https://www.juniper.net/support/downloads/?p=contrail#sw>.

Image 4.1 (Red Hat Enterprise Linux 7.X):contrail-kubernetes-docker-images\_4.1.0.0-8.tgz

```
(ansible-node)# cd /tmp && wget <contrail-container-image.tgz>
```

2. Clone the openshift-ansible repo.

```
(ansible-node)# cd /root
(ansible-node)# git clone https://github.com/savithruml/openshift-ansible -b contrail-openshift
```

3. Copy the install files.

```
(ansible-node)# wget -O /root/openshift-ansible/inventory/byo/ose-prerequisites.yml https://raw.githubusercontent.com/savithruml/openshift-contrail/master/openshift/install-files/all-in-one/ose-prerequisites.yml
(ansible-node)# wget -O /root/openshift-ansible/inventory/byo/ose-install https://raw.githubusercontent.com/savithruml/openshift-contrail/master/openshift/install-files/all-in-one/ose-install
```

4. Populate the install file with Contrail configuration parameters specific to your system. Refer to the following example.

Be sure to add the masters in the [nodes] section of the inventory, to ensure that the Contrail control pods will come up on OpenShift masters.

```
(ansible-node)# vi /root/openshift-ansible/inventory/byo/ose-install

[OSEv3:vars]
...
os_sdn_network_plugin_name='cni'
openshift_use_contrail=true
contrail_os_release=redhat7
contrail_version=4.1.0.0-8
analyticsdb_min_diskgb=50
configdb_min_diskgb=25
vrouters_physical_interface=en01
contrail_docker_images_path=/tmp
cni_version=v0.5.2
...
```

For an example for a single master, see <https://github.com/savithruml/openshift-contrail/blob/master/openshift/install-files/all-in-one/ose-install>

For an example for a HA master, see <https://github.com/savithruml/openshift-contrail/blob/master/openshift/install-files/all-in-one/ose-install-ha>

5. Run the Ansible playbook to install the OpenShift container platform with Contrail Networking.

```
(ansible-node)# cd /root/openshift-ansible
(ansible-node)# ansible-playbook -i inventory/byo/ose-install inventory/byo/ose-
prerequisites.yml
(ansible-node)# ansible-playbook -i inventory/byo/ose-install playbooks/byo/
openshift_facts.yml
(ansible-node)# ansible-playbook -i inventory/byo/ose-install playbooks/byo/config.yml
```

6. Verify that Contrail has been installed and is operational.

```
(master)# oc get ds -n kube-system
(master)# oc get pods -n kube-system
```

7. Create a password for the admin user to log in to the Contrail UI.

```
(master-node)# htpasswd /etc/origin/master/htpasswd admin
(master-node)# oc login -u admin
```

8. Use the following to access the Contrail and OpenShift Web user interfaces, and attempt to log in to each.

```
Contrail: https://<master-node-ip>:8143
OpenShift: https://<master-node-ip>:8443
```

**NOTE:** If access and log in is unsuccessful, flush the iptables.

9. Perform the following setups in the Contrail UI.
  - Set up BGP peering with the gateway router.
 

**Configure > Infrastructure > BGP Routers**
  - Set up a network IPAM under the “default” project.
 

**Configure > Networking > IP Address Management > default-domain > default**
  - Create a public virtual network.
 

**Configure > Networking > Networks > default-domain > default**

## Installing a Contrail System on an Existing OpenShift Setup

1. Remove any existing SDN system, such as OVS, Calico, Nuage, and the like. Use removal instructions as published by the vendor of the existing system.
2. Download the contrail-container-image package from the Juniper site. Untar the package and load the containers.

```
(all-nodes)# wget <contrail-container-image.tgz> && tar -xvzf <contrail-container-image.tgz>
```

```
(all-nodes)# docker load <contrail-container-image.tgz>
```

3. The following Docker containers must be on the masters.
  - contrail-controller
  - contrail-analytics

- contrail-analyticsdb
  - contrail-kube-manager
4. The following Docker containers must be on the minions.
    - contrail-agent
    - contrail-kubernetes-agent
  5. Add contrail and daemon-set-controller to the OpenShift privileged security context constraints (scc).

```
(master)# oadm policy add-scc-to-user privileged system:serviceaccount:kube-system:contrail
(master)# oadm policy add-scc-to-user privileged system:serviceaccount:kube-system:daemon-
set-controller
```

6. Label the master nodes prior to launching the Contrail pods.

```
(master)# oc label nodes <all-master-nodes> opencontrail.org/controller=true
```

7. Make the masters schedulable.

```
(master)# oadm manage -<all-master-nodes> --schedulable
```

8. Open relevant Contrail ports in the iptables.

On master instances, refer to the following to open ports:

<https://github.com/savithruml/openshift-contrail/blob/master/openshift/install-files/all-in-one/iptables-master>

On node instances, refer to the following to open ports:

<https://github.com/savithruml/openshift-contrail/blob/master/openshift/install-files/all-in-one/iptables-node>

9. Populate the single YAML file with your environment variables and launch the installer.

```
(master)# wget https://raw.githubusercontent.com/savithruml/openshift-contrail/master/
openshift/install-files/all-in-one/contrail-installer.yaml
(master)# oc create -f contrail-installer.yml
```

10. Verify that all services are up and running.

```
(master)# oc get ds -n kube-system
(master)# oc get pods -n kube-system
(master)# oc exec <contrail-pod-name> contrail-status -n kube-system
```

11. Create a password for the admin user to log in to the UI.

```
(master-node)# htpasswd /etc/origin/master/htpasswd admin
(master-node)# oc login -u admin
```

12. Patch the scc restricted.

```
master-node)# oc patch scc restricted --patch='{ "runAsUser": { "type": "RunAsAny" } }'
```

13. Use the following to access the Contrail and OpenShift Web user interfaces, and log in to each.

```
Contrail: https://<master-node-ip>:8143
OpenShift: https://<master-node-ip>:8443
```

## Upgrade Procedure for RHOSP-based Contrail 3.2.x to Contrail 4.1

### IN THIS SECTION

- [Prerequisite | 255](#)
- [Upgrade the Undercloud | 255](#)
- [Update Red Hat Director Image Archives | 257](#)
- [Prepare Repositories on all Nodes | 259](#)
- [Upgrade the Operating System on Contrail Nodes | 260](#)
- [Prepare the Contrail Packages | 260](#)
- [Upgrade the Contrail Heat Templates | 260](#)
- [Modify the Yum Update Script for TripleO Puppet | 261](#)
- [Update the Overcloud Deployment Plan | 261](#)
- [Upgrade Cautions | 262](#)
- [Upgrade the Overcloud | 265](#)
- [Contrail Service Recovery After Upgrade | 266](#)

This section presents the steps to upgrade an OSP-based Contrail deployment from Contrail version 3.2.x to Contrail version 4.1.

## Prerequisite

Ensure that you have a cloud up and running with RHOSP10 Z4 and Contrail 3.2 before you proceed with the upgrade procedure.

This procedure has been validated with the following Contrail, Red Hat, and OpenStack versions.

| Contrail Version | Red Hat Version | OpenStack Version  |
|------------------|-----------------|--|
| 3.2.3            | RHEL 7.3        | RHOSP10 (packages dated Apr. 15, 2017)   |
| 3.2.6            | RHEL 7.4        | RHOSP10 (packages dated Feb. 2, 2018)  |
| 4.1              | RHEL 7.4        | RHOSP10 (packages dated Feb. 27, 2018)   |
| 4.1.1            | RHEL 7.5        | RHOSP10 (packages dated Jun. 4, 2018)<br>RHOSP11 (packages dated Jun. 4, 2018) |
| 4.1.2            | RHEL 7.5        | RHOSP10 (packages dated Oct 29, 2018)  |

**NOTE:** For Contrail Release 4.1.1, you must ensure that the OpenJDK version is java-1.8.0-openjdk-1.8.0.151-5.b12.el7\_4.x86\_64. This is because of a compatibility issue that the Cassandra 3.0 package has with the latest version of Open JDK provided in RHEL 7.5. You must add the correct version of OpenJDK to the Contrail repository and remove the older version of the package from the contrail controllers and analytics database.

To upgrade to OpenStack Platform release 10, you must first apply the patch for updating puppet-tripleo. This is because of a bug in the OpenStack Platform release OSP10 z8. For more information, see [https://bugzilla.redhat.com/show\\_bug.cgi?id=1579184](https://bugzilla.redhat.com/show_bug.cgi?id=1579184). This bug has been resolved in build puppet-tripleo-5.6.8-7.el7ost; see <https://access.redhat.com/errata/RHBA-2018:2101>. For more information about this bug, see <https://bugs.launchpad.net/tripleo/+bug/1771324>.

## Upgrade the Undercloud

Upgrade the undercloud to the most current RHOSP10 version.

1. Log in to the undercloud as the stack user.

```
su - stack
```

2. Update the Contrail repositories.

```
sudo rm -rf /etc/yum.repos.d/*contrail*
```

```
curl http://newrepo.contrail41-dev.repo -o /etc/yum.repos.d/localrepo.repo
```

3. Stop the main OpenStack platform services.

```
sudo systemctl stop 'openstack-*' 'neutron-*' httpd
```

4. Update the python-tripleoclient package and its dependencies to ensure you have the most current scripts for the minor version update.

```
$ sudo yum update python-tripleoclient
```

5. Upgrade the undercloud.

```
$ openstack undercloud upgrade
```

6. Reboot the node.

```
$ sudo reboot
```

7. Wait until the node reboots, then check the status of all services.

**NOTE:** It can take as much as 10 minutes or more for the `openstack-nova-compute` to become active after a reboot.

```
$ sudo systemctl list-units "openstack*" "neutron*" "openvswitch"
```

8. Verify the version of RHEL after the undercloud upgrade.

```
[root@undercloud ~]# cat /etc/redhat-release
Red Hat Enterprise Linux Server release 7.4 (Maipo)
[root@undercloud ~]#
```

9. Check the versions of the RHOSP images to verify that the new images are available.

In the following example, the new images are **rhosp-director-images-ipa-10.0-20180103.3.el7ost.noarch** and **rhosp-director-images-10.0-20180103.3.el7ost.noarch**.

```
[root@undercloud ~]# rpm -qa | grep -i rhosp
rhosp-director-images-ipa-10.0-20180103.3.el7ost.noarch
```



```

rhaps-director-images-10.0-20180103.3.el7ost.noarch
rhaps-director-images-ipa-10.0-20170615.1.el7ost.noarch
rhaps-director-images-10.0-20170615.1.el7ost.noarch
[root@undercloud ~]#

```

**10.** Verify the existence of the overcloud and its nodes.

```
$ openstack stack list
```

```
$ ironic node-list
```

**11.** Review the power status, provision state, and maintenance, ensuring:

- Power state is set to Power on
- Provision state is set to Active
- Maintenance is set to False

**12.** Verify that all OpenStack servers are Active.

```
$ openstack server list
```

**Figure 33: Server List**

| ID                                   | Name                                  | Status | Networks            | Image Name     |
|--------------------------------------|---------------------------------------|--------|---------------------|----------------|
| e37480b0-e098-4216-aea4-04b028aa5fb9 | overcloud-contrailanalytics-2         | ACTIVE | ctiplane=192.0.2.11 | overcloud-full |
| c77dcc49-c039-4855-bc91-e217bd34a51e | overcloud-contrailanalytics-1         | ACTIVE | ctiplane=192.0.2.12 | overcloud-full |
| 63d147ec-61e0-4d49-b1cb-6b68b7e5f48f | overcloud-contrailanalytics-0         | ACTIVE | ctiplane=192.0.2.21 | overcloud-full |
| 913d0bcb-9e9b-4987-a858-0389345e7025 | overcloud-controller-0                | ACTIVE | ctiplane=192.0.2.14 | overcloud-full |
| e66e86b1-df4a-4246-8e24-41461d617c18 | overcloud-controller-2                | ACTIVE | ctiplane=192.0.2.15 | overcloud-full |
| 9bbc1552-9ac1-4f27-846c-7756b14b84de | overcloud-controller-1                | ACTIVE | ctiplane=192.0.2.22 | overcloud-full |
| 8f8f5b9b-d5e2-4349-8d46-a488f2b7ab56 | overcloud-contrailanalyticsdatabase-0 | ACTIVE | ctiplane=192.0.2.17 | overcloud-full |
| 505c00f5-6715-48a7-a9ea-61ff157aa28a | overcloud-contrailanalyticsdatabase-2 | ACTIVE | ctiplane=192.0.2.24 | overcloud-full |
| ba8abe03-071b-4326-8fc0-715e93db3e4d | overcloud-contrailanalyticsdatabase-1 | ACTIVE | ctiplane=192.0.2.19 | overcloud-full |
| a7763383-92a0-4304-afc3-8aebf0ed2a05 | overcloud-contrailcontroller-2        | ACTIVE | ctiplane=192.0.2.20 | overcloud-full |
| e381db99-add0-4887-ac96-1e40d4108fc8 | overcloud-contrailcontroller-0        | ACTIVE | ctiplane=192.0.2.18 | overcloud-full |
| c808dcf1-4906-491e-8780-1f099c6a776c | overcloud-novacompute-0               | ACTIVE | ctiplane=192.0.2.16 | overcloud-full |
| 7eabc822-16d6-4fc6-afd7-7ba662ce7e92 | overcloud-contrailcontroller-1        | ACTIVE | ctiplane=192.0.2.23 | overcloud-full |

## Update Red Hat Director Image Archives

The undercloud update process might download new image archives from the rhosp-director images and the rhosp-director-ipa packages. You'll want to update your existing system with any new image archives.

1. Check the yum log to determine if new image archives are available.

```
$ sudo grep "rhosp-director-images" /var/log/yum.log
```

```
[stack@undercloud ~]$ sudo grep "rhosp-director-images" /var/log/yum.log
Feb 05 16:03:59 Installed: rhosp-director-images-ipa-10.0-20180103.3.el7ost.noarch
Feb 05 16:04:54 Installed: rhosp-director-images-10.0-20180103.3.el7ost.noarch
[stack@undercloud ~]$
```

2. If new image archives are available, replace your current images with the new images. Before deploying any new images, remove any existing images from the images undercloud on the stack user's home (/home/stack/images).

```
$ rm -rf ~/images/*
```

3. Extract the new image archives.

```
mkdir images
cd images
for i in /usr/share/rhosp-director-images/overcloud-full-latest-10.0.tar /usr/share/rhosp-
director-images/ironic-python-agent-latest-10.0.tar; do tar -xvf $i; done
```

4. Install the Contrail package on the overcloud image by using the virt-customize command.

```
export LIBGUESTFS_BACKEND=direct /usr/bin/virt-customize -a /home/stack/images/overcloud-
full.qcow2 \
--copy-in /etc/yum.repos.d/mylocalrepo.repo:/etc/yum.repos.d \
--run-command 'yum -y install puppet-tripleo contrail-tripleo-puppet puppet-contrail'\
--run-command ' cp -r /usr/share/contrail-tripleo-puppet/ /usr/share/openstack-puppet/
modules/tripleo/ ' \
--run-command 'rm -fr /var/cache/yum/*' \
--run-command 'yum clean all' \ --selinux-relabel
```

5. Import the new image archives into the undercloud and configure nodes to use the new images.

```
$ openstack overcloud image upload --update-existing --image-path /home/stack/images/ $ openstack baremetal
configure boot
```

6. Verify that the images are uploaded.

```
$ glance image-list
```

| ID                                   | Name                   | Disk Format | Container Format | Size       | Status |
|--------------------------------------|------------------------|-------------|------------------|------------|--------|
| 8a91bf65-c86e-4749-9377-549708d381cc | bm-deploy-kernel       | aki         | aki              | 5889712    | active |
| d8992dc4-6716-482e-87bf-b618af7d7e6b | bm-deploy-ramdisk      | ari         | ari              | 380417527  | active |
| d6c52d67-3667-4542-ab60-a6a7e20e7f9d | overcloud-full         | qcow2       | bare             | 1398873088 | active |
| 4a8e2e28-a29d-4d21-9416-8933144f7e31 | overcloud-full-initrd  | ari         | ari              | 47844596   | active |
| 16fea5e5-d980-4f6f-8015-6d267c772e07 | overcloud-full-vmlinux | aki         | aki              | 5889712    | active |

7. Show the details of the new image that has been created. The new image will be used to add a new node in the overcloud.

```
$ glance image-show overcloud-full
```

| Property              | Value                                |
|-----------------------|--------------------------------------|
| Property 'kernel_id'  | 16fea5e5-d980-4f6f-8015-6d267c772e07 |
| Property 'ramdisk_id' | 4a8e2e28-a29d-4d21-9416-8933144f7e31 |
| checksum              | c7cff91234f3c323fb38236f4bc1bbd3     |
| container_format      | bare                                 |
| created_at            | 2018-02-05T21:05:52.000000           |
| deleted               | False                                |
| disk_format           | qcow2                                |
| id                    | d6c52d67-3667-4542-ab60-a6a7e20e7f9d |
| is_public             | True                                 |
| min_disk              | 0                                    |
| min_ram               | 0                                    |
| name                  | overcloud-full                       |
| owner                 | 370bedc65518486c9aa3f9875ef3934c     |
| protected             | False                                |
| size                  | 1398873088                           |
| status                | active                               |
| updated_at            | 2018-02-05T21:06:05.000000           |

8. Observe the contrail-status on all Contrail nodes. All services in the Contrail nodes, except the controller (OpenStack), should be up and running before proceeding with the upgrade.

```
[stack@undercloud ~]$ for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '=' -f2); do ssh heat-admin@$i sudo contrail-status; done
```

## Prepare Repositories on all Nodes

1. Delete existing repositories on all overcloud nodes. Be sure to verify each deletion.

```
for ipnode in $(nova list | sed '4,$ !d;$d' | awk -F 'ctlplane=' '{print $2}' | tr -d '|'); do echo cleaning yum repolist on $ipnode && ssh heat-admin@$ipnode 'find /etc/yum.repos.d/ ! -name 'contrail-install.repo' -type f -exec sudo rm -f {} +' ; done
```

2. Add new repositories on all overcloud nodes. Be sure to verify each addition.

```
for ipnode in $(nova list | sed '4,$ !d;$d' | awk -F 'ctlplane=' '{print $2}' | tr -d '|'); do echo cleaning
yum repolist on $ipnode && ssh heat-admin@$ipnode ' curl http://newrepo.contrail41-dev.repo -o /etc/
yum.repos.d/localrepo.rep' ; done
```

## Upgrade the Operating System on Contrail Nodes

1. Define a list (\$iplist) that contains all Contrail nodes.

```
Iplist=" @IPcontrailController1 @IPcontrailController2 ..."
```

2. Upgrade the operating system for all nodes in the iplist.

```
for ipnode in $iplist; do echo -e "\n\n\t*****upgrade node : $ipnode *****" && ssh heat-admin@$ipnode 'sudo
yum update -y --disablerepo=*contrail* --skip-broken && exit' ; done
```

## Prepare the Contrail Packages

Check the undercloud Contrail packages versions for `contrail-tripleo-puppet`, `puppet-contrail`, and `contrail-tripleo-puppet`. The newest versions of those packages must be installed before proceeding with the overcloud upgrade. See the following example, with current packages versions.

```
[root@director-ctl ~]# rpm -qa | grep contrail
contrail-tripleo-puppet-4.1.0.0-8.el7.noarch
contrail-tripleo-heat-templates-4.1.0.0-8.el7.noarch
puppet-contrail-4.1.0.0-8.el7.noarch
```

## Upgrade the Contrail Heat Templates

You must copy the new Contrail Heat templates package to the undercloud node, while retaining a copy of the Heat templates that were used for the existing deployment.

1. Make a copy of all of the Heat templates that were used for deployment and save the copies, because the existing files will be overwritten by the new versions. The templates to copy are of the form `contrail-services.yaml`, `contrail-net.yaml`, and so on.
2. Copy the new `contrail-tripleo-heat` templates to the undercloud node.

```
cp -r /usr/share/contrail-tripleo-heat-templates/environments/contrail /home/stack/templates/
openstack-tripleo-heat-templates/environments/
```

```
cp -r /usr/share/contrail-tripleo-heat-templates/puppet/services/network/* /home/stack/
templates/openstack-tripleo-heat-templates/puppet/services/network
```

## Modify the Yum Update Script for TripleO Puppet

Before starting the upgrade, a few Puppet commands must be added to the yum\_update script, located at:

**/home/stack/templates/openstack-tripleo-heat-templates/extraconfig/tasks/yum\_update.sh**

1. Update the following Puppet commands in the yum\_update.sh. Refer to the following patch for details regarding the exact placement of the commands patch: [https://github.com/Juniper/contrail-tripleo-heat-templates/blob/stable/newton/environments/contrail/yum\\_updates.patch](https://github.com/Juniper/contrail-tripleo-heat-templates/blob/stable/newton/environments/contrail/yum_updates.patch)

```
yum install -y contrail-tripleo-puppet puppet-contrail
```

```
rsync -a /usr/share/contrail-tripleo-puppet/ /usr/share/openstack-puppet/modules/tripleo/
```

2. Ensure that the new version of the packages puppet-contrail and contrail-tripleo-puppet have been installed on the Contrail nodes, and that contrail-tripleo-puppet has been copied to openstack-puppet.

```
rm -rf /usr/share/openstack-puppet/modules/tripleo/contrail-tripleo-puppet
```

```
cp -R /usr/share/contrail-tripleo-puppet/ /usr/share/openstack-puppet/modules/tripleo
```

3. Update the fields \*contrail version and \*contrail repo in contrail-services.yaml.

Filename: **\templates/environments/contrail/contrail-services.yaml.**

Add the following parameters:

ContrailVersion: 4

ContrailRepo : *<location of the contrail-41 repo>*

## Update the Overcloud Deployment Plan

1. Update the current plan by rerunning the command used for cloud deployment and adding the suffix - update-plan-only.

```
<openstack overcloud deploy> -update-plan-only
```

## Example

```
openstack overcloud deploy --update-plan-only --templates /home/stack/tripleo-heat-templates/
\
--roles-file /home/stack/tripleo-heat-templates/environments/contrail/roles_data.yaml \
-e /home/stack/tripleo-heat-templates/environments/puppet-pacemaker.yaml \
-e /home/stack/tripleo-heat-templates/environments/contrail/contrail-services.yaml \
-e /home/stack/tripleo-heat-templates/environments/contrail/network-isolation.yaml \
-e /home/stack/tripleo-heat-templates/environments/contrail/contrail-net.yaml \
-e /home/stack/tripleo-heat-templates/environments/ips-from-pool-all.yaml \
-e /home/stack/tripleo-heat-templates/environments/network-management.yaml \
-e /home/stack/tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/environment-
rhel-registration.yaml \
-e /home/stack/tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/rhel-
registration-resource-registry.yaml \
--libvirt-type qemu
```

2. Make a copy of the existing deploy script to the update-stack.sh. The update-stack.sh is the script used to update the overcloud plan, and it references the same templates that were used to deploy the stack. All files used for the overcloud update should be identical to the files used for deployment, with the exception of the contrail-services file that was updated with the latest contrail-version and contrail-repo.

```
cp deploy.sh update-stack.sh
```

3. Update the deployment plan.

```
./update-stack.sh
```

## Upgrade Cautions



**CAUTION:** The steps to perform the overcloud upgrade are service disrupting, and should only be performed within a maintenance window.

## Potential Packages Failures

Read the following before proceeding with the overcloud upgrade. The upgrade may fail due to packages conflicts in Contrail nodes for analytics, analytics database, and Contrail controllers. Some observed failures due to packages conflicts are detailed in this section.

## OpenStack Undercloud Upgrade Failure

**From Version:** RHEL-7.3 with rhosp-director-images-10.0-20170228.1.el7ost.noarch  
**To Version:** RHEL-7.5 with rhosp-director-images-10.0-20180628.2.el7ost.noarch

OpenStack undercloud upgrade may fail with the following error message:

```
INFO: 2018-06-13 20:17:03 - Could not retrieve fact='current_nova_host',
resolution='<anonymous>': uninitialized constant Tempfile 2018-06-13 20:17:04,404
INFO: 2018-06-13 20:17:04 - ^[[1;31mError: Puppet::Parser::AST::Resource failed with error
ArgumentError: Invalid resource type sysctl::value at /etc/puppet/manifests/puppet-stack-
config.pp:24 on node undercloud.example.com^[[0m
```

For more information about this error, see <https://bugs.launchpad.net/juniperopenstack/+bug/1792036>.

## Contrail-api Service Not Restarted During Upgrade

Contrail-api remains in the initializing state during upgrade. Modify the puppet file as described in <https://github.com/Juniper/puppet-contrail/commit/b5ec14743d6c13cf847f277f1d28a404931c7b0e>.

**NOTE:** This issue has been resolved in Contrail Release 4.1.2.

## Contrail-api Service Down After Upgrade

You can resolve this issue by removing the `admin_token` entry from the `/etc/contrail/contrail-keystone-auth.conf` file. For more information, see <https://bugs.launchpad.net/juniperopenstack/+bug/1791861>.

**NOTE:** This issue has been resolved in Contrail Release 4.1.2.

## Overcloud Update Failure

Upgrade from OpenStack Platform release 10 GA image to the latest version hangs because of a systemd bug. For more information, see [https://bugzilla.redhat.com/show\\_bug.cgi?id=1582338](https://bugzilla.redhat.com/show_bug.cgi?id=1582338)

**Solution:**

**To upgrade from Contrail Release 3.2.x to 4.1.1:**

1. Run the following command before you start the update:

```
source ~/stackrc
for address in $(openstack server list -f json | jq -r -c '[] | .Networks' | grep -oP
'[0-9.]+'); do \
  ssh -q -o StrictHostKeyChecking=no heat-admin@$address \
  'sudo yum install -y yum-plugin-versionlock; \
  sudo yum versionlock add systemd systemd-libs libgudev1 systemd-sysv rsyslog;'
done
```

2. Run the overcloud update.

3. Run the following command:

```
source ~/stackrc
for address in $(openstack server list -f json | jq -r -c '[] | .Networks' | grep -oP
'[0-9.]+'); do \
  ssh -q -o StrictHostKeyChecking=no heat-admin@$address \
  'sudo yum versionlock del systemd systemd-libs libgudev1 systemd-sysv rsyslog;
  sudo yum versionlock add java-1.8.0-openjdk-headless-1.8.0.151 java-1.8.0-
openjdk-1.8.0.151;
  sudo yum update -y'
done
```

4. Restart the node.

## Analytics Database Failure

You may encounter a Cassandra package version conflict issue.

Error message: `cassandra22` conflicts with `cassandra-3.10-0contrail0.el7.centos.noarch`

Solution: Remove the `cassandra22` package.

```
sudo rpm -e --nodeps cassandra22-2.2.8-1.noarch
```

## Analytics Node Failure

Error message: file `/usr/lib/python2.7/site-packages/redis/__init__.py` from install of `python-redis-2.10.3-3.el7ost.noarch` conflicts with file from package `redis-py-0.1-2contrail.el7.noarch`

Solution: Remove the `redis-py` package.



```
sudo rpm -e python-redis-2.10.3-3.el7ost.noarch
```

### Contrail Controller Failure

Error message: `cassandra22` conflicts with `cassandra-3.10-0contrail0.el7.centos.noarch`

Solution: Remove the `cassandra22` package.

```
sudo rpm -e --nodeps cassandra22-2.2.8-1.noarch
```

**NOTE:** You can delete `redis-py` from the analytics node, and delete `cassandra22` from the analytics database and the Contrail controllers node.

### Contrail vRouter Package Conflict

Error message: `/etc/init.d/contrail-vrouter-nodemgr` conflicts between attempted installs of `contrail-openstack-vrouter-4.1.0-8.el7.noarch` and `contrail-vrouter-agent-4.1.0-8.el7.x86_64`

Solution: Remove the `contrail-openstack-vrouter`.

```
yum remove contrail-openstack-vrouter
```

### Analytics Node snmp-lib Version Conflict

Error message: duplicate versions of `net-snmp-libs` -> `net-snmp-libs-5.7.2-28.el7_4.1.x86_64` is a duplicate with `1:net-snmp-libs-5.7.2-28.el7.i686`

Solution:

```
rpm -e --nodeps net-snmp-libs
```

### Upgrade the Overcloud



**CAUTION:** The steps in this section are service disrupting, and should only be performed within a maintenance window.

#### 1. Update the overcloud stack.

```
$ openstack overcloud update stack -i overcloud
2018-02-09 16:24:46Z [overcloud]: UPDATE_COMPLETE Stack UPDATE completed successfully
Stack overcloud UPDATE_COMPLETE
```

```
Overcloud Endpoint: http://19x.xxx.xxx.xx:5000/v2.0
Overcloud Deployed
```

2. Verify the overcloud stack status, the contrail-status, and the contrail-version after the upgrade.

### Overcloud Stack Status

```
[stack@undercloud ~]$ openstack stack list

+-----+-----+-----+-----+
+-----+
| ID                | Stack Name | Stack Status   | Creation Time
| Updated Time      |            |                |
+-----+-----+-----+-----+
+-----+
| e56c5512-3e32-4940-a05a-e194f48ce67a | overcloud  | UPDATE_COMPLETE | 2018-02-16T20:32:17Z
| 2018-02-21T01:12:08Z |
+-----+-----+-----+-----+
+-----+
[stack@undercloud ~]$
```

### Contrail Stack Status

```
for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '=' -f2); do ssh heat-admin@
$i sudo contrail-status; done
```

### Contrail Version Check

```
for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '=' -f2); do ssh heat-admin@
$i sudo contrail-version; done
```

## Contrail Service Recovery After Upgrade

Upon completing the upgrade, you might have some Contrail services reporting status down on some nodes. this section provides recovery suggestions for different services.

### Contrail Controller Services

In a recent upgrade, the following services were down in the Contrail controller nodes.

- `contrail-control` – failed
- `contrail-dns` – failed

To determine the root cause of these failures, check the Contrail logs in `/var/log/contrail/`.

In this example, it was the IFMap and Discovery services causing the failures, because those services are no longer used in the Contrail 4.x releases.

## Solution

1. Remove the IFMap and Discover configuration sections from:
  - a. `/etc/contrail/contrail-control.conf`
  - b. `/etc/contrail/contrail-dns.conf`
2. Restart the services `contrail-controller` and `contrail-dns`.
  - a. `service contrail-control restart`
  - b. `service contrail-dns restart`
3. Verify that the services `contrail-controller` and `contrail-dns` are up and running after the restart.
  - a. `contrail-status`

## Contrail Compute Services

On the Contrail compute node, the `contrail-vrouter-agent` service might be down after the upgrade. The vrouter agent is dependent on the kernel module. The RHEL7.3 has been upgraded to RHEL7.4, and the node needs to be rebooted to ensure the correct kernel module gets loaded.

1. Reboot the node.
2. After reboot, restart the vrouter service..  
`service supervisor-vrouter restart`
3. Verify that all Contrail services are up and running.  
`contrail-status`

## RELATED DOCUMENTATION

[Upgrade Procedure for RHOSP-based Contrail 4.1.1 to Contrail 4.1.2](#) | 268

---

[Upgrade Procedure for RHOSP-based Contrail 4.1.2 to Contrail 4.1.3 | 279](#)

*Upgrade Procedure for RHOSP-based Contrail 4.1.2 to Contrail 4.1.3*

## Upgrade Procedure for RHOSP-based Contrail 4.1.1 to Contrail 4.1.2

### IN THIS SECTION

- [Prerequisite | 268](#)
- [Upgrade the Undercloud | 269](#)
- [Update Red Hat Director Image Archives | 271](#)
- [Prepare Repositories on all Nodes | 273](#)
- [Upgrade the Operating System on Contrail Nodes | 273](#)
- [Prepare the Contrail Packages | 274](#)
- [Upgrade the Contrail Heat Templates | 274](#)
- [Modify the Yum Update Script for TripleO Puppet | 275](#)
- [Update the Overcloud Deployment Plan | 275](#)
- [Upgrade Cautions | 277](#)
- [Upgrade the Overcloud | 277](#)

This section presents the steps to upgrade an OSP-based Contrail deployment from Contrail version 4.1.1 to Contrail version 4.1.2.

### Prerequisite

Ensure you have a cloud up and running with RHOSP10 Z4 and Contrail 4.1.1 before you proceed with the upgrade procedure.

This procedure has been validated with the following Contrail, Red Hat, and OpenStack versions.

| Contrail Version | Red Hat Version | OpenStack Version                      |
|------------------|-----------------|--|
| 3.2.3            | RHEL 7.3        | RHOSP10 (packages dated Apr. 15, 2017) |

*(Continued)*

| Contrail Version | Red Hat Version | OpenStack Version  |
|------------------|-----------------|--|
| 3.2.6            | RHEL 7.4        | RHOSP10 (packages dated Feb. 2, 2018)  |
| 4.1              | RHEL 7.4        | RHOSP10 (packages dated Feb. 27, 2018)   |
| 4.1.1            | RHEL 7.5        | RHOSP10 (packages dated Jun. 4, 2018)<br>RHOSP11 (packages dated Jun. 4, 2018) |
| 4.1.2            | RHEL 7.5        | RHOSP10 (packages dated Oct 29, 2018)  |
| 4.1.3            | RHEL 7.5        | RHOSP10 (packages dated Oct 29, 2018)  |



**CAUTION:** Set the Red Hat Satellite filter end date to October 29, 2018 before proceeding with the upgrade.

## Upgrade the Undercloud

Upgrade the undercloud to the most current RHOSP10 version.

1. Log in to the undercloud as the stack user.

```
$ su - stack
```

2. Update the Contrail repositories.

```
$ sudo rm -rf /etc/yum.repos.d/*contrail*
```

```
$ curl http://newrepo.contrail41-dev.repo -o /etc/yum.repos.d/localrepo.repo
```

3. Stop the main OpenStack platform services.

```
$ sudo systemctl stop 'openstack-*' 'neutron-*' httpd
```

4. Update the python-tripleoclient package and its dependencies to ensure you have the most current scripts for the minor version update.

```
$ sudo yum update python-tripleoclient
```

5. Upgrade the undercloud.

```
$ openstack undercloud upgrade
```

6. Reboot the node.

```
$ sudo reboot
```

7. Wait until the node reboots, then check the status of all services.

**NOTE:** It can take as much as 10 minutes or more for the `openstack-nova-compute` to become active after a reboot.

```
$ sudo systemctl list-units "openstack*" "neutron*" "openvswitch"
```

8. Verify the version of RHEL after the undercloud upgrade.

**NOTE:** Contrail does not support undercloud Red Hat version running with RHEL-7.6 as part of Contrail 4.1.2 release.

```
[root@undercloud ~]# cat /etc/redhat-release
Red Hat Enterprise Linux Server release 7.5 (Maipo)
[root@undercloud ~]#
```

9. Verify the existence of the overcloud and its nodes.

```
$ openstack stack list
```

```
$ ironic node-list
```

10. Verify that all OpenStack servers are Active.

```
$ openstack server list
```

Figure 34: Server List

| ID                                   | Name                                  | Status | Networks            | Image Name     |
|--------------------------------------|---------------------------------------|--------|---------------------|----------------|
| e37480b0-e098-4216-aea4-04b028aa5fb9 | overcloud-contrailanalytics-2         | ACTIVE | ctiplane=192.0.2.11 | overcloud-full |
| c77dcc49-c039-4855-bc91-e217bd34a51e | overcloud-contrailanalytics-1         | ACTIVE | ctiplane=192.0.2.12 | overcloud-full |
| 63d147ec-61e0-4d49-b1cb-6b68b7e5f48f | overcloud-contrailanalytics-0         | ACTIVE | ctiplane=192.0.2.21 | overcloud-full |
| 913d0bcb-9e9b-4987-a858-0389345e7025 | overcloud-controller-0                | ACTIVE | ctiplane=192.0.2.14 | overcloud-full |
| e66e86b1-df4a-4246-8e24-41461d617c18 | overcloud-controller-2                | ACTIVE | ctiplane=192.0.2.15 | overcloud-full |
| 9bbc1552-9ac1-4f27-846c-7756b14b84de | overcloud-controller-1                | ACTIVE | ctiplane=192.0.2.22 | overcloud-full |
| 8f8f5b9b-d5e2-4349-8d46-a488f2b7ab56 | overcloud-contrailanalyticsdatabase-0 | ACTIVE | ctiplane=192.0.2.17 | overcloud-full |
| 505c00f5-6715-48a7-a9ea-61ff157aa28a | overcloud-contrailanalyticsdatabase-2 | ACTIVE | ctiplane=192.0.2.24 | overcloud-full |
| ba8abe03-071b-4326-8fc0-715e93db3e4d | overcloud-contrailanalyticsdatabase-1 | ACTIVE | ctiplane=192.0.2.19 | overcloud-full |
| a7763383-92a0-4304-afc3-8aebf0ed2a05 | overcloud-contrailcontroller-2        | ACTIVE | ctiplane=192.0.2.20 | overcloud-full |
| e381db99-add0-4887-ac96-1e40d4108fc8 | overcloud-contrailcontroller-0        | ACTIVE | ctiplane=192.0.2.18 | overcloud-full |
| c808dcf1-4906-491e-8780-1f099c6a776c | overcloud-novacompute-0               | ACTIVE | ctiplane=192.0.2.16 | overcloud-full |
| 7eabc822-16d6-4fc6-afd7-7ba662ce7e92 | overcloud-contrailcontroller-1        | ACTIVE | ctiplane=192.0.2.23 | overcloud-full |

## Update Red Hat Director Image Archives

The undercloud update process might download new image archives from the rhosp-director images and the rhosp-director-ipa packages. You will have to update your existing system with any new image archives.

1. Check the yum log to determine if new image archives are available.

```
$ sudo grep "rhosp-director-images" /var/log/yum.log
```

```
[stack@undercloud]$ sudo grep "rhosp-director-images" /var/log/yum.log
```

```
Oct 26 15:09:20 Installed: rhosp-director-images-ipa-10.0-20180821.1.el7ost.noarch
```

```
Oct 26 15:10:10 Installed: rhosp-director-images-10.0-20180821.1.el7ost.noarch
```

2. If new image archives are available, replace your current images with the new images. Before deploying any new images, remove any existing images from the images undercloud on the stack user's home (/home/stack/images).

```
$ rm -rf ~/images/*
```

3. Extract the new image archives.

```
mkdir images
cd images
for i in /usr/share/rhosp-director-images/overcloud-full-latest-10.0.tar /usr/share/rhosp-director-images/ironic-python-agent-latest-10.0.tar; do tar -xvf $i; done
```

4. Install the Contrail package on the overcloud image by using the virt-customize command.

```
export LIBGUESTFS_BACKEND=direct /usr/bin/virt-customize -a /home/stack/images/overcloud-
full.qcow2 \
--copy-in /etc/yum.repos.d/mylocalrepo.repo:/etc/yum.repos.d \
--run-command 'yum -y install puppet-tripleo contrail-tripleo-puppet puppet-contrail'\
--run-command ' cp -r /usr/share/contrail-tripleo-puppet/ /usr/share/openstack-puppet/modules/
tripleo/ ' \
--run-command 'rm -fr /var/cache/yum/*' \
--run-command 'yum clean all' \ --selinux-relabel
```

5. Import the new image archives into the undercloud and configure nodes to use the new images.

```
$ openstack overcloud image upload --update-existing --image-path /home/stack/images/
```

6. Verify that the images are uploaded.

```
$ openstack image list
```

| ID                                   | Name                   | Status |
|--------------------------------------|------------------------|--------|
| a518a08c-3c49-47b7-9705-a7da5c90dcc6 | bm-deploy-ramdisk      | active |
| b7d75ff7-926b-426c-aa2e-424f5d9f8328 | bm-deploy-kernel       | active |
| cd7eedc0-2ed2-4a21-8359-9d8ee89374ac | overcloud-full         | active |
| 9c9e46f4-d571-49fb-a485-4653b6a52b44 | overcloud-full-initrd  | active |
| 740b6cb9-5757-475b-896b-49c526807671 | overcloud-full-vmlinuz | active |

7. Show the details of the new image that has been created. The new image will be used to add a new node in the overcloud.

```
$ openstack image show overcloud-full
```

| Field            | Value   |
|------------------|---|
| checksum         | 13e67f5039dc7e69b5bbc494d8838b8d  |
| container_format | bare  |
| created_at       | 2018-10-26T20:02:18.000000  |
| deleted          | False   |
| deleted_at       | None  |
| disk_format      | qcow2   |
| id               | cd7eedc0-2ed2-4a21-8359-9d8ee89374ac  |
| is_public        | True  |
| min_disk         | 0   |
| min_ram          | 0   |
| name             | overcloud-full  |
| owner            | 0cd0e938b0fe46ef9c431715395f9469  |
| properties       | kernel_id='740b6cb9-5757-475b-896b-49c526807671', ramdisk_id='9c9e46f4-d571-49fb-a485-4653b6a52b44' |
| protected        | False   |
| size             | 1469448192  |
| status           | active  |
| updated_at       | 2018-10-26T20:02:52.000000  |
| virtual_size     | None  |



8. Verify `contrail-status` on all Contrail nodes. All services in the Contrail nodes, except the controller (OpenStack), should be up and running before proceeding with the upgrade.

```
[stack@undercloud ~]$ for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '=' -f2); do ssh heat-admin@$i sudo contrail-status; done
```

## Prepare Repositories on all Nodes

1. Delete existing repositories on all overcloud nodes. Verify each deletion.

```
sudo for ipnode in $(nova list | sed '4,$ !d;$d' | awk -F 'ctlplane=' '{print $2}' | tr -d '|'); do echo cleaning yum repolist on $ipnode && ssh heat-admin@$ipnode 'find /etc/yum.repos.d/ ! -name 'contrail-install.repo' -type f -exec sudo rm -f {} +' ; done
```

2. Add new repositories on all overcloud nodes. Verify each addition.

```
sudo for ipnode in $(nova list | sed '4,$ !d;$d' | awk -F 'ctlplane=' '{print $2}' | tr -d '|'); do echo cleaning yum repolist on $ipnode && ssh heat-admin@$ipnode 'curl http://newrepo.contrail41-dev.repo -o /etc/yum.repos.d/localrepo.rep' ; done
```

## Upgrade the Operating System on Contrail Nodes

1. Define a list (`$iplist`) that contains all Contrail nodes. Run the following command on undercloud VM as stack user.

```
Iplist=" @IPcontrailController1 @IPcontrailController2 ..."
```



**CAUTION:** Attach the new satellite subscription key on all overcloud nodes before upgrading the overcloud packages. Satellite must be synced with filter end date *29th Oct 2018*. Make sure to clear cache by typing **sudo yum clean all**.

2. Upgrade the operating system for all nodes in the `iplist`. Run the following command on undercloud VM as stack user

```
sudo for ipnode in $iplist; do echo -e "\n\n\t*****upgrade node : $ipnode *****" && ssh heat-admin@$ipnode 'sudo yum update -y --disablerepo=*contrail* --skip-broken && exit' ; done
```

3. Reboot overcloud contrail compute nodes, if there is any change in the kernel version. This needs to be done before installing contrail packages on compute VM.

Supported kernel versions: 3.10.0-862.11.6.el7.x86\_64 and 3.10.0-957.el7.x86\_64

```
----- [root@overcloud-novacompute-2 ~]# modinfo vrouter filename: /lib/modules/
```

```
3.10.0-862.11.6.el7.x86_64/extra/net/vrouter/vrouter.ko version: 4.1.2.0 license: GPL retpoline: Y
rhelversion: 7.5
```

## Prepare the Contrail Packages

Check the undercloud Contrail packages versions for `contrail-tripleo-puppet`, `puppet-contrail`, and `contrail-tripleo-heat-templates`. The newest versions of those packages must be installed before proceeding with the overcloud upgrade. See the following example, with current packages versions.

```
[stack@undercloud~]$ rpm -qa | grep contrail

puppet-contrail-4.1.2.0-NN.el7.noarch
contrail-tripleo-heat-templates-4.1.2.0-NN.el7.noarch
contrail-tripleo-puppet-4.1.2.0-NN.el7.noarch
```

## Upgrade the Contrail Heat Templates

You must copy the new Contrail Heat templates package to the undercloud node, while retaining a copy of the Heat templates that were used for the existing deployment.

1. Make a copy of all of the Heat templates that were used for deployment and save the copies, because the existing files will be overwritten by the new versions. The templates to copy are of the form `contrail-services.yaml`, `contrail-net.yaml`, and so on.

**NOTE:** Red Hat does not support changing IP address of the existing cluster as a part of upgrade. Do not change IP address of the cluster while creating new `tripleo-heat-templates`

2. Copy the new `contrail-tripleo-heat` templates to the undercloud node.

```
sudo cp -r /usr/share/contrail-tripleo-heat-templates/environments/contrail /home/stack/
tripleo-heat-templates/environments/

sudo cp -r /usr/share/contrail-tripleo-heat-templates/puppet/services/network/* /home/stack/
tripleo-heat-templates/puppet/services/network
```

**NOTE:** The directory `/home/stack/tripleo-heat-templates` is user defined, it can be *User Defined-directory>/openstack-tripleo-heat-templates*

## Modify the Yum Update Script for TripleO Puppet

Following Puppet commands must be added to the `yum_update` script before starting the upgrade. The script is located at:

**`/home/stack/tripleo-heat-templates/extraconfig/tasks/yum_update.sh`**

1. Update the following Puppet commands in the `yum_update.sh` after the line `“echo -n “false” > $heat_outputs_path.update_managed_packages”`.

Refer to the following patch for details regarding the exact placement of the commands patch:  
[https://github.com/Juniper/contrail-tripleo-heat-templates/blob/stable/newton/environments/contrail/yum\\_updates.patch](https://github.com/Juniper/contrail-tripleo-heat-templates/blob/stable/newton/environments/contrail/yum_updates.patch)

```
yum install -y contrail-tripleo-puppet puppet-contrail
```

```
rsync -a /usr/share/contrail-tripleo-puppet/ /usr/share/openstack-puppet/modules/tripleo/
```

2. Update the fields `*contrail version` and `*contrail repo` in `contrail-services.yaml`.

Default parameter for `contrailVersion` is 4.

Filename: **`/home/stack/tripleo-heat-templates/environments/contrail/contrail-services.yaml`**.

Add the following parameters:

`ContrailVersion: 4`

`ContrailRepo : <location of the contrail-41 repo>`

**NOTE:** `/home/stack/tripleo-heat-templates` directory is user defined and it can be directory name under `stack` user.

## Update the Overcloud Deployment Plan

1. Update the current plan by re-running the command used for cloud deployment and adding the suffix `-update-plan-only`.

```
<openstack overcloud deploy> -update-plan-only
```

## Example

```

openstack overcloud deploy --update-plan-only --templates /home/stack/tripleo-heat-templates/
\
  --roles-file /home/stack/tripleo-heat-templates/environments/contrail/roles_data.yaml \
  -e /home/stack/tripleo-heat-templates/environments/puppet-pacemaker.yaml \
  -e /home/stack/tripleo-heat-templates/environments/contrail/contrail-services.yaml \
  -e /home/stack/tripleo-heat-templates/environments/contrail/network-isolation.yaml \
  -e /home/stack/tripleo-heat-templates/environments/contrail/contrail-net.yaml \
  -e /home/stack/tripleo-heat-templates/environments/ips-from-pool-all.yaml \
  -e /home/stack/tripleo-heat-templates/environments/network-management.yaml \
  -e /home/stack/tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/environment-
rhel-registration.yaml \
  -e /home/stack/tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/rhel-
registration-resource-registry.yaml \
  --libvirt-type qemu

```

2. Make a copy of the existing deploy script to the update-stack.sh. The update-stack.sh is the script used to update the overcloud plan, and it references the same templates that were used to deploy the stack. All files used for the overcloud update should be identical to the files used for deployment, except contrail-services file that was updated with the latest contrail-version and contrail-repo.

```
cp deploy.sh update-stack.sh
```

3. Update the deployment plan.

```
./update-stack.sh
```

## Example

```

[stack@undercloud-R4-1-1-upg-R4-1-2 ~]$ ./update-stack.sh
Removing the current plan files
Uploading new plan files
Started Mistral Workflow. Execution ID: 6c8fb5b7-6eda-4d92-8245-f7ac46bb369d
Plan updated
Deploying templates in the directory /tmp/tripleoclient-CdyN2I/tripleo-heat-templates
Overcloud Endpoint: http://10.87.67.232:5000/v2.0

```

```
Overcloud Deployed
[stack@undercloud-R4-1-1-upg-R4-1-2 ~]$
```

## Upgrade Cautions



**CAUTION:** The steps to perform the overcloud upgrade are service disrupting, and should only be performed within a maintenance window.

The upgrade procedure may fail due to packages conflicts in Contrail analytics nodes. Some observed failures due to packages conflicts are detailed in this section. Continue with the deployment after applying the recommended solution.

### Analytics Node snmp-lib Version Conflict

Error message: Protected multilib versions: 1:net-snmp-libs-5.7.2-37.el7.x86\_64 != 1:net-snmp-libs-5.7.2-33.el7\_5.2.i686

Solution:

```
rpm -e --nodeps net-snmp-libs
```

## Upgrade the Overcloud



**CAUTION:** The steps in this section are service disrupting, and should only be performed within a maintenance window.

### 1. Update the overcloud stack.

```
$ openstack overcloud update stack -i overcloud
on_breakpoint: [u'overcloud-contrailanalyticsdatabase-0']
Breakpoint reached, continue? Regexp or Enter=proceed (will clear
4386bdc7-5087-4a4d-865c-0b0181ce9345), no=cancel update, C-c=quit interactive mode:
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
```



```
+-----+
[stack@undercloud ~]$
```

### Contrail Stack Status

```
sudo for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '=' -f2); do ssh heat-admin@$i sudo contrail-status; done
```

### Contrail Version Check

```
for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '=' -f2); do ssh heat-admin@$i sudo contrail-version; done
```

## RELATED DOCUMENTATION

[Upgrade Procedure for RHOSP-based Contrail 3.2.x to Contrail 4.1](#) | 254

*Upgrade Procedure for RHOSP-based Contrail 4.1.2 to Contrail 4.1.3*

## Upgrade Procedure for RHOSP-based Contrail 4.1.2 to Contrail 4.1.3

### IN THIS SECTION

- [Prerequisite](#) | 280
- [Upgrade the Undercloud](#) | 281
- [Update Red Hat Director Image Archives](#) | 282
- [Prepare Repositories on all Nodes](#) | 284
- [Upgrade the Operating System on Contrail Nodes](#) | 285
- [Prepare the Contrail Packages](#) | 285
- [Upgrade the Contrail Heat Templates](#) | 287
- [Modify the Yum Update Script for TripleO Puppet](#) | 288
- [Update the Overcloud Deployment Plan](#) | 288
- [Upgrade the Overcloud](#) | 290

This section presents the steps to upgrade an OSP-based Contrail deployment from Contrail version 4.1.2 to Contrail version 4.1.3.

## Prerequisite

Before upgrading to Contrail Release 4.1.3, you must update the `net-snmp` package to the `net-snmp #37` version. The following `net-snmp` packages must be available in the upgrade repository and are installed automatically on Contrail Analytics nodes during the upgrade process:

- `net-snmp-5.7.2-37.el7.x86_64.rpm`
- `net-snmp-agent-libs-5.7.2-37.el7.x86_64.rpm`
- `net-snmp-libs-5.7.2-37.el7.x86_64.rpm`
- `net-snmp-utils-5.7.2-37.el7.x86_64.rpm`

Ensure you have a cloud up and running with RHOSP10 and Contrail 4.1.2 before you proceed with the upgrade procedure.

This procedure has been validated with the following Contrail, Red Hat, and OpenStack versions.

| Contrail Version | Red Hat Version | OpenStack Version  |
|------------------|-----------------|--|
| 3.2.3            | RHEL 7.3        | RHOSP10 (packages dated Apr. 15, 2017)   |
| 3.2.6            | RHEL 7.4        | RHOSP10 (packages dated Feb. 2, 2018)  |
| 4.1              | RHEL 7.4        | RHOSP10 (packages dated Feb. 27, 2018)   |
| 4.1.1            | RHEL 7.5        | RHOSP10 (packages dated Jun. 4, 2018)<br>RHOSP11 (packages dated Jun. 4, 2018) |
| 4.1.2            | RHEL 7.5        | RHOSP10 (packages dated Oct 29, 2018)  |
| 4.1.3            | RHEL 7.5        | RHOSP10 (packages dated Oct 29, 2018)  |





**CAUTION:** Set the Red Hat Satellite filter end date to October 29, 2018 before proceeding with the upgrade.

## Upgrade the Undercloud

Upgrade the undercloud to the most current RHOSP10 version.

1. Log in to the undercloud as the stack user.

```
$ su - stack
```

2. Update the Contrail repositories.

```
$ sudo rm -rf /etc/yum.repos.d/*contrail*
```

```
$ curl http://newrepo.contrail41-dev.repo -o /etc/yum.repos.d/localrepo.repo
```

3. Stop the main OpenStack platform services.

```
$ sudo systemctl stop 'openstack-*' 'neutron-*' httpd
```

4. Update the python-tripleoclient package and its dependencies to ensure you have the most current scripts for the minor version update.

```
$ sudo yum update python-tripleoclient
```

5. Upgrade the undercloud.

```
$ openstack undercloud upgrade
```

6. Reboot the node.

```
$ sudo reboot
```

7. Wait until the node reboots, then check the status of all services.

**NOTE:** It can take as much as 10 minutes or more for the openstack-nova-compute to become active after a reboot.

```
$ sudo systemctl list-units "openstack*" "neutron*" "openvswitch"
```

8. Verify the version of RHEL after the undercloud upgrade.

**NOTE:** Contrail does not support undercloud Red Hat version running with RHEL-7.6 as part of Contrail 4.1.3 release.

```
[root@undercloud ~]# cat /etc/redhat-release
Red Hat Enterprise Linux Server release 7.5 (Maipo)
[root@undercloud ~]#
```

9. Verify the existence of the overcloud and its nodes.

```
$ openstack stack list
```

```
$ ironic node-list
```

10. Verify that all OpenStack servers are Active.

```
$ openstack server list
```

**Figure 35: Server List**

| ID                                   | Name                                  | Status | Networks            | Image Name     |
|--------------------------------------|---------------------------------------|--------|---------------------|----------------|
| e37480b0-e098-4216-aea4-04b028aa5fb9 | overcloud-contrailanalytics-2         | ACTIVE | ctiplane=192.0.2.11 | overcloud-full |
| c77dcc49-c039-4855-bc91-e217bd34a51e | overcloud-contrailanalytics-1         | ACTIVE | ctiplane=192.0.2.12 | overcloud-full |
| 63d147ec-61e0-4d49-b1cb-6b68b7e5f48f | overcloud-contrailanalytics-0         | ACTIVE | ctiplane=192.0.2.21 | overcloud-full |
| 913d0bc9-9e9b-4987-a858-0389345e7025 | overcloud-controller-0                | ACTIVE | ctiplane=192.0.2.14 | overcloud-full |
| e66e86b1-df4a-4246-8e24-41461d617c18 | overcloud-controller-2                | ACTIVE | ctiplane=192.0.2.15 | overcloud-full |
| 9bbc1552-9ac1-4f27-846c-7756b14b84de | overcloud-controller-1                | ACTIVE | ctiplane=192.0.2.22 | overcloud-full |
| 8f8f5b9b-d5e2-4349-8d46-a488f2b7ab56 | overcloud-contrailanalyticsdatabase-0 | ACTIVE | ctiplane=192.0.2.17 | overcloud-full |
| 505c00f5-6715-48a7-a9ea-61ff157aa28a | overcloud-contrailanalyticsdatabase-2 | ACTIVE | ctiplane=192.0.2.24 | overcloud-full |
| ba8abe03-071b-4326-8fc0-715e93db3e4d | overcloud-contrailanalyticsdatabase-1 | ACTIVE | ctiplane=192.0.2.19 | overcloud-full |
| a7763383-92a0-4304-afc3-8aebf0ed2a05 | overcloud-contrailcontroller-2        | ACTIVE | ctiplane=192.0.2.20 | overcloud-full |
| e381db99-add0-4887-ac96-1e40d4108fc8 | overcloud-contrailcontroller-0        | ACTIVE | ctiplane=192.0.2.18 | overcloud-full |
| c808dcf1-4906-491e-8780-1f099c6a776c | overcloud-novacompute-0               | ACTIVE | ctiplane=192.0.2.16 | overcloud-full |
| 7eabc822-16d6-4fc6-afd7-7ba662ce7e92 | overcloud-contrailcontroller-1        | ACTIVE | ctiplane=192.0.2.23 | overcloud-full |

## Update Red Hat Director Image Archives

The undercloud update process might download new image archives from the rhosp-director images and the rhosp-director-ipa packages. You will have to update your existing system with any new image archives.

1. Check the yum log to determine if new image archives are available.

```
$ sudo grep "rhaps-director-images" /var/log/yum.log
```

```
[stack@undercloud]$ sudo grep "rhaps-director-images" /var/log/yum.log

Oct 26 15:09:20 Installed: rhosp-director-images-ipa-10.0-20180821.1.el7ost.noarch
Oct 26 15:10:10 Installed: rhosp-director-images-10.0-20180821.1.el7ost.noarch
```

2. If new image archives are available, replace your current images with the new images. Before deploying any new images, remove any existing images from the images undercloud on the stack user's home (/home/stack/images).

```
$ rm -rf ~/images/*
```

3. Extract the new image archives.

```
mkdir images
cd images
for i in /usr/share/rhaps-director-images/overcloud-full-latest-10.0.tar /usr/share/rhaps-
director-images/ironic-python-agent-latest-10.0.tar; do tar -xvf $i; done
```

4. Install the Contrail package on the overcloud image by using the virt-customize command.

```
export LIBGUESTFS_BACKEND=direct /usr/bin/virt-customize -a /home/stack/images/overcloud-
full.qcow2 \
--copy-in /etc/yum.repos.d/mylocalrepo.repo:/etc/yum.repos.d \
--run-command 'yum -y install puppet-tripleo contrail-tripleo-puppet puppet-contrail'\
--run-command ' cp -r /usr/share/contrail-tripleo-puppet/ /usr/share/openstack-puppet/modules/
tripleo/ ' \
--run-command 'rm -fr /var/cache/yum/*' \
--run-command 'yum clean all' \ --selinux-relabel
```

5. Import the new image archives into the undercloud and configure nodes to use the new images.

```
$ openstack overcloud image upload --update-existing --image-path /home/stack/images/
```

6. Verify that the images are uploaded.

```
$ openstack image list
```

| ID                                   | Name                   | Status |
|--------------------------------------|------------------------|--------|
| a518a08c-3c49-47b7-9705-a7da5c90dcc6 | bm-deploy-ramdisk      | active |
| b7d75ff7-926b-426c-aa2e-424f5d9f8328 | bm-deploy-kernel       | active |
| cd7eedc0-2ed2-4a21-8359-9d8ee89374ac | overcloud-full         | active |
| 9c9e46f4-d571-49fb-a485-4653b6a52b44 | overcloud-full-initrd  | active |
| 740b6cb9-5757-475b-896b-49c526807671 | overcloud-full-vmlinuz | active |

7. Show the details of the new image that has been created. The new image will be used to add a new node in the overcloud.

```
$ openstack image show overcloud-full
```

| Field            | Value   |
|------------------|---|
| checksum         | 13e67f5039dc7e69b5bbc494d8838b8d  |
| container_format | bare  |
| created_at       | 2018-10-26T20:02:18.000000  |
| deleted          | False   |
| deleted_at       | None  |
| disk_format      | qcow2   |
| id               | cd7eedc0-2ed2-4a21-8359-9d8ee89374ac  |
| is_public        | True  |
| min_disk         | 0   |
| min_ram          | 0   |
| name             | overcloud-full  |
| owner            | 0cd0e938b0fe46ef9c431715395f9469  |
| properties       | kernel_id='740b6cb9-5757-475b-896b-49c526807671', ramdisk_id='9c9e46f4-d571-49fb-a485-4653b6a52b44' |
| protected        | False   |
| size             | 1469448192  |
| status           | active  |
| updated_at       | 2018-10-26T20:02:52.000000  |
| virtual_size     | None  |

8. Verify contrail-status on all Contrail nodes. All services in the Contrail nodes, except the controller (OpenStack), should be up and running before proceeding with the upgrade.

```
[stack@undercloud ~]$ for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '=' -f2); do ssh heat-admin@$i sudo contrail-status; done
```

## Prepare Repositories on all Nodes

1. Delete existing repositories on all overcloud nodes. Verify each deletion.

```
sudo for ipnode in $(nova list | sed '4,$ !d;$d' | awk -F 'ctlplane=' '{print $2}' | tr -d '|'); do echo cleaning yum repolist on $ipnode && ssh heat-admin@$ipnode 'find /etc/yum.repos.d/ ! -name 'contrail-install.repo' -type f -exec sudo rm -f {} +' ; done
```

2. Add new repositories on all overcloud nodes. Verify each addition.

```
sudo for ipnode in $(nova list | sed '4,$ !d;$d' | awk -F 'ctlplane=' '{print $2}' | tr -d '|'); do echo cleaning yum repolist on $ipnode && ssh heat-admin@$ipnode ' curl http://newrepo.contrail41-dev.repo -o /etc/yum.repos.d/localrepo.rep' ; done
```

## Upgrade the Operating System on Contrail Nodes

1. Define a list (\$iplist) that contains all Contrail nodes. Run the following command on undercloud VM as stack user.

```
Iplist=" @IPcontrailController1 @IPContrailController2 ..."
```



**CAUTION:** Attach the new satellite subscription key on all overcloud nodes before upgrading the overcloud packages. Satellite must be synced with filter end date *29th Oct 2018*. Make sure to clear cache by typing **sudo yum clean all**.

2. Upgrade the operating system for all nodes in the iplist. Run the following command on undercloud VM as stack user

```
sudo for ipnode in $iplist; do echo -e "\n\n\t*****upgrade node : $ipnode *****" && ssh heat-admin@$ipnode 'sudo yum update -y --disablerepo=*contrail* --skip-broken && exit' ; done
```

3. Reboot overcloud contrail compute nodes, if there is any change in the kernel version. This needs to be done before installing contrail packages on compute VM.

```
Supported kernel versions: 3.10.0-862.11.6.el7.x86_64 and 3.10.0-957.el7.x86_64
----- [root@overcloud-novacompute-2 ~]# modinfo vrouter filename: /lib/modules/
3.10.0-862.11.6.el7.x86_64/extra/net/vrouter/vrouter.ko version: 4.1.3.0 license: GPL retpoline: Y
rhelversion: 7.5
```

## Prepare the Contrail Packages

To prepare the Contrail packages for the installation from a local repository:

1. Navigate to the Contrail repository and perform the following tasks:

- Delete the existing Contrail repositories.

All existing repositories in the undercloud and overcloud will be deleted during these steps.

- Access the Contrail update package.
- Copy the SNMP packages into the repository:
  - net-snmp-5.7.2-37.el7.x86\_64.rpm
  - net-snmp-agent-libs-5.7.2-37.el7.x86\_64.rpm
  - net-snmp-libs-5.7.2-37.el7.x86\_64.rpm

- net-snmp-utils-5.7.2-37.el7.x86\_64.rpm

In the provided example, all 4 of these files are in the `/mnt/net-snmp/` directory and all files from the directory are copied into the repository.

- Unsubscribe every node with all registered satellite server repositories.
- Delete all repositories on undercloud and overcloud nodes, and replace these deleted repositories with a Contrail repository.
- Clean the yum cache, verify the repository list, and check for yum updates.

A sample procedure:

```
[stack@undercloud ~]#
sudo su -
cd /var/www/html/contrail
rm -rf /var/www/html/contrail/*
#enter the location of the contrail update package
tar -xzf /mnt/contrail-install-packages_4.1.3.0-30-newton.tgz
#copy prerequisite snmp packages; in this setup packages are in /mnt/net-snmp/
cp /mnt/net-snmp/* .
rm -rf /var/www/html/contrail/repodata/usr/bin/createrepo /var/www/html/contrail/subscription-
manager repos --disable=*subscription-manager unregister
rm -f /etc/yum.repos.d/*
#create local repo file
echo -e '[Contrail]\nname=Contrail Repo\nbaseurl=http://192.168.24.1/contrail
\nenabled=1\npggcheck=0' > /etc/yum.repos.d/contrail.repo
# disable yum plugins
sed -i 's/plugins=1/plugins=0/g' /etc/yum.conf
yum clean all
rm -rf /var/cache/yum/*
yum check-update
exit
yum repolist

[stack@undercloud ~]#
. stackrc;for ipnode in $(nova list | sed '4,$ !d;$d'| awk -F 'ctlplane=' '{print $2}' | tr -
d '|');
do echo "Node $ipnode";
echo "sudo subscription-manager repos --disable=*;
sudo subscription-manager unregister;
sudo rm -f /etc/yum.repos.d/*;
sudo echo -e '[Contrail]\nname=Contrail Repo\nbaseurl=http://192.168.24.1/contrail
```

```
\nenabled=1\npggcheck=0' > /tmp/contrail.repo;
sudo mv /tmp/contrail.repo /etc/yum.repos.d/;
sudo sed -i 's/plugins=1/plugins=0/g' /etc/yum.conf;
sudo yum clean all;sudo rm -rf /var/cache/yum/*;
sudo yum repolist;sudo yum check-update" | ssh heat-admin@$ipnode bash;
done
```

2. Check the undercloud Contrail packages versions for `contrail-tripleo-puppet`, `puppet-contrail`, and `contrail-tripleo-heat-templates`. The newest versions of those packages must be installed before proceeding with the overcloud upgrade. See the following example, with current package versions.

```
[stack@undercloud~]$ rpm -qa | grep contrail

puppet-contrail-4.1.3.0-NN.el7.noarch
contrail-tripleo-heat-templates-4.1.3.0-NN.el7.noarch
contrail-tripleo-puppet-4.1.3.0-NN.el7.noarch
```

## Upgrade the Contrail Heat Templates

You must copy the new Contrail Heat templates package to the undercloud node, while retaining a copy of the Heat templates that were used for the existing deployment.

1. Make a copy of all of the Heat templates that were used for deployment and save the copies, because the existing files will be overwritten by the new versions. The templates to copy are of the form `contrail-services.yaml`, `contrail-net.yaml`, and so on.

**NOTE:** Red Hat does not support changing IP address of the existing cluster as a part of upgrade. Do not change IP address of the cluster while creating new `tripleo-heat-templates`

2. Copy the new `contrail-tripleo-heat-templates` to the undercloud node.

```
cp /home/stack/tripleo-heat-templates /home/stack/tripleo-heat-templates-bk

sudo cp -r /usr/share/contrail-tripleo-heat-templates/environments/contrail /home/stack/
tripleo-heat-templates/environments/

sudo cp -r /usr/share/contrail-tripleo-heat-templates/puppet/services/network/* /home/stack/
tripleo-heat-templates/puppet/services/network
```

**NOTE:** The directory `/home/stack/tripleo-heat-templates` is user defined, it can be *User Defined-directory*>`/openstack-tripleo-heat-templates`

## Modify the Yum Update Script for TripleO Puppet

Following Puppet commands must be added to the `yum_update` script before starting the upgrade. The script is located at:

`/home/stack/tripleo-heat-templates/extraconfig/tasks/yum_update.sh`

1. Update the following Puppet commands in the `yum_update.sh` after the line `“echo -n “false” > $heat_outputs_path.update_managed_packages”`.

Refer to the following patch for details regarding the exact placement of the commands patch:  
[https://github.com/Juniper/contrail-tripleo-heat-templates/blob/stable/newton/environments/contrail/yum\\_updates.patch](https://github.com/Juniper/contrail-tripleo-heat-templates/blob/stable/newton/environments/contrail/yum_updates.patch)

```
yum install -y contrail-tripleo-puppet puppet-contrail
```

```
rsync -a /usr/share/contrail-tripleo-puppet/ /usr/share/openstack-puppet/modules/tripleo/
```

2. Update the fields `*contrail version` and `*contrail repo` in `contrail-services.yaml`.

Default parameter for `contrailVersion` is 4.

Filename: `/home/stack/tripleo-heat-templates/environments/contrail/contrail-services.yaml`.

Add the following parameters:

`ContrailVersion: 4`

`ContrailRepo : <location of the contrail-41 repo>`

**NOTE:** `/home/stack/tripleo-heat-templates` directory is user defined and it can be directory name under stack user.

## Update the Overcloud Deployment Plan



1. Make a copy of the existing deploy script to the **update-stack.sh** file by re-running the command used for cloud deployment and adding the suffix `-update-plan-only`.

```
<openstack overcloud deploy> -update-plan-only
```

Example:

```
openstack overcloud deploy --update-plan-only --templates /home/stack/tripleo-heat-templates/ \
\
  --roles-file /home/stack/tripleo-heat-templates/environments/contrail/roles_data.yaml \
-e /home/stack/tripleo-heat-templates/environments/puppet-pacemaker.yaml \
-e /home/stack/tripleo-heat-templates/environments/contrail/contrail-services.yaml \
-e /home/stack/tripleo-heat-templates/environments/network-isolation.yaml \
-e /home/stack/tripleo-heat-templates/environments/contrail/contrail-net.yaml \
-e /home/stack/tripleo-heat-templates/environments/ips-from-pool-all.yaml \
-e /home/stack/tripleo-heat-templates/environments/network-management.yaml \
-e /home/stack/tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/environment- \
rhel-registration.yaml \
-e /home/stack/tripleo-heat-templates/environments/hostname-map.yaml \
-e /home/stack/tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/rhel- \
registration-resource-registry.yaml \
--libvirt-type qemu
```

2. If you are using a local repository for the update and the `environment-rhel-registration.yaml` and `rhel-registration-resource-registry.yaml` files are present, delete these lines from the deploy script:

```
-e /home/stack/tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/environment- \
rhel-registration.yaml \
-e /home/stack/tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/rhel- \
registration-resource-registry.yaml \
```

3. Prepare the YAML files for the update:

- Verify each `.yaml` template referenced in the `update-stack.sh` file contains the original settings that match the files that were backed up.
- In the `contrail-net.yaml` file, adapt all referenced templates from **heat\_template\_version: newton** to **heat\_template\_version: 2015-04-30**. Keep all other original installation settings in this file.

4. Update the deployment plan.



```

IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
COMPLETE
update finished with status COMPLETE

```

2. Verify the overcloud stack status, the contrail-status, and the contrail-version after the upgrade.

### Overcloud Stack Status

```

[stack@undercloud]# heat stack-list
WARNING (shell) "heat stack-list" is deprecated, please use "openstack stack list" instead
+-----+-----+-----+-----+
+-----+
| id                | stack_name | stack_status  | creation_time
| updated_time      |            |                |
+-----+-----+-----+-----+
+-----+
| e873706c-7fb3-44ba-80dc-30b0fdbd519e | overcloud  | UPDATE_COMPLETE | 2019-03-13T19:20:52Z
| 2019-03-13T22:01:05Z |
+-----+-----+-----+-----+
+-----+
[stack@undercloud ~]$

```

### Contrail Stack Status

```

sudo for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '=' -f2); do ssh heat-admin@$i sudo contrail-status; done

```

## Contrail Version Check

```
for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '=' -f2); do ssh heat-admin@  
$i sudo contrail-version; done
```

### RELATED DOCUMENTATION

[Upgrade Procedure for RHOSP-based Contrail 3.2.x to Contrail 4.1](#)

[Upgrade Procedure for RHOSP-based Contrail 4.1.1 to Contrail 4.1.2](#)

## Upgrade Procedure for RHOSP-based Contrail 4.1.3 to Contrail 4.1.4

### IN THIS SECTION

- [Prerequisites | 293](#)
- [Post-Installation | 293](#)
- [Acquire the Software | 294](#)
- [Upgrade the Undercloud | 294](#)
- [Update Red Hat Director Image Archives | 299](#)
- [Upgrade the Operating System on Contrail Nodes | 301](#)
- [Prepare the Contrail Packages | 303](#)
- [Upgrade the Contrail Heat Templates | 303](#)
- [Modify the Yum Update Script for TripleO Puppet | 303](#)
- [Update the Overcloud Deployment Plan | 304](#)
- [Upgrade the Overcloud | 305](#)
- [Upgrade Cautions | 307](#)

This section presents the steps to upgrade a RHOSP-based Contrail deployment from Contrail version 4.1.3 to Contrail version 4.1.4.

## Prerequisites

Ensure you have a cloud up and running with RHOSP10 and Contrail 4.1.3 before you proceed with the upgrade procedure.

This procedure has been validated with the following Contrail, Red Hat, and OpenStack versions.

**Table 6: Pre-Installation Software Versions**

| Contrail Version | Red Hat Version                       | OpenStack Version                         |
|------------------|---------------------------------------|---|
| 4.1.3            | RHEL 7.6 (3.10.0-957.el7.x86_64)      | RHOSP10 (packages dated October 29, 2018) |
| 4.1.3            | RHEL 7.5 (3.10.0-862.11.6.el7.x86_64) | RHOSP10 (packages dated October 29, 2018) |



**CAUTION:** Set the Red Hat Satellite filter end date to December 9, 2019 before proceeding with the upgrade.

## Post-Installation

After the installation, you'll have a cloud networking running RHOSP10 and Contrail 4.1.4. The Red Hat Enterprise Linux (RHEL) kernel version updates to 7.7 during this procedure.

[Table 7 on page 293](#) summarizes the post-installation software versions.

**Table 7: Post Installation Software Summary**

| Contrail Version | Red Hat Version   | OpenStack Version                         |
|------------------|---|---|
| 4.1.4            | RHEL 7.7 (3.10.0-1062.el7.x86_64)<br>RHEL 7.7 (3.10.0-1062.1.2.el7.x86_64)<br>RHEL 7.7 (3.10.0-1062.9.1.el7.x86_64) | RHOSP10 (packages dated December 9, 2019) |

Contrail version R4.1.4 supports net-snmp package version 5.7.2-43 to support SNMP. The net-snmp packages come from Red Hat, with the exception of the *net-snmp-python-5.7.2-43.el7.x86\_64.rpm* package which is provided in the Contrail repository.

Table 8 on page 294 summarizes the net-snmp depend packages and their associated repository locations.

**Table 8: Post Installation Software Summary**

| Net-SNMP Depend Packages                           | Repository        |
|--|-------------------|
| <i>net-snmp-5.7.2-43.el7.x86_64.rpm</i>            | Red Hat Satellite |
| <i>net-snmp-agent-libs-5.7.2-43.el7.x86_64.rpm</i> | Red Hat Satellite |
| <i>net-snmp-libs-5.7.2-43.el7.x86_64.rpm</i>       | Red Hat Satellite |
| <i>net-snmp-python-5.7.2-43.el7.x86_64.rpm</i>     | Contrail          |
| <i>net-snmp-utils-5.7.2-43.el7.x86_64.rpm</i>      | Red Hat Satellite |

## Acquire the Software

To download the software images for this procedure:

1. Go to the [Juniper Networks Support site for Contrail](#).
2. Select **OS** as *Contrail* and **Version** as *4.1.4*. Download the images that apply to your environment.

## Upgrade the Undercloud

1. Log in to the undercloud as the stack user.

```
$ su - stack
```

2. Update the Contrail repositories.

- Backup the Contrail 4.1.3 packages to a repository with a different name. In this example, the packages are moved to a repository named *contrail-R4-1-3*.

```
[stack@undercloud ~]$ cd /var/www/html/
[stack@undercloud html]$ sudo mv contrail/ contrail-R4-1-3
```

- Create a new repository directory to store the Contrail 4.1.4 packages:

```
[stack@undercloud html]$ sudo mkdir contrail
```

3. Copy the downloaded file—in the provided sample, the file is *contrail-install-packages\_4.1.4.0-63-newton.tgz*—to the Contrail repository created in Step 2.

**NOTE:** This step assumes that you've already downloaded the Contrail software. See ["Acquire the Software" on page 294](#).

```
[stack@undercloud contrail]$ ls -lrt
total 377104
-rw-r--r--. 1 root root 386151602 Mar 14 06:58 contrail-install-packages_4.1.4.0-63-
newton.tgz
```

4. Untar the downloaded tgz file.

```
[stack@undercloud contrail]$ sudo tar -xvf contrail-install-packages_4.1.4.0-63-newton.tgz
```

5. Create a repository in the new directory:

```
[stack@undercloud contrail]$ pwd
/var/www/html/contrail

[stack@undercloud contrail]$ sudo createrepo .
```

If the **createrepo** command is not available, download the createrepo package from Red Hat (Red Hat subscription required).

6. (Clusters deployed using Swift Puppet files only) If your Contrail 4.1 cluster was deployed using Swift Puppet, perform these steps:

- a. Remove overcloud artifacts from the undercloud:

```
[stack@undercloud ~]$ swift delete overcloud-artifacts
puppet-modules.tgz
overcloud-artifacts
```

- b. Delete the *deployments-artifacts.yaml* file if the file is present.

```
[stack@undercloud ~]$ ls /home/stack/.tripleo/environments/deployment-artifacts.yaml
[stack@undercloud ~]$ rm -rf /home/stack/.tripleo/environments/deployment-artifacts.yaml
```

- c. Clean the repositories and confirm that all repositories are available.

```
[stack@undercloud ~]$ sudo yum clean all
[stack@undercloud ~]$ sudo yum repolist
```

7. Stop the main OpenStack platform services.

```
$ sudo systemctl stop 'openstack-*' 'neutron-*' httpd
```

8. Update the *python-tripleoclient* package and its dependencies to ensure you have the most current scripts for the minor version update.

```
$ sudo yum update python-tripleoclient
```

9. Upgrade the undercloud.

```
$ openstack undercloud upgrade
```

10. Reboot the node.

```
$ sudo reboot
```

Wait for the node to reboot. The reboot process can take 10 or more minutes to complete.

11. Ensure the undercloud has the latest Contrail R4.1.4 contrail packages:

```
[stack@undercloud ~]$ rpm -qa | grep contrail

puppet-contrail-4.1.4.0-X.el7.noarch
contrail-tripleo-heat-templates-4.1.4.0-x.el7.noarch
contrail-tripleo-puppet-4.1.4.0-x.el7.noarch
python-gevent-1.1rc5-1contrail1.el7.x86_64
```

12. Ensure the undercloud has the latest RHOSP images:

```
[stack@undercloud]$ rpm -qa | grep direct
```



```

rhaps-director-images-10.0-20180821.1.el7ost.noarch
rhaps-director-images-10.0-20190829.1.el7ost.noarch
rhaps-director-images-10.0-20190918.1.el7ost.noarch
rhaps-director-images-ipa-10.0-20190829.1.el7ost.noarch
rhaps-director-images-ipa-10.0-20180821.1.el7ost.noarch
rhaps-director-images-ipa-10.0-20190918.1.el7ost.noarch

```

13. Review the `ironic node-list` output to confirm the following statuses for each package::

- **Power state** is *power on*.
- **Provision State** is *active*.
- **Maintenance** is *False*.

```

[stack@undercloud ~]$ ironic node-list
+-----+-----+-----+-----+
| Name                | Power | Provisioning | Maintenance |
|                    | State | State        |             |
+-----+-----+-----+-----+
| controller-3        | power on | active      | False      |
| compute-5c5s35      | power on | active      | False      |
| contrail-controller1 | power on | active      | False      |
| contrail-analytics1 | power on | active      | False      |
| contrail-controller-3 | power on | active      | False      |
| contrail-controller-2 | power on | active      | False      |
| contrail-analytics-database1 | power on | active      | False      |
| controller-2        | power on | active      | False      |
| controller1         | power on | active      | False      |
| compute-5c5s37      | power on | active      | False      |
| compute-5c5s36      | power on | active      | False      |
| contrail-analytics-2 | power on | active      | False      |
| contrail-analytics-3 | power on | active      | False      |
| compute-5c5s38      | power on | active      | False      |
| contrail-analytics-database-3 | power on | active      | False      |
| contrail-analytics-database-2 | power on | active      | False      |
+-----+-----+-----+-----+

```

**NOTE:** This output presentation has been modified for readability. The *UUID* and *Instance UUID* fields were removed as part of this modification.

14. Verify that all OpenStack servers are in the Active state.

```
[stack@undercloud ~]$ openstack server list
+-----+-----+
| Name                                     | Status |
+-----+-----+
| overcloud-contrailanalytics-2-4-1-4-7-7 | ACTIVE |
| overcloud-controller-0-4-1-4-7-7       | ACTIVE |
| overcloud-contrailanalytics-0-4-1-4-7-7 | ACTIVE |
| overcloud-contrailanalyticsdatabase-2-4-1-4-7-7 | ACTIVE |
| overcloud-contrailanalytics-1-4-1-4-7-7 | ACTIVE |
| overcloud-contrailanalyticsdatabase-0-4-1-4-7-7 | ACTIVE |
| overcloud-contrailcontroller-1-4-1-4-7-7 | ACTIVE |
| overcloud-contrailanalyticsdatabase-1-4-1-4-7-7 | ACTIVE |
| overcloud-contrailcontroller-2-4-1-4-7-7 | ACTIVE |
| overcloud-contrailcontroller-0-4-1-4-7-7 | ACTIVE |
| compute-0-4-1-4-rhel-7-7               | ACTIVE |
| overcloud-contraildpk-0-4-1-4-7-7      | ACTIVE |
| overcloud-contraildpk-1-4-1-4-7-7     | ACTIVE |
| compute-1-4-1-4-rhel-7-7              | ACTIVE |
+-----+-----+
```

**NOTE:** This output presentation has been modified for readability. The *ID*, *Image Name*, and *Networks* fields were removed as part of this modification.

15. If new image archives are available, replace your current images with the new images.

Before uploading the new images onto the undercloud node, move any existing images from the images directory on the stack user's home directory (/home/stack/images).

```
$ mv /home/stack/images /home/stack/images-old
```

16. Extract the new image archives.

```
mkdir images
cd images
for i in /usr/share/rhosp-director-images/overcloud-full-latest-10.0.tar /usr/share/rhosp-
director-images/ironic-python-agent-latest-10.0.tar; do tar -xvf $i; done
```

17. Import the new image archives into the undercloud and configure the nodes to use the new images.

```
$ openstack overcloud image upload --update-existing --image-path /home/stack/images/
```

18. Verify that the images are uploaded:

```
$ glance image-list
```

19. Observe the contrail-status on all Contrail nodes. All services in the Contrail nodes, except the controller (OpenStack), should be up and running before proceeding with the upgrade.

```
[stack@undercloud ~]$ source stackrc
[stack@undercloud ~]$ for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '=' -f2); do ssh heat-admin@$i sudo contrail-status; done
```

20. Ensure that all overcloud node contrail repository pointers are properly pointing to the contrail repository.

*Contrail Analytics Example:*

```
[root@overcloud-contrailanalytics-0 heat-admin]# cat /etc/yum.repos.d/contrail.repo
[Contrail]
name=Contrail Repo
baseurl=http://192.168.24.1/contrail
enabled=1
gpgcheck=0
protect=1
metadata_expire=30
```

## Update Red Hat Director Image Archives

The undercloud update process might download new image archives from the rhosp-director images and the rhosp-director-ipa packages. You will have to update your existing system with any new image archives.

1. Check the yum log to determine if new image archives are available.

```
$ sudo grep "rhaps-director-images" /var/log/yum.log
```

```
[stack@undercloud]$ sudo grep "rhaps-director-images" /var/log/yum.log

Dec 12 15:09:20 Installed: rhosp-director-images-ipa-10.0-20190918.1.el7ost.noarch
Dec 12 15:10:10 Installed: rhosp-director-images-10.0-20190918.1.el7ost.noarch
```

2. If new image archives are available, replace your current images with the new images. Before deploying any new images, remove any existing images from the images undercloud on the stack user's home (/home/stack/images).

```
$ rm -rf ~/images/*
```

3. Extract the new image archives.

```
mkdir images
cd images
for i in /usr/share/rhaps-director-images/overcloud-full-latest-10.0.tar /usr/share/rhaps-
director-images/ironic-python-agent-latest-10.0.tar; do tar -xvf $i; done
```

4. Install the Contrail package on the overcloud image by using the virt-customize command.

```
export LIBGUESTFS_BACKEND=direct /usr/bin/virt-customize -a /home/stack/images/overcloud-
full.qcow2 \
-copy-in /etc/yum.repos.d/mylocalrepo.repo:/etc/yum.repos.d \
-run-command 'yum -y install puppet-tripleo contrail-tripleo-puppet puppet-contrail'\
-run-command ' cp -r /usr/share/contrail-tripleo-puppet/ /usr/share/openstack-puppet/modules/
tripleo/ ' \
-run-command 'rm -fr /var/cache/yum/*' \
-run-command 'yum clean all' \ -selinux-relabel
```

5. Import the new image archives into the undercloud and configure nodes to use the new images.

```
$ openstack overcloud image upload -update-existing -image-path /home/stack/images/
```

6. Verify that the images are uploaded.

```
$ openstack image list
```

| ID                                   | Name                   | Status |
|--------------------------------------|------------------------|--------|
| a518a08c-3c49-47b7-9705-a7da5c90dcc6 | bm-deploy-ramdisk      | active |
| b7d75ff7-926b-426c-aa2e-424f5d9f8328 | bm-deploy-kernel       | active |
| cd7eedc0-2ed2-4a21-8359-9d8ee89374ac | overcloud-full         | active |
| 9c9e46f4-d571-49fb-a485-4653b6a52b44 | overcloud-full-initrd  | active |
| 740b6cb9-5757-475b-896b-49c526807671 | overcloud-full-vmlinuz | active |

7. Show the details of the new image that has been created. The new image will be used to add a new node in the overcloud.

```
$ openstack image show overcloud-full
```

8. Verify contrail-status on all Contrail nodes. All services in the Contrail nodes, except the controller (OpenStack), should be up and running before proceeding with the upgrade.

```
[stack@undercloud ~]$ for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '=' -f2); do ssh heat-admin@$i sudo contrail-status; done
```

## Upgrade the Operating System on Contrail Nodes

To upgrade the operating system on Contrail nodes:

1. Define a list (\$iplist) that contains all Contrail nodes. Run the following command on undercloud VM as a stack user.

```
Iplist=" @IPcontrailController1 @IPcontrailController2 ..."
```



**CAUTION:** Attach the new satellite subscription key on all overcloud nodes before upgrading the overcloud packages. Satellite must be synced with filter end date *9th Dec 2019*. Make sure to clear cache by typing **sudo yum clean all**.

2. Upgrade the operating system for all nodes in the iplist.

Run the following command on undercloud VM as a stack user:

```
sudo for ipnode in $iplist; do echo -e "\n\n\t*****upgrade node : $ipnode *****" && ssh heat-admin@$ipnode 'sudo yum update -y --disablerepo=*contrail* --skip-broken && exit' ; done
```

3. (Compute nodes only) Reboot overcloud contrail compute nodes. After the reboot, stop the supervisor-vrouter service.

This step needs to be performed before installing contrail packages on the compute VM.

Compute services may go down after rebooting with the latest kernel. These services return later in this procedure during the **openstack overcloud deploy** process.

*Reboot Procedure:*

```
[root@compute-1-7-6 modules]# sudo reboot
Connection to 192.0.2.16 closed by remote host.
Connection to 192.0.2.16 closed.
```

*Post-Reboot:*

```
[stack@undercloud-R4-1-2-b22 ~]$ ssh heat-admin@192.0.2.16
Warning: Permanently added '192.0.2.16' (ECDSA) to the list of known hosts.
Last login: Sat Dec  7 03:46:07 2019 from gateway
[heat-admin@compute-1-7-6 ~]$ sudo su
[root@compute-1-7-6 heat-admin]# contrail-status
vRouter is NOT PRESENT

== Contrail vRouter ==
supervisor-vrouter:      active
contrail-vrouter-agent   initializing
contrail-vrouter-nodemgr initializing
```

*Stop the supervisor-vrouter service:*

```
[root@compute-1-7-6 heat-admin]# service supervisor-vrouter stop
Stopping supervisor-vrouter (via systemctl): [ OK ]

[root@compute-1-7-6 heat-admin]# contrail-status
vRouter is NOT PRESENT

== Contrail vRouter ==
supervisor-vrouter: inactive
unix:///var/run/supervisord_vrouter.sockno
```

## Prepare the Contrail Packages

Check the undercloud Contrail packages versions for `contrail-tripleo-puppet`, `puppet-contrail`, and `contrail-tripleo-heat-templates`.

```
[stack@undercloud~]$ rpm -qa | grep contrail
```

## Upgrade the Contrail Heat Templates

You must copy the new Contrail Heat templates package to the undercloud node, while retaining a copy of the Heat templates that were used for the existing deployment.

1. Make a copy of all of the Heat templates that were used for deployment and save the copies, because the existing files will be overwritten by the new versions. The templates to copy are of the form `contrail-services.yaml`, `contrail-net.yaml`, and so on.

**NOTE:** Red Hat does not support changing IP address of the existing cluster as a part of upgrade. Do not change IP address of the cluster while creating new `tripleo-heat-templates`

2. Copy the new `contrail-tripleo-heat` templates to the undercloud node.

```
sudo cp -r /usr/share/contrail-tripleo-heat-templates/environments/contrail /home/stack/
tripleo-heat-templates/environments/
```

```
sudo cp -r /usr/share/contrail-tripleo-heat-templates/puppet/services/network/* /home/stack/
tripleo-heat-templates/puppet/services/network
```

**NOTE:** The directory `/home/stack/tripleo-heat-templates` is user defined, it can be *User Defined-directory>/openstack-tripleo-heat-templates*

## Modify the Yum Update Script for TripleO Puppet

Following Puppet commands must be added to the `yum_update` script before starting the upgrade. The script is located at:

```
/home/stack/tripleo-heat-templates/extraconfig/tasks/yum_update.sh
```

1. Update the following Puppet commands in the `yum_update.sh` after the line `“echo -n “false” > $heat_outputs_path.update_managed_packages”`.

Refer to the following patch for details regarding the exact placement of the commands patch:  
[https://github.com/Juniper/contrail-tripleo-heat-templates/blob/stable/newton/environments/contrail/yum\\_updates.patch](https://github.com/Juniper/contrail-tripleo-heat-templates/blob/stable/newton/environments/contrail/yum_updates.patch)

```
yum install -y contrail-tripleo-puppet puppet-contrail

rsync -a /usr/share/contrail-tripleo-puppet/ /usr/share/openstack-puppet/modules/tripleo/
```

2. Update the fields `*contrail version` and `*contrail repo` in `contrail-services.yaml`.

Default parameter for `contrailVersion` is 4.

Filename: `/home/stack/tripleo-heat-templates/environments/contrail/contrail-services.yaml`.

Add the following parameters:

`ContrailVersion: 4`

`ContrailRepo : <location of the contrail-41 repo>`

**NOTE:** `/home/stack/tripleo-heat-templates` directory is user defined and it can be directory name under `stack` user.

## Update the Overcloud Deployment Plan

1. Update the current plan by re-running the command used for cloud deployment and adding the suffix `-update-plan-only`.

```
openstack overcloud deploy -update-plan-only
```

Example:

```
openstack overcloud deploy --update-plan-only --templates /home/stack/tripleo-heat-templates/ \
\
  --roles-file /home/stack/tripleo-heat-templates/environments/contrail/roles_data.yaml \
  -e /home/stack/tripleo-heat-templates/environments/puppet-pacemaker.yaml \
  -e /home/stack/tripleo-heat-templates/environments/contrail/contrail-services.yaml \
  -e /home/stack/tripleo-heat-templates/environments/contrail/network-isolation.yaml \
  -e /home/stack/tripleo-heat-templates/environments/contrail/contrail-net.yaml \
```



```
-e /home/stack/tripleo-heat-templates/environments/ips-from-pool-all.yaml \
-e /home/stack/tripleo-heat-templates/environments/network-management.yaml \
-e /home/stack/tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/environment-
rhel-registration.yaml \
-e /home/stack/tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/rhel-
registration-resource-registry.yaml \
--libvirt-type qemu
```

2. Make a copy of the existing deploy script to the `update-stack.sh`. The `update-stack.sh` is the script used to update the overcloud plan, and it references the same templates that were used to deploy the stack. All files used for the overcloud update should be identical to the files used for deployment, except `contrail-services` file that was updated with the latest `contrail-version` and `contrail-repo`.

```
cp deploy.sh update-stack.sh
```

3. Update the deployment plan.

```
./update-stack.sh
```

Example:

```
[stack@undercloud ~]$ ./update-stack.sh
  Removing the current plan files
  Uploading new plan files
  Started Mistral Workflow. Execution ID: 998a1b40--a034-8cff453acfb1
  Plan updated
  Deploying templates in the directory /tmp/tripleoclient-JulIDe/tripleo-heat-
templates
  Overcloud Endpoint: http://10.0.0.35:5000/v2.0
  Overcloud Deployed
```

## Upgrade the Overcloud



**CAUTION:** The steps in this section are service disrupting, and should only be performed within a maintenance window.

1. Update the overcloud stack.

```
$ openstack overcloud update stack -i overcloud
on_breakpoint: [u'overcloud-contrailanalyticsdatabase-0']
Breakpoint reached, continue? Regexp or Enter=proceed (will clear
```

```
4386bdc7-5087-4a4d-865c-0b0181ce9345), no=cancel update, C-c=quit interactive mode:
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
COMPLETE
update finished with status COMPLETE
```

2. Verify the overcloud stack status, the contrail-status, and the contrail-version after the upgrade.

### Overcloud Stack Status

```
[stack@undercloud ~]$ openstack stack list
+-----+-----+-----+-----+
| Stack Name | Stack Status | Creation Time | Updated Time |
+-----+-----+-----+-----+
| overcloud | UPDATE_COMPLETE | 2019-12-06T23:30:26Z | 2019-12-09T22:40:01Z |
+-----+-----+-----+-----+
```

**NOTE:** The `openstack stack list` output presentation has been modified for readability. The `ID` field was removed as part of this modification.

### Contrail Stack Status

```
sudo for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '=' -f2); do ssh heat-admin@$i sudo contrail-status; done
```

### Contrail Version Check

```
for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '=' -f2); do ssh heat-admin@$i sudo contrail-version; ssh heat-admin@$i sudo contrail-status -d ; done
```

## Upgrade Cautions



**CAUTION:** The steps to perform the overcloud upgrade are service disrupting, and should only be performed within a maintenance window.

The upgrade procedure may fail due to packages conflicts in Contrail analytics nodes. Some observed failures due to packages conflicts are detailed in this section. Continue with the deployment after applying the recommended solution.

### Analytics Node snmp-lib Version Conflict

Error message: Protected multilib versions: 1:net-snmp-libs-5.7.2-37.el7.x86\_64 != 1:net-snmp-libs-5.7.2-33.el7\_5.2.i686

Solution:

```
rpm -e --nodeps net-snmp-libs
```

### Services Need Manual Restart After Upgrade

Services may need to be restarted after performing this upgrade. The services might continue to run using Contrail 4.1.3-related processes for a period of time.

Enter the **contrail-status** command to see if the processes continued to run through the upgrade, and monitor the warning messages that appear.

Manually restart the services if you run into this issue.

In the following example, this issue is seen for the Contrail Analytics services immediately after the upgrade:

```
[heat-admin@overcloud-contrailanalytics ~]$ sudo contrail-status -d
Warning: supervisor-analytics.service changed on disk. Run 'systemctl daemon-reload' to reload
units.
== Contrail Analytics ==
supervisor-analytics:      active
contrail-alarm-gen         active      pid 975462, uptime 15 days, 19:07:11
contrail-analytics-api     active      pid 127224, uptime 20 days, 19:48:28
contrail-analytics-nodemgr active      pid 127219, uptime 20 days, 19:48:28
contrail-collector         active      pid 127222, uptime 20 days, 19:48:28
contrail-query-engine      active      pid 127223, uptime 20 days, 19:48:28
```

```

contrail-snmp-collector    active    pid 127220, uptime 20 days, 19:48:28
contrail-topology         active    pid 127221, uptime 20 days, 19:48:28

```

## Restoring Contrail Nodes in a RHOSP-based Environment

### IN THIS SECTION

- [Prerequisites | 308](#)
- [Verify the Controller Node Status and Rebuild the Node | 308](#)
- [Finish Rebuilding One or Two Contrail Controller Nodes | 310](#)
- [Finish the Rebuilding of all Contrail Controller Nodes | 311](#)
- [Rebuilding Contrail Analytics And Analytics Database Nodes | 312](#)
- [Finish Rebuilding the Analytics Nodes | 314](#)

Contrail nodes are virtual machines hosted on a KVM hypervisor. In the Contrail RHOSP environment, the Contrail nodes are of three types– controller nodes, analytics nodes, and analytics database nodes. From time-to-time, the system may encounter a node crash or other node failure. This topic describes how to restore one or more failed Contrail controller nodes or analytics and analytics database nodes.

Use the following procedures to rebuild one or more Contrail nodes.

### Prerequisites

Before attempting to rebuild one or more failed Contrail nodes, ensure that:

- The system is stable and the node has the correct status to be deployed again.
- The MAC address of the network interface card (NIC) used for PXE boot has not changed.
- If you are restoring more than one node, make a backup of the Contrail databases and make sure you have access to the backup file (\*.tar.bz2). For more information about backups, see "[Backing Up Contrail Databases Using JSON Format](#)" on page 1058.

### Verify the Controller Node Status and Rebuild the Node

This is the initial procedure for verifying the node is ready to be rebuilt.

1. Check the node status. The failed node status should be listed as Power State power off and Maintenance False.

In this example, `contrail-controller02` is the node to be rebuilt, and its status is Power State power off and Maintenance False. The status indicates that `contrail-controller02` is ready for rebuilding.

```
[stack@<director-controlzone>]$ ironic node-list --fields uuid name power_state maintenance
+-----+-----+-----+-----+
| UUID                                | Name                            | Power State | Maintenance |
+-----+-----+-----+-----+
| 92242f30-7131-462a-bbe4-b9535d8f6b77 | contrail-controller01          | power on   | False       |
| 3555e9a6-eb0b-45ba-9bc0-51964cf3851f | contrail-controller02          | power off  | False       |
| a0adffdb-410f-471d-9647-9925d9b28ae2 | contrail-controller03          | power on   | False       |
+-----+-----+-----+-----+
```

2. In some cases, the power state of a failed node could be none. If the power state is none, you must set the power to off. To set the Power State power off:

```
[stack@<director-controlzone> ~]$ ironic node-set-power-state <UUID of failed node> off
```

3. In some cases, the Maintenance mode could be set to True. If this is the case, you must set the Maintenance mode to False.

```
[stack@<director-controlzone> ~]$ ironic node-set-maintenance <UUID of failed node> false
```

4. Restore the node, and wait until its status turns to ACTIVE.

```
[stack@<director-computezone> ~]$ source ~/stackrc
[stack@<director-computezone> ~]$ openstack baremetal configure boot
[stack@<director-computezone> ~]$ nova rebuild contrail-controller-1 overcloud-compute
[stack@<director-controlzone> ~]$ openstack server list
+-----+-----+-----+-----+
| ID                                | Name                            | Status |
```

```

Networks          | Image Name      |
+-----+-----+
+-----+-----+
| 03af3384-3220-4b76-a0d0-b221d22a03a7 | contrail-controller-0      | ACTIVE |
ctlplane=192.168.210.32 | overcloud-full |
| a905105b-efb5-4d11-bec3-23a7cd19655e | contrail-controller-1      | ACTIVE |
ctlplane=192.168.210.31 | overcloud-full |
| 8a001ca2-0bf6-4662-ad81-252cdfccf09d | contrail-controller-2      | ACTIVE |
ctlplane=192.168.210.26 | overcloud-full |
+-----+-----+
+-----+-----+

```

5. Repeat steps 1-4 for each node that you want to restore.
6. After restoration, finish rebuilding the node or nodes, using the next procedures.

## Finish Rebuilding One or Two Contrail Controller Nodes

This procedure provides details to finish the rebuilding of the nodes when you are restoring one or two Contrail controller nodes.

1. Verify the status of the rebuild process. Wait until the status turns ACTIVE.

```

[stack@director-controlzone ~]$ openstack server list
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| ID          | Name          | Status |
Networks     | Image Name   |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| 03af3384-3220-4b76-a0d0-b221d22a03a7 | contrail-controller-0      | ACTIVE |
ctlplane=192.168.210.32 | overcloud-full |
| a905105b-efb5-4d11-bec3-23a7cd19655e | contrail-controller-1      | REBUILD|
ctlplane=192.168.210.31 | overcloud-full |
| 8a001ca2-0bf6-4662-ad81-252cdfccf09d | contrail-controller-2      | ACTIVE |
ctlplane=192.168.210.26 | overcloud-full |
+-----+-----+-----+-----+
+-----+-----+-----+-----+

```

2. Establish an SSH connection to the node you have rebuilt and observe the journal of the `os-collect-config` process until you see multiple occurrences of the message `No local metadata found ([/var/lib/os-collect-config/local-data])`.

If you want to rebuild two controller nodes, repeat this step for the other node before moving to the next step.

```
[stack@<director-controlzone> ~]$ ssh heat-admin@19x.xxx.xxx.31 sudo journalctl -u os-collect-config -f
```

3. Perform a full stack update to reconverge the stack and bring the system back to operational state.

```
[stack@<director-controlzone> ~]$ source ~/stackrc
[stack@<director-controlzone> ~]$ ./deploy.sh
```

## Finish the Rebuilding of all Contrail Controller Nodes

This procedure provides details to finish the rebuilding of the nodes when you are restoring all of the Contrail controller nodes.

1. Observe the status of the rebuild process. The status of nodes will display REBUILD while the rebuilding process is occurring.

```
[stack@<director-controlzone> ~]$ openstack server list
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| ID                               | Name                               | Status |
Networks                            | Image Name                        |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| 03af3384-3220-4b76-a0d0-b221d22a03a7 | contrail-controller-0             | REBUILD|
ctlplane=192.168.210.32 | overcloud-full |
| a905105b-efb5-4d11-bec3-23a7cd19655e | contrail-controller-1             | REBUILD|
ctlplane=192.168.210.31 | overcloud-full |
| 8a001ca2-0bf6-4662-ad81-252cdfccf09d | contrail-controller-2             | REBUILD|
ctlplane=192.168.210.26 | overcloud-full |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

2. Wait until the status of all nodes changes to ACTIVE.

```
[stack@<director-controlzone> ~]$ openstack server list
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

| ID                                   | Name                  | Status |
|--------------------------------------|-----------------------|--------|
| 03af3384-3220-4b76-a0d0-b221d22a03a7 | contrail-controller-0 | ACTIVE |
| a905105b-efb5-4d11-bec3-23a7cd19655e | contrail-controller-1 | ACTIVE |
| 8a001ca2-0bf6-4662-ad81-252cdfccf09d | contrail-controller-2 | ACTIVE |

3. Establish an SSH connection to each of the nodes that have been rebuilt and observe the journal of the `os-collect-config` process until you see multiple occurrences of the string `No local metadata found (/var/lib/os-collect-config/local-data')`.

```
[stack@<director-controlzone> ~]$ ssh heat-admin@19x.xxx.xxx.32 sudo journalctl -u os-collect-config -f
[stack@<director-controlzone> ~]$ ssh heat-admin@19x.xxx.xxx.31 sudo journalctl -u os-collect-config -f
[stack@<director-controlzone> ~]$ ssh heat-admin@19x.xxx.xxx.26 sudo journalctl -u os-collect-config -f
```

4. Retrieve the Contrail controller databases backup `.tar.bz2` and put it into your Director Control Zone, and perform a database restore. For more information about backups, see ["Backing Up Contrail Databases Using JSON Format" on page 1058](#)
5. Verify that the Contrail services are running.

```
[stack@<director-controlzone> ~]$ ssh heat-admin@19x.xxx.xxx.31 "sudo contrail-status"
```

6. Perform a full stack update to reconverge the stack and bring the system back to operational state.

```
[stack@<director-controlzone> ~]$ source ~/stackrc
[stack@<director-controlzone> ~]$ ./deploy.sh
UPDATE_COMPLETE
```

## Rebuilding Contrail Analytics And Analytics Database Nodes

This topic describes how to rebuild failed Contrail analytics and analytics database nodes. The same procedure is used for analytics nodes and analytics database nodes.



To rebuild Contrail analytics and analytics database nodes:

1. Verify that the failed node is ready to be redeployed. To be ready, the failed node must have a Power State power off and Maintenance False.

```
[stack@<director-controlzone> ~]$ ironic node-list --fields uuid name power_state maintenance
+-----+-----+-----+
+-----+
| UUID                                | Name                                | Power State |
Maintenance |
+-----+-----+-----+
+-----+
| 92242f30-7131-462a-bbe4-b9535d8f6b77 | contrail-analytics01            | power on    |
False      |
| 3555e9a6-eb0b-45ba-9bc0-51964cf3851f | contrail-analytics02            | power off   |
False      |
| a0adffdb-410f-471d-9647-9925d9b28ae2 | contrail-analytics03            | power on    |
False      |
+-----+-----+-----+
+-----+
```

2. In some cases, the power state might be None. If the power state is None, set it to off.

```
[stack@<director-controlzone> ~]$ ironic node-set-power-state <UUID of failed node> off
```

3. In some cases, the Maintenance mode might be set to True. If maintenance mode is True, set it to False.

```
[stack@<director-controlzone> ~]$ ironic node-set-maintenance <UUID of failed node> false
```

4. Rebuild the node, and wait for the node status to turn ACTIVE. Repeat the procedure for each node you need to replace.

```
[stack@director-computezone ~]$ source ~/stackrc
[stack@director-computezone ~]$ openstack baremetal configure boot
[stack@director-computezone ~]$ nova rebuild contrail-analytics-1 overcloud-compute
[stack@<director-controlzone> ~]$ openstack server list
+-----+-----+-----+
+-----+
| ID                                | Name                                | Status |
Networks                            | Image Name                            |
+-----+-----+-----+
```

```

+-----+-----+-----+
+-----+-----+
| 03af3384-3220-4b76-a0d0-b221d22a03a7 | contrail-analytics-0 | ACTIVE |
ctlplane=192.168.210.32 | overcloud-full |
| a905105b-efb5-4d11-bec3-23a7cd19655e | contrail-analytics-1 | ACTIVE |
ctlplane=192.168.210.31 | overcloud-full |
| 8a001ca2-0bf6-4662-ad81-252cdfccf09d | contrail-analytics-2 | ACTIVE |
ctlplane=192.168.210.26 | overcloud-full |
+-----+-----+-----+
+-----+-----+

```

## Finish Rebuilding the Analytics Nodes

Use this procedure to finish the rebuilding of the Contrail analytics or analytics database nodes that have been rebuilt.

1. Observe the status of the rebuild process. Nodes undergoing rebuilding will have Status of REBUILD. Wait until the status of all nodes being rebuilt turns ACTIVE.

```

[stack@<director-controlzone> ~]$ openstack server list
+-----+-----+-----+
+-----+-----+
| ID | Name | Status |
Networks | Image Name |
+-----+-----+
+-----+-----+
| 03af3384-3220-4b76-a0d0-b221d22a03a7 | contrail-analytics-0 | ACTIVE |
ctlplane=192.168.210.32 | overcloud-full |
| a905105b-efb5-4d11-bec3-23a7cd19655e | contrail-analytics-1 | REBUILD |
ctlplane=192.168.210.31 | overcloud-full |
| 8a001ca2-0bf6-4662-ad81-252cdfccf09d | contrail-analytics-2 | ACTIVE |
ctlplane=192.168.210.26 | overcloud-full |
+-----+-----+
+-----+-----+

```

2. Activate `os-collect-config` on the node.

```

[stack@<director-controlzone> ~]$ ssh heat-admin@19x.xxx.xxx.31 sudo systemctl start os-
collect-config

```

3. Establish an SSH connection to the node you have rebuilt and observe the journal of the `os-collect-config` process until you see multiple occurrences of the message `No local metadata found (['/var/lib/os-collect-config/local-data'])`. Repeat this step for each node being rebuilt.

```
[stack@<director-controlzone> ~]$ ssh heat-admin@19x.xxx.xxx.31 sudo journalctl -u os-collect-config -f
```

4. Perform a full stack update to reconverge the stack and bring the system back to operational state.

```
[stack@<director-controlzone> ~]$ source ~/stackrc  
[stack@<director-controlzone> ~]$ ./deploy.sh
```

## RELATED DOCUMENTATION

| [Backing Up Conrail Databases Using JSON Format | 1058](#)

# Using Server Manager to Automate Provisioning

## IN THIS CHAPTER

- [Installing Server Manager | 316](#)
- [Using Server Manager to Automate Provisioning | 323](#)
- [Using the Server Manager Web User Interface | 357](#)
- [Installing and Using Server Manager Lite | 381](#)

## Installing Server Manager

### IN THIS SECTION

- [Installation Requirements for Server Manager | 316](#)
- [Installing Server Manager | 318](#)
- [Upgrading Server Manager Software | 319](#)
- [Server Manager Installation Completion Checks | 320](#)
- [Sample Configurations for Server Manager Templates | 321](#)

### Installation Requirements for Server Manager

This document provides details for installing Server Manager.

#### Platform Support

As of Contrail 4.0, Server Manager can be installed on, and used to reimage and provision, the following platform operating systems:

- Ubuntu 16.04.01

- Ubuntu 16.04.02
- Ubuntu 14.04.5
- Ubuntu 14.04.4
- Ubuntu 14.04.2
- Ubuntu 14.04.1
- Ubuntu 14.04

As of Contrail 4.0, Server Manager installation supports Contrail provisioning for only the following OpenStack versions:

- Ocata, on Ubuntu 16.04 platform, only
- Newton, on Ubuntu 16.04 platform, only
- Mitaka
- Liberty
- Kilo, on Contrail networking only

## Installation Prerequisites

Before installing Server Manager ensure the following prerequisites are met.

- The system has Internet access to get dependent packages. Ensure access is available to the Ubuntu archive mirrors/repos at `/etc/apt/sources.list`.

**NOTE:** Server Manager is tested with only the following versions of dependent packages: Ansible 2.2.0.0, Docker 1.13.0, Puppet 3.7.3-1, and Cobbler 2.6.3-1. These tested versions are installed during Server Manager installation.

- Puppet Master requires the fully-qualified domain name (FQDN) of the Server Manager for key generation. The domain name is taken from the `/etc/hosts` file. If the server is part of multiple domains, specify the domain name by using the `--domain` option during the installation.
- On multi-interface systems, specify the interface on which Server Manager needs to listen by using the `--hostip` option during installation. If the listening interface is not specified, the first available interface from the `ifconfig` list is used.
- The system administrator might need to configure the Linux kernel security module AppArmor to allow `server-manager` access.

## Installing Server Manager

Server Manager and all of its components (Server Manager, monitoring, Server Manager client, Server Manager Web user interface) are provided together in a wrapper installation package:

Ubuntu: `contrail-server-manager-installer_<version-sku>.deb`

You can choose to install all components at once or install individual components one at a time.

Use the following steps to install and set up Server Manager and its components.

### 1. Install the Server Manager packages:

Ubuntu: `dpkg -i contrail-server-manager-installer_<version-sku>.deb`

**NOTE:** Make sure to select the correct version package that corresponds to the platform for which you are installing.

### 2. Set up the Server Manager components. Use the `setup.sh` command to install all of the components, or you can install individual components.

```
cd /opt/contrail/contrail_server_manager; ./setup.sh [--hostip=<ip address>] [--domain=<domain name>]
```

- To set up all components:

```
./setup.sh --all
```

- To set up only the Server Manager server:

```
./setup.sh --sm=contrail-server-manager_<version-sku>.deb
```

- To set up only the Server Manager client:

```
setup.sh --sm-client=contrail-server-manager_<version-sku>.deb
```

- To set up only the Server Manager user interface:

```
setup.sh --webui=contrail-server-manager_<version-sku>.deb
```

- To set up only Server Manager monitoring:

```
setup.sh --sm-mon=contrail-server-manager_<version-sku>.deb
```

Other options include:

- `--sm-cliff-client`
- `--nowebui`

- `--nosm-mon`

3. Installation logs are located at `/var/log/contrail/install_logs/`.

## Finishing the Installation

The Server Manager service does not start automatically upon successful installation. You must finish the installation by modifying the following templates. Refer to the sample configuration section included in this topic for details about configuring these files.

1. `/etc/cobbler/dhcp.template`
2. `/etc/cobbler/named.template`
3. `/etc/bind/named.conf.options`
4. `/etc/cobbler/settings`
5. `/etc/cobbler/modules.conf`
6. `/etc/mail/sendmail.cf`

## Starting the Server Manager Service

When you are finished modifying the templates to match your environment, start the Server Manager service using the following command:

```
service contrail-server-manager start
```

## Upgrading Server Manager Software

If you are upgrading Server Manager software from a previous version to the current version, use the following guidelines to ensure successful installation.

### Steps for Upgrading

Use the following steps to upgrade your Server Manager installation.

**NOTE:** You do not need to manually delete your previous Server Manager installation before upgrading.

1. `dpkg -i <contrail-server-manager-installer*deb>`

2. `cd /opt/contrail/contrail_server_manager`

3. `./setup.sh -all`

4. After the setup script has completed running, you can restart Server Manager by issuing:

```
service contrail-server-manager restart
```

It is not necessary to reconfigure the templates of DHCP, bind, and so on. Previous template configurations and configured data are preserved during the upgrade.

## Server Manager Installation Completion Checks

The following are various checks you can use to investigate the status of your Server Manager installation.

### Server Manager Checks

Use the following to check that the Server Manager installation is complete.

- Use the following commands to verify that the services are running:

```
service contrail-server-manager status
```

```
service cobblerd status
```

```
cobbler sync
```

```
service bind9 status
```

```
service isc-dhcp-server status
```

```
service apache2 status
```

```
service docker status
```

- Also verify processes using the following command:

```
ps auwx | grep Passenger
```

### Server Manager Client Checks

- Verify the items listed:

```
which server-manager
```

- Check the client configuration at `/etc/contrail/sm-client-config.ini`



- Make sure that `listen_ip_addr` is configured with the correct Server Manager IP address.

### Server Manager WebUI Checks

- Verify the status of the Server Manager WebUI:

```
service supervisor-webui-sm status
```

- Check the webui access from the browser:
  - Contrail release 4.0 and greater—`http: <server manager ip> :9143`
  - Contrail releases 3.0, 3.1, and 3.2—`http: <server manager ip> :9080`
  - Contrail release 2.2 and lower—`http: <server manager ip> :8080`

### Sample Configurations for Server Manager Templates

The following are sample parameters for the Server Manager templates. Use settings specific for your environment. Typically, you configure parameters for DHCP, bind, and e-mail services.

#### Sample Settings

```
bind_master: 10.XX.11.6

manage_forward_zones: ['contrail.juniper.net']

manage_reverse_zones: ['10.XX.11']

next_server: 10.XX.11.6

server: 10.XX.11.6
```

#### Sample dhcp.template File

Add Server Manager hooks into the `dhcp.template` file, so that when DHCP actions occur, such as commit, release, or expire, the Server Manager is notified. The DHCP servers are detected on the Server Manager and the *Discovered* status is maintained.

Use the following sample to help define the subnet blocks that the DHCP server needs to support:

<https://github.com/Juniper/contrail-server-manager/blob/master/src/cobbler/dhcp.template>

**NOTE:** Your DHCP template must have a separate block for each subnet for which Server Manager will be the DHCP server.

### Sample named.conf.options File

Use the following sample to help configure the `/etc/bind/named.conf.options`:

<https://github.com/Juniper/contrail-server-manager/blob/master/src/cobbler/named.conf.options.u>

You can also configure the following parameter:

```
forwarders {  
    0.0.0.0;  
};
```

### Sample named.template File

Use the following sample to help configure the `/etc/cobbler/named.template`:

<https://github.com/Juniper/contrail-server-manager/blob/master/src/cobbler/named.template>

### The sendmail.cf File

The `sendmail.cf` template is present with a `juniper.net` configuration. Populate it with configuration specific to your environment. The Server Manager uses the template to generate e-mails when reimaging or provisioning is completed.

## RELATED DOCUMENTATION

---

*Using Server Manager to Automate Provisioning*

---

*Using the Server Manager Web User Interface*

---

[Installing and Using Server Manager Lite](#)

## Using Server Manager to Automate Provisioning

### IN THIS SECTION

- [Overview of Server Manager | 323](#)
- [Server Manager Requirements and Assumptions | 324](#)
- [Server Manager Component Interactions | 325](#)
- [Configuring Server Manager | 326](#)
- [Configuring the Cobbler DHCP Template | 328](#)
- [User-Defined Tags for Server Manager | 329](#)
- [Server Manager Client Configuration File | 329](#)
- [Restart Services | 330](#)
- [Accessing Server Manager | 330](#)
- [Communicating with the Server Manager Client | 331](#)
- [Server Manager Commands for Configuring Servers | 332](#)
- [Server Manager REST API Calls | 349](#)
- [Example: Reimaging and Provisioning a Server | 355](#)

### Overview of Server Manager

The Contrail Server Manager is used to provision, configure, and reimage a Contrail virtual network system of servers, clusters, and nodes.

This section describes the functions and usage guidelines for the Contrail Server Manager.

The Server Manager provides a simple, centralized way for users to manage and configure components of a virtual network system running across multiple physical and virtual servers in a cloud infrastructure.

You can use Server Manager to configure, provision, and reimage servers with the correct software version and packages for the nodes that are running on each server in multiple virtual network system clusters.

The Server Manager:

- Provides REST APIs to handle customer requests.
- Manages its own database to store information about the servers.

- Interacts with other open source products such as Cobbler, Puppet, and Ansible to configure servers based on user requests.

## Server Manager Requirements and Assumptions

The following are requirements and assumptions for the Server Manager:

- The Server Manager runs on a Linux server (bare metal or virtual machine) and assumes availability of several software products with which it interacts to provide the functionality of managing servers.
- The Server Manager has network connectivity to the servers it is trying to manage.
- The Server Manager has access to a remote power management tool to power cycle the servers that it manages.
- The Server Manager uses Cobbler software for Linux provisioning to configure and download software to physical servers. Cobbler resides on the same server that is running the Server Manager daemon.
  - Server Manager assumes that DNS and DHCP servers embedded with Cobbler provide IP addresses and names to the servers being managed, although it is possible to use external DNS and DHCP servers.
- The Server Manager uses Puppet software, an open source configuration management tool, to accomplish the configuration management of target servers, including the installation and configuration of different software packages and the launching of various services.
- Starting with Contrail Release 4.0, Server Manager uses Ansible software, an open source configuration management tool primarily used to automate the configuration and provisioning of Contrail components inside containers.
- The Server Manager also uses Docker to load and move these Contrail containers to the target servers. The Server Manager maintains a local registry on the Server Manager machine and users also have an option to use an external registry from which they can copy their Contrail Docker images directly onto the target servers.
- SQLite3 database management software is used to maintain and manage server configurations and it runs on the same machine where the Server Manager daemon is running.
- Because the server-manager process listens on port 9001, and the server-manager webui listens on ports 9080 and 9143, the firewall must be enabled for those ports.
- Server Manager needs a minimum of 4GB of RAM, 2 CPU cores, and 80GB of disks (to support multiple Contrail installations).
- Server Manager assumes that SSH is enabled on target nodes.

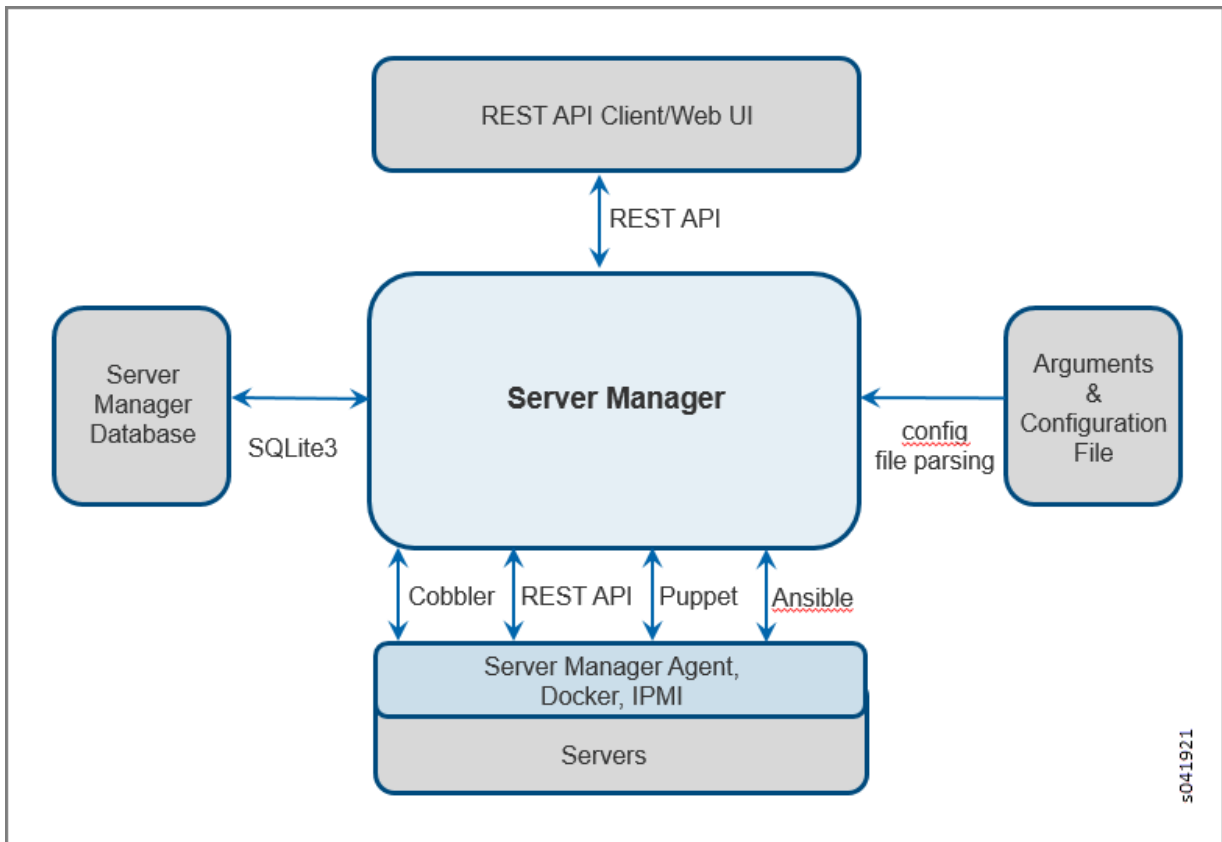
## Server Manager Component Interactions

The Server Manager runs as a daemon and provides REST APIs for interaction with the client. The Server Manager accepts user input in the form of REST API requests, performs the requested function on the resources, and responds with a REST API response.

Configuration parameters required by the Server Manager are provided in the Server Manager configuration file. However, the parameters can be overridden by Server Manager command line parameters.

Figure 36 on page 325 illustrates several high-level components with which the Server Manager interacts.

Figure 36: Server Manager Component Interactions



Internally, the Server Manager uses a SQLite3 database to hold server configuration information. The Server Manager coordinates the database configuration information and user requests to manage the servers defined in the database.

While managing the servers, the Server Manager also communicates with other software components. It uses Cobbler for reimaging target servers, Docker to host Contrail containers, and Ansible and Puppet

for provisioning, thereby ensuring necessary software packages are installed and configured, required services are running, and so on.

A Server Manager agent runs on each of the servers and communicates with the Server Manager, providing the information needed to monitor the operation of the servers. The Server Manager agent also uses REST APIs to communicate with the Server Manager, and it can use other software tools to fetch other information, such as Intelligent Platform Interface (IPMI). Monitoring functionality is enabled by default with Server Manager installation but can be skipped if the user wishes.

## Configuring Server Manager

When the installation of all Server Manager components and dependent packages is finished, configure the Server Manager with parameters that identify your environment and make it available for clients to serve REST API requests.

Upon installation, a sample Server Manager configuration file is created at:

```
/opt/contrail/server_manager/sm-config.ini
```

Modify the `sm-config.ini` configuration file to include parameter values specific to your environment.

The environment-specific configuration section of the `sm-config.ini` file is named `SERVER-MANAGER`.

The following example shows the format and parameters of the `SERVER-MANAGER` section. Typically, only the `listen_ip_addr`, `cobbler_username`, and `cobbler_passwd` values need to be modified.

```
[SERVER-MANAGER]

listen_ip_addr = <SM-IP-address>

listen_port    = <port-number>

cobbler_ip_address = <cobbler-ip-address>

cobbler_port    = <cobbler-port-number>

cobbler_username = <cobbler-username>

cobbler_password = <cobbler-password>

ipmi_username = <IPMI username>

ipmi_password = <IPMI password>
```

```
ipmi_type = <IPMI type>
```

Table 9 on page 327 provides details for each of the parameters in the SERVER-MANAGER section.

**Table 9: Server Manager Parameters**

| Parameter          | Configuration  |
|--------------------|--|
| listen_ip_addr     | Specify the IP address of the server on which the Server Manager is listening for REST API requests.   |
| listen_port        | The port number on which the Server Manager is listening for REST API requests. The default is 9001.   |
| cobbler_ip_address | The IP address used to access Cobbler. This address <b>MUST</b> be the same address as the listen_ip_address. The Server Manager assumes that the Cobbler service is running on the same server as the Server Manager service. |
| cobbler_port       | The port on which Cobbler listens for user requests. Leave this field blank.   |
| cobbler_username   | Specify the user name to access the Cobbler service. Specify testing unless your Cobbler settings have been modified to use a different user name.   |
| cobbler_password   | Specify the password to access the Cobbler service. Specify testing unless your Cobbler settings have been modified to use a different password.   |
| ipmi_username      | The IPMI username for power management.  |
| ipmi_password      | The IPMI password for power management.  |
| ipmi_type          | The IPMI type (ipmilan, lanplus, or other Cobbler-supported types).  |

Starting with Contrail Release 4.0, there is an ANSIBLE-SERVER section for parameters for running the Server Manager Ansible daemon, which is used to set up a Docker registry. This registry is used by Ansible to provision Contrail Release 4.0 containers onto targets. These values can be modified to reflect any remote or non-Server Manager Docker registry that the user wants to use to host the Contrail Release

4.0 Docker containers. The following example shows the format and parameters of the ANSIBLE-SERVER section:

```
[ANSIBLE-SERVER]

docker_insecure_registries = <IP address:Port>

docker_registry = <IP address:Port>

ansible_srvr_ip = <IP address>

ansible_srvr_port = <Port>

ansible_log_path = /var/log/contrail-server-manager/debug.log
```

[Table 10 on page 328](#) provides details for each of the parameters in the ANSIBLE-SERVER section.

**Table 10: Ansible Server Parameters**

| Parameter                  | Configuration  |
|----------------------------|--|
| docker_insecure_registries | Specify the IP address and port of the server on which the insecure Docker registry used by the Server Manager resides |
| docker_registry            | Specify the IP address and port of the server on which the Docker registry used by the Server Manager resides          |
| ansible_srvr_ip            | Specify the IP address of the Server Manager machine on which the Ansible daemon will run                              |
| ansible_srvr_port          | Specify the port on the Server Manager machine on which the Ansible daemon will run                                    |
| ansible_log_path           | Specify the log path where the Ansible daemon stores its log messages  |

## Configuring the Cobbler DHCP Template

In addition to configuring the `sm_config.ini` file, you must manually change the settings in the `/etc/cobbler/dhcp.template` file to use the correct subnet address, mask, and DNS domain name for your environment.



Optionally, you can also restrict the use of the current instance of Server Manager and Cobbler to a subset of servers in the network.

The following is a link to a sample `dhcp.template` file, which you can modify to match the subnets in your setup.

**NOTE:** The IP addresses and other values in the sample are for example purposes only. Be sure to use values that are correct for your environment.

*Sample dhcp.template*

<https://github.com/Juniper/contrail-server-manager/blob/master/src/cobbler/dhcp.template.u.sample>

## User-Defined Tags for Server Manager

Server Manager enables you to define tags that can be used to group servers for performing a particular operation, such as show information, reimage, provision, and so on. Server Manager supports up to seven different tags that can be configured and used for grouping servers.

The names of user-defined tags are kept in the `tags.ini` file, at `/etc/contrail_smgr/tags.ini`.

It is possible to modify tag names, and add or remove tags dynamically using the Server Manager REST API interface. However, if a tag is already being used to group servers, the tag must be removed from the servers before tag modification is allowed.

The following is a sample `tags.ini` file that is copied on installation. In the sample file, five tags are defined – `datacenter`, `floor`, `hall`, `rack`, and `user_tag`. Use the tags to group servers together.

```
[TAGS]
tag1 = datacenter
tag2 = floor
tag3 = hall
tag4 = rack
tag5 = user_tag
```

## Server Manager Client Configuration File

The Server Manager client application installation copies the `/etc/contrail/sm-client-config.ini` sample configuration file. The sample file contains parameter values such as the IP address to reach the Server Manager and the port used by Server Manager. You must modify the values in the `sm-client-config.ini` file to match your environment.

The `CLUSTER` and `SERVER` subsections have fields that represent the password for a host or a service. If a value for the password field is not explicitly provided, the Server Manager selects a default password.

Starting with Contrail Release 3.0.2, if you don't explicitly specify a password, a password is automatically generated by the system. This makes the clusters provisioned by Server Manager more secure. There are no default passwords. The system administrator can specify the passwords to configure, or you can use the passwords that are automatically generated by Server Manager.

The following fields get an autogenerated password whenever an explicit password is not provided.

- Ceilometer MongoDB password
- Ceilometer keystone auth password
- Cinder keystone auth password
- Glance keystone auth password
- Heat encryption key
- Heat keystone auth password
- Keystone admin password
- Keystone admin token
- MYSQL root password
- MYSQL service password
- Neutron keystone auth password
- Nova keystone auth password
- Swift keystone auth password

## Restart Services

When all user changes have been made to the configuration files, restart the Server Manager so that it runs with the modifications:

```
service contrail-server-manager restart
```

## Accessing Server Manager

When the Server Manager configuration has been customized to your environment, and the required daemon services are running, clients can request and use services of the Server Manager by using REST APIs. Any standard REST API client can be used to construct and send REST API requests and process Server Manager responses.

The following steps are typically required to fully implement a new cluster of servers being managed by the Server Manager.

1. Add a boot image (ISO) to server-manager, along with the kickstart and preseed files compatible with your datacenter server. Each Server Manager release has a default kickstart file. If your system administrator doesn't provide the kickstart files, Server Manager default files will be used.
2. Add the Contrail image you are using to Server Manager.
3. Add the cluster(s) to Server Manager. You can add common provisioning parameters for servers to the cluster, and the parameters get passed to the server when provisioning starts.
4. Add the servers that will be managed by Server Manager. Remember to add the `cluster_id` to link with the cluster.

The following are the minimum parameters needed for reimaging or provisioning:

- ID
  - cluster
  - domain
  - interface details
  - roles assigned to each server
  - password
5. Specify the name and location of boot images, packages, and repositories used to bring up the servers with needed software of the supported versions.
  6. Provision or configure the servers by installing necessary packages, creating configuration files, and bringing up the correct services so that each server can perform the functions or role(s) configured for that server.

A Contrail system of servers has several components or roles that work together to provide the functionality of the virtual network system, including: control, config, analytics, compute, web-ui, OpenStack, and database. Each of the roles has different requirements for the software and services needed. The provisioning REST API enables the client to configure the roles on servers using the Server Manager.

7. Set up API calls for monitoring servers.

Once the servers in the Contrail system are correctly reimaged and provisioned to run configured roles, the server monitoring REST API calls allow clients to monitor performance of the servers as they provide one or more role functions.

## Communicating with the Server Manager Client

Server Manager provides a REST API interface for clients to talk to the Server Manager software. Any client that can send and receive REST API requests and responses can be used to communicate with Server Manager, for example, Curl or Postman. Additionally, the Server Manager software provides a

client with a simplified CLI interface, in a separate package. The Server Manager client can be installed and run on the Server Manager machine itself or on another server with an IP connection to the Server Manager machine.

Prior to using the Server Manager client CLI commands, you need to modify the `sm-client-config.ini` file to specify the IP address and the port for the Server Manager.

Each of the commands described in this section takes a set of parameters you specify, constructs a REST API request to the Server Manager, and provides the server's response.

The following describes each Server Manager client CLI command in detail.

## Server Manager Commands for Configuring Servers

### IN THIS SECTION

- [Server Manager Commands Common Options | 332](#)
- [Add New Servers or Update Existing Servers | 333](#)
- [Delete Servers | 334](#)
- [Display Server Configuration | 335](#)
- [Server Manager Commands for Managing Clusters | 336](#)
- [Server Manager Commands for Managing Tags | 338](#)
- [Server Manager Commands for Managing Images | 340](#)
- [Server Manager Operational Commands for Managing Servers | 344](#)
- [Reimaging Server\(s\) | 344](#)
- [Provisioning and Configuring Roles on Servers | 346](#)
- [Restarting Server\(s\) | 347](#)
- [Show Status of Server\(s\) | 348](#)
- [Show Status of Provision | 349](#)

This section describes commands that are used to configure servers and server parameters in the Server Manager database. These commands allow you to add, modify, delete, or view servers, clusters, images, and tags.

### Server Manager Commands Common Options

The common options in [Table 11 on page 333](#) are available with every Server Manager command.

**Table 11: Common Command Options**

| Option                                    | Description   |
|---|---|
| -h, --help                                | Show the options available for the current command and exit.  |
| --config_file CONFIG_FILE, -c CONFIG_FILE | The name of the Server Manager client configuration file. The default file is /etc/contrail/sm-client-config.ini. |
| --smgr_ip SMGR_IP                         | The IP address of the Server Manager process if different from that specified in the config file.                 |
| --smgr_port SMGR_PORT                     | The port that the Server Manager process is listening on if different from that in the config file.               |

### Add New Servers or Update Existing Servers

Use the `server-manager add` command to create a new server or update a server in the Server Manager database.

```
server-manager [-h] [--smgr_ip SMGR_IP] [--smgr_port SMGR_PORT]
[--config_file CONFIG_FILE] add server [-f FILE_NAME]
```

Table 12 on page 333 lists the optional arguments.

**Table 12: Server Manager Add Server Command Options**

| Option                              | Description  |
|-------------------------------------|--|
| --file_name FILE_NAME, -f FILE_NAME | The JSON file that contains the server parameter values. |

The JSON file contains a number of server entries, in the format shown in the following example:

<https://github.com/Juniper/contrail-server-manager/blob/R3.1/src/client/new-server.json>

Most of the parameters in the JSON sample file are self-explanatory. `Cluster_id` defines the cluster to which the server belongs. The sample `roles` array in the example lists all valid role values. `Tag` defines the mapping of tag names and values for grouping and classifying the server.

The `server-manager add` command will add a new entry if the server with the given ID or `mac_address` does not exist in the Server Manager database. If an entry already exists, the add command modifies the fields in the existing entry with any new parameters specified.

**NOTE:** It is not possible to re-add an existing MAC address under a new server, even if the ID and IP address of that new server are unique.

## Delete Servers

Use the `server-manager delete` command to delete one or more servers from the Server Manager database.

```
server-manager [-h] [--smgr_ip SMGR_IP] [--smgr_port SMGR_PORT][--config_file CONFIG_FILE]
delete server (--server_id SERVER_ID | --mac MAC | --ip IP | --cluster_id CLUSTER_ID | --tag
<tag_name=tag_value>.. )
```

Table 13 on page 334 lists the optional arguments.

**Table 13: Server Manager Delete Server Command Options**

| Option                               | Description  |
|--------------------------------------|--|
| <code>--server_id SERVER_ID</code>   | The server ID for the server or servers to be deleted.   |
| <code>--mac MAC</code>               | The MAC address for the server or servers to be deleted.   |
| <code>--ip IP</code>                 | The IP address for the server or servers to be deleted.  |
| <code>--cluster_id CLUSTER_ID</code> | The cluster ID for the server or servers to be deleted.  |
| <code>--tag TagName=TagValue</code>  | The TagName that is to be matched with the Tagvalue. Up to seven TagName and Tagvalue pairs separated by commas can be provided. |

The criteria for identifying servers to be deleted can be specified by providing the `server_id` or the `server:mac` address, `ip`, `cluster_id`, or the `TagName = TagValue`.

Provide one of the server matching criteria to display a list of servers available to be deleted.

## Display Server Configuration

Use the `server-manager display` command to display the configuration of servers from the Server Manager database.

```
server-manager display [--smgr_ip SMGR_IP] [--smgr_port SMGR_PORT][--config_file CONFIG_FILE]
    server (--server_id SERVER_ID | --mac MAC | --ip IP | --cluster_id
    CLUSTER_ID | --tag <tag_name=tag_value>.. ) [--detail]
```

Table 14 on page 335 lists the optional arguments.

**Table 14: Server Manager Display Server Command Options**

| Option                               | Description  |
|--------------------------------------|--|
| <code>--server_id SERVER_ID</code>   | The server ID for the server or servers to be deleted.   |
| <code>--mac MAC</code>               | The MAC address for the server or servers to be displayed.   |
| <code>--ip IP</code>                 | The IP address for the server or servers to be displayed.  |
| <code>--cluster_id CLUSTER_ID</code> | The cluster ID for the server or servers to be displayed.  |
| <code>--tag TagName=TagValue</code>  | The TagName that is to be matched with the Tagvalue. Up to seven TagName and Tagvalue pairs separated by commas can be provided. |
| <code>--detail, -d</code>            | Flag to indicate if details are requested.   |

The criteria for identifying servers to be displayed can be specified by providing the `server_id` or one of the following server parameters: `mac` address, `ip`, `cluster_id`, or `TagName=TagValue`.

Provide one or more of the server matching criteria to display a list of servers.

## Server Manager Commands for Managing Clusters

### IN THIS SECTION

- [Create a New Cluster or Update an Existing Cluster | 336](#)
- [Delete a Cluster | 337](#)
- [Display Cluster Configuration | 337](#)

A cluster is used to store parameter values that are common to all servers belonging to that cluster. The commands in this section facilitate managing clusters in the Server Manager database, enabling you to add, modify, delete, and view clusters.

**NOTE:** Whenever a server is created with a specific `cluster_id`, Server Manager checks to see if a cluster with that ID has already been created. If there is no matching `cluster_id` already in the database, an error is returned.

### *Create a New Cluster or Update an Existing Cluster*

Use the `server-manager add cluster` command to create a new cluster or update an existing cluster in the Server Manager database.

```
server-manager add cluster [--file_name FILE_NAME]
```

[Table 15 on page 336](#) lists the optional arguments.

**Table 15: Server Manager Add Cluster Command Options**

| Option   | Description   |
|--|---|
| <code>--file_name FILE_NAME, -f FILE_NAME</code> | The JSON file that contains the cluster parameter values. |

The JSON file contains a number of cluster entries, in the format shown in the following example:

<https://github.com/Juniper/contrail-server-manager/blob/master/src/client/new-cluster-contrail-4.x.json>

Server membership to a cluster is determined by specifying the ID corresponding to the cluster when defining the server. All of the cluster parameters are available to the server when provisioning roles on the server.



### Delete a Cluster

Use the `server-manager delete cluster` command to delete a cluster from the Server Manager database that are no longer needed. Use this command after all servers in the cluster have been deleted.

**NOTE:** A cluster can only be deleted if no servers are attached to it. If any servers are attached, deletion will fail.

```
server-manager delete cluster [--cluster_id CLUSTER_ID]
```

[Table 16 on page 337](#) lists the optional arguments.

**Table 16: Server Manager Delete Cluster Command Options**

| Option                               | Description   |
|--------------------------------------|---|
| <code>--cluster_id CLUSTER_ID</code> | The cluster ID for the server or servers to be displayed. |

### Display Cluster Configuration

Use the `server-manager display cluster` command to list the configuration of a cluster.

```
server-manager display cluster [--cluster_id CLUSTER_ID] [--detail]
```

[Table 17 on page 337](#) lists the optional arguments.

**Table 17: Server Manager Display Cluster Command Options**

| Option                               | Description                                 |
|--------------------------------------|---|
| <code>--detail, -d</code>            | Flag to indicate if details are requested.  |
| <code>--cluster_id CLUSTER_ID</code> | The cluster ID for the cluster or clusters. |

You can optionally specify a cluster ID to get server information about a particular cluster. If the optional parameter is not specified, server information about all clusters in the system is returned.

## Server Manager Commands for Managing Tags

### IN THIS SECTION

- [Create a New Tag or Update an Existing Tag | 338](#)
- [Display Tag Configuration | 339](#)

Tags are used for grouping servers together so that an operation such as show, reimage, provision, status, and so on can be easily performed on servers that have matching tags. The Server Manager provides a flexible way for you to define your own tags, and then use those tags to assign values to servers. Servers with matching tag values can be easily grouped together. The Server Manager can store a maximum of seven tag values. At initialization, the Server Manager reads the tag names from the configuration file. The tag names can be retrieved or modified using CLI commands. When modifying tag names, the Server Manager ensures that the tag name being modified is not used by any of the server entries.

### *Create a New Tag or Update an Existing Tag*

Use the `server-manager add` command to create a new tag or update an existing tag in the Server Manager database.

```
server-manager add tag [--file_name FILE_NAME] [--tags TAG_LIST]
```

[Table 18 on page 338](#) lists the optional arguments.

**Table 18: Server Manager Add New Tag**

| Option   | Description   |
|--|---|
| <code>--file_name FILE_NAME, -f FILE_NAME</code> | The JSON file that contains the tag names.  |
| <code>--tags TAG_LIST</code>                     | Comma separated list of tag number and tag. For example: <code>tag0=abc,tag1=xyz</code> |

The sample JSON file contains a number of tag entries, in the format shown in the following example:

```
{
```

```
"tag1" : "data-center",  
  
"tag2" : "floor",  
  
"tag3" : "",  
  
"tag4" : "pod",  
  
"tag5" : "rack",  
  
}
```

In the example, you specify a JSON file to add or modify the tags, tag1 through tag5. For tag3, the "" value specifies that if the tag is defined prior to the CLI command, it is removed on execution of the command. The tag name for tag1 is set to data-center. This is allowed if, and only if, none of the server entries are using tag1.

### ***Display Tag Configuration***

Use the `server-manager display tag` command to list the configuration of a tag.

```
server-manager display tag
```

The following is sample output for the `display tag` command.

```
{  
  
  "tag1": "datacenter",  
  
  "tag2": "floor",  
  
  "tag3": "hall",  
  
  "tag4": "rack",  
  
  "tag5": "user_tag"  
  
}
```

## Server Manager Commands for Managing Images

### IN THIS SECTION

- [Creating New Images or Updating Existing Images | 341](#)
- [Add an Image | 341](#)
- [Upload an Image | 342](#)
- [Delete an Image | 343](#)
- [Display Image Configuration | 344](#)

In addition to servers and clusters, the Server Manager also manages information about images and packages that can be used to reimage and configure servers. Images and packages are both stored in the database as images. When new images are added to the database, or existing images are deleted, the Server Manager interfaces with Cobbler to make corresponding modifications in the Cobbler distribution profile for the specified image.

[Table 19 on page 340](#) lists the image types supported.

**Table 19: Server Manager Image Types**

| Image Type              | Description   |
|-------------------------|---|
| centos                  | Manages the CentOS ISO base.  |
| contrail-centos-package | Maintains a repository of the package to be installed on the CentOS system image.   |
| ubuntu                  | Manages the base Ubuntu ISO.  |
| contrail-ubuntu-package | Maintains a repository of packages that contain Contrail and dependent packages to be installed on an Ubuntu base system. |
| ESXi5.1/ESXi5.5         | Manages VMware ESXi 5.1 or 5.5 ISO.   |

### *Creating New Images or Updating Existing Images*

The Server Manager maintains four types of images – CentOS ISO, Ubuntu ISO, Contrail CentOS package, and Contrail Ubuntu package.

Use the `server-manager add` command or the `server-manager upload` command to add new images to the Server Manager database.

- Use `add` when the new image is present locally on the Server Manager machine. The path provided is the image path on the Server Manager machine.
- Use `upload_image` when the new image is present on the machine where the client program is being invoked. The path provided is the image path on the client machine.

### *Add an Image*

```
server-manager add image [--file_name FILE_NAME]
```

[Table 20 on page 341](#) lists the optional arguments.

**Table 20: Server Manager Add Image**

| Option   | Description   |
|--|---|
| <code>--file_name FILE_NAME, -f FILE_NAME</code> | The name of the JSON file that contains the image parameter values. |

The JSON file contains an array of possible entries, in the following sample format. The sample shows three images: one CentOS ISO containing Contrail packages, one Ubuntu base ISO, and one Contrail Ubuntu package. When the images are added, corresponding distribution, profile, and repository entries are created in Cobbler by the Server Manager.

**NOTE:** Release numbers are represented in the sample with `<x.xx>`. Be sure to use the correct release numbers for your image versions.

```
{
  "image": [
    {
```

```

    "id": "ubuntu-<x.xx.x>",

    "type": "ubuntu",

    "version": "ubuntu-<x.xx.x>",

    "path": "/iso/ubuntu-<x.xx.x>-server-amd64.iso"

  },

  {

    "id": "centos-<x.xx>",

    "type": "centos",

    "version": "centos-<x.xx>",

    "path": "/iso/CentOS-<x.xx>-x86_64-minimal.iso"

  },

  {

    "id": "contrail-ubuntu-<x.xx>",

    "type": "contrail-ubuntu-package",

    "version": "contrail-ubuntu-<x.xx>",

    "path": "/iso/contrail-cloud-docker-<x.xx-xx>-all.deb"

  }

]

}

```

### ***Upload an Image***

The `server-manager upload_image` command is similar to the `server-manager add` command, except that the path provided for the image being added is the local path on the client machine. This command is useful

if the client is being run remotely, not on the Server Manager machine, and the image being added is not physically present on the Server Manager machine.

```
server-manager upload_image image_id image_version image_type file_name
```

Table 21 on page 343 lists the optional arguments.

**Table 21: Server Manager Upload Image**

| Option        | Description   |
|---------------|---|
| image_id      | Name of the new image.  |
| image_version | Version number of the new image.  |
| image_type    | Type of image: fedora, centos, ubuntu, contrail-ubuntu-package, contrail-centos-package |
| file_name     | Complete path for the file.   |

### **Delete an Image**

Use the `server-manager delete image --image_id <image_id>` command to delete an image from the Server Manager database. When an image is deleted from the Server Manager database, the corresponding distribution, profile, or repository for the image is also deleted from the Cobbler database.

```
server-manager delete image --image_id <image_id>
```

Table 22 on page 343 lists the optional arguments.

**Table 22: Server Manager Delete Image**

| Option   | Description                               |
|----------|---|
| image_id | The image ID for the image to be deleted. |

### Display Image Configuration

Use the `server-manager display` command to list the configuration of images from the Server Manager database. If the detail flag is specified, detailed information about the image is returned. If the optional `image_id` is not specified, information about all the images is returned.

```
server-manager display image [--image_id IMAGE_ID] [--detail]
```

Table 23 on page 344 lists the optional arguments.

**Table 23: Server Manager Display Image Configuration**

| Option                    | Description                                |
|---------------------------|--|
| <code>image_id</code>     | The image ID for the image or images.      |
| <code>--detail, -d</code> | Flag to indicate if details are requested. |

### Server Manager Operational Commands for Managing Servers

The Server Manager commands in the following sections are operational commands for performing a specific operation on a server or a group of servers. These commands assume that the base configuration of entities required to execute the operational commands is already completed using configuration CLI commands.

#### Reimaging Server(s)

Use the `server-manager reimage` command to reimage a server or servers with a provided base ISO and package. Servers are specified by providing match conditions to select them from the database.

Before issuing the `reimage` command, the images must be added to the Server Manager, which also adds the images to Cobbler. The set of servers to be reimaged can be specified by providing match criteria for servers already added to the Server Manager database, using `server_id`.

You must identify the base image ID to be used to reimage.



The command prompts for a confirmation before making the REST API call to the Server Manager to start reimaging the servers. This confirmation message can be bypassed by specifying the optional `--no_confirm` or `-F` parameter on the command line.

```
server-manager reimage
  [--package_image_id PACKAGE_IMAGE_ID]

  [--no_reboot]

  (--server_id SERVER_ID | --cluster_id CLUSTER_ID | --tag <tag_name=tag_value>)

  [--no_confirm]
  base_image_id
```

Options include the following:

[Table 24 on page 345](#) lists the optional arguments.

**Table 24: Server Manager Reimage**

| Option   | Description  |
|--|--|
| base_image_id  | The image ID of the base image to be used.   |
| --package_image_id PACKAGE_IMAGE_ID, -p PACKAGE_IMAGE_ID | The optional Contrail package to be used to reimage the server or servers.                         |
| --no_reboot, -n  | Optional parameter to indicate that the server should not be rebooted following the reimage setup. |
| --server_id SERVER_ID                                    | The server ID for the server or servers to be reimaged.  |
| --cluster_id CLUSTER_ID                                  | The cluster ID for the server or servers to be reimaged.   |
| --tag TagName=TagValue                                   | TagName which is to be matched with Tagvalue   |
| --no_confirm, -F   | Flag to bypass confirmation message, default = do NOT bypass.                                      |

## Provisioning and Configuring Roles on Servers

Use the `server-manager provision` command to provision identified server(s) with configured roles for the virtual network system. The servers can be selected from the database configuration (using standard server match criteria), identified in a JSON file, or provided interactively.

From the configuration of servers in the database, the Server Manager determines which roles to configure on which servers and uses this information along with other parameters from the database to achieve the task of configuring the servers with specific roles.

When the `server-manager provision` command is used, the Server Manager pushes the specified server configurations to the servers.

```
server-manager provision
  (--server_id SERVER_ID | --cluster_id CLUSTER_ID | --tag <tag_name=tag_value> )
  [--no_confirm]
  package_image_id
```

Options include the following:

[Table 25 on page 346](#) lists the optional arguments.

**Table 25: Server Manager Provision**

| Option   | Description  |
|--|--|
| <code>package_image_id</code>  | The Contrail package image ID to be used for provisioning.                   |
| <code>--server_id SERVER_ID</code>   | The server ID for the server or servers to be provisioned.                   |
| <code>--cluster_id CLUSTER_ID</code>   | The cluster ID for the server or servers to be provisioned.                  |
| <code>--tag TagName=TagValue</code>  | TagName to be matched with Tagvalue.   |
| <code>--provision_params_file PROVISION_PARAMS_FILE, -f PROVISION_PARAMS_FILE</code> | Optional JSON file containing the parameters for provisioning the server(s). |
| <code>--no_confirm, -F</code>  | Flag to bypass confirmation message, default = do NOT bypass.                |

**NOTE:** Adding and deleting roles is not supported in Contrail Release 4.0.

## Restarting Server(s)

Use the `server-manager restart` command to reboot identified server(s). Servers can be specified from the database by providing standard match conditions. The restart command provides a way to reboot or power-cycle the servers, using the Server Manager REST API interface. If reimaging is intended, use the restart command with the `net-boot` flag enabled. When netbooted, the Puppet agent is also installed and configured on the servers. If there are Puppet manifest files created for the server prior to rebooting, the agent pulls those from the Server Manager and executes the configured Puppet manifests. The restart command uses an IPMI mechanism to power cycle the servers, if available and configured. Otherwise, the restart command uses SSH to the server and the existing reboot command mechanism is used.

```
server-manager restart
    (--server_id SERVER_ID | --cluster_id CLUSTER_ID | --tag <tag_name=tag_value>)

    [--net_boot]

    [--no_confirm]
```

[Table 26 on page 347](#) lists the optional arguments.

**Table 26: Server Manager Restart**

| Option                               | Description   |
|--------------------------------------|---|
| <code>--server_id SERVER_ID</code>   | The server ID for the server or servers to be restarted.          |
| <code>--cluster_id CLUSTER_ID</code> | The cluster ID for the server or servers to be restarted.         |
| <code>--tag TagName=TagValue</code>  | TagName to be matched with Tagvalue.                              |
| <code>--net_boot, -n</code>          | Optional parameter to indicate if the server should be netbooted. |

**Table 26: Server Manager Restart (Continued)**

| Option           | Description   |
|------------------|---|
| --no_confirm, -F | Flag to bypass confirmation message, default = do NOT bypass. |

**Show Status of Server(s)**

Use the `server-manager status` command to view the reimaging or provisioning status of server(s).

```
server-manager status server (--server_id SERVER_ID | --cluster_id CLUSTER_ID | --tag
<tag_name=tag_value>)
```

[Table 27 on page 348](#) lists the optional arguments.

**Table 27: Server Manager Status Server**

| Option                | Description   |
|-----------------------|---|
| --server_id SERVER_ID | The server ID for the server whose status is to be fetched. |

**Table 27: Server Manager Status Server (Continued)**

|                         |   |
|-------------------------|---|
| --cluster_id CLUSTER_ID | The cluster ID for the server or servers to be restarted. |
| --tag TagName=TagValue  | TagName to be matched with Tagvalue.                      |

The status command provides a way to fetch the current status of a server.

Status outputs include the following:

1. restart\_issued
  - reimage\_started
  - provision\_issued
  - provision\_completed
  - openstack\_started

```
openstack_completed
```

## Show Status of Provision

Use the `server-manager status provision` to view the detailed provisioning status of servers or cluster. The `status` command provides a way to fetch the current status of a provision command.

```
server-manager status provision (--server_id SERVER_ID | --cluster_id CLUSTER_ID | --tag
<tag_name=tag_value>)
```

Table 28 on page 349 lists the optional arguments.

**Table 28: Server Manager Status Provision**

| Option                             | Description   |
|------------------------------------|---|
| <code>--server_id SERVER_ID</code> | The server ID for the server whose status is to be fetched. |

**Table 28: Server Manager Status Provision (Continued)**

|                                      |   |
|--------------------------------------|---|
| <code>--cluster_id CLUSTER_ID</code> | The cluster ID for the server or servers to be restarted. |
| <code>--tag TagName=TagValue</code>  | TagName to be matched with Tagvalue.                      |

## Server Manager REST API Calls

### IN THIS SECTION

- REST APIs for Server Manager Configuration Database Entries | 350
- API: Add a Server | 350
- API: Delete Servers | 350
- API: Retrieve Server Configuration | 351
- API: Add an Image | 351
- API: Upload an Image | 352
- API: Get Image Information | 352
- API: Delete an Image | 352

- API: Add or Modify a Cluster | 353
- API: Delete a Cluster | 353
- API: Get Cluster Configuration | 353
- API: Get All Server Manager Configurations | 354
- API: Reimage Servers | 354
- API: Provision Servers | 354
- API: Restart Servers | 355

This section describes all of the REST API calls to the Server Manager. Each description includes an example configuration.

### REST APIs for Server Manager Configuration Database Entries

The REST API calls in this section help in configuring different elements in the Server Manager database.

**NOTE:** The IP addresses and other values in the following are shown for example purposes only. Be sure to use values that are correct for your environment.

#### API: Add a Server

To add a new server to the service manager configuration database:

URL: `http://<SM-IP-Address>:<SM-Port>/server`

Method: PUT

Payload: JSON payload containing an array of servers to be added. For each server in the array, all the parameters are specified as JSON fields. The mask, gateway, password, and domain fields are optional, and if not specified, the values of these fields are taken from the cluster to which the server belongs.

The following is a sample JSON file for adding a server in Contrail Release 4.0.

<https://github.com/Juniper/contrail-server-manager/blob/master/src/client/new-server-contrail-4.x.json>

#### API: Delete Servers

Use one of the following formats to delete a server.

URL: `http://<SM-IP-Address>:<SM-Port>/server?server_id=SERVER_ID`

`http://<SM-IP-Address>:<SM-Port>/server?cluster_id=CLUSTER_ID`

`http://<SM-IP-Address>:<SM-Port>/server?mac=MAC`

`http://<SM-IP-Address>:<SM-Port>/server?ip=IP`

`http://<SM-IP-Address>:<SM-Port>/server[?tag=<tag_name>=<tag_value>,.]`

Method : DELETE

Payload : None

### API: Retrieve Server Configuration

Use one of the following methods to retrieve a server configuration. The detail argument is optional, and specified as part of the URL if details of the server entry are requested.

URL: `http://<SM-IP-Address>:<SM-Port>/server[?server_id=SERVER_ID&detail]`

`http://<SM-IP-Address>:<SM-Port>/server[?cluster_id=CLUSTER_ID&detail]`

`http://<SM-IP-Address>:<SM-Port>/server[?tag=<tag_name>=<tag_value>,.]`

`http://<SM-IP-Address>:<SM-Port>/server[?mac=MAC&detail]`

`http://<SM-IP-Address>:<SM-Port>/server[?ip=IP&detail]`

`http://<SM-IP-Address>:<SM-Port>/server[?tag=<tag_name>=<tag_value>,.]`

Method : GET

Payload : None

### API: Add an Image

Use the following to add a new image to the Server Manager configuration database from the Server Manager machine.

An image is either an ISO for a CentOS or Ubuntu distribution or an Ubuntu Contrail package repository. When adding an image, the image file is assumed to be available on the Server Manager machine.

URL : `http://<SM-IP-Address>:<SM-Port>/image`

Method: PUT

Payload: Specifies all the parameters that define the image being added.

See sample payload in the following:

<https://github.com/Juniper/contrail-server-manager/blob/master/src/client/new-package.json>

### API: Upload an Image

Use the following to upload a new image from a client to the Server Manager configuration database.

An image is an ISO for a CentOS or Ubuntu distribution or an Ubuntu Contrail package repository. Add image assumes the file is available on the Server Manager, whereas upload image transfers the image file from the client machine to the Server Manager machine.

URL : `http://<SM-IP-Address>:<SM-Port>/image/upload`

Method: PUT

Payload: Specifies all the parameters that define the image being added.

```
{
  "image": [
    {
      "id": "Image-id",
      "type": "image_type", <ubuntu or centos or esxi5.1 or esxi5.5 or contrail-ubuntu-
package or contrail-centos-package>
      "version": "image_version",
      "path": "path-to-image-on-client-machine"
    }
  ]
}
```

### API: Get Image Information

Use the following to get image information.

URL : `http://<SM-IP-Address>:<SM-Port>/image[?image_id=IMAGE_ID&detail]`

Method: GET

Payload: Specifies criteria for the image being sought. If no match criteria is specified, information about all the images is provided. The details field specifies if details of the image entry in the database are requested.

### API: Delete an Image

Use the following to delete an image.



URL: `http://<SM-IP-Address>:<SM-Port>/image?image_id=IMAGE_ID`

Method: DELETE

Payload: Specifies criteria for the image being deleted.

### API: Add or Modify a Cluster

Use the following to add a cluster to the Server Manager configuration database. A cluster maintains parameters for a set of servers that work together in different roles to provide complete functions for a Contrail cluster.

URL: `http://<SM-IP-Address>:<SM-Port>/cluster`

Method: PUT

Payload: Contains the definition of the cluster, including all the global parameters needed by all the servers in the cluster. The `subnet_mask`, `gateway`, `password`, and `domain` fields define parameters that apply to all servers in the VNS. These parameter values can be individually overridden for a server by specifying different values in the server entry.

A sample JSON for Contrail Release 4.0 is at the following:

<https://github.com/Juniper/contrail-server-manager/blob/master/src/client/new-cluster-contrail-4.x.json>

### API: Delete a Cluster

Use this API to delete a cluster from the Server Manager database.

URL: `http://<SM-IP-Address>:<SM-Port>/cluster?cluster_id=CLUSTER_ID`

Method: DELETE

Payload: None

### API: Get Cluster Configuration

Use this API to get a cluster configuration.

URL: `http://<SM-IP-Address>:<SM-Port>/cluster[?cluster_id=CLUSTER_ID&detail]`

Method: GET

Payload: None

The optional detail argument is specified as part of the URL if details of the VNS entry are requested.

### API: Get All Server Manager Configurations

Use this API to get all configurations of Server Manager objects, including servers, clusters, images, and tags.

URL: `http://<SM-IP-Address>:<SM-Port>/all[?detail]`

Method: GET

Payload: None

The optional detail argument is specified as part of the URL if details of the Server Manager configuration are requested.

### API: Reimage Servers

Use one of the following API formats to reimage one or more servers.

URL: `http://<SM-IP-Address>:<SM-Port>/server/reimage?server_id=SERVER_ID`

`http://<SM-IP-Address>:<SM-Port>/server/reimage?cluster_id=CLUSTER_ID`

`http://<SM-IP-Address>:<SM-Port>/server/reimage?mac=MAC`

`http://<SM-IP-Address>:<SM-Port>/server/reimage?ip=IP`

`http://<SM-IP-Address>:<SM-Port>/server/reimage [?tag=<tag_name>=<tag_value>, .]`

Method: POST

Payload: None

### API: Provision Servers

Use this API to provision or configure one or more servers for roles configured on them.

URL: `http://<SM-IP-Address>:<SM-Port>/server/provision`

Method: POST

Payload: Specifies the criteria to be used to identify servers which are being provisioned. The servers can be identified by server\_id, mac, cluster\_id or tags. See the following example.

```
{
  server_id : <server_id> OR
  mac : <server_mac_address> OR
  cluster_id : <cluster_id> OR
  tag : {"data-center" : "dc1"} }
}
```

## API: Restart Servers

This REST API is used to power cycle the servers and reboot either with net-booting enabled or disabled.

If the servers are to be reimaged and reprovisioned, the **net-boot** flag should be set.

If servers are only being reprovisioned, the **net-boot** flag is not needed, however, the Puppet agent must be running on the target systems with the correct puppet configuration to communicate to the puppet master running on the Server Manager.

URL: `http://<SM-IP-Address>:<SM-Port>/server/restart?server_id=SERVER_ID`  
`http://<SM-IP-Address>:<SM-Port>/server/restart?[netboot&]cluster_id=CLUSTER_ID`  
`http://<SM-IP-Address>:<SM-Port>/server/restart? [netboot&]mac=MAC`  
`http://<SM-IP-Address>:<SM-Port>/server/restart? [netboot&]ip=IP`  
`http://<SM-IP-Address>:<SM-Port>/server/restart ? [netboot&]tag=<tag_name>=<tag_value>`

Method: POST

Payload: Specifies the criteria to be used to identify servers which are being restarted. The servers can be identified by their **server\_id**, **mac**, **cluster\_id**, or **tag**. The netboot parameter specifies if the servers being power-cycled are to be booted from Cobbler or locally.

## Example: Reimaging and Provisioning a Server

This example shows the steps used in Server Manager software to configure, reimage, and provision a server running all roles of the Contrail system in a single-node configuration.

**NOTE:** Component names and IP addresses in the following are used for example only. To use this example in your own environment, be sure to use addresses and names specific to your environment.

The Server Manager client configuration file used for the following CLI commands, is `/opt/contrail/server_manager/client/sm-client-config.ini`. It contains the values for the server IP address and port number as follows:

```
[SERVER-MANAGER]
listen_ip_addr = 192.168.1.10 (Server Manager IP address)
listen_port = 9001
```

### Overview

The steps to be followed include:

1. Configure cluster.
2. Configure servers.
3. Configure images.
4. Reimage servers (either using servers configured above or using explicitly specified reimage parameters with the request).
5. Provision servers (either using servers configured above or using explicitly specified provision parameters with the request).

#### *Procedure*

1. Configure a cluster.

```
server-manager add cluster -f cluster.json
```

2. Configure the server.

```
server-manager add server -f server.json
```

3. Configure images.

In the example, the image files for ubuntu-xx.xx.x and contrail-ubuntu-164 are located at the corresponding image path specified on the Server Manager.

```
server-manager add -c smgr_client_config.ini image -f image.json
```

4. Reimage servers.

This step can be performed after the configuration from the previous steps is in the Server Manager database.

```
server-manager reimage -server_id demo-server ubuntu-<x.xx.x>
```

5. Provision servers.

```
server-manager provision -server_id demo-server contrail-ubuntu-164
```

**NOTE:** The samples for all JSONs used in the procedure above are available as links in the documentation for the API calls for those respective commands.

#### SEE ALSO

*[Installing Server Manager](#)*

*[Using the Server Manager Web User Interface](#)*

[Installing and Using Server Manager Lite](#)

## Using the Server Manager Web User Interface

### IN THIS SECTION

- [Log In to Server Manager | 357](#)
- [Create a Cluster for Server Manager | 358](#)
- [Edit a Cluster through Edit JSON | 369](#)
- [Working with Servers in the Server Manager User Interface | 369](#)
- [Add a Server | 370](#)
- [Edit Tags for Servers | 373](#)
- [Using the Edit Config Option for Multiple Servers | 373](#)
- [Edit a Server through Server Manager, Edit JSON | 374](#)
- [Filter Servers by Tag | 375](#)
- [Viewing Server Details | 375](#)
- [Configuring Images and Packages | 378](#)
- [Add New Image or Package | 379](#)
- [Selecting Server Manager Actions for Clusters | 379](#)
- [Reimage a Cluster | 380](#)
- [Provision a Cluster | 380](#)

When the Server Manager is installed on your Contrail system, you can also install a Server Manager Web user interface that you can use to access the features of Server Manager.

### Log In to Server Manager

The Server Manager user interface can be accessed using:

```
http://<server-manager-user-interface-ip>:9080
```

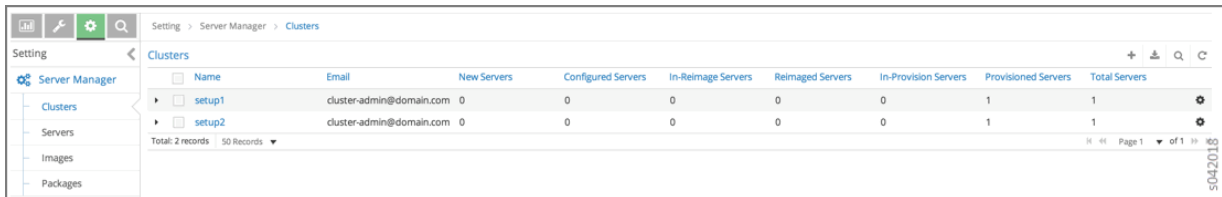
Where *<server-manager-user-interface-ip>* is the IP address of the server on which the Server Manager web user interface is installed.

From the Contrail user interface, select **Setting > Server Manager** to access the Server Manager home page. From this page you can manage Server Manager settings for clusters, servers, images, and packages.

## Create a Cluster for Server Manager

Select **Add Cluster** to identify a cluster to be managed by the Server Manager. Select **Setting > Server Manager > Clusters**, to access the **Clusters** page, see [Figure 37 on page 358](#).

**Figure 37: Server Manager > Clusters**

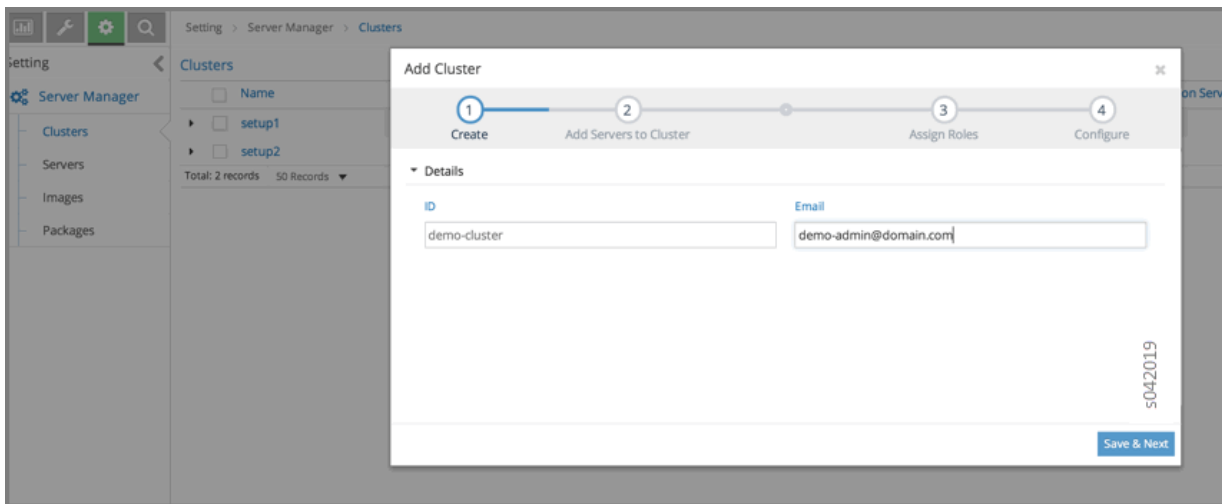


| Name   | Email                    | New Servers | Configured Servers | In-Reimage Servers | Reimaged Servers | In-Provision Servers | Provisioned Servers | Total Servers |
|--------|--------------------------|-------------|--------------------|--------------------|------------------|----------------------|---------------------|---------------|
| setup1 | cluster-admin@domain.com | 0           | 0                  | 0                  | 0                | 0                    | 1                   | 1             |
| setup2 | cluster-admin@domain.com | 0           | 0                  | 0                  | 0                | 0                    | 1                   | 1             |

Total: 2 records 50 Records

To create a new cluster, click the plus icon in the upper right of the **Clusters** page. The **Add Cluster** window is displayed. In the **Add Cluster** window, you can add a new cluster ID and the domain e-mail address of the cluster. See [Figure 38 on page 358](#).

**Figure 38: Add Cluster**



**Add Cluster**

1 Create 2 Add Servers to Cluster 3 Assign Roles 4 Configure

**Details**

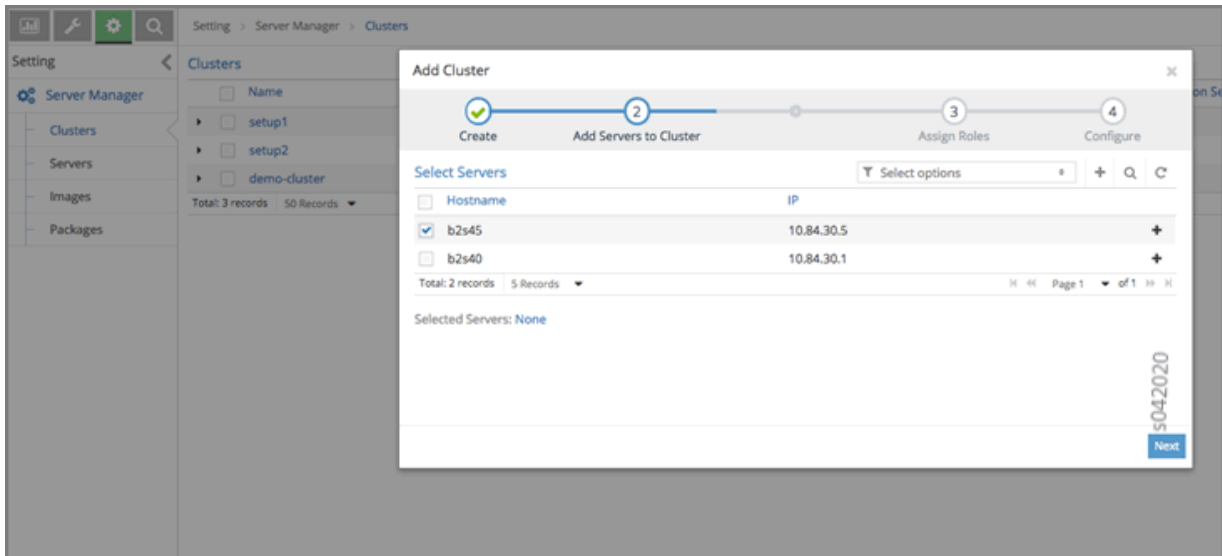
ID: demo-cluster

Email: demo-admin@domain.com

Save & Next

When you are finished adding information about the new cluster in the **Add Clusters** window, click **Save & Next**. Now you can add servers to the cluster, see [Figure 39 on page 359](#).

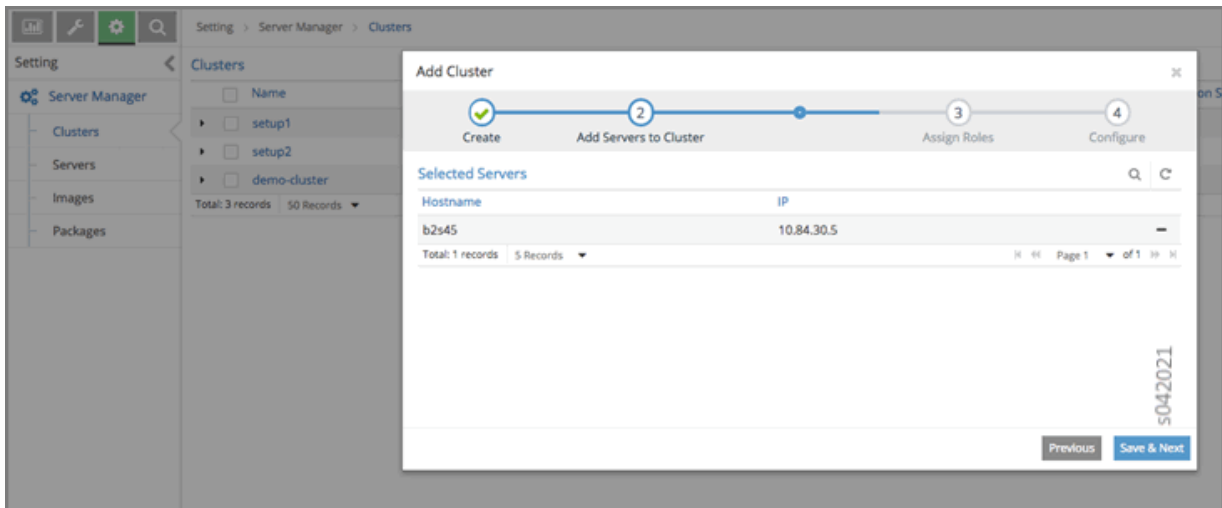
Figure 39: Add Servers to Cluster



Click the check box of each server to be added to the cluster.

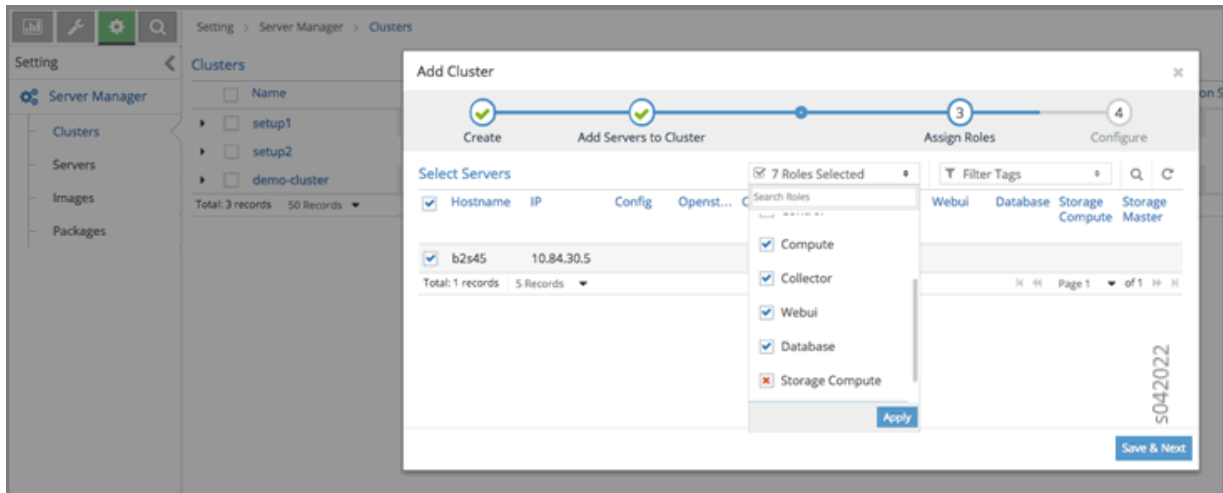
When you are finished, click **Next**. The selected servers are added to the cluster, see [Figure 40 on page 359](#).

Figure 40: Add Servers to Cluster, Next



When you are finished adding servers, click **Save & Next**. Now you can assign Contrail roles to servers that you select in the cluster. Roles available are Config, OpenStack, Control, Compute, and Collector. Select each role assignment for the selected server. You can also unselect any assigned role. The assigned roles correspond to the role functions in operation on the server, see [Figure 41 on page 360](#).

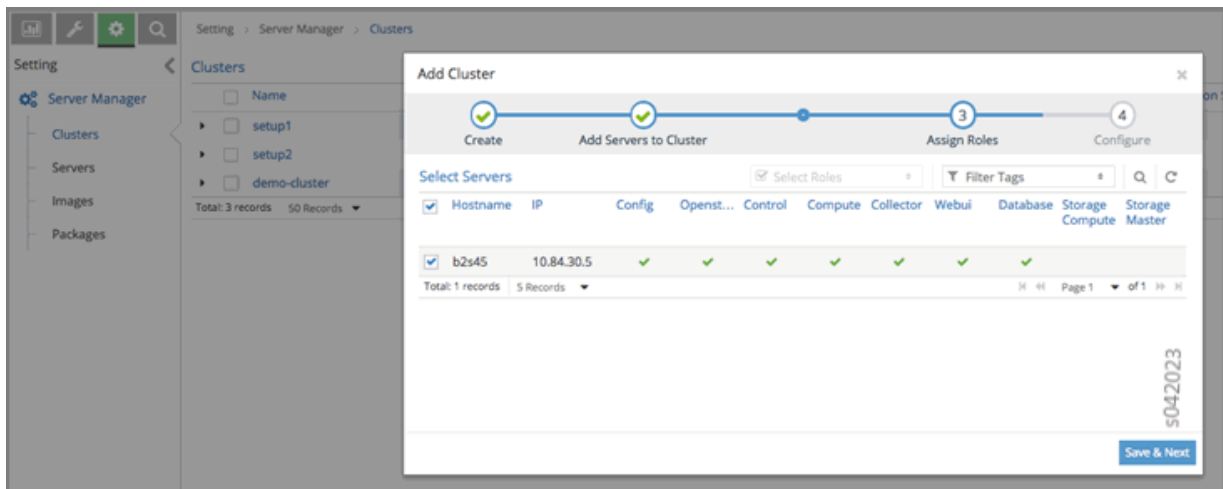
Figure 41: Assign Roles



When you are finished selecting roles for the selected server in the **Roles** window, click **Apply** to save your choices.

Click **Save & Next** to view your selections. Check marks are displayed in the columns of the **Add Cluster** window, see [Figure 42 on page 360](#).

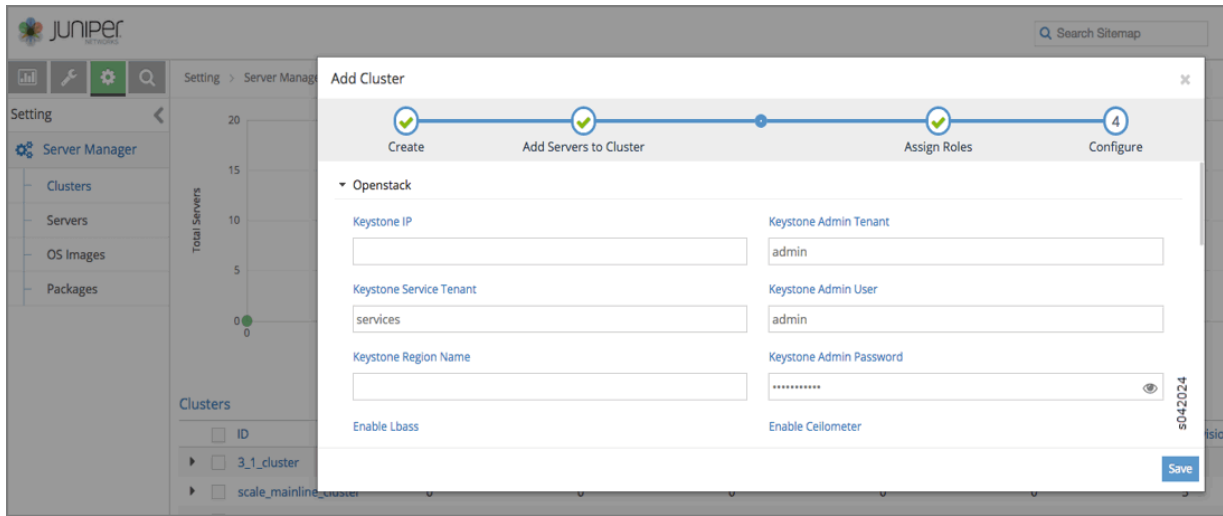
Figure 42: Roles Assigned



The next step after roles are assigned is to enter the cluster configuration information for OpenStack. After viewing the assigned roles, click **Save & Next**. The **Add Cluster** window is displayed. Click an icon that opens a set of fields where you can enter OpenStack or Contrail configuration information for the cluster. In the following image, the **Openstack** icon is selected. You can enter **Keystone** configuration information, such as IP, Admin tenant, user, and password, service tenant, and region name. You can also enable LBaaS and Ceilometer, see [Figure 43 on page 361](#).

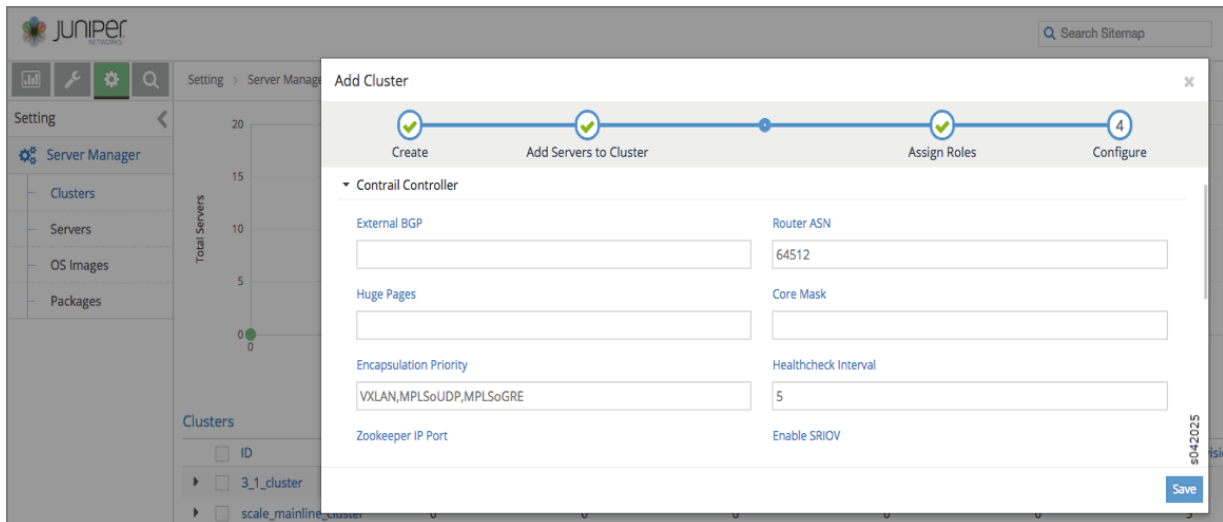


Figure 43: OpenStack Configuration



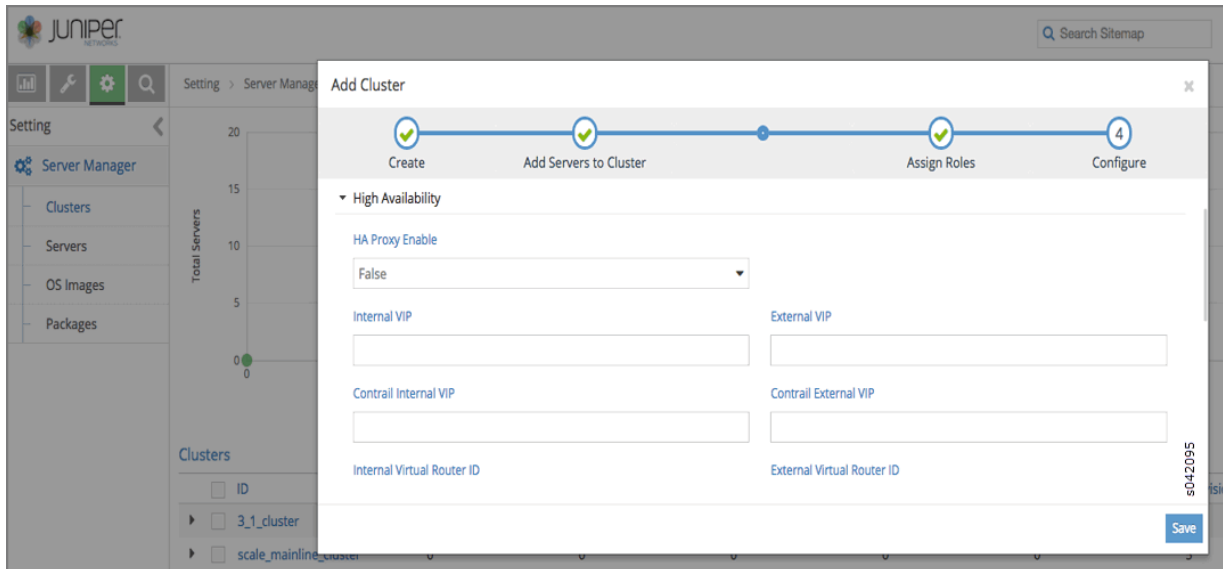
In the following image, the Contrail controller icon is selected. You can enter configuration information for Contrail, such as **External BGP, Router ASN, Huge Pages, Core Mask, Encapsulation Priority, Healthcheck Interval, Zookeeper IP Port, Enable SRIOV**, and so on, see [Figure 44 on page 361](#).

Figure 44: Configure Contrail



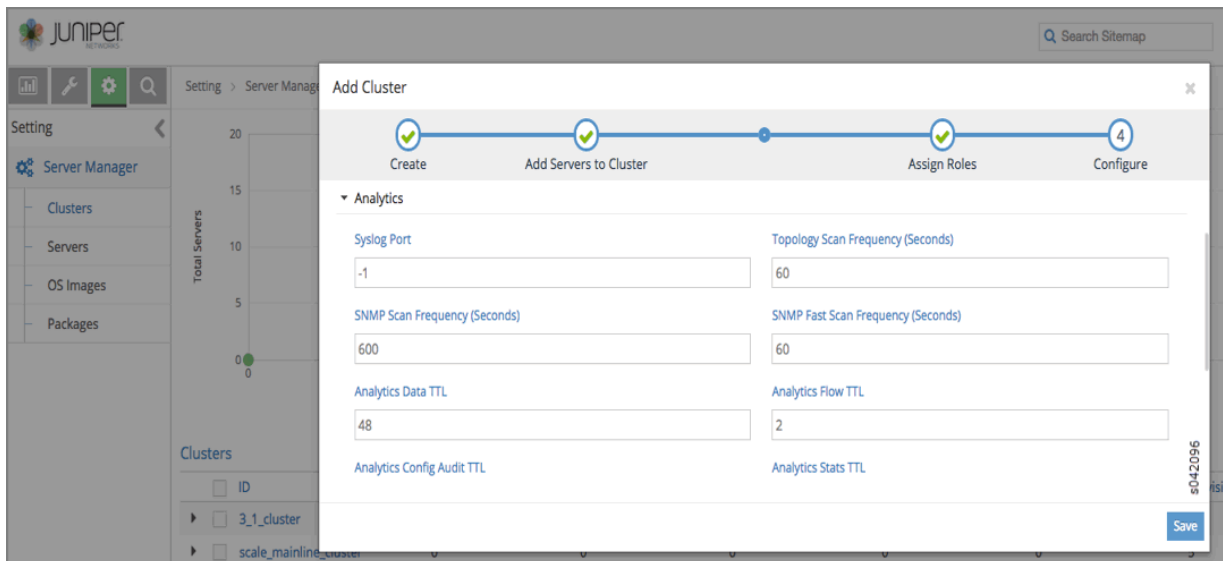
In the following image, the High Availability (HA) icon is selected. You can configure high availability parameters such as HA Proxy Enable, Internal and External VIP, and so on, see [Figure 45 on page 362](#).

Figure 45: Configure High Availability



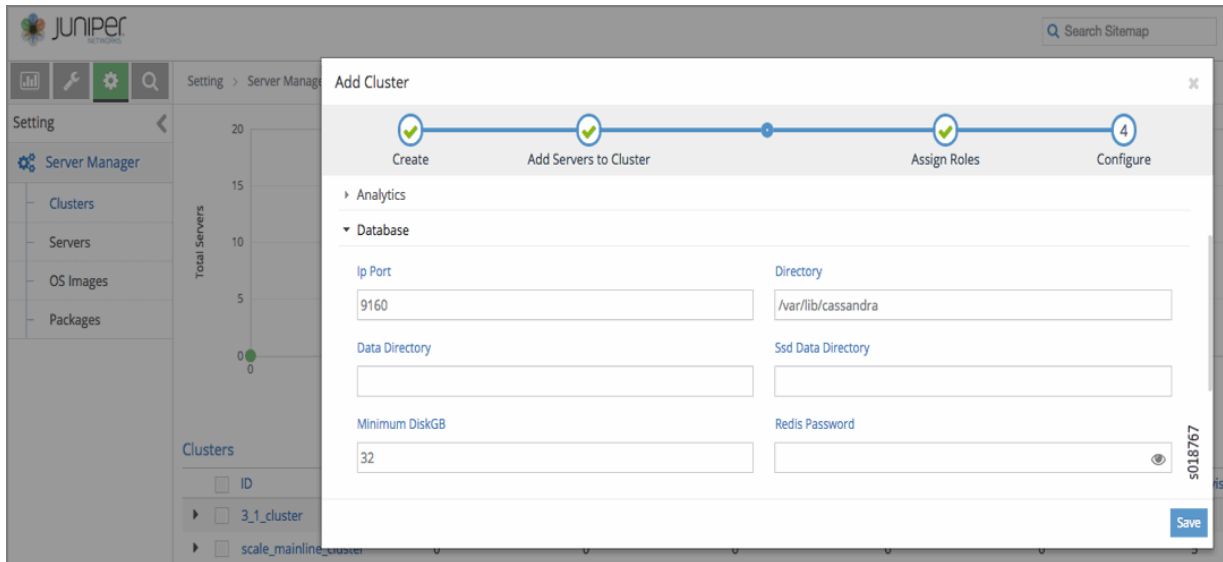
In the following image, the **Analytics** icon is selected. Here you can configure parameters for Contrail Analytics, including **Syslog Port**, various scan frequencies, and various TTL settings, see [Figure 46 on page 362](#).

Figure 46: Configure Analytics



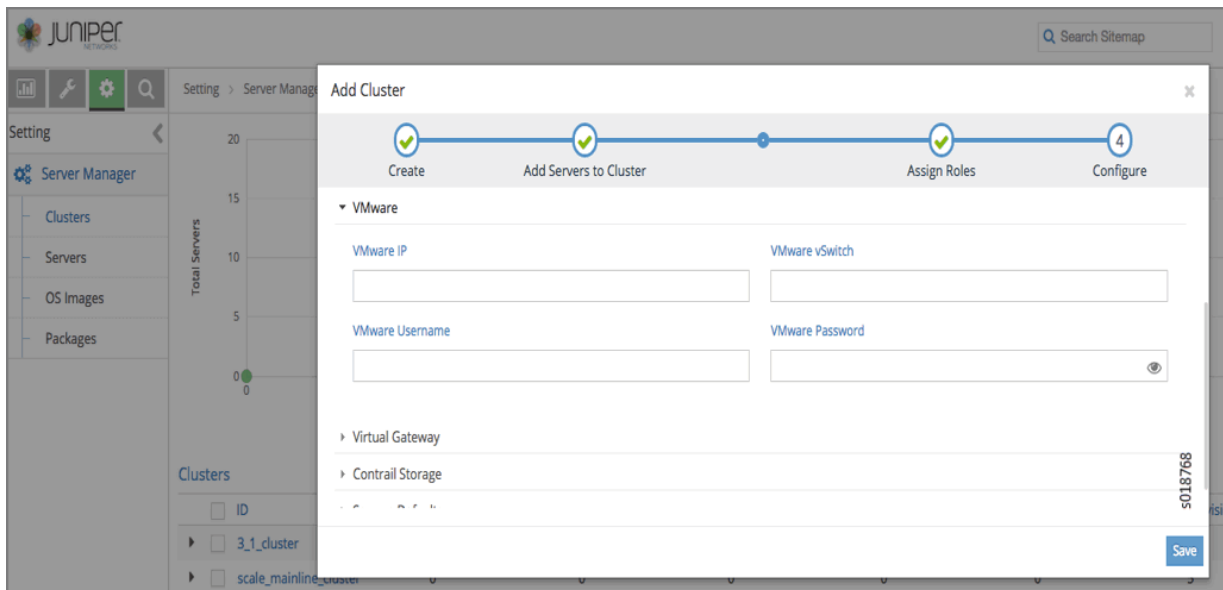
In following image, the **Database** icon is selected. You can configure parameters for the Contrail database, including **IP Port**, **Directory**, **Minimum Disk GB**, and so on, see [Figure 47 on page 363](#).

Figure 47: Configure Database



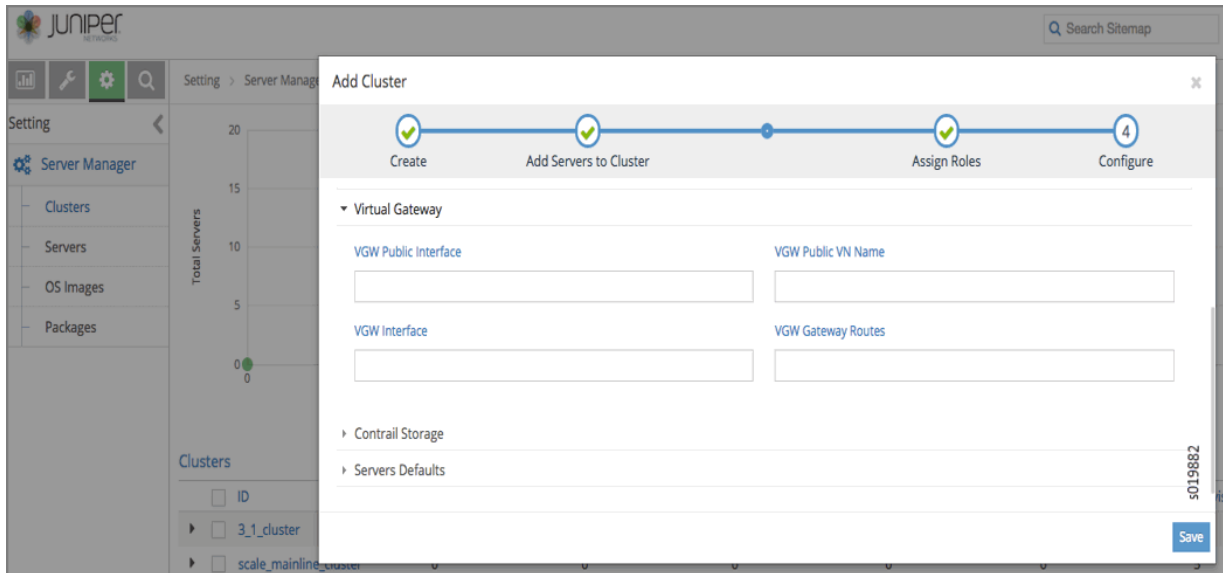
In following image, the **VMware** icon is selected. You can configure parameters for Contrail VMware , including **VMware IP**, **VMware vSwitch**, **Username**, **Password** , and so on, see [Figure 48 on page 363](#).

Figure 48: Configure VMware



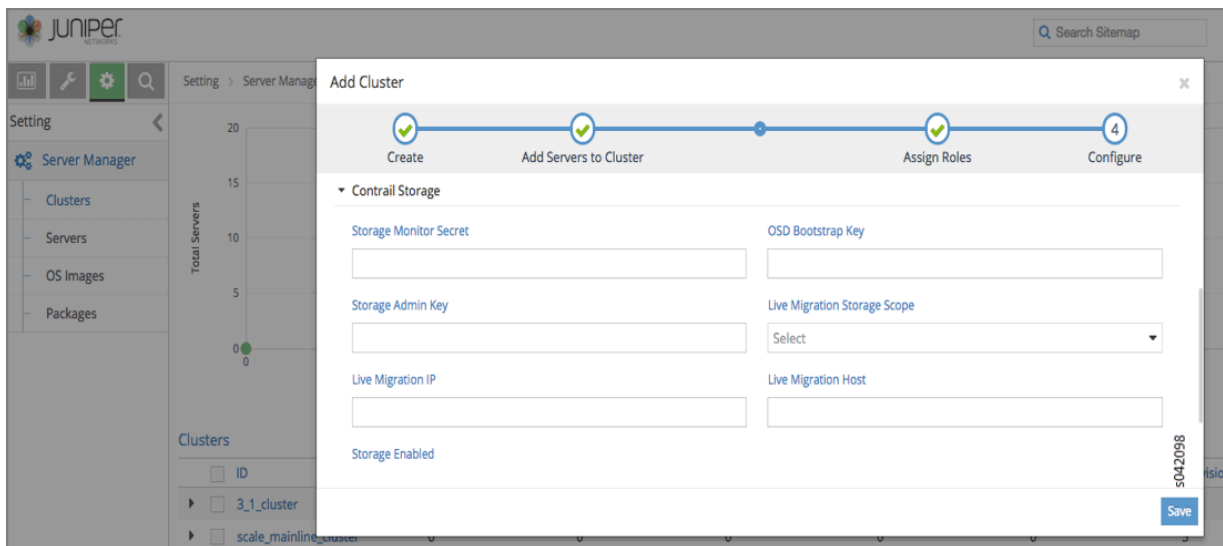
In following image, the **Virtual Gateway** icon is selected. You can configure parameters for the Contrail Virtual Gateway, including **VGW Public Interface**, **VGW Public VN Name**, **VGW Interface**, **Routes** , and so on, see [Figure 49 on page 364](#).

Figure 49: Configure Virtual Gateway



In following image, the **Contrail Storage** icon is selected. You can configure parameters for Contrail Storage, including **Storage Monitor Secret**, **OSD Bootstrap Key**, **Admin Key**, and so on, see [Figure 50 on page 364](#).

Figure 50: Configure Contrail Storage



When you are finished entering all of the cluster configuration information, click **Save** to submit the configurations. You can view all configured clusters on the **Clusters** window by selecting **Setting > Server Manager > Clusters**, see [Figure 51 on page 365](#).

Figure 51: View Configured Clusters

| Name         | Email                    | New Servers | Configured Servers | In-Reimage Servers | Reimagined Servers | In-Provision Servers | Provisioned Servers |
|--------------|--------------------------|-------------|--------------------|--------------------|--------------------|----------------------|---------------------|
| setup1       | cluster-admin@domain.com | 0           | 0                  | 0                  | 0                  | 0                    | 0                   |
| setup2       | cluster-admin@domain.com | 0           | 0                  | 0                  | 0                  | 0                    | 1                   |
| demo-cluster | demo-admin@domain.com    | 0           | 0                  | 0                  | 0                  | 0                    | 1                   |

Total: 3 records | 50 Records

To perform an action on one of the configured clusters, click the gear wheel icon at the right to select from a menu of actions available for that cluster, including **Add Servers**, **Remove Servers**, **Assign Roles**, **Edit Config**, **Reimage**, **Provision**, and **Delete**, see [Figure 52 on page 365](#).

Figure 52: Select Cluster Action

Setting > Server Manager > Clusters

Search Sitemap | Alarms | admin

Total Servers

Max. CPU Utilization (%)

| ID                     | New | Configured | In-Reimage | Reimagined | In-Provision | Provisioned | Total |
|------------------------|-----|------------|------------|------------|--------------|-------------|-------|
| 3_1_cluster            | 0   | 0          | 0          | 0          | 0            | 1           | 1     |
| scale_mainline_cluster | 0   | 0          | 0          | 0          | 0            | 5           | 5     |
| 3_0_2_cluster          | 0   | 0          | 0          | 0          | 0            | 1           | 1     |
| mainline_cluster       | 0   | 0          | 0          | 0          | 0            | 1           | 1     |
| test-cluster           | 0   | 0          | 0          | 0          | 0            | 0           | 0     |

Total: 5 records | 8 Records

- + Add Servers
- Remove Servers
- Assign Roles
- Edit Config
- Edit JSON
- Reimage
- Provision
- Refresh Inventory
- Delete

You can also click the expansion icon on the left side of the cluster name to display the details of that cluster in an area below the name line, see [Figure 53 on page 366](#).

Figure 53: Display Cluster Details

The screenshot displays the 'Clusters' page in the Server Manager interface. The top navigation bar shows 'Setting > Server Manager > Clusters'. The left sidebar contains 'Server Manager' with sub-items: Clusters, Servers, OS Images, and Packages. The main content area features a table of clusters with columns: ID, New, Configured, In-Reimage, Reimaged, In-Provision, Provisioned, and Total. Below the table, the details for the 'test-cluster' are shown in a two-column layout: Overview, Openstack, Contrail Controller, High Availability, Analytics, Status, Contrail Storage, and Servers Defaults.

| ID                     | New | Configured | In-Reimage | Reimaged | In-Provision | Provisioned | Total |
|------------------------|-----|------------|------------|----------|--------------|-------------|-------|
| 3_1_cluster            | 0   | 0          | 0          | 0        | 0            | 1           | 1     |
| scale_mainline_cluster | 0   | 0          | 0          | 0        | 0            | 5           | 5     |
| 3_0_2_cluster          | 0   | 0          | 0          | 0        | 0            | 1           | 1     |
| mainline_cluster       | 0   | 0          | 0          | 0        | 0            | 1           | 1     |
| test-cluster           | 0   | 0          | 0          | 0        | 0            | 0           | 0     |

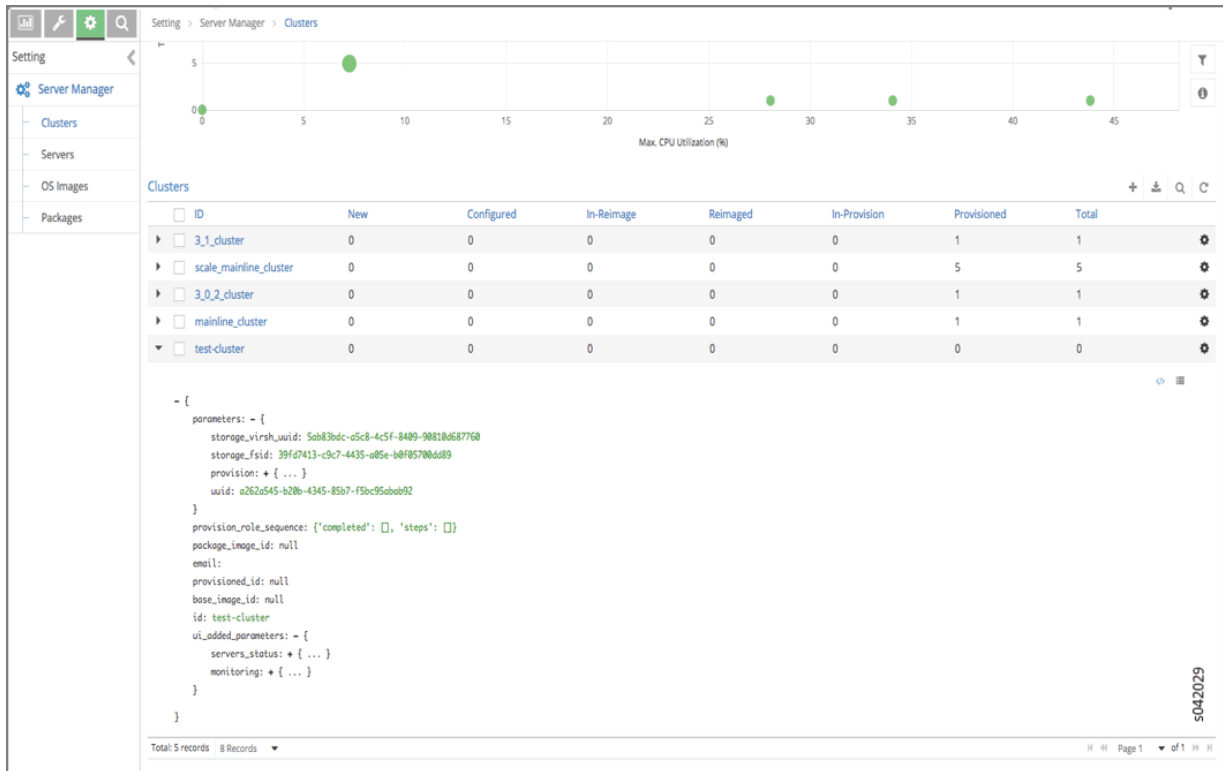
  

| Overview               |                        | Status               |      |
|------------------------|------------------------|----------------------|------|
| ID                     | test-cluster           | Total Servers        | 0    |
| Openstack              |                        | New Servers          | 0    |
| Admin Tenant           | admin                  | Configured Servers   | 0    |
| Service Tenant         | services               | In-Reimage Servers   | 0    |
| Admin User             | admin                  | Reimaged Servers     | 0    |
| Contrail Controller    |                        | In-Provision Servers | 0    |
| Encapsulation Priority | VLAN,MPLSoUDP,MPLSoGRE | Provisioned Servers  | 0    |
| Multi Tenancy          | true                   | Contrail Storage     |      |
| Router ASN             | 64512                  | Storage Virsh UUID   | -    |
| Directory              | /var/lib/cassandra     | Servers Defaults     |      |
| Minimum DiskGB         | 32                     | Domain               | -    |
| Healthcheck Interval   | 5                      | Kernel Upgrade       | true |
| High Availability      |                        |                      |      |
| Haproxy Enable         | false                  |                      |      |
| Analytics              |                        |                      |      |
| Data TTL               | 48                     |                      |      |
| Syslog Port            | -1                     |                      |      |

Total: 5 records | 8 Records | Page 1 of 1

Click the upper right icon to switch to the JSON view to see the contents of the JSON file for the cluster, see [Figure 54 on page 367](#).

Figure 54: View Cluster JSON



The cluster name is a link, click the cluster name to display the cluster **Details** page, see [Figure 55 on page 368](#).

Figure 55: Link to View Cluster Details

The screenshot shows the Juniper Server Manager interface. The breadcrumb navigation is 'Setting > Server Manager > Clusters > test-cluster'. The left sidebar shows 'Server Manager' with sub-items: Clusters, Servers, OS Images, and Packages. The main content area is divided into two columns. The left column contains several expandable sections: Overview (ID: test-cluster), Openstack (Admin Tenant: admin, Service Tenant: services, Admin User: admin), Contrail Controller (Encapsulation Priority: VXLAN,MPLSoUDP,MPLSoGRE, Multi Tenancy: true, Router ASN: 64512, Directory: /var/lib/cassandra, Minimum DiskGB: 32, Healthcheck Interval: 5), High Availability (Haproxy Enable: false), and Analytics (Data Ttl: 48, Syslog Port: -1). The right column contains sections for Status (Total Servers: 0, New Servers: 0, Configured Servers: 0, In-Reimage Servers: 0, Reimaged Servers: 0, In-Provision Servers: 0, Provisioned Servers: 0), Contrail Storage (Storage Virsh UUID: -), and Servers Defaults (Domain: -, Kernel Upgrade: true). A vertical ID 'S042080' is visible on the right edge.

Click the **Servers** tab to display the servers under that cluster, see [Figure 56 on page 368](#).

Figure 56: Display Servers for Cluster

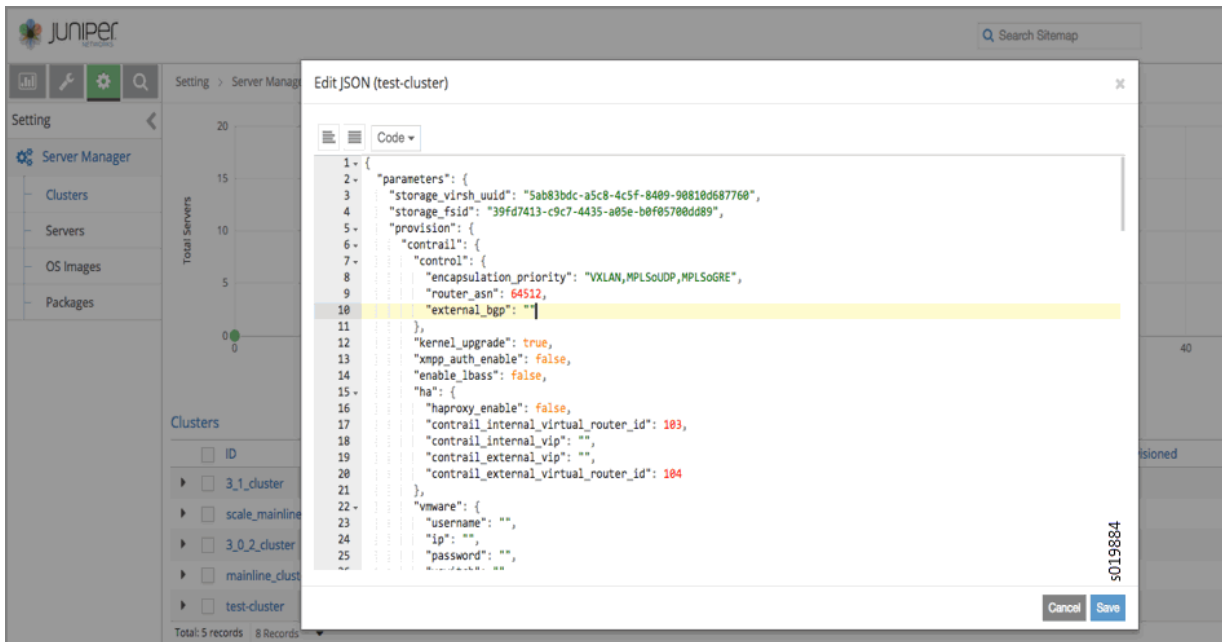
The screenshot shows the Juniper Server Manager interface with the 'Servers' tab selected. The breadcrumb navigation is 'Setting > Server Manager > Clusters > test-cluster'. The left sidebar is the same as in Figure 55. The main content area features a large empty chart with 'Memory Usage (%)' on the y-axis and 'CPU Utilization (%)' on the x-axis, both ranging from 0.0 to 1.0. The chart contains the text 'No Data found.'. Below the chart is a table header for 'Servers' with columns: ID, IP, Roles, Status, and Provisioned Id. The table body contains the text 'No data available.'. A 'Filter Tags' section with icons for adding, removing, and clearing filters is located above the table. A vertical ID 'S0198833' is visible on the right edge.



## Edit a Cluster through Edit JSON

Select **Edit JSON** to edit a cluster by editing the JSON file. Make changes to the JSON code and click **Save** to save the edited configuration for the cluster, see [Figure 57 on page 369](#).

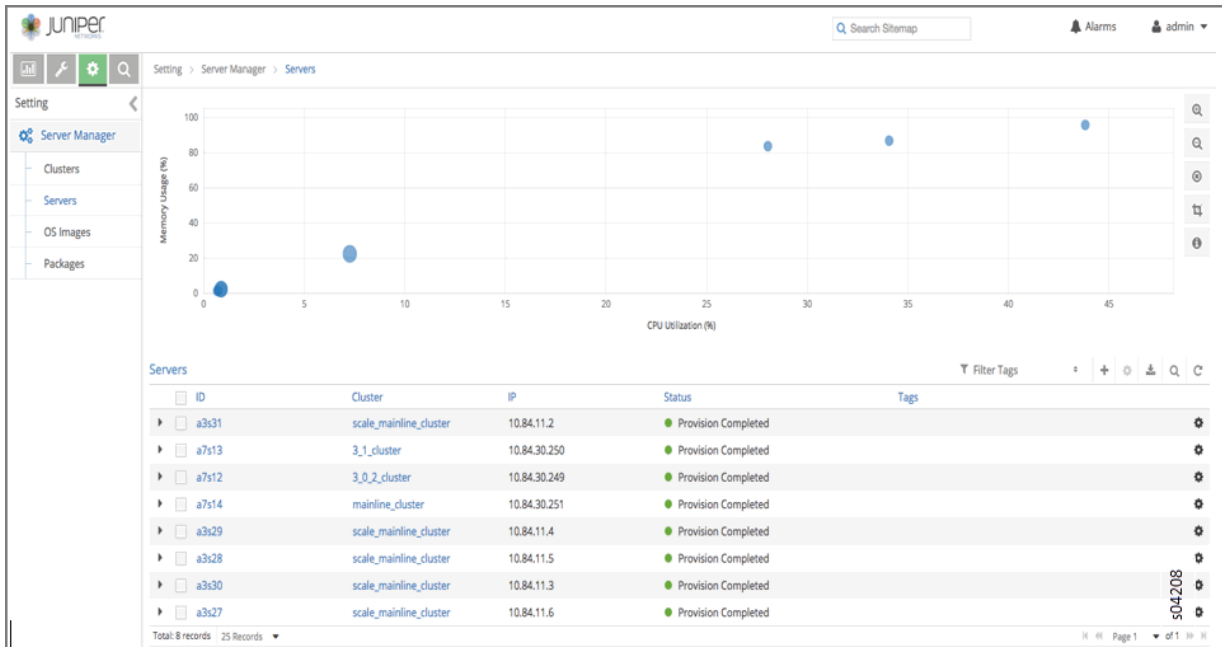
Figure 57: Edit Cluster JSON



## Working with Servers in the Server Manager User Interface

Select **Setting > Server Manager** and click the **Servers** link in the left sidebar to view a list of all servers, see [Figure 58 on page 370](#).

Figure 58: View Servers



## Add a Server

To add a new server, select **Setting > Server Manager > Servers** and click the plus (+) icon at the upper right side in the header line. The **Add Server** window is displayed, see [Figure 59 on page 371](#), in which the **System Management** tab is expanded. Here you enter the details of ID, Password, Domain, Partition, and so on for the server.

Figure 59: Add Server, System Management

Setting > Server Manager > Servers

Setting < Servers

Server Manager

- Clusters
- Servers
- Images
- Packages

Total: 2 records 50 Records

### Add Server

System Management

ID: demo-server Password: [password field]

Host Name: demo-server Domain: englab.juniper.net

Static IP: [empty] IPMI Address: 10.84.60.148

IPMI Username: ADMIN IPMI Password: [password field]

Partition: [empty]

Interfaces: [empty]

Cancel Save

s042082

In the following image, the **Physical Interfaces** icon is selected. You can add new interfaces or edit existing interfaces. To enable editing for any field, hover the cursor on any selected field to open it, see [Figure 60 on page 371](#).

Figure 60: Add Server, Physical Interfaces

### Add Server

System Management

Physical Interfaces

| Name  | IP/Mask | MAC Address       | Gateway | DHCP                                | TOR                      | TOR Port |
|-------|---------|-------------------|---------|-------------------------------------|--------------------------|----------|
| eth01 | 1.2.3.4 | aa:aa:aa:aa:aa:aa | 2.2.3.4 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | -        |

+Add

Bond Interfaces

OVS Type Switches

Contrail Storage

Provisioning

Cancel Save

s042097

3\_1\_cluster 10.84.30.250 Provision Completed

In the following image, the **Contrail Storage** icon is selected. You can configure parameters for Contrail Storage, including selecting a package and adding storage disks locations, see [Figure 61 on page 372](#).

**Figure 61: Add Server, Contrail Storage**

The screenshot shows a web interface window titled "Add Server". The left sidebar contains a navigation menu with the following items: System Management, Physical Interfaces, Bond Interfaces, OVS Type Switches, **Contrail Storage** (selected), and Provisioning. The main content area is for the "Contrail Storage" configuration. It includes a "Storage Repo ID" dropdown menu with the text "Select Repo ID". Below that is a "Chassis ID" dropdown menu with the text "Select Chassis ID" and a text input field labeled "Add New Chassis ID". There is also a "Storage Disks" section with a "+Add" button. At the bottom right of the window, there are "Cancel" and "Save" buttons. The status bar at the bottom of the window displays "scale\_mainline\_cluster", "10.84.11.4", and "Provision Completed". A vertical ID "s042099" is visible on the right side of the window.

When you are finished entering new server details in the **Add Server** window, click **Save** to add the new server configuration to the list of servers.

You can change details of the new server by clicking the gear wheel icon to the right side to get a list of actions available, including **Edit Config**, **Edit JSON**, **Edit Tags**, **Reimage**, **Provision**, **Refresh Inventory**, and **Delete**, see [Figure 62 on page 373](#).

Figure 62: Select Server Actions

| ID    | Cluster                | IP           | Status              | Tags |
|-------|------------------------|--------------|---------------------|------|
| a3s31 | scale_mainline_cluster | 10.84.11.2   | Provision Completed |      |
| a7s13 | 3_1_cluster            | 10.84.30.250 | Provision Completed |      |
| a7s12 | 3_0_2_cluster          | 10.84.30.249 | Provision Completed |      |
| a7s14 | mainline_cluster       | 10.84.30.251 | Provision Completed |      |
| a3s29 | scale_mainline_cluster | 10.84.11.4   | Provision Completed |      |
| a3s28 | scale_mainline_cluster | 10.84.11.5   | Provision Completed |      |
| a3s30 | scale_mainline_cluster | 10.84.11.3   | Provision Completed |      |
| a3s27 | scale_mainline_cluster | 10.84.11.6   | Provision Completed |      |

## Edit Tags for Servers

Select **Edit Tags** from the gear wheel icon menu. The **Edit Tags** window is displayed. Enter any user-defined tags to be associated with the selected server, then click **Save** to add the tags to the server configuration, see [Figure 63 on page 373](#).

Figure 63: Edit Tags

Setting > Server Manager > Servers

Setting < Servers

Server Manager

- Clusters
- Servers
- Images
- Packages

Total: 3 records 50 Records

Edit Tags (demo-server)

Datacenter:  Floor:

Hall:  Rack:

Custom Tag:

Cancel Save

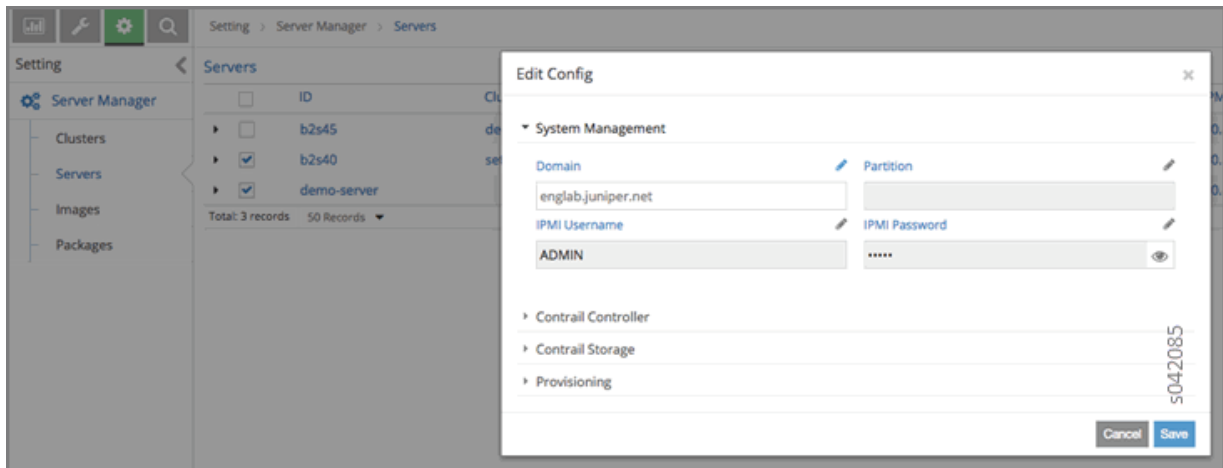
## Using the Edit Config Option for Multiple Servers

You can also edit the configuration of multiple servers at one time. From the **Servers** window at **Setting > Server Manager > Servers**, select the servers you want to edit, then click a gear wheel icon at the right to open the action menu, and select **Edit Config**.

The **Edit Config** window is displayed, as shown.

Click a pencil icon to open configuration fields that can be edited. Fields include **System Management**, **Contrail Controller**, **Contrail Storage**, and so on, see [Figure 64 on page 374](#).

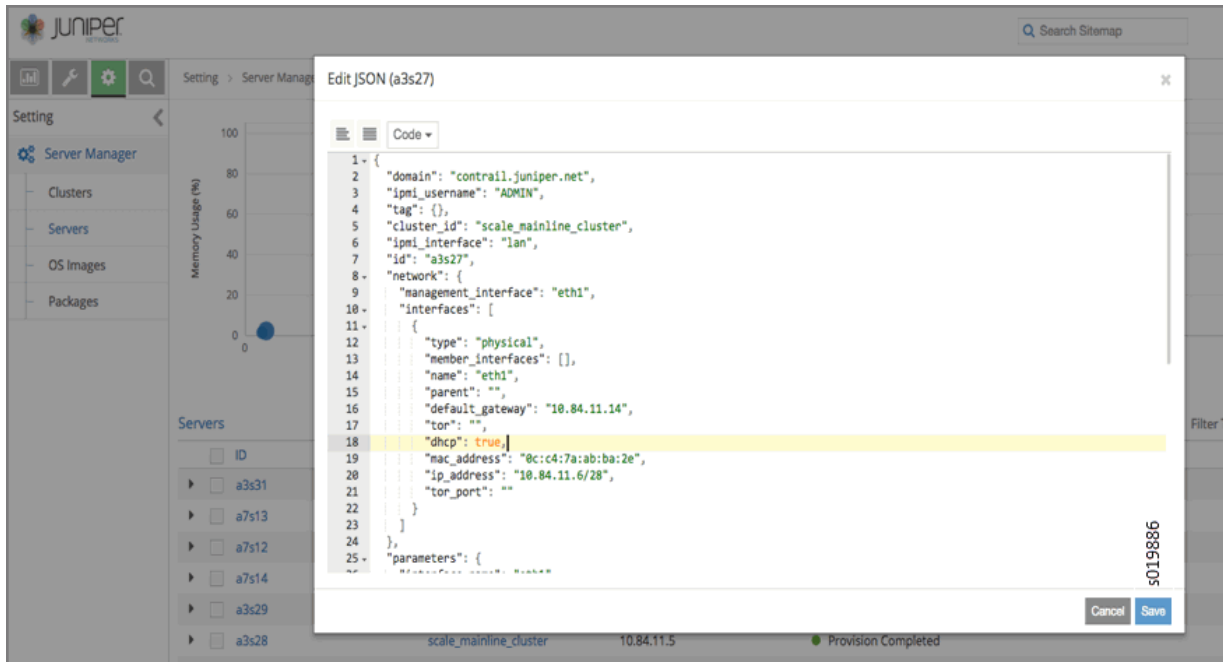
**Figure 64: Edit Config, Multiple Servers**



### Edit a Server through Server Manager, Edit JSON

Select **Edit JSON** to edit the server through JSON file. Make changes to the server details in the JSON, then click **Save**, see [Figure 65 on page 375](#).

Figure 65: Server Edit JSON



## Filter Servers by Tag

You can filter servers according to the tags defined for them. In the **Servers** window, click the **Filter Tags** field in the upper right heading. A list of configured tags is displayed. Select a tag by which to filter the list of servers, see [Figure 66 on page 375](#).

Figure 66: Filter Servers by Tag



## Viewing Server Details

Each server name on the **Servers** page is a link to the details page for that server. Click any server name to open the details for that server, including **System Management** information, **Status**, **Contrail Controller**, **Contrail Storage**, **Roles**, **Tags**, and **Provisioning**, see [Figure 67 on page 376](#).

Figure 67: View Server Details, System Management

The screenshot displays the Juniper Server Manager web interface. The breadcrumb navigation shows 'Setting > Server Manager > Servers > a3s27'. The left sidebar contains a 'Server Manager' menu with options for Clusters, Servers, OS Images, and Packages. The main content area is divided into two columns. The left column, titled 'System Management', contains several expandable sections: 'System Management' (with fields for ID, MAC Address, Domain, IP Address, IPMI Address, Gateway, and Subnet Mask), 'Contrail Controller' (with fields for Configured and Installed Packages), 'Contrail Storage' (with Storage Repo ID), 'Disks', and 'Interfaces' (with a table listing interface details). The right column contains sections for 'Status' (with Status and Last Updated fields), 'Roles' (with Roles field), 'Tags' (with Datacenter field), and 'Provisioning' (with Cluster, Installed OS Image, and Management Interface fields). A search bar and user profile are visible at the top right.

| Na... | IP Address    | Default Gateway | MAC Address       | DH... | Type     |
|-------|---------------|-----------------|-------------------|-------|----------|
| eth1  | 10.84.11.6/28 | 10.84.11.14     | 0c:c4:7a:ab:ba:2e | true  | physi... |

At the **Servers** page, click the **Monitoring** tab to see detailed information regarding **CPU/Memory Information, Chassis State, Sensors, Interface Monitoring, File System, and Disk Usage**, see [Figure 68 on page 377](#).



Figure 68: Server Monitoring

The screenshot displays the Juniper Server Manager interface for server 'a3s27'. The 'Inventory' tab is selected, showing various monitoring sections:

- CPU/Memory Information:**

|                 |         |
|-----------------|---------|
| CPU Utilization | 0.86 %  |
| Memory Usage    | 2.18 %  |
| Memory Used     | 5.49 GB |
- Chassis State:**

|                      |          |
|----------------------|----------|
| Power Restore Policy | previous |
| System Power         | on       |
| Cooling Fan Fault    | false    |
| Front Panel Lockout  | inactive |
| Drive Fault          | false    |
| Chassis Intrusion    | inactive |
| Main Power Fault     | false    |
| Power Control Fault  | false    |
| Power Overload       | false    |
| Power Interlock      | inactive |
- Sensors:**

| Name            | Type        | Reading | Status |
|-----------------|-------------|---------|--------|
| CPU1 Temp       | temperature | 40 C    | ok     |
| CPU2 Temp       | temperature | 39 C    | ok     |
| PCH Temp        | temperature | 41 C    | ok     |
| System Temp     | temperature | 27 C    | ok     |
| Peripheral Temp | temperature | 38 C    | ok     |
- Interface Monitoring:**

| Name   | Tx Bytes | Tx Packets  | Rx Bytes | Rx Packets  |
|--------|----------|-------------|----------|-------------|
| em1    | 543.9 GB | 661,564,361 | 127.4 GB | 393,175,718 |
| em2    | 0 B      | 0           | 0 B      | 0           |
| p514p1 | 0 B      | 0           | 0 B      | 0           |
| p514p2 | 0 B      | 0           | 0 B      | 0           |
- File System:**

| Name                       | Type      | Size     | Used |
|----------------------------|-----------|----------|------|
| /dev/mapper/a3s27--vg-r... | lvm       | 2.8 TB   | 1 %  |
| /dev/sda1                  | partition | 235.3 MB | 31 % |
- Disk Usage:**

| Disk Name | Read  | Write    |
|-----------|-------|----------|
| sda       | 1 KB  | 27.7 KB  |
| sdb       | 410 B | 450 KB   |
| sdc       | 425 B | 447.2 KB |
| sdd       | 626 B | 153.6 KB |

At the **Servers** page, click the **Inventory** tab to see detailed information regarding **Overview of the server, Interface Information, CPU information, Memory, and FRU Information**, see [Figure 69 on page 378](#).

Figure 69: Server Inventory

The screenshot displays the Juniper Server Manager interface for a server named 'a3b27'. The interface is organized into several sections:

- Overview:**
  - Hardware Model: x86\_64
  - Physical Processors: 2
  - Operating System: Ubuntu
  - OS Family: Debian
  - OS Version: 14.04
  - Virtual Machine: physical
  - Uptime (secs): 2660971
  - Interface Controller Ports: 4
  - Total Disks: 4
- CPU:**
  - Model: Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz
  - Clock Speed (MHz): 1200
  - Threads Per Core: 2
  - Processor Count: 32
  - VCPU Count: 8
- Memory:**
  - Dimms: 16
  - Memory Speed (MHz): 2133
  - Dimm Size (MB): 16384
  - Mem Type:
  - Total Memory (MB): 257597
  - Swap Size (MB): 262028
- Interface Information:**

| Name           | IP Address | MAC Address       | Speed (mbps) |
|----------------|------------|-------------------|--------------|
| em1            | N/A        | 0cc4:7aab:ba:2e   | 1000         |
| em2            | N/A        | 0cc4:7aab:ba:2f   | 0            |
| p514p1         | N/A        | 90e2:ba:b8:4f:30  | 0            |
| p514p2         | N/A        | 90e2:ba:b8:4f:31  | 0            |
| pkt0           | N/A        | c2:ea:2f:18:52:40 | 10           |
| pkt1           | N/A        | N/A               | 0            |
| pkt2           | N/A        | N/A               | 0            |
| pkt3           | N/A        | N/A               | 0            |
| tap1e7bde74_b8 | N/A        | 2a:02:d6:d4:95:fc | 0            |
| tap9520c49b_a2 | N/A        | 86:33:b7:e1:f4:06 | 0            |
- FRU Information:**

| Description               | Product Name | Chassis Type |
|---------------------------|--------------|--------------|
| Builtin FRU Device (ID 0) | N/A          | N/A          |

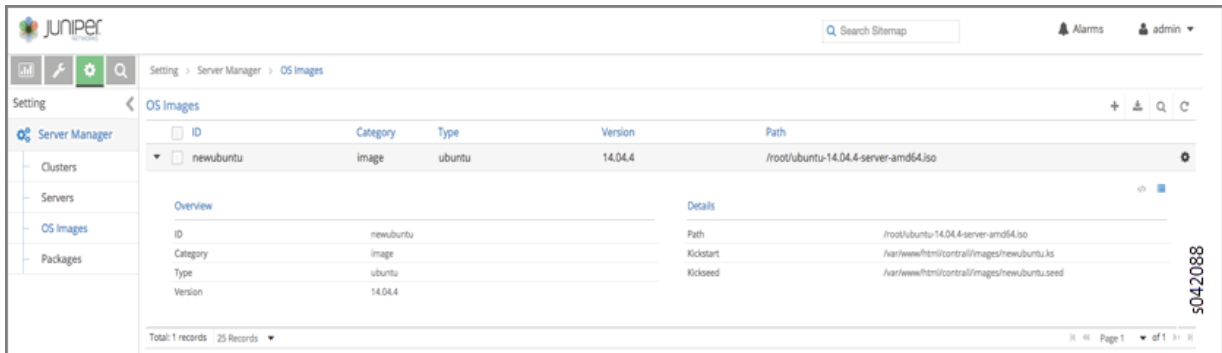
## Configuring Images and Packages

Use the sidebar **Images and Packages** options to configure the software images and packages to be used by the Server Manager. Images are typically used to reimage clusters with an operating system version. Packages are used to provision clusters with a Contrail setup.

Both areas of the Server Manager user interface operate in a similar fashion. The figure shows the **Images** section. The **Packages** section has similar options.

Select **Images**. The Images page is displayed, see [Figure 70 on page 379](#).

Figure 70: Servers OS Images

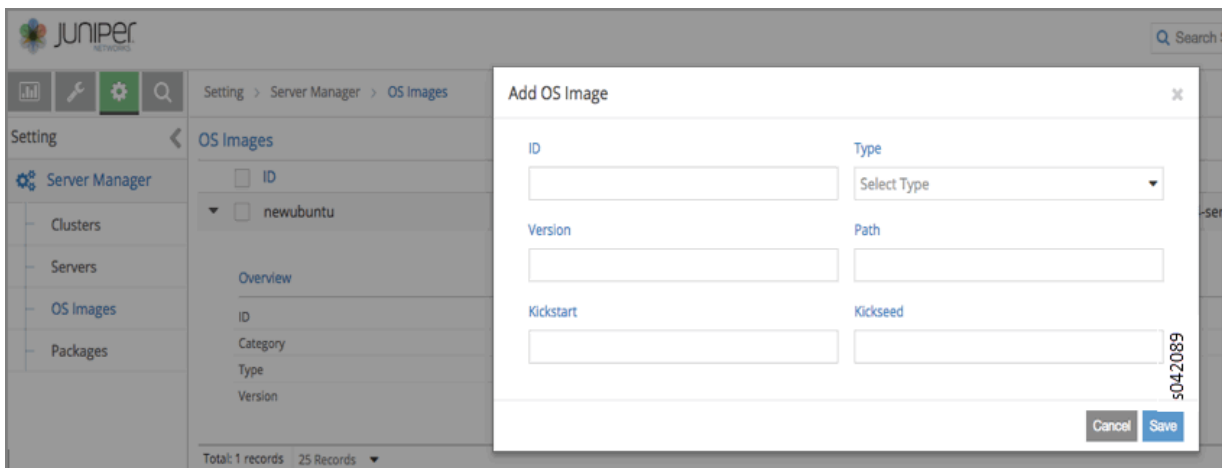


## Add New Image or Package

To add a new image or package, on the respective **Images** or **Packages** page, click the plus (+) icon in the upper right header. The **Add Image** window is displayed. Enter the information for the new image (or package) and click **Save** to add the new item to the list of configured items, see [Figure 71 on page 379](#).

**NOTE:** The path field requires the path of the image where it is located on the server upon which the server-manager process is running.

Figure 71: Add OS Image



## Selecting Server Manager Actions for Clusters

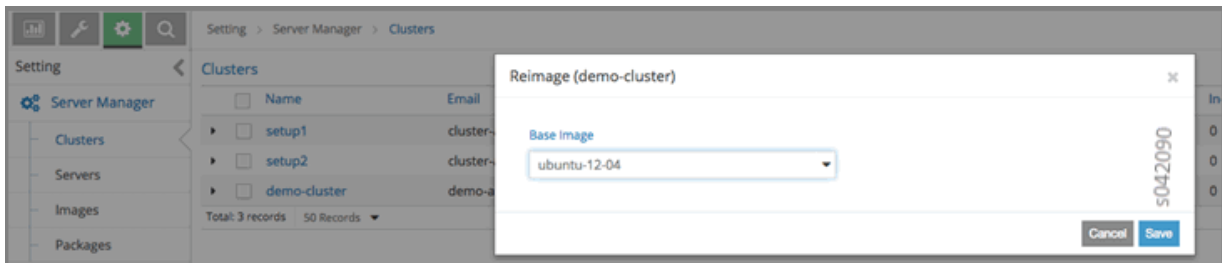
After all aspects of a cluster are configured, you can select actions for the Server Manager to perform on the cluster, such as **Reimage** or **Provision**.

## Reimage a Cluster

Select **Setting > Servers > Clusters**. The **Clusters** window is displayed. Click the right side gear wheel icon of the cluster to be reimaged, then select **Reimage** from the action menu.

The **Reimage** dialog box is displayed, as shown. Verify that the correct image is selected in the **Default Image** field, then click **Save** to initiate the reimage action, see [Figure 72 on page 380](#).

Figure 72: Reimage Cluster

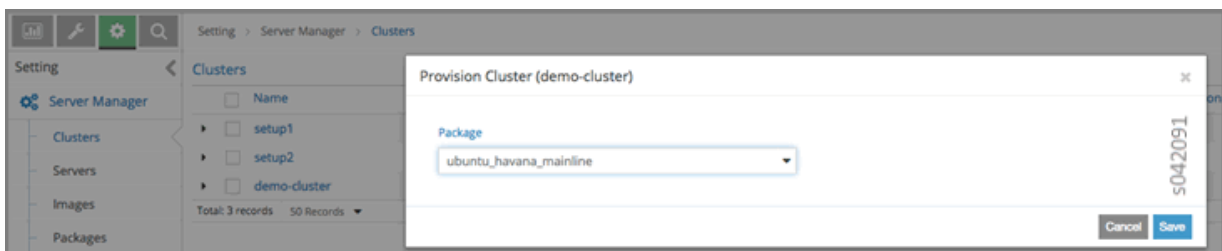


## Provision a Cluster

The process to provision a cluster is similar to the process to reimage a cluster. Select **Setting > Servers > Clusters**. The **Clusters** window is displayed. Click the right side gear wheel icon of the cluster to be provisioned, then select **Provision** from the action menu.

The **Provision Cluster** dialog box is displayed, as shown. Verify that the correct package for provisioning is selected in the **Default Package** field, then click **Save** to initiate the provisioning action, see [Figure 73 on page 380](#).

Figure 73: Provision Cluster



## SEE ALSO

*Using Server Manager to Automate Provisioning*

## Installing and Using Server Manager Lite

### IN THIS SECTION

- [Server Manager Lite Overview | 381](#)
- [Installing Server Manager Lite | 382](#)
- [Provisioning Using SM-Lite with Contrail 4.0 | 382](#)
- [Displaying the Cluster Status | 383](#)
- [Displaying the SM-Lite Installation and Provisioning Log Files | 383](#)
- [Contrail Provisioning Log Files | 383](#)

This topic describes how to install and troubleshoot Server Manager Lite.

### Server Manager Lite Overview

Server Manager Lite (SM-Lite), is a streamlined version of the Server Manager software that does not include the reimage function.

SM-Lite supports the Server Manager functions of provisioning, monitoring, inventory, and WebUI. SM-Lite is intended to replace fab command provisioning. It allows easy deployment of Contrail provisioning and enables developers to work in isolated environments for Contrail provisioning.

SM-Lite eliminates installation and configuration of DHCP, DNS, and Cobbler services. Additionally, SM-Lite installation setup scripts are enhanced to reduce installation time.

SM-Lite provides a single command to install SM-Lite and provision a Contrail cluster.

SM-Lite introduces additional capabilities into Server Manager. The SM-Lite package is part of the Contrail Server Manager installer Debian package (`contrail-server-manager-installer_<version string>.deb`).

SM-Lite works with or without having a separate node for the SM-Lite installation, it can be installed on any Contrail node, but it is recommended to install it on the config node.

SM-Lite preserves the existing Server Manager WebUI functionality and it can be run on the same node as the Contrail WebUI. Because of that, the default port for the Server Manager WebUI has been changed to port 9080.

It is important to note that the code base used for SM-Lite and Server Manager is common. Therefore, any changes or enhancements made to Server Manager provisioning functionality are automatically available in the SM-Lite software.

## Installing Server Manager Lite

The SM-Lite package is included as part of the Server Manager installer package.

The installer package also has other packages such as Server Manager, Server Manager client, Server Manager WebUI, and Server Manager inventory. Before provisioning commands can be executed using SM-Lite, you need to install the Server Manager installer package.

Use the following command to install the Server Manager installer package.

```
dpkg -i <contrail-server-manager-installer-deb>
```

After the Server Manager installer package is installed, all necessary Server Manager packages, scripts, and so on are made available on the server where it is installed. You can then start using Server Manager Lite commands.

## Provisioning Using SM-Lite with Contrail 4.0

For Contrail 4.0, to provision the target systems, use the script.

The `provision_containers.sh` script performs the following functions:

1. Installs SM-Lite.

Uses the `setup.sh` installation script with the `-smlite` option to install the SM-Lite package (`contrail-server-manager-lite_<version-sku>_all.deb`) and all other needed packages on the system.

2. Prepares the cluster for Contrail provisioning.

Translates the parameters in the `testbed.py` file into Server Manager objects and stores them in the Server Manager database. This specifies the servers in the cluster and the configuration parameters. The cluster-id value is used, if it is specified.

3. Performs a pre-check on the target systems to ensure that they are ready for running provisioning. SM-Lite uses from the Contrail package to provision the Contrail cluster.

4. This step issues provisioning commands for the cluster with the given Contrail package.

Server Manager Lite can be installed on any node. We recommend that you install it on the config node. Server Manager Lite can be installed on a separate node other than the Contrail cluster nodes.

The Server Manager WebUI default port is 9080. You can change the port by editing the `/etc/contrail/config.global.sm.js` file, and then restarting the `supervisor-webui-sm` process.

## Displaying the Cluster Status

The `server-manager cluster -detail` command displays the provisioning status of a cluster by role and by role progress.

Use the `server-manager status server` command to display the current status of the servers.

## Displaying the SM-Lite Installation and Provisioning Log Files

Log files that provide information during installation and use of SM-Lite software are available at:

- `/var/log/contrail/install_logs/install_<timestamp>.log` (SM-Lite install)
- `/var/log/contrail/install_logs/provision_<timestamp>.log` (provisioning command logs)
- `testbed_parser.log` and `preconfig.log`

## Contrail Provisioning Log Files

For each Puppet run, log files are automatically uploaded to the Server Manager at the following locations:

- `http: <sm-lite-ip-address>/logs`
- `/var/log/contrail_server_manager/ <target>/ <timestamp>.log`
- `/var/log/contrail/*`

You can also display the status of the processes and services using the `contrail-status` command.

## RELATED DOCUMENTATION

*Using Server Manager to Automate Provisioning*

*Using the Server Manager Web User Interface*

# Extending Contrail to Physical Routers, Bare Metal Servers, Switches, and Interfaces

## IN THIS CHAPTER

- [Using ToR Switches and OVSDB to Extend the Contrail Cluster to Other Instances | 384](#)
- [Configuring High Availability for the Contrail OVSDB ToR Agent | 397](#)
- [Using Device Manager to Manage Physical Routers | 404](#)
- [SR-IOV VF as the Physical Interface of vRouter | 436](#)
- [Using Gateway Mode to Support Remote Instances | 438](#)
- [REST APIs for Extending the Contrail Cluster to Physical Routers, and Physical and Logical Interfaces | 440](#)

## Using ToR Switches and OVSDB to Extend the Contrail Cluster to Other Instances

### IN THIS SECTION

- [Support for ToR Switch and OVSDB Overview | 385](#)
- [ToR Services Node \(TSN\) | 385](#)
- [Contrail ToR Agent | 385](#)
- [Using the Web Interface to Configure ToR Switch and Interfaces | 388](#)
- [Configuration Parameters for Provisioning ToR and TSN | 390](#)
- [Prerequisite Configuration for QFX5100 Series Switch | 394](#)
- [Changes to Agent Configuration File | 396](#)
- [REST APIs | 397](#)



## Support for ToR Switch and OVSDB Overview

Contrail Releases 2.1 and greater support extending a cluster to include bare metal servers and other virtual instances connected to a top-of-rack (ToR) switch that supports the Open vSwitch Database Management (OVSDB) protocol. The bare metal servers and other virtual instances can belong to any of the virtual networks configured in the Contrail cluster, facilitating communication with the virtual instances running in the cluster. Contrail policy configurations can be used to control this communication.

The OVSDB protocol is used to configure the ToR switch and to import dynamically-learned addresses. VXLAN encapsulation is used in the data plane communication with the ToR switch.

### ToR Services Node (TSN)

A ToR services node (TSN) can be provisioned as a role in the Contrail system. The TSN acts as the multicast controller for the ToR switches. The TSN also provides DHCP and DNS services to the bare metal servers or virtual instances running behind ToR switch ports.

The TSN receives all the broadcast packets from the ToR switch, and replicates them to the required compute nodes in the cluster and to other EVPN nodes. Broadcast packets from the virtual machines in the cluster are sent directly from the respective compute nodes to the ToR switch.

The TSN can also act as the DHCP server for the bare metal servers or virtual instances, leasing IP addresses to them, along with other DHCP options configured in the system. The TSN also provides a DNS service for the bare metal servers. Multiple TSN nodes can be configured in the system based on the scaling needs of the cluster.

### Contrail ToR Agent

A ToR agent provisioned in the Contrail cluster acts as the OVSDB client for the ToR switch, and all of the OVSDB interactions with the ToR switch are performed by using the ToR agent. The ToR agent programs the different OVSDB tables onto the ToR switch and receives the local unicast table entries from the ToR switch.

The ToR agent receives the configuration information for the ToR switch, translates the Contrail configuration to OVSDB, and populates the relevant OVSDB table entries in the ToR switch.

Contrail recognizes the ToR after you configure `tsn` and `toragent` roles.

The typical practice is to run the ToR agent on the TSN node.

### Configuration Model

[Figure 74 on page 386](#) depicts the configuration model used in the system.

Figure 74: Configuration Model

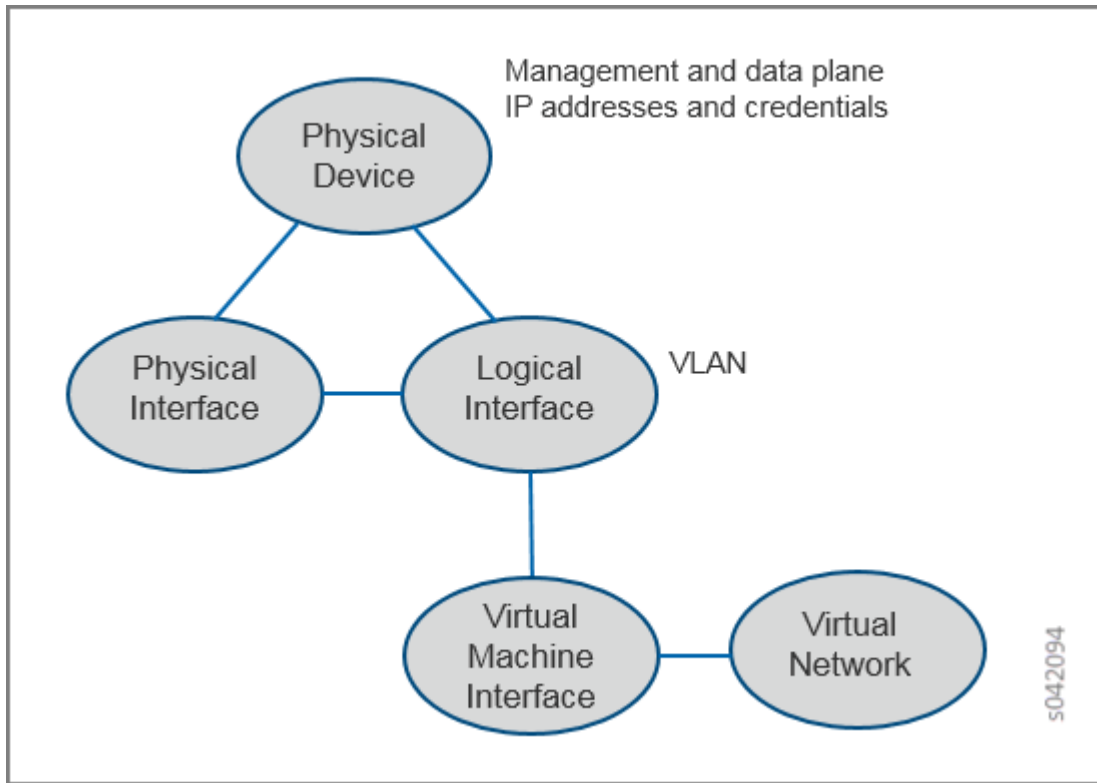


Table 29 on page 386 maps the Contrail configuration objects to the OVSDB tables.

Table 29: Contrail Objects in the OVSDB

| Contrail Object             | OVSDB Table                                    |
|-----------------------------|--|
| Physical device             | Physical switch                                |
| Physical interface          | Physical port                                  |
| Logical interface           | <VLAN physical port> binding to logical switch |
| Virtual networks            | Logical switch                                 |
| Layer 2 unicast route table | Unicast remote and local table                 |

**Table 29: Contrail Objects in the OVSDB (Continued)**

| Contrail Object | OVSDB Table                |
|-----------------|----------------------------|
|                 | Multicast remote table     |
|                 | Multicast local table      |
|                 | Physical locator table     |
|                 | Physical locator set table |

### Control Plane

The ToR agent receives the EVPN route entries for the virtual networks in which the ToR switch ports are members, and adds the entries to the unicast remote table in the OVSDB.

MAC addresses learned in the ToR switch for different logical switches (entries from the local table in OVSDB) are propagated to the ToR agent. The ToR agent exports the addresses to the control node in the corresponding EVPN tables, which are further distributed to other controllers and subsequently to compute nodes and other EVPN nodes in the cluster.

The TSN node receives the replication tree for each virtual network from the control node. It adds the required ToR addresses to the received replication tree, forming its complete replication tree. The other compute nodes receive the replication tree from the control node, whose tree includes the TSN node.

### Data Plane

The data plane encapsulation method is VXLAN. The virtual tunnel endpoint (VTEP) for the bare metal end is on the ToR switch.

Unicast traffic from bare metal servers is VXLAN-encapsulated by the ToR switch and forwarded, if the destination MAC address is known within the virtual switch.

Unicast traffic from the virtual instances in the Contrail cluster is forwarded to the ToR switch, where VXLAN is terminated and the packet is forwarded to the bare metal server.

Broadcast traffic from bare metal servers is received by the TSN node. The TSN node uses the replication tree to flood the broadcast packets in the virtual network.

Broadcast traffic from the virtual instances in the Contrail cluster is sent to the TSN node, which replicates the packets to the ToR switches.

## Using the Web Interface to Configure ToR Switch and Interfaces

The Contrail Web user interface can be used to configure a ToR switch and the interfaces on the switch. To add a switch, select **Configure > Physical Devices > Physical Routers**.

The **Physical Routers** list is displayed.

Click the + symbol to open the **Add** menu. From the **Add** menu you can select one of the following:

- **Add OVSDB Managed ToR**
- **Add Netconf Managed Physical Router**
- **CPE Router**
- **Physical Router**

To add a physical ToR, select **Add OVSDB Managed ToR**. The **Create** window is displayed, as shown in [Figure 75 on page 389](#). Enter the IP address and VTEP address of the ToR switch . Also configure the TSN and ToR agent names for the ToR.

Figure 75: Create OVSDB Managed ToR

The screenshot shows a 'Create' dialog box with a close button (X) in the top right corner. The dialog is titled 'Create' and contains two tabs: 'OVSDB Managed ToR' (selected) and 'Permissions'. Below the tabs, there are several input fields and checkboxes:

- Name:** A text input field.
- Vendor:** A text input field.
- Model:** A text input field.
- Management IP:** A text input field.
- VTEP Address:** A text input field.
- TOR Agent(s):** Two dropdown menus, each with the text 'Select or Enter TOR Agent Name' and a downward arrow.
- TSN(s):** Two dropdown menus, each with the text 'Select or Enter TSN Name' and a downward arrow.
- SNMP Monitored:** A checkbox that is currently unchecked.

At the bottom right of the dialog, there are two buttons: 'Cancel' and 'Save'. A vertical ID 's042093' is visible on the right side of the dialog.

To add the logical interfaces to be configured on the ToR switch, select **Configure > Physical Devices > Interfaces**.

The **Physical Routers** list is displayed. Click the + symbol. The **Add Interface** window is displayed, as shown in [Figure 76 on page 390](#).

At **Add Interface**, enter the name of the logical interface. The name must match the name on the ToR, for example, ge-0/0/0.10. Also enter other logical interface configuration parameters, such as VLAN ID, MAC address, and IP address of the bare metal server and the virtual network to which it belongs.

Figure 76: Add Interface

**Add Interface**

Type: Logical | Name: | Parent: No Physical Router found

▼ Logical Interface Properties

Logical Interface Type: Server

Vlan ID:

Virtual Network: Project1-VN09E776A (default-domain:Project1) (48.248.113...)

Server Details

| Server | IP |   |
|--------|----|---|
|        |    | + |

Enter or Choose mac | Auto Allocate or Enter an IP | - | +

5042092

Cancel Save

### Configuration Parameters for Provisioning ToR and TSN

This section presents the configuration parameters for different methods of provisioning ToR and TSN.

The following information can be provided for each ToR agent.

- IP address of the ToR
- a unique numeric identifier for the ToR
- a unique (optional) name for the ToR Agent
- the OVS protocol (TCP or SSL)
- the OVS port
  - when OVS protocol is TCP, port indicates the TCP port to connect on the ToR

- when OVS protocol is pssl, port indicates the SSL port on which the ToR agent listens for connections from the TOR
- TSN IP address of the ToR
- name of the TSN node
- IP address of the data tunnel endpoint
- HTTP server port of the ToR Agent using which introspect data can be checked
- vendor name for ToR (optional)
- product name of ToR switch (optional)
- OVS keepalive timeout (optional)

### Inventory Format ToR and TSN

Indicate the compute node to act as TSN.

```
[contrail-computes]
1.1.1.7 ctrl_data_ip=10.1.1.7 tsn_mode=True
```

```
tor_agent = { 'host1': [
                { 'tor_ip': '10.xxx.221.35',
                  'tor_agent_id': '1',
                  'tor_agent_name': 'node-1',
                  'tor_ovs_protocol': 'tcp',
                  'tor_ovs_port': '9999',
                  'tor_tsn_ip': '10.xxx.221.33',
                  'tor_tsn_name': 'tsn1',
                  'tor_name': 'contrail-tor-1',
                  'tor_tunnel_ip': '5.5.5.5',
                  'tor_http_server_port': '9090',
                  'tor_vendor_name': 'Juniper',
                  'tor_product_name': 'QFX5100',
                  'tor_agent_ovs_ka': '1000'
                } ,
                { ... }
            ],
```

```

    'host2': [ ... ]
  }

```

## JSON Format ToR and TSN

If you are provisioning using JSON, the following example is the JSON format.

For ToR in server.json.

```

{
  "server": [
    {
      "id": "new-server",
      "parameters": {
        "top_of_rack": {
          "switches": [
            {
              "agent_id": "1",
              "ip": "10.x.141.84",
              "tunnel_ip": "10.xx.141.84",
              "name": "TOR1",
              "tsn_name": "TSN1",
              "agent_name": "AGENT1",
              "ovs_port": "6632",
              "agent_ovs_ka": "1000",
              "ovs_protocol": "tcp",
              "http_server_port": "9912",
              "vendor_name": "Juniper"
            },
            {
              "agent_id": "2",
              "ip": "10.xx.141.83",
              "tunnel_ip": "10.xx.141.83",
              "name": "TOR2",
              "ovs_port": "6632",
              "ovs_protocol": "tcp",
              "http_server_port": "9913",
              "vendor_name": "Juniper"
            }
          ]
        }
      }
    }
  ]
}

```



```
    ],
  },
```

For TSN in server.json.

```
{
  "server" : [
    {
      "id": "new-server",
      "parameters" : {
        "provision": {
          "contrail_4": {
            "tsn_mode": false
          }
        }
      }
    }
  ]
}
```

## Testbed.py Format ToR and TSN

Starting with Contrail 4.0, if you are provisioning using SM-Lite, you can provision with JSON or testbed.py. The following is the testbed.py format.

The ToR agent and TSN can be provisioned using the testbed.py configured with the following:

- The env.roledef section is configured with the tsn and toragent roles. The hosts for these roles should also host a compute node.
- The env.tor\_agent section should be present and configured.

For ToR:

```
#env.tor_agent = {host10:[{
#           'tor_ip':'10.xxx.217.39',
#           'tor_agent_id':'1',
#           'tor_agent_name':'nodexx-1',
#           'tor_type':'ovs',
#           'tor_ovs_port':'9999',
#           'tor_ovs_protocol':'tcp',
#           'tor_tsn_name':'nodec45',
#           'tor_name':'bng-contrail-qfx51-2',
```

```

#           'tor_tunnel_ip': '34.34.34.34',
#           'tor_vendor_name': 'Juniper',
#           'tor_product_name': 'QFX5100',
#           'tor_agent_http_server_port': '9010',
#           'tor_agent_ovs_ka': '10000',
#           ]]
#       }

```

For TSN:

```

env.roledefs = {
  'tsn': [host1], # Optional, Only to enable TSN. Only compute can support TSN
}

```

For more information, see <https://github.com/Juniper/contrail-controller/wiki/Baremetal-Support> .

## Prerequisite Configuration for QFX5100 Series Switch

When using the Juniper Networks QFX5100 Series switches, ensure the following configurations are made on the switch before extending the Contrail cluster.

1. Enable OVSDB.
2. Set the connection protocol.
3. Identify the interfaces that are managed by means of OVSDB.
4. Configure the controller (in case pssl is used). If HA Proxy is used, use the address of the HA Proxy node and use the VIP when VRRP is used between multiple nodes running HA Proxy. The following is an example:

```

set interfaces lo0 unit 0 family inet address

set switch-options ovssdb-managed

set switch-options vtep-source-interface lo0.0

set protocols ovssdb interfaces

set protocols ovssdb passive-connection protocol tcp port

```

```
set protocols ovsdb controller <tor-agent-ip> inactivity-probe-duration 10000 protocol ssl
port <tor-agent-port>
```

5. When using SSL to connect, CA-signed certificates must be copied to the `/var/db/certs` directory in the QFX device. The following example shows one way to get the certificates. The following commands could be run on any server.

```
apt-get install openvswitch-common
ovs-pki init
ovs-pki req+sign vtep
scp vtep-cert.pem root@<qfx>:/var/db/certs
scp vtep-privkey.pem root@<qfx>:/var/db/certs
cacert.pem file will be available in /var/lib/openvswitch/pki/switchca, when the above are
done. This is the file to be provided in the above testbed (in env.ca_cert_file).
```

## Debug QFX5100 Configuration

You can use the following commands on the QFX switch to show the OVSDB configuration.

```
show ovsdb logical-switch

show ovsdb interface

show ovsdb mac

show ovsdb controller

show vlans
```

You can use the agent introspect on the ToR agent and the TSN nodes to show the configuration and operational state of these modules.

- The TSN module is like any other `contrail-vrouter-agent` on a compute node, with introspect access available on port 8085 by default. Use the introspect on port 8085 to view operational data such as interfaces, virtual network, and VRF information, along with their routes.
- The port on which the ToR agent introspect access is available is in the configuration file provided to the `contrail-tor-agent`. This provides the OVSDB data available through the client interface, apart from the other data available in a Contrail Agent.

## Changes to Agent Configuration File

You can make changes to the agent features by making changes in the configuration file.

In the `/etc/contrail/contrail-vrouter-agent.conf` file for TSN, the `agent_mode` option is available in the `DEBUG` section to configure the agent to be in TSN mode.

```
agent_mode = tsn
```

The following are typical configuration items in a ToR agent configuration file.

```
[DEFAULT]

agent_name = noded2-1 # Name (formed with hostname and TOR id from below)

agent_mode = tor # Agent mode

http_server_port=9010 # Port on which Introspect access is available

[TOR]

tor_ip=<ip> # IP address of the TOR to manage

tor_id=1 # Identifier for ToR Agent.

tor_type=ovs # ToR management scheme - only "ovs" is supported

tor_ovs_protocol=tcp # IP-Transport protocol used to connect to TOR, can be tcp or pssl

tor_ovs_port=port # OVS server port number on the ToR

tsn_ip=<ip> # IP address of the TSN

tor_keepalive_interval=10000 # keepalive timer in ms

ssl_cert=/etc/contrail/ssl/certs/tor.1.cert.pem # path to SSL certificate on TOR Agent, needed
for pssl

ssl_privkey=/etc/contrail/ssl/private/tor.1.privkey.pem # path to SSL private key on TOR Agent,
needed for pssl
```

```
ssl_cacert=/etc/contrail/ssl/certs/cacert.pem # path to SSL CA cert on the node, needed for pssl
```

## REST APIs

For information regarding REST APIs for physical routers and physical and logical interfaces, see "[REST APIs for Extending the Contrail Cluster to Physical Routers, and Physical and Logical Interfaces](#)" on page 440.

## RELATED DOCUMENTATION

[REST APIs for Extending the Contrail Cluster to Physical Routers, and Physical and Logical Interfaces | 440](#)

[Using Device Manager to Manage Physical Routers | 404](#)

[Configuring High Availability for the Contrail OVSDB ToR Agent | 397](#)

## Configuring High Availability for the Contrail OVSDB ToR Agent

### IN THIS SECTION

- [Overview: High Availability for a ToR Switch | 397](#)
- [High Availability Solution for Contrail ToR Agent | 398](#)
- [Failover Methodology Description | 399](#)
- [Failure Scenarios | 399](#)
- [Redundancy for HAProxy | 401](#)
- [Configuration for ToR Agent High Availability | 402](#)

This topic describes how high availability can be configured for the Contrail ToR agent.

### Overview: High Availability for a ToR Switch

In Contrail Release 2.20 and later, high availability can be configured for the Contrail ToR agent.

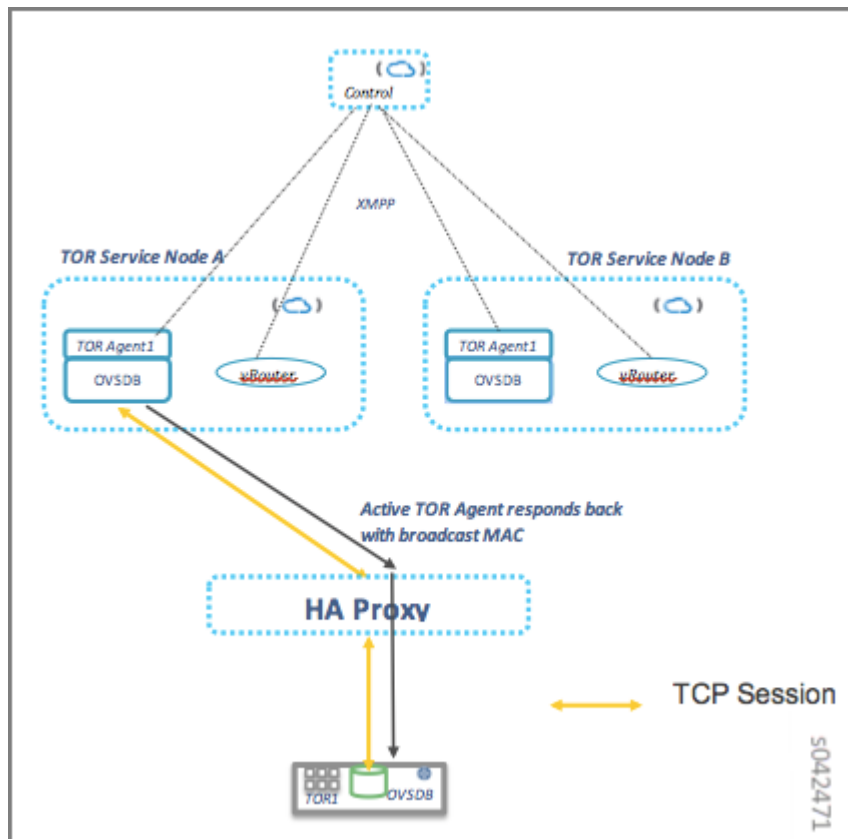
When a top-of-rack (ToR) switch is managed through the Open vSwitch Database (OVSDB) protocol by using a ToR agent on Contrail, a high availability configuration is necessary to maintain ToR agent redundancy. With ToR agent redundancy, if the ToR agent responsible for a ToR switch is unable to act as the vRouter agent for the ToR switch, due to any failure condition in the network or the node, then another ToR agent takes over and manages the ToR switch.

ToR agent redundancy (high availability) is achieved using HAProxy. HAProxy is an open source, reliable solution that offers high availability and proxy service for TCP applications. The solution uses HAProxy to initiate an SSL connection from the ToR switch to the ToR agent. This configuration ensures that the ToR switch is connected to exactly one active ToR agent at any given point in time.

## High Availability Solution for Contrail ToR Agent

The following figure illustrates the method for achieving high availability for the ToR agent in Contrail.

Figure 77: High Availability Solution for Contrail ToR Agent



The following describes the events shown in the figure:

- ToR agent redundancy is achieved using HAProxy.

- Two ToR agents are provisioned on different TSN nodes, to manage the same ToR switch.
- Both ToR agents created in the cluster are active and get the same information from the control node.
- HAProxy monitors these ToR agents.
- An SSL connection is established from the ToR switch to the ToR agent, via HAProxy.
- HAProxy selects one ToR agent to establish the SSL connection (e.g., ToR Agent 1 running on TSN A).
- Upon connection establishment, this ToR Agent adds the ff:ff:ff:ff:ff:ff broadcast MAC address in the OVSDB with its own TSN IP address.
- The ToR Agent sends the MAC addresses of the bare metal servers learned by the ToR switch to the control node using XMPP.
- The control node reflects the addresses to other ToR agents and vRouter agents.

## Failover Methodology Description

The ToR switch connects to the HAProxy that is configured to use one of the ToR agents on the two ToR services nodes (TSNs). An SSL connection is established from the ToR switch to the ToR agent, making that agent the active ToR agent. The active ToR agent is responsible for managing the OVSDB on the ToR switch. It configures the OVSDB tables based on the configuration. It advertises the MAC routes learned on the ToR switch as Ethernet VPN (EVPN) routes to the Contrail controller. It also programs any routes learned by means of EVPN over XMPP, southbound into OVSDB on the ToR switch.

The active ToR agent also advertises the multicast route (ff:ff:ff:ff:ff:ff) to the ToR switch, ensuring that there is only one multicast route in OVSDB pointing to the active TSN.

Both the ToR agents, active and standby, receive the same configuration from the control node, and all routes are synchronized by means of BGP.

After the SSL connection is established, keepalive messages are exchanged between the ToR switch and the ToR agent. The messages can be sent from either end and are responded to from the other end. When any message exchange is seen on the connection, the keepalive message is skipped for that interval. When the ToR switch sees that keepalive has failed, it closes the current SSL session and attempts to reconnect. When the ToR agent side sees that keepalive has failed, it closes the SSL session and retracts the routes it exported to the control node.

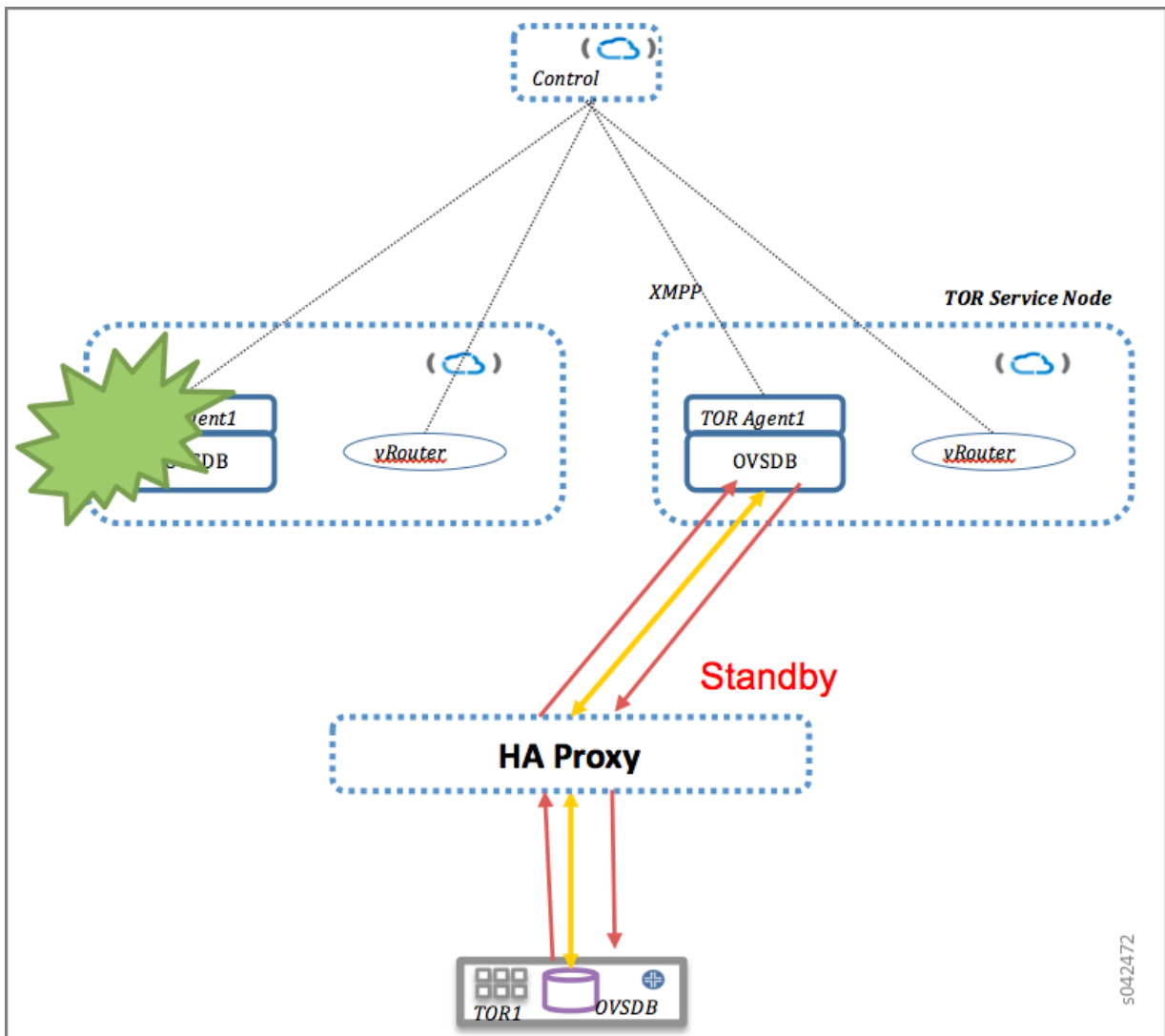
## Failure Scenarios

Whenever the HAProxy cannot communicate with the ToR agent, a new SSL connection from the ToR switch is established to the other ToR agent.

HAProxy communication failures can occur under several scenarios, including:

- The node on which the ToR agent is running goes down or fails.
- The ToR agent crashes.
- A network or other issue prevents or interrupts HAProxy communication with the ToR agent.

Figure 78: Failure Scenarios



When a connection is established to the other ToR agent, the new ToR agent does the following:

- Updates the multicast route in OVSDB to point to the new TSN.
- Gets all of the OVSDB entries.



- Audits the data with the configurations available.
- Updates the database.
- Exports entries from the OVSDB local table to the control node.

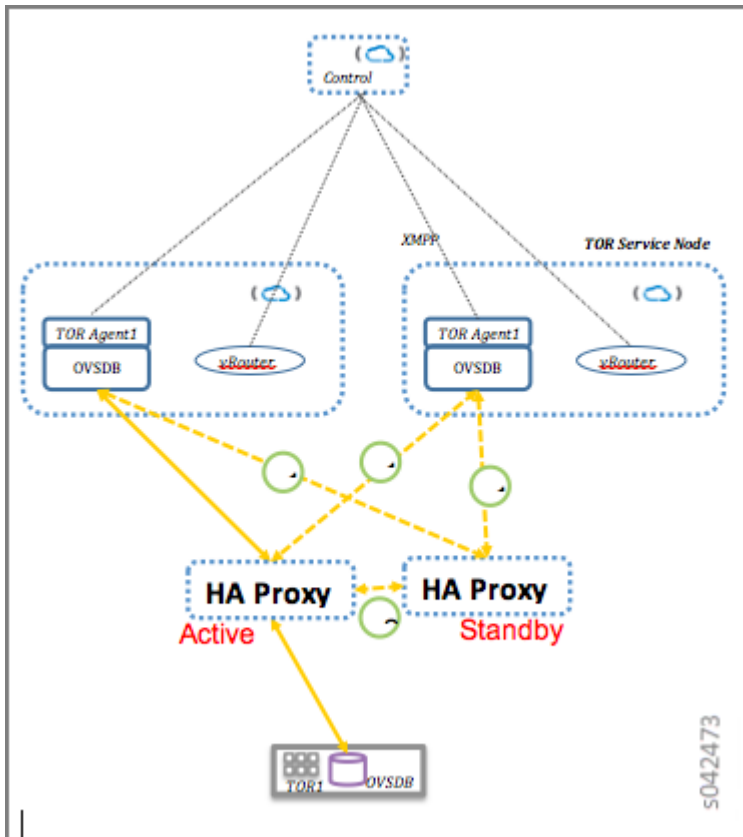
Because the configuration and routes from the control node are already synchronized to the new ToR Services Node (TSN), the new TSN can immediately act on the broadcast traffic from the ToR switch. Any impact to the service is only for the time needed for the SSL connection to be set up and for programming the multicast and unicast routes in the OVSDB.

When the SSL connection goes down, the ToR agent retracts the routes exported. Also, if the Extensible Messaging and Presence Protocol (XMPP) connection between the ToR agent and the control node goes down, the control node removes the routes exported by the ToR agent. In these scenarios, the entries from the OVSDB local table are retracted and then added back from the new ToR agent.

## **Redundancy for HAProxy**

In a high availability configuration, multiple HAProxy nodes are configured, with Virtual Router Redundancy Protocol (VRRP) running between them. The ToR agents are configured to use the virtual IP address of the HAProxy nodes to make the SSL connection to the controller. The active TCP connections go to the virtual IP master node, which proxies them to the chosen ToR agent. A ToR agent is chosen based on the number of connections from the HA Proxy to that node (the node with lower number of connections gets the new connection) and can be controlled through configuration of the HAProxy.

Figure 79: Redundancy for HAProxy



If the HAProxy node fails, a standby node becomes the virtual IP master and sets up the connections to the ToR agents. The SSL connections are reestablished, following the same methods discussed earlier.

### Configuration for ToR Agent High Availability

To get the required configuration downloaded from the control node to the TSN agent and to the ToR agent, the physical router node must be linked to the virtual router nodes that represent the two ToR agents and the two TSNs.

In the Contrail Web user interface select **Configure > Physical Devices > Physical Routers**. In the **Physical Routers** window, click the + icon. The **Add OVSDB Managed ToR** window is displayed. See [Figure 80 on page 403](#).

Figure 80: Add OVSDB Managed ToR Window

The screenshot shows a configuration window titled "Add OVSDB Managed ToR". The window contains the following fields and options:

- Name:** A text input field.
- Vendor:** A text input field.
- Model:** A text input field.
- Management IP:** A text input field.
- VTEP Address:** A text input field.
- TOR Agent(s):** Two dropdown menus, each with the text "Select or Enter Name".
- TSN(s):** Two dropdown menus, each with the text "Select or Enter Name".
- SNMP Monitored:** A checked checkbox.
- SNMP Settings:** A section with a dropdown arrow and two radio buttons labeled "2" and "3". The "2" radio button is selected.
- Buttons:** "Cancel" and "Save" buttons at the bottom right.

Enter a name to create an entry for the ToR switch, enter the ToR switch management IP address, and enter the virtual tunnel endpoint (VTEP) address. The router name should match the hostname of the ToR switch. Both ToR agents and their respective TSN nodes can be configured here.

## RELATED DOCUMENTATION

[Using ToR Switches and OVSDB to Extend the Contrail Cluster to Other Instances](#) | 384

## Using Device Manager to Manage Physical Routers

### IN THIS SECTION

- [Support for Physical Routers Overview | 404](#)
- [Configuration Model | 404](#)
- [Alternate Ways to Configure a Physical Router | 407](#)
- [Device Manager Configurations | 407](#)
- [Prerequisite Configuration Required on MX Series Device | 408](#)
- [Configuration Scenarios | 408](#)
- [Device Manager Functionality | 409](#)
- [Dynamic Tunnels | 409](#)
- [Extending the Public Network | 417](#)
- [Ethernet VPN Configuration | 419](#)
- [Floating IP Addresses and Source Network Address Translation for Guest Virtual Machines and Bare Metal Servers | 420](#)
- [Samples of Generated Configurations for an MX Series Device | 430](#)

### Support for Physical Routers Overview

A configuration node daemon named Device Manager can be used to manage physical routers in the Contrail system.

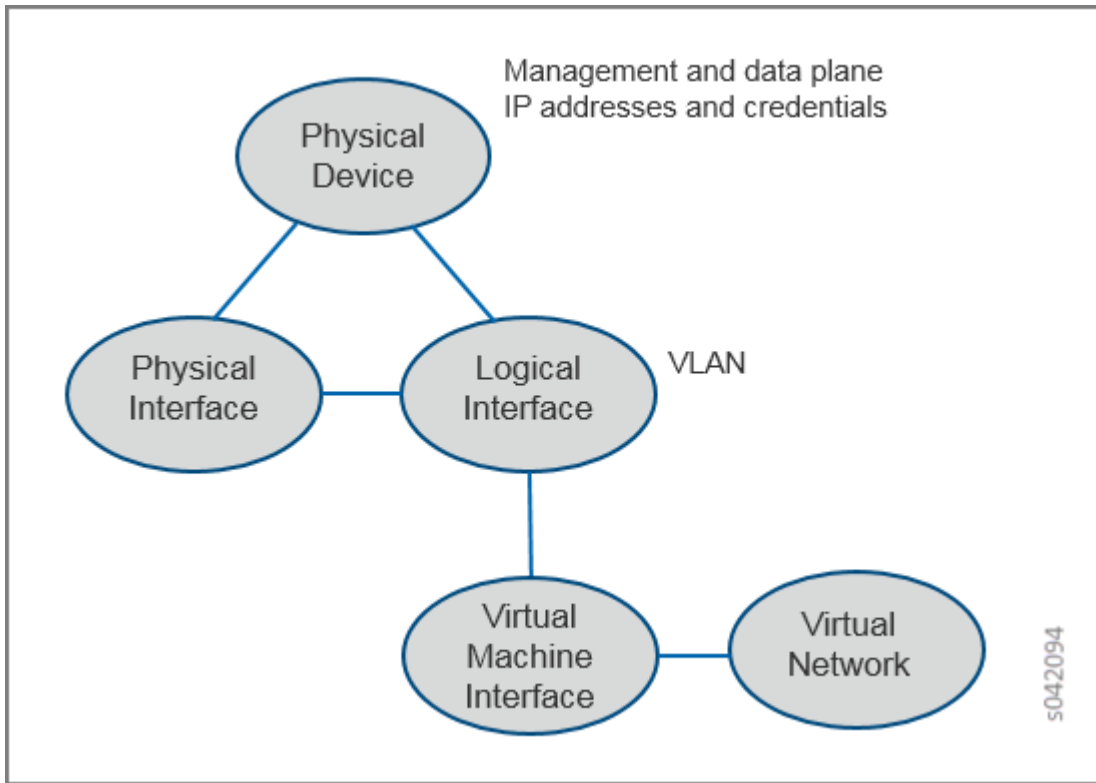
The Device Manager daemon listens to configuration events from the API server, creates any necessary configurations for all physical routers it is managing, and programs those physical routers.

You can extend a cluster to include physical Juniper Networks MX Series routers and other physical routers that support the Network Configuration (NETCONF) protocol. You can configure physical routers to be part of any of the virtual networks configured in the Contrail cluster, facilitating communication between the physical routers and the Contrail control nodes. Contrail policy configurations can be used to control this communication.

### Configuration Model

[Figure 81 on page 405](#) depicts the configuration model used in the system. The Physical Router, Physical Interface, and Logical Interface all represent physical router entities.

Figure 81: Contrail Configuration Model



### Configuring a Physical Router

The Contrail Web user interface can be used to configure a physical router into the Contrail system. Select **Configure > Physical Devices > Physical Routers** to create an entry for the physical router and provide the router's management IP address and user credentials.

The following shows how a Juniper Networks MX Series device can be configured from the Contrail Web user interface.

Figure 82: Add Physical Router Window

The screenshot shows a window titled "Add Netconf Managed Physical Router" with a close button (X) in the top right corner. The form contains the following fields and options:

- Name:** A single-line text input field.
- Vendor:** A single-line text input field.
- Model:** A single-line text input field.
- Management IP:** A single-line text input field.
- Netconf Username:** A single-line text input field.
- Password:** A single-line text input field.
- Junos Service Ports:** A checkbox that is checked.
- Port:** A section with a light gray background containing the text "Port" and a plus sign (+) on the right.
- SNMP Monitored:** A checkbox that is checked.

At the bottom right of the window, there are two buttons: "Cancel" (gray) and "Save" (blue). A vertical ID "s042493" is located on the right side of the window.

Select **Configure > Physical Devices > Interfaces** to add the logical interfaces to be configured on the router. The name of the logical interface must match the name on the router (for example, ge-0/0/0.10).

Figure 83: Add Interface Window

The screenshot shows the 'Add Interface' configuration window. At the top, there are three main fields: 'Type' (set to 'Logical'), 'Name' (empty), and 'Parent' (set to 'No Physical Router found'). Below these is a section titled 'Logical Interface Properties' which is expanded. This section contains several configuration options: 'Logical Interface Type' is set to 'Server'; 'Vlan ID' is an empty text box; 'Virtual Network' is set to 'Project1-VN09E776A (default-domain:Project1) (48.248.113...'. Underneath is a 'Server Details' section that includes a table with two columns, 'Server' and 'IP', and a '+' button to add more servers. Below the table are two dropdown menus: 'Enter or Choose mac' and 'Auto Allocate or Enter an IP', with '-' and '+' buttons. At the bottom right of the window are 'Cancel' and 'Save' buttons. A vertical ID '5042092' is visible on the right side of the window.

### Alternate Ways to Configure a Physical Router

You can also configure a physical router by using a Contrail REST API, see ["REST APIs for Extending the Contrail Cluster to Physical Routers, and Physical and Logical Interfaces"](#) on page 440.

### Device Manager Configurations

Device Manager can configure all of the following on a Juniper Networks MX Series device and other physical routers.

- Create configurations for physical interfaces and logical interfaces as needed.
- Create VRF table entries as needed by the configuration.
- Add interfaces to VRF tables as needed.
- Create public VRF tables corresponding to external virtual networks.

- Create BGP protocol configuration for internal or external BGP groups as needed and adding iBGP and eBGP peers in appropriate groups.
- Program route-target import and export rules as needed by policy configurations.
- Create policies and firewalls as needed.
- Configure Ethernet VPNs (EVPNs).

## Prerequisite Configuration Required on MX Series Device

Before using Device Manager to manage the configuration for an MX Series device, use the following Junos CLI commands to enable NETCONF on the device:

```
set system services netconf ssh
set system services netconf traceoptions file nc
set system services netconf traceoptions flag all
```

## Debugging Device Manager Configuration

If there is any failure during a Device Manager configuration, the failed configuration is stored on the MX Series device as a candidate configuration. An appropriate error message is logged in the local system log by the Device Manager.

The log level in the Device Manager configuration file should be set to INFO for logging NETCONF XML messages sent to physical routers.

## Configuration Scenarios

This section presents different configuration scenarios and shows snippets of generated MX Series configurations.

### Configuring Physical Routers Using REST APIs

For information regarding configurations using REST APIs, see "[REST APIs for Extending the Contrail Cluster to Physical Routers, and Physical and Logical Interfaces](#)" on page 440.

### Sample Python Script Using Rest API for Configuring an MX Device

Refer to the following link for a Python-based script for configuring required MX Series device resources in the Contrail system, using the VNC Rest API provided by Contrail.

[https://github.com/Juniper/contrail-controller/blob/master/src/config/utils/provision\\_physical\\_router.py](https://github.com/Juniper/contrail-controller/blob/master/src/config/utils/provision_physical_router.py)



## Device Manager Functionality

Device Manager auto configures physical routers when it detects associations in the Contrail database.

The following naming conventions are used for generating MX Series router configurations:

- Device Manager generated configuration group name: `__contrail__`
- BGP groups:
  - Internal group name: `__contrail__`
  - External group name: `__contrail_external`
- VRF name: `_contrai_{l2|l3}_{vn-id}_{vn-name}`
- NAT VRF name: `_contrai_{l2|l3}_{vn-id}_{vn-name}-nat`
- Import policy: `[vrf-name]-import`, Export policy: `[vrf-name]-export`
- Service set: `sv-[vrf-name]`
- NAT rules, SNAT: `sv-[vrf-name]-sn-rule`, DNAT: `sv-[vrf-name]-dn-rule`
- SNAT term name: `term_[private_ip]`, DNAT term name: `term_[public_ip]`
- Firewall filters:
  - Public VRF filter: `redirect_to_public_vrf_filter`
  - Private VRF filter: `redirect_to_[vrf_name]_vrf`
- Logical interface unit numbers:
  - Service ports: `2*vn_id -1`, `2*vn_id`
  - IRB interface: `vn_id`

## Dynamic Tunnels

Dynamic tunnel configuration in Contrail allows you to configure GRE tunnels on the Contrail Web user interface. When Contrail detects this configuration, the Device Manager module constructs GRE tunnel configuration and pushes it to the MX Series router. A property named `ip-fabric-subnets` is used in the global system configuration of the Contrail schema. Each IP fabric subnet and BGP router is configured as a dynamic tunnel destination point in the MX Series router. The physical router data plane IP address is considered the source address for the dynamic tunnel. You must configure the data plane IP address for auto configuring dynamic tunnels on a physical router. The IP fabric subnets is a global configuration;

all of the subnets are configured on all the physical routers in the cluster that have data plane IP configuration.

The following naming conventions are used in the API configuration:

- Global System Config: ip-fabric-subnets
- Physical Router: data-plane-ip

## Web UI Configuration

Figure 84 on page 410 shows the web user interface used to configure dynamic tunnels.

Figure 84: Edit Global Config Window

The screenshot shows the 'Edit Global Config' window. At the top, there is a section titled 'Encapsulation Priority Order' with three dropdown menus: 'VxLAN', 'MPLS Over GRE', and 'MPLS Over UDP'. Each dropdown menu has '+' and '-' buttons to its right. Below this is a section titled 'BGP Options' with a 'Global ASN' field containing the value '64512' and a checked checkbox for 'Enable iBGP Auto Mesh'. At the bottom, there is a section titled 'IP Fabric Subnets' with a '+' button to its right and a text input field with '+' and '-' buttons below it. A vertical ID 's042474' is visible on the right side of the window.

In the **Edit Global Config** window, the VTEP address is used for the data-plane-ip address.

The following is an example of the MX Series router configuration generated by the Device Manager.

```
root@host# show groups __contrail__ routing-options
```

```
router-id 172.16.184.200;
```

```
route-distinguisher-id 10.87.140.107;
```

```
autonomous-system 64512;
```

```
dynamic-tunnels {
```

```
  __contrail__ {
```

```
    source-address 172.16.184.200;
```

```
    gre;
```

```
    destination-networks {
```

```
      172.16.180.0/24;
```

```
      172.16.180.8/32;
```

```
      172.16.185.200/32;
```

```
      172.16.184.200/32;
```

```
      172.16.180.5/32;
```

```
      172.16.180.7/32;
```

```
    }
```

```
  }
```

```
}
```

## BGP Groups

When Device Manager detects BGP router configuration and its association with a physical router, it configures BGP groups on the physical router.

Figure 85 on page 412 shows the web user interface used to configure BGP groups.

Figure 85: Edit BGP Router Window

The screenshot shows the 'Edit BGP Router' configuration window. The fields are as follows:

- Hostname:** tasman
- Router Type:** Control Node (unselected), BGP Router (selected)
- Vendor ID:** mx
- IP Address:** 172.16.184.200
- Router ID:** 10.87.140.107
- Autonomous System:** 64512
- Address Families:** route-target x, inet-vpn, e-vpn x, inet6-vpn x

Below these fields are sections for 'Advanced Options' and 'Peer(s) Selection'. The 'Peer(s) Selection' section has sub-sections for 'Available Peer(s)' and 'Configured Peer(s)'. At the bottom right, there are 'Cancel' and 'Save' buttons. A vertical ID 's042475' is visible on the right side.

Figure 86 on page 413 shows the web user interface used to configure the physical router.

Figure 86: Edit Physical Router Window for BGP Groups

### Edit Physical Router

|                      |  |                     |   |
|----------------------|--|---------------------|---|
| <b>Name</b>          | <input type="text" value="tasman"/>        |                     |   |
| <b>Vendor</b>        | <input type="text" value="juniper"/>       | <b>Model</b>        | <input type="text" value="mx"/>             |
| <b>Management IP</b> | <input type="text" value="10.87.140.107"/> | <b>VTEP Address</b> | <input type="text" value="172.16.184.200"/> |
| <b>BGP Gateway</b>   | <input type="text" value="tasman"/>        |                     |   |

5042476

The following is an example of the MX Series router configuration generated by the Device Manager.

```

root@host show groups __contrail__ protocols bgp
group __contrail__ {
  type internal;
  multihop;
  local-address 172.16.184.200;
  hold-time 90;
  keep all;
  family inet-vpn {
    unicast;
  }
  family inet6-vpn {
    unicast;
  }
  family evpn {
    signaling;
  }
  family route-target;
  neighbor 172.16.180.8;
  neighbor 172.16.185.200;
  neighbor 172.16.180.5;
  neighbor 172.16.180.7;
}

group __contrail_external__ {
  type external;
  multihop;

```

```
local-address 172.16.184.200;  
hold-time 90;  
keep all;  
family inet-vpn {  
    unicast;  
}  
family inet6-vpn {  
    unicast;  
}  
family evpn {  
    signaling;  
}  
family route-target;  
}
```

## Extending the Private Network

Device Manager allows you to extend a private network and ports to a physical router. When Device Manager detects a VNC configuration, it pushes Layer 2 (EVPN) and Layer 3 VRF, import and export rules and interface configuration to the physical router.

[Figure 87 on page 415](#) shows the web user interface for configuring the physical router for extending the private network.

Figure 87: Edit Physical Router Window for Extending Private Networks

The screenshot shows the 'Edit Physical Router' configuration window. The fields are as follows:

|                  |  |              |        |
|------------------|--|--------------|--------|
| Name             | tasman   |              |        |
| Vendor           | juniper  | Model        | mx     |
| Management IP    | 10.87.140.107  | VTEP Address | 172.16 |
| BGP Gateway      | tasman   |              |        |
| Virtual Networks | vn_private-x1-63 (default-domain:default-project) ×<br>vn_private-x2-17 (default-domain:default-project) × |              |        |

A vertical ID '5042477' is visible on the right side of the window.

The following is an example of the MX Series router configuration generated by the Device Manager.

```

/* L2 VRF */

root@host# show groups __contrail__ routing-instances _contrail_l2_147_vn_private-
x1-63
vtep-source-interface lo0.0;
instance-type virtual-switch;
vrf-import _contrail_l2_147_vn_private-x1-63-import;
vrf-export _contrail_l2_147_vn_private-x1-63-export;
protocols {
    evpn {
        encapsulation vxlan;
        extended-vni-list all;
    }
}
bridge-domains {
    bd-147 {
        vlan-id none;
        routing-interface irb.147;
        vxlan {
            vni 147;
        }
    }
}

```

```

    }
}

/* L3 VRF */
root@host# show groups __contrail__ routing-instances _contrail_l3_147_vn_private-x1-63
instance-type vrf;
interface irb.147;
vrf-import _contrail_l3_147_vn_private-x1-63-import;
vrf-export _contrail_l3_147_vn_private-x1-63-export;
vrf-table-label;
routing-options {
    static {
        route 1.0.63.0/24 discard;
    }
    auto-export {
        family inet {
            unicast;
        }
    }
}

/* L2 Import policy */

root@host# ...cy-options policy-statement _contrail_l2_147_vn_private-x1-63-import
term t1 {
    from community target_64512_8000066;
    then accept;
}
then reject;

/* L2 Export Policy */
root@host# ...ail__ policy-options policy-statement _contrail_l2_147_vn_private-x1-63-export
term t1 {
    then {
        community add target_64512_8000066;
        accept;
    }
}

/* L3 Import Policy */

```



```

root@host# ...ail__ policy-options policy-statement _conrail_l3_147_vn_private-x1-63-import
term t1 {
    from community target_64512_8000066;
    then accept;
}
then reject;

/*L3 Export Policy */
root@host# ...ail__ policy-options policy-statement _conrail_l3_147_vn_private-x1-63-export
term t1 {
    then {
        community add target_64512_8000066;
        accept;
    }
}

```

## Extending the Public Network

When a public network is extended to a physical router, a static route is configured on the MX Series router. The configuration copies the next hop from the **public.inet.0** routing table to the **inet.0** default routing table, and copies a forwarding table filter from the **inet.0** routing table to the **public.inet.0** routing table. The filter is applied to all packets being looked up in the **inet.0** routing table and matches destinations that are in the subnet(s) for the public virtual network. The policy action is to perform the lookup in the **public.inet.0** routing table.

[Figure 88 on page 418](#) shows the web user interface for extending the public network.

Figure 88: Edit Network Gateway Window

**Edit Network GW1**

▼ **Advanced Options**

**Admin State** Up ▼

Shared  External

**DNS Servers** DNS Servers +

**VxLAN Identifier** Automatic

Allow Transit

Flood unknown unicast

Extend To Physical Router(s)

**Physical Router(s)** bali x

5042478

The following is an example of the MX Series router configuration generated by the Device Manager.

```
/* forwarding options */

root@host show groups __contrail__ forwarding-options
family inet {
  filter {
    input redirect_to_public_vrf_filter;
  }
}

/* firewall filter configuration */

root@host# show groups __contrail__ firewall family inet filter redirect_to_public_vrf_filter

term term-__contrail_l3_184_vn_public-x1- {
```

```

    from {

        destination-address {

            20.1.0.0/16;

        }

    }

    then {

        routing-instance _contrail_l3_184_vn_public-x1-;

    }

}

term default-term {

    then accept;

}

/* L3 VRF static route 0.0.0.0/0 configuration */

root@host# ...instances _contrail_l3_184_vn_public-x1- routing-options static route 0.0.0.0/0
next-table inet.0;

```

## Ethernet VPN Configuration

For every private network, a Layer 2 Ethernet VPN (EVPN) instance is configured on the MX Series router. If any Layer 2 interfaces are associated with the virtual network, logical interfaces are also created under the bridge domain.

The following is an example of the MX Series router configuration generated by the Device Manager.

```

root@host# show groups __contrail__ routing-instances _contrail_l2_147_vn_private-
x1-63
vtep-source-interface lo0.0;
instance-type virtual-switch;
vrf-import _contrail_l2_147_vn_private-x1-63-import;

```

```

vrf-export _contrail_l2_147_vn_private-x1-63-export;
protocols {
  evpn {
    encapsulation vxlan;
    extended-vni-list all;
  }
}
bridge-domains {
  bd-147 {
    vlan-id none;

    interface ge-1/0/5.0;
    routing-interface irb.147;
    vxlan {
      vni 147;
    }
  }
}
}

```

## Floating IP Addresses and Source Network Address Translation for Guest Virtual Machines and Bare Metal Servers

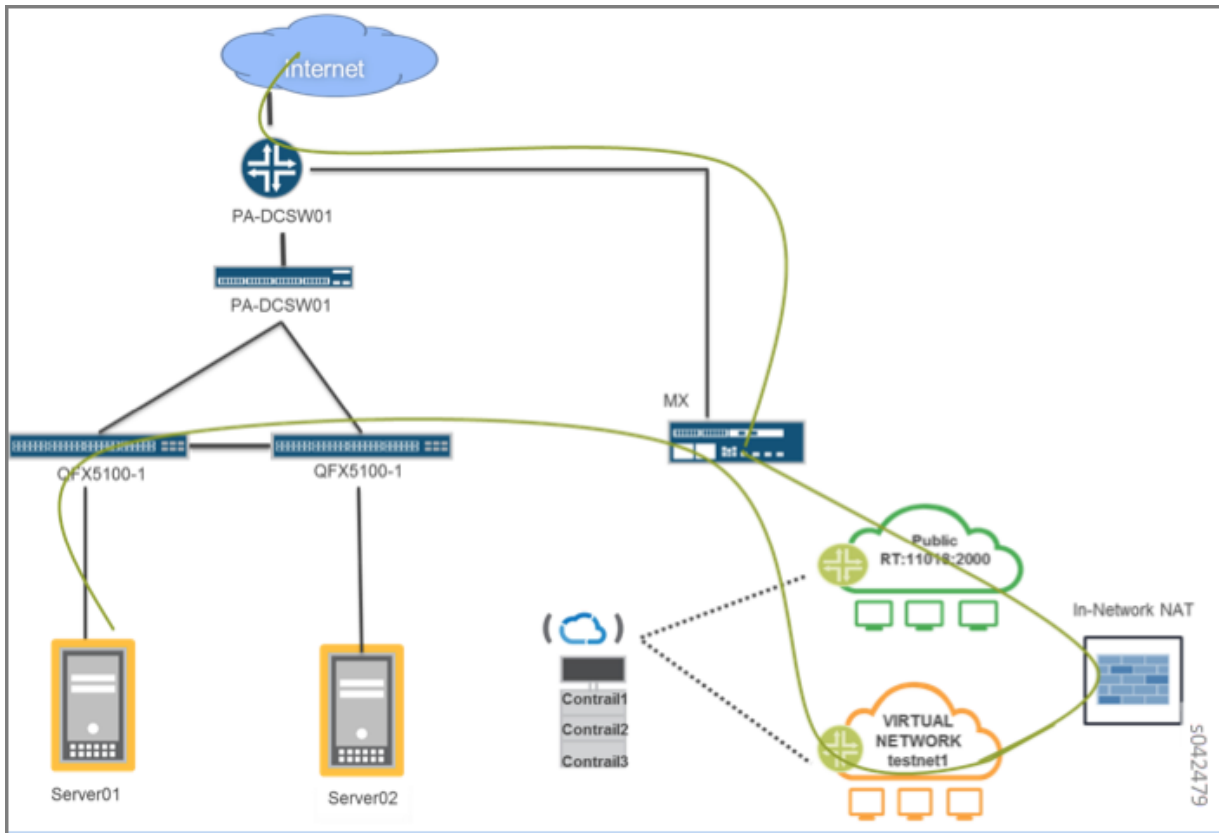
This section describes a bare metal server deployment scenario in which servers are connected to a TOR QFX device inside a private network and an MX Series router is the gateway for the public network connection.

The MX Series router provides the NAT capability that allows traffic from a public network to enter a private network and also allows traffic from the private network to the public network. To do this, you need to configure NAT rules on the MX Series router. The Device Manager is responsible for programming these NAT rules on MX Series routers when it detects that a bare metal server is connected to a public network.

You must configure virtual network computing for the TOR device, the MX Series router, the private network, and the public network, including the address pool. When a logical interface on the TOR device is associated with the virtual machine interface and a floating IP address is assigned to the same virtual machine interface (VMI), Contrail detects this and the Device Manager configures the necessary floating IP NAT rules on each of the MX Series routers associated with the private network.

[Figure 89 on page 421](#) illustrates that the Device Manager configures two special logical interfaces called *service-ports* on the MX Series router for NAT translation from the private network to the public network.

Figure 89: Logical Topology for Floating IP and SNAT



The Contrail schema allows a user to specify a service port name using the virtual network computing API. The service port must be a physical link on the MX Series router and the administrative and operational state must be up. The Device Manager creates two logical interfaces on this service port, one for each private virtual network, and applies NAT rules.

The private network routing instance on the MX Series router has a default static route (0.0.0.0/0) next hop pointing to the inside service interface. A public network routing instance on the MX Series router has a route for the private IP prefix next hop pointing to the outside service interface. The public IP address to private IP address and the reverse NAT rules are configured on the MX Series router.

A special routing instance for each private network to one or more public networks association is created on the MX Series router. This VRF has two interfaces on one side allowing traffic to and from the public network and another interface allowing traffic to and from the private network. Firewall filters on the MX Series router are configured so that, if the public network has floating IP addresses associated with a guest VM managed by the Contrail vRouter, the vRouter performs the floating IP address functionality. Otherwise, the MX Series router performs the NAT functions to send and receive the traffic to and from the bare metal server VM.

As illustrated in [Figure 89 on page 421](#), you must create the necessary physical device, interface, and virtual network configuration that is pushed to the to the MX Series router.

Contrail configuration can be done using the Web UI or VNC API. The required configuration is:

- Create the private virtual network.
- Create one or more TOR physical routers (No Junos OS configuration needs to be pushed to this device by Contrail. Therefore set the `vnc managed` attribute to `False`).
- Extend the private virtual network to the TOR device.
- Create physical and logical interfaces on the TOR device.
- Create the VMI on the private network for the bare metal server and associate the VMI with the logical interface. Doing that indicates that the bare metal server is connected to the TOR device through the logical interface. An instance IP address must be assigned to this VMI. The VMI uses a private IP address for the bare metal server.
- Create the gateway router. This is a physical router that is managed by the Device Manager.
- Configure the `service-port` physical interface information for the physical MX Series router. Device Manager configures two logical service interfaces on the MX Series router for each private network associated with the device, and automatically configures NAT rules on these interfaces for the private-to-public IP address translation and SNAT rules for the opposite direction. The logical port ID is calculated from the virtual network ID allocated by Contrail VNC. Two logical ports are required for each private network
- Associate the floating IP address, including creating the public network, the floating IP address pool, and a floating IP address in Contrail, and associate this IP address with the VMI bare metal server.
- The private network and public network must be extended to the physical router.

When the required configuration is present in Contrail, the Device Manager pushes the generated Junos OS configuration to the MX Series device. An example configuration is shown in the following.

```
/* NAT VRF configuration */

root@host# show groups __contrail__ routing-instances _contrail_13_147_vn_private-x1-63-nat

instance-type vrf;

interface si-2/0/0.293;

vrf-import _contrail_13_147_vn_private-x1-63-nat-import;

vrf-export _contrail_13_147_vn_private-x1-63-nat-export;
```

```
vrf-table-label;

routing-options {

    static {

        route 0.0.0.0/0 next-hop si-2/0/0.293;

    }

    auto-export {

        family inet {

            unicast;

        }

    }

}

/* NAT VRF import policy */

root@host# ...y-statement _contrail_l3_147_vn_private-x1-63-nat-import

term t1 {

    from community target_64512_8000066;

    then accept;

}

then reject;

/* NAT VRF Export policy */

root@host# ..._ policy-options policy-statement _contrail_l3_147_vn_private-x1-63-nat-export

term t1 {

    then reject;
```

```
}

/* The following additional config is generated for public l3 vrf */

root@host# show groups __contrail__ routing-instances _contrail_l3_184_vn_public-x1-

interface si-2/0/0.294;

routing-options {

    static {

        route 20.1.252.8/32 next-hop si-2/0/0.294;

        route 20.1.252.9/32 next-hop si-2/0/0.294;

    }

}

/* Services set configuration */

root@host# show groups __contrail__

services {

    service-set sv-_contrail_l3_147_vn_ {

        nat-rules sv-_contrail_l3_147_vn_-sn-rule;

        nat-rules sv-_contrail_l3_147_vn_-dn-rule;

        next-hop-service {

            inside-service-interface si-2/0/0.293;

            outside-service-interface si-2/0/0.294;

        }

    }

}
```



```
}

/* Source Nat Rules*/

root@host# show groups __contrail__ services nat rule sv-contrail_13_147_vn_-sn-rule

match-direction input;

term term_1_0_63_248 {

    from {

        source-address {

            1.0.63.248/32;

        }

    }

    then {

        translated {

            source-prefix 20.1.252.8/32;

            translation-type {

                basic-nat44;

            }

        }

    }

}

term term_1_0_63_249 {

    from {

        source-address {
```

```
        1.0.63.249/32;
    }
}

then {

    translated {

        source-prefix 20.1.252.9/32;

        translation-type {

            basic-nat44;

        }

    }

}

}

/* Destination NAT rules */

root@host# show groups __contrail__ services nat rule sv_contrail_l3_147_vn_dn-rule

match-direction output;

term term_20_1_252_8 {

    from {

        destination-address {

            20.1.252.8/32;

        }

    }

}
```

```
then {  
  
    translated {  
  
        destination-prefix 1.0.63.248/32;  
  
        translation-type {  
  
            dnat-44;  
  
        }  
  
    }  
  
}  
  
}  
  
term term_20_1_252_9 {  
  
    from {  
  
        destination-address {  
  
            20.1.252.9/32;  
  
        }  
  
    }  
  
    then {  
  
        translated {  
  
            destination-prefix 1.0.63.249/32;  
  
            translation-type {  
  
                dnat-44;  
  
            }  
  
        }  
  
    }  
  
}
```

```
    }  
  }  
  
  /* Public Vrf Filter */  
  
  root@host# show groups __contrail__ firewall family inet filter redirect_to_public_vrf_filter  
  
  term term-contrail_l3_184_vn_public-x1- {  
  
    from {  
  
      destination-address {  
  
        20.1.0.0/16;  
  
      }  
  
    }  
  
    then {  
  
      routing-instance _contrail_l3_184_vn_public-x1-;  
  
    }  
  
  }  
  
  term default-term {  
  
    then accept;  
  
  }  
  
  /* NAT Vrf filter */  
  
  root@host# ...all family inet filter redirect_to__contrail_l3_147_vn_private-x1-63-nat_vrf  
  
  term term-contrail_l3_147_vn_private-x1-63-nat {
```

```
from {  
  
    source-address {  
  
        1.0.63.248/32;  
  
        1.0.63.249/32;  
  
    }  
  
}  
  
then {  
  
    routing-instance _contrail_l3_147_vn_private-x1-63-nat;  
  
}  
  
}  
  
term default-term {  
  
    then accept;  
  
}  
  
/* IRB interface for NAT VRF */  
  
root@host# show groups __contrail__ interfaces  
  
irb {  
  
    gratuitous-arp-reply;  
  
    unit 147 {  
  
        family inet {  
  
            filter {  
  
                input redirect_to__contrail_l3_147_vn_private-x1-63-nat_vrf;  
  
            }  
  
        }  
  
    }  
  
}
```

```

        address 1.0.63.254/24;

    }

}

/* Service Interfaces config */

root@host# show groups __contrail__ interfaces si-2/0/0

unit 293 {

    family inet;

    service-domain inside;

}

unit 294 {

    family inet;

    service-domain outside;

}

```

## Samples of Generated Configurations for an MX Series Device

This section provides several scenarios and samples of MX Series device configurations generated using Python script.

### Scenario 1: Physical Router With No External Networks

The following describes the use case of basic vn, vmi, li, pr, pi configuration with no external virtual networks. When the Python script shown in the following is executed with the parameters of this use case, the configuration is applied on the MX Series physical router.

Script executed on the Contrail controller:

```
# python provision_physical_router.py --api_server_ip 127.0.0.1 --api_server_port 8082 --  
admin_user user1 --admin_password password1 --admin_tenant_name default-domain --op add_basic
```

Generated configuration for MX Series device:

```
root@host# show groups __contrail__  
routing-options {  
    route-distinguisher-id 10.84.63.133;  
    autonomous-system 64512;  
}  
protocols {  
    bgp {  
        group __contrail__ {  
            type internal;  
            multihop;  
            local-address 10.84.63.133;  
            keep all;  
            family inet-vpn {  
                unicast;  
            }  
            family inet6-vpn {  
                unicast;  
            }  
            family evpn {  
                signaling;  
            }  
            family route-target;  
        }  
        group __contrail_external__ {  
            type external;  
            multihop;  
            local-address 10.84.63.133;  
            keep all;  
            family inet-vpn {  
                unicast;  
            }  
            family inet6-vpn {  
                unicast;  
            }  
        }  
    }  
}
```

```
        family evpn {
            signaling;
        }
        family route-target;
    }
}
policy-options {
    policy-statement __contrail__default-domain_default-project_vn1-export {
        term t1 {
            then {
                community add target_64200_8000008;
                accept;
            }
        }
    }
    policy-statement __contrail__default-domain_default-project_vn1-import {
        term t1 {
            from community target_64200_8000008;
            then accept;
        }
        then reject;
    }
    community target_64200_8000008 members target:64200:8000008;
}
routing-instances {
    __contrail__default-domain_default-project_vn1 {
        instance-type vrf;
        interface ge-1/0/5.0;
        vrf-import __contrail__default-domain_default-project_vn1-import;
        vrf-export __contrail__default-domain_default-project_vn1-export;
        vrf-table-label;
        routing-options {
            static {
                route 10.0.0.0/24 discard;
            }
            auto-export {
                family inet {
                    unicast;
                }
            }
        }
    }
}
```



```

}
}

```

## Scenario 2: Physical Router With External Network, Public VRF

This section describes the use case of vn, vmi, li, pr, pi configuration with an external virtual network, public VRF. When the Python script shown is executed with the parameters of this use case, the configuration is applied on the MX Series physical router.

This example assumes that the configuration already described in Scenario 1 has been executed.

*Script executed on the Contrail controller:*

```

# python provision_physical_router.py --api_server_ip 127.0.0.1 --api_server_port 8082 --
admin_user user1 --admin_password password1 --admin_tenant_name default-domain --op add_basic --
public_vrf_test True

```

*Generated configuration for MX Series device:*

The following additional configuration is pushed to the MX Series device, in addition to the configuration generated in Scenario 1.

```

forwarding-options {
  family inet {
    filter {
      input redirect_to___contrail__default-domain_default-project_vn1_vrf;
    }
  }
}
firewall {
  filter redirect_to___contrail__default-domain_default-project_vn1_vrf {
    term t1 {
      from {
        destination-address {
          10.0.0.0/24;
        }
      }
      then {
        routing-instance __contrail__default-domain_default-project_vn1;
      }
    }
    term t2 {

```

```

        then accept;
    }
}
}
routing-instances {
    __contrail__default-domain_default-project_vn1 {
        routing-options {
            static {
                route 0.0.0.0/0 next-table inet.0;
            }
        }
    }
}
}
}

```

### Scenario 3: Physical Router With External Network, Public VRF, and EVPN

The scenario in this section describes the use case of vn, vmi, li, pr, pi physical router configuration with external virtual networks (public VRF) and EVPN configuration. When the Python script (as in the previous examples) is executed with the parameters of this scenario, the following configuration is applied on the MX Series physical router.

This example assumes that the configuration already described in Scenario 1 has been executed.

*Script executed on the Contrail controller:*

```

# python provision_physical_router.py --api_server_ip 127.0.0.1 --api_server_port 8082 --
admin_user user1 --admin_password password1 --admin_tenant_name default-domain --op add_basic --
public_vrf_test True -vxlan 2002

```

*Generated configuration for MX Series device:*

The following additional configuration is pushed to the MX Series device, in addition to the configuration generated in Scenario 1.

```

protocols {
    mpls {
        interface all;
    }
}
firewall {
    filter redirect_to__contrail__default-domain_default-project_vn1_vrf {
        term t1 {

```

```
        from {
            destination-address {
                10.0.0.0/24;
            }
        }
    }
    then {
        routing-instance __contrail__default-domain_default-project_vn1;
    }
}
term t2 {
    then accept;
}
}
}
routing-instances {
    __contrail__default-domain_default-project_vn1 {
        vtep-source-interface lo0.0;
        instance-type virtual-switch;
        vrf-target target:64200:8000008;
        protocols {
            evpn {
                encapsulation vxlan;
                extended-vni-all;
            }
        }
        bridge-domains {
            bd-2002 {
                vlan-id 2002;
                interface ge-1/0/5.0;
                routing-interface irb.2002;
                vxlan {
                    vni 2002;
                    ingress-node-replication;
                }
            }
        }
    }
}
}
```

## Scenario 4: Physical Router With External Network, Public VRF, and Floating IP Addresses for a Bare Metal Server

The scenario in this section describes the user case of `vn`, `vmi`, `li`, `pr`, `pi` physical router configuration with external virtual networks (public VRF) and floating IP addresses for bare metal server configuration.

*Script executed on the Contrail controller:*

```
#python provision_physical_router.py --api_server_ip <ip address> --api_server_port 8082 --
admin_user admin --admin_password <password> --admin_tenant_name default-domain --op {fip_test|
delete_fip_test}
```

### RELATED DOCUMENTATION

[REST APIs for Extending the Contrail Cluster to Physical Routers, and Physical and Logical Interfaces | 440](#)

## SR-IOV VF as the Physical Interface of vRouter

Starting with Contrail 3.0, support is provided for single-root I/O virtualization (SR-IOV) virtual functions used as the physical router for vRouter.

SR-IOV allows a network interface to separate the access to its resources across multiple PCI Express functions. The functions can be physical or virtual.

The Contrail vRouter can use an SR-IOV virtual function as its physical interface. One virtual function on a network interface can be used by vRouter, while the remaining virtual functions can be used by virtual machines on the same compute node. It is also possible to create a VLAN interface on a virtual function, and use that as the physical interface of the vRouter.

Alternatively, virtual functions from two different interfaces can be bonded together, and that bonded interface can be used as the physical interface of the vRouter. It is also possible to create a VLAN on a bonded interface, like the one just described, and then use that bonded interface as the physical interface of the vRouter.

To set up virtual functions for the physical interface of a vRouter:

1. Include the `env.sriov` section in the `testbed.py` file, and complete the following steps to define the SR-IOV virtual functions, so that the virtual functions are created during the provisioning of the cluster.

2. Within `env.sriov`, create SR-IOV virtual functions on the compute nodes (`host1` and `host2`, in this example). Virtual functions are usually identified with the following naming scheme: `p6p2_1`, `p6p2_2`, and so on. For example:

```
env.sriov = {

host1 : [ {'interface' : 'p6p2', 'VF' : 7, 'physnets' : ['physnet1']}],
host2 : [ {'interface' : 'p6p2', 'VF' : 7, 'physnets' : ['physnet1']}]

}
```

3. Specify the virtual function interfaces in the `control_data` section of the `testbed.py` file, with or without a VLAN, so that they can be used by the vRouter. For example:

```
control_data = {

host1 : { 'ip': '10.x.x.100/2x', 'gw' : '10.x.x.254', 'device': 'p6p2_1' },
host2 : { 'ip': '10.x.x.200/2x', 'gw' : '10.x.x.254', 'device': 'p6p2_2' }

}
```

4. Optionally, for bonded interfaces (`bond0` in this example), specify the virtual functions in the `bond` section of the `testbed.py` file, with or without a VLAN. For example:

```
bond= {

host1 : {'name': 'bond0', 'member': ['p6p2_4', 'p6p1_5'], 'mode': 'balance-xor' },
host2 : {'name': 'bond0', 'member': ['p6p2_2', 'p6p1_3'], 'mode': 'balance-xor' }

}
```

## Using Gateway Mode to Support Remote Instances

### IN THIS SECTION

- [Gateway Mode Overview | 438](#)
- [Provisioning Gateway Mode | 438](#)
- [Configuring Gateway Mode | 439](#)
- [Configuring Gateway Mode and High Availability | 439](#)
- [Scaling | 440](#)

Extending virtual instances running non-Openstack clusters or extending bare metal servers into Contrail virtual networks can be achieved using the OVSDB protocol.

Additionally, starting with Contrail Release 3.1, an experimental mode has been added that enables you to configure a Contrail compute node to run in gateway mode to support remote instances.

With Contrail Release 3.2, the gateway mode can also be used with VMware, using the VMware VMs as the remote instances.

### Gateway Mode Overview

Traffic from each external virtual instance is tagged with a unique VLAN, which is then mapped to a virtual machine interface (VMI) in the Contrail cluster. A Contrail compute node can be configured to map VLAN-tagged traffic coming on a physical port, other than the cluster's underlay IP fabric port, to a VMI configured in the Contrail cluster. The VMI corresponds to the interface of the remote instance. A vRouter on a gateway compute node operates like a local VMI, with traffic subjected to the forwarding decisions and policies that would be on the local VMI.

**NOTE:** For gateway mode, one VLAN is mapped to one virtual machine interface.

### Provisioning Gateway Mode

To provision gateway mode:

In `/etc/contrail/contrail-vrouter-agent.conf`, in the `DEFAULT` section add:

```
gateway_mode = server
```

When finished, restart `contrail-vrouter-agent`.

## Configuring Gateway Mode

To configure gateway mode:

1. Configure a physical router, using the host name of the compute node that acts as the gateway.
2. Create a physical interface on the physical router, using the name of the interface on the compute node that will be used for the gateway traffic.
3. Create a logical interface on the physical interface, using a unique VLAN ID and set the type to L2.
4. Create a virtual machine interface (VMI) in the required virtual network, identifying the MAC address and IP address of the remote instance, then link the VMI to the logical interface.

## External Configuration

The traffic from the remote instance should come to the gateway port with the required VLAN tag.

## Configuring Gateway Mode and High Availability

Multiple gateway nodes can be configured to have high availability.

The selection of the active gateway node is expected to be handled by using (R)STP from the switches connecting the gateway node. For this, a special virtual network is configured in Contrail that will flood the STP BPDUs.

For high availability for the gateway mode, make the following changes to the gateway mode configuration procedure:

1. On the gateway compute nodes, create logical interfaces with VLAN 0.
2. Create a dummy VMI belonging to the special virtual network, following the regular gateway mode configuration procedure.
3. Link the VMI to the VLAN 0 logical interfaces on the gateway nodes that will form a high availability group. This enables STP to allow traffic to one of the gateway ports, while blocking others.
4. For each remote instance, create a logical interface on the gateway nodes in the high availability group. Link the logical interface to the VMI created for the remote instance. The corresponding instance IP should have active-backup mode set, which is the default mode.

Upon completion of this procedure, Contrail will handle the switch over of the traffic to a different gateway node.

**NOTE:** This procedure can also be used to set up a vCenter gateway. Because the VMware VMs are the remote instances, their traffic must be configured to arrive VLAN-tagged at the gateway node's physical interface and their interfaces must be configured in Contrail. Using this configuration, all Contrail features available on a virtual machine interface will be applied for any traffic between VMware and Contrail.

## Scaling

The default number of interfaces supported on a compute node is 4352. Because each remote instance has a logical interface and a virtual machine interface, up to 2K remote interfaces can be supported with the default configuration. To support 4K remote instances, the maximum number of interfaces on the compute node that is acting as the gateway should be configured to be 8K. For more information about how the vRouter options can be modified, see <https://github.com/Juniper/contrail-controller/wiki/Vrouter-Module-Parameters>.

## REST APIs for Extending the Contrail Cluster to Physical Routers, and Physical and Logical Interfaces

### IN THIS SECTION

- Introduction: REST APIs for Extending Contrail Cluster | 440
- REST API for Physical Routers | 441
- REST API for Physical Interfaces | 443
- REST API for Logical Interfaces | 444

### Introduction: REST APIs for Extending Contrail Cluster

Use the following REST APIs when extending the Contrail cluster to include physical routers, physical interfaces, and logical interfaces.



## REST API for Physical Routers

Use the following REST API when extending the Contrail cluster to include physical routers.

```
{  
  
  u'physical-router': {  
  
    u'physical_router_management_ip': u'100.100.100.100',  
  
    u'virtual_router_refs': [],  
  
    u'fq_name': [  
  
      u'default-global-system-config',  
  
      u'test-router'  
  
    ],  
  
    u'name': u'test-router',  
  
    u'physical_router_vendor_name': u'juniper',  
  
    u'parent_type': u'global-system-config',  
  
    u'virtual_network_refs': [],  
  
    'id_perms': {  
  
      u'enable': True,  
  
      u'uuid': None,  
  
      u'creator': None,  
  
      u'created': 0,  
  
      u'user_visible': True,  
  
      u'last_modified': 0,  
  
    }  
  
  }  
  
}
```

```
u'permissions': {  
  
    u'owner': u'cloud-admin',  
  
    u'owner_access': 7,  
  
    u'other_access': 7,  
  
    u'group': u'cloud-admin-group',  
  
    u'group_access': 7  
  
},  
  
u'description': None  
  
},  
  
u'bgp_router_refs': [],  
  
u'physical_router_user_credentials': {  
  
    u'username': u'',  
  
    u'password': u''  
  
},  
  
'display_name': u'test-router',  
  
u'physical_router_dataplane_ip': u'101.1.1.1'  
  
}  
  
}
```

## REST API for Physical Interfaces

Use the following REST API when extending the Contrail cluster to include physical interfaces.

```
{  
  
  u'physical-interface': {  
  
    u'parent_type': u'physical-router',  
  
    'id_perms': {  
  
      u'enable': True,  
  
      u'uuid': None,  
  
      u'creator': None,  
  
      u'created': 0,  
  
      u'user_visible': True,  
  
      u'last_modified': 0,  
  
      u'permissions': {  
  
        u'owner': u'cloud-admin',  
  
        u'owner_access': 7,  
  
        u'other_access': 7,  
  
        u'group': u'cloud-admin-group',  
  
        u'group_access': 7  
  
      },  
  
      u'description': None  
  
    },  
  
  },  
  
}
```

```

u'fq_name': [
    u'default-global-system-config',
    u'test-router',
    u'ge-0/0/1'
],
u'name': u'ge-0/0/1',
'display_name': u'ge-0/0/1'
}
}

```

## REST API for Logical Interfaces

Use the following REST API when extending the Contrail cluster to include logical interfaces.

```

{
  u'logical-interface': {
    u'fq_name': [
      u'default-global-system-config',
      u'test-router',
      u'ge-0/0/1',
      u'ge-0/0/1.0'
    ],
    u'parent_uuid': u'6608b8ef-9704-489d-8cbc-fed4fb5677ca',
    u'logical_interface_vlan_tag': 0,

```

```
u'parent_type': u'physical-interface',

u'virtual_machine_interface_refs': [

  {

u'to': [

    u'default-domain',

    u'demo',

    u'4a2edbb8-b69e-48ce-96e3-7226c57e5241'

  ]

  }

],

'id_perms': {

  u'enable': True,

  u'uuid': None,

  u'creator': None,

  u'created': 0,

  u'user_visible': True,

  u'last_modified': 0,

  u'permissions': {

    u'owner': u'cloud-admin',

    u'owner_access': 7,

    u'other_access': 7,
```

```
    u'group': u'cloud-admin-group',  
  
    u'group_access': 7  
  
  },  
  
  u'description': None  
  
},  
  
u'logical_interface_type': u'l2',  
  
'display_name': u'ge-0/0/1.0',  
  
u'name': u'ge-0/0/1.0'  
  
}  
  
}
```

## RELATED DOCUMENTATION

[Using ToR Switches and OVSDB to Extend the Contrail Cluster to Other Instances | 384](#)

[Using Device Manager to Manage Physical Routers | 404](#)

# Installing and Using Contrail Storage

## IN THIS CHAPTER

- [Installing and Using Contrail Storage | 447](#)

## Installing and Using Contrail Storage

### IN THIS SECTION

- [Overview of the Contrail Storage Solution | 447](#)
- [Basic Storage Functionality with Contrail | 448](#)
- [Ceph Block and Object Storage Functionality | 448](#)
- [Using the Contrail Storage User Interface | 449](#)
- [Hardware Specifications | 450](#)
- [Contrail Storage Provisioning | 450](#)

### Overview of the Contrail Storage Solution

Contrail provides a storage support solution using OpenStack Cinder configured to work with Ceph. Ceph is a unified, distributed storage system whose infrastructure provides storage services to Contrail.

The Contrail storage solution has the following features:

- Provides storage class features to Contrail clusters, including replication, reliability, and robustness.
- Uses open source components.
- Uses Ceph block and object storage functionality.
- Integrates with OpenStack Cinder functionality.

- Does not require virtual machines (VMs) to configure mirrors for replication.
- Allows nodes to provide both compute and storage services.
- Provides easy installation of basic storage functionality based on Contrail roles.
- Provides a Contrail-integrated user interface from which the user can monitor Ceph components and drill down for more information about components.
- Provides native live-migration support if the VM is booted with Ceph storage as its root volume.
- Provides object storage support through Swift and S3 APIs.

### **Basic Storage Functionality with Contrail**

The following are basic interaction points between Contrail and the storage solution.

- Cinder volumes must be manually configured prior to installing the Contrail storage solution. The Cinder volumes can be attached to virtual machines (VMs) to provide additional storage.
- The storage solution stores virtual machine boot images and snapshots in Glance, using Ceph object storage functionality.
- All storage nodes can be monitored through a graphical user interface (GUI).
- It is possible to migrate virtual machines that have ephemeral storage in Ceph.

### **Ceph Block and Object Storage Functionality**

In Contrail Release 4.0, installing the Contrail storage solution creates the following Ceph configurations.

- Each disk is configured as a standalone storage device, enhancing optimal performance and creating proper failure boundaries. Ceph allocates and assigns a process called object storage daemon (OSD) to each disk.
- A replication factor of 2 is configured, consisting of one original instance plus one replica copy. Ceph ensures that each replica is on a different storage node.
- A Ceph monitor process (mon) is configured on the contrail-ceph-controller node.
- The correct number of placement groups are automatically configured, based on the number of disk drives in the cluster.
- Properly identified SSD drives are set up for use as Ceph OSD journals to reduce write latencies.
- Multi-pool configuration is set up to segregate the OSD disks into logical pools improving performance and efficiency.



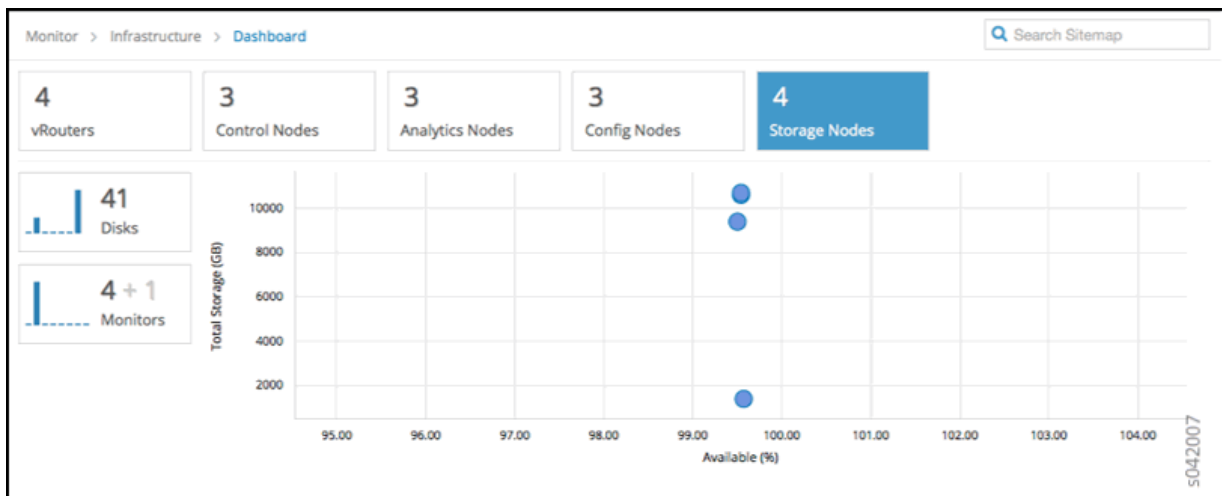
- If multiple storage nodes are in a single chassis, the chassis option helps in defining replication of data and also disabling replication of data within the nodes of the same chassis. Replication helps in avoiding data loss during a power failure to the chassis.

## Using the Contrail Storage User Interface

The Contrail storage solution provides a user interface integrated into the Contrail user interface. The storage solution user interface displays the following:

- Customer usable space, which is different from Ceph total space. The displayed usable space does not display the space used by replication and other Ceph functions.
- Monitor OSDs (disks), monitoring processes (mon), and state changes, enabling quick identification of resource failures within storage components.
- Total cluster I/O statistics and individual drive statistics.
- Ceph-specific information about each OSD (disk).
- Ceph logs, Ceph nodes, and Ceph alerts.

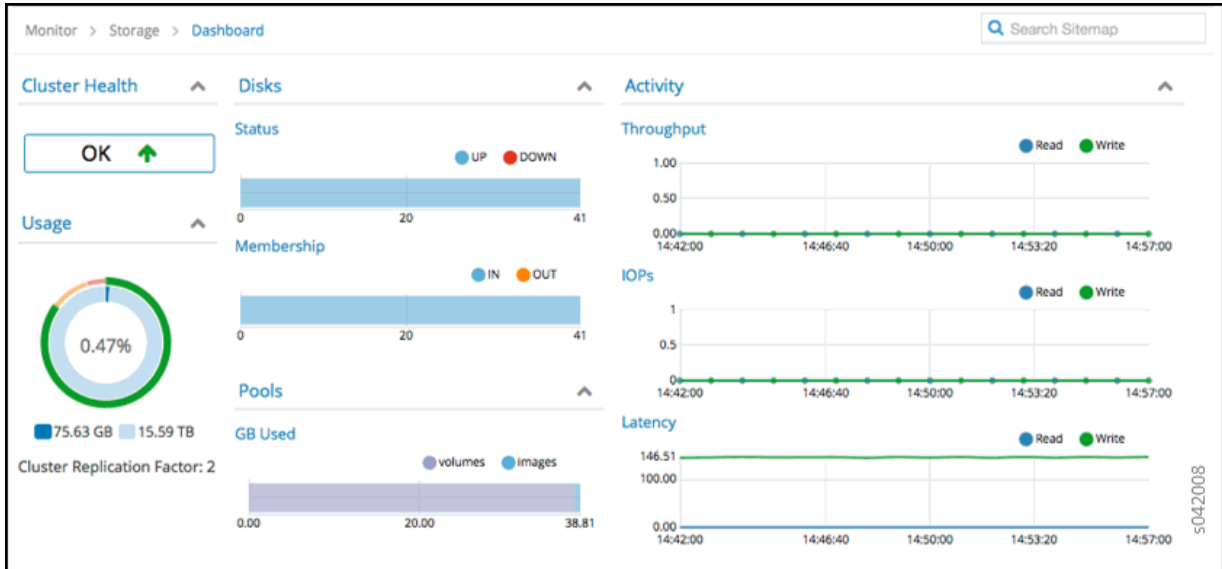
Select **Monitor > Infrastructure > Dashboard** to display an at-a-glance view of the system infrastructure components, including the numbers of virtual routers, control nodes, analytics nodes, config nodes, and storage nodes currently operational, and a bubble chart of storage nodes showing the Available (%) and Total Storage (GB). See the following figure.



Bubble charts use the following color-coding scheme for storage nodes:

- Blue—working as configured.
- Red—error, node is down.
- Yellow—one of the node disks is down.

Select **Monitor > Storage > Dashboard** to see a summary of cluster health, usage, pools, and disk status, and to gain insight into activity statistics for all nodes. See the following figure.



## Hardware Specifications

The following are additional hardware specifications needed for the Contrail storage solution.

Additional minimum specifications:

- Two 500 GB, 7200 RPM drives in the server 4 and server 5 cluster positions (those with the compute storage role) in the Contrail installation. This configuration provides 1 TB of clustered, replicated storage.

Recommended compute storage configuration:

- For every 4-5 HDD devices on one compute storage node, use one SSD device to provide the OSD journals for that set of HDD devices.

## Contrail Storage Provisioning

The `contrail-ceph-controller` and `contrail-ceph-compute` are two roles required to enable Ceph storage. The `contrail-ceph-controller` role is added to the Ceph monitor servers. The number of mons is limited to three for small clusters and five for large clusters with more than 1000 disks. The `contrail-ceph-compute` role is added to the servers that have the physical disks required for Ceph storage and also to the OpenStack Nova compute nodes that require Ceph storage services.

The following example displays sample `cluster.json` to provide Ceph storage configurations.

```
"parameters": {
    "provision": {
```

```

        "contrail_4": {
            "storage_ceph_config": {
                "replica_size": 2,
"ceph_object_storage": "True",
"object_store_pool": "volumes"
            }
        }
    }
}

```

The `replica_size` is added to change the default replica size of 2. The `ceph_object_storage` option enables the Ceph-based object storage to support Swift and S3 APIs and the `object_store_pool` option specifies the Ceph pool used for the Ceph object storage functionality.

The following example displays sample `server.json` to enable Ceph storage.

Server.json :

```

"parameters": {
    "provision": {
        "contrail_4":{
            "storage":{
                "storage_osd_disks":[
                    "/dev/sdb:/dev/sdd:Pool_1",
                    "/dev/sdc:/dev/sdd:Pool_2"
                ],
                "storage_osd_ssd_disks":[
                    "/dev/sde:Pool_1",
                    "/dev/sdf:Pool_2"
                ],
                "chassis_id": "chassis_1"
            }
        }
    }
    "roles": [
        "contrail-ceph-controller", "contrail-ceph-compute"
        [p0-
    ]

```

The `storage_osd_disks` or `storage_osd_ssd_disk` is needed to provision the disks for Ceph. The first disk is OSD disk and the second optional disk is used as a Journal disk. If a multi-pool configuration is required, the pool name can be added along the OSD disk as shown in the `server.json` to enable Ceph storage. The `chassis_id` option can also be included per server. Pools and the `chassis` option cannot co-exist.

**NOTE:** The disks added to Ceph are not included in the OS disk. The partition parameter in the server JSON lists only the required OS disks.

```
"parameters": {  
  "partition": "/dev/sda"  
}
```

The disks added to Ceph cannot be part of LVM.

# Upgrading Contrail Software

## IN THIS CHAPTER

- [Upgrading Contrail 4.0 to 4.1 | 453](#)
- [Upgrade Procedure for RHOSP-based Contrail 4.1.2 to Contrail 4.1.3 | 455](#)
- [Upgrade Procedure for Ubuntu-based Contrail 4.1.3 to Contrail 4.1.4 Using Juju with Netronome SmartNIC | 468](#)
- [Upgrade Procedure for RHOSP-based Contrail 4.1.3 to Contrail 4.1.4 | 477](#)
- [Dynamic Kernel Module Support \(DKMS\) for vRouter | 492](#)
- [Backup and Restore Contrail Configuration Database | 493](#)

## Upgrading Contrail 4.0 to 4.1

### IN THIS SECTION

- [Upgrade Assumptions | 453](#)
- [Upgrade Procedure | 454](#)

This section provides the process for upgrading an existing Contrail Release 4.0 system to Contrail Release 4.1.

### Upgrade Assumptions

This upgrade procedure assumes the following.

- The initial cluster (4.0.x) was provisioned using Server Manager.
- The OpenStack SKU is the same in the “from” and “to” versions.

- A backup has been made of the analytics database, see ["Backing Up Contrail Databases Using JSON Format" on page 1058](#).

## Upgrade Procedure

1. Make a backup of the analytics database, because the upgrade procedure removes the analytics database information, see ["Backing Up Contrail Databases Using JSON Format" on page 1058](#).
2. Add the new Contrail 4.1 Debian image to the Server Manager JSON used for provisioning.

```
server-manager add image -f contrail_image.json
```

3. Upgrade the cluster by reprovisioning the cluster with the new image.

- For an all-in-one, single-node demo system:

```
server-manager provision--cluster_id <all_in_one_cluster> combined_image_mainline
```

- For a multinode system:

```
server-manager provision --cluster_id <multi_node> combined_image_mainline
```

4. Monitor progress of the provisioning by observing cluster status or log entries.

- Cluster status: `server-manager display server --cluster_id <cluster_id> --select "id,ip_address,roles,status"`
- Log entries: `/var/log/contrail-server-manager/debug.log`

**NOTE:** Log entries from the previous version are lost in the upgrade process.

For more upgrade instructions, see:

- [Upgrade Procedure for RHOSP-based Contrail 4.1.3 to Contrail 4.1.4](#)
- [Upgrade Procedure for RHOSP-based Contrail 4.1.2 to Contrail 4.1.3](#)
- [Upgrade Procedure for RHOSP-based Contrail 4.1.1 to Contrail 4.1.2](#)
- [Upgrade Procedure for RHOSP-based Contrail 3.2.x to Contrail 4.1](#)

## Upgrade Procedure for RHOSP-based Contrail 4.1.2 to Contrail 4.1.3

### IN THIS SECTION

- Prerequisite | 455
- Upgrade the Undercloud | 456
- Update Red Hat Director Image Archives | 458
- Prepare Repositories on all Nodes | 460
- Upgrade the Operating System on Contrail Nodes | 460
- Prepare the Contrail Packages | 461
- Upgrade the Contrail Heat Templates | 462
- Modify the Yum Update Script for TripleO Puppet | 463
- Update the Overcloud Deployment Plan | 464
- Upgrade the Overcloud | 465

This section presents the steps to upgrade an OSP-based Contrail deployment from Contrail version 4.1.2 to Contrail version 4.1.3.

### Prerequisite

Before upgrading to Contrail Release 4.1.3, you must update the `net-snmp` package to the `net-snmp #37` version. The following `net-snmp` packages must be available in the upgrade repository and are installed automatically on Contrail Analytics nodes during the upgrade process:

- `net-snmp-5.7.2-37.el7.x86_64.rpm`
- `net-snmp-agent-libs-5.7.2-37.el7.x86_64.rpm`
- `net-snmp-libs-5.7.2-37.el7.x86_64.rpm`
- `net-snmp-utils-5.7.2-37.el7.x86_64.rpm`

Ensure you have a cloud up and running with RHOSP10 and Contrail 4.1.2 before you proceed with the upgrade procedure.

This procedure has been validated with the following Contrail, Red Hat, and OpenStack versions.

| Contrail Version | Red Hat Version | OpenStack Version  |
|------------------|-----------------|--|
| 3.2.3            | RHEL 7.3        | RHOSP10 (packages dated Apr. 15, 2017)   |
| 3.2.6            | RHEL 7.4        | RHOSP10 (packages dated Feb. 2, 2018)  |
| 4.1              | RHEL 7.4        | RHOSP10 (packages dated Feb. 27, 2018)   |
| 4.1.1            | RHEL 7.5        | RHOSP10 (packages dated Jun. 4, 2018)<br>RHOSP11 (packages dated Jun. 4, 2018) |
| 4.1.2            | RHEL 7.5        | RHOSP10 (packages dated Oct 29, 2018)  |
| 4.1.3            | RHEL 7.5        | RHOSP10 (packages dated Oct 29, 2018)  |



**CAUTION:** Set the Red Hat Satellite filter end date to October 29, 2018 before proceeding with the upgrade.

## Upgrade the Undercloud

Upgrade the undercloud to the most current RHOSP10 version.

1. Log in to the undercloud as the stack user.

```
$ su - stack
```

2. Update the Contrail repositories.

```
$ sudo rm -rf /etc/yum.repos.d/*contrail*
```

```
$ curl http://newrepo.contrail41-dev.repo -o /etc/yum.repos.d/localrepo.repo
```

3. Stop the main OpenStack platform services.

```
$ sudo systemctl stop 'openstack-*' 'neutron-*' httpd
```

4. Update the python-tripleoclient package and its dependencies to ensure you have the most current scripts for the minor version update.

```
$ sudo yum update python-tripleoclient
```



5. Upgrade the undercloud.

```
$ openstack undercloud upgrade
```

6. Reboot the node.

```
$ sudo reboot
```

7. Wait until the node reboots, then check the status of all services.

**NOTE:** It can take as much as 10 minutes or more for the openstack-nova-compute to become active after a reboot.

```
$ sudo systemctl list-units "openstack*" "neutron*" "openvswitch*"
```

8. Verify the version of RHEL after the undercloud upgrade.

**NOTE:** Contrail does not support undercloud Red Hat version running with RHEL-7.6 as part of Contrail 4.1.3 release.

```
[root@undercloud ~]# cat /etc/redhat-release
Red Hat Enterprise Linux Server release 7.5 (Maipo)
[root@undercloud ~]#
```

9. Verify the existence of the overcloud and its nodes.

```
$ openstack stack list
```

```
$ ironic node-list
```

10. Verify that all OpenStack servers are Active.

```
$ openstack server list
```

Figure 90: Server List

| ID                                   | Name                                  | Status | Networks            | Image Name     |
|--------------------------------------|---------------------------------------|--------|---------------------|----------------|
| e37480b0-e098-4216-aea4-04b028aa5fb9 | overcloud-contrailanalytics-2         | ACTIVE | ctiplane=192.0.2.11 | overcloud-full |
| c77dcc49-c039-4855-bc91-e217bd34a51e | overcloud-contrailanalytics-1         | ACTIVE | ctiplane=192.0.2.12 | overcloud-full |
| 63d147ec-61e0-4d49-b1cb-6b68b7e5f48f | overcloud-contrailanalytics-0         | ACTIVE | ctiplane=192.0.2.21 | overcloud-full |
| 913d0bcb-9e9b-4987-a858-0389345e7025 | overcloud-controller-0                | ACTIVE | ctiplane=192.0.2.14 | overcloud-full |
| e66e86b1-df4a-4246-8e24-41461d617c18 | overcloud-controller-2                | ACTIVE | ctiplane=192.0.2.15 | overcloud-full |
| 9bbc1552-9ac1-4f27-846c-7756b14b84de | overcloud-controller-1                | ACTIVE | ctiplane=192.0.2.22 | overcloud-full |
| 8f8f5b9b-d5e2-4349-8d46-a488f2b7ab56 | overcloud-contrailanalyticsdatabase-0 | ACTIVE | ctiplane=192.0.2.17 | overcloud-full |
| 505c00f5-6715-48a7-a9ea-61ff157aa28a | overcloud-contrailanalyticsdatabase-2 | ACTIVE | ctiplane=192.0.2.24 | overcloud-full |
| ba8abe03-071b-4326-8fc0-715e93db3e4d | overcloud-contrailanalyticsdatabase-1 | ACTIVE | ctiplane=192.0.2.19 | overcloud-full |
| a7763383-92a0-4304-afc3-8aebf0ed2a05 | overcloud-contrailcontroller-2        | ACTIVE | ctiplane=192.0.2.20 | overcloud-full |
| e381db99-add0-4887-ac96-1e40d4108fc8 | overcloud-contrailcontroller-0        | ACTIVE | ctiplane=192.0.2.18 | overcloud-full |
| c808dcf1-4906-491e-8780-1f099c6a776c | overcloud-novacompute-0               | ACTIVE | ctiplane=192.0.2.16 | overcloud-full |
| 7eabc822-16d6-4fc6-afd7-7ba662ce7e92 | overcloud-contrailcontroller-1        | ACTIVE | ctiplane=192.0.2.23 | overcloud-full |

## Update Red Hat Director Image Archives

The undercloud update process might download new image archives from the rhosp-director images and the rhosp-director-ipa packages. You will have to update your existing system with any new image archives.

1. Check the yum log to determine if new image archives are available.

```
$ sudo grep "rhosp-director-images" /var/log/yum.log
```

```
[stack@undercloud]$ sudo grep "rhosp-director-images" /var/log/yum.log
```

```
Oct 26 15:09:20 Installed: rhosp-director-images-ipa-10.0-20180821.1.el7ost.noarch
```

```
Oct 26 15:10:10 Installed: rhosp-director-images-10.0-20180821.1.el7ost.noarch
```

2. If new image archives are available, replace your current images with the new images. Before deploying any new images, remove any existing images from the images undercloud on the stack user's home (/home/stack/images).

```
$ rm -rf ~/images/*
```

3. Extract the new image archives.

```
mkdir images
cd images
for i in /usr/share/rhosp-director-images/overcloud-full-latest-10.0.tar /usr/share/rhosp-director-images/ironic-python-agent-latest-10.0.tar; do tar -xvf $i; done
```

4. Install the Contrail package on the overcloud image by using the virt-customize command.

```
export LIBGUESTFS_BACKEND=direct /usr/bin/virt-customize -a /home/stack/images/overcloud-
full.qcow2 \
--copy-in /etc/yum.repos.d/mylocalrepo.repo:/etc/yum.repos.d \
--run-command 'yum -y install puppet-tripleo contrail-tripleo-puppet puppet-contrail'\
--run-command ' cp -r /usr/share/contrail-tripleo-puppet/ /usr/share/openstack-puppet/modules/
tripleo/ ' \
--run-command 'rm -fr /var/cache/yum/*' \
--run-command 'yum clean all' \ --selinux-relabel
```

5. Import the new image archives into the undercloud and configure nodes to use the new images.

```
$ openstack overcloud image upload --update-existing --image-path /home/stack/images/
```

6. Verify that the images are uploaded.

```
$ openstack image list
```

| ID                                   | Name                   | Status |
|--------------------------------------|------------------------|--------|
| a518a08c-3c49-47b7-9705-a7da5c90dcc6 | bm-deploy-ramdisk      | active |
| b7d75ff7-926b-426c-aa2e-424f5d9f8328 | bm-deploy-kernel       | active |
| cd7eedc0-2ed2-4a21-8359-9d8ee89374ac | overcloud-full         | active |
| 9c9e46f4-d571-49fb-a485-4653b6a52b44 | overcloud-full-initrd  | active |
| 740b6cb9-5757-475b-896b-49c526807671 | overcloud-full-vmlinuz | active |

7. Show the details of the new image that has been created. The new image will be used to add a new node in the overcloud.

```
$ openstack image show overcloud-full
```

| Field            | Value   |
|------------------|---|
| checksum         | 13e67f5039dc7e69b5bbc494d8838b8d  |
| container_format | bare  |
| created_at       | 2018-10-26T20:02:18.000000  |
| deleted          | False   |
| deleted_at       | None  |
| disk_format      | qcow2   |
| id               | cd7eedc0-2ed2-4a21-8359-9d8ee89374ac  |
| is_public        | True  |
| min_disk         | 0   |
| min_ram          | 0   |
| name             | overcloud-full  |
| owner            | 0cd0e938b0fe46ef9c431715395f9469  |
| properties       | kernel_id='740b6cb9-5757-475b-896b-49c526807671', ramdisk_id='9c9e46f4-d571-49fb-a485-4653b6a52b44' |
| protected        | False   |
| size             | 1469448192  |
| status           | active  |
| updated_at       | 2018-10-26T20:02:52.000000  |
| virtual_size     | None  |

- Verify `contrail-status` on all Contrail nodes. All services in the Contrail nodes, except the controller (OpenStack), should be up and running before proceeding with the upgrade.

```
[stack@undercloud ~]$ for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '=' -f2); do ssh heat-admin@$i sudo contrail-status; done
```

## Prepare Repositories on all Nodes

- Delete existing repositories on all overcloud nodes. Verify each deletion.

```
sudo for ipnode in $(nova list | sed '4,$ !d;$d' | awk -F 'ctlplane=' '{print $2}' | tr -d '|'); do echo cleaning yum repolist on $ipnode && ssh heat-admin@$ipnode 'find /etc/yum.repos.d/ ! -name 'contrail-install.repo' -type f -exec sudo rm -f {} +' ; done
```

- Add new repositories on all overcloud nodes. Verify each addition.

```
sudo for ipnode in $(nova list | sed '4,$ !d;$d' | awk -F 'ctlplane=' '{print $2}' | tr -d '|'); do echo cleaning yum repolist on $ipnode && ssh heat-admin@$ipnode 'curl http://newrepo.contrail41-dev.repo -o /etc/yum.repos.d/localrepo.rep' ; done
```

## Upgrade the Operating System on Contrail Nodes

- Define a list (`$iplist`) that contains all Contrail nodes. Run the following command on undercloud VM as stack user.

```
Iplist=" @IPcontrailController1 @IPcontrailController2 ..."
```



**CAUTION:** Attach the new satellite subscription key on all overcloud nodes before upgrading the overcloud packages. Satellite must be synced with filter end date *29th Oct 2018*. Make sure to clear cache by typing **sudo yum clean all**.

- Upgrade the operating system for all nodes in the `iplist`. Run the following command on undercloud VM as stack user

```
sudo for ipnode in $iplist; do echo -e "\n\n\t*****upgrade node : $ipnode *****" && ssh heat-admin@$ipnode 'sudo yum update -y --disablerepo=*contrail* --skip-broken && exit' ; done
```

- Reboot overcloud contrail compute nodes, if there is any change in the kernel version. This needs to be done before installing contrail packages on compute VM.

Supported kernel versions: 3.10.0-862.11.6.el7.x86\_64 and 3.10.0-957.el7.x86\_64

```
----- [root@overcloud-novacompute-2 ~]# modinfo vrouter filename: /lib/modules/
```

3.10.0-862.11.6.el7.x86\_64/extra/net/vrouter/vrouter.ko version: 4.1.3.0 license: GPL retpoline: Y  
rhelversion: 7.5

## Prepare the Contrail Packages

To prepare the Contrail packages for the installation from a local repository:

1. Navigate to the Contrail repository and perform the following tasks:

- Delete the existing Contrail repositories.

All existing repositories in the undercloud and overcloud will be deleted during these steps.

- Access the Contrail update package.
- Copy the SNMP packages into the repository:
  - net-snmp-5.7.2-37.el7.x86\_64.rpm
  - net-snmp-agent-libs-5.7.2-37.el7.x86\_64.rpm
  - net-snmp-libs-5.7.2-37.el7.x86\_64.rpm
  - net-snmp-utils-5.7.2-37.el7.x86\_64.rpm

In the provided example, all 4 of these files are in the `/mnt/net-snmp/` directory and all files from the directory are copied into the repository.

- Unsubscribe every node with all registered satellite server repositories.
- Delete all repositories on undercloud and overcloud nodes, and replace these deleted repositories with a Contrail repository.
- Clean the yum cache, verify the repository list, and check for yum updates.

A sample procedure:

```
[stack@undercloud ~]#
sudo su -
cd /var/www/html/contrail
rm -rf /var/www/html/contrail/*
#enter the location of the contrail update package
tar -xzf /mnt/contrail-install-packages_4.1.3.0-30-newton.tgz
#copy prerequisite snmp packages; in this setup packages are in /mnt/net-snmp/
cp /mnt/net-snmp/* .
rm -rf /var/www/html/contrail/repodata/usr/bin/createrepo /var/www/html/contrail/subscription-
manager repos --disable=*subscription-manager unregister
```

```

rm -f /etc/yum.repos.d/*
#create local repo file
echo -e '[Contrail]\nname=Contrail Repo\nbaseurl=http://192.168.24.1/contrail\nenabled=1\nngpgcheck=0' > /etc/yum.repos.d/contrail.repo
# disable yum plugins
sed -i 's/plugins=1/plugins=0/g' /etc/yum.conf
yum clean all
rm -rf /var/cache/yum/*
yum check-update
exit
yum repolist

[stack@undercloud ~]#
. stackrc;for ipnode in $(nova list | sed '4,$ !d;$d' | awk -F 'ctlplane=' '{print $2}' | tr -d '|');
do echo "Node $ipnode";
echo "sudo subscription-manager repos --disable=*";
sudo subscription-manager unregister;
sudo rm -f /etc/yum.repos.d/*;
sudo echo -e '[Contrail]\nname=Contrail Repo\nbaseurl=http://192.168.24.1/contrail\nenabled=1\nngpgcheck=0' > /tmp/contrail.repo;
sudo mv /tmp/contrail.repo /etc/yum.repos.d/;
sudo sed -i 's/plugins=1/plugins=0/g' /etc/yum.conf;
sudo yum clean all;sudo rm -rf /var/cache/yum/*;
sudo yum repolist;sudo yum check-update" | ssh heat-admin@$ipnode bash;
done

```

2. Check the undercloud Contrail packages versions for `contrail-tripleo-puppet`, `puppet-contrail`, and `contrail-tripleo-heat-templates`. The newest versions of those packages must be installed before proceeding with the overcloud upgrade. See the following example, with current package versions.

```

[stack@undercloud~]$ rpm -qa | grep contrail

puppet-contrail-4.1.3.0-NN.el7.noarch
contrail-tripleo-heat-templates-4.1.3.0-NN.el7.noarch
contrail-tripleo-puppet-4.1.3.0-NN.el7.noarch

```

## Upgrade the Contrail Heat Templates

You must copy the new Contrail Heat templates package to the undercloud node, while retaining a copy of the Heat templates that were used for the existing deployment.

1. Make a copy of all of the Heat templates that were used for deployment and save the copies, because the existing files will be overwritten by the new versions. The templates to copy are of the form `contrail-services.yaml`, `contrail-net.yaml`, and so on.

**NOTE:** Red Hat does not support changing IP address of the existing cluster as a part of upgrade. Do not change IP address of the cluster while creating new tripleo-heat-templates

2. Copy the new `contrail-tripleo-heat` templates to the undercloud node.

```
cp /home/stack/tripleo-heat-templates /home/stack/tripleo-heat-templates-bk

sudo cp -r /usr/share/contrail-tripleo-heat-templates/environments/contrail /home/stack/
tripleo-heat-templates/environments/

sudo cp -r /usr/share/contrail-tripleo-heat-templates/puppet/services/network/* /home/stack/
tripleo-heat-templates/puppet/services/network
```

**NOTE:** The directory `/home/stack/tripleo-heat-templates` is user defined, it can be *User Defined-directory>/openstack-tripleo-heat-templates*

## Modify the Yum Update Script for TripleO Puppet

Following Puppet commands must be added to the `yum_update` script before starting the upgrade. The script is located at:

`/home/stack/tripleo-heat-templates/extraconfig/tasks/yum_update.sh`

1. Update the following Puppet commands in the `yum_update.sh` after the line `“echo -n “false” > $heat_outputs_path.update_managed_packages”`.

Refer to the following patch for details regarding the exact placement of the commands patch:  
[https://github.com/Juniper/contrail-tripleo-heat-templates/blob/stable/newton/environments/contrail/yum\\_updates.patch](https://github.com/Juniper/contrail-tripleo-heat-templates/blob/stable/newton/environments/contrail/yum_updates.patch)

```
yum install -y contrail-tripleo-puppet puppet-contrail

rsync -a /usr/share/contrail-tripleo-puppet/ /usr/share/openstack-puppet/modules/tripleo/
```

2. Update the fields `*contrail version` and `*contrail repo` in `contrail-services.yaml`.

Default parameter for `contrailVersion` is 4.

Filename: `/home/stack/tripleo-heat-templates/environments/contrail/contrail-services.yaml`.

Add the following parameters:

ContrailVersion: 4

ContrailRepo : *<location of the contrail-41 repo>*

**NOTE:** `/home/stack/tripleo-heat-templates` directory is user defined and it can be directory name under stack user.

## Update the Overcloud Deployment Plan

1. Make a copy of the existing deploy script to the `update-stack.sh` file by re-running the command used for cloud deployment and adding the suffix `-update-plan-only`.

```
<openstack overcloud deploy> -update-plan-only
```

Example:

```
openstack overcloud deploy --update-plan-only --templates /home/stack/tripleo-heat-templates/ \
\
  --roles-file /home/stack/tripleo-heat-templates/environments/contrail/roles_data.yaml \
-e /home/stack/tripleo-heat-templates/environments/puppet-pacemaker.yaml \
-e /home/stack/tripleo-heat-templates/environments/contrail/contrail-services.yaml \
-e /home/stack/tripleo-heat-templates/environments/network-isolation.yaml \
-e /home/stack/tripleo-heat-templates/environments/contrail/contrail-net.yaml \
-e /home/stack/tripleo-heat-templates/environments/ips-from-pool-all.yaml \
-e /home/stack/tripleo-heat-templates/environments/network-management.yaml \
-e /home/stack/tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/environment- \
rhel-registration.yaml \
-e /home/stack/tripleo-heat-templates/environments/hostname-map.yaml \
-e /home/stack/tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/rhel- \
registration-resource-registry.yaml \
--libvirt-type qemu
```



2. If you are using a local repository for the update and the `environment-rhel-registration.yaml` and `rhel-registration-resource-registry.yaml` files are present, delete these lines from the deploy script:

```
-e /home/stack/tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/environment-
rhel-registration.yaml \
-e /home/stack/tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/rhel-
registration-resource-registry.yaml \
```

3. Prepare the YAML files for the update:

- Verify each `.yaml` template referenced in the `update-stack.sh` file contains the original settings that match the files that were backed up.
- In the `contrail-net.yaml` file, adapt all referenced templates from `heat_template_version: newton` to `heat_template_version: 2015-04-30`. Keep all other original installation settings in this file.

4. Update the deployment plan.

```
./update-stack.sh
```

Example

```
[stack@undercloud ~]$ ./update-stack.sh
Removing the current plan files
Uploading new plan files
Started Mistral Workflow. Execution ID: 6c8fb5b7-6eda-4d92-8245-f7ac46bb369d
Plan updated
Deploying templates in the directory /tmp/tripleoclient-CdyN2I/tripleo-heat-templates
Overcloud Endpoint: http://10.87.67.232:5000/v2.0
Overcloud Deployed
[stack@undercloud ~]$
```

## Upgrade the Overcloud



**CAUTION:** The steps in this section are service disrupting, and should only be performed within a maintenance window.



## Overcloud Stack Status

```
[stack@undercloud]# heat stack-list
WARNING (shell) "heat stack-list" is deprecated, please use "openstack stack list" instead
+-----+-----+-----+-----+
+-----+
| id                | stack_name | stack_status  | creation_time
| updated_time      |            |               |
+-----+-----+-----+-----+
+-----+
| e873706c-7fb3-44ba-80dc-30b0fdbd519e | overcloud  | UPDATE_COMPLETE | 2019-03-13T19:20:52Z
| 2019-03-13T22:01:05Z |
+-----+-----+-----+-----+
+-----+
[stack@undercloud ~]$
```

## Contrail Stack Status

```
sudo for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '=' -f2); do ssh heat-admin@$i sudo contrail-status; done
```

## Contrail Version Check

```
for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '=' -f2); do ssh heat-admin@$i sudo contrail-version; done
```

## RELATED DOCUMENTATION

[Upgrade Procedure for RHOSP-based Contrail 3.2.x to Contrail 4.1](#)

[Upgrade Procedure for RHOSP-based Contrail 4.1.1 to Contrail 4.1.2](#)

## Upgrade Procedure for Ubuntu-based Contrail 4.1.3 to Contrail 4.1.4 Using Juju with Netronome SmartNIC

### IN THIS SECTION

- Prerequisites | 468
- Acquire the Software | 468
- Attach Contrail Packages using Juju | 469
- Upgrade the Contrail Clusters | 469

This section presents the steps to upgrade from an Ubuntu-16.04-based Contrail deployment from Contrail version 4.1.3 to Contrail version 4.1.4 using Juju charms.

### Prerequisites

These instructions assume that these requirements for installing Contrail Release 4.1.3 are already present in your environment:

- MaaS Server—MaaS version 2.3 is installed on this server. This procedure was tested using MaaS version 2.3.5.
- Juju Controller—Juju version 2.3 is installed, and the Juju controller is bootstrapped on a VM or a bare metal server. This procedure was tested using Juju version 2.3.7-xenial-amd64.
- A repository to get Netronome, patched Openstack packages, and Contrail vRouter packages is operational.
- A Contrail Controller using Ubuntu 16.04 xenial is operational.
- A Contrail cluster with one or more compute nodes using Agilio SmartNICs.

### Acquire the Software

To acquire the Contrail 4.1.4 software images to perform this procedure:

1. Go to the [Juniper Networks Support site for Contrail](#).
2. Select **OS** as *Contrail* and **Version** as *4.1.4* from the drop-down menus.
3. Download the *contrail-cloud-docker\_4.1.4.0-63-ocata\_xenial.tgz* file.

4. Extract the following images from the *contrail-cloud-docker\_4.1.4.0-63-ocata\_xenial.tgz* file:

- Contrail Analytics package: *contrail-analytics-ubuntu16.04-4.1.4.0-63.tar.gz*.
- Contrail Analytics Database package: *contrail-analyticsdb-ubuntu16.04-4.1.4.0-63.tar.gz*.
- Contrail Controller package: *contrail-controller-ubuntu16.04-4.1.4.0-63.tar.gz*

The images need to be extracted because the Contrail Analytics, Contrail Analytics Database, and Contrail Controller packages must be upgraded individually to perform this upgrade.

## Attach Contrail Packages using Juju

The Contrail Controller, Contrail Analytics, and Contrail Analytics DB packages need to be attached using Juju to perform this upgrade.

To attach these software packages into Juju:

1. Attach the Contrail Controller, Contrail Analytics, & Contrail Analytics DB packages into Juju:

```
juju attach contrail-analytics contrail-analytics=/home/jenkins/docker/contrail-analytics-ubuntu16.04-4.1.4.0-63.tar.gz
juju attach contrail-controller contrail-controller=/home/jenkins/docker/contrail-controller-ubuntu16.04-4.1.4.0-63.tar.gz
juju attach contrail-analyticsdb contrail-analyticsdb=/home/jenkins/docker/contrail-analyticsdb-ubuntu16.04-4.1.4.0-63.tar.gz
```

2. Check status of the software image attachments into Juju using the `juju status` command.

Wait for the `juju status` command output to indicate that the upgrade is successful. The output in the `juju status` should indicate that all processes are *Active* and all machine states are *started*.

## Upgrade the Contrail Clusters

This section provides the steps to update the Contrail clusters for this upgrade.

It includes the following sections:

### Upgrade the Contrail Controllers

The Contrail controllers must be upgraded one by one to complete this procedure.

To upgrade the Contrail controllers:

1. SSH into the Contrail controller server and decommission the Contrail controller from the Cassandra cluster:

```
sudo docker exec -it contrail-controller /usr/bin/nodetool decommission
```

2. Remove the Contrail Controller container:

```
sudo docker rm -f contrail-controller
```

3. Update the hooks to the Contrail Controller from the Juju Controller:

```
juju run --application contrail-controller hooks/update-status
```

4. Wait for the Contrail status for all packages on the upgrading node to change to *active*. This step can take up to 10 minutes.

Enter the **contrail-status** command to check status. All packages in the *Contrail Control* section of the output must move to the *active* state before proceeding.

5. Check Juju status by entering the `juju status` command.

All Contrail components in this output should be in the *active* state.

6. After each controller update, check the controllers to make sure the databases are consistent across all controllers:

- Enter the **nodetool describeclasser** command. Confirm that the *schema version* output is identical on all 3 controllers.
- Enter the **echo stat | nc localhost 2181** command. The *node count* output should be identical on all 3 controllers.
- Ensure that the **contrail-status** output is *active* for all components in all 3 controllers.

If your upgrade is not successful after 15 minutes, retry steps 1 through 5.

If you need to decommission a node that is not upgrading successfully, use the **nodetool removemode *node-ID*** command.

7. Repeat steps 1 through 6 for all other Contrail controller nodes.

## Upgrade Contrail Analytics Nodes

To upgrade the Contrail Analytics nodes:

1. SSH into the first Contrail Analytics node and remove the Contrail Analytics container:

```
sudo docker rm -f contrail-analytics
```

2. Confirm Juju status using the `juju status` command.

The output in the `juju status` should indicate that all processes are *Active* and all machine states are *started*.

3. From the MaaS server, update hooks to the Contrail Analytics controller:

```
juju run --application contrail-analytics/0 hooks/update-status
```

4. Wait for the Contrail status for all packages on the upgrading node to change to *active*. This step can take up to 10 minutes.

Enter the `contrail-status` command to check status. All packages in the *Contrail Analytics* section of the output must move to the *active* state before proceeding.

5. Repeat steps 1 through 4 for all other Contrail Analytics nodes.

## Upgrade Analytics Database Nodes

To upgrade the Contrail Analytics database nodes:

1. SSH into a Contrail analytics database server and decommission the node from the Cassandra cluster:

```
sudo docker exec -it contrail-analyticsdb /usr/bin/nodetool decommission
```

2. Remove the AnalyticsDB container:

```
sudo docker rm -f contrail-analyticsdb
```

3. From the Juju controller, update the hooks to the Contrail Analytics DB controller:

```
juju run --application contrail-analyticsdb hooks/update-status
```

4. Wait for the Contrail status for all packages on the upgrading node to change to *active*. This step can take up to 10 minutes.

Enter the **contrail-status** command to check status. All packages in the *Contrail Database* section of the output must move to the *active* state before proceeding.

5. Check Juju status by entering the `juju status` command.

All Contrail components in this output should be in the *active* state.

6. After each analytics database node update, check the nodes to ensure the databases are consistent inside the contrail analytics database containers:

- Enter the **nodetool describecluster** command. Confirm that the *schema version* output is identical on all 3 nodes.
- Enter the **echo stat | nc localhost 2181** command. The *node count* output should be identical on all 3 nodes.
- Ensure that the **contrail-status** output is *active* for all components in all 3 contrail analytics db nodes.

If your upgrade is not successful after 15 minutes, retry steps 1 through 5.

If you need to decommission a node that is not upgrading successfully, use the **nodetool removemode *node-ID*** command.

7. Repeat steps 1 through 6 for all other Contrail Analytics database nodes.

## Updating the Neutron Plugin and the vRouter Agent

The process for updating the neutron plugin and the vRouter agent is different for compute nodes than it is for other nodes.

This section covers both procedures and includes these sections:

### Updating the Neutron Plugin and the vRouter Agent on Non-Compute Nodes

Use this procedure to update the Neutron Plugin and the vRouter agent on all non-compute nodes in your environment:



**NOTE:** This procedure assumes that the APT Get repository was created during the previous installation, and that the latest Contrail packages can be placed into the repository.

1. SSH into the Neutron API plugin unit.
2. From the Neutron API plugin unit, get the latest APT Get update:

```
sudo apt-get update
```

3. Upgrade APT GET:

```
sudo apt-get upgrade
```

**NOTE:** This step shows how to upgrade APT get for all packages. You can also manually update the `neutron-plugin-contrail` and `python-contrail` packages to complete this step, if you'd rather not perform the complete upgrade. This procedure does not provide the steps to manually update these packages.

4. Restart the Neutron service:

```
sudo systemctl restart neutron-server.service
```

## Updating the Neutron Plugin and the vRouter Agent on Compute Nodes

Use this procedure to update the Neutron Plugin and the vRouter agent on all compute devices in your environment:

**NOTE:** This procedure assumes that the APT Get repository was created during the previous installation, and that the latest Contrail packages can be placed into the repository.

1. SSH into the Neutron API plugin unit.

2. From the Neutron API plugin unit, get the latest APT Get update:

```
sudo apt-get update
```

3. Upgrade APT GET:

```
sudo apt-get upgrade
```

**NOTE:** This step shows how to upgrade APT get for all packages. You can also manually update the following packages to complete this step:

- contrail-lib
- contrail-nodemgr
- contrail-setup
- contrail-utils
- contrail-vrouter-agent
- contrail-vrouter-common
- contrail-vrouter-dkms
- contrail-vrouter-init
- contrail-vrouter-utils
- python-contrail
- python-contrail-vrouter-api
- python-opencontrail-vrouter-netns

This procedure does not provide the steps to manually update these packages.

4. Upgrade the vRouter agent and, if using Netronome SmartNICs, the netronome plugin.
  - If you are performing this procedure on a compute node without a Netronome SmartNIC:

**NOTE:** The network connection over the vhost is down while this procedure is performed. Traffic will be lost.

- a. Stop the Contrail vRouter agent:

```
sudo systemctl stop contrail-vrouter-agent
```

- b. Remove the Contrail vRouter module:

```
sudo rmmmod vrouter
```

- c. Insert the vRouter module:

```
sudo insmod /lib/modules/4.4.0-116-generic/updates/dkms/vrouter.ko
```

- d. Activate the vhost:

```
sudo ifup vhost0
```

- e. Restart the Contrail vRouter agent:

```
sudo systemctl start contrail-vrouter-agent
```

- If you are performing this procedure on a compute node with a Netronome SmartNIC:

**NOTE:** The network connection over the vhost is down while this procedure is performed. Traffic will be lost.

- a. Stop the Contrail vRouter agent:

```
sudo systemctl stop contrail-vrouter-agent
```

- b. Stop the Virtio forwarder module:

```
sudo systemctl stop virtio-forwarder
```

- c. Stop the vRouter control module:

```
sudo /opt/netronome/bin/ns-vrouter-ctl stop
```

- d. Restart the Virtio forwarder module:

```
sudo systemctl start virtio-forwarder
```

- e. Restart the Contrail vRouter agent:

```
sudo /opt/netronome/bin/ns-vrouter-ctl start
```

- f. Activate the vhost:

```
sudo ifup vhost0
```

- g. Restart the Contrail vRouter agent:

```
sudo systemctl start contrail-vrouter-agent
```

5. Verify Contrail status:

```
sudo contrail-status
```

All packages in the *Contrail vRouter* section of the output should be in the *active* state. This step can take several minutes.

## RELATED DOCUMENTATION

| [Deploying Contrail Release 4.1 with Netronome SmartNICs by Using Juju](#)

## Upgrade Procedure for RHOSP-based Contrail 4.1.3 to Contrail 4.1.4

### IN THIS SECTION

- [Prerequisites | 477](#)
- [Post-Installation | 478](#)
- [Acquire the Software | 479](#)
- [Upgrade the Undercloud | 479](#)
- [Update Red Hat Director Image Archives | 484](#)
- [Upgrade the Operating System on Contrail Nodes | 486](#)
- [Prepare the Contrail Packages | 487](#)
- [Upgrade the Contrail Heat Templates | 487](#)
- [Modify the Yum Update Script for TripleO Puppet | 488](#)
- [Update the Overcloud Deployment Plan | 489](#)
- [Upgrade the Overcloud | 490](#)
- [Upgrade Cautions | 491](#)

This section presents the steps to upgrade a RHOSP-based Contrail deployment from Contrail version 4.1.3 to Contrail version 4.1.4.

### Prerequisites

Ensure you have a cloud up and running with RHOSP10 and Contrail 4.1.3 before you proceed with the upgrade procedure.

This procedure has been validated with the following Contrail, Red Hat, and OpenStack versions.

**Table 30: Pre-Installation Software Versions**

| Contrail Version | Red Hat Version                       | OpenStack Version                         |
|------------------|---------------------------------------|---|
| 4.1.3            | RHEL 7.6 (3.10.0-957.el7.x86_64)      | RHOSP10 (packages dated October 29, 2018) |
| 4.1.3            | RHEL 7.5 (3.10.0-862.11.6.el7.x86_64) | RHOSP10 (packages dated October 29, 2018) |



**CAUTION:** Set the Red Hat Satellite filter end date to December 9, 2019 before proceeding with the upgrade.

## Post-Installation

After the installation, you'll have a cloud networking running RHOSP10 and Contrail 4.1.4. The Red Hat Enterprise Linux (RHEL) kernel version updates to 7.7 during this procedure.

[Table 31 on page 478](#) summarizes the post-installation software versions.

**Table 31: Post Installation Software Summary**

| Contrail Version | Red Hat Version   | OpenStack Version                         |
|------------------|---|---|
| 4.1.4            | RHEL 7.7 (3.10.0-1062.el7.x86_64)<br>RHEL 7.7 (3.10.0-1062.1.2.el7.x86_64)<br>RHEL 7.7 (3.10.0-1062.9.1.el7.x86_64) | RHOSP10 (packages dated December 9, 2019) |

Contrail version R4.1.4 supports net-snmp package version 5.7.2-43 to support SNMP. The net-snmp packages come from Red Hat, with the exception of the *net-snmp-python-5.7.2-43.el7.x86\_64.rpm* package which is provided in the Contrail repository.

[Table 32 on page 478](#) summarizes the net-snmp depend packages and their associated repository locations.

**Table 32: Post Installation Software Summary**

| Net-SNMP Depend Packages                           | Repository        |
|--|-------------------|
| <i>net-snmp-5.7.2-43.el7.x86_64.rpm</i>            | Red Hat Satellite |
| <i>net-snmp-agent-libs-5.7.2-43.el7.x86_64.rpm</i> | Red Hat Satellite |
| <i>net-snmp-libs-5.7.2-43.el7.x86_64.rpm</i>       | Red Hat Satellite |

**Table 32: Post Installation Software Summary (Continued)**

| Net-SNMP Depend Packages                       | Repository        |
|--|-------------------|
| <i>net-snmp-python-5.7.2-43.el7.x86_64.rpm</i> | Contrail          |
| <i>net-snmp-utils-5.7.2-43.el7.x86_64.rpm</i>  | Red Hat Satellite |

## Acquire the Software

To download the software images for this procedure:

1. Go to the [Juniper Networks Support site for Contrail](#).
2. Select **OS** as *Contrail* and **Version** as *4.1.4*. Download the images that apply to your environment.

## Upgrade the Undercloud

1. Log in to the undercloud as the stack user.

```
$ su - stack
```

2. Update the Contrail repositories.

- Backup the Contrail 4.1.3 packages to a repository with a different name. In this example, the packages are moved to a repository named *contrail-R4-1-3*.

```
[stack@undercloud ~]$ cd /var/www/html/
[stack@undercloud html]$ sudo mv contrail/ contrail-R4-1-3
```

- Create a new repository directory to store the Contrail 4.1.4 packages:

```
[stack@undercloud html]$ sudo mkdir contrail
```

3. Copy the downloaded file—in the provided sample, the file is *contrail-install-packages\_4.1.4.0-63-newton.tgz*—to the Contrail repository created in Step 2.

**NOTE:** This step assumes that you've already downloaded the Contrail software. See ["Acquire the Software" on page 479](#).

```
[stack@undercloud contrail]$ ls -lrt
total 377104
-rw-r--r--. 1 root root 386151602 Mar 14 06:58 contrail-install-packages_4.1.4.0-63-
newton.tgz
```

4. Untar the downloaded tgz file.

```
[stack@undercloud contrail]$ sudo tar -xvf contrail-install-packages_4.1.4.0-63-newton.tgz
```

5. Create a repository in the new directory:

```
[stack@undercloud contrail]$ pwd
/var/www/html/contrail

[stack@undercloud contrail]$ sudo createrepo .
```

If the **createrepo** command is not available, download the createrepo package from Red Hat (Red Hat subscription required).

6. (Clusters deployed using Swift Puppet files only) If your Contrail 4.1 cluster was deployed using Swift Puppet, perform these steps:
  - a. Remove overcloud artifacts from the undercloud:

```
[stack@undercloud ~]$ swift delete overcloud-artifacts
puppet-modules.tgz
overcloud-artifacts
```

- b. Delete the *deployments-artifacts.yaml* file if the file is present.

```
[stack@undercloud ~]$ ls /home/stack/.tripleo/environments/deployment-artifacts.yaml
[stack@undercloud ~]$ rm -rf /home/stack/.tripleo/environments/deployment-artifacts.yaml
```



- c. Clean the repositories and confirm that all repositories are available.

```
[stack@undercloud ~]$ sudo yum clean all
[stack@undercloud ~]$ sudo yum repolist
```

7. Stop the main OpenStack platform services.

```
$ sudo systemctl stop 'openstack-*' 'neutron-*' httpd
```

8. Update the python-tripleoclient package and its dependencies to ensure you have the most current scripts for the minor version update.

```
$ sudo yum update python-tripleoclient
```

9. Upgrade the undercloud.

```
$ openstack undercloud upgrade
```

10. Reboot the node.

```
$ sudo reboot
```

Wait for the node to reboot. The reboot process can take 10 or more minutes to complete.

11. Ensure the undercloud has the latest Contrail R4.1.4 contrail packages:

```
[stack@undercloud ~]$ rpm -qa | grep contrail

puppet-contrail-4.1.4.0-X.el7.noarch
contrail-tripleo-heat-templates-4.1.4.0-x.el7.noarch
contrail-tripleo-puppet-4.1.4.0-x.el7.noarch
python-gevent-1.1rc5-1contrail1.el7.x86_64
```

12. Ensure the undercloud has the latest RHOSP images:

```
[stack@undercloud]$ rpm -qa | grep direct

rhosp-director-images-10.0-20180821.1.el7ost.noarch
rhosp-director-images-10.0-20190829.1.el7ost.noarch
rhosp-director-images-10.0-20190918.1.el7ost.noarch
rhosp-director-images-ipa-10.0-20190829.1.el7ost.noarch
rhosp-director-images-ipa-10.0-20180821.1.el7ost.noarch
rhosp-director-images-ipa-10.0-20190918.1.el7ost.noarch
```

13. Review the **ironic node-list** output to confirm the following statuses for each package::

- **Power state** is *power on*.
- **Provision State** is *active*.
- **Maintenance** is *False*.

```
[stack@undercloud ~]$ ironic node-list
+-----+-----+-----+
| Name                | Power | Provisioning | Maintenance |
|                    | State | State        |             |
+-----+-----+-----+
| controller-3       | power on | active      | False      |
| compute-5c5s35     | power on | active      | False      |
| contrail-controller1 | power on | active      | False      |
| contrail-analytics1 | power on | active      | False      |
| contrail-controller-3 | power on | active      | False      |
| contrail-controller-2 | power on | active      | False      |
| contrail-analytics-database1 | power on | active      | False      |
| controller-2       | power on | active      | False      |
| controller1        | power on | active      | False      |
| compute-5c5s37     | power on | active      | False      |
| compute-5c5s36     | power on | active      | False      |
| contrail-analytics-2 | power on | active      | False      |
| contrail-analytics-3 | power on | active      | False      |
| compute-5c5s38     | power on | active      | False      |
| contrail-analytics-database-3 | power on | active      | False      |
| contrail-analytics-database-2 | power on | active      | False      |
+-----+-----+-----+
```

**NOTE:** This output presentation has been modified for readability. The *UUID* and *Instance UUID* fields were removed as part of this modification.

14. Verify that all OpenStack servers are in the Active state.

```
[stack@undercloud ~]$ openstack server list
+-----+-----+
| Name                | Status |
+-----+-----+
```

```

| overcloud-contrailanalytics-2-4-1-4-7-7      | ACTIVE |
| overcloud-controller-0-4-1-4-7-7            | ACTIVE |
| overcloud-contrailanalytics-0-4-1-4-7-7     | ACTIVE |
| overcloud-contrailanalyticsdatabase-2-4-1-4-7-7 | ACTIVE |
| overcloud-contrailanalytics-1-4-1-4-7-7     | ACTIVE |
| overcloud-contrailanalyticsdatabase-0-4-1-4-7-7 | ACTIVE |
| overcloud-contrailcontroller-1-4-1-4-7-7    | ACTIVE |
| overcloud-contrailanalyticsdatabase-1-4-1-4-7-7 | ACTIVE |
| overcloud-contrailcontroller-2-4-1-4-7-7    | ACTIVE |
| overcloud-contrailcontroller-0-4-1-4-7-7    | ACTIVE |
| compute-0-4-1-4-rhel-7-7                    | ACTIVE |
| overcloud-contraildpdk-0-4-1-4-7-7          | ACTIVE |
| overcloud-contraildpdk-1-4-1-4-7-7          | ACTIVE |
| compute-1-4-1-4-rhel-7-7                    | ACTIVE |
+-----+-----+

```

**NOTE:** This output presentation has been modified for readability. The *ID*, *Image Name*, and *Networks* fields were removed as part of this modification.

15. If new image archives are available, replace your current images with the new images.

Before uploading the new images onto the undercloud node, move any existing images from the images directory on the stack user's home directory (/home/stack/images).

```
$ mv /home/stack/images /home/stack/images-old
```

16. Extract the new image archives.

```

mkdir images
cd images
for i in /usr/share/rhosp-director-images/overcloud-full-latest-10.0.tar /usr/share/rhosp-
director-images/ironic-python-agent-latest-10.0.tar; do tar -xvf $i; done

```

17. Import the new image archives into the undercloud and configure the nodes to use the new images.

```
$ openstack overcloud image upload --update-existing --image-path /home/stack/images/
```

18. Verify that the images are uploaded:

```
$ glance image-list
```

19. Observe the contrail-status on all Contrail nodes. All services in the Contrail nodes, except the controller (OpenStack), should be up and running before proceeding with the upgrade.

```
[stack@undercloud ~]$ source stackrc
[stack@undercloud ~]$ for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '='
-f2); do ssh heat-admin@$i sudo contrail-status; done
```

20. Ensure that all overcloud node contrail repository pointers are properly pointing to the contrail repository.

*Contrail Analytics Example:*

```
[root@overcloud-contrailanalytics-0 heat-admin]# cat /etc/yum.repos.d/contrail.repo
[Contrail]
name=Contrail Repo
baseurl=http://192.168.24.1/contrail
enabled=1
gpgcheck=0
protect=1
metadata_expire=30
```

## Update Red Hat Director Image Archives

The undercloud update process might download new image archives from the rhosp-director images and the rhosp-director-ipa packages. You will have to update your existing system with any new image archives.

1. Check the yum log to determine if new image archives are available.

```
$ sudo grep "rhosp-director-images" /var/log/yum.log
```

```
[stack@undercloud]$ sudo grep "rhosp-director-images" /var/log/yum.log
```

```
Dec 12 15:09:20 Installed: rhosp-director-images-ipa-10.0-20190918.1.el7ost.noarch
```

```
Dec 12 15:10:10 Installed: rhosp-director-images-10.0-20190918.1.el7ost.noarch
```

2. If new image archives are available, replace your current images with the new images. Before deploying any new images, remove any existing images from the images undercloud on the stack user's home (/home/stack/images).

```
$ rm -rf ~/images/*
```

3. Extract the new image archives.

```
mkdir images
cd images
for i in /usr/share/rhosp-director-images/overcloud-full-latest-10.0.tar /usr/share/rhosp-director-images/ironic-python-agent-latest-10.0.tar; do tar -xvf $i; done
```

4. Install the Contrail package on the overcloud image by using the virt-customize command.

```
export LIBGUESTFS_BACKEND=direct /usr/bin/virt-customize -a /home/stack/images/overcloud-full.qcow2 \
-copy-in /etc/yum.repos.d/mylocalrepo.repo:/etc/yum.repos.d \
-run-command 'yum -y install puppet-tripleo contrail-tripleo-puppet puppet-contrail'\
-run-command ' cp -r /usr/share/contrail-tripleo-puppet/ /usr/share/openstack-puppet/modules/tripleo/ ' \
-run-command 'rm -fr /var/cache/yum/*' \
-run-command 'yum clean all' \ -selinux-relabel
```

5. Import the new image archives into the undercloud and configure nodes to use the new images.

```
$ openstack overcloud image upload -update-existing -image-path /home/stack/images/
```

6. Verify that the images are uploaded.

```
$ openstack image list
```

```
+-----+-----+-----+
| ID                                     | Name                               | Status |
+-----+-----+-----+
| a518a08c-3c49-47b7-9705-a7da5c90dcc6 | bm-deploy-ramdisk                 | active |
| b7d75ff7-926b-426c-aa2e-424f5d9f8328 | bm-deploy-kernel                   | active |
| cd7eedc0-2ed2-4a21-8359-9d8ee89374ac | overcloud-full                     | active |
| 9c9e46f4-d571-49fb-a485-4653b6a52b44 | overcloud-full-initrd              | active |
| 740b6cb9-5757-475b-896b-49c526807671 | overcloud-full-vmlinuz             | active |
+-----+-----+-----+
```

7. Show the details of the new image that has been created. The new image will be used to add a new node in the overcloud.

```
$ openstack image show overcloud-full
```

8. Verify contrail-status on all Contrail nodes. All services in the Contrail nodes, except the controller (OpenStack), should be up and running before proceeding with the upgrade.

```
[stack@undercloud ~]$ for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '=' -f2); do ssh heat-admin@$i sudo contrail-status; done
```

## Upgrade the Operating System on Contrail Nodes

To upgrade the operating system on Contrail nodes:

1. Define a list (\$iplist) that contains all Contrail nodes. Run the following command on undercloud VM as a stack user.

```
Iplist=" @IPcontrailController1 @IPcontrailController2 ..."
```



**CAUTION:** Attach the new satellite subscription key on all overcloud nodes before upgrading the overcloud packages. Satellite must be synced with filter end date *9th Dec 2019*. Make sure to clear cache by typing **sudo yum clean all**.

2. Upgrade the operating system for all nodes in the iplist.

Run the following command on undercloud VM as a stack user:

```
sudo for ipnode in $iplist; do echo -e "\n\n\t*****upgrade node : $ipnode *****" && ssh heat-admin@$ipnode 'sudo yum update -y --disablerepo=*contrail* --skip-broken && exit' ; done
```

3. (Compute nodes only) Reboot overcloud contrail compute nodes. After the reboot, stop the supervisor-vrouter service.

This step needs to be performed before installing contrail packages on the compute VM.

Compute services may go down after rebooting with the latest kernel. These services return later in this procedure during the **openstack overcloud deploy** process.

*Reboot Procedure:*

```
[root@compute-1-7-6 modules]# sudo reboot
Connection to 192.0.2.16 closed by remote host.
Connection to 192.0.2.16 closed.
```

*Post-Reboot:*

```
[stack@undercloud-R4-1-2-b22 ~]$ ssh heat-admin@192.0.2.16
Warning: Permanently added '192.0.2.16' (ECDSA) to the list of known hosts.
Last login: Sat Dec  7 03:46:07 2019 from gateway
[heat-admin@compute-1-7-6 ~]$ sudo su
[root@compute-1-7-6 heat-admin]# contrail-status
vRouter is NOT PRESENT

== Contrail vRouter ==
supervisor-vrouter:          active
contrail-vrouter-agent      initializing
contrail-vrouter-nodemgr    initializing
```

*Stop the supervisor-vrouter service.*

```
[root@compute-1-7-6 heat-admin]# service supervisor-vrouter stop
Stopping supervisor-vrouter (via systemctl): [ OK ]

[root@compute-1-7-6 heat-admin]# contrail-status
vRouter is NOT PRESENT

== Contrail vRouter ==
supervisor-vrouter: inactive
unix:///var/run/supervisord_vrouter.sockno
```

**Prepare the Contrail Packages**

Check the undercloud Contrail packages versions for `contrail-tripleo-puppet`, `puppet-contrail`, and `contrail-tripleo-heat-templates`.

```
[stack@undercloud~]$ rpm -qa | grep contrail
```

**Upgrade the Contrail Heat Templates**

You must copy the new Contrail Heat templates package to the undercloud node, while retaining a copy of the Heat templates that were used for the existing deployment.

1. Make a copy of all of the Heat templates that were used for deployment and save the copies, because the existing files will be overwritten by the new versions. The templates to copy are of the form `contrail-services.yaml`, `contrail-net.yaml`, and so on.

**NOTE:** Red Hat does not support changing IP address of the existing cluster as a part of upgrade. Do not change IP address of the cluster while creating new tripleo-heat-templates

2. Copy the new `contrail-tripleo-heat` templates to the undercloud node.

```
sudo cp -r /usr/share/contrail-tripleo-heat-templates/environments/contrail /home/stack/
tripleo-heat-templates/environments/

sudo cp -r /usr/share/contrail-tripleo-heat-templates/puppet/services/network/* /home/stack/
tripleo-heat-templates/puppet/services/network
```

**NOTE:** The directory `/home/stack/tripleo-heat-templates` is user defined, it can be *User Defined-directory*>/*openstack-tripleo-heat-templates*

## Modify the Yum Update Script for TripleO Puppet

Following Puppet commands must be added to the `yum_update` script before starting the upgrade. The script is located at:

**`/home/stack/tripleo-heat-templates/extraconfig/tasks/yum_update.sh`**

1. Update the following Puppet commands in the `yum_update.sh` after the line `"echo -n "false" > $heat_outputs_path.update_managed_packages"`.

Refer to the following patch for details regarding the exact placement of the commands patch:  
[https://github.com/Juniper/contrail-tripleo-heat-templates/blob/stable/newton/environments/contrail/yum\\_updates.patch](https://github.com/Juniper/contrail-tripleo-heat-templates/blob/stable/newton/environments/contrail/yum_updates.patch)

```
yum install -y contrail-tripleo-puppet puppet-contrail

rsync -a /usr/share/contrail-tripleo-puppet/ /usr/share/openstack-puppet/modules/tripleo/
```

2. Update the fields `*contrail version` and `*contrail repo` in `contrail-services.yaml`.

Default parameter for `contrailVersion` is 4.



Filename: `/home/stack/tripleo-heat-templates/environments/contrail/contrail-services.yaml`.

Add the following parameters:

ContrailVersion: 4

ContrailRepo : *<location of the contrail-41 repo>*

**NOTE:** `/home/stack/tripleo-heat-templates` directory is user defined and it can be directory name under stack user.

## Update the Overcloud Deployment Plan

1. Update the current plan by re-running the command used for cloud deployment and adding the suffix `-update-plan-only`.

```
openstack overcloud deploy -update-plan-only
```

Example:

```
openstack overcloud deploy --update-plan-only --templates /home/stack/tripleo-heat-templates/ \
  \
  --roles-file /home/stack/tripleo-heat-templates/environments/contrail/roles_data.yaml \
  -e /home/stack/tripleo-heat-templates/environments/puppet-pacemaker.yaml \
  -e /home/stack/tripleo-heat-templates/environments/contrail/contrail-services.yaml \
  -e /home/stack/tripleo-heat-templates/environments/contrail/network-isolation.yaml \
  -e /home/stack/tripleo-heat-templates/environments/contrail/contrail-net.yaml \
  -e /home/stack/tripleo-heat-templates/environments/ips-from-pool-all.yaml \
  -e /home/stack/tripleo-heat-templates/environments/network-management.yaml \
  -e /home/stack/tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/environment- \
  rhel-registration.yaml \
  -e /home/stack/tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/rhel- \
  registration-resource-registry.yaml \
  --libvirt-type qemu
```

2. Make a copy of the existing deploy script to the `update-stack.sh`. The `update-stack.sh` is the script used to update the overcloud plan, and it references the same templates that were used to deploy the stack. All files used for the overcloud update should be identical to the files used for deployment, except `contrail-services` file that was updated with the latest `contrail-version` and `contrail-repo`.

```
cp deploy.sh update-stack.sh
```

### 3. Update the deployment plan.

```
./update-stack.sh
```

Example:

```
[stack@undercloud ~]$ ./update-stack.sh
  Removing the current plan files
  Uploading new plan files
  Started Mistral Workflow. Execution ID: 998a1b40--a034-8cff453acfb1
  Plan updated
  Deploying templates in the directory /tmp/tripleoclient-JulIDe/tripleo-heat-templates
  Overcloud Endpoint: http://10.0.0.35:5000/v2.0
  Overcloud Deployed
```

## Upgrade the Overcloud



**CAUTION:** The steps in this section are service disrupting, and should only be performed within a maintenance window.

### 1. Update the overcloud stack.

```
$ openstack overcloud update stack -i overcloud
on_breakpoint: ['overcloud-contrailanalyticsdatabase-0']
Breakpoint reached, continue? Regexp or Enter=proceed (will clear
4386bdc7-5087-4a4d-865c-0b0181ce9345), no=cancel update, C-c=quit interactive mode:
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
COMPLETE
update finished with status COMPLETE
```

### 2. Verify the overcloud stack status, the contrail-status, and the contrail-version after the upgrade.

## Overcloud Stack Status

```
[stack@undercloud ~]$ openstack stack list
+-----+-----+-----+-----+
| Stack Name | Stack Status   | Creation Time       | Updated Time       |
+-----+-----+-----+-----+
| overcloud  | UPDATE_COMPLETE | 2019-12-06T23:30:26Z | 2019-12-09T22:40:01Z |
+-----+-----+-----+-----+
```

**NOTE:** The `openstack stack list` output presentation has been modified for readability. The `ID` field was removed as part of this modification.

## Contrail Stack Status

```
sudo for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '=' -f2); do ssh heat-admin@$i sudo contrail-status; done
```

## Contrail Version Check

```
for i in $(nova list | grep contrail | awk '{print $12}' | cut -d '=' -f2); do ssh heat-admin@$i sudo contrail-version; ssh heat-admin@$i sudo contrail-status -d ; done
```

## Upgrade Cautions



**CAUTION:** The steps to perform the overcloud upgrade are service disrupting, and should only be performed within a maintenance window.

The upgrade procedure may fail due to packages conflicts in Contrail analytics nodes. Some observed failures due to packages conflicts are detailed in this section. Continue with the deployment after applying the recommended solution.

### Analytics Node snmp-lib Version Conflict

Error message: Protected multilib versions: 1:net-snmp-libs-5.7.2-37.el7.x86\_64 != 1:net-snmp-libs-5.7.2-33.el7\_5.2.i686

Solution:

```
rpm -e --nodeps net-snmp-libs
```

## Services Need Manual Restart After Upgrade

Services may need to be restarted after performing this upgrade. The services might continue to run using Contrail 4.1.3-related processes for a period of time.

Enter the **contrail-status** command to see if the processes continued to run through the upgrade, and monitor the warning messages that appear.

Manually restart the services if you run into this issue.

In the following example, this issue is seen for the Contrail Analytics services immediately after the upgrade:

```
[heat-admin@overcloud-contrailanalytics ~]$ sudo contrail-status -d
Warning: supervisor-analytics.service changed on disk. Run 'systemctl daemon-reload' to reload
units.
== Contrail Analytics ==
supervisor-analytics:      active
contrail-alarm-gen         active    pid 975462, uptime 15 days, 19:07:11
contrail-analytics-api     active    pid 127224, uptime 20 days, 19:48:28
contrail-analytics-nodemgr active    pid 127219, uptime 20 days, 19:48:28
contrail-collector         active    pid 127222, uptime 20 days, 19:48:28
contrail-query-engine      active    pid 127223, uptime 20 days, 19:48:28
contrail-snmp-collector    active    pid 127220, uptime 20 days, 19:48:28
contrail-topology          active    pid 127221, uptime 20 days, 19:48:28
```

## Dynamic Kernel Module Support (DKMS) for vRouter

Dynamic Kernel Module Support (DKMS) is a framework provided by Linux to automatically build out-of-tree driver modules for Linux kernels whenever the Linux distribution upgrades the existing kernel to a newer version.

In Contrail, the vRouter kernel module is an out-of-tree, high performance packet forwarding module that provides advanced packet forwarding functionality in a reliable and stable manner. Contrail provides a DKMS-compatible source package for Ubuntu so that if you deploy an Ubuntu-based Contrail system you do not need to manually compile the kernel module each time the Linux deployment gets upgraded.

The `contrail-vrouter-dkms` package provides the DKMS compatibility for Contrail. Prior to installing the `contrail-vrouter-dkms` package, you must install both the DKMS package and the `contrail-vrouter-utils`

package, because the `contrail-vrouter-dkms` package is dependent on both. Installing the `contrail-vrouter-dkms` package adds the vRouter sources to the DKMS database, builds the vRouter module, and installs it in the existing kernel modules tree. When a kernel upgrade occurs, DKMS ensures that the module is compiled for the newer kernel and installed in the proper location so that upon reboot, the newer module can be used with the upgraded kernel.

For more information about DKMS, refer to:

- DKMS Ubuntu documentation at <https://help.ubuntu.com/community/DKMS>
- DKMS Ubuntu manual pages at <http://manpages.ubuntu.com/manpages/lucid/man8/dkms.8.html>
- Linux Journal article on DKMS at <http://www.linuxjournal.com/article/6896>

## Backup and Restore Contrail Configuration Database

### IN THIS SECTION

- Backup config database | 494
- Restore config database | 497

This document provides information on how to backup and restore the Contrail configuration databases –Cassandra and Zookeeper, for Contrail Networking deployed with Canonical Openstack through Juju Charms.

The backup and restore procedure must be completed for the nodes running the same Contrail Networking release. The procedure is used to backup the Contrail Networking databases only; it does not include instructions for backing up orchestration system databases.



**CAUTION:** Database backups must be consistent across all systems because the state of the Contrail database is associated with other system databases, such as OpenStack databases. Database changes associated with northbound APIs must be stopped on all the systems before performing any backup operation. For example, you might block the external VIP for northbound APIs at the load balancer level, such as HAProxy.

The following procedure was tested with Juju version 2.7 and version 2.3.7 running on Ubuntu 16.04 LTS (Xenial Xerus).

Additionally, the procedure contains an example with Juju machine numbers—1, 2 and 3. You must replace it with your Juju machine numbers.

You can identify your Juju machine numbers by running the following command on the host:

```
juju status contrail-controller | grep "^contrail-controller\/" | awk '{print $4}'
```

## Backup config database

Follow the procedure to backup config database:

All the commands are run on the host where Juju client is installed, unless stated otherwise.

**NOTE:** `db_manage.py` script is a disaster recovery script. If any errors occur after running this script, contact Juniper Networks support.

### 1. Update `db_manage.py` script.

```
for i in juju status contrail-controller | grep "^contrail-controller\/" | awk '{print $4}'; do juju ssh $i sudo docker exec contrail-controller curl -k https://raw.githubusercontent.com/tungstenfabric/tf-controller/master/src/config/api-server/vnc_cfg_api_server/db_manage.py --output /tmp/db_manage.py; done
```

### 2. Update `db_json_exim.py` script.

```
for i in juju status contrail-controller | grep "^contrail-controller\/" | awk '{print $4}'; do juju ssh $i sudo docker exec contrail-controller curl -k https://raw.githubusercontent.com/tungstenfabric/tf-controller/master/src/config/common/cfgm_common/db_json_exim.py --output /tmp/db_json_exim.py; done
```

Latest versions of `db_json_exim.py` script requires python future library.

```
for i in `juju status contrail-controller | grep "^contrail-controller\/" | awk '{print $4}'`; do juju ssh $i sudo docker exec contrail-controller pip install future; done
```

3. Stop Juju agents for contrail-controller application.

```
for i in `juju status contrail-controller | grep '^contrail-controller\/' | awk '{print $1}' | sed -e 's/^contrail-controller\/' | sed -e s\/\*\/'; do juju ssh contrail-controller/$i sudo systemctl stop jujud-unit-contrail-controller-$i; done
```

On each controller node, run `juju-status` command to confirm that agents are in the *lost* state.

```
$ juju status contrail-controller
```

4. Stop Contrail config services on all the nodes.

```
for i in contrail-svc-monitor contrail-dns contrail-device-manager contrail-schema contrail-api contrail-control; do for j in 1 2 3; do juju ssh $j sudo docker exec contrail-controller systemctl stop $i; done; done
```

5. Verify status for contrail-controller node. It must be in the *inactive* state.

```
for i in `juju status contrail-controller | grep "^contrail-controller\/" | awk '{print $4}'`; do juju ssh $i sudo docker exec contrail-controller contrail-status; done
== Contrail Config ==
contrail-api:           inactive
contrail-schema:       inactive
contrail-svc-monitor:  inactive
contrail-device-manager: inactive
```

6. Check Contrail config DB for consistency on one of the controller nodes.

```
juju ssh 1 sudo docker exec contrail-controller python /tmp/db_manage.py check
```

7. Synchronize the data by running `repair` command on the Contrail config DB.

```
juju ssh 1 sudo docker exec contrail-controller nodetool repair
```

8. Save database status. You may need it later to compare with the post procedure database status.

```
for i in `juju status contrail-controller | grep "^contrail-controller\/" | awk '{print $4}'`; do juju ssh $i sudo docker exec contrail-controller nodetool status; done
```

9. Log in to one of the controller nodes and take backup of Contrail config DB.

You can follow either one of the following methods:

- Take backup by default db\_json\_exim.py script.

```
juju ssh 1 sudo docker exec contrail-controller python /usr/lib/python2.7/dist-packages/cfgm_common/db_json_exim.py --export-to /tmp/db-dump.json
```

- Take backup by db\_json\_exim.py script which you downloaded in the step 2.

```
juju ssh 1 sudo docker exec contrail-controller python /tmp/db_json_exim.py --export-to /tmp/db-dump.json
```

10. Copy the database backup file from the container to the host.

```
juju ssh 1 sudo docker cp contrail-controller:/tmp/db-dump.json
```

11. Restart the Contrail config services on all the controller nodes.

```
for i in contrail-control contrail-svc-monitor contrail-dns contrail-device-manager contrail-schema contrail-api; do for j in 1 2 3; do juju ssh $j sudo docker exec contrail-controller systemctl start $i; done; done
```

On each controller node, run the `contrail-status` command to confirm that services are in the *active* or *backup* state.

```
for i in `juju status | grep -e "^contrail-[a-z]*\/" | awk '{print $1}' | sed -e 's/\.*//'; do echo "----- $i -----"; TMP=`echo $i | sed -e 's/\.*//'; juju ssh $i sudo docker exec ${TMP} contrail-status; done
```



- Restart the Juju agents for contrail-controller application.

```
for i in `juju status contrail-controller | grep '^contrail-controller\/' | awk '{print $1}' | sed -e 's/^contrail-controller\/\///'; do juju ssh contrail-controller/$i sudo systemctl start jujud-unit-contrail-controller-$i; done
```

Run the `juju status` command from a machine where Juju client is configured. Confirm that Juju agents are in the *active* state.

- Verify the db dump json file for logical structure. Make sure it's not empty.

Node *1* contains db dump.

```
juju ssh 1 sudo docker exec contrail-controller cat /tmp/db-dump.json | jq .
```

Verify the db dump file contains the correct configuration for UUIDs and VMs' IP addresses for your environment.

```
juju ssh 1 sudo cat /tmp/db-dump.json | jq . |grep \"ref:virtual_machine:
juju ssh 1 sudo cat /tmp/db-dump.json | jq . |grep __FEW__OF_IPS__
```

**NOTE:** If there are no VMs loaded on the environment, the above commands will not show any output.

## Restore config database

Follow the procedure to restore config database:

- Stop Juju agents for contrail-controller, contrail-analytics and contrail-analyticsdb applications.

```
for i in `juju status contrail-controller | grep '^contrail-controller\/' | awk '{print $1}' | sed -e 's/^contrail-controller\/\///;s\/\*\/'; do juju ssh contrail-controller/$i sudo systemctl stop jujud-unit-contrail-controller-$i jujud-unit-contrail-analytics-$i jujud-unit-contrail-analyticsdb-$i; done
```

- Stop Contrail services on all the controller nodes.

```
for i in contrail-control contrail-svc-monitor contrail-dns contrail-device-manager
contrail-schema contrail-api contrail-config-nodemgr contrail-control-nodemgr contrail-
```

```

database; do for j in 1 2 3; do juju ssh $j sudo docker exec contrail-controller systemctl
stop $i; done; done
for i in contrail-topology contrail-analytics-nodemgr contrail-snmp-collector contrail-
alarm-gen; do for j in 1 2 3; do juju ssh $j sudo docker exec contrail-analytics systemctl
stop $i; done; done
for i in datastax-agent confluent-kafka; do for j in 1 2 3; do juju ssh $j sudo docker exec
contrail-analyticsdb systemctl stop $i; done; done
for i in contrail-query-engine contrail-collector contrail-analytics-api redis-server; do
for j in 1 2 3; do juju ssh $j sudo docker exec contrail-analytics systemctl stop $i; done;
done

```

3. Run the `contrail-status` command on each controller node to confirm that services are in the *inactive* state.

```

for i in `juju status | grep -e "^contrail-[a-z]*\/" | awk '{print $1}' | sed -e 's/
\*\/'`; do echo "----- $i -----"; TMP=`echo $i | sed -e 's/\./\./'`; juju ssh $i sudo
docker exec ${TMP} contrail-status; done

```

4. Take backup of the Zookeeper data directory on all the controllers.

```

for j in 1 2 3; do juju ssh $j sudo docker exec contrail-controller systemctl stop
zookeeper; done
for j in 1 2 3; do juju ssh $j sudo docker exec contrail-controller tar -cvzf /tmp/
backup_configdatabase_config_zookeeper.tgz /var/lib/zookeeper/version-2; done
for j in 1 2 3; do juju ssh $j sudo docker exec contrail-controller systemctl start
zookeeper; done

```

5. Clean the current data from one of the Zookeeper instances using `rmr` command.

```

for i in `juju ssh 1 sudo docker exec contrail-controller /usr/share/zookeeper/bin/zkCli.sh
ls / | grep "^[" | sed -e 's/\[//;s/\]//;s/,//g;s/\r//'`; do juju ssh 1 sudo docker exec
contrail-controller /usr/share/zookeeper/bin/zkCli.sh rmr /$i; done

```

6. Stop Zookeeper services on all the controllers.

```

for j in 1 2 3; do juju ssh $j sudo docker exec contrail-controller systemctl stop
zookeeper; done

```

- Clean the Zookeeper data directory contents from all the controllers.

```
for j in 1 2 3; do juju ssh $j sudo docker exec contrail-controller "sh -c 'rm -
rvf /var/lib/zookeeper/version-2/*'"; done
```

- Backup the Cassandra data directory from all the controllers.

```
for j in 1 2 3; do juju ssh $j sudo docker exec contrail-controller tar -cvzf /tmp/
backup_configdatabase_config_cassandra.tgz /var/lib/cassandra; done
```

- Clean the Cassandra data directory contents from all the controllers.

```
for j in 1 2 3; do juju ssh $j sudo docker exec contrail-controller "sh -c 'rm -
rvf /var/lib/cassandra/data/*'"; done
for j in 1 2 3; do juju ssh $j sudo docker exec contrail-controller "sh -c 'rm -
rvf /var/lib/cassandra/commitlog/*'"; done
```

After running the above commands, the old password is erased.

- Modify Cassandra configuration on each controller, one at a time, to reset the password.

Edit the *authenticator* variable in the */etc/cassandra/cassandra.yaml* file.

```
juju ssh <node> sudo docker exec -it contrail-controller vim /etc/cassandra/cassandra.yaml
```

Replace *authenticator: PasswordAuthenticator* with *authenticator: AllowAllAuthenticator*.

- Verify that no old Contrail services like *db \** scripts are running. If you find any old services, kill them.

Run the following command on Contrail nodes outside the docker containers.

```
root:~# ps -fe |grep -i contrail | grep -v docker
root    2305  2287  0 12:41 ?        00:00:11 python /tmp/db_json_exim.py --import-
from /tmp/db-dump.json --api-conf /etc/contrail/contrail-api-dbrestore.conf
root    5168  5150  0 12:43 ?        00:00:11 python /tmp/db_json_exim.py --import-
from /tmp/db-dump.json --api-conf /etc/contrail/contrail-api-dbrestore.conf
root    5871  5854  0 12:43 ?        00:00:11 python /tmp/db_json_exim.py --import-
from /tmp/db-dump.json --api-conf /etc/contrail/contrail-api-dbrestore.conf
root    13528 13511  0 12:47 ?        00:00:11 python /tmp/db_json_exim.py --import-
from /tmp/db-dump.json --api-conf /etc/contrail/contrail-api-dbrestore.conf
root    16826 16807  0 11:58 ?        00:00:13 python /tmp/db_json_exim.py --import-
from /tmp/db-dump.json --api-conf /etc/contrail/contrail-api-dbrestore.conf
```

```

root    29493 29473  0 12:56 ?          00:00:10 python /tmp/db_json_exim.py --import-
from /tmp/db-dump.json --api-conf /etc/contrail/contrail-api-dbrestore.conf
root    30810 30784  0 12:06 ?          00:00:12 python /tmp/db_json_exim.py --import-
from /tmp/db-dump.json --api-conf /etc/contrail/contrail-api-dbrestore.conf --debug --
verbose
root    32675 32658  0 12:07 ?          00:00:12 python /tmp/db_json_exim.py --import-
from /tmp/db-dump.json --api-conf /etc/contrail/contrail-api-dbrestore.conf
root    49265 88219  0 17:48 pts/3      00:00:00 grep --color=auto -i contrail
root    60141 60124  0 12:23 ?          00:00:12 python /tmp/db_json_exim.py --import-
from /tmp/db-dump.json --api-conf /etc/contrail/contrail-api-dbrestore.conf
root    60452 60435  0 12:23 ?          00:00:11 python /tmp/db_json_exim.py --import-
from /tmp/db-dump.json --api-conf /etc/contrail/contrail-api-dbrestore.conf
root    61231 54435  0 15:48 pts/1      00:00:00 less /etc/contrail/contrail-api.conf
root    63507 63489  0 12:25 ?          00:00:11 python /usr/lib/python2.7/dist-packages/
cfgm_common/db_json_exim.py --import-from /tmp/db-dump.json --api-conf /etc/contrail/
contrail-api-dbrestore.conf
root    67126 67109  0 12:27 ?          00:00:11 python /tmp/db_json_exim.py --import-
from /tmp/db-dump.json --api-conf /etc/contrail/contrail-api-dbrestore.conf
root    80449 80431  0 12:35 ?          00:00:12 python /tmp/db_json_exim.py --import-
from /tmp/db-dump.json --api-conf /etc/contrail/contrail-api-dbrestore.conf
root    85457 7860  0 16:54 ?          00:00:00 /bin/sh -c contrail-api
root    85458 85457  0 16:54 ?          00:00:04 /usr/bin/python /usr/bin/contrail-api
root    86585 86567  0 12:38 ?          00:00:11 python /tmp/db_json_exim.py --import-
from /tmp/db-dump.json --api-conf /etc/contrail/contrail-api-dbrestore.conf

```

**12. Restart Contrail-Database and Zookeeper service on all the controllers.**

```

for i in contrail-database zookeeper; do for j in 1 2 3; do juju ssh $j sudo docker exec
contrail-controller systemctl start $i; done; done

```

**13. Verify the status of Zookeeper service.**

```

juju ssh 1 sudo docker exec contrail-controller usr/share/zookeeper/bin/zkCli.sh ls /

```

14. Verify the status of Cassandra service.

```
for i in `juju status contrail-controller | grep "^contrail-controller\/" | awk '{print $4}'`; do juju ssh $i sudo docker exec contrail-controller nodetool status; done
```

```
root@(controller):/# nodetool status
Datacenter: datacenter1
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
-- Address      Load       Tokens      Owns (effective)  Host
ID
UN 100.x.x.x 1.15 MiB 256          100.0%           eeb0f764-xxxx-4ca6-
xxxx-84829624d588 rack1
UN 100.x.x.x 1.15 MiB 256          100.0%           d6cf381c-xxxx-4208-
xxxx-5916f09da6a2 rack1
UN 100.x.x.x 1.15 MiB 256          100.0%           ffee7451-xxxx-4058-
xxxx-5efe9f1286f1 rack1
```

For details on `nodetool status` command, see <https://docs.datastax.com/en/archived/cassandra/3.0/cassandra/tools/toolsStatus.html>.

15. Copy the config DB backup.

```
for j in 1 2 3; do juju ssh $j sudo docker cp contrail-controller:/tmp/
backup_configdatabase_config_zookeeper.tgz .; done
for j in 1 2 3; do juju ssh $j sudo docker cp contrail-controller:/tmp/
backup_configdatabase_config_cassandra.tgz .; done
```

16. Restore config DB.

- a. Prepare temporary **contrail-api.conf** file for db restoration.

```
juju ssh 1 sudo docker exec contrail-controller cp /etc/contrail/contrail-api.conf /tmp/
contrail-api-dbrestore.conf
```

- b. Modify `cassandra_password` and `cassandra_user` in the **contrail-api.conf** file.

```
juju ssh 1 sudo docker exec -it contrail-controller vim /tmp/contrail-api-dbrestore.conf
```

```
[CASSANDRA]
cassandra_password = cassandra
cassandra_user = cassandra
```

c. Import database from **/tmp/db-dump/ db-dump.json** file.

You can follow any one of the following methods:

- Import database by default `db_json_exim.py` script.

```
juju ssh 1 sudo docker exec contrail-controller python /usr/lib/python2.7/dist-packages/cfgm_common/db_json_exim.py --import-from /tmp/db-dump.json --api-conf /tmp/contrail-api-dbrestore.conf
```

- Import database by downloaded `db_json_exim.py` script.

```
juju ssh 1 sudo docker exec contrail-controller python /tmp/db_json_exim.py --import-from /tmp/db-dump.json --api-conf /tmp/contrail-api-dbrestore.conf
```

If any error occurs, repeat the procedure to restore config database starting from step 5.

**17.** Synchronize the Cassandra data between nodes.

```
juju ssh 1 sudo docker exec contrail-controller nodetool status
juju ssh 1 sudo docker exec contrail-controller nodetool repair
```

**18.** Modify Cassandra configuration on each controller, one at a time, to reset the password.

Edit the `authenticator` variable in the `/etc/cassandra/cassandra.yaml` file.

```
juju ssh <node> sudo docker exec -it contrail-controller vim /etc/cassandra/cassandra.yaml
```

```
juju ssh <node> sudo docker exec contrail-controller systemctl restart contrail-database
```

Replace `authenticator: AllowAllAuthenticator` with `authenticator: PasswordAuthenticator`.

19. Create Contrail user on any of the controller nodes.

```

root@(controller):/tmp/taj# cqlsh 100.x.108.1 9041 -u cassandra -p cassandra
Connected to ContrailConfigDB at 100.x.108.1:9041. [cqlsh 5.0.1 | Cassandra 3.11.2 | CQL
spec 3.4.4 | Native protocol v4] Use HELP for help.
cassandra@cqlsh> list roles;
role | super | login | options
-----+-----+-----+-----
cassandra | True | True | {}
(1 rows)
cassandra@cqlsh> CREATE USER IF NOT EXISTS controller WITH PASSWORD
'00108521aaa2410aaa44da5dd5a863a3' AND SUPERUSER = true;
cassandra@cqlsh> list roles;
role | super | login | options
-----+-----+-----+-----
cassandra | True | True | {}
controller | True | True | {}

CASSANDRA_ADDR=`juju ssh contrail-controller/0 sudo docker exec contrail-controller ss -l |
grep 9041 | awk '{print $5}' | sed -e 's:// //'`
CASSANDRA_PASS=`juju ssh contrail-controller/0 sudo docker exec contrail-controller
cat /etc/contrail/contrail-api.conf | grep cassandra_password | sed -e 's/^.*=[ ]*//' | sed -
e 's/\r//g'`
CQL_QUERY="CREATE ROLE controller with SUPERUSER = true AND LOGIN = true and PASSWORD = '$
{CASSANDRA_PASS}';"
DB_QUERY="cqlsh ${CASSANDRA_ADDR} -u cassandra -p cassandra -e \"${CQL_QUERY}\"
juju ssh contrail-controller/0 "sudo docker exec contrail-controller ${DB_QUERY}"

```

20. Verify if Contrail user is available on other controller nodes.

```

CASSANDRA_ADDR=`juju ssh contrail-controller/0 sudo docker exec contrail-controller ss -l |
grep 9041 | awk '{print $5}' | sed -e 's:// //'`
CASSANDRA_PASS=`juju ssh contrail-controller/0 sudo docker exec contrail-controller
cat /etc/contrail/contrail-api.conf | grep cassandra_password | sed -e 's/^.*=[ ]*//' | sed -
e 's/\r//g'`
CQL_QUERY="list roles;"
DB_QUERY="cqlsh ${CASSANDRA_ADDR} -u cassandra -p cassandra -e \"${CQL_QUERY}\"
juju ssh contrail-controller/0 "sudo docker exec contrail-controller ${DB_QUERY}"

```

If you don't see Contrail user created on these nodes, check replication factor for *system\_auth* keyspace on all the controller nodes.

## 21. Check replication factor by one of the following methods:

- Using `nodetool` command.

```
juju ssh 1 sudo docker exec contrail-controller nodetool status
```

The output must show that each node owns 100% of tokens and partitions.

- Querying Cassandra db.

```
CASSANDRA_ADDR=`juju ssh contrail-controller/0 sudo docker exec contrail-controller ss -
l | grep 9041 | awk '{print $5}' | sed -e 's:// //'`
CASSANDRA_PASS=`juju ssh contrail-controller/0 sudo docker exec contrail-controller
cat /etc/contrail/contrail-api.conf | grep cassandra_password | sed -e 's/^.*=[ ]*//' |
sed -e 's/\r//g'`
CQL_QUERY="select * from system_schema.keyspaces;"
DB_QUERY="cqlsh ${CASSANDRA_ADDR} -u controller -p ${CASSANDRA_PASS} -e '${CQL_QUERY}'"
juju ssh contrail-controller/0 "sudo docker exec contrail-controller ${DB_QUERY}"
deployer@infra1:~$ juju ssh contrail-controller/0 "sudo docker exec contrail-controller $
{DB_QUERY}"
```

```
keyspace_name      | durable_writes | replication
-----+-----
+-----+-----
          system_auth |             True | {'class':
'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}
          system_schema |             True | {'class':
'org.apache.cassandra.locator.LocalStrategy'}
          svc_monitor_keyspace |             True | {'class':
'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}
          to_bgp_keyspace |             True | {'class':
'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}
          system_distributed |             True | {'class':
'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}
          system |             True | {'class':
'org.apache.cassandra.locator.LocalStrategy'}
          config_db_uuid |             True | {'class':
'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}
          dm_keyspace |             True | {'class':
'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}
          system_traces |             True | {'class':
'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '2'}
```



```

useragent |          True | {'class':
'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}

(10 rows)

```

The `system_auth` parameter must have *replication\_factor* of 3.

If the *replication\_factor* is not set to 3, run the following commands:

```

CQL_QUERY="ALTER KEYSPACE system_auth WITH replication = {'class':
'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'};"
DB_QUERY="cqlsh ${CASSANDRA_ADDR} -u controller -p ${CASSANDRA_PASS} -e \"${CQL_QUERY}\""
juju ssh contrail-controller/0 "sudo docker exec contrail-controller ${DB_QUERY}"

```

22. Restart Contrail services on all the controller nodes.

```

for i in contrail-svc-monitor contrail-dns contrail-device-manager contrail-schema contrail-
api contrail-config-nodemgr contrail-control-nodemgr contrail-control; do for j in 1 2 3;
do juju ssh $j sudo docker exec contrail-controller systemctl start $i; done; done
for i in contrail-topology contrail-analytics-nodemgr contrail-snmp-collector contrail-
alarm-gen; do for j in 1 2 3; do juju ssh $j sudo docker exec contrail-analytics systemctl
start $i; done; done
for i in datastax-agent confluent-kafka; do for j in 1 2 3; do juju ssh $j sudo docker exec
contrail-analyticsdb systemctl start $i; done; done
for i in contrail-collector contrail-analytics-api redis-server contrail-query-engine; do
for j in 1 2 3; do juju ssh $j sudo docker exec contrail-analytics systemctl start $i;
done; done

```

23. On each controller node, enter the `contrail-status` command to confirm that services are in the *active* or *backup* state.

```

for i in `juju status | grep -e "^contrail-[a-z]*\//" | awk '{print $1}' | sed -e 's/
\*//'; do echo "----- $i -----"; TMP=`echo $i | sed -e 's/\././'; juju ssh $i sudo
docker exec ${TMP} contrail-status; done

```

24. Restart Juju agents for `contrail-controller`, `contrail-analytics` and `contrail-analyticsdb` applications.

```

for i in `juju status contrail-controller | grep '^contrail-controller\//' | awk '{print
$1}' | sed -e 's/^contrail-controller\///'; do juju ssh contrail-controller/$i sudo

```

```
systemctl start jujud-unit-contrail-controller-$i jujud-unit-contrail-analytics-$i jujud-
unit-contrail-analyticsdb-$i; done
```

**25. Check Zookeeper status.**

```
juju ssh 1 sudo docker exec contrail-controller /usr/share/zookeeper/bin/zkCli.sh ls /
```

**26. Check the log files on all the controller nodes for any errors.**

**27. Check the database using db\_manage.py script.**

```
juju ssh 1 sudo docker exec contrail-controller python /tmp/db_manage.py check
root(controller):~# python db_manage.py check
2020-06-15 20:25:29,714 INFO: (v1.31) Checker check_zk_mode_and_node_count: Success
2020-06-15 20:25:30,095 INFO: (v1.31) Checker check_cassandra_keyspace_replication: Success
2020-06-15 20:25:31,025 INFO: (v1.31) Checker check_obj_mandatory_fields: Success
2020-06-15 20:25:32,537 INFO: (v1.31) Checker check_orphan_resources: Success
2020-06-15 20:25:33,963 INFO: (v1.31) Checker check_fq_name_uuid_match: Success
2020-06-15 20:25:33,963 WARNING: Be careful, that check can return false positive errors if
stale FQ names and stale resources were not cleaned before. Run at least commands
'clean_obj_missing_mandatory_fields', 'clean_orphan_resources' and 'clean_stale_fq_names'
before.
2020-06-15 20:25:34,707 INFO: (v1.31) Checker check_duplicate_fq_name: Success
2020-06-15 20:25:34,776 INFO: (v1.31) Checker
check_route_targets_routing_instance_backrefs: Success
2020-06-15 20:25:35,384 INFO: (v1.31) Checker check_subnet_uuid: Success
2020-06-15 20:25:35,830 INFO: (v1.31) Checker check_subnet_addr_alloc: Success
2020-06-15 20:25:36,216 INFO: (v1.31) Checker check_route_targets_id: Success
2020-06-15 20:25:36,261 INFO: (v1.31) Checker check_virtual_networks_id: Success
2020-06-15 20:25:36,315 INFO: (v1.31) Checker check_security_groups_id: Success
```

# 3

PART

## Configuring Contrail

---

[Configuring Virtual Networks | 508](#)

[Example of Deploying a Multi-Tier Web Application Using Contrail | 559](#)

[Configuring Services | 574](#)

[Configuring Service Chaining | 606](#)

[Examples: Configuring Service Chaining | 658](#)

[Adding Physical Network Functions in Service Chains | 701](#)

[Configuring High Availability | 711](#)

[QoS Support in Contrail | 726](#)

[Load Balancers | 743](#)

[Optimizing Contrail | 766](#)

---

# Configuring Virtual Networks

## IN THIS CHAPTER

- Creating Projects in OpenStack for Configuring Tenants in Contrail | 508
- Creating a Virtual Network with Juniper Networks Contrail | 510
- Creating a Virtual Network with OpenStack Contrail | 514
- Creating an Image for a Project in OpenStack Contrail | 516
- Creating a Floating IP Address Pool | 520
- Using Security Groups with Virtual Machines (Instances) | 522
- Security Policy Enhancements | 526
- Support for IPv6 Networks in Contrail | 545
- Configuring EVPN and VXLAN | 549

## Creating Projects in OpenStack for Configuring Tenants in Contrail

In Contrail, a tenant configuration is called a project. A project is created for each set of virtual machines (VMs) and virtual networks (VNs) that are configured as a discrete entity for the tenant.

Projects are created, managed, and edited at the OpenStack **Projects** page.

1. Click the **Admin** tab on the OpenStack dashboard, then click the **Projects** link to access the **Projects** page; see [Figure 91 on page 509](#).



4. In the **Add Project** window, select the **Project Members** tab, and assign users to this project. Designate each user as **admin** or as **Member**.  
As a general rule, one person should be a super user in the **admin** role for all projects and a user with a **Member** role should be used for general configuration purposes.
5. Click **Finish** to create the project.

Refer to OpenStack documentation for more information about creating and managing projects.

## RELATED DOCUMENTATION

*Creating a Virtual Network with Juniper Networks Contrail*

*Creating a Virtual Network with OpenStack Contrail*

[OpenStack documentation](#)

## Creating a Virtual Network with Juniper Networks Contrail

Contrail makes creating a virtual network very easy for a self-service user. You create networks and network policies at the user dashboard, then associate policies with each network. The following procedure shows how to create a virtual network when using Juniper Networks Contrail.

1. You need to create an IP address management (IPAM) for your project for to create a virtual network. Select **Configure > Networking > IP Address Management**, then click the **Create** button.  
The **Add IP Address Management** window appears, see [Figure 93 on page 511](#).

Figure 93: Add IP Address Management

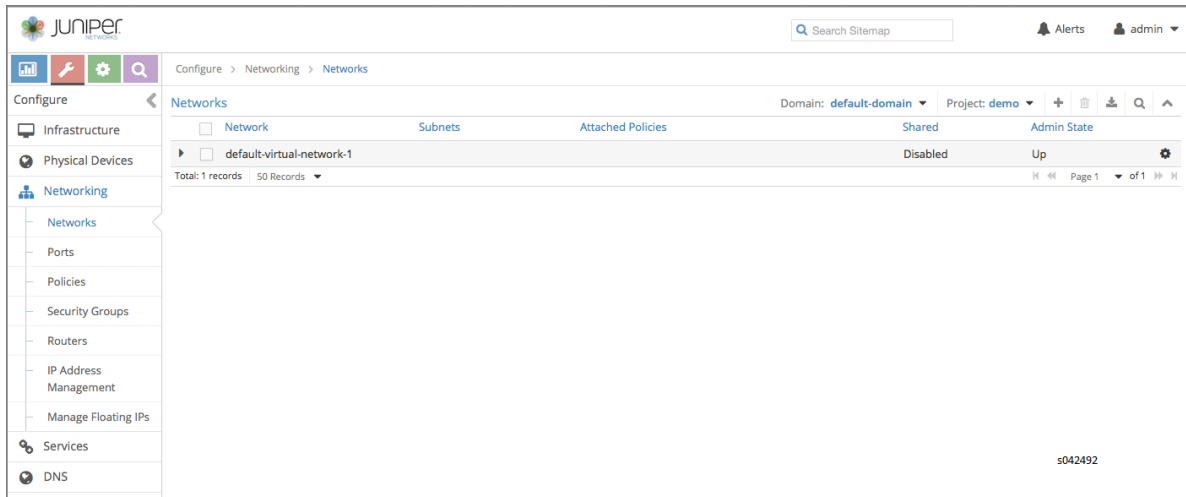
2. Complete the fields in **Add IP Address Management**: The fields are described in [Table 33 on page 511](#).

Table 33: Add IP Address Management Fields

| Field                | Description  |
|----------------------|--|
| <b>Name</b>          | Enter a name for the IPAM you are creating.  |
| <b>DNS Method</b>    | Select from a list the domain name server method for this IPAM: <b>Default, Virtual DNS, Tenant</b> , or <b>None</b> . |
| <b>NTP Server IP</b> | Enter the IP address of an NTP server to be used for this IPAM.  |
| <b>Domain Name</b>   | Enter a domain name to be used for this IPAM.  |

3. Select **Configure > Networking > Networks** to access the **Configure Networks** page; see [Figure 94 on page 512](#).

Figure 94: Configure Networks



4. Verify that your project is displayed as active in the upper-right field, then click the



icon. The **Create Network** window is displayed. See [Figure 95 on page 512](#). Use the scroll bar to access all sections of this window.

Figure 95: Create Network

5. Complete the fields in the **Create Network** window with values that identify the network name, network policy, and IP options as needed. See field descriptions in [Table 34 on page 513](#).



Table 34: Create Network Fields

| Field                    | Description   |
|--------------------------|---|
| <b>Name</b>              | Enter a name for the virtual network you are creating.  |
| <b>Network Policy</b>    | Select the policy to be applied to this network from the list of available policies. You can select more than one policy by clicking each one needed.   |
| <b>Subnets</b>           | Use this area to identify and manage subnets for this virtual network. Click the + icon to open fields for IPAM, CIDR, Allocation Pools, Gateway, DNS, and DHCP. Select the subnet to be added from a drop down list in the IPAM field. Complete the remaining fields as necessary. You can add multiple subnets to a network. When finished, click the + icon to add the selections into the columns below the fields. Alternatively, click the - icon to remove the selections. |
| <b>Host Routes</b>       | Use this area to add or remove host routes for this network. Click the + icon to open fields where you can enter the Route Prefix and the Next Hop. Click the + icon to add the information, or click the - icon to remove the information.   |
| <b>Advanced Options</b>  | Use this area to add or remove advanced options, including identifying the Admin State as Up or Down, to identify the network as Shared or External, to add DNS servers, or to define a VxLAN Identifier.   |
| <b>Floating IP Pools</b> | Use this area to identify and manage the floating IP address pools for this virtual network. Click the + icon to open fields where you can enter the Pool Name and Projects. Click the + icon to add the information, or click the - icon to remove the information.  |
| <b>Route Target</b>      | Move the scroll bar down to access this area, then specify one or more route targets for this virtual network. Click the + icon to open fields where you can enter route target identifiers. Click the + icon to add the information, or click the - icon to remove the information.  |

6. To save your network, click the **Save** button, or click **Cancel** to discard your work and start over.

Now you can create a network policy, see [Creating a Network Policy—Juniper Networks Contrail](#).

## RELATED DOCUMENTATION

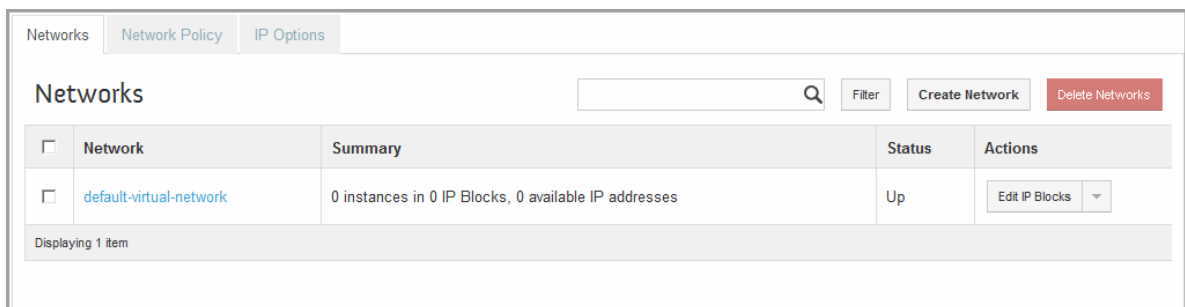
| *Creating an Image for a Project in OpenStack Contrail*

## Creating a Virtual Network with OpenStack Contrail

Contrail makes creating a virtual network very easy for you. You create networks and network policies at the user dashboard, then associate policies with each network. The following procedure shows how to create a virtual network when using OpenStack.

1. To create a virtual network when using OpenStack Contrail, select **Project > Other > Networking**. The **Networks** window is displayed. See [Figure 96 on page 514](#).

Figure 96: Networks Window



2. Verify that the correct project is displayed in the **Current Project** box, then click **Create Network**. The **Create Network** window is displayed. See [Figure 97 on page 514](#) and [Figure 98 on page 515](#).

Figure 97: Create Network Window

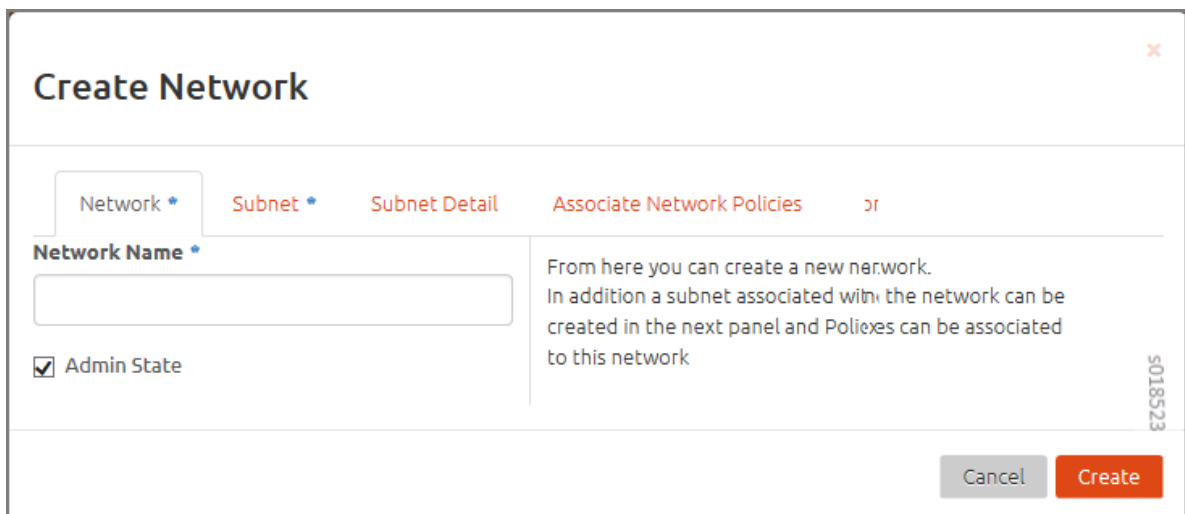


Figure 98: Create Network Window Subnet Tab

3. Click the **Network**, **Subnet**, **Subnet Detail**, and **Associate Network Policies** tabs to complete the fields in the **Create Network** window. See field descriptions in [Table 35 on page 515](#).

Table 35: Create Network Fields

| Field               | Description                      |
|---------------------|----------------------------------|
| <b>Network Name</b> | Enter a name for the network.    |
| <b>Subnet Name</b>  | Enter a name for the subnetwork. |

Table 35: Create Network Fields (Continued)

| Field                  | Description   |
|------------------------|---|
| <b>IPAM</b>            | Select the IPAM associated with the IP block.<br><br>For new projects, an IPAM can be added while creating the virtual network. VM instances created in this virtual network are assigned an address from this address block automatically by the system when a VM is launched. |
| <b>Network Address</b> | Enter the network address in CIDR format.   |
| <b>IP Version*</b>     | Select IPv4 or IPv6.  |
| <b>Gateway IP</b>      | Optionally, enter an explicit gateway IP address for the IP address block. Check the Disable Gateway box if no gateway is to be used.   |
| <b>Network Policy</b>  | Any policies already created are listed. To select a policy, click the check box for the policy.  |

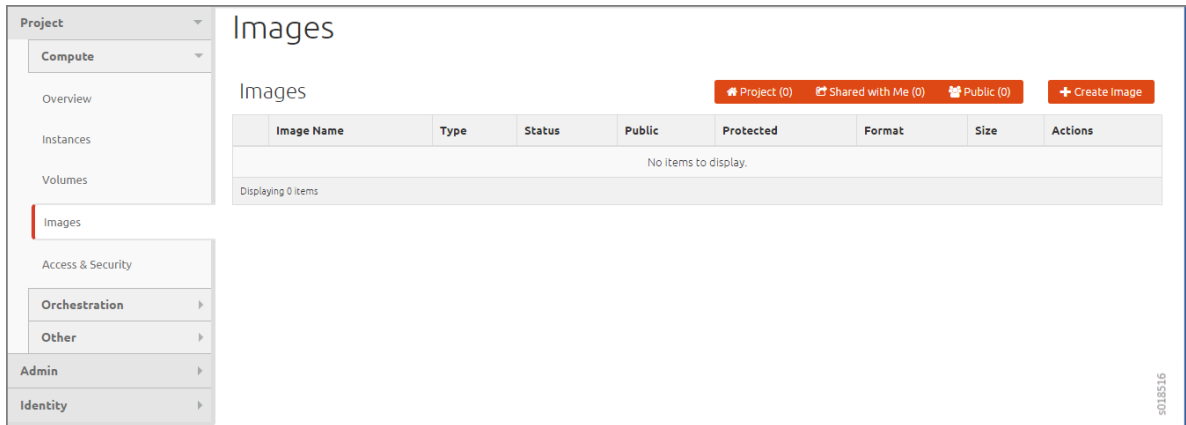
4. Click the **Subnet Details** tab to specify the Allocation Pool, DNS Name Servers, and Host Routes.
5. Click the **Associate Network Policies** tab to associate policies to the network.
6. To save your network, click **Create Network**, or click **Cancel** to discard your work and start over.

## Creating an Image for a Project in OpenStack Contrail

To specify an image to upload to the Image Service for a project in your system by using the OpenStack dashboard:

1. In OpenStack, select **Project > Compute > Images**. The Images window is displayed. See [Figure 99 on page 517](#).

Figure 99: OpenStack Images Window



2. Make sure you have selected the correct project to which you are associating an image.
3. Click **Create Image**.

The **Create An Image** window is displayed. See [Figure 100 on page 518](#).

Figure 100: OpenStack Create An Image Window

## Create An Image ✕

**Name \***

**Description**

**Image Source**

Image Location ▼

**Image Location ?**

**Format \***

Select format ▼

**Architecture**

**Minimum Disk (GB) ?**

**Minimum RAM (MB) ?**

Public

Protected

s018515

Cancel Create Image

4. Complete the fields to specify your image. [Table 36 on page 519](#) describes each of the fields on the window.

**NOTE:** Only images available through an HTTP URL are supported, and the image location must be accessible to the Image Service. Compressed image binaries are supported (\*.zip and \*.tar.gz).

**Table 36: Create an Image Fields**

| Field                 | Description  |
|-----------------------|--|
| <b>Name</b>           | Enter a name for this image.   |
| <b>Description</b>    | Enter a description for the image.   |
| <b>Image Source</b>   | Select <b>Image File</b> or <b>Image Location</b> .<br><br>If you select <b>Image File</b> , you are prompted to browse to the local location of the file.   |
| <b>Image Location</b> | Enter an external HTTP URL from which to load the image. The URL must be a valid and direct URL to the image binary. URLs that redirect or serve error pages result in unusable images.  |
| <b>Format</b>         | Required field. Select the format of the image from a list:<br>AKI- Amazon Kernel Image<br>AMI- Amazon Machine Image<br>ARI- Amazon Ramdisk Image<br>ISO- Optical Disk Image<br>QCOW2- QEMU Emulator<br>Raw- An unstructured image format<br>VDI- Virtual Disk Image<br>VHD- Virtual Hard Disk<br>VMDK- Virtual Machine Disk |
| <b>Architecture</b>   | Enter the architecture.  |

Table 36: Create an Image Fields *(Continued)*

| Field                    | Description  |
|--------------------------|--|
| <b>Minimum Disk (GB)</b> | Enter the minimum disk size required to boot the image. If you do not specify a size, the default is 0 (no minimum). |
| <b>Minimum Ram (MB)</b>  | Enter the minimum RAM required to boot the image. If you do not specify a size, the default is 0 (no minimum).       |
| <b>Public</b>            | Select this check box if this is a public image. Leave unselected for a private image.                               |
| <b>Protected</b>         | Select this check box for a protected image.   |

5. When you are finished, click **Create Image**.

## Creating a Floating IP Address Pool

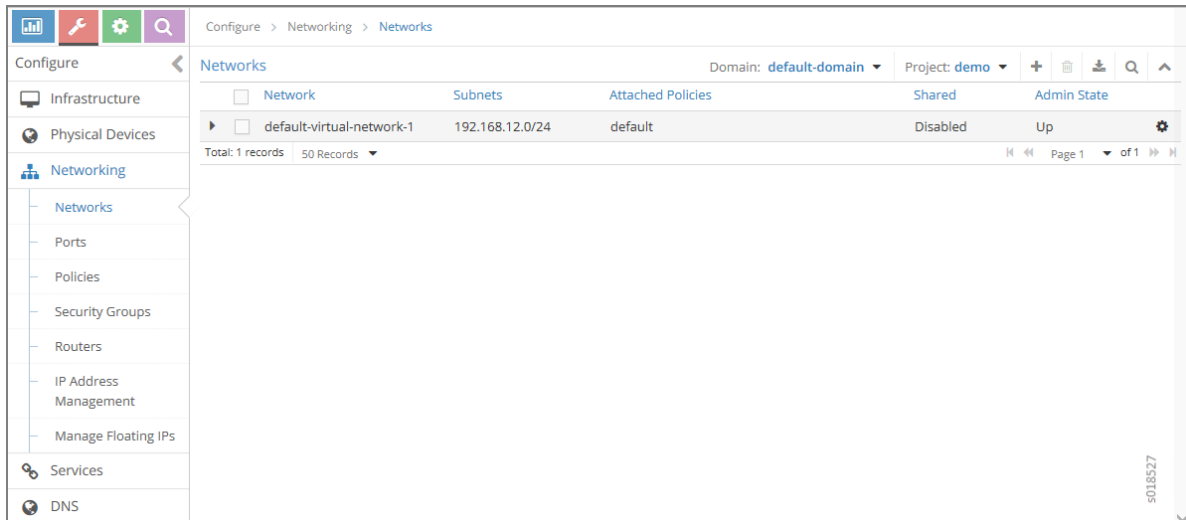
A floating IP address is an IP address (typically public) that can be dynamically assigned to a running virtual instance.

To configure floating IP address pools in project networks in Contrail, then allocate floating IP addresses from the pool to virtual machine instances in other virtual networks:

1. Select **Configure > Networking > Networks**; see [Figure 101 on page 521](#). Make sure your project is the active project in the upper right.



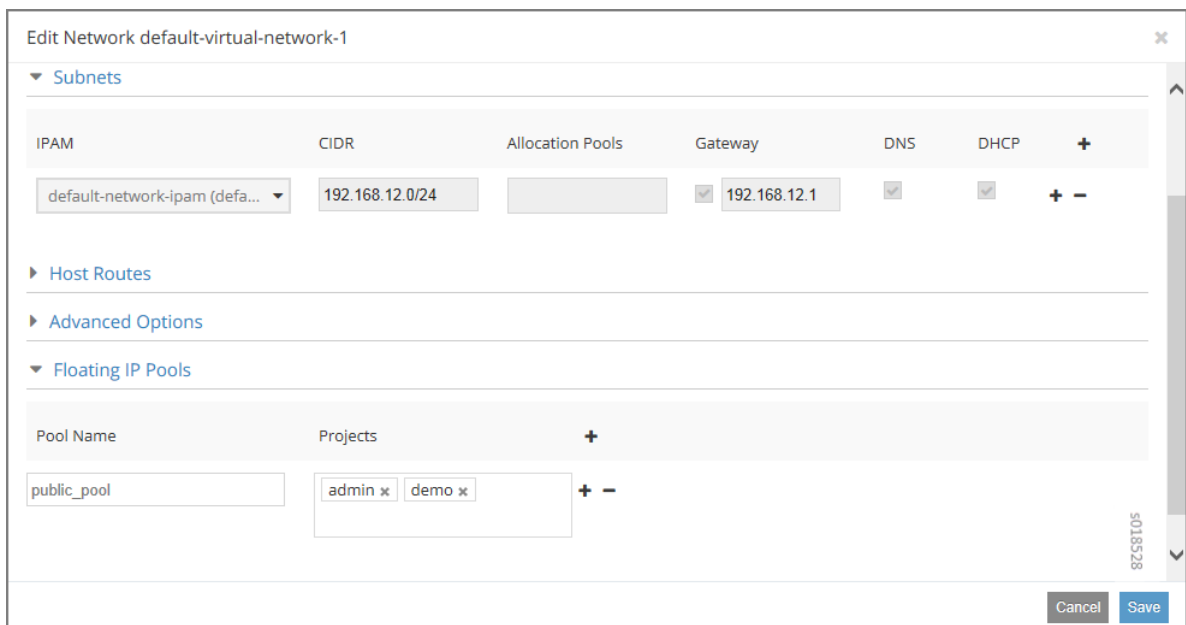
Figure 101: Configure &gt; Networking &gt; Networks



2. Click the network you want to associate with a floating IP pool, then in the **Action** column, click the action icon and select **Edit**.

The **Edit Network** window for the selected network is displayed; see [Figure 102 on page 521](#).

Figure 102: Edit Network



3. In the **Floating IP Pools** section, click the **Pool Name** field, enter a name for your floating IP pool, and click the + (plus sign) to add the IP pool to the table below the field.
  - Multiple floating IP pools can be created at the same time.

- A floating IP pool can be associated with multiple projects.
4. Click **Save** to create the floating IP address pool, or click **Cancel** to remove your work and start over.

## Using Security Groups with Virtual Machines (Instances)

### IN THIS SECTION

- [Security Groups Overview | 522](#)
- [Creating Security Groups and Adding Rules | 522](#)

### Security Groups Overview

A **security group** is a container for security group rules. Security groups and security group rules allow administrators to specify the type of traffic that is allowed to pass through a port. When a virtual machine (VM) is created in a virtual network (VN), a security group can be associated with the VM when it is launched. If a security group is not specified, a port is associated with a default security group. The default security group allows both ingress and egress traffic. Security rules can be added to the default security group to change the traffic behavior.

### Creating Security Groups and Adding Rules

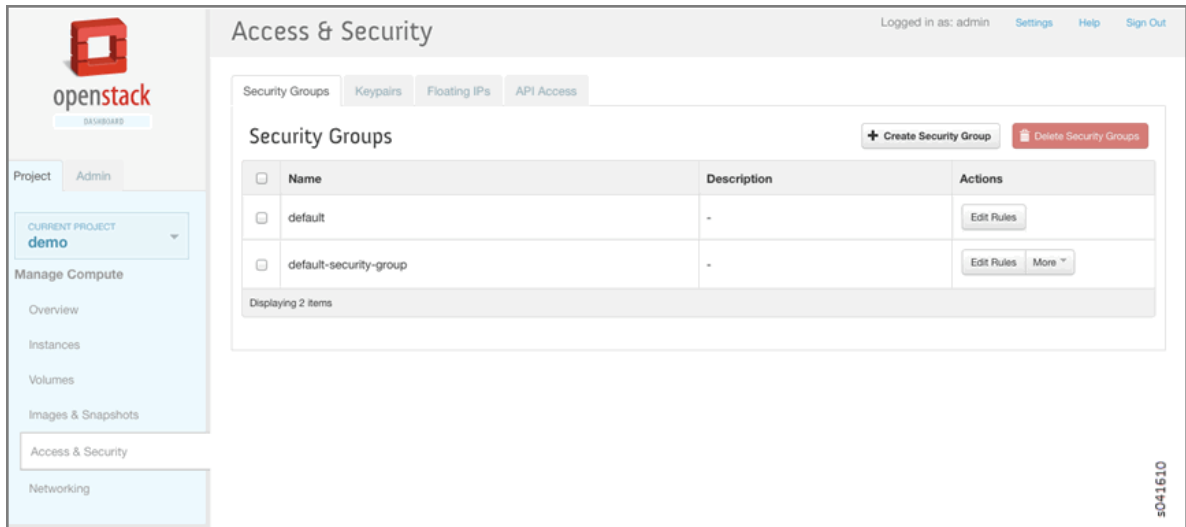
A default security group is created for each project. You can add security rules to the default security group and you can create additional security groups and add rules to them. The security groups are then associated with a VM, when the VM is launched or at a later date.

To add rules to a security group:

1. From the OpenStack interface, click the **Project** tab, select **Access & Security**, and click the **Security Groups** tab.

Any existing security groups are listed under the **Security Groups** tab, including the default security group; see [Figure 103 on page 523](#).

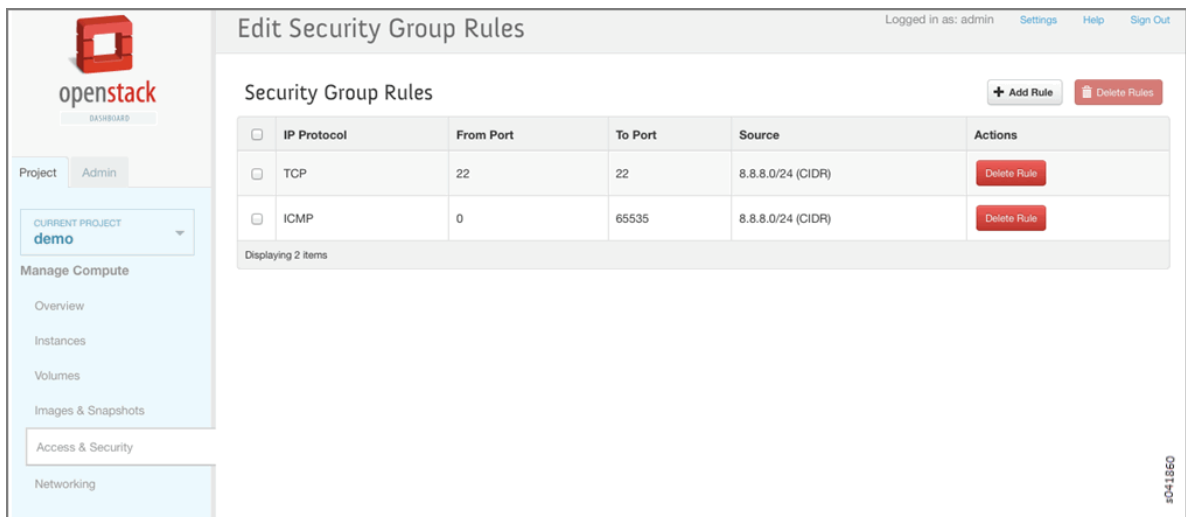
Figure 103: Security Groups



2. Select the **default-security-group** and click **Edit Rules** in the **Actions** column.

The **Edit Security Group Rules** window is displayed; see [Figure 104 on page 523](#). Any rules already associated with the security group are listed.

Figure 104: Edit Security Group Rules



3. Click **Add Rule** to add a new rule; see [Figure 105 on page 524](#).

Figure 105: Add Rule

**Add Rule** [Close]

**IP Protocol**

**Type**

**Code**

**Source**

**Description:**  
 Rules define which traffic is allowed to instances assigned to the security group. A security group rule consists of three main parts:

**Protocol:** You must specify the desired IP protocol to which this rule will apply; the options are TCP, UDP, or ICMP.

**Open Port/Port Range:** For TCP and UDP rules you may choose to open either a single port or a range of ports. Selecting the "Port Range" option will provide you with space to provide both the starting and ending ports for the range. For ICMP rules you instead specify an ICMP type and code in the spaces provided.

**Source:** You must specify the source of the traffic to be allowed via this rule. You may do so either in the form of an IP address block (CIDR) or via a source group (Security Group). Selecting a security group as the source will allow any other instance in that security group access to any other instance via this rule.

[Cancel] [Add]

s041862

Table 37: Add Rule Fields

| Column             | Description  |
|--------------------|--|
| <b>IP Protocol</b> | Select the IP protocol to apply for this rule: TCP, UDP, ICMP.   |
| <b>From Port</b>   | Select the port from which traffic originates to apply this rule. For TCP and UDP, enter a single port or a range of ports. For ICMP rules, enter an ICMP type code. |
| <b>To Port</b>     | The port to which traffic is destined that applies to this rule, using the same options as in the <b>From Port</b> field.  |

Table 37: Add Rule Fields (Continued)

| Column        | Description   |
|---------------|---|
| <b>Source</b> | Select the source of traffic to be allowed by this rule. Specify subnet—the CIDR IP address or address block of the inter-domain source of the traffic that applies to this rule, or you can choose security group as source. Selecting security group as source allows any other instance in that security group access to any other instance via this rule. |

4. Click **Create Security Group** to create additional security groups.

The **Create Security Group** window is displayed; see [Figure 106 on page 525](#).

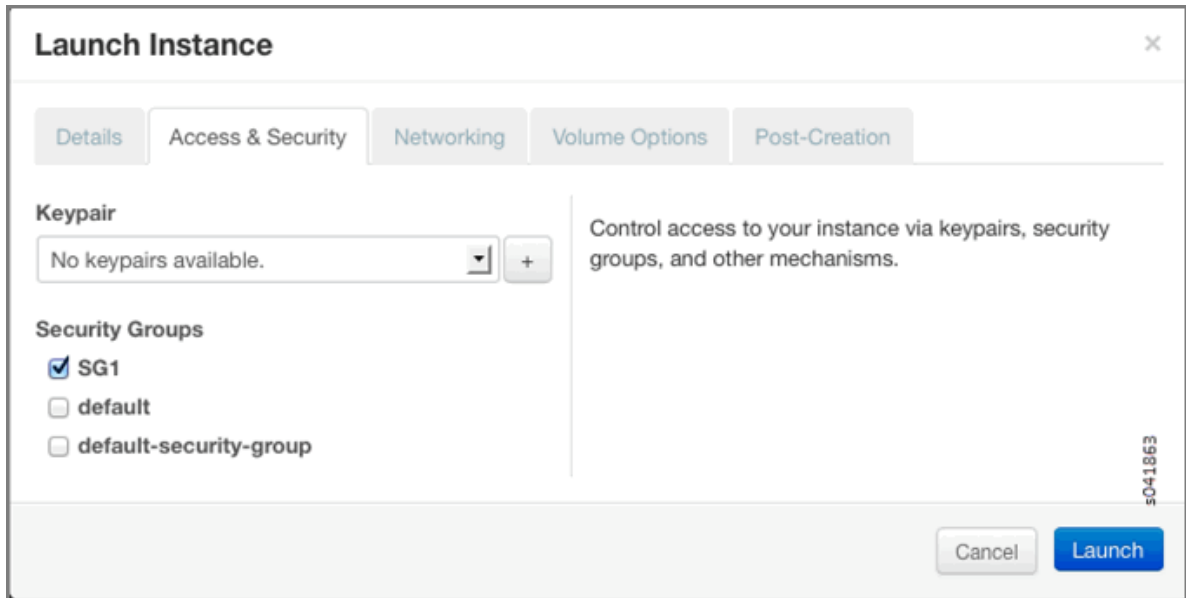
Each new security group has a unique 32-bit security group ID and an ACL is associated with the configured rules.

Figure 106: Create Security Group

5. When an instance is launched, there is an opportunity to associate a security group; see [Figure 107 on page 526](#).

In the **Security Groups** list, select the security group name to associate with the instance.

Figure 107: Associate Security Group at Launch Instance



6. You can verify that security groups are attached by viewing the `SgListReq` and `IntfReq` associated with the `agent.xml`.

## Security Policy Enhancements

### IN THIS SECTION

- [Overview of Existing Network Policy and Security Groups in OpenStack and Contrail | 527](#)
- [Security Policy Enhancements | 527](#)
- [Using Tags and Configuration Objects to Enhance Security Policy | 528](#)
- [Configuration Objects | 529](#)
- [Using the Contrail Web User Interface to Manage Security Policies | 535](#)

## Overview of Existing Network Policy and Security Groups in OpenStack and Contrail

OpenStack tenant networks are isolated, and by default, cannot communicate with other networks. Virtual networks within a tenant require a Neutron router for connectivity, which provides a way to isolate and segment traffic between networks. Each tenant is isolated from other tenants.

Contrail network policy provides security between networks by allowing and denying certain traffic. Contrail network policy also provides connectivity between virtual networks.

OpenStack security groups allow access between workloads and instances for specified traffic types and any other types are denied.

A security policy model for any given customer first needs to map to the OpenStack and Contrail network policy framework and security group constructs.

Customer deployments may contain entities with multiple dimensions, such as multiple deployments, multiple applications, multiple tiers, and so on. A security policy model might contain many ways of cross-cutting those dimensions to control traffic among workloads.

A user might want to segregate traffic on different categories, such as the following examples:

- Site---country, city, rack, or region, or all together country/city/rack or another arbitrary way of dividing place
- OS---a user might want to communicate among same OSs
- Environment---modeling, testing, production
- Application---HR, Salesforce app, oracle ordering app
- Workload type---low sensitivity, financial, or personal identifiable information
- Application Tier---web tier, database tier
- Many more possibilities for needing to segregate traffic.

Additionally, a user might need to cross section between segregation categories, which is hard to express with existing network policy and security group constructs. As a user requires more and different ways to segregate traffic, the number of security groups required explodes.

## Security Policy Enhancements

As the Contrail environment has grown and become more complex, it has become harder to achieve desired security results with the existing network policy and security group constructs. The Contrail network policies have been tied to routing, making it difficult to express security policies for environments such as cross sectioning between categories, or having a multi-tier application supporting development and production environment workloads with no cross environment traffic.

Contrail 4.1 addresses limitations of the current network policy and security group constructs by supporting decoupling of routing from security policies, multidimension segmentation, and policy portability. This release also enhances user visibility and analytics functions for security.

Contrail 4.1 introduces new firewall security policy objects, including the following enhancements:

- Routing and policy decoupling—introducing new firewall policy objects, which decouples policy from routing.
- Multidimension segmentation—segment traffic and add security features, based on multiple dimensions of entities, such as application, tier, deployment, site, usergroup, and so on.
- Policy portability—security policies can be ported to different environments, such as ‘from development to production’, ‘from pci-complaint to production’, ‘to bare metal environment’ and ‘to container environment’.
- Visibility and analytics

## Using Tags and Configuration Objects to Enhance Security Policy

In Contrail 4.1, tags and configuration objects are used to create new firewall policy objects that decouple routing and network policies, enabling multidimension segmentation and policy portability.

Multidimension traffic segmentation helps you segment traffic based on dimensions such as application, tier, deployment, site, and usergroup.

You can also port security policies to different environments. Portability of policies are enabled by providing match conditions for tags. Match tags must be added to the policy rule to match tag values of source and destination workloads without mentioning tag values. For example, in order for the ‘allow protocol tcp source application-tier=web destination application-tier=application match application and site’ rule to take effect, the application and site values must match.

**NOTE:** Contrail supports up to 32 tag types.

### Predefined Tags

You can choose predefined tags based on the environment and deployment requirements.

Predefined tags include:

- application
- application-tier
- deployment



- site
- label (a special tag that allows the user to label objects)

### Example Tag Usage

application = HRApp application-tier = Web site = USA

### Implicit Tags

You can add implicit tags to an existing environment by configuring the tags manually or by adding them as part of provisioning.

Implicit tags include:

- compute node
- rack
- pod
- cluster
- dc

### Tagging Objects

A user can tag the objects project, VN, VM, and VMI with tags and values to map their security requirements. Tags follow the hierarchy of project, VN, VM and VMI and are inherited in that order. This gives an option for the user to provide default settings for any tags at any level. Policies can specify their security in terms of tagged endpoints, in addition to expressing in terms of ip prefix, network, and address groups endpoints.

### Policy Application

Policy application is a new object, implemented by means of the application tag. The user can create a list of policies per application to be applied during the flow acceptance evaluation. Introducing global scoped policies and project scoped policies. There are global scoped policies, which can be applied globally for all projects, and project scoped policies, which are applied to specific projects.

### Configuration Objects

The following are the configuration objects for the new security features.

- firewall-policy

- firewall-rule
- policy-management
- application-policy
- service-group
- address-group
- tag
- global-application-policy

### Configuration Object Tag Object

Each configuration object tag object contains:

- tag—one of the defined tag types, stored as string.
- value—a string
- description—a string to describe the tag
- configuration\_id—a 32-bit value: 5 bits for tag types, 27 bits for tag values

Each value entered by the user creates a unique ID that is set in the tag\_id field. The system can have up to 64 million tag values. On average, each tag can have up to 2k values, but there are no restrictions per tag.

Tags and labels can be attached to any object, for example, project, VN, VM, VMI, and policy, and these objects have a tag reference list to support multiple tags.

RBAC controls the users allowed to modify or remove attached tags. Some tags (typically facts) are attached by the system by default or by means of introspection.

### Tag APIs

Tag APIs are used to give RBAC per tag in any object (VMI, VM, Project ...).

- REST: HTTP POST to /set\_tag\_<tag\_type>/<obj\_uuid>
- Python: set\_tag\_<tag\_type> (object\_type, object\_uuid, tag\_value)

Configuration also supports the following APIs:

- tag query
- tags (policy)

- tags (application tag)
- object query
- tags (object)
- tags (type, value)

## Label

Label is special tag type, used to assign labels for objects. All of the tag constructs are valid, except that tag type is 'label'. One difference from other tags is that an object can have any number of labels. All other tag types are restricted to one tag per object.

The following APIs are available for labels.

- REST: HTTP POST to /add\_tag\_label/<obj\_uuid>
- REST: HTTP POST to /delete\_tag\_label/<obj\_uuid>
- Python: add\_tag\_label (object\_type, object\_uuid, tag\_value)
- Python: delete\_tag\_label (object\_type, object\_uuid, tag\_value)

## Local and Global Tags

Tags can be defined globally or locally under a project; tag objects are children of either config-root or a project. An object can be tagged with a tag in its project or with a globally-scoped tag.

## Analytics

When given a tag query with a SQL where clause and select clause, analytics should give out objects. The query can also contain labels, and the labels can have different operators.

Example:

User might want to know: a list of VMIs where 'site == USA and deployment == Production'

list of VMIs where 'site == USA and deployment == Production has '

Given tag SQL where clause and select clause, analytics should give out flows.

## Control Node

The control node passes the tags, along with route updates, to agents and other control nodes.

## Agent

Agent gets attached tags along with configuration objects. Agent also gets route updates containing tags associated with IP route. This process is similar to getting security group IDs along with the route update.

## Address-Group Configuration Object

There are multiple ways to add IP address to address-group.

- Manually add IP prefixes to the address-group by means of configuration.
- Label a work load with the address-group's specified label. All ports that are labelled with the same label are considered to be part of that address-group.
- Use introspect workloads, based on certain criteria, to add ip-address to address-group.

## Configuration

The address-group object refers to a label object, description, and list of IP prefixes. The label - object is created using the tag APIs.

## Agent

Agent gets address-group and label objects referenced in policy configuration. Agent uses this address group for matching policy rules.

## Analytics

When given address group label, analytics gets all the objects associated with it. Given address group label, get all the flows associated with it.

## Service-Group Configuration Object

### Configuration

The service-group contains a list of ports and protocols. The open stack service-group has a list of service objects; the service object contains attributes: id, name, service group id, protocol, source\_port, destination\_port, icmp\_code, icmp\_type, timeout, tenant id.

### Agent

Agent gets service-group object as it is referred to in a policy rule. Agent uses this service group during policy evaluation.

## Application-policy-set Configuration Object

The application-policy-set configuration object can refer to a tag of type application, network-policy objects, and firewall-policy objects. This object can be local (project) or globally scoped.

When an application tag is attached to an application-policy-set object, the policies referred by that object are automatically applied to the ports that have the same application tag.

Any firewall-policies referred by the application-policy-set objects are ordered using sequence numbers. If the same application tag is attached to multiple application-policy-sets, all those sets will apply, but order among those sets is undefined.

One application-policy-set (called default-policy-application-set) is special in that policies referred by it are applied to all interfaces by default, after applying policies referred to other application-policy-sets.

Upon seeing the application tag for any object, the associated policies are sent to agent. Agent will use this information to find out the list of policies to be applied and their sequence during flow evaluation. User can attach application tag to allowed objects (Project, VN, VM or VMI).

## Policy-management Configuration Object

Policy-management is a global container object for all policy-related configuration.

Policy-management object contains

- network-policies (NPs)
- firewall-policies (FWPs)
- application-policy-sets
- global-policy objects
- global-policy-apply objects
- NPs - List of contrail networking policy objects
- FWPs - List of new firewall policy objects
- Application-policies - List of Application-policy objects
- Global-policies - List of new firewall policy objects, that are defined for global access
- Global-policy-apply - List of global policies in a sequence, and these policies applied during flow evaluation.
- Network Policies (NP) references are available, as they are today.

## Firewall-policy Configuration Object

Firewall-policy is a new policy object that contains a list of firewall-rule-objects and audited flag. Firewall-policy can be project or global scoped depending on usage. Includes an audited Boolean flag to indicate that the owner of the policy indicated that the policy is audited. Default is False, and will have to explicitly be set to True after review. Generates a log event for audited with timestamp and user details.

## Firewall-rule Configuration Object

Firewall-rule is a new rule object, which contains the following fields. The syntax is to give information about their layout inside the rule.

- <sequence number>  
There is a string object sequence number on the link from firewall-policy to firewall-policy-rule objects. The sequence number decides the order in which the rules are applied.
- [< id >]  
uuid
- [name < name >]  
Unique name selected by user
- [description < description >]
- public
- {permit | deny}
- [ protocol {< protocol-name > | any } destination-port { < port range > | any } [ source-port { < port range > | any} ] ] | service-group < name >
- endpoint-1 { [ip < prefix > ] | [virtual-network < vnname > ] | [address-group < group name > ] | [tags T1 == V1 && T2 == V2 ... && Tn == Vn && label == label name...] | any }
- { -> | <- | <-> }  
Specifies connection direction. All the rules are connection oriented and this option gives the direction of the connection.
- endpoint-2 { [ip < prefix > ] | [virtual-network < vnname > ] | [address-group < group name > ] | [tags T1 == V1 && T2 == V2 ... && Tn == Vn && label == label name...] | any }

Tags at endpoints support an expression of tags. We support only '=' and '&&' operators. User can specify labels also as part the expression. Configuration object contains list of tag names (or global:tag-name in case of global tags) for endpoints.

- [ match\_tags {T1 .... Tn} | none ]

List of tag types or none. User can specify either match with list of tags or none. Match with list of tags mean, source and destination tag values should match for the rule to take effect.

- [ timer < start-time > < limit > ]
- [ log | mirror | alert | activate | drop | reject | sdrop ]

complex actions

- { enable | disable }

A boolean flag to indicate the rule is enabled or disabled. Facilitates selectively turn off the rules, without remove the rule from the policy. Default is True.

- filter

## Compilation of Rules

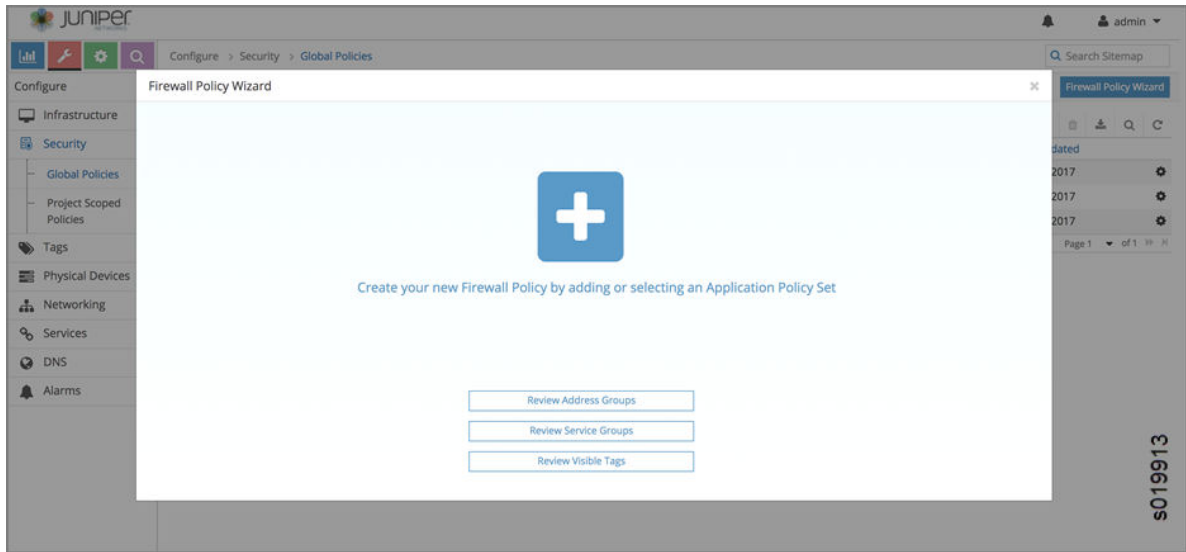
Whenever the API server receives a request to create/update a firewall policy rule object, it analyzes the object data to make sure that all virtual-networks, address-group, tag objects exist. If any of them do not exist, the request will be rejected. In addition, it will actually create a reference to those objects mentioned in the two endpoints. This achieves two purposes. First, we don't allow users to name non-existent objects in the rule and second, the user is not allowed to delete those objects without first removing them from all rules that are referring to them.

## Using the Contrail Web User Interface to Manage Security Policies

### Adding Security Policies

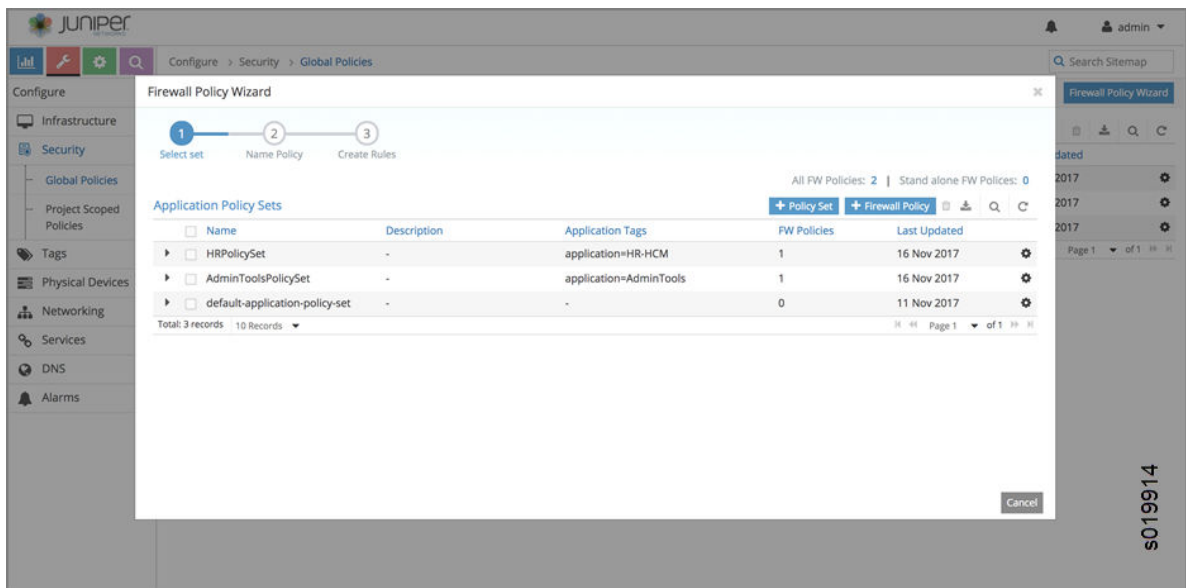
1. To add a security policy, go to **Configure > Security > Global Policies**. Near the upper right, click the button **Firewall Policy Wizard**. The **Firewall Policy Wizard** appears, where you can create your new firewall policy by adding or selecting an application policy set. See [Figure 108 on page 536](#).

Figure 108: Firewall Policy Wizard



2. Click the large + on the Firewall Policy Wizard screen to view the **Application Policy Sets** window. The existing application policy sets are displayed. See [Figure 109 on page 536](#).

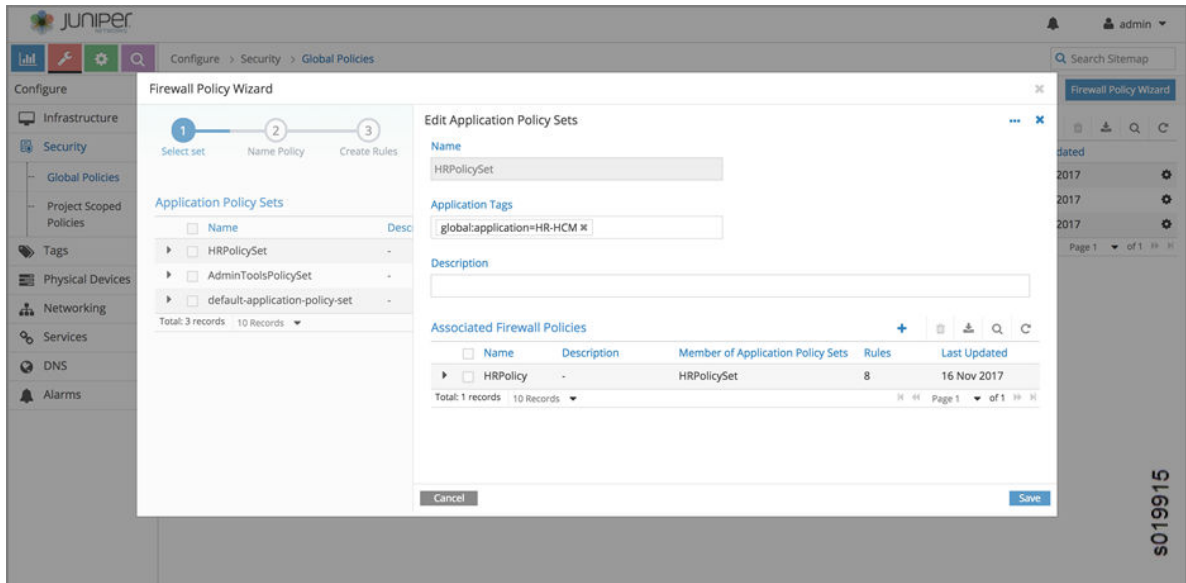
Figure 109: Application Policy Sets



3. To create a new firewall policy, click the application policy set in the list to which the new firewall policy will belong. The **Edit Application Policy Sets** window appears, displaying a field for the description of the selected policy set and listing firewall policies associated with the set. See [Figure 110 on page 537](#), where the **HRPolicySet** has been selected.



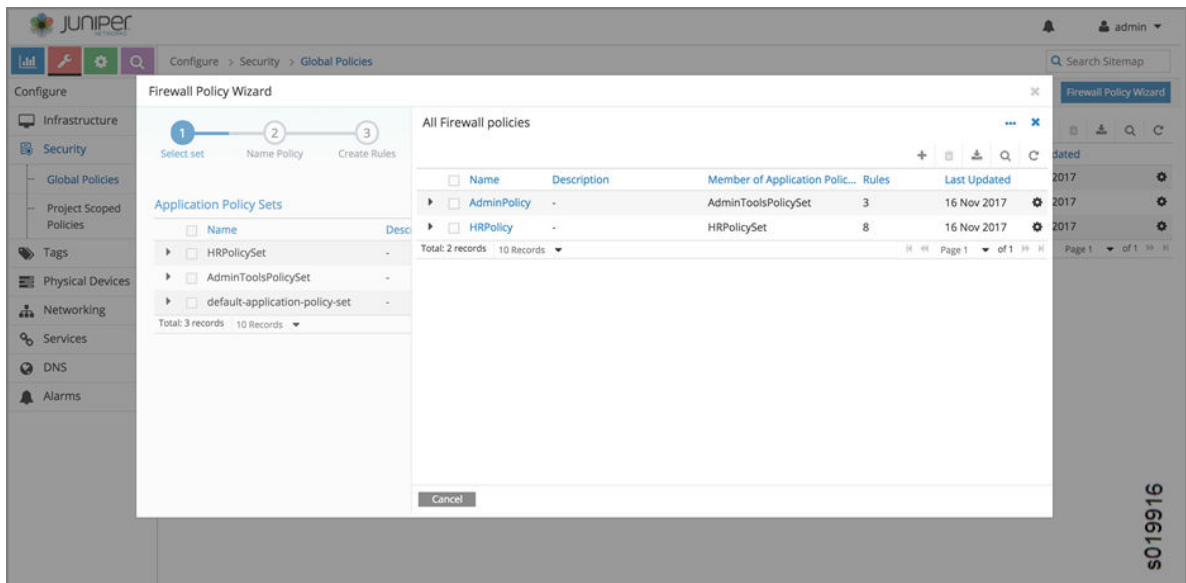
Figure 110: Edit Application Policy Sets



4. To view all firewall policies, click the Application Policy Sets link in the left side.

See [Figure 111 on page 537](#).

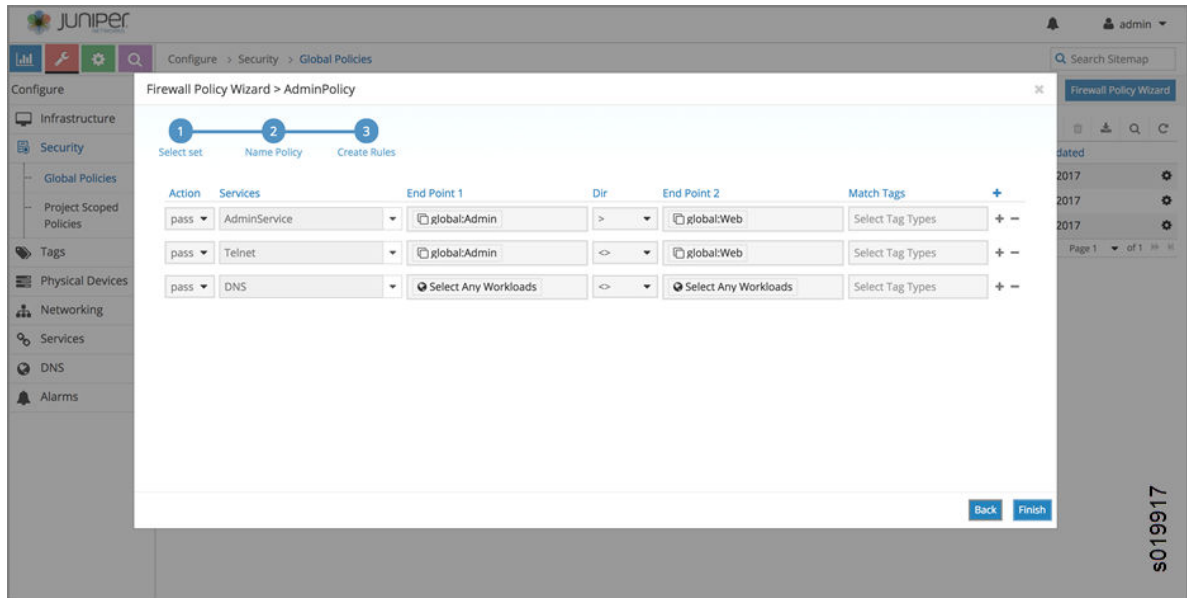
Figure 111: All Firewall Policies



5. Select any listed firewall policy to view or edit the rules associated with that policy. See [Figure 112 on page 538](#), where all the rules for the **AdminPolicy** are listed. Use the dropdown menus in each

field to add or change policy rules, and use the +, - icons to the right of each rule to add or delete the rule.

**Figure 112: Firewall Policy Rules**

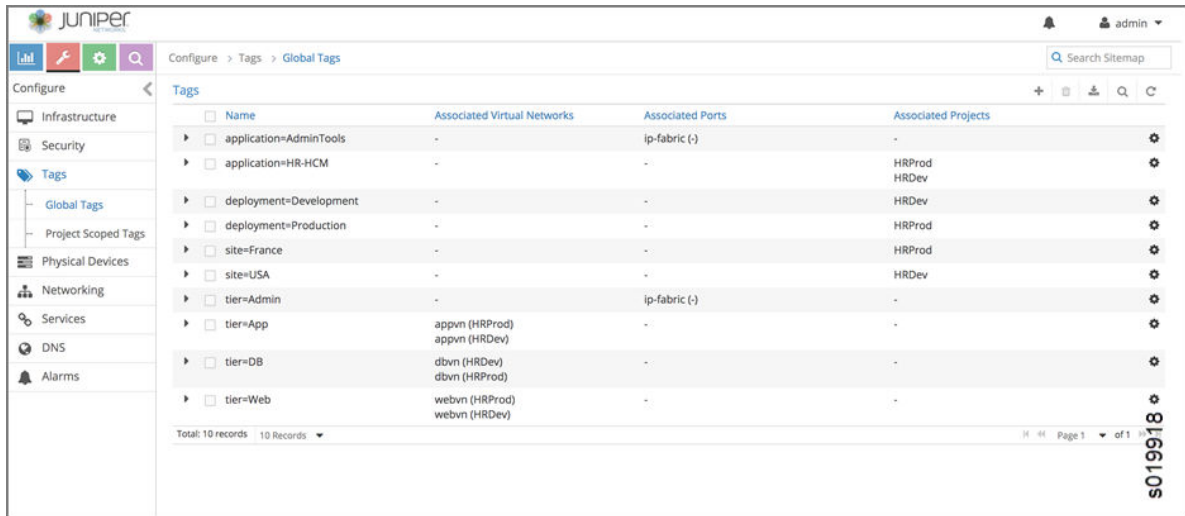


## Managing Policy Tags

You can use the Contrail web user interface to create and manage the tags used to provide granularity to security policies. You can have global tags, applicable to the entire system, or project tags, defined for specific uses in specific projects.

1. To manage policy tags, go to **Configure > Tags > Global Tags**. The **Tags** window appears, listing all of the tags in use in the system, with the associated virtual networks, ports, and projects for each tag. Tags are defined first by type, such as application, deployment, site, tier, and so on. See [Figure 113 on page 539](#).

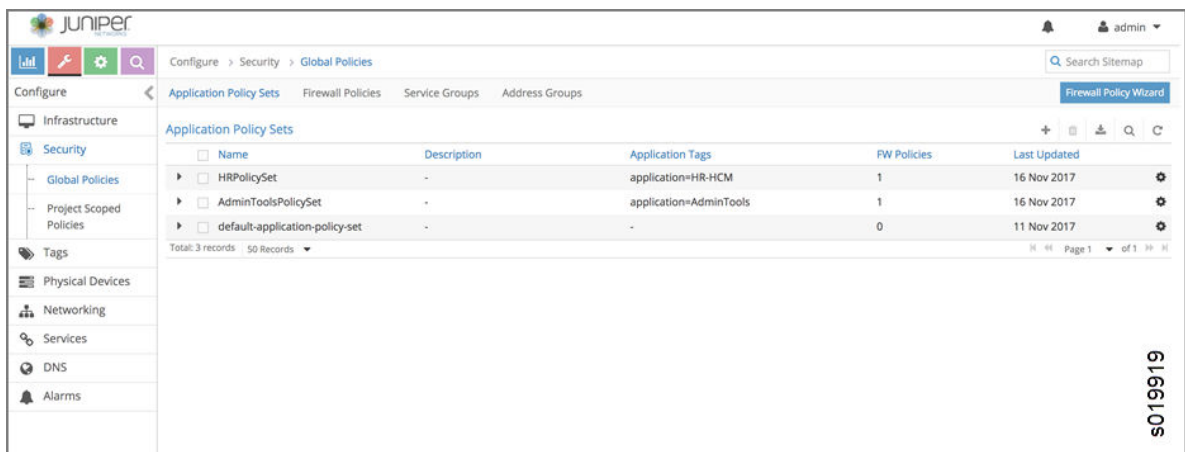
Figure 113: Tags



| Name                   | Associated Virtual Networks     | Associated Ports | Associated Projects |
|------------------------|---------------------------------|------------------|---------------------|
| application=AdminTools | -                               | ip-fabric (-)    | -                   |
| application=HR-HCM     | -                               | -                | HRProd<br>HRDev     |
| deployment=Development | -                               | -                | HRDev               |
| deployment=Production  | -                               | -                | HRProd              |
| site=France            | -                               | -                | HRProd              |
| site=USA               | -                               | -                | HRDev               |
| tier=Admin             | -                               | ip-fabric (-)    | -                   |
| tier=App               | appvn (HRProd)<br>appvn (HRDev) | -                | -                   |
| tier=DB                | dbvn (HRDev)<br>dbvn (HRProd)   | -                | -                   |
| tier=Web               | webvn (HRProd)<br>webvn (HRDev) | -                | -                   |

2. You can click through any listed tag to see the rules to which the tag is applied. See [Figure 114 on page 539](#), which shows the application tags that are applied to the current application sets. You can also reach this page from **Configure > Security > Global Policies**.

Figure 114: View Application Tags



| Name                           | Description | Application Tags       | FW Policies | Last Updated |
|--------------------------------|-------------|------------------------|-------------|--------------|
| HRPolicySet                    | -           | application=HR-HCM     | 1           | 16 Nov 2017  |
| AdminToolsPolicySet            | -           | application=AdminTools | 1           | 16 Nov 2017  |
| default-application-policy-set | -           | -                      | 0           | 11 Nov 2017  |

## Viewing Global Policies

From **Configure > Security > Global Policies**, in addition to viewing the policies included in application policy sets, you can also view all firewall policies, all service groups policies, and all address groups policies.

1. To view and manage the global firewall policies, from **Configure > Security > Global Policies**, click the **Firewall Policies** tab to view the details for system firewall policies, see [Figure 115 on page 540](#)

**Figure 115: Firewall Policies**

| Name        | Description | Member of Application Policy Sets | Rules | Last Updated |
|-------------|-------------|-----------------------------------|-------|--------------|
| AdminPolicy | -           | AdminToolsPolicySet               | 3     | 16 Nov 2017  |
| HRPolicy    | -           | HRPolicySet                       | 8     | 16 Nov 2017  |

Total: 2 records 50 Records

2. To view and manage the service groups policies, from **Configure > Security > Global Policies**, click the **Service Groups** tab to view the details for system policies for service groups, see [Figure 116 on page 540](#).

**Figure 116: Service Groups**

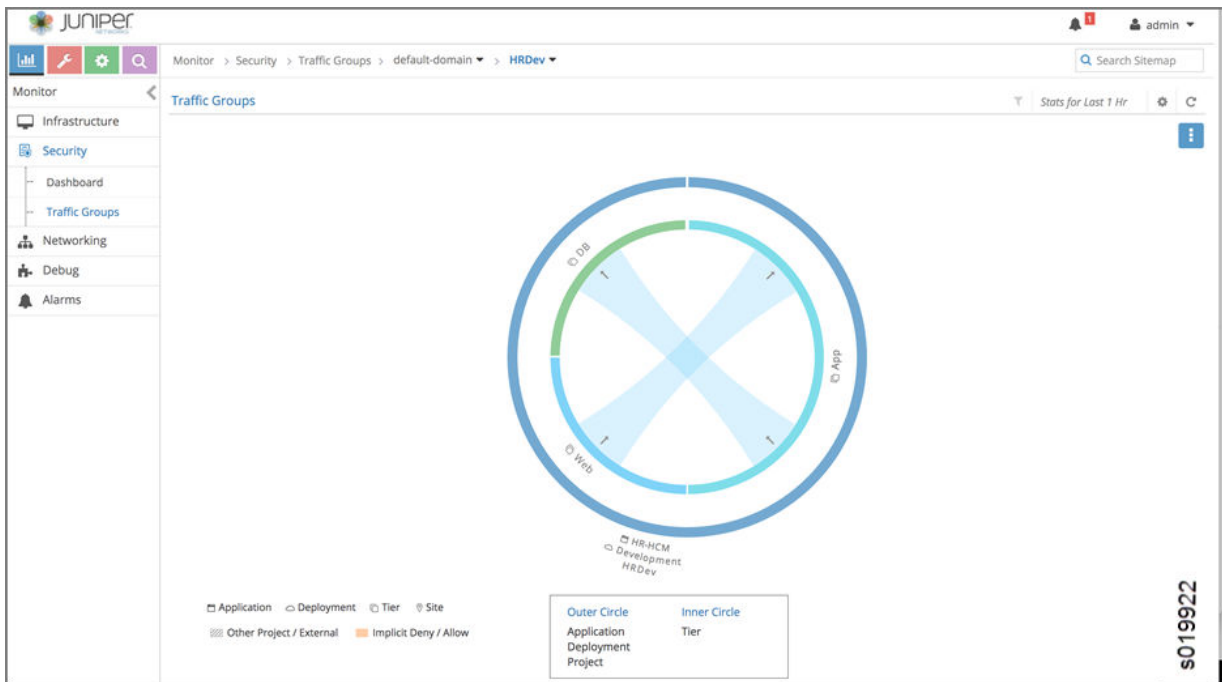
| Name              | Protocol   | Port |      |
|-------------------|------------|------|------|
| DNS               | TCP        | 53   |      |
|                   | UDP        | 53   |      |
| AdminService      | TCP        | 22   |      |
|                   | TCP        | 8080 |      |
|                   | (6 more)   |      |      |
|                   | WebService | TCP  | 22   |
|                   |            | TCP  | 80   |
|                   | AppService | TCP  | 80   |
|                   |            | TCP  | 8080 |
| (1 more)          |            |      |      |
| TestCommunication | TCP        | 22   |      |
|                   | TCP        | 53   |      |
|                   | (2 more)   |      |      |
| Telnet            | TCP        | 23   |      |
| DBService         | TCP        | 3306 |      |

Total: 7 records 50 Records

## Visualizing Traffic Groups

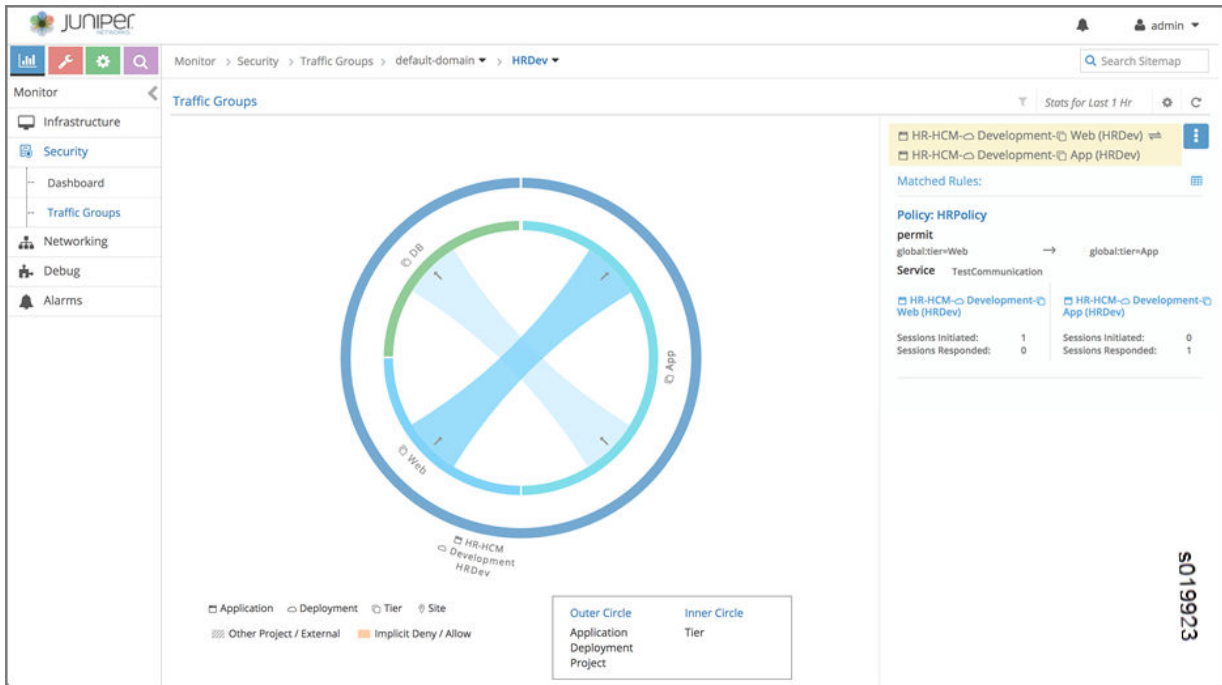
Use **Monitor > Security > Traffic Groups** to explore visual representations of how policies are applied to traffic groups. See [Figure 117 on page 541](#), which is a visual representation of the source and destination traffic for the past one hour of a traffic group named Traffic Groups. The outer circle represents traffic tagged with application, deployment, or project. The inner circle represents traffic tagged with tier. The center of the circle shows the traffic origination and destination.

**Figure 117: Traffic Groups**



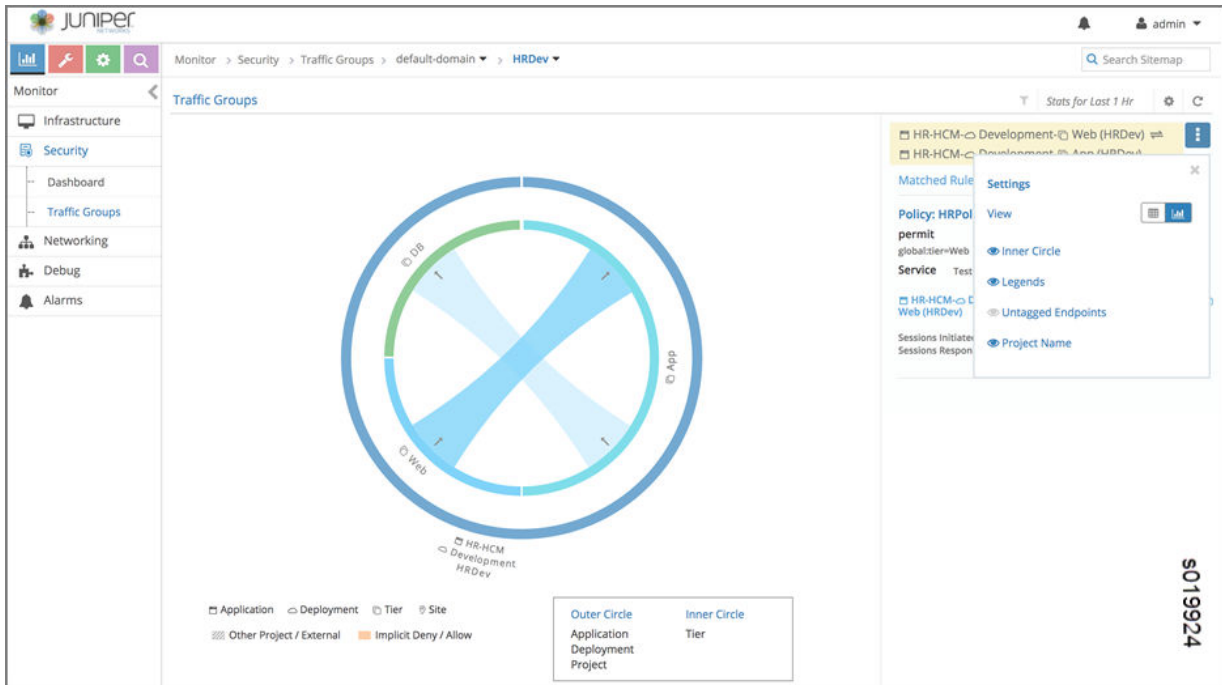
You can click in the right side of the screen to get details of the policy rules that have been matched by the selected traffic. See [Figure 118 on page 542](#).

Figure 118: Traffic Groups, Policy Details



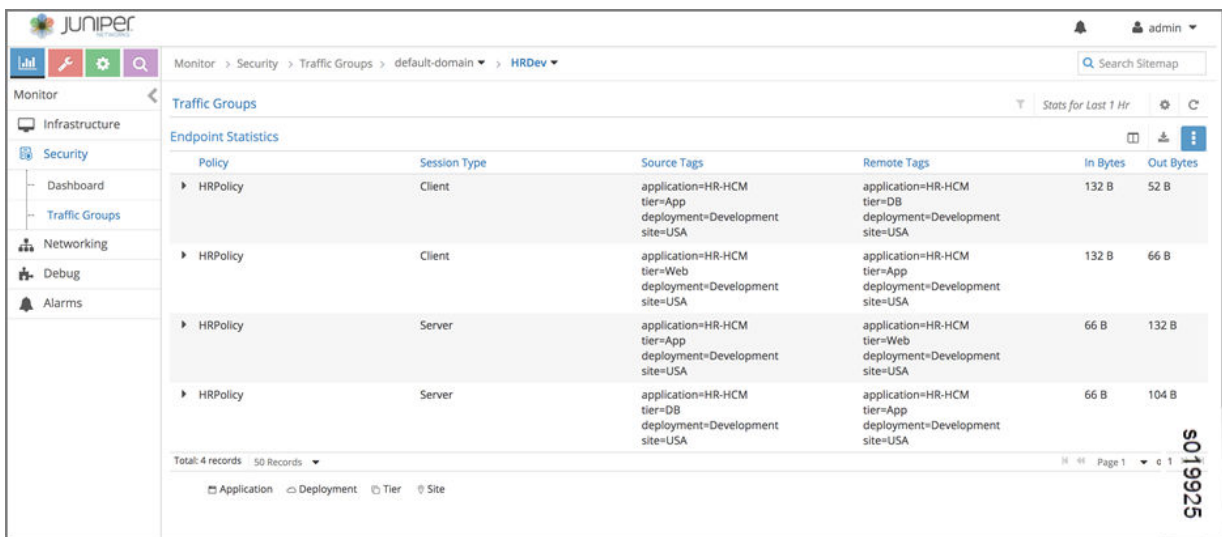
You can click in the right side of the screen to get to the **Settings** window, where you can change the type of view and change which items appear in the visual representation. See [Figure 119 on page 543](#).

Figure 119: Traffic Groups, Settings



You can click on the name of a policy that has been matched to view the endpoint statistics, including source tags and remote tags, of the traffic currently represented in the visual. See [Figure 120 on page 543](#).

Figure 120: Traffic Groups, Endpoint Statistics



You can click deeper through any linked statistic to view more details about that statistic, see [Figure 122 on page 544](#) and [Figure 122 on page 544](#).

**Figure 121: Traffic Groups, Details**

Monitor > Security > Traffic Groups > default-domain > HRDev

Monitor

Infrastructure

Security

Dashboard

Traffic Groups

Networking

Debug

Alarms

Traffic Groups

All

- HR-HCM- Development- App
- HR-HCM- Development- DB

Select Endpoint

- HR-HCM- Development- App
- HR-HCM- Development- DB

All Sessions

| Protocol (Server Port) | Session Type | Bytes (In/Out) |
|------------------------|--------------|----------------|
| TCP (22)               | client       | 132 B / 0 B    |

Total: 1 records 50 Records

Application Deployment Tier Site

Matched Rules:

Policy: HRPolicy

permit global:tier=App → global:tier=DB

Service TestCommunication

Sessions Initiated: 1  
Sessions Responded: 0

Sessions Initiated: 1  
Sessions Responded: 0

s019926

**Figure 122: Traffic Groups, Details**

Monitor > Security > Traffic Groups > default-domain > HRDev

Monitor

Infrastructure

Security

Dashboard

Traffic Groups

Networking

Debug

Alarms

Traffic Groups

All

- HR-HCM- Development- App
- HR-HCM- Development- DB

Protocol: TCP

Port: 22

Client Sessions

| Local IP | VN            | Bytes (In/Out) |
|----------|---------------|----------------|
| 3.3.3.3  | appvn (HRDev) | 132 B / 0 B    |

Total: 1 records 50 Records

Application Deployment Tier Site

Matched Rules:

Policy: HRPolicy

permit global:tier=App → global:tier=DB

Service TestCommunication

Sessions Initiated: 1  
Sessions Responded: 0

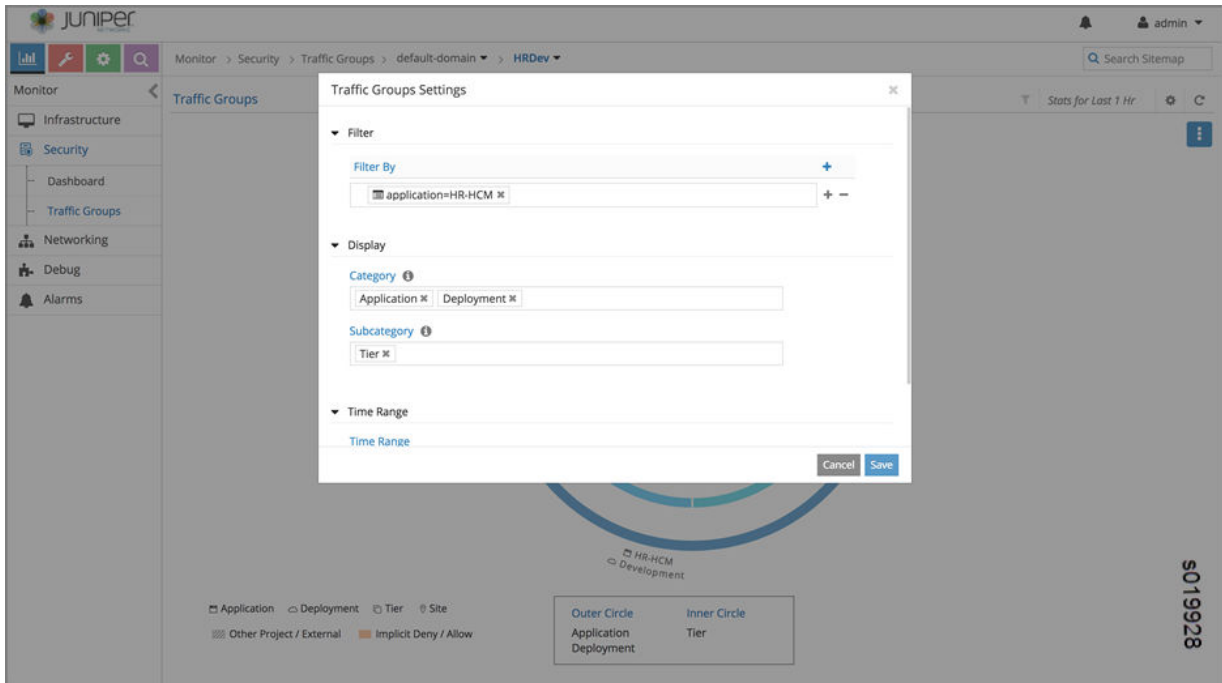
Sessions Initiated: 1  
Sessions Responded: 0

s019927

You can change the settings of what statistics are displayed in each traffic group at the **Traffic Groups Settings** screen see [Figure 123 on page 545](#).



Figure 123: Traffic Groups Settings



## Support for IPv6 Networks in Contrail

### IN THIS SECTION

- [Overview: IPv6 Networks in Contrail | 545](#)
- [Creating IPv6 Virtual Networks in Contrail | 546](#)
- [Adding IPv6 Peers | 548](#)

Starting with Contrail Release 2.0, support for IPv6 overlay networks is provided.

### Overview: IPv6 Networks in Contrail

The following features are supported for IPv6 networks and overlay. The underlay network must be IPv4.

- Virtual machines with IPv6 and IPv4 interfaces

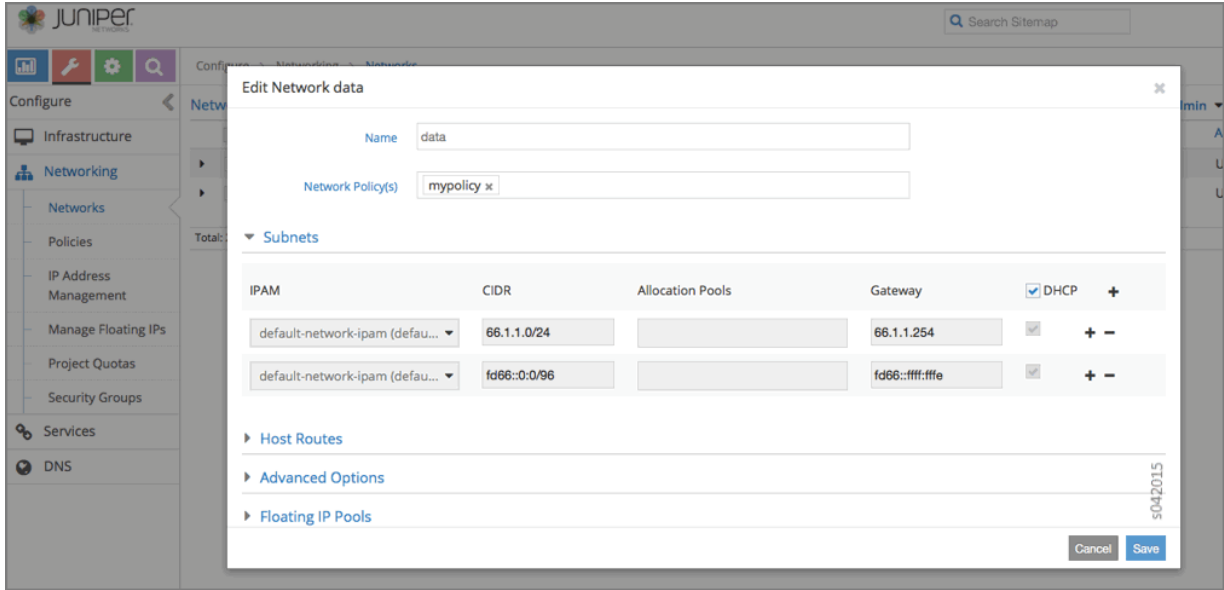
- Virtual machines with IPv6-only interfaces
- DHCPv6 and neighbor discovery
- Policy and Security groups
- IPv6 flow set up, tear down, and aging
- Flow set up and tear down based on TCP state machine
- Protocol-based flow aging
- Fat flow
- Allowed address pair configuration with IPv6 addresses
- IPv6 service chaining
- Equal Cost Multi-Path (ECMP)
- Connectivity with gateway (MX Series device)
- Virtual Domain Name Services (vDNS), name-to-IPv6 address resolution
- User-Visible Entities (UVEs)

*NOT* present is support for the following:

- Source Network Address Translation (SNAT)
- Load Balancing as a Service (LBaaS)
- IPv6 fragmentation
- Floating IP
- Link-local and metadata services
- Diagnostics for IPv6
- Contrail Device Manager
- Virtual customer premises equipment (vCPE)

## Creating IPv6 Virtual Networks in Contrail

You can create an IPv6 virtual network from the Contrail user interface in the same way you create an IPv4 virtual network. When you create a new virtual network by selecting **Configure > Networking > Networks**, the Edit fields accept IPv6 addresses, as shown in the following image.



### Address Assignments

When virtual machines are launched with an IPv6 virtual network created in the Contrail user interface, the virtual machine interfaces get assigned addresses from all the families configured in the virtual network.

The following is a sample of IPv6 instances with address assignments, as listed in the OpenStack Horizon user interface.

| Instance Name                             | Image Name  | IP Address  | Size  | Keypair | Status | Task | Power State | Uptime          | Actions              |
|---|-------------|---|---|---------|--------|------|-------------|-----------------|----------------------|
| Test-6dba4281-ada9-41fc-8609-bcd89f978ee3 | ubuntu-jdof | data<br>66.1.1.251<br>fd66::ffff:fff<br>vn-jdaf<br>76.1.1.252 | m1.medium   4GB RAM   2 VCPU   40.0GB<br>Disk | -       | Active | None | Running     | 4 days, 9 hours | Create Snapshot More |
| Test-7a3b7c5b-e5a5-46b3-9346-29079a1abdba | ubuntu-jdof | data<br>66.1.1.250<br>fd66::ffff:fff<br>vn-jdaf<br>76.1.1.250 | m1.medium   4GB RAM   2 VCPU   40.0GB<br>Disk | -       | Active | None | Running     | 4 days, 9 hours | Create Snapshot More |
| Test-663309b7-1765-4cc4-9edc-f9025ecd4ee6 | ubuntu-jdof | data<br>66.1.1.245<br>fd66::ffff:fff<br>vn-jdaf<br>76.1.1.244 | m1.medium   4GB RAM   2 VCPU   40.0GB<br>Disk | -       | Active | None | Running     | 4 days, 9 hours | Create Snapshot More |
| Test-s20de6d7-3d2b-447e-8894-d794eaa620ab | ubuntu-jdof | data<br>66.1.1.252<br>fd66::ffff:fff<br>vn-jdaf<br>76.1.1.251 | m1.medium   4GB RAM   2 VCPU   40.0GB<br>Disk | -       | Active | None | Running     | 4 days, 9 hours | Create Snapshot More |
| Test-43345608-455f-4766-9346-5c81f5be2197 | ubuntu-jdof | data<br>66.1.1.247<br>fd66::ffff:fff<br>vn-jdaf<br>76.1.1.247 | m1.medium   4GB RAM   2 VCPU   40.0GB<br>Disk | -       | Active | None | Running     | 4 days, 9 hours | Create Snapshot More |

## Enabling DHCPv6 In Virtual Machines

To allow IPv6 address assignment using DHCPv6, the virtual machine network interface configuration must be updated appropriately.

For example, to enable DHCPv6 for Ubuntu-based virtual machines, add the following line in the `/etc/network/interfaces` file:

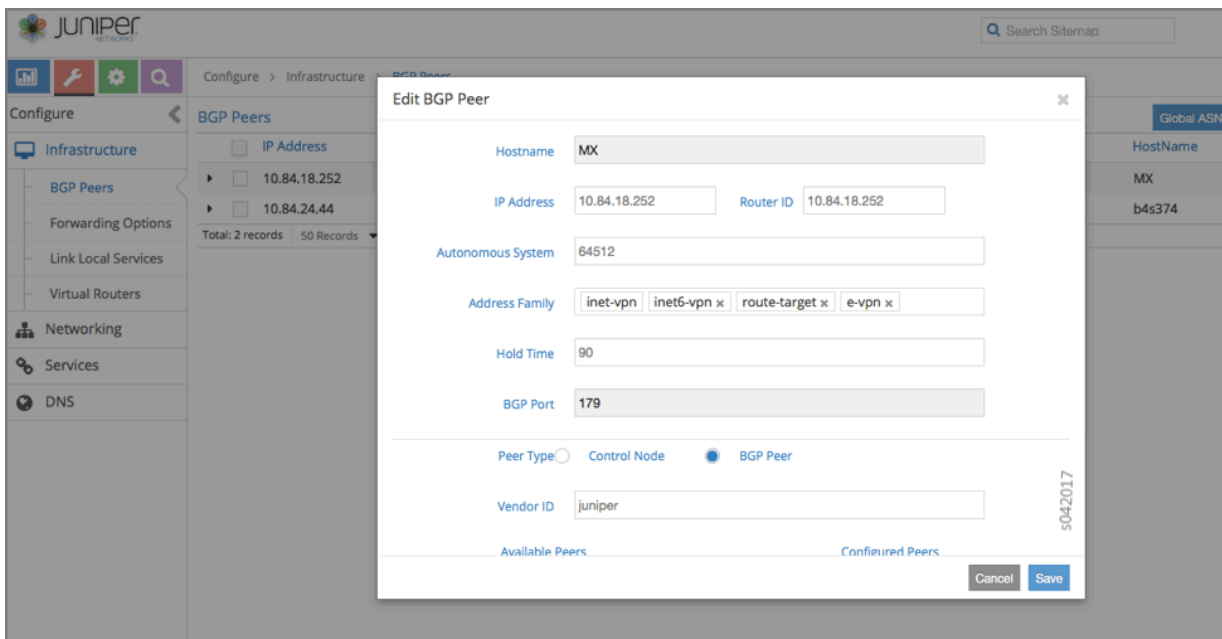
```
iface eht0 inet6 dhcp
```

Also, `dhclient -6` can be run from within the virtual machine to get IPv6 addresses using DHCPv6.

## Adding IPv6 Peers

The procedure to add an IPv6 BGP peer in Contrail is similar to adding an IPv4 peer. Select **Configure > Infrastructure > BGP Peers**, include `inet6-vpn` in the Address Family list to allow advertisement of IPv6 addresses.

A sample is shown in the following.



**NOTE:** Additional configuration is required on the peer router to allow `inet6-vpn` peering.

## Configuring EVPN and VXLAN

### IN THIS SECTION

- [Configuring the VXLAN Identifier Mode | 551](#)
- [Configuring Forwarding | 553](#)
- [Configuring the VXLAN Identifier | 554](#)
- [Configuring Encapsulation Methods | 555](#)

Contrail supports Ethernet VPNs (EVPN) and Virtual Extensible Local Area Networks (VXLAN).

EVPN is a flexible solution that uses Layer 2 overlays to interconnect multiple edges (virtual machines) within a data center. Traditionally, the data center is built as a flat Layer 2 network with issues such as flooding, limitations in redundancy and provisioning, and high volumes of MAC address learning, which cause churn during node failures. EVPNs are designed to address these issues without disturbing flat MAC connectivity.

In EVPNs, MAC address learning is driven by the control plane, rather than by the data plane, which helps control learned MAC addresses across virtual forwarders, thus avoiding flooding. The forwarders advertise locally learned MAC addresses to the controllers. The controllers use MP-BGP to communicate with peers. The peering of controllers using BGP for EVPN results in better and faster convergence.

With EVPN, MAC learning is confined to the virtual networks to which the virtual machine belongs, thus isolating traffic between multiple virtual networks. In this manner, virtual networks can share the same MAC addresses without any traffic crossover.

#### *Unicast in EVPNs*

Unicast forwarding is based on MAC addresses where traffic can terminate on a local endpoint or is encapsulated to reach the remote endpoint. Encapsulation can be MPLS/UDP, MPLS/GRE, or VXLAN.

#### *BUM Traffic in EVPN*

Multicast and broadcast traffic is flooded in a virtual network. The replication tree is built by the control plane, based on the advertisements of end nodes (virtual machines) sent by forwarders. Each virtual network has one distribution tree, a method that avoids maintaining multicast states at fabric nodes, so the nodes are unaffected by multicast. The replication happens at the edge forwarders. Per-group subscription is not provided. Broadcast, unknown unicast, and multicast (BUM) traffic is handled the same way, and gets flooded in the virtual network to which the virtual machine belongs.

## VXLAN

VXLAN is an overlay technology that encapsulates MAC frames into a UDP header at Layer 2. Communication is established between two virtual tunnel endpoints (VTEPs). VTEPs encapsulate the virtual machine traffic into a VXLAN header, as well as strip off the encapsulation. Virtual machines can only communicate with each other when they belong to the same VXLAN segment. A 24-bit virtual network identifier (VNID) uniquely identifies the VXLAN segment. This enables having the same MAC frames across multiple VXLAN segments without traffic crossover. Multicast in VXLAN is implemented as Layer 3 multicast, in which endpoints subscribe to groups.

### *Design Details of EVPN and VXLAN*

In Contrail Release 1.03 and later, EVPN is enabled by default. The supported forwarding modes include:

- Fallback bridging—IPv4 traffic lookup is performed using the IP FIB. All non-IPv4 traffic is directed to a MAC FIB.
- Layer 2-only— All traffic is forwarded using a MAC FIB lookup.

You can configure the forwarding mode individually on each virtual network.

EVPN is used to share MAC addresses across different control planes in both forwarding models. The result of a MAC address lookup is a next hop, which, similar to IP forwarding, points to a local virtual machine or a tunnel to reach the virtual machine on a remote server. The tunnel encapsulation methods supported for EVPN are MPLSoGRE, MPLSoUDP, and VXLAN. The encapsulation method selected is based on a user-configured priority.

In VXLAN, the VNID is assigned uniquely for every virtual network carried in the VXLAN header. The VNID uniquely identifies a virtual network. When the VXLAN header is received from the fabric at a remote server, the VNID lookup provides the VRF of the virtual machine. This VRF is used for the MAC lookup from the inner header, which then provides the destination virtual machine.

Non-IP multicast traffic uses the same multicast tree as for IP multicast (255.255.255.255). The multicast is matched against the all-broadcast prefix in the bridging table (FF:FF:FF:FF:FF:FF). VXLAN is not supported for IP/non-IP multicast traffic.

The following table summarizes the traffic and encapsulation types supported for EVPN.

|              |            | Encapsulation |          |       |
|--------------|------------|---------------|----------|-------|
|              |            | MPLS-GRE      | MPLS-UDP | VXLAN |
| Traffic Type | IP unicast | Yes           | Yes      | No    |

|                |     |     |     |
|----------------|-----|-----|-----|
| IP-BUM         | Yes | Yes | No  |
| non IP unicast | Yes | Yes | Yes |
| non IP-BUM     | Yes | Yes | No  |

## Configuring the VXLAN Identifier Mode

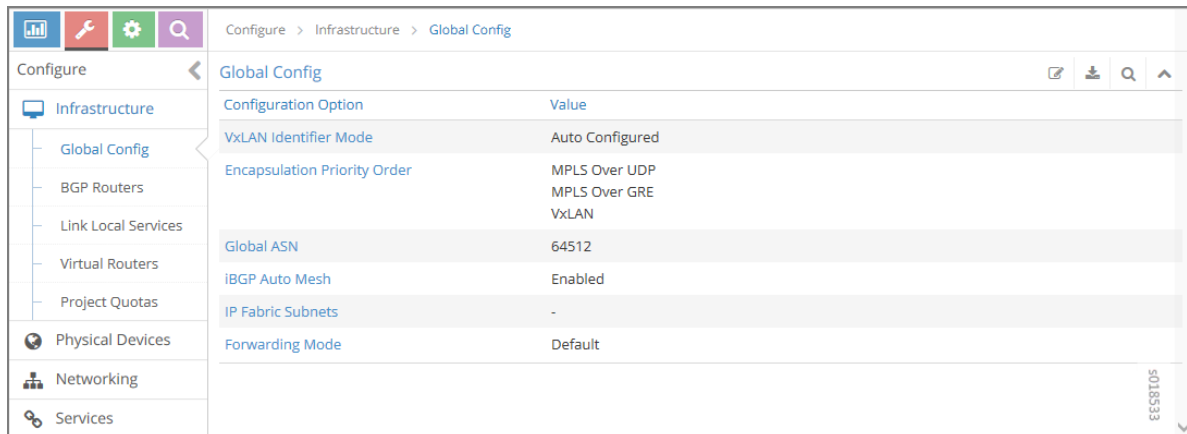
You can configure the global VXLAN identifier mode to select an auto-generated VNID or a user-generated VXLAN ID, either through the Contrail Web UI or by modifying a python file.

To configure the global VXLAN identifier mode:

1. From the Contrail Web UI, select **Configure > Infrastructure > Global Config**.

The Global Config options and values are displayed in the Global Config window.

**Figure 124: Global Config Window for VXLAN ID**



2. Click the edit icon



.

The Edit Global Config window is displayed as shown in [Figure 125 on page 552](#).

Figure 125: Edit Global Config Window for VXLAN Identifier Mode

The screenshot shows the 'Edit Global Config' window with the following settings:

- Forwarding Options:**
  - Forwarding Mode: Default
  - VxLAN Identifier Mode:  User Configured (Auto Configured is unselected)
  - Encapsulation Priority Order:
    - MPLS Over UDP (+ -)
    - MPLS Over GRE (+ -)
    - VxLAN (+ -)
- BGP Options:**
  - Global ASN: 64512

Buttons: Cancel, Save

3. Select one of the following:

- **Auto Configured**— The VXLAN identifier is automatically assigned for the virtual network.
- **User Configured**— You must provide the VXLAN identifier for the virtual network.

**NOTE:** When **User Configured** is selected, if you do not provide an identifier, then VXLAN encapsulation *is not used* and the mode falls back to MPLS.

Alternatively, you can set the VXLAN identifier mode by using Python to modify the `/opt/contrail/utlils/encap.py` file as follows:

```
python encap.py <add | update | delete > <username > < password > < tenant_name > < config_node_ip >
```



## Configuring Forwarding

In Contrail, the default forwarding mode is enabled for fallback bridging (IP FIB and MAC FIB). The mode can be changed, either through the Contrail Web UI or by using python provisioning commands.

To change the forwarding mode:

1. From the Contrail Web UI, select **Configure > Networking > Networks**.
2. Select the virtual network that you want to change the forwarding mode for.
3. Click the gear icon



and select **Edit**.

The Edit Network window is displayed as shown in [Figure 126 on page 553](#).

**Figure 126: Edit Network Window**

| IPAM                         | CIDR            | Allocation Pools | Gateway      | DNS                                 | DHCP                                |     |
|------------------------------|-----------------|------------------|--------------|-------------------------------------|-------------------------------------|-----|
| TestProjectC5Ca5C-ipam655... | 31.222.172.0/24 |                  | 31.222.172.1 | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | + - |

Under the Advanced Options select the forwarding mode from the following choices:

- Select **Default** to enable the default forwarding mode.
- Select **L2 and L3** to enable IP and MAC FIB (fallback bridging).
- Select **L2 Only** to enable only MAC FIB.
- Select **L3 Only** to enable only IP.

**NOTE:** The full list of forwarding modes are only displayed if you change entries in the `/usr/src/contrail/contrail-web-core/config/config.global.js` file. For example:

1. To make the **L2** selection available locate the following:

```
config.network = {};
config.network.L2_enable = false;
```

2. Change the entry to the following:

```
config.network = {};
config.network.L2_enable = true;
```

3. To make the other selections available, modify the corresponding entries.
4. Save the file and quit the editor.
5. Restart the Contrail Web user interface process (webui).

Alternatively, you can use the following python provisioning command to change the forwarding mode:

```
python provisioning_forwarding_mode --project_fq_name 'defaultdomain: admin' --vn_name vn1 --forwarding_mode <
12_13| 12 >
```

Options:

12\_13 = Enable IP FIB and MAC FIB (fallback bridging)

12 = Enable MAC FIB only (Layer 2 only)

## Configuring the VXLAN Identifier

The VXLAN identifier can be set only if the VXLAN network identifier mode has been set to User Configured. You can then set the VXLAN ID by either using the Contrail Web UI or by using Python commands.

To configure the global VXLAN identifier:

1. From the Contrail Web UI, select **Configure > Networking > Networks**.
2. Select the virtual network that you want to change the forwarding mode for.

3. Click the gear icon



and select **Edit**.

The Edit Network window is displayed. Select the **Advanced Options** as shown in [Figure 127 on page 555](#).

**Figure 127: Edit Network Window for VXLAN Identifier**

4. Type the VXLAN identifier.
5. Click **Save**.

Alternatively, you can use the following Python provisioning command to configure the VXLAN identifier:

```
python provisioning_forwarding_mode --project_fq_name 'defaultdomain: admin' --vn_name vn1 --forwarding_mode < vxlan_id >
```

## Configuring Encapsulation Methods

The default encapsulation mode for EVPN is MPLS over UDP. All packets on the fabric are encapsulated with the label allocated for the virtual machine interface. The label encoding and decoding is the same as for IP forwarding. Additional encapsulation methods supported for EVPN include MPLS over GRE and VXLAN. MPLS over UDP is different from MPLS over GRE only in the method of tunnel header encapsulation.

VXLAN has its own header and uses a VNID label to carry the traffic over the fabric. A VNID is assigned with every virtual network and is shared by all virtual machines in the virtual network. The VNID is mapped to the VRF of the virtual network to which it belongs.

The priority order in which to apply encapsulation methods is determined by the sequence of methods set either from the Contrail Web UI or in the `encap.py` file.

To configure the global VXLAN identifier mode:

- From the Contrail Web UI, select **Configure > Infrastructure > Global Config**.
- The Global Config options are displayed.
- Click the edit icon



The Edit Global Config window is displayed as shown in [Figure 128 on page 557](#).

Figure 128: Edit Global Config Window for Encapsulation Priority Order

Under Encapsulation Priority Order select one of the following:

- **MPLS over UDP**
- **MPLS over GRE**
- **VxLAN**

Click the + plus symbol to the right of the first priority to add a second priority or third priority.

Use the following procedure to change the default encapsulation method to VXLAN by editing the `encap.py` file.

**NOTE:** VXLAN is *only* supported for EVPN unicast. It is not supported for IP traffic or multicast traffic. VXLAN priority and presence in the `encap.py` file or configured in the Web UI is ignored for traffic not supported by VXLAN.

To set the priority of encapsulation methods to VXLAN:

1. Modify the **encap.py** file found in the **/opt/contrail/utils/** directory.

The default encapsulation line is:

```
encap_obj=EncapsulationPrioritiesType(encapsulation=['MPLSoUDP', 'MPLSoGRE'])
```

Modify the line to:

```
encap_obj=EncapsulationPrioritiesType(encapsulation=['VXLAN', 'MPLSoUDP', 'MPLSoGRE'])
```

2. After the status is modified, execute the following script:

```
python encap_set.py <add|update|delete> <username> <password> <tenant_name> <config_node_ip>
```

The configuration is applied globally for all virtual networks.

# Example of Deploying a Multi-Tier Web Application Using Contrail

## IN THIS CHAPTER

- [Example: Deploying a Multi-Tier Web Application | 559](#)
- [Sample Network Configuration for Devices for Simple Tiered Web Application | 567](#)

## Example: Deploying a Multi-Tier Web Application

### IN THIS SECTION

- [Multi-Tier Web Application Overview | 559](#)
- [Example: Setting Up Virtual Networks for a Simple Tiered Web Application | 560](#)
- [Verifying the Multi-Tier Web Application | 563](#)
- [Sample Addressing Scheme for Simple Tiered Web Application | 563](#)
- [Sample Physical Topology for Simple Tiered Web Application | 564](#)
- [Sample Physical Topology Addressing | 565](#)

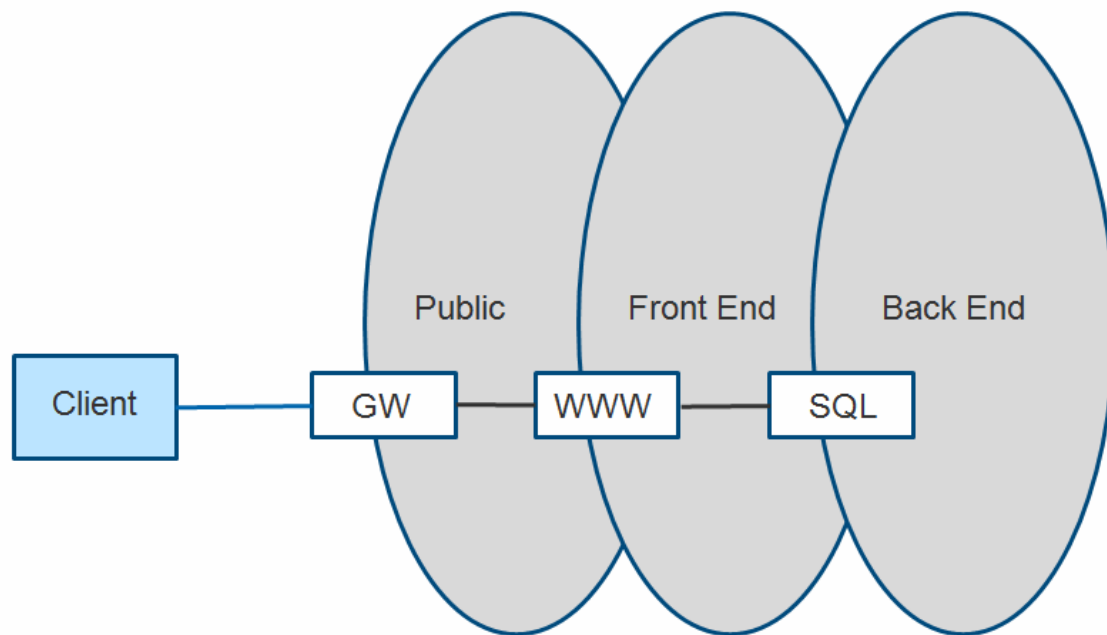
### Multi-Tier Web Application Overview

A common requirement for a cloud tenant is to create a tiered web application in leased cloud space. The tenant enjoys the favorable economics of a private IT infrastructure within a shared services environment. The tenant seeks speedy setup and simplified operations.

The following example shows how to set up a simple tiered web application using Contrail. The example has a web server that a user accesses by means of a public floating IP address. The front-end web server gets the content it serves to customers from information stored in a SQL database server that resides on a back-end network. The web server can communicate directly with the database server without going

through any gateways. The public (or client) can only communicate to the web server on the front-end network. The client is not allowed to communicate directly with any other parts of the infrastructure. See [Figure 129 on page 560](#).

**Figure 129: Simple Tiered Web Use Case**



### Example: Setting Up Virtual Networks for a Simple Tiered Web Application

This example provides basic steps for setting up a simple multi-tier network application. Basic creation steps are provided, along with links to the full explanation for each of the creation steps. Refer to the links any time you need more information about completing a step.

1. Working with a system that has the Contrail software installed and provisioned, create a project named **demo**.

For more information; see *Creating Projects in OpenStack for Configuring Tenants in Contrail*.

2. In the **demo** project, create three virtual networks:

- a. A network named **public** with IP address **10.84.41.0/24**

This is a special use virtual network for floating IP addresses— it is assigned an address block from the public floating address pool that is assigned to each web server. The assigned block is the only address block advertised outside of the data center to clients that want to reach the web services provided.



- b. A network named **frontend** with IP address **192.168.1.0/24**

This network is the location where the web server virtual machine instances are launched and attached. The virtual machines are identified with private addresses that have been assigned to this virtual network.

- c. A network named **backend** with IP address **192.168.2.0/24**

This network is the location where the database server virtual machines instances are launched and attached. The virtual machines are identified with private addresses that have been assigned to this virtual network.

For more information; see *Creating a Virtual Network with OpenStack Contrail* or *Creating a Virtual Network with Juniper Networks Contrail*.

3. Create a floating IP pool named **public\_pool** for the **public** network within the **demo** project; see [Figure 130 on page 562](#).

Figure 130: Create Floating IP Pool

The screenshot shows the 'Edit Network public' dialog box with the following configuration:

- Network Name:** public
- Network Policy(s):** Select Policies...
- Address Management:** default-network... (dropdown), xxx.xxx.xxx.xxx/xx (input), + - (toggle)
- Table:**

| IPAM                 | IP Block      |
|----------------------|---------------|
| default-network-ipam | 10.84.41.0/24 |
- Floating IP Pools:** public\_pool (input), demo x (input), + - (toggle)
- Pool Name:** admin (dropdown)

Buttons: Cancel, Save

Vertical text on the right: s041841

- Allocate the floating IP pool **public\_pool** to the **demo** project; see [Figure 131 on page 562](#).

Figure 131: Allocate Floating IP

The screenshot shows the 'Allocate Floating IP' dialog box with the following configuration:

- Floating IP Pool:** public:public\_pool (dropdown)

Buttons: Cancel, Save

Vertical text on the right: s041842

5. Verify that the floating IP pool has been allocated; see **Configure > Networking > Allocate Floating IPs**.
6. Create a policy that allows any host to talk to any host using any IP address, protocol, and port, and apply this policy between the **frontend** network and the **backend** network.  
This now allows communication between the web servers in the front-end network and the database servers in the back-end network.
7. Launch the virtual machine instances that represent the web server and the database server.

**NOTE:** Your installation might not include the virtual machines needed for the web server and the database server. Contact your account team if you need to download the VMs for this setup.

On the **Instances** tab for this project, select **Launch Instance** and for each instance that you launch, complete the fields to make the following associations:

- Web server VM: select **frontend** network and the policy created to allow communication between **frontend** and **backend** networks. Apply the floating IP address pool to the web server.
- Database server VM: select **backend** network and the policy created to allow communication between **frontend** and **backend** networks.

## Verifying the Multi-Tier Web Application

Verify your web setup.

- To demonstrate this web application setup, go to the client machine, open a browser, and navigate to the address in the **public** network that is assigned to the web server in the **frontend** network.  
The result will display the Contrail interface with various data populated, verifying that the web server is communicating with the database server in the **backend** network and retrieving data.  
  
The client machine only has access to the public IP address. Attempts to browse to any of the addresses assigned to the **frontend** network or to the **backend** network should fail.

## Sample Addressing Scheme for Simple Tiered Web Application

Use the information in [Table 38 on page 564](#) as a guide for addressing devices in the simple tiered web example.

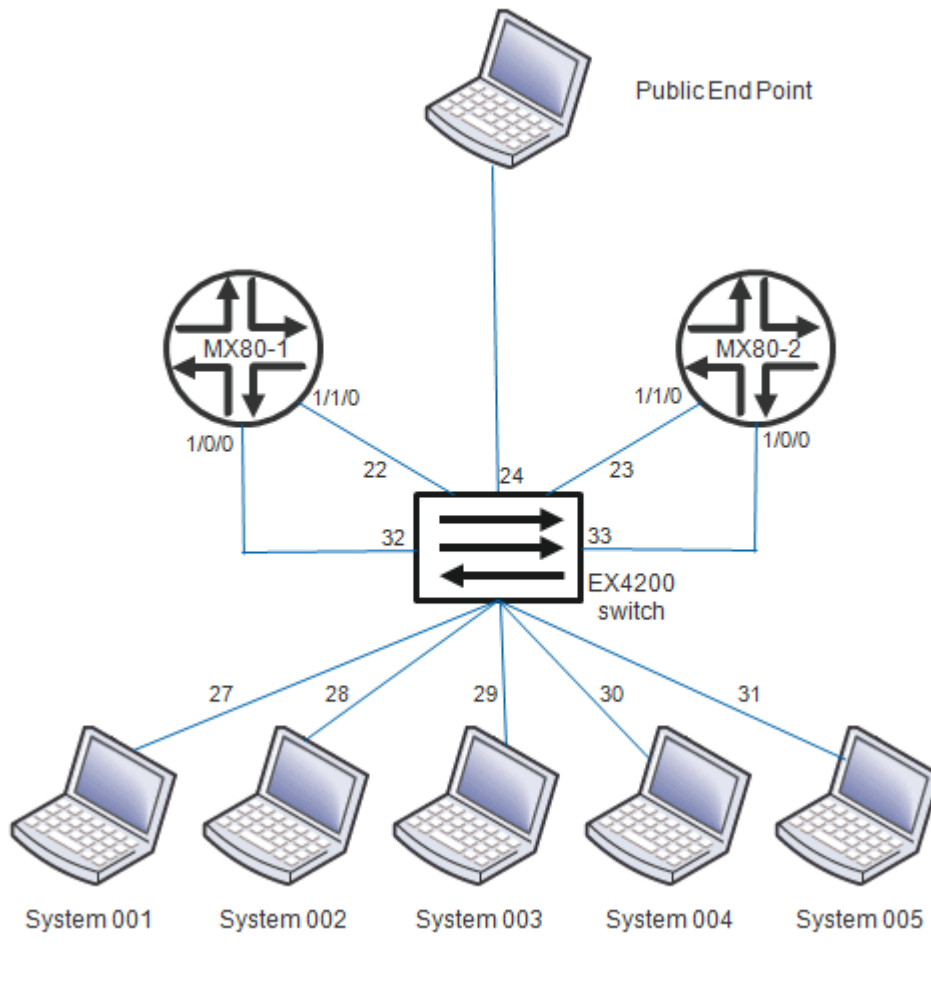
**Table 38: Sample Addressing Scheme for Example**

| System Name                       | Address Allocation   |
|-----------------------------------|--|
| System001                         | 10.84.11.100   |
| System002                         | 10.84.11.101   |
| System003                         | 10.84.11.102   |
| System004                         | 10.84.11.103   |
| System005                         | 10.84.11.104   |
| MX80-1                            | 10.84.11.253<br>10.84.45.1 (public connection)                                       |
| MX80-2                            | 10.84.11.252<br>10.84.45.2 (public connection)                                       |
| EX4200                            | 10.84.11.254<br>10.84.45.254 (public connection)<br>10.84.63.259 (public connection) |
| frontend network                  | 192.168.1.0/24   |
| backend network                   | 192.168.2.0/24   |
| public network (floating address) | 10.84.41.0/24  |

### Sample Physical Topology for Simple Tiered Web Application

[Figure 132 on page 565](#) provides a guideline diagram for the physical topology for the simple tiered web application example.

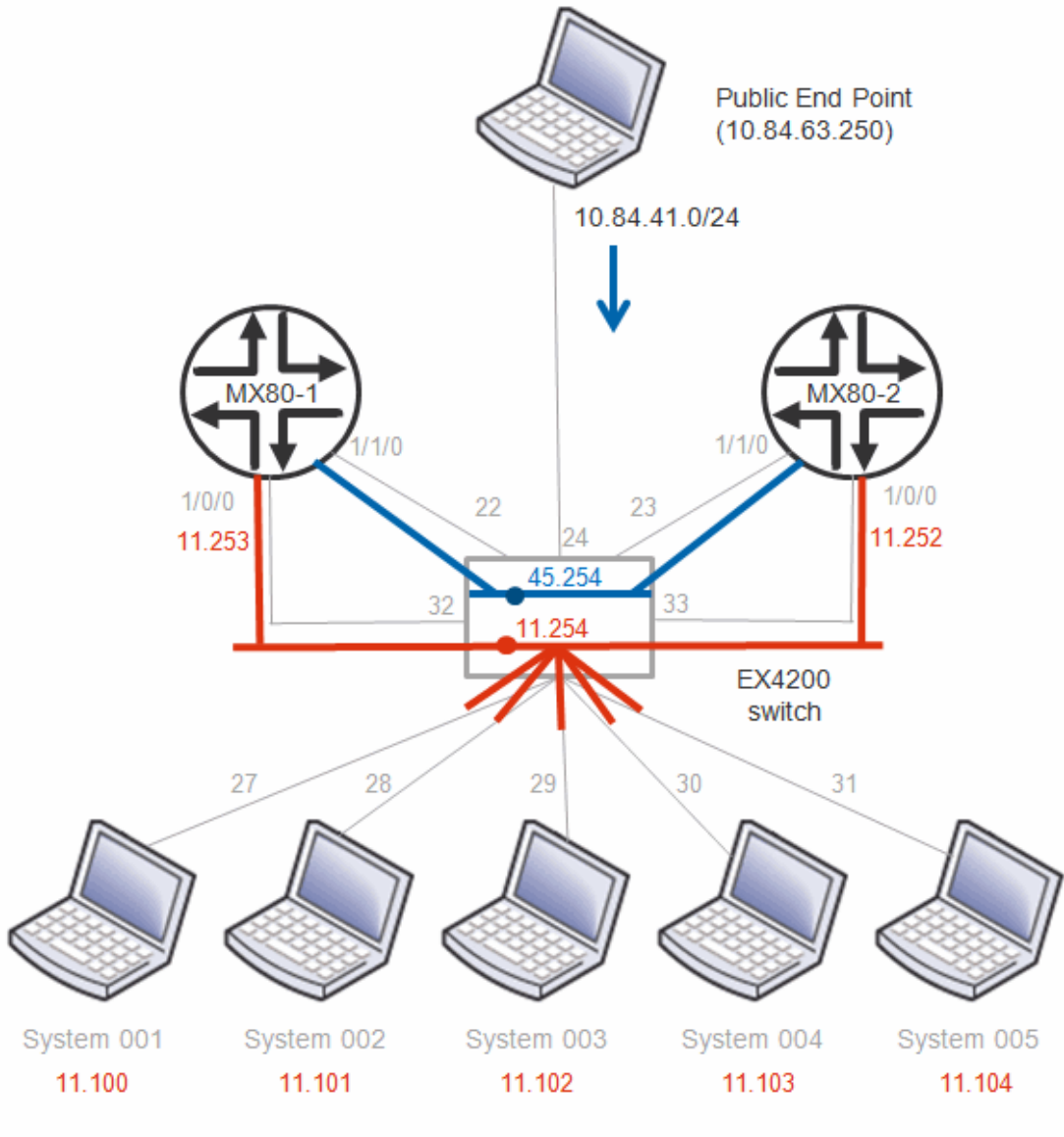
Figure 132: Sample Physical Topology for Simple Tiered Web Application



### Sample Physical Topology Addressing

Figure 133 on page 566 provides a guideline diagram for addressing the physical topology for the simple tiered web application example.

Figure 133: Sample Physical Topology Addressing



**SEE ALSO**

| *Sample Network Configuration for Devices for Simple Tiered Web Application*

## Sample Network Configuration for Devices for Simple Tiered Web Application

This section shows sample device configurations that can be used to create the *Example: Deploying a Multi-Tier Web Application*. Configurations are shown for Juniper Networks devices: two MX80s and one EX4200.

### *MX80-1 Configuration*

```
version 12.2R1.3;
system {
  root-authentication {
    encrypted-password "xxxxxxxxxx"; ## SECRET-DATA
  }
  services {
    ssh {
      root-login allow;
    }
  }
  syslog {
    user * {
      any emergency;
    }
    file messages {
      any notice;
      authorization info;
    }
  }
}
chassis {
  fpc 1 {
    pic 0 {
      tunnel-services;
    }
  }
}
interfaces {
  ge-1/0/0 {
    unit 0 {
      family inet {
        address 10.84.11.253/24;
      }
    }
  }
}
```

```
    }
  }
}
ge-1/1/0 {
  description "IP Fabric interface";
  unit 0 {
    family inet {
      address 10.84.45.1/24;
    }
  }
}
lo0 {
  unit 0 {
    family inet {
      address 127.0.0.1/32;
    }
  }
}
}
routing-options {
  static {
    route 0.0.0.0/0 next-hop 10.84.45.254;
  }
  route-distinguisher-id 10.84.11.253;
  autonomous-system 64512;
  dynamic-tunnels {
    setup1 {
      source-address 10.84.11.253;
      gre;
      destination-networks {
        10.84.11.0/24;
      }
    }
  }
}
}
protocols {
  bgp {
    group mx {
      type internal;
      local-address 10.84.11.253;
      family inet-vpn {
        unicast;
      }
    }
  }
}
```



```

        neighbor 10.84.11.252;
    }
    group contrail-controller {
        type internal;
        local-address 10.84.11.253;
        family inet-vpn {
            unicast;
        }
        neighbor 10.84.11.101;
        neighbor 10.84.11.102;
    }
}
}
routing-instances {
    customer-public {
        instance-type vrf;
        interface ge-1/1/0.0;
        vrf-target target:64512:10000;
        routing-options {
            static {
                route 0.0.0.0/0 next-hop 10.84.45.254;
            }
        }
    }
}
}
}

```

### *MX80-2 Configuration*

```

version 12.2R1.3;
system {
    root-authentication {
        encrypted-password "xxxxxxxx"; ## SECRET-DATA
    }
    services {
        ssh {
            root-login allow;
        }
    }
    syslog {
        user * {
            any emergency;
        }
    }
}

```

```
    }
    file messages {
        any notice;
        authorization info;
    }
}
chassis {
    fpc 1 {
        pic 0 {
            tunnel-services;
        }
    }
}
interfaces {
    ge-1/0/0 {
        unit 0 {
            family inet {
                address 10.84.11.252/24;
            }
        }
    }
    ge-1/1/0 {
        description "IP Fabric interface";
        unit 0 {
            family inet {
                address 10.84.45.2/24;
            }
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 127.0.0.1/32;
            }
        }
    }
}
routing-options {
    static {
        route 0.0.0.0/0 next-hop 10.84.45.254;
    }
    route-distinguisher-id 10.84.11.252;
}
```

```
autonomous-system 64512;
dynamic-tunnels {
    setup1 {
        source-address 10.84.11.252;
        gre;
        destination-networks {
            10.84.11.0/24;
        }
    }
}
}
protocols {
    bgp {
        group mx {
            type internal;
            local-address 10.84.11.252;
            family inet-vpn {
                unicast;
            }
            neighbor 10.84.11.253;
        }
        group contrail-controller {
            type internal;
            local-address 10.84.11.252;
            family inet-vpn {
                unicast;
            }
            neighbor 10.84.11.101;
            neighbor 10.84.11.102;
        }
    }
}
}
routing-instances {
    customer-public {
        instance-type vrf;
        interface ge-1/1/0.0;
        vrf-target target:64512:10000;
        routing-options {
            static {
                route 0.0.0.0/0 next-hop 10.84.45.254;
            }
        }
    }
}
```

```

    }
  }
}

```

### *EX4200 Configuration*

```

system {
  host-name EX4200;
  time-zone America/Los_Angeles;
  root-authentication {
    encrypted-password "xxxxxxxxxxxx"; ## SECRET-DATA
  }
  login {
    class read {
      permissions [ clear interface view view-configuration ];
    }
    user admin {
      uid 2000;
      class super-user;
      authentication {
        encrypted-password "xxxxxxxxxxxx"; ## SECRET-DATA
      }
    }
    user user1 {
      uid 2002;
      class read;
      authentication {
        encrypted-password "xxxxxxxxxxxx"; ## SECRET-DATA
      }
    }
  }
}
services {
  ssh {
    root-login allow;
  }
  telnet;
  netconf {
    ssh;
  }
  web-management {
    http;
  }
}

```

```
}
syslog {
  user * {
    any emergency;
  }
  file messages {
    any notice;
    authorization info;
  }
  file interactive-commands {
    interactive-commands any;
  }
}
}
chassis {
  aggregated-devices {
    ethernet {
      device-count 64;
    }
  }
}
}
```

# Configuring Services

## IN THIS CHAPTER

- [Configuring DNS Servers | 574](#)
- [Distributed Service Resource Allocation with Containerized Contrail | 586](#)
- [Support for Multicast | 597](#)
- [Using Static Routes with Services | 600](#)
- [Configuring Metadata Service | 604](#)

## Configuring DNS Servers

### IN THIS SECTION

- [DNS Overview | 574](#)
- [Defining Multiple Virtual Domain Name Servers | 575](#)
- [IPAM and Virtual DNS | 576](#)
- [DNS Record Types | 576](#)
- [Configuring DNS Using the Interface | 577](#)
- [Configuring DNS Using Scripts | 585](#)

### DNS Overview

Domain Name System (DNS) is the standard protocol for resolving domain names into IP addresses so that traffic can be routed to its destination. DNS provides the translation between human-readable domain names and their IP addresses. The domain names are defined in a hierarchical tree, with a root followed by top-level and next-level domain labels.

A DNS server stores the records for a domain name and responds to queries from clients based on these records. The server is authoritative for the domains for which it is configured to be the name server. For other domains, the server can act as a caching server, fetching the records by querying other domain name servers.

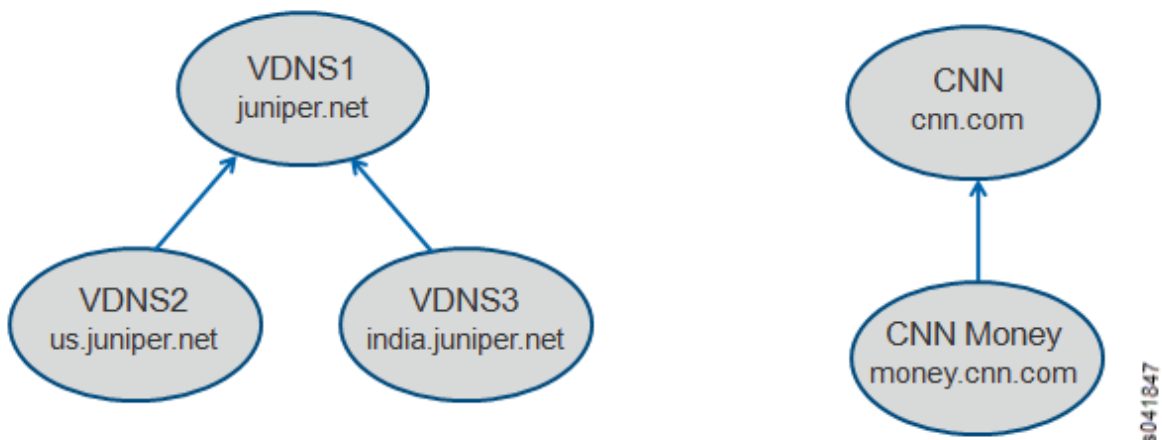
The following are the key attributes of domain name service in a virtual world:

- It should be possible to configure multiple domain name servers to provide name resolution service for the virtual machines spawned in the system.
- It should be possible to configure the domain name servers to form DNS server hierarchies required by each tenant.
  - The hierarchies can be independent and completely isolated from other similar hierarchies present in the system, or they can provide naming service to other hierarchies present in the system.
- DNS records for the virtual machines spawned in the system should be updated dynamically when a virtual machine is created or destroyed.
- The service should be scalable to handle an increase in servers and the resulting increased numbers of virtual machines and DNS queries handled in the system.

### Defining Multiple Virtual Domain Name Servers

Conrail provides the flexibility to define multiple virtual domain name servers under each domain in the system. Each virtual domain name server is an authoritative server for the DNS domain configured. [Figure 134 on page 575](#) shows examples of virtual DNS servers defined in **default-domain**, providing the name service for the DNS domains indicated.

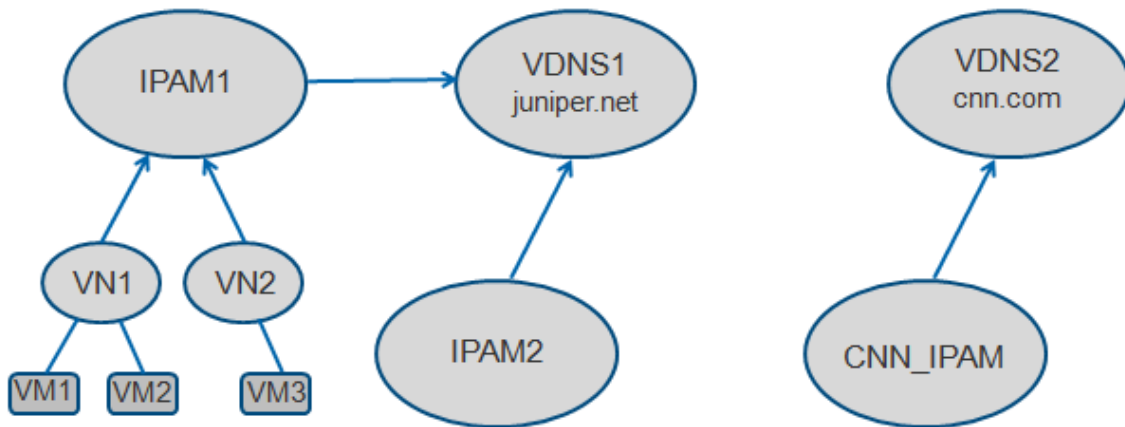
Figure 134: DNS Servers Examples



## IPAM and Virtual DNS

Each IP address management (IPAM) service in the system can refer to one of the virtual DNS servers configured. The virtual networks and virtual machines spawned are associated with the DNS domain specified in the corresponding IPAM. When the VMs are configured with DHCP, they receive the domain assignment in the DHCP **domain-name** option. Examples are shown in [Figure 135 on page 576](#)

Figure 135: IPAM and Virtual DNS



8041848

## DNS Record Types

DNS records can be added statically. DNS record types **A**, **CNAME**, **PTR**, and **NS** are currently supported in the system. Each record includes the type, class (IN), name, data, and TTL values. See [Table 39 on page 576](#) for descriptions of the record types.

Table 39: DNS Record Types Supported

| DNS Record Type | Description  |
|-----------------|--|
| <b>A</b>        | Used for mapping hostnames to IPv4 addresses. Name refers to the name of the virtual machine, and data is the IPv4 address of the virtual machine. |
| <b>CNAME</b>    | Provides an alias to a name. Name refers to the name of the virtual machine, and data is the new name (alias) for the virtual machine.             |

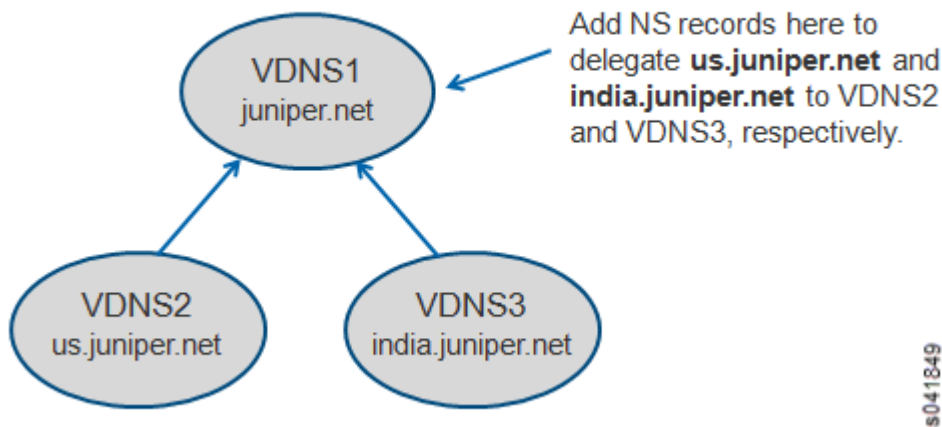


Table 39: DNS Record Types Supported (Continued)

| DNS Record Type | Description  |
|-----------------|--|
| <b>PTR</b>      | A pointer to a record, it provides reverse mapping from an IP address to a name. Name refers to the IP address, and data is the name for the virtual machine. The address in the PTR record should be part of a subnet configured for a VN within one of the IPAMs referring to this virtual DNS server.                                     |
| <b>NS</b>       | Used to delegate a subdomain to another DNS server. The DNS server could be another virtual DNS server defined in the system or the IP address of an external DNS server reachable via the infrastructure. Name refers to the subdomain being delegated, and data is the name of the virtual DNS server or IP address of an external server. |

Figure 136 on page 577 shows an example usage for the DNS record type of NS.

Figure 136: Example Usage for NS Record Type



S041849

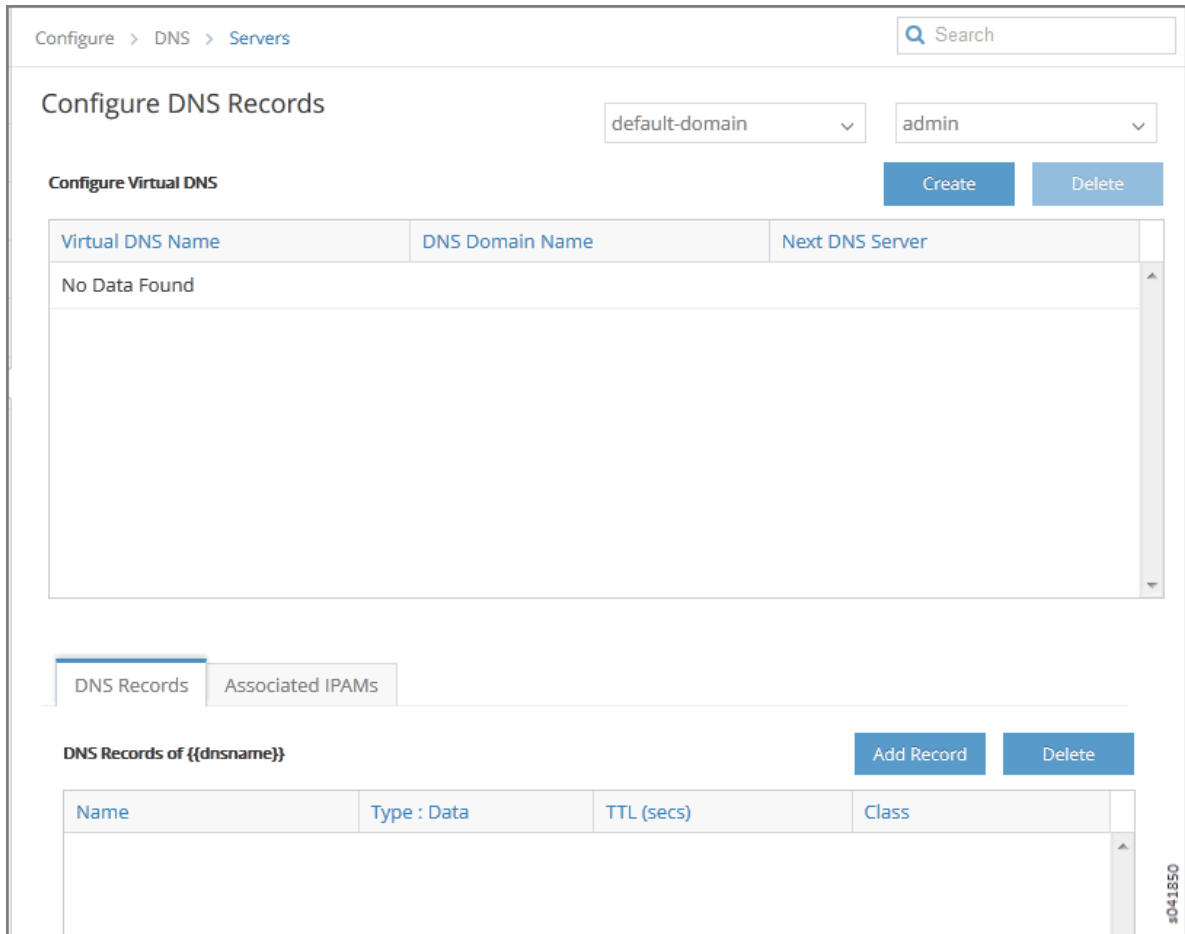
## Configuring DNS Using the Interface

DNS can be configured by using the user interface or by using scripts. The following procedure shows how to configure DNS through the Juniper Networks Contrail interface.

1. Access **Configure > DNS > Servers** to create or delete virtual DNS servers and records.

The **Configure DNS Records** page appears; see [Figure 137 on page 578](#).

Figure 137: Configure DNS Records



- To add a new DNS server, click the **Create** button.  
Enter DNS server information in the **Add DNS** window; see [Figure 138 on page 579](#)

Figure 138: Add DNS

Complete the fields for the new server; see [Table 40 on page 579](#).

Table 40: Add DNS Fields

| Field                  | Description   |
|------------------------|---|
| <b>Server Name</b>     | Enter a name for this server.   |
| <b>Domain Name</b>     | Enter the name of the domain for this server.   |
| <b>Time To Live</b>    | Enter the <b>TTL</b> in seconds.  |
| <b>Next DNS Server</b> | Select from a list the name of the next DNS server to process DNS requests if they cannot be processed at this server, or <b>None</b> . |

Table 40: Add DNS Fields *(Continued)*

| Field                       | Description   |
|-----------------------------|---|
| <b>Load Balancing Order</b> | Select the load-balancing order from a list— <b>Random</b> , <b>Fixed</b> , <b>Round Robin</b> . When a name has multiple records matching, the configured record order determines the order in which the records are sent in the response. Select <b>Random</b> to have the records sent in random order. Select <b>Fixed</b> to have records sent in the order of creation. Select <b>Round Robin</b> to have the record order cycled for each request to the record. |
| <b>OK</b>                   | Click <b>OK</b> to create the record.   |
| <b>Cancel</b>               | Click <b>Cancel</b> to clear the fields and start over.   |

- To add a new DNS record, from the **Configure DNS Records** page, click the **Add Record** button in the lower right portion of the screen.

The **Add DNS Record** window appears; see [Figure 139 on page 581](#).

Figure 139: Add DNS Record

4. Complete the fields for the new record; see [Table 41 on page 581](#).

Table 41: Add DNS Record Fields

| Field               | Description  |
|---------------------|--|
| <b>Record Name</b>  | Enter a name for this record.  |
| <b>Type</b>         | Select the record type from a list— <b>A</b> , <b>CNAME</b> , <b>PTR</b> , <b>NS</b> . |
| <b>IP Address</b>   | Enter the IP address for the location for this record.                                 |
| <b>Class</b>        | Select the record class from a list— <b>IN</b> is the default.                         |
| <b>Time To Live</b> | Enter the <b>TTL</b> in seconds.   |
| <b>OK</b>           | Click <b>OK</b> to create the record.  |

Table 41: Add DNS Record Fields (Continued)

| Field         | Description   |
|---------------|---|
| <b>Cancel</b> | Click <b>Cancel</b> to clear the fields and start over. |

- To associate an IPAM to a virtual DNS server, from the **Configure DNS Records** page, select the **Associated IPAMs** tab in the lower right portion of the screen and click the **Edit** button.

The **Associate IPAMs to DNS** window appears; see [Figure 140 on page 582](#).

Figure 140: Associate IPAMs to DNS

Complete the IPAM associations, using the field descriptions in [Table 42 on page 582](#).

Table 42: Associate IPAMs to DNS Fields

| Field                         | Description  |
|-------------------------------|--|
| <b>Associate to All IPAMs</b> | Select this box to associate the selected DNS server to all available IPAMs. |

Table 42: Associate IPAMs to DNS Fields *(Continued)*

| Field                   | Description  |
|-------------------------|--|
| <b>Available IPAMs</b>  | This column displays the currently available IPAMs.  |
| <b>Associated IPAMs</b> | This column displays the IPAMs currently associated with the selected DNS server.  |
| <b>&gt;&gt;</b>         | Use this button to associate an available IPAM to the selected DNS server, by selecting an available IPAM in the left column and clicking this button to move it to the Associated IPAMs column. The selected IPAM is now associated with the selected DNS server.       |
| <b>&lt;&lt;</b>         | Use this button to disassociate an IPAM from the selected DNS server, by selecting an associated IPAM in the right column and clicking this button to move it to the left column (Available IPAMs). The selected IPAM is now disassociated from the selected DNS server. |
| <b>OK</b>               | Click <b>OK</b> to commit the changes indicated in the window.   |
| <b>Cancel</b>           | Click <b>Cancel</b> to clear all entries and start over.   |

6. Use the **IP Address Management** page (**Configure > Networking > IP Address Management**); see [Figure 141 on page 583](#)) to configure the DNS mode for any DNS server and to associate an IPAM to DNS servers of any mode or to tenants' IP addresses.

Figure 141: Configure IP Address Management



7. To associate an IPAM to a virtual DNS server or to tenant's IP addresses, at the **IP Address Management** page, select the network associated with this IPAM, then click the **Action** button in the last column, and click **Edit**.

The **Edit IP Address Management** window appears; see [Figure 142 on page 584](#).

Figure 142: DNS Server

8. In the first field, select the **DNS Method** from a list (**None**, **Default DNS**, **Tenant DNS**, **Virtual DNS**); see [Table 43 on page 584](#).

Table 43: DNS Modes

| DNS Mode       | Description   |
|----------------|---|
| <b>None</b>    | Select <b>None</b> when no DNS support is required for the VMs.   |
| <b>Default</b> | In default mode, DNS resolution for VMs is performed based on the name server configuration in the server infrastructure. The subnet default gateway is configured as the DNS server for the VM, and the DHCP response to the VM has this DNS server option. DNS requests sent by a VM to the default gateway are sent to the name servers configured on the respective compute nodes. The responses are sent back to the VM. |



Table 43: DNS Modes (*Continued*)

| DNS Mode           | Description   |
|--------------------|---|
| <b>Tenant</b>      | Configure this mode when a tenant wants to use its own DNS servers. Configure the list of servers in the IPAM. The server list is sent in the DHCP response to the VM as DNS servers. DNS requests sent by the VMs are routed the same as any other data packet based on the available routing information. |
| <b>Virtual DNS</b> | Configure this mode to support virtual DNS servers (VDNS) to resolve the DNS requests from the VMs. Each IPAM can have a virtual DNS server configured in this mode.  |

9. Complete the remaining fields on this page, and click **OK** to commit the changes, or click **Cancel** to clear the fields and start over.

## Configuring DNS Using Scripts

DNS can be configured via the user interface or by using scripts that are available in the **opt/contrail/utils** directory. The scripts are described in [Table 44 on page 585](#).



**CAUTION:** Be aware of the following cautions when using scripts to configure DNS:

- DNS doesn't allow special characters in the names, other than - (dash) and . (period). Any records that include special characters in the name will be discarded by the system.
- The IPAM DNS mode and association should only be edited when there are *no* virtual machine instances in the virtual networks associated with the IPAM.

Table 44: DNS Scripts

| Action                   | Script   |
|--------------------------|--|
| Add a virtual DNS server | <p>Script: <code>add_virtual_dns.py</code></p> <p>Sample usage: <code>python add_virtual_dns.py --api_server_ip 10.204.216.21 --api_server_port 8082 --name vdns1 --domain_name default-domain --dns_domain juniper.net --dyn_updates --record_order random --ttl 1200 --next_vdns default-domain:vdns2</code></p> |

Table 44: DNS Scripts (Continued)

| Action   | Script  |
|--|---|
| Delete a virtual DNS server                    | <p>Script: <code>del_virtual_dns_record.py</code></p> <p>Sample usage: <code>python del_virtual_dns.py --api_server_ip 10.204.216.21 --api_server_port 8082 --fq_name default-domain:vdns1</code></p>   |
| Add a DNS record                               | <p>Script: <code>add_virtual_dns_record.py</code></p> <p>Sample usage: <code>python add_virtual_dns_record.py --api_server_ip 10.204.216.21 --api_server_port 8082 --name rec1 --vdns_fqname default-domain:vdns1 --rec_name one --rec_type A --rec_class IN --rec_data 1.2.3.4 --rec_ttl 2400</code></p> |
| Delete a DNS record                            | <p>Script: <code>del_virtual_dns_record.py</code></p> <p>Sample usage: <code>python del_virtual_dns_record.py --api_server_ip 10.204.216.21 --api_server_port 8082 --fq_name default-domain:vdns1:rec1</code></p>   |
| Associate a virtual DNS server with an IPAM    | <p>Script: <code>associate_virtual_dns.py</code></p> <p>Sample usage: <code>python associate_virtual_dns.py --api_server_ip 10.204.216.21 --api_server_port 8082 --ipam_fqname default-domain:demo:ipam1 --vdns_fqname default-domain:vdns1</code></p>  |
| Disassociate a virtual DNS server with an IPAM | <p>Script: <code>disassociate_virtual_dns.py</code></p> <p>Sample usage: <code>python disassociate_virtual_dns.py --api_server_ip 10.204.216.21 --api_server_port 8082 --ipam_fqname default-domain:demo:ipam1 --vdns_fqname default-domain:vdns1</code></p>  |

## Distributed Service Resource Allocation with Containerized Contrail

### IN THIS SECTION

- Replacement of Centralized Discovery Service in Contrail 4.0 | 587

- [New Distributed Resource Allocation Manager | 587](#)
- [Changes in Configuration Files | 588](#)

Starting with Contrail Release 4.0, the existing centralized Contrail discovery service is replaced with a distributed method of allocating service resources.

## Replacement of Centralized Discovery Service in Contrail 4.0

In Contrail releases prior to Release 4.0, the Contrail discovery service is a centralized service resource allocation module with high availability, used primarily to automatically load-balance service resources in the system.

In the previous centralized discovery method, new service resources are registered (published) directly to the Contrail discovery module and allocated to the requester (subscriber) of the service resource, without disrupting the running state of the subscribers.

The centralized discovery method requires using a database to:

- synchronize across Contrail discovery nodes.
- maintain the list of publishers, subscribers, and the health of published services across reloads.
- provide a centralized view of the service allocation and health of the services.

This centralized discovery method resulted in unnecessary system churn when services were falsely marked as down, due to periodic health updates of services made to the database nodes, resulting in reallocation of healthy services.

Starting with Contrail 4.0, the Contrail discovery services centralized resource allocation manager has been removed. Its replacement is a distributed resource allocation list of service nodes, maintained in each module of the system.

## New Distributed Resource Allocation Manager

Starting with Contrail Release 4.0, service resources are managed with a distributed allocation manager, with the following features:

- Each system module is provisioned with a list of service nodes (publishers).
- Each system module randomizes the list of service nodes and uses the resources. The randomized list is expected to be fairly load-balanced.

- When currently-used services are down, the system module detects the down immediately and reacts with no downtime by selecting another service from the list. This is distinctly different from the previous model, in which the module would need to contact the discovery service to check for available services, resulting in a finite time loss for allocation, distribution, and application of a new set of services.
- When service nodes are added or deleted, the system administrator updates the configuration file of all daemons using the service type of the service node added or deleted, sending a SIGHUP to the respective daemons.
- Each daemon randomizes the service list independently and reallocates the resources.

## Deprecation of IF-MAP

In Contrail 4.0, the Interface for Metadata Access Points (IF-MAP) methodology has been deprecated. Contrail 4.0 uses CONFIGDB sections in configuration files instead of IF-MAP sections.

## Changes in Configuration Files

[Table 45 on page 588](#) lists configuration files in the Contrail system that have changes to enable the distributed service resource allocation system, starting with Contrail 4.0. In general, the changes include removing (deprecating) discovery server sections and subsections, and adding parameters needed to identify service resources in all modules.

Each daemon randomizes the published service list and uses the resources. Additionally, each daemon provides a SIGHUP handler to manage the addition or deletion of publishers.

**Table 45: Contrail 4.0 Changes in Configuration Files**

| Configuration File          | Configuration Parameter | Changes   |
|-----------------------------|-------------------------|---|
| contrail-vrouter-agent.conf | [DISCOVERY]             | Section deprecated  |
|                             | [CONTROL-NODE].servers  | Provisioned list of control-node [role=control] service providers in the format:<br>ip-address:port ip-address2:port<br>Example: 10.1.1.1:5269 10.1.1.12:5269 |

Table 45: Contrail 4.0 Changes in Configuration Files (*Continued*)

| Configuration File    | Configuration Parameter         | Changes   |
|-----------------------|---------------------------------|---|
|                       | [DNS].servers                   | Provisioned list of DNS [role=control] service providers in the format:<br><br>ip-address:port ip-address2:port<br>Example: 10.1.1.1:53 10.1.1.2:5              |
|                       | [DEFAULT].collectors            | Provisioned list of Collector [role=collector] service providers in the format:<br><br>ip-address:port ip-address2:port Example:<br>10.1.1.1:8086 10.1.1.2:8086 |
| contrail-control.conf | [DISCOVERY]                     | Section deprecated  |
|                       | [DEFAULT].collectors            | Provisioned list of Collector [role=collector] service providers in the format:<br><br>ip-address:port ip-address2:port Example:<br>10.1.1.1:8086 10.1.1.2:8086 |
|                       | [CONFIGDB].rabbitmq_server_list | Provisioned list of config-node [role=cfgm] service providers in the format:<br><br>ip-address:port ip-address2:port Example:<br>10.1.1.1:5672 10.1.1.2:5672    |
|                       | [CONFIGDB].rabbitmq_user        | guest (default string)  |
|                       | [CONFIGDB].rabbitmq_password    | guest (default string)  |

Table 45: Contrail 4.0 Changes in Configuration Files (*Continued*)

| Configuration File | Configuration Parameter                  | Changes  |
|--------------------|--|--|
|                    | [CONFIGDB].config_db_server_list         | Provisioned list of Config DB [role=database] service providers in the format:<br><br>ip-address:port ip-address2:port Example:<br>10.1.1.1:9042 10.1.1.2:9042<br><br>NOTE: Docker uses 9041 as port |
|                    | [CONFIGDB].certs_store                   | Deprecated   |
|                    | [CONFIGDB].password                      | Deprecated   |
|                    | [CONFIGDB].server_url                    | Deprecated   |
|                    | [CONFIGDB].user                          | Deprecated   |
|                    | [CONFIGDB].stale_entries_cleanup_timeout | Deprecated   |
|                    | [CONFIGDB].end_of_rib_timeout            | Deprecated   |
| contrail-dns.conf  |  |  |
|                    | [DISCOVERY]                              | Deprecated   |
|                    | [DEFAULT].collectors                     | Provisioned list of Collector [role=collector] service providers in the format:<br><br>ip-address:port ip-address2:port Example:<br>10.1.1.1:8086 10.1.1.2:8086                                      |

Table 45: Contrail 4.0 Changes in Configuration Files (*Continued*)

| Configuration File      | Configuration Parameter                  | Changes  |
|-------------------------|--|--|
|                         | [CONFIGDB].rabbitmq_server_list          | Provisioned list of config-node [role=cfgm] service providers in the format:<br><br>ip-address:port ip-address2:port Example:<br>10.1.1.1:5672 10.1.1.2:5672                                     |
|                         | [CONFIGDB].rabbitmq_user                 | guest (default string)   |
|                         | [CONFIGDB].rabbitmq_password             | guest (default string)   |
|                         | [CONFIGDB].config_db_server_list         | Provisioned list of Config DB [role=database] service providers in the format:<br><br>ip-address:port ip-address2:port Example:<br>10.1.1.1:9042 10.1.1.2:9042 NOTE:<br>Dockers use 9041 as port |
|                         | [CONFIGDB].certs_store                   | Deprecated   |
|                         | [CONFIGDB].password                      | Deprecated   |
|                         | [CONFIGDB].server_url                    | Deprecated   |
|                         | [CONFIGDB].user                          | Deprecated   |
|                         | [CONFIGDB].stale_entries_cleanup_timeout | Deprecated   |
|                         | [CONFIGDB].end_of_rib_timeout            | Deprecated   |
| contrail-collector.conf | [DISCOVERY]                              | Deprecated   |

Table 45: Contrail 4.0 Changes in Configuration Files (*Continued*)

| Configuration File          | Configuration Parameter      | Changes   |
|-----------------------------|------------------------------|---|
|                             | [API_SERVER].api_server_list | Provisioned list of api-servers [role=config] in the format:<br><br>ip-address:port<br><br>Example: 10.1.1.1:8082 10.1.1.2:8082                                     |
| contrail-alarm-gen.conf     | [DISCOVERY]                  | Deprecated  |
|                             | [DEFAULTS].collectors        | Provisioned list of Collector [role=collector] service providers in the format:<br><br>ip-address:port ip-address2:port<br><br>Example: 10.1.1.1:8086 10.1.1.2:8086 |
|                             | [API_SERVER].api_server_list | Provisioned list of api-servers [role=config] in the format:<br><br>ip-address:port<br><br>Example: 10.1.1.1:8082 10.1.1.2:8082                                     |
|                             | [REDIS].redis_uve_list       | Provisioned list of redis instances [role=collector]<br><br>Example: 192.168.0.29:6379<br>192.168.0.30:6379   |
| contrail-analytics-api.conf | [DISCOVERY]                  | Section deprecated  |
|                             | [DEFAULTS].collectors        | Provisioned list of collector [role=collector] service providers in the format:<br><br>ip-address:port ip-address2:port<br><br>Example: 10.1.1.1:8086 10.1.1.2:8086 |



Table 45: Contrail 4.0 Changes in Configuration Files (Continued)

| Configuration File           | Configuration Parameter | Changes   |
|------------------------------|-------------------------|---|
|                              | [REDIS].redis_uve_list  | Provisioned list of redis instances<br>[role=collector]<br><br>Example: 192.168.0.29:6379<br>192.168.0.30:6379  |
| contrail-api.conf            | [DISCOVERY]             | Section deprecated  |
|                              | [DEFAULTS].collectors   | Provisioned list of collector<br>[role=collector] service providers in the<br>format:<br><br>ip-address:port ip-address2:port<br><br>Example: 10.1.1.1:8086 10.1.1.2:8086 |
| contrail-schema.conf         | [DISCOVERY]             | Section deprecated  |
|                              | [DEFAULTS].collectors   | Provisioned list of Collector<br>[role=collector] service providers in ip-<br>address:port ip-address2:port<br><br>Example: 10.1.1.1:8086 10.1.1.2:8086                   |
| contrail-svc-monitor.conf    | [DISCOVERY]             | Section deprecated  |
|                              | [DEFAULTS].collectors   | Provisioned list of Collector<br>[role=collector] service providers in the<br>format:<br><br>ip-address:port ip-address2:port<br><br>Example: 10.1.1.1:8086 10.1.1.2:8086 |
| contrail-device-manager.conf | [DISCOVERY]             | Section deprecated  |

Table 45: Conrail 4.0 Changes in Configuration Files (Continued)

| Configuration File             | Configuration Parameter | Changes   |
|--------------------------------|-------------------------|---|
|                                | [DEFAULTS].collectors   | Provisioned list of Collector [role=collector] service providers in ip-address:port ip-address2:port format<br><br>Example: 10.1.1.1:8086 10.1.1.2:8086             |
| conrail-analytics-nodemgr.conf | [DISCOVERY]             | Section deprecated  |
|                                | [COLLECTOR].server_list | Provisioned list of Collector [role=collector] service providers in the format:<br><br>ip-address:port ip-address2:port Example: 10.1.1.1:8086 10.1.1.2:8086        |
| conrail-config-nodemgr.conf    | [DISCOVERY]             | Section deprecated  |
|                                | [COLLECTOR].server_list | Provisioned list of Collector [role=collector] service providers in the format:<br><br>ip-address:port ip-address2:port<br><br>Example: 10.1.1.1:8086 10.1.1.2:8086 |
| conrail-control-nodemgr.conf   | [DISCOVERY]             | Section deprecated  |
|                                | [COLLECTOR].server_list | Provisioned list of Collector [role=collector] service providers in ip-address:port ip-address2:port<br><br>Example: 10.1.1.1:8086 10.1.1.2:8086                    |
| conrail-database-nodemgr.conf  | [DISCOVERY]             | Section deprecated  |

Table 45: Contrail 4.0 Changes in Configuration Files (Continued)

| Configuration File            | Configuration Parameter | Changes   |
|-------------------------------|-------------------------|---|
|                               | [COLLECTOR].server_list | Provisioned list of Collector [role=collector] service providers in the format:<br><br>ip-address:port ip-address2:port<br><br>Example: 10.1.1.1:8086 10.1.1.2:8086 |
| contrail-vrouter-nodemgr.conf | [DISCOVERY]             | Section deprecated  |
|                               | [COLLECTOR].server_list | Provisioned list of Collector [role=collector] service providers in the format:<br><br>ip-address:port ip-address2:port<br><br>Example: 10.1.1.1:8086 10.1.1.2:8086 |
| contrail-query-engine.conf    | [DISCOVERY]             | Section deprecated  |
|                               | [COLLECTOR].server_list | Provisioned list of Collector [role=collector] service providers in the format:<br><br>ip-address:port ip-address2:port<br><br>Example: 10.1.1.1:8086 10.1.1.2:8086 |
| contrail-snmp-collector.conf  | [DISCOVERY]             | Section deprecated  |
|                               | [DEFAULTS].collectors   | Provisioned list of Collector [role=collector] service providers in the format:<br><br>ip-address:port ip-address2:port<br><br>Example: 10.1.1.1:8086 10.1.1.2:8086 |

Table 45: Contrail 4.0 Changes in Configuration Files (*Continued*)

| Configuration File     | Configuration Parameter      | Changes   |
|------------------------|------------------------------|---|
|                        | [API_SERVER].api_server_list | Provisioned list of api-servers [role=config] in the format:<br><br>ip-address:port<br><br>Example: 10.1.1.1:8082 10.1.1.2:8082                                 |
| contrail-topology.conf | [DISCOVERY]                  | Section deprecated  |
|                        | [DEFAULTS].collectors        | Provisioned list of Collector [role=collector] service providers in the format:<br><br>ip-address:port ip-address2:port Example:<br>10.1.1.1:8086 10.1.1.2:8086 |
|                        | [API_SERVER].api_server_list | Provisioned list of api-servers [role=config] in ip-address:port<br><br>Example: 10.1.1.1:8082 10.1.1.2:8082  |

**Contrail Web UI**

|                  |                         |  |
|------------------|-------------------------|--|
| config.global.js | config.discovery.server | Discovery subsection deprecated  |
|                  | config.discovery.port   | Discovery subsection deprecated  |
|                  | config.cnfg.server_ip   | Provisioned list of Config [role=cfgm] service providers as list of ip-address<br><br>Example: ['10.1.1.1 10.1.1.2'] |
|                  | config.cnfg.server_port | Server port as a string<br><br>Example: '8082'   |

**Table 45: Contrail 4.0 Changes in Configuration Files (Continued)**

| Configuration File | Configuration Parameter                   | Changes  |
|--------------------|---|--|
|                    | <code>config.analytics.server_ip</code>   | Provisioned list of Collector [role=collector] service providers as a list of ip-address<br><br>Example: ['10.1.1.1 10.1.1.2'] |
|                    | <code>config.analytics.server_port</code> | Server port as a string<br><br>Example: '8081'   |
|                    | <code>config.dns.server_ip</code>         | Provisioned list of Controller [role=control] service providers as a list of ip-address<br><br>Example: ['10.1.1.1 10.1.1.2']  |
|                    | <code>config.dns.server_port</code>       | Server port as a string<br><br>Example: '8092'   |

## Support for Multicast

### IN THIS SECTION

- [Subnet Broadcast | 598](#)
- [All-Broadcast/Limited-Broadcast and Link-Local Multicast | 598](#)
- [Host Broadcast | 599](#)

This section describes how the Contrail Controller supports broadcast and multicast.

## Subnet Broadcast

Multiple subnets can be attached to a virtual network when it is spawned. Each of the subnets has one subnet broadcast route installed in the unicast routing table assigned to that virtual network. The recipient list for the subnet broadcast route includes all of the virtual machines that belong to that subnet. Packets originating from any VM in that subnet are replicated to all members of the recipient list, except the originator. Because the next hop is the list of recipients, it is called a composite next hop.

If there is no virtual machine spawned under a subnet, the subnet routing entry discards the packets received. If all of the virtual machines in a subnet are turned off, the routing entry points to discard. If the IPAM is deleted, the subnet route corresponding to that IPAM is deleted. If the virtual network is turned off, all of the subnet routes associated with the virtual network are removed.

### *Subnet Broadcast Example*

The following configuration is made:

1. Virtual network name – **vn1**
2. Unicast routing instance – `vn1.uc.inet`
3. Subnets (IPAM) allocated – `1.1.1.0/24`; `2.2.0.0/16`; `3.3.0.0/16`
4. Virtual machines spawned – `vm1 (1.1.1.253)`; `vm2 (1.1.1.252)`; `vm3 (1.1.1.251)`; `vm4 (3.3.1.253)`

The following subnet route additions are made to the routing instance `vn1.uc.inet.0`:

1. `1.1.1.255 -> forward to NH1 (composite next hop)`
2. `2.2.255.255 -> DROP`
3. `3.3.255.255 -> forward to NH2`
- 4.
5. The following entries are made to the next-hop table:
6. **NH1** – `1.1.1.253`; `1.1.1.252`; `1.1.1.251`
7. **NH2** – `3.3.1.253`

If traffic originates for `1.1.1.255` from `vm1 (1.1.1.253)`, it will be forwarded to `vm2 (1.1.1.252)` and `vm3 (1.1.1.251)`. The originator `vm1 (1.1.1.253)` will not receive the traffic even though it is listed as a recipient in the next hop.

## All-Broadcast/Limited-Broadcast and Link-Local Multicast

The address group `255.255.255.255` is used with all-broadcast (limited-broadcast) and multicast traffic. The route is installed in the multicast routing instance. The source address is recorded as ANY, so the route is

ANY/255.255.255.255 (\*,G). It is unique per routing instance, and is associated with its corresponding virtual network. When a virtual network is spawned, it usually contains multiple subnets, in which virtual machines are added. All of the virtual machines, regardless of their subnets, are part of the recipient list for ANY/255.255.255.255. The replication is sent to every recipient except the originator.

Link-local multicast also uses the all-broadcast method for replication. The route is deleted when all virtual machines in this virtual network are turned off or the virtual network itself is deleted.

### *All-Broadcast Example*

The following configuration is made:

1. Virtual network name – vn1
2. Unicast routing instance – vn1.uc.inet
3. Subnets (IPAM) allocated – 1.1.1.0/24; 2.2.0.0/16; 3.3.0.0/16
4. Virtual machines spawned – vm1 (1.1.1.253); vm2 (1.1.1.252); vm3 (1.1.1.251); vm4 (3.3.1.253)

The following subnet route addition is made to the routing instance vn1.uc.inet.0:

1. 255.255.255.255/\* -> NH1
- 2.

The following entries are made to the next-hop table:

1. NH1 – 1.1.1.253; 1.1.1.252; 1.1.1.251; 3.3.1.253

If traffic originates for 1.1.1.255 from vm1 (1.1.1.253), the traffic is forwarded to vm2 (1.1.1.252), vm3 (1.1.1.251), and vm4 (3.3.1.253). The originator vm1 (1.1.1.253) will not receive the traffic even though it is listed as a recipient in the next hop.

## **Host Broadcast**

The host broadcast route is present in the host routing instance so that the host operating system can send a subnet broadcast/all-broadcast (limited-broadcast). This type of broadcast is sent to the fabric by means of a **vhost** interface. Additionally, any subnet broadcast/all-broadcast received from the fabric will be handed over to the host operating system.

## Using Static Routes with Services

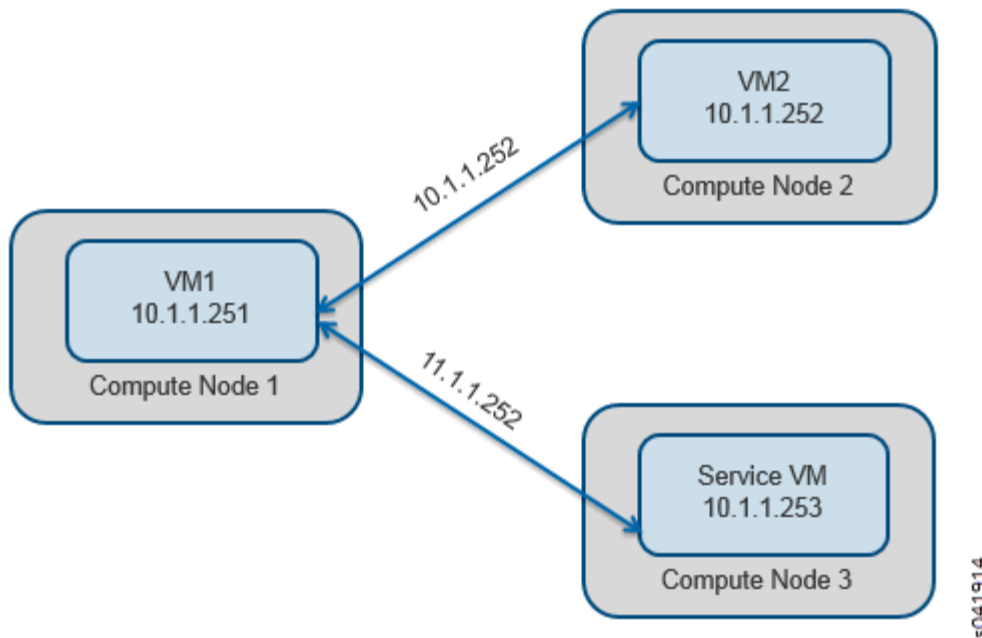
### IN THIS SECTION

- [Static Routes for Service Instances | 600](#)
- [Configuring Static Routes on a Service Instance | 601](#)
- [Configuring Static Routes on Service Instance Interfaces | 602](#)
- [Configuring Static Routes as Host Routes | 603](#)

### Static Routes for Service Instances

Static routes can be configured in a virtual network to direct traffic to a service virtual machine.

The following figure shows a virtual network with subnet 10.1.1.0/24. All of the traffic from a virtual machine that is directed to subnet 11.1.1.0/24 can be configured to be routed by means of a service virtual machine, by using the static route 11.1.1.252 configured on the service virtual machine interface.





## Configuring Static Routes on a Service Instance

To configure static routes on a service instance, first enable the static route option in the service template to be used for the service instance.

To enable the static route option in a service template:

1. Go to **Configure > Services > Service Templates** and click **Create**.
2. At **Add Service Template**, complete the fields for **Name**, **Service Mode**, and **Image Name**.
3. Select the **Interface Types** to use for the template, then for each interface type that might have a static route configured, click the check box under the **Static Routes** column to enable the static route option for that interface.

The following figure shows a service template in which the left and right interfaces of service instances have the static routes option enabled. Now a user can configure a static route on a corresponding interface on a service instance that is based on the service template shown.

Add Service Template
✕

**Name**

**Service Mode**

**Image Name**

| Interface Types  | Shared IP                | Static Routes                       |   |   |  |
|--|--------------------------|-------------------------------------|---|---|--|
| Management <input style="width: 20px;" type="text" value="v"/> | <input type="checkbox"/> | <input type="checkbox"/>            | + | - |  |
| Left <input style="width: 20px;" type="text" value="v"/>       | <input type="checkbox"/> | <input checked="" type="checkbox"/> | + | - |  |
| Right <input style="width: 20px;" type="text" value="v"/>      | <input type="checkbox"/> | <input checked="" type="checkbox"/> | + | - |  |

▶ [Advanced options](#)

---

s041915

Cancel
Save

## Configuring Static Routes on Service Instance Interfaces

To configure static routes on a service instance interface:

1. Go to **Configure > Services > Service Instances** and click **Create**.
2. At **Create Service Instances**, complete the fields for **Instance Name** and **Services Template**.
3. Select the virtual network for each of the interfaces
4. Click the **Static Routes** dropdown menu under each interface field for which the static routes option is enabled to open the **Static Routes** menu and configure the static routes in the fields provided.

**NOTE:** If the **Auto Configured** option is selected, traffic destined to the static route subnet is load balanced across service instances.

The following figure shows a configuration to apply a service instance between VN1 (10.1.1.0/24) and VN2 (11.1.1.0/24). The left interface of the service instance is configured with VN1 and the right interface is configured to be VN2 (11.1.1.0/24). The static route 11.1.1.0/24 is configured on the left interface, so that all traffic from VN1 that is destined to VN2 reaches the left interface of the service instance.

The screenshot shows the 'Create Service Instances' configuration window. The fields are as follows:

- Instance Name:** nat
- Services Template:** nat - [in-network (management, left, right)]
- Interface 1:** Management, Auto Configured
- Interface 2:** Left, vn1
- Static Routes (under Interface 2):**

| Prefix      | Next hop    |     |
|-------------|-------------|-----|
| 11.1.1.0/24 | Interface 2 | + - |
- Interface 3:** Right, vn2
- Static Routes (under Interface 3):** (empty)

Buttons: Cancel, Save

Reference ID: s041916

The following figure shows static route 10.1.1.0/24 configured on the right interface, so that all traffic from VN2 that is destined to VN1 reaches the right interface of the service virtual machine.

The screenshot shows the 'Create Service Instances' dialog box with two interface configurations:

- Interface 2:** Set to 'Left' with a dropdown menu showing 'vn1'. Under 'Static Routes', a table shows a route for prefix '11.1.1.0/24' with next hop 'Interface 2'.
- Interface 3:** Set to 'Right' with a dropdown menu showing 'vn2'. Under 'Static Routes', a table shows a route for prefix '10.1.1.0/24' with next hop 'Interface 3'.

At the bottom right, there are 'Cancel' and 'Save' buttons. A vertical ID '#041917' is visible on the right side of the dialog.

When the static routes are configured for both the left and the right interfaces, all inter-virtual network traffic is forwarded through the service instance.

## Configuring Static Routes as Host Routes

You can also use static routes for host routes for a virtual machine, by using the classless static routes option in the DHCP server response that is sent to the virtual machine.

The routes to be sent in the DHCP response to the virtual machine can be configured for each virtual network as it is created.

To configure static routes as host routes:

1. Go to **Configure > Network > Networks** and click **Create**.
2. At **Create Network**, click the **Host Routes** option and add the host routes to be sent to the virtual machines.

An example is shown in the following figure.

Create Network
✕

Address Management ipam1 ▾ IP Block Gateway + -

| IPAM  | IP Block   | Gateway   |
|-------|------------|-----------|
| ipam1 | 1.2.3.0/24 | 1.2.3.254 |

▶ Route Targets

---

▶ Floating IP Pools

---

▼ Host Routes

| IPAM    | Route Prefix | +   |
|---------|--------------|-----|
| ipam1 ▾ | 1.1.1.0/24   | + - |
| ipam1 ▾ | 2.2.2.0/24   | + - |

Cancel Save

s041918

## Configuring Metadata Service

OpenStack enables virtual machines to access metadata by sending an HTTP request to the link-local address 169.254.169.254. The metadata request from the virtual machine is proxied to Nova with additional HTTP header fields that Nova uses to identify the source instance, then responds with appropriate metadata.

In Contrail, the vRouter acts as the proxy, by trapping the metadata requests, adding the necessary header fields, and sending the requests to the Nova API server.

The metadata service is configured by setting the `linklocal-services` property on the `global-vrouter-config` object.

Use the following elements to configure the `linklocal-services` element for metadata service:

- `linklocal-service-name = metadata`
- `linklocal-service-ip = 169.254.169.254`

- linklocal-service-port = 80
- ip-fabric-service-ip = *[server-ip-address]*
- ip-fabric-service-port = *[server-port]*

The linklocal-services properties can be set from the Contrail UI (**Configure > Infrastructure > Link Local Services**) or by using the following command:

```
python /opt/contrail/utils/provision_linklocal.py --admin_user <user> --admin_password <passwd> --  
linklocal_service_name metadata --linklocal_service_ip 169.254.169.254 --linklocal_service_port 80 --  
ipfabric_service_ip --ipfabric_service_port 8775
```

# Configuring Service Chaining

## IN THIS CHAPTER

- [Service Chaining | 606](#)
- [Service Chaining MX Series Configuration | 611](#)
- [ECMP Load Balancing in the Service Chain | 613](#)
- [Customized Hash Field Selection for ECMP Load Balancing | 614](#)
- [Service Chain Version 2 with Port Tuple | 619](#)
- [Using the Contrail Heat Template | 623](#)
- [Service Chain Route Reorigination | 628](#)
- [Service Instance Health Checks | 650](#)

## Service Chaining

### IN THIS SECTION

- [Service Chaining Basics | 606](#)
- [Service Chaining Configuration Elements | 608](#)

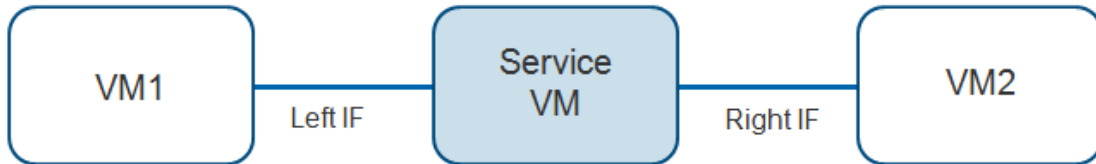
Contrail Controller supports chaining of various Layer 2 through Layer 7 services such as firewall, NAT, IDP, and so on.

### Service Chaining Basics

Services are offered by instantiating service virtual machines to dynamically apply single or multiple services to virtual machine (VM) traffic. It is also possible to chain physical appliance-based services.

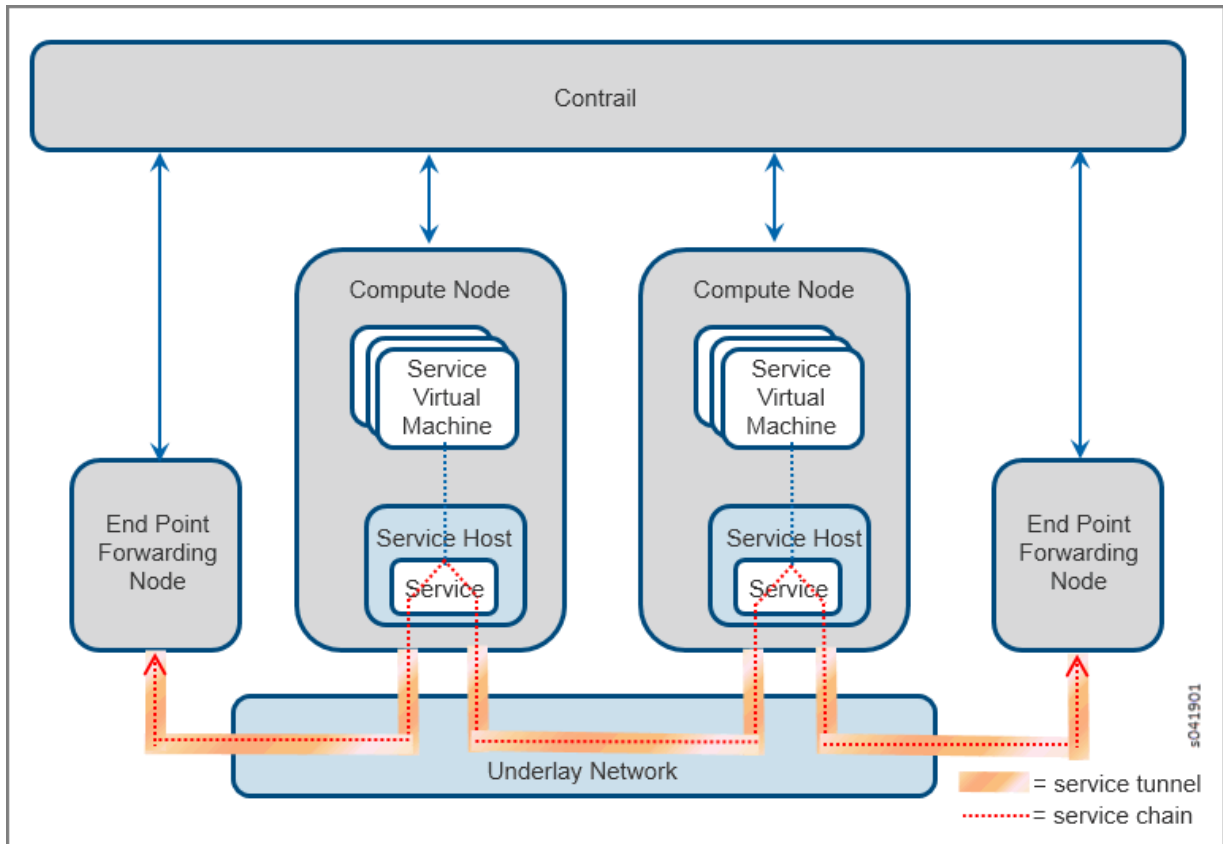
[Figure 143 on page 607](#) shows the basic service chain schema, with a single service. The service VM spawns the service, using the convention of left interface (left IF) and right interface (right IF). Multiple services can also be chained together.

**Figure 143: Service Chaining**



When you create a service chain, the Contrail software creates tunnels across the underlay network that span through all services in the chain. [Figure 144 on page 608](#) shows two end points and two compute nodes, each with one service instance and traffic going to and from one end point to the other.

Figure 144: Contrail Service Chain



The following are the modes of services that can be configured.

1. *Transparent or bridge mode*

- a. Used for services that do not modify the packet. Also known as bump-in-the-wire or Layer 2 mode. Examples include Layer 2 firewall, IDP, and so on.

2. *In-network or routed mode*

- a. Provides a gateway service where packets are routed between the service instance interfaces. Examples include NAT, Layer 3 firewall, load balancer, HTTP proxy, and so on.

3. *In-network-nat mode*

- a. Similar to in-network mode, however, return traffic does not need to be routed to the source network. In-network-nat mode is particularly useful for NAT service.

## Service Chaining Configuration Elements

Service chaining requires the following configuration elements in the solution:

- Service template



- Service instance
- Service policy

### *Service Template*

Service templates are always configured in the scope of a domain, and the templates can be used on all projects within a domain. A template can be used to launch multiple service instances in different projects within a domain.

The following are the parameters to be configured for a service template:

- Service template name
- Domain name
- Service mode
  - Transparent
  - In-Network
  - In-Network NAT
- Image name (for virtual service)
  - If the service is a virtual service, then the name of the image to be used must be included in the service template. In an OpenStack setup, the image must be added to the setup by using Glance.
- Interface list
  - Ordered list of interfaces---this determines the order in which Interfaces will be created on the service instance.
  - Most service templates will have management, left, and right interfaces. For service instances requiring more interfaces, “other” interfaces can be added to the interface list.
  - Shared IP attribute, per interface
  - Static routes enabled attribute, per interface
- Advanced options
  - Service scaling— use this attribute to enable a service instance to have more than one instance of the service instance virtual machine.
  - Flavor—assign an OpenStack flavor to be used while launching the service instance. Flavors are defined in OpenStack Nova with attributes such as assignments of CPU cores, memory, and disk space.

### *Service Instance*

A service instance is always maintained within the scope of a project. A service instance is launched using a specified service template from the domain to which the project belongs.

The following are the parameters to be configured for a service instance:

- Service instance name
- Project name
- Service template name
- Number of virtual machines that will be spawned
  - Enable service scaling in the service template for multiple virtual machines
- Ordered virtual network list
  - Interfaces listed in the order specified in the service template
  - Identify virtual network for each interface
  - Assign static routes for virtual networks that have static route enabled in the service template for their interface
    - Traffic that matches an assigned static route is directed to the service instance on the interface created for the corresponding virtual network

### *Service Policy*

The following are the parameters to be configured for a service policy:

- Policy name
- Source network name
- Destination network name
- Other policy match conditions, for example direction and source and destination ports
- Policy configured in “routed/in-network” or “bridged/” mode
- An action type called **apply\_service** is used:
  1. Example: **'apply\_service': [DomainName:ProjectName:ServiceInstanceName]**

## RELATED DOCUMENTATION

[Example: Creating an In-Network Service Chain by Using Contrail Command](#)

[Example: Creating an In-Network-NAT Service Chain by Using Contrail Command](#)

[Example: Creating a Transparent Service Chain by Using Contrail Command](#)

*ECMP Load Balancing in the Service Chain*

## Service Chaining MX Series Configuration

This topic shows how to extend service chaining to the MX Series routers.

To configure service chaining for MX Series routers, extend the virtual networks to the MX Series router and program routes so that traffic generated from a host connected to the router can be routed through the service.

1. The following configuration snippet for an MX Series router has a left virtual network called enterprise and a right virtual network called public. The configuration creates two routing instances with loopback interfaces and route targets.

```
routing-instances {
  enterprise {
    instance-type vrf;
    interface lo0.1;
    vrf-target target:100:20000;
  }
  public {
    instance-type vrf;
    interface lo0.2;
    vrf-target target:100:10000;
  }
  routing-options {
    static {
      route 0.0.0.0/0 next-hop 10.84.20.1
    }
  }
  interface xe-0/0/0.0;
}
```

2. The following configuration snippet shows the configuration for the loopback interfaces.

```
interfaces {
  lo0 {
    unit 1 {
      family inet {
        address 2.1.1.100/32;
      }
    }
    unit 2 {
      family inet {
        address 200.1.1.1/32;
      }
    }
  }
}
```

3. The following configuration snippet shows the configuration to enable BGP. The neighbor 10.84.20.39 and neighbor 10.84.20.40 are control nodes.

```
protocols {
  bgp {
    group demo_contrail {
      type internal;
      description "To Contrail Control Nodes & other MX";
      local-address 10.84.20.252;
      keep all;
      family inet-vpn {
        unicast;
      }
      neighbor 10.84.20.39;
      neighbor 10.84.20.40;
    }
  }
}
```

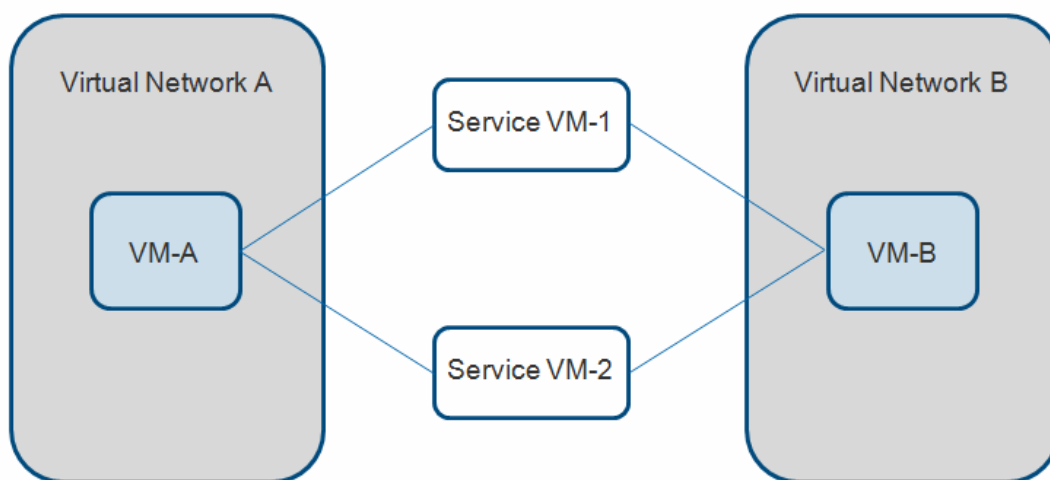
4. The final step is to add target:100:10000 to the public virtual network and target:100:20000 to the enterprise virtual network, using the Contrail Juniper Networks interface.

A full MX Series router configuration for Contrail can be seen in *Sample Network Configuration for Devices for Simple Tiered Web Application*.

## ECMP Load Balancing in the Service Chain

Traffic flowing through a service chain can be load-balanced by distributing traffic streams to multiple service virtual machines (VMs) that are running identical applications. This is illustrated in [Figure 145 on page 613](#), where the traffic streams between VM-A and VM-B are distributed between Service VM-1 and Service VM-2. If Service VM-1 goes down, then all streams that are dependent on Service VM-1 will be moved to Service VM-2.

**Figure 145: Load Balancing a Service Chain**



s041830

The following are the major features of load balancing in the service chain:

- Load balancing can be configured at every level of the service chain.
- Load balancing is supported in routed and bridged service chain modes.
- Load balancing can be used to achieve high availability—if a service VM goes down, the traffic passing through that service VM can be distributed through another service VM.
- A load balanced traffic stream always follows the same path through the chain of service VM.

### RELATED DOCUMENTATION

*Service Chaining*

## Customized Hash Field Selection for ECMP Load Balancing

### IN THIS SECTION

- [Overview: Custom Hash Feature | 614](#)
- [Using ECMP Hash Fields Selection | 616](#)
- [Sample Flows | 617](#)

### Overview: Custom Hash Feature

Starting with Contrail Release 3.0, it is possible to configure the set of fields used to hash upon during equal-cost multipath (ECMP) load balancing.

Earlier versions of Contrail had this set of fields fixed to the standard 5-tuple set of: source L3 address, destination L3 address, L4 protocol, L4 SourcePort, and L4 DestinationPort.

With the custom hash feature, users can configure an exact subset of fields to hash upon when choosing the forwarding path among a set of eligible ECMP candidates.

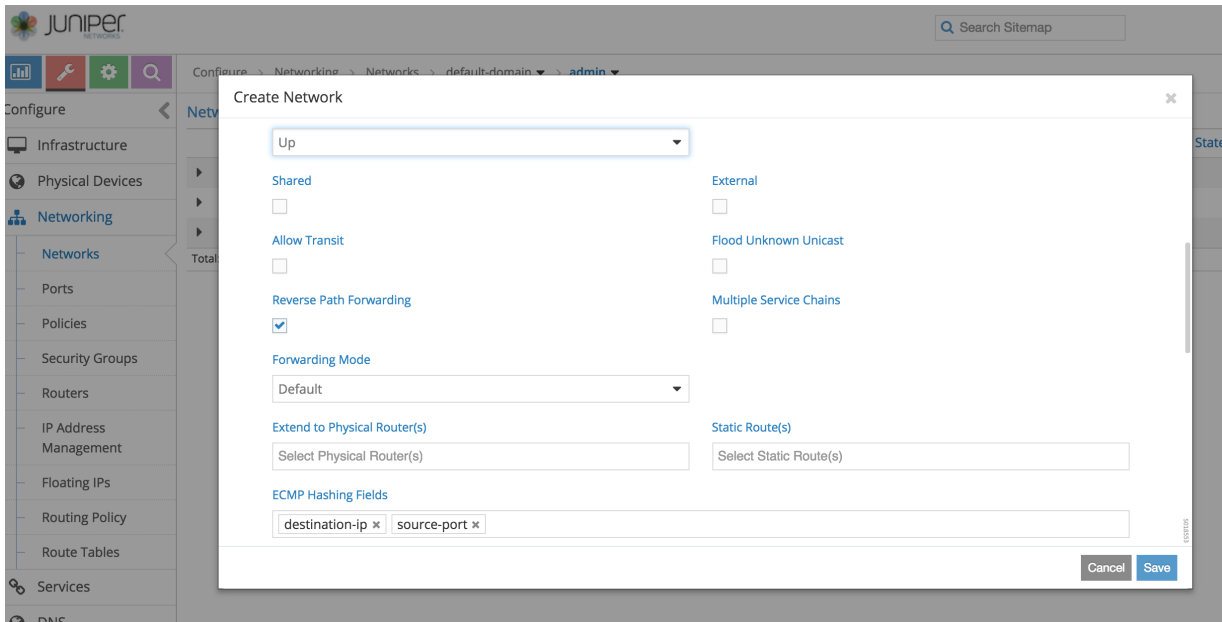
The custom hash configuration can be applied in the following ways:

- globally
- per virtual network (VN)
- per virtual network interface (VNI)

VNI configurations take precedence over VN configurations, and VN configurations take precedence over global level configuration (if present).

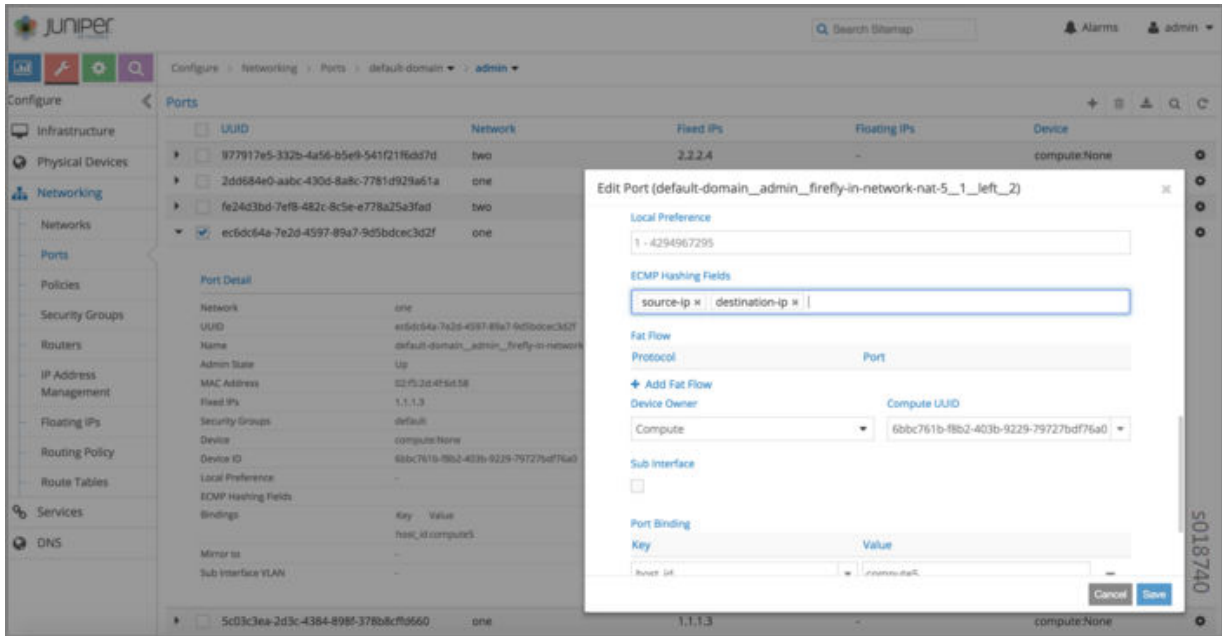
Custom hash is useful whenever packets originating from a particular source and addressed to a particular destination must go through the same set of service instances during transit. This might be required if source, destination, or transit nodes maintain a certain state based on the flow, and the state behavior could also be used for subsequent new flows, between the same pair of source and destination addresses. In such cases, subsequent flows must follow the same set of service nodes followed by the initial flow.

You can use the Contrail UI to identify specific fields in the network upon which to hash at the **Configure > Networking > Network, Create Network** window, in the **ECMP Hashing Fields** section as shown in the following figure.



If the hashing fields are configured for a virtual network, all traffic destined to that VN will be subject to the customized hash field selection during forwarding over ECMP paths by vRouters. This may not be desirable in all cases, as it could potentially skew all traffic to the destination network over a smaller set of paths across the IP fabric.

A more practical scenario is one in which flows between a source and destination must go through the same service instance in between, where one could configure customized ECMP fields for the virtual machine interface (VMI) of the service instance. Then, each service chain route originating from that VMI would get the desired ECMP field selection applied as its path attribute, and eventually get propagated to the ingress vRouter node. See the following example.



## Using ECMP Hash Fields Selection

Custom hash fields selection is most useful in scenarios where multiple ECMP paths exist for a destination. Typically, the multiple ECMP paths point to ingress service instance nodes, which could be running anywhere in the Contrail cloud.

## Configuring ECMP Hash Fields Over Service Chains

Use the following steps to create customized hash fields with ECMP over service chains.

1. Create the virtual networks needed to interconnect using service chaining, with ECMP load-balancing.
2. Create a service template and enable scaling.
3. Create a service instance, and using the service template, configure by selecting:
  - the desired number of instances for scale-out
  - the left and right virtual network to connect
  - the shared address space, to make sure that instantiated services come up with the same IP address for left and right, respectively

This configuration enables ECMP among all those service instances during forwarding.

4. Create a policy, then select the service instance previously created and apply the policy to to the desired VMLs or VNs.



5. After the service VMs are instantiated, the ports of the left and right interfaces are available for further configuration. At the Contrail UI Ports section under Networking, select the left port (VMI) of the service instance and apply the desired ECMP hash field configuration.

**NOTE:** Currently the ECMP field selection configuration for the service instance left or right interface must be applied by using the Ports (VMIs) section under Networking and explicitly configuring the ECMP fields selection for each of the instantiated service instances' VMIs. This must be done for all service interfaces of the group, to ensure the end result is as expected, because the load balance attribute of only the best path is carried over to the ingress vRouter. If the load balance attribute is not configured, it is not propagated to the ingress vRouter, even if other paths have that configuration.

When the configuration is finished, the vRouters get programmed with routing tables with the ECMP paths to the various service instances. The vRouters are also programmed with the desired ECMP hash fields to be used during load balancing of the traffic.

## Sample Flows

This section provides sample flows with and without ECMP custom hash field selection.

### Sample Traffic Flow Path Without Custom ECMP Hash Fields

The following is an example of a traffic flow path without using a customized ECMP hash fields selection configuration. The flow is configured with standard 5-tuple flow fields.

```
tcpdump -i eth0 'port 1023 and tcp[tcpflags] & (tcp-syn) != 0 and tcp[tcpflags] & (tcp-ack) == 0'
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
14:55:10.115122 IP 2.2.2.5.18337 > 2.2.2.100.1023: Flags [S], seq 2276852196, win 29200, options
[mss 1398,sackOK,TS val 25208882 ecr 0,nop,wscale 7], length 0
14:55:10.132753 IP 2.2.2.4.21193 > 2.2.2.100.1023: Flags [S], seq 4161487314, win 29200, options
[mss 1398,sackOK,TS val 25208886 ecr 0,nop,wscale 7], length 0
14:55:10.152053 IP 2.2.2.5.24230 > 2.2.2.100.1023: Flags [S], seq 2466454857, win 29200, options
[mss 1398,sackOK,TS val 25208892 ecr 0,nop,wscale 7], length 0
14:55:11.146029 IP 2.2.2.5.24230 > 2.2.2.100.1023: Flags [S], seq 2466454857, win 29200, options
[mss 1398,sackOK,TS val 25209142 ecr 0,nop,wscale 7], length 0
14:55:13.147616 IP 2.2.2.5.24230 > 2.2.2.100.1023: Flags [S], seq 2466454857, win 29200, options
[mss 1398,sackOK,TS val 25209643 ecr 0,nop,wscale 7], length 0
14:55:13.164367 IP 2.2.2.3.25582 > 2.2.2.100.1023: Flags [S], seq 2259034580, win 29200, options
[mss 1398,sackOK,TS val 25209644 ecr 0,nop,wscale 7], length 0
```

```

14:55:13.179939 IP 2.2.2.5.24895 > 2.2.2.100.1023: Flags [S], seq 2174031724, win 29200, options
[mss 1398,sackOK,TS val 25209648 ecr 0,nop,wscale 7], length 0
14:55:14.168282 IP 2.2.2.5.24895 > 2.2.2.100.1023: Flags [S], seq 2174031724, win 29200, options
[mss 1398,sackOK,TS val 25209898 ecr 0,nop,wscale 7], length 0
14:55:16.172384 IP 2.2.2.5.24895 > 2.2.2.100.1023: Flags [S], seq 2174031724, win 29200, options
[mss 1398,sackOK,TS val 25210399 ecr 0,nop,wscale 7], length 0
14:55:16.189864 IP 2.2.2.5.22952 > 2.2.2.100.1023: Flags [S], seq 3099816842, win 29200, options
[mss 1398,sackOK,TS val 25210401 ecr 0,nop,wscale 7], length 0
14:55:16.205142 IP 2.2.2.4.16487 > 2.2.2.100.1023: Flags [S], seq 3961114202, win 29200, options
[mss 1398,sackOK,TS val 25210405 ecr 0,nop,wscale 7], length 0
14:55:17.196763 IP 2.2.2.4.16487 > 2.2.2.100.1023: Flags [S], seq 3961114202, win 29200, options
[mss 1398,sackOK,TS val 25210655 ecr 0,nop,wscale 7], length 0
14:55:19.200623 IP 2.2.2.4.16487 > 2.2.2.100.1023: Flags [S], seq 3961114202, win 29200, options
[mss 1398,sackOK,TS val 25211156 ecr 0,nop,wscale 7], length 0
14:55:19.215809 IP 2.2.2.3.18914 > 2.2.2.100.1023: Flags [S], seq 3157557440, win 29200, options
[mss 1398,sackOK,TS val 25211158 ecr 0,nop,wscale 7], length 0
14:55:19.228405 IP 2.2.2.7.15569 > 2.2.2.100.1023: Flags [S], seq 3850648420, win 29200, options
[mss 1398,sackOK,TS val 25211161 ecr 0,nop,wscale 7], length 0
14:55:20.223482 IP 2.2.2.7.15569 > 2.2.2.100.1023: Flags [S], seq 3850648420, win 29200, options
[mss 1398,sackOK,TS val 25211412 ecr 0,nop,wscale 7], length 0
14:55:22.232068 IP 2.2.2.7.15569 > 2.2.2.100.1023: Flags [S], seq 3850648420, win 29200, options
[mss 1398,sackOK,TS val 25211913 ecr 0,nop,wscale 7], length 0
14:55:22.247325 IP 2.2.2.4.28388 > 2.2.2.100.1023: Flags [S], seq 3609240658, win 29200, options
[mss 1398,sackOK,TS val 25211915 ecr 0,nop,wscale 7], length 0

```

## Sample Traffic Flow Path With Custom ECMP Hash Fields

The following is an example of a traffic flow path using a customized ECMP hash fields selection configuration, for source-ip and destination-ip only.

```

tcpdump -i eth0 'port 1023 and tcp[tcpflags] & (tcp-syn) != 0 and tcp[tcpflags] & (tcp-ack) == 0'
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
15:57:18.680853 IP 2.2.2.4.21718 > 2.2.2.100.1023: Flags [S], seq 2052086108, win 29200, options
[mss 1398,sackOK,TS val 26141024 ecr 0,nop,wscale 7], length 0
15:57:18.696114 IP 2.2.2.4.13585 > 2.2.2.100.1023: Flags [S], seq 2039627277, win 29200, options
[mss 1398,sackOK,TS val 26141028 ecr 0,nop,wscale 7], length 0
15:57:18.714846 IP 2.2.2.4.16414 > 2.2.2.100.1023: Flags [S], seq 3252526560, win 29200, options
[mss 1398,sackOK,TS val 26141033 ecr 0,nop,wscale 7], length 0
15:57:18.731281 IP 2.2.2.4.32499 > 2.2.2.100.1023: Flags [S], seq 1389133175, win 29200, options
[mss 1398,sackOK,TS val 26141037 ecr 0,nop,wscale 7], length 0

```

```

15:57:18.747051 IP 2.2.2.4.6081 > 2.2.2.100.1023: Flags [S], seq 427936299, win 29200, options
[mss 1398,sackOK,TS val 26141041 ecr 0,nop,wscale 7], length 0
15:57:19.740204 IP 2.2.2.4.6081 > 2.2.2.100.1023: Flags [S], seq 427936299, win 29200, options
[mss 1398,sackOK,TS val 26141291 ecr 0,nop,wscale 7], length 0
15:57:21.743951 IP 2.2.2.4.6081 > 2.2.2.100.1023: Flags [S], seq 427936299, win 29200, options
[mss 1398,sackOK,TS val 26141792 ecr 0,nop,wscale 7], length 0
15:57:21.758532 IP 2.2.2.4.13800 > 2.2.2.100.1023: Flags [S], seq 3020971712, win 29200, options
[mss 1398,sackOK,TS val 26141794 ecr 0,nop,wscale 7], length 0
15:57:21.772646 IP 2.2.2.4.23894 > 2.2.2.100.1023: Flags [S], seq 3373734307, win 29200, options
[mss 1398,sackOK,TS val 26141797 ecr 0,nop,wscale 7], length 0
15:57:22.764469 IP 2.2.2.4.23894 > 2.2.2.100.1023: Flags [S], seq 3373734307, win 29200, options
[mss 1398,sackOK,TS val 26142047 ecr 0,nop,wscale 7], length 0
15:57:24.768511 IP 2.2.2.4.23894 > 2.2.2.100.1023: Flags [S], seq 3373734307, win 29200, options
[mss 1398,sackOK,TS val 26142548 ecr 0,nop,wscale 7], length 0
15:57:24.784119 IP 2.2.2.4.21858 > 2.2.2.100.1023: Flags [S], seq 2212369297, win 29200, options
[mss 1398,sackOK,TS val 26142550 ecr 0,nop,wscale 7], length 0
15:57:24.797149 IP 2.2.2.4.29440 > 2.2.2.100.1023: Flags [S], seq 2007897735, win 29200, options
[mss 1398,sackOK,TS val 26142554 ecr 0,nop,wscale 7], length 0
15:57:25.792816 IP 2.2.2.4.29440 > 2.2.2.100.1023: Flags [S], seq 2007897735, win 29200, options
[mss 1398,sackOK,TS val 26142804 ecr 0,nop,wscale 7], length 0
15:57:27.797538 IP 2.2.2.4.29440 > 2.2.2.100.1023: Flags [S], seq 2007897735, win 29200, options
[mss 1398,sackOK,TS val 26143305 ecr 0,nop,wscale 7], length 0
15:57:27.814002 IP 2.2.2.4.23452 > 2.2.2.100.1023: Flags [S], seq 1659332655, win 29200, options
[mss 1398,sackOK,TS val 26143307 ecr 0,nop,wscale 7], length 0

```

## Service Chain Version 2 with Port Tuple

### IN THIS SECTION

- [Overview of Port Tuple | 620](#)
- [Service Chain Version 2 Sample Workflow | 621](#)
- [Service Chain with Health Check | 623](#)

Starting with Contrail 3.0, the user can create a port-tuple object for binding service instances to ports.

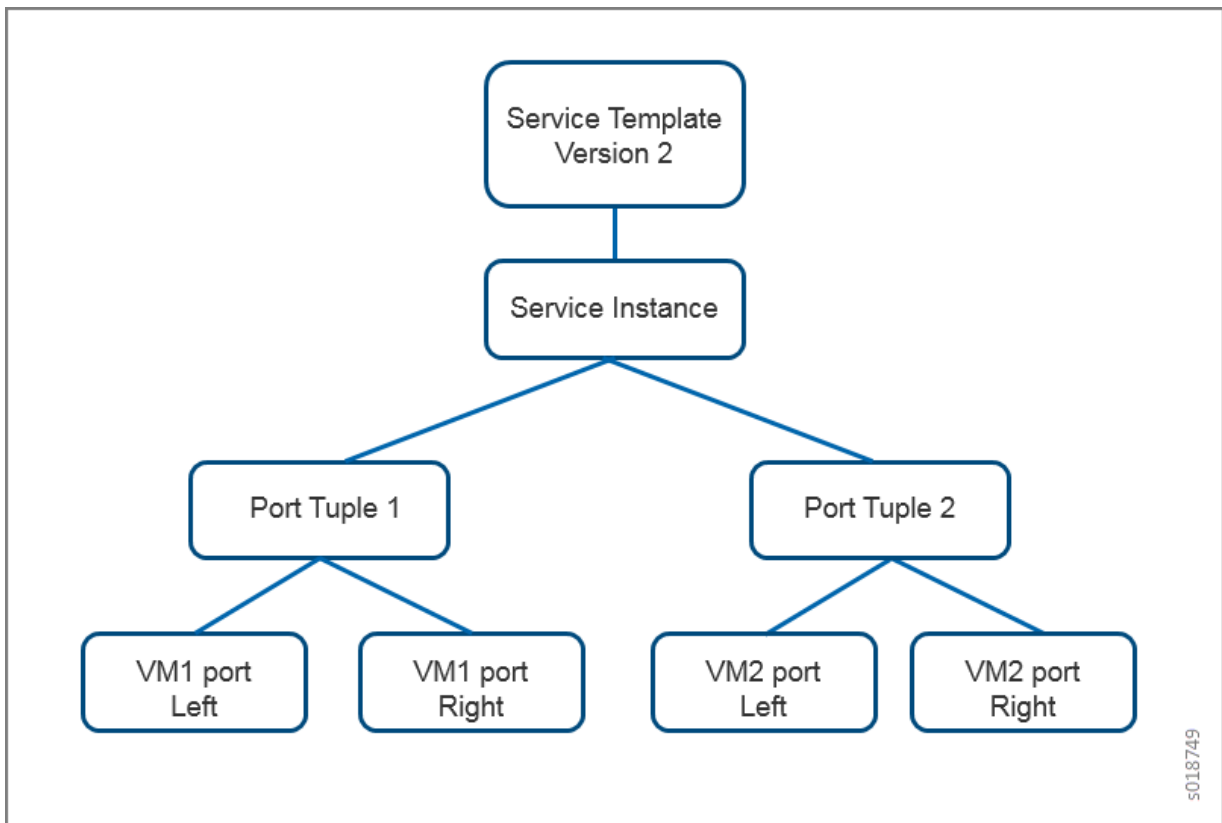
## Overview of Port Tuple

In previous versions of Contrail, when a service instance is created for a virtual machine (VM)-based service, the service monitor creates one or more VM objects and creates a port for each VM object. Each VM object is a placeholder for binding a service instance to a port. The VM object also acts as a placeholder for the instance ID when using equal-cost multipath (ECMP).

Using the VM object as a placeholder doesn't add value beyond binding information between the service instance object and the port objects. By using a port-tuple object, the service instance can be linked directly to the port objects, eliminating the need to create a VM object. This simplifies the implementation of service instance objects, and also allows integration with Heat templates.

With a port-tuple object, the user can create ports and pass the port information when creating a service instance. The ports can be created as part of a VM launch from Nova or without using a VM launch. The ports are linked to a port-tuple object that is a child of a service instance. This functionality can also be leveraged in Heat stacks. See [Figure 146 on page 620](#).

**Figure 146: Port Tuple Overview**



## Service Chain Version 2 Sample Workflow

With Contrail service templates Version 2, the user can create ports and bind them to a VM-based or container-based service instance, by means of a port-tuple object. All objects created with the Version 2 service template are visible to the Contrail Heat engine, and are managed by Heat.

The following shows the basic workflow steps for creating a port tuple and service instance that will be managed by Heat:

1. Create a service template. Select 2 in the Version field.
2. Create a service instance for the service template just created.
3. Create a port-tuple object.
4. Create ports, using Nova VM launch or without a VM launch.
5. Label each port as `left`, `right`, `mgmt`, and so on, and add the ports to the port-tuple object.

Use a unique label for each of the ports in a single port tuple. The labels `left` and `right` are used for forwarding.

6. Link the port tuple to a service instance.
7. Launch the service instance. This creates the necessary objects in the Contrail database.

**NOTE:** Port-tuple is *not* supported on transparent service instance, whether active/active, active/standby, or scale-out.

## Service Chain with Equal-Cost Multipath in Active-Active Mode

Equal-cost multipath (ECMP) can be used to distribute traffic across VMs. To support ECMP in the service chain, create multiple port tuples within the same service instance. The labels should be the same for the VM ports in each port tuple. For example, if port tuple 1 uses the labels `left` and `right`, then port tuple 2 in the same service instance should also use the labels `left` and `right` for its ports.

When there are multiple port tuples, the default mode of operation is active-active.

## Service Chain Active-Standby Mode with Allowed Address Pair

To support active-standby mode, you must configure an allowed address pair on the interfaces. The active-standby is used as the high availability mode in the allowed address pair. The allowed address pair is configured as part of the service instance for a particular VM port label. For example, if the allowed

address pair is configured in a service instance for the port with the label left, then all of the port-tuple VM ports with the label left will use the allowed address pair high availability mode.

## Allowed Address Pair

An allowed address pair extension is an OpenStack feature supported by Contrail.

By default, there is no way to specify additional MAC/IP address pairs that are allowed to pass through a port in Neutron, because ports are locked down to their MAC address and the fixed IPs associated with their port for anti-spoofing reasons. This locking can sometimes prevent protocols such as VRRP from providing a high availability failover strategy. Using the allowed address pair extension enables additional IP/MAC pairs to be allowed through ports in Neutron.

In Contrail, you can configure allowed address pairs in the service instance configuration, using **Configure > Services > Service Instances > Allowed Address Pair**, see [Figure 147 on page 622](#).

**Figure 147: Edit Service Instance, Allowed Address Pair**

The screenshot shows the 'Edit Service Instance' dialog box. The 'Allowed Address Pair' section is expanded and highlighted with a green border. It contains a table with the following data:

| Interface Type | IP        | MAC               |     |
|----------------|-----------|-------------------|-----|
| left           | 1.1.1.254 | 00:00:5e:00:01:03 | + - |
| right          | 2.2.2.254 | 00:00:5e:00:01:04 | + - |

For more information about OpenStack allowed address pairs, see [https://specs.openstack.org/openstack/neutron-specs/specs/api/allowed\\_address\\_pairs.html](https://specs.openstack.org/openstack/neutron-specs/specs/api/allowed_address_pairs.html).

## Service Chain with Static Route Table

The service chain Version 2 also supports static route tables. A static route table is configured similar to how the allowed address pair is configured, except with using the label `right`. The route table will be attached to the correct VM ports of the port tuples, based on the configuration of the port with the label `right`.

## Service Chain with Health Check

Service chain Version 2 also allows service instance health check configuration on a per interface label. This is used to monitor the health of the service.

For more information about the service instance health check, see *Health Check Object*.

### RELATED DOCUMENTATION

| [Health Check Object](#)

## Using the Contrail Heat Template

### IN THIS SECTION

- [Introduction to Heat | 623](#)
- [Heat Architecture | 624](#)
- [Support for Heat Version 2 Resources | 624](#)
- [Heat Version 2 with Service Chaining and Port Tuple Sample Workflow | 625](#)
- [Example: Creating a Service Template Using Heat | 625](#)

Heat is the orchestration engine of the OpenStack program. Heat enables launching multiple cloud applications based on templates that are comprised of text files.

### Introduction to Heat

A Heat template describes the infrastructure for a cloud application, such as networks, servers, floating IP addresses, and the like, and can be used to manage the entire life cycle of that application.

When the application infrastructure changes, the Heat templates can be modified to automatically reflect those changes. Heat can also delete all application resources if the system is finished with an application.

Heat templates can record the relationships between resources, for example, which networks are connected by means of policy enforcements, and consequently call OpenStack REST APIs that create the necessary infrastructure, in the correct order, needed to launch the application managed by the Heat template.

## Heat Architecture

Heat is implemented by means of Python applications, including the following:

- `heat-client`—The CLI tool that communicates with the `heat-api` application to run Heat APIs.
- `heat-api`—Provides an OpenStack native REST API that processes API requests by sending them to the Heat engine over remote procedure calls (RPCs).
- `heat-engine`—Responsible for orchestrating the launch of templates and providing events back to the API consumer.

## Support for Heat Version 2 Resources

Starting with Contrail Release 3.0.2, Contrail Heat resources and templates are autogenerated from the Contrail schema, using Heat Version 2 resources. Contrail Release 3.0.2 is the minimum required version for using Heat with Contrail in 3.x releases. The Contrail Heat Version 2 resources are of the following hierarchy: `OS::ContrailV2::<ResourceName>`.

The generated resources and templates are part of the Contrail Python package, and are located in the following directory in the target installation:

```
/usr/lib/python2.7/dist-packages/vnc_api/gen/heat/
```

The `heat/` directory has the following subdirectories:

- `resources/`—Contains all the resources for the `contrail-heat` plugin, which runs in the context of the Heat engine service.
- `templates/`—Contains sample templates for each resource. Each sample template presents every possible parameter in the schema. Use the sample templates as a reference when you build up more complex templates for your network design.
- `env/`—Contains the environment for input to each template.

The following contains a list of all the generated plug-in resources that are supported by `contrail-heat` in Contrail Release 3.0.2 and greater:



<https://github.com/Juniper/contrail-heat/tree/master/generated/resources>

The following contains a list of new example templates:

[https://github.com/Juniper/contrail-heat/tree/master/contrail\\_heat/new\\_templates](https://github.com/Juniper/contrail-heat/tree/master/contrail_heat/new_templates)

## Deprecation of Heat Version 1 Resources

Heat Version 1 resources within the hierarchy `OS::Contrail::<ResourceName>` are being deprecated, and you should not create new service chains using the Heat Version 1 templates.

## Heat Version 2 with Service Chaining and Port Tuple Sample Workflow

With Contrail service templates Version 2, the user can create ports and bind them to a virtual machine (VM)-based service instance, by means of a port-tuple object. All objects created with the Version 2 service template are directly visible to the Contrail Heat engine, and are directly managed by Heat.

The following shows the basic workflow steps for creating a port tuple and service instance that will be managed by Heat:

1. Create a service template. Select 2 in the Version field.
2. Create a service instance for the service template just created.
3. Create a port-tuple object.
4. Create ports, using Nova VM launch or without a VM launch.
5. Label each port as left, right, mgmt, and so on, and add the ports to the port-tuple object.

Use a unique label for each of the ports in a single port tuple. The labels named left and right are used for forwarding.

6. Link the port tuple to a service instance.
7. Launch the service instance.

## Example: Creating a Service Template Using Heat

The following is an example of how to create a service template using Heat.

1. Define a template to create the service template.

```
service_template.yaml
heat_template_version: 2013-05-23
description: >
```

```

HOT template to create a service template
parameters:
  name:
    type: string
    description: Name of service template
  mode:
    type: string
    description: service mode
  type:
    type: string
    description: service type
  image:
    type: string
    description: Name of the image
  flavor:
    type: string
    description: Flavor
  service_interface_type_list:
    type: string
    description: List of interface types
  shared_ip_list:
    type: string
    description: List of shared ip enabled--disabled
  static_routes_list:
    type: string
    description: List of static routes enabled--disabled

resources:
  service_template:
    type: OS::ContrailV2::ServiceTemplate
    properties:
      name: { get_param: name }
      service_mode: { get_param: mode }
      service_type: { get_param: type }
      image_name: { get_param: image }
      flavor: { get_param: flavor }
      service_interface_type_list: { "Fn::Split" : [ ",", Ref:
service_interface_type_list ] }
      shared_ip_list: { "Fn::Split" : [ ",", Ref: shared_ip_list ] }
      static_routes_list: { "Fn::Split" : [ ",", Ref: static_routes_list ] }
    outputs:
      service_template_fq_name:
        description: FQ name of the service template

```

```

    value: { get_attr: [ service_template, fq_name] }
}

```

2. Create an environment file to define the values to put in the variables in the template file.

```

service_template.env

parameters:

    name: contrail_svc_temp

    mode: transparent

    type: firewall

    image: cirros

    flavor: m1.tiny

    service_interface_type_list: management,left,right,other

    shared_ip_list: True,True,False,False

    static_routes_list: False,True,False,False

```

3. Create the Heat stack by launching the template and the environment file, using the following command:

```
heat stack create stack1 -f service_template.yaml -e service_template.env
```

OR use this command for recent versions of OpenStack

```
openstack stack create -e <env-file-name> -t <template-file-name> <stack-name>
```

## RELATED DOCUMENTATION

| [Service Chain Version 2 with Port Tuple](#) | 619

## Service Chain Route Reorigination

### IN THIS SECTION

- [Overview: Service Chaining in Contrail | 628](#)
- [Route Aggregation | 629](#)
- [Routing Policy | 637](#)
- [Control for Route Reorigination | 647](#)

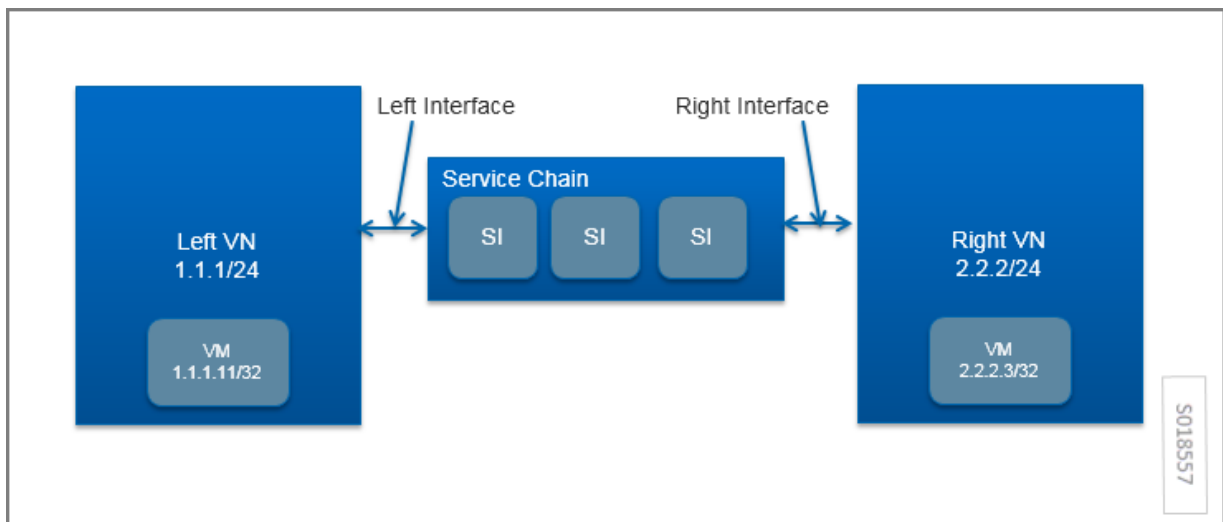
### Overview: Service Chaining in Contrail

In Contrail, the service chaining feature allows the operator to insert dynamic services to control the traffic between two virtual networks. The service chaining works on a basic rule of next-hop stitching.

In [Figure 148 on page 628](#), the service chain is inserted between the Left VN and the Right VN. The service chain contains one or more service instances to achieve a required network policy.

In the example, the route for the VM in the Right VN is added to the routing table for the Left VN, with the next hop modified to ensure that the traffic is sent by means of the left interface of the service chain. This is an example of route reorigination.

**Figure 148: Route Reorigination**



Using reorigination of routes for service chaining (for example, putting the route for the right network in the left routing table) requires the following features:

- **Route aggregation**

For scaling purposes, it is useful to publish an aggregated route as the service chain route, rather than publishing every route of each VM (/32). This reduces the memory footprint for the route table in the gateway router and also reduces route exchanges between control nodes and the gateway router. The route can be aggregated to the default route (0/0), to the VN subnet prefix, or to any arbitrary route prefix.

- **Path attribute modification for reoriginated routes**

There are cases where the `BgpPath` attribute for the service chain route needs to be modified. An example is the case of service chain failover, in which there are two service chains with identical services that are connected between the same two VNs. The operator needs to control which service chain is used for traffic between two networks, in addition to ensuring redundancy and high availability by providing failover support. Path attribute modification for reoriginated routes is implemented by means of routing policy, by providing an option to alter the MED (multi-exit discriminator) or `local-pref` of the reoriginated service chain route.

- **Control to enable and disable reorigination of the route**

In some scenarios, the operator needs a control to stop reorigination of the route as the service chain route, for example, when static routes are configured on service VM interfaces. Control to enable or disable reorigination of the route is implemented by tagging the routes with the `no-reoriginate` community. Routes with the `no-reoriginate` community tag are skipped for route reorigination.

## Route Aggregation

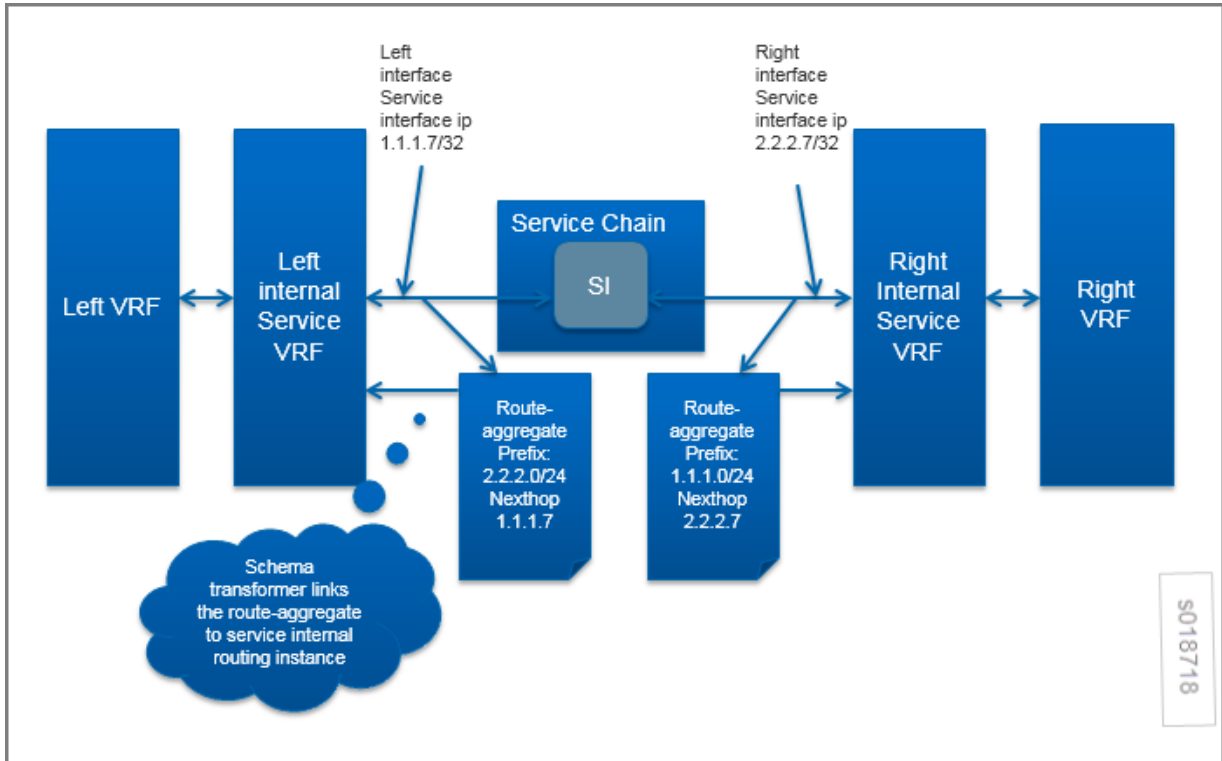
The route aggregation configuration object contains a list of prefixes to aggregate. The next-hop field in the route aggregate object contains the address of the route whose next hop is stitched as a next hop of the aggregate route.

Route aggregation is configured on the service instance. The operator can attach multiple route aggregation objects to a service instance. For example, if routes from the Right VN need to be aggregated and reoriginated in the route table of the Left VN, the route aggregate object is created with a prefix of the Right VN's subnet prefix and attached to the left interface of the service instance.

If the service chain has multiple service instances, the route aggregate object is attached to the left interface of the left-most service instance and to the right interface of the right-most service instance.

The relationships are shown in [Figure 149 on page 630](#).

Figure 149: Route Aggregate Relationships



The schema transformer sets the next-hop field of the route aggregate object to the service chain interface address. The schema transformer also links the route aggregate object to the internal routing instance created for the service instance.

Using the configuration as described, the Contrail control service reads the route aggregation object on the routing instance. When the first, more specific route or contributing route is launched (when the first VM is launched on the right VN), the aggregate route is published. Similarly, the aggregated route is deleted when the last, more specific route or contributing route is deleted (when the last VM is deleted in the right VN). The aggregated route is published when the next hop for the aggregated route gets resolved.

By default, in BGP or XMPP route exchanges, the control node will not publish contributing routes of an aggregate route.

## Schema for Route Aggregation

### Route Aggregate Object

The following is the schema for route aggregate objects. Multiple prefixes can be specified in a single route aggregate object.

```
<xsd:element name="route-aggregate" type="ifmap:IdentityType"/>
<xsd:complexType name="RouteListType">
  <xsd:element name="route" type="xsd:string" maxOccurs="unbounded"/>
</xsd:complexType>

<xsd:element name='aggregate-route-entries' type='RouteListType' />
<!--#IFMAP-SEMANTICS-IDL
  Property('aggregate-route-entries', 'route-aggregate') -->

<xsd:element name='aggregate-route-nexthop' type='xsd:string' />
<!--#IFMAP-SEMANTICS-IDL
  Property('aggregate-route-nexthop', 'route-aggregate') -->
```

### Service Instance Link to Route Aggregate Object

The following is the schema for the service instance link to route aggregation objects. The operator can link multiple route aggregate objects to a single service interface.

```
<xsd:element name="route-aggregate" type="ifmap:IdentityType"/>
<xsd:complexType name="RouteListType">
  <xsd:element name="route" type="xsd:string" maxOccurs="unbounded"/>
</xsd:complexType>

<xsd:element name='aggregate-route-entries' type='RouteListType' />
<!--#IFMAP-SEMANTICS-IDL
  Property('aggregate-route-entries', 'route-aggregate') -->

<xsd:element name='aggregate-route-nexthop' type='xsd:string' />
<!--#IFMAP-SEMANTICS-IDL
  Property('aggregate-route-nexthop', 'route-aggregate') -->

<xsd:simpleType name="ServiceInterfaceType">
```

```

    <xsd:restriction base="xsd:string">
    <xsd:pattern value="management|left|right|other[0-9]*"/>
    </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name='ServiceInterfaceTag'>
    <xsd:element name="interface-type" type="ServiceInterfaceType"/>
</xsd:complexType>

<xsd:element name="route-aggregate-service-instance" type="ServiceInterfaceTag"/>
<!--#IFMAP-SEMANTICS-IDL
    Link('route-aggregate-service-instance',
        'bgp:route-aggregate', 'service-instance', ['ref']) -->

```

## Routing Instance Link to Route Aggregate Object

The following is the schema for the routing instance link to the route aggregation object. A routing instance can be linked to multiple route aggregate objects to perform route aggregation for multiple route prefixes.

```

<xsd:element name="route-aggregate-routing-instance"/>
<!--#IFMAP-SEMANTICS-IDL
    Link('route-aggregate-routing-instance',
        'route-aggregate', 'routing-instance', ['ref']) -->

```

## Configuring and Troubleshooting Route Aggregation

### Configure Route Aggregate Object

You can use the Contrail UI, **Configure > Networking > Routing > Create > Route Aggregate** screen to name the route aggregate object and identify the routes to aggregate. See [Figure 150 on page 633](#).



Figure 150: Create Route Aggregate

The screenshot shows a 'Create Route Aggregate' dialog box. The 'Name' field is filled with 'left-to-right'. Below it, under 'Aggregate Route Entries', there is a 'Route' field containing '1.1.1.0/24'. To the right of the 'Route' field is a vertical label '5018719'. At the bottom right, there are 'Cancel' and 'Save' buttons.

### Example VNC Script to Create a Route Aggregate Object

You can use a VNC script to create a route aggregate object, as in the following example:

```
from vnc_api.vnc_api import *
vnc_lib = VncApi("admin", "<password>.", "admin")
project=vnc_lib.project_read(fq_name=["default-domain", "admin"])
route_aggregate=RouteAggregate(name="left_to_right", parent_obj=project)
route_list=RouteListType(["<ip address>"])
route_aggregate.set_aggregate_route_entries(route_list)
vnc_lib.route_aggregate_create(route_aggregate)
```

### Configuring a Service Instance

Create a service instance with the route aggregate object linked to the aggregate left network subnet prefix in the right virtual network. See the example in [Figure 151 on page 634](#).

Figure 151: Create Service Instance

si-aggregate      st-with-aggregate - [transparent (left, right)...

▼ Interface Details

Interface Type: left      Virtual Network: Auto Configured

Interface Type: right      Virtual Network: Auto Configured

▼ Advanced Options

▸ Routing Policy

▾ Route Aggregate

Interface Type: right      Route Aggregate: left-to-right

5018720

Cancel Save

### Create a Virtual Network and Network Policy

Create a left and right virtual network with the subnets 1.1.1/24 and 2.2.2/24, respectively. Create a network policy to apply a service chain between the left VN and the right VN. See the following example.

Create Policy

Policy Name: service-chain-policy

Policy Rules

| Action | Protocol | Source | Ports | Direction     | Destination | Ports | Log                      | Services                            | Mirror                   |
|--------|----------|--------|-------|---------------|-------------|-------|--------------------------|-------------------------------------|--------------------------|
| PASS   | ANY      | left   | ANY   | left to right | right       | ANY   | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> |

Service Instance: si-aggregate

+ Add Rule

5018721

Cancel Save

Attach the network policy to create the service chain between the left and right VNs. See the following example.

**Edit Network**

Name  
left

Network Policy(s)  
default-domain:admin:service-chain-policy

Subnets

| IPAM                  | CIDR       | Allocation Pools | Gateway | DNS                                 | DHCP                                |     |
|-----------------------|------------|------------------|---------|-------------------------------------|-------------------------------------|-----|
| default-network-ip... | 1.1.1.0/24 | start-end        | 1.1.1.1 | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | + - |

Host Route(s)

Route Prefix      Next Hop      +

Advanced Options

Cancel Save

5018722

### Validate the Route Aggregate Object in the API Server

Validate the route aggregate object in the API server configuration database. Verify the routing instance reference and the service instance reference for the aggregate object. The `aggregate_route_nexthop` field in the route aggregate object is initialized by the schema transformer to the service chain address. See the following example.

```

{
  - route-aggregate: {
    - fq_name: {
      "default-domain",
      "admin",
      "left-to-right"
    },
    uuid: "872b1fbd-b36c-4165-8723-7e10806d7716",
    parent_uuid: "6861d89d-a02f-4215-b329-1864084c8a75",
    aggregate_route_nexthop: "1.1.1.3",
    - routing_instance_refs: [
      - {
        - to: {
          "default-domain",
          "admin",
          "right",
          "service-ace7ae00-56e3-42d1-96ec-7fe77088d97f-default-domain_admin_si-aggregate"
        },
        href: "http://nodes27.enlab.juniper.net:8082/routing-instance/d291a95a-1a5a-4fce-94c8-4abd0968d992",
        attr: null,
        uuid: "d291a95a-1a5a-4fce-94c8-4abd0968d992"
      }
    ],
    parent_href: "http://nodes27.enlab.juniper.net:8082/project/6861d89d-a02f-4215-b329-1864084c8a75",
    parent_type: "project",
    + perms2: {(-)},
    href: "http://nodes27.enlab.juniper.net:8082/route-aggregate/872b1fbd-b36c-4165-8723-7e10806d7716",
    - id_perms: {(-)},
    - aggregate_route_entries: {
      - route: [
        "1.1.1.0/24"
      ]
    },
    display_name: "left-to-right",
    - service_instance_refs: [
      - {
        - to: {
          "default-domain",
          "admin",
          "si-aggregate"
        },
        href: "http://nodes27.enlab.juniper.net:8082/service-instance/62accf30-8cc8-4148-b7b8-975573b0d950",
        - attr: {
          interface_type: "right"
        },
        uuid: "62accf30-8cc8-4148-b7b8-975573b0d950"
      }
    ],
    name: "left-to-right"
  }
}

```

5018723

## Validate the Route Aggregate Object in the Control Node

Validate the instance configurations of the route aggregate by checking the control node introspect for the service instance internal routing instance. For example:

`http://<control-node>:8083/Snh_ShowBgpInstanceConfigReq?search_string=default-domain:admin:right:service-ace7ae00-56e3-42d1-96ec-7fe77088d97f-default-domain_admin_si-aggregate`

See the following example.

| service_chain_infos |                                |               |            |                                   | static_routes aggregate_routes |                  |
|---------------------|--------------------------------|---------------|------------|-----------------------------------|--------------------------------|------------------|
| family              | routing_instance               | chain_address | prefixes   | service_instance                  | static_routes                  | aggregate_routes |
|                     |                                |               | prefixes   |                                   | prefix                         | nexthop          |
| inet                | default-domain:admin:left:left | 1.1.1.3       | 1.1.1.0/24 | default-domain:admin:si-aggregate | 1.1.1.0/24                     | 1.1.1.3          |

5018724

To check the state of the route aggregate object on the control node, point your browser to:

[http://<control-node>:8083/Snh\\_ShowRouteAggregateReq](http://<control-node>:8083/Snh_ShowRouteAggregateReq)

See the following example.

The screenshot displays two tables side-by-side. The left table, titled 'service\_chain\_infos', has columns: family, routing\_instance, chain\_address, prefixes, and service\_instance. The right table, titled 'static\_routes aggregate\_routes', has columns: prefix and nexthop. A vertical label '5018725' is on the right side.

| service_chain_infos |                                |               |          |                                   | static_routes aggregate_routes |         |
|---------------------|--------------------------------|---------------|----------|-----------------------------------|--------------------------------|---------|
| family              | routing_instance               | chain_address | prefixes | service_instance                  | prefix                         | nexthop |
| inet                | default-domain:admin:left:left | 1.1.1.3       | 1.1.0/24 | default-domain:admin:si-aggregate | 1.1.1.0/24                     | 1.1.1.3 |

You can also check the route table for the aggregate route in the right VN BGP able. For example:

[http://<control-node>:8083/Snh\\_ShowRouteReq?x=default-domain:admin:right:right.inet.0](http://<control-node>:8083/Snh_ShowRouteReq?x=default-domain:admin:right:right.inet.0)

See the following example.

The screenshot displays a table titled 'routes' with columns: prefix, last\_modified, and paths. The 'paths' column is expanded to show a detailed table with columns: protocol, last\_modified, local\_preference, local\_as, peer\_as, peer\_router\_id, source\_as, path, next\_hop, and label. A vertical label '5018726' is on the right side.

| prefix     | last_modified               | paths   |          |               |                  |           |               |                |           |      |          |       |           |                             |     |   |   |   |   |               |    |  |
|------------|-----------------------------|---|----------|---------------|------------------|-----------|---------------|----------------|-----------|------|----------|-------|-----------|-----------------------------|-----|---|---|---|---|---------------|----|--|
| 1.1.1.0/24 | 2016-Feb-18 05:00:29.211876 | <table border="1"> <thead> <tr> <th>protocol</th> <th>last_modified</th> <th>local_preference</th> <th>local_as</th> <th>peer_as</th> <th>peer_router_id</th> <th>source_as</th> <th>path</th> <th>next_hop</th> <th>label</th> </tr> </thead> <tbody> <tr> <td>Aggregate</td> <td>2016-Feb-18 05:00:29.211876</td> <td>100</td> <td>0</td> <td>0</td> <td>-</td> <td>-</td> <td>10.204.216.23</td> <td>22</td> <td></td> </tr> </tbody> </table> | protocol | last_modified | local_preference | local_as  | peer_as       | peer_router_id | source_as | path | next_hop | label | Aggregate | 2016-Feb-18 05:00:29.211876 | 100 | 0 | 0 | - | - | 10.204.216.23 | 22 |  |
| protocol   | last_modified               | local_preference  | local_as | peer_as       | peer_router_id   | source_as | path          | next_hop       | label     |      |          |       |           |                             |     |   |   |   |   |               |    |  |
| Aggregate  | 2016-Feb-18 05:00:29.211876 | 100   | 0        | 0             | -                | -         | 10.204.216.23 | 22             |           |      |          |       |           |                             |     |   |   |   |   |               |    |  |

## Routing Policy

Contrail uses routing policy infrastructure to manipulate the route and path attribute dynamically. Contrail also supports attaching the import routing policy on the service instances.

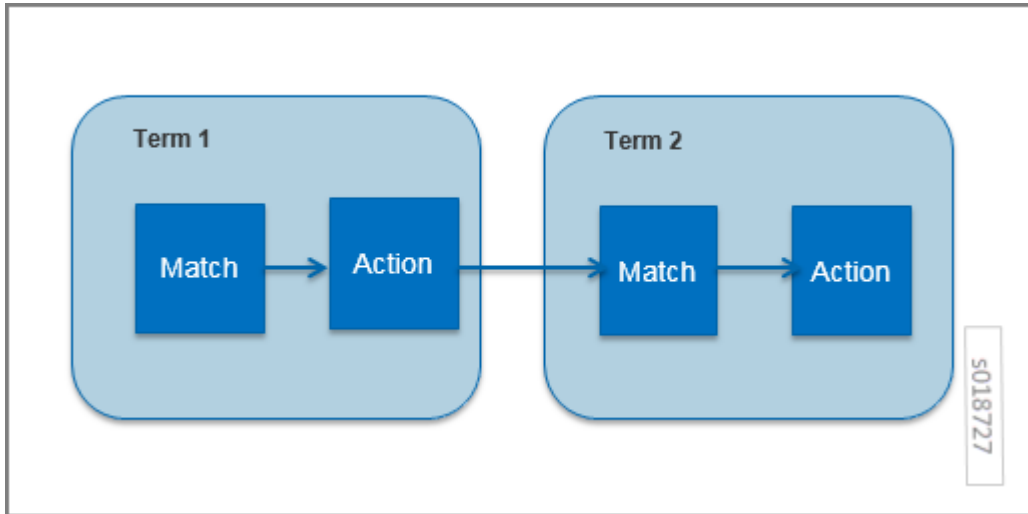
The routing policy contains list terms. A term can be a terminal rule, meaning that upon a match on the specified term, no further terms are evaluated and the route is dropped or accepted, based on the action in that term.

If the term is not a terminal rule, subsequent terms are evaluated for the given route.

The list terms are structured as in the following example.

```
Policy {
  Term-1
  Term-2
}
```

The matches and actions of the policy term lists operate similarly to the Junos language match and actions operations. A visual representation is the following.



Each term is represented as in the following:

```

from {
  match-condition-1
  match-condition-2
  ..
  ..
}
then {
  action
  update-action-1
  update-action-2
  ..
  ..
}

```

The term should not contain an any match condition, for example, an empty `from` should not be present.

If an any match condition is present, all routes are considered as matching the term.

However, the `then` condition can be empty or the action can be unspecified.

### Applying Routing Policy

The routing policy evaluation has the following key points:

- If the term of a routing policy consists of multiple match conditions, a route must satisfy all match conditions to apply the action specified in the term.
- If a term in the policy does not specify a match condition, all routes are evaluated against the match.
- If a match occurs but the policy does not specify an accept, reject, or next term action, one of the following occurs:
  - The next term, if present, is evaluated.
  - If no other terms are present, the next policy is evaluated.
  - If no other policies are present, the route is accepted. The default routing policy action is “accept”.
- If a match does not occur with a term in a policy, and subsequent terms in the same policy exist, the next term is evaluated.
- If a match does not occur with any terms in a policy, and subsequent policies exist, the next policy is evaluated.
- If a match does not occur by the end of a policy or all policies, the route is accepted.

A routing policy can consist of multiple terms. Each term consists of match conditions and actions to apply to matching routes.

Each route is evaluated against the policy as follows:

1. The route is evaluated against the first term. If it matches, the specified action is taken. If the action is to accept or reject the route, that action is taken and the evaluation of the route ends. If the next term action is specified or if no action is specified, or if the route does not match, the evaluation continues as described above to subsequent terms.
2. Upon hitting the last non-terminal term of the given routing policy, the route is evaluated against the next policy, if present, in the same manner as described in step 1.

### **Match Condition: From**

The match condition `from` contains a list of match conditions to be satisfied for applying the action specified in the term. It is possible that the term doesn't have any match condition. This indicates that all routes match this term and action is applied according to the action specified in the term.

The following table describes the match conditions supported by Contrail.

| Match Condition | User Input                                     | Description  |
|-----------------|--|--|
| Prefix          | List of prefixes to match                      | <p>Each prefix in the list is represented as prefix and match type, where the prefix match type can be:</p> <ul style="list-style-type: none"> <li>• exact</li> <li>• orlonger</li> <li>• longer</li> </ul> <p>Example: 1.1.0.0/16 orlonger</p> <p>A route matches this condition if its prefix matches any of the prefixes in the list.</p> |
| Community       | Community string to match                      | <p>Represented as either a well-known community string with no export or no reoriginate, or a string representation of a community (64512:11).</p>   |
| Protocol        | Array of path source or path protocol to match | <p>BGP   XMPP   StaticRoute   ServiceChain   Aggregate. A path is considered as matching this condition if the path protocol is one of protocols in the list.</p>  |

## Routing Policy Action and Update Action

The policy action contains two parts, action and update action.

The following table describes action as supported by Contrail.

| Action | Terminal? | Description  |
|--------|-----------|--|
| Reject | Yes       | <p>Reject the route that matches this term. No more terms are evaluated after hitting this term.</p>   |
| Accept | Yes       | <p>Accept the route that matches this term. No more terms are evaluated after hitting this term. The route is updated using the update specified in the policy action.</p> |



*(Continued)*

| Action    | Terminal? | Description   |
|-----------|-----------|---|
| Next Term | No        | This is the default action taken upon matching the policy term. The route is updated according to the update specified in the policy action. Next terms present in the routing policy are processed on the route. If there are no more terms in the policy, the next routing policy is processed, if present. |

The update action section specifies the route modification to be performed on the matching route.

The following table describes update action as supported by Contrail.

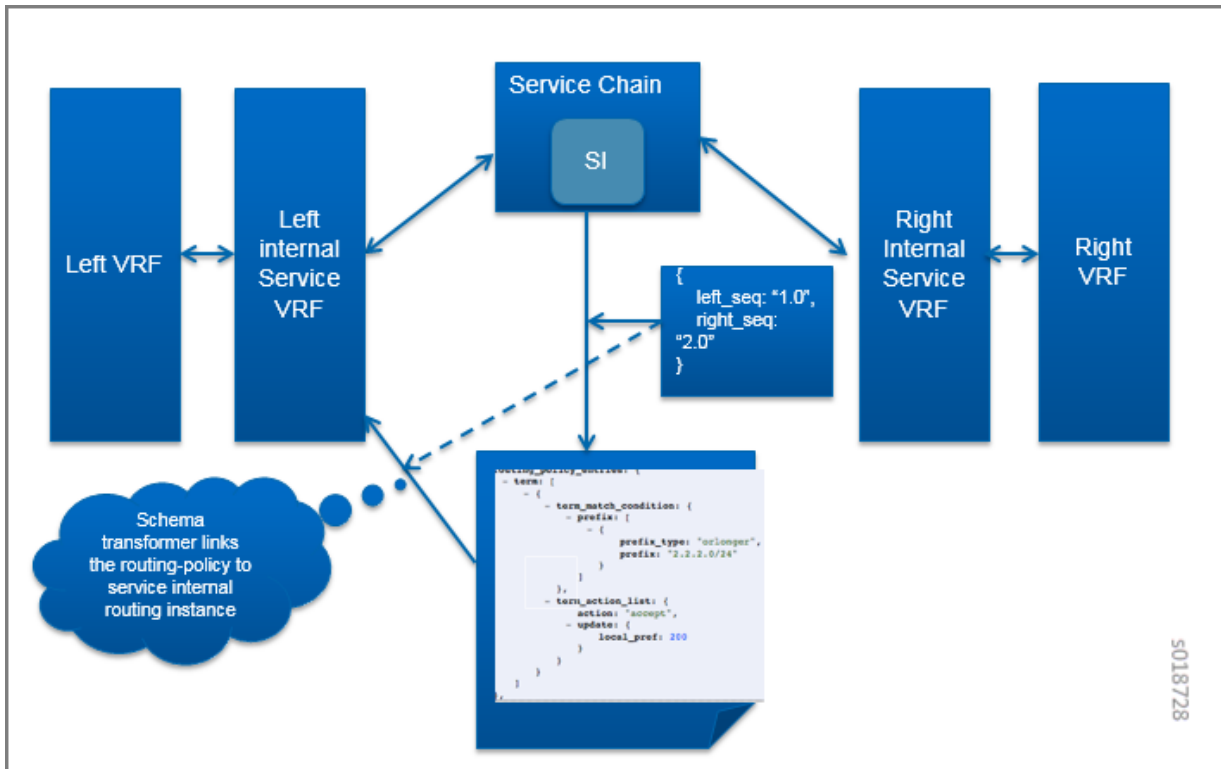
| Update Action | User Input                           | Description  |
|---------------|--------------------------------------|--|
| community     | List of community                    | As part of the policy update, the following actions can be taken for community: <ul style="list-style-type: none"> <li>• Add a list of community to the existing community.</li> <li>• Set a list of community.</li> <li>• Remove a list of community (if present) from the existing community.</li> </ul> |
| MED           | Update the MED of the BgpPath        | Unsigned integer representing the MED  |
| local-pref    | Update the local-pref of the BgpPath | Unsigned integer representing local-pref   |

## Routing Policy Configuration

Routing policy is configured on the service instance. Multiple routing policies can be attached to a single service instance interface.

When the policy is applied on the left interface, the policy is evaluated for all the routes that are reoriginated in the left VN for routes belonging to the right VN. Similarly, the routing policy attached to the right interface influences the route reorigination in the right VN, for routes belonging to the left VN.

The following figure illustrates a routing policy configuration.



The policy sequence number specified in the routing policy link data determines the order in which the routing policy is evaluated. The routing policy link data on the service instance also specifies whether the policy needs to be applied to the left service interface, to the right service interface, or to both interfaces.

It is possible to attach the same routing policy to both the left and right interfaces for a service instance, in a different order of policy evaluation. Consequently, the routing policy link data contains the sequence number for policy evaluation separately for the left and right interfaces.

The schema transformer links the routing policy object to the internal routing instance created for the service instance. The transformer also copies the routing policy link data to ensure the same policy order.

## Configuring and Troubleshooting Routing Policy

This section shows how to create a routing policy for service chains and how to validate the policy.

## Create Routing Policy

First, create the routing policy, **Configure > Networking > Routing > Create > Routing Policy**. See the following example.

**Create Routing Policy**

Name  
failover

Term(s)  
from: { prefix 2.2.2.0/24 orlonger } then: { local-preference 200 }

From  
prefix 2.2.2.0/24 orlonger

Then  
local-preference 200

Cancel Save

s018729

## Configure Service Instance

Create a service instance and attach the routing policy to both the left and right interfaces. The order of the policy is calculated by the UI, based on the order of the policy specified in the list.

**Create Service Instance**

ha-chain st-with-policy - [transparent (left, right)] - v1

Interface Details

Interface Type left Virtual Network Auto Configured

Interface Type right Virtual Network Auto Configured

Advanced Options

Routing Policy

| Interface Type | Routing Policy |
|----------------|----------------|
| left           | failover       |
| right          | failover       |

Cancel Save

s018730

## Configure the Network Policy for the Service Chain

At **Edit Policy**, create a policy for the service chain, see the following example.

## Using a VNC Script to Create Routing Policy

The following example shows use of a VNC API script to create a routing policy.

```

from vnc_api.vnc_api import *
vnc_lib = VncApi("admin", "<password>", "admin")
project=vnc_lib.project_read(fq_name=["default-domain", "admin"])
routing_policy=RoutingPolicy(name="vnc_3", parent_obj=project)
policy_term=PolicyTermType()
policy_statement=PolicyStatementType()

match_condition=TermMatchConditionType(protocol=["bgp"], community="22:33")
prefix_match=PrefixMatchType(prefix="1.1.1.0/24", prefix_type="orlonger")
match_condition.set_prefix([prefix_match])

term_action=TermActionListType(action="accept")
action_update=ActionUpdateType(local_pref=101, med=10)
add_community=ActionCommunityType()
comm_list=CommunityListType(["11:22"])
add_community.set_add(comm_list)
action_update.set_community(add_community)
term_action.set_update(action_update)

policy_term.set_term_action_list(term_action)
policy_term.set_term_match_condition(match_condition)

policy_statement.add_term(policy_term)

```

```
routing_policy.set_routing_policy_entries(policy_statement)
vnc_lib.routing_policy_create(routing_policy)
```

## Verify Routing Policy in API Server

You can verify the service instance references and the routing instance references for the routing policy by looking in the API server configuration database. See the following example.

```
- routing_policy_entries: {
  - term: {
    - {
      - term_match_condition: {
        - prefix: {
          - {
            prefix_type: "orlonger",
            prefix: "2.2.2.0/24"
          }
        }
      },
      - term_action_list: {
        action: "accept",
        - update: {
          local_pref: 200
        }
      }
    }
  }
},
+ id_perms: (-),
- routing_instance_refs: [
  - {
    - to: [
      "default-domain",
      "admin",
      "right",
      "service-ace7ae00-56e3-42d1-96ec-7fe77088d97f-default-domain_admin_ha-chain"
    ],
    href: "http://nodes27.enlab.1uniper.net:8082/routing-instance/32b7eed4-57ce-4c44-bbb0-513f78db6068",
    - attr: {
      sequence: "1"
    },
    uuid: "32b7eed4-57ce-4c44-bbb0-513f78db6068"
  },
  - {
    - to: [
      "default-domain",
      "admin",
      "left",
      "service-ace7ae00-56e3-42d1-96ec-7fe77088d97f-default-domain_admin_ha-chain"
    ],
    href: "http://nodes27.enlab.1uniper.net:8082/routing-instance/6ad868d1-a412-4765-b8c4-f93ec5d9f4b2",
    - attr: {
      sequence: "1"
    },
    uuid: "6ad868d1-a412-4765-b8c4-f93ec5d9f4b2"
  }
],
- service_instance_refs: [
  - {
    - to: [
      "default-domain",
      "admin",
      "ha-chain"
    ],
    href: "http://nodes27.enlab.1uniper.net:8082/service-instance/983bb90b-b3f4-4d6c-be54-33a474eee7de",
    - attr: {
      left_sequence: "1",
      right_sequence: "1"
    },
    uuid: "983bb90b-b3f4-4d6c-be54-33a474eee7de"
  }
],
name: "failover"
```

s018732

## Verify Routing Policy in the Control Node

You can verify the routing policy in the control node.

Point your browser to:

[http://<control-node>:8083/Snh\\_ShowRoutingPolicyReq?search\\_string=failover](http://<control-node>:8083/Snh_ShowRoutingPolicyReq?search_string=failover)

See the following example.

| routing_policies                                      |  |           |  |   |         |                          |      |  |         |                                |   |         |                          |       |
|---|--|-----------|--|---|---------|--------------------------|------|--|---------|--------------------------------|---|---------|--------------------------|-------|
| name  | generation   | ref_count | terms  | deleted   |         |                          |      |  |         |                                |   |         |                          |       |
| default-domain:admin:failover                         | 0  | 2         | <table border="1"> <thead> <tr> <th>terminal</th> <th>matches</th> <th>actions</th> </tr> </thead> <tbody> <tr> <td>true</td> <td> <table border="1"> <thead> <tr> <th>matches</th> </tr> </thead> <tbody> <tr> <td>prefix [ 2.2.2.0/24 orlonger ]</td> </tr> </tbody> </table> </td> <td> <table border="1"> <thead> <tr> <th>actions</th> </tr> </thead> <tbody> <tr> <td>accept<br/>local-pref 200</td> </tr> </tbody> </table> </td> </tr> </tbody> </table> | terminal  | matches | actions                  | true | <table border="1"> <thead> <tr> <th>matches</th> </tr> </thead> <tbody> <tr> <td>prefix [ 2.2.2.0/24 orlonger ]</td> </tr> </tbody> </table> | matches | prefix [ 2.2.2.0/24 orlonger ] | <table border="1"> <thead> <tr> <th>actions</th> </tr> </thead> <tbody> <tr> <td>accept<br/>local-pref 200</td> </tr> </tbody> </table> | actions | accept<br>local-pref 200 | false |
| terminal  | matches  | actions   |  |   |         |                          |      |  |         |                                |   |         |                          |       |
| true  | <table border="1"> <thead> <tr> <th>matches</th> </tr> </thead> <tbody> <tr> <td>prefix [ 2.2.2.0/24 orlonger ]</td> </tr> </tbody> </table> | matches   | prefix [ 2.2.2.0/24 orlonger ]   | <table border="1"> <thead> <tr> <th>actions</th> </tr> </thead> <tbody> <tr> <td>accept<br/>local-pref 200</td> </tr> </tbody> </table> | actions | accept<br>local-pref 200 |      |  |         |                                |   |         |                          |       |
| matches   |  |           |  |   |         |                          |      |  |         |                                |   |         |                          |       |
| prefix [ 2.2.2.0/24 orlonger ]                        |  |           |  |   |         |                          |      |  |         |                                |   |         |                          |       |
| actions   |  |           |  |   |         |                          |      |  |         |                                |   |         |                          |       |
| accept<br>local-pref 200                              |  |           |  |   |         |                          |      |  |         |                                |   |         |                          |       |
| default-domain:default-project:default-routing-policy | 0  | 0         | terms  | false   |         |                          |      |  |         |                                |   |         |                          |       |

5018745

## Verify Routing Policy Configuration in the Control Node

You can verify the routing policy configuration in the control node.

Point your browser to:

[http://<control-node>:8083/Snh\\_ShowBgpRoutingPolicyConfigReq?search\\_string=failover](http://<control-node>:8083/Snh_ShowBgpRoutingPolicyConfigReq?search_string=failover)

See the following example.

| ShowBgpRoutingPolicyConfigResp                   |   |       |        |  |   |
|--|---|-------|--------|--|---|
| routing_policies                                 |   |       |        |  |   |
| name   | terms   |       |        |  |   |
| default-domain:admin:failover                    | <table border="1"> <thead> <tr> <th>match</th> <th>action</th> </tr> </thead> <tbody> <tr> <td> <pre>from {   prefix 2.2.2.0/24 orlonger }</pre> </td> <td> <pre>then {   local-preference 200   accept }</pre> </td> </tr> </tbody> </table> | match | action | <pre>from {   prefix 2.2.2.0/24 orlonger }</pre> | <pre>then {   local-preference 200   accept }</pre> |
| match  | action  |       |        |  |   |
| <pre>from {   prefix 2.2.2.0/24 orlonger }</pre> | <pre>then {   local-preference 200   accept }</pre>   |       |        |  |   |

5018733

## Verify Routing Policy Configuration on the Routing Instance

You can verify the routing policy configuration on the internal routing instance.

Point your browser to:

[http://<control-node>:8083/Snh\\_ShowBgpInstanceConfigReq?search\\_string=<name-of-internal-vrf>](http://<control-node>:8083/Snh_ShowBgpInstanceConfigReq?search_string=<name-of-internal-vrf>)

See the following example.

| service_chain_info |                                  |               |            |                               | static_routes |                  | aggregate_routes |                               | routing_policies |         |
|--------------------|----------------------------------|---------------|------------|-------------------------------|---------------|------------------|------------------|-------------------------------|------------------|---------|
| 2                  |                                  |               |            |                               |               |                  |                  |                               |                  |         |
| family             | routing_instance                 | chain_address | prefixes   | service_instance              | static_routes | aggregate_routes | routing_policies | policy_name                   | sequence         |         |
| inet               | default-domain:admin:right:right | 1.1.1.6       | prefixes   | default-domain:admin:ha-chain |               |                  |                  | default-domain:admin:failover | 1                | 5018734 |
|                    |                                  |               | 2.2.0.0/24 |                               |               |                  |                  |                               |                  |         |

You can also verify the routing policy on the routing instance operational object.

Point your browser to:

[http://<control-node>:8083/Snh\\_ShowRoutingInstanceReq?x=<name-of-internal-vrf>](http://<control-node>:8083/Snh_ShowRoutingInstanceReq?x=<name-of-internal-vrf>)

See the following example.

| routing_policies              |            |
|-------------------------------|------------|
| routing_policies              |            |
| policy_name                   | generation |
| default-domain:admin:failover | 0          |

5018735

## Control for Route Reorigination

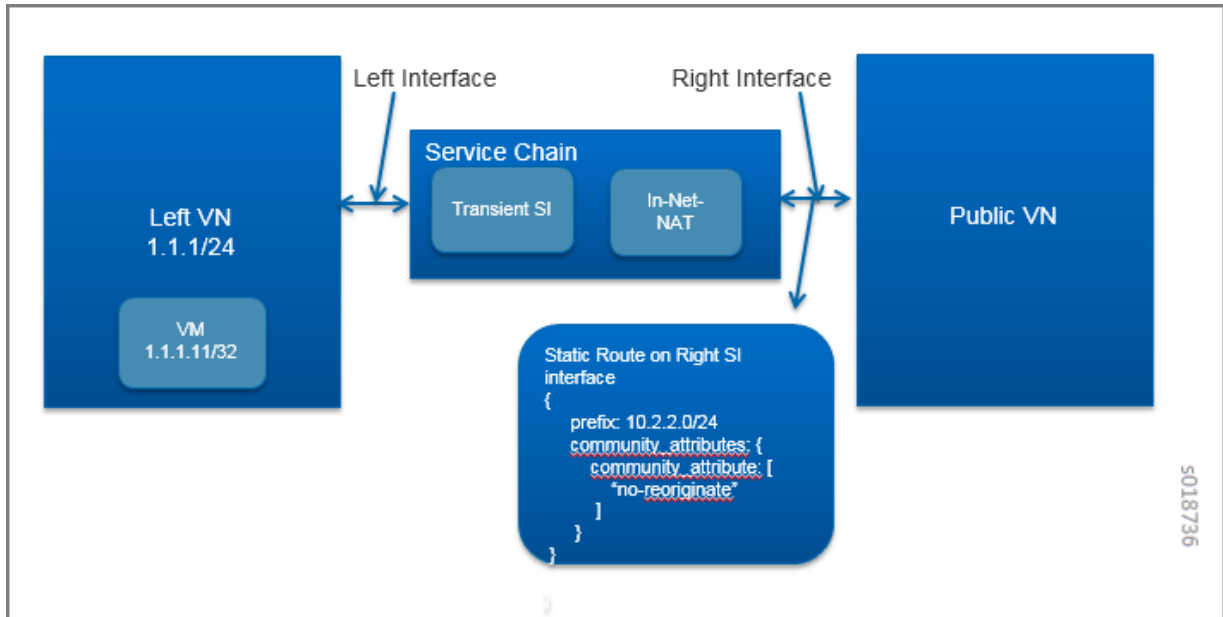
The ability to prevent reorigination of interface static routes is typically required when routes are configured on an interface that belongs to a service VM.

As an example, the following image shows a service chain that has multiple service instances, with an `in-net-nat` service instance as the last service VM, also with the right VN as the public VN.

The last service instance performs NAT by using a NAT pool. The right interface of the service VM must be configured with an interface static route for the NAT pool so that the destination in the right VN knows how to reach addresses in the NAT pool. However, the NAT pool prefix should not be reoriginated into the left VN.

To prevent route reorigination, the interface static route is tagged with a well-known BGP community called `no-reoriginate`.

When the control node is reoriginating the route, it skips the routes that are tagged with the BGP community.



### Configuring and Troubleshooting Reorigination Control

The community attribute on the static routes for the interface static route of the service instance is specified during creation of the service instance. See the following example.



Create Service Instance
✕

Name

Service Template

Interface Type

Virtual Network

Interface Type

Virtual Network

+ Add Static Routes

| Prefix                                   | Next Hop                                 | Community                                   |
|--|--|---|
| <input type="text" value="10.2.2.0/24"/> | <input type="text" value="Interface 2"/> | <input type="text" value="no-reoriginate"/> |

S018737

Routing Policy

Interface Type

Routing Policy

Cancel Save

Use the following example to verify that the service instance configuration object in the API server has the correct community set for the static route. See the following example.

```

{
  - service-instance: {
    + virtual_machine_back_refs: [...],
    + fq_name: [...],
    uuid: "ace1e71f-f828-43de-a493-b193bdb73ded",
    parent_type: "project",
    parent_uuid: "634f90d9-da62-4c2f-a238-7cc1c1a055a5",
    parent_href: "http://nodeq2:8082/project/634f90d9-da62-4c2f-a2
  - service_instance_properties: {
    right_virtual_network: "default-domain:admin:twig",
    - interface_list: [
      - {
        virtual_network: "default-domain:admin:fifo"
      },
      - {
        virtual_network: "default-domain:admin:twig",
        - static_routes: {
          - route: [
            - {
              prefix: "10.2.2.0/24",
              next_hop: null,
              - community_attributes: {
                - community_attribute: [
                  "no-reoriginate"
                ],
              },
              next_hop_type: null
            }
          ]
        }
      }
    ]
  },
  left_virtual_network: "default-domain:admin:fifo",
  - scale_out: {
    max_instances: 1
  }
},
}

```

S018738

## Service Instance Health Checks

### IN THIS SECTION

- [Health Check Object | 650](#)
- [Bidirectional Forwarding and Detection Health Check over Virtual Machine Interfaces | 655](#)
- [Bidirectional Forwarding and Detection Health Check for BGPaaS | 655](#)
- [Health Check of Transparent Service Chain | 656](#)
- [Service Instance Fate Sharing | 656](#)

In Contrail Release 3.0 and greater, a service instance health check can be used to determine the liveness of a service provided by a virtual machine (VM).

### Health Check Object

#### IN THIS SECTION

- [Health Check Overview | 650](#)
- [Health Check Object Configuration | 651](#)
- [Creating a Health Check with the Contrail User Interface | 652](#)
- [Using the Health Check | 654](#)
- [Health Check Process | 654](#)

### Health Check Overview

The service instance health check is used to determine the liveness of a service provided by a VM, checking whether the service is operationally up or down. The vRouter agent uses ping and an HTTP URL to the link-local address to check the liveness of the interface.

If the health check determines that a service is no longer operational, it removes the routes for the VM, thereby disabling packet forwarding to the VM.

The service instance health check is used with service template version 2.

## Health Check Object Configuration

Table 46 on page 651 shows the configurable properties of the health check object.

**Table 46: Health Check Configurable Parameters**

| Field               | Description   |
|---------------------|---|
| - enabled           | Indicates that health check is enabled. The default is False.   |
| - health-check-type | Indicates the health check type: link-local, end-to-end, bgp-as-a-service, and so on.. The default is link-local.     |
| - monitor-type      | The protocol type to be used: PING or HTTP.   |
| - delay             | The delay, in seconds, to repeat the health check.  |
| - timeout           | The number of seconds to wait for a response.   |
| - max-retries       | The number of retries to attempt before declaring an instance health down.  |
| - http-method       | When the monitor protocol is HTTP, the type of HTTP method used, such as GET, PUT, POST, and so on.                   |
| - url-path          | When the monitor protocol is HTTP, the URL to be used. For all other cases, such as ICMP, the destination IP address. |
| - expected-codes    | When the monitor protocol is HTTP, the expected return code for HTTP operations.                                      |

## Health Check Modes

The following modes are supported for the service instance health check:

- **link-local**—A local check for the service VM on the vRouter where the VM is running. In this case, the source IP of the packet is the service chain IP.

- end-to-end—A remote address or URL is provided for a service health check through a chain of services. The destination of the health check probe is allowed to be outside the service instance. However, the health check probe must be reachable through the interface of the service instance where the health check is attached. The end-to-end health check probe is transmitted all the way to the actual destination outside the service instance. The response to the health check probe is received and processed by the service health check to evaluate the status.

Restrictions include:

- This check is applicable for a chain where the services are not scaled out.
- When this mode is configured, a new health check IP is allocated and used as the source IP of the packet.
- The health check IP is allocated per `virtual-machine-interface` of the service VM where the health check is attached.
- The agent relies on the `service-health-check-ip` flag to use as the source IP.

**NOTE:** In versions prior to Contrail 4.1, end-to-end health check is not supported on a transparent service chain. However, a link-local health check is possible on a transparent service instance if the corresponding service instance interface is configured with its IP address. Contrail 4.1 supports a segment-based health check for transparent service chain.

### Creating a Health Check with the Contrail User Interface

To create a health check with the Contrail Web UI:

1. Navigate to **Configure > Services > Health Check Service**, and click to open the **Create** screen. See [Figure 152 on page 653](#).

Figure 152: Create Health Check Screen

The screenshot shows a 'Create' dialog box with the following fields and values:

- Name:** ext\_hc\_service
- Protocol:** PING
- Monitor Target:** 8.8.8.8
- Delay (secs):** 3
- Timeout (secs):** 5
- Retries:** 2
- Health Check Type:** End-To-End

Buttons: Cancel, Save

Vertical ID: 5018766

- Complete the fields to define the permissions for the health check, see [Table 47 on page 653](#).

Table 47: Create Health Check Fields

| Field          | Description  |
|----------------|--|
| Name           | Enter a name for the health check service you are creating.                                |
| Protocol       | Select from the list the protocol to use for the health check, PING, HTTP, BFD, and so on. |
| Monitor Target | Select from the list the address of the target to be monitored by the health check.        |
| Delay (secs)   | The delay, in seconds, to repeat the health check.   |
| Timeout (secs) | The number of seconds to wait for a response.  |

**Table 47: Create Health Check Fields (Continued)**

| Field             | Description   |
|-------------------|---|
| Retries           | The number of retries to attempt before declaring an instance health down.  |
| Health Check Type | Select from the list the type of health check—link-local, end-to-end, segment-based, bgp-as-a-service, and so on. |

### Using the Health Check

A REST API can be used to create a health check object and define its associated properties, then a link is added to the VM interface.

The health check object can be linked to multiple VM interfaces. Additionally, a VM interface can be associated with multiple health check objects. The following is an example:

```
HealthCheckObject 1 ----- VirtualMachineInterface 1 -----
HealthCheckObject 2
  |
  |
VirtualMachineInterface 2
```

### Health Check Process

The Contrail vRouter agent is responsible for providing the health check service. The agent spawns a Python script to monitor the status of a service hosted on a VM on the same compute node, and the script updates the status to the vRouter agent.

The vRouter agent acts on the status provided by the script to withdraw or restore the exported interface routes. It is also responsible for providing a link-local metadata IP for allowing the script to communicate with the destination IP from the underlay network, using appropriate NAT translations. In a running system, this information is displayed in the vRouter agent introspect at:

```
http://<compute-node-ip>:8085/Snh_HealthCheckSandeshReq?uuid=
```

**NOTE:** Running health check creates flow entries to perform translation from underlay to overlay. Consequently, in a heavily loaded environment with a full flow table, it is possible to observe false failures.

## Bidirectional Forwarding and Detection Health Check over Virtual Machine Interfaces

Contrail Networking Release 4.1 and later support for BFD-based health checks for VMIs.

Health check for VMIs is already supported as poll-based checks with ping and curl commands. When enabled, these health checks run periodically, once every few seconds. Consequently, failure detection times can be quite large, always in seconds.

Health checks based on the BFD protocol provide failure detection and recovery in sub-second intervals, because applications are notified immediately upon BFD session state changes.

If BFD-based health check is configured, whenever a BFD session status is detected as Up or Down by the health-checker, corresponding logs are generated.

Logging is enabled in the `contrail-vrouter-agent.conf` file with the log severity level `SYS_NOTICE`.

You can view the log file in the location `/var/log/contrail/contrail-vrouter-agent.log`

### Snippet of sample log message related to BFD session events

```
2019-02-26 Tue 14:38:49:417.479 SYS_NOTICE BFD session Down interface: test-bfd-hc-vmi.st2
vrf: default-domain:admin:VN.hc.st2:VN.hc.st2
2019-02-26 Tue 14:38:49:479.733 PST SYS_NOTICE BFD session Up interface: test-bfd-hc-vmi.st2
vrf: default-domain:admin:VN.hc.st2:VN.hc.st2
```

## Bidirectional Forwarding and Detection Health Check for BGPaaS

Contrail Release 4.1 adds support for BFD-based health check for BGP as a Service (BGPaaS) sessions.

This health check should not be confused with the BFD-based health check over VMIs feature, also introduced in Release 4.1. The BFD-based health check for VMIs cannot be used for a BGPaaS session, because the session shares a tenant destination address over a set of VMIs, with only one VMI active at any given time.

When the BFD-based health check for BGP as a Service (BGPaaS) is configured, any time a BFD-for-BGP session is detected as down by the health-checker, corresponding logs and alarms are generated.

To enable this health check, configure the `ServiceHealthCheckType` property and associate it with a `bgp-as-a-service` configuration object. This can also be accomplished in the Contrail WebUI.

## Health Check of Transparent Service Chain

Contrail 4.1 enhances service chain redundancy by implementing an end-to-end health check for the transparent service chain. The service health check monitors the status of the service chain and if there is a failure, the control node no longer considers the service chain as a valid next hop, triggering traffic failover.

A segment-based health check is used to verify the health of a single instance in a transparent service chain. The user creates a service-health-check object, with type `segment-based`, and attaches it to either the left or right interface of the service instance. The service health check packet is injected to the interface to which it is attached. When the packet comes out of the other interface, a reply packet is injected on that interface. If health check requests fail after 30-second retries, the service instance is considered unhealthy and the service VLAN routes of the left and right interfaces are removed. When the agent receives health check replies successfully, it adds the retracted routes back onto both interfaces, which triggers the control node to start reoriginating routes to other service instances on that service chain.

For more information, see [https://github.com/Juniper/contrail-specs/blob/master/transparent\\_sc\\_health\\_check.md](https://github.com/Juniper/contrail-specs/blob/master/transparent_sc_health_check.md)

## Service Instance Fate Sharing

A service chain contains multiple service instances (SI) and the failure of a single SI can cause a traffic black hole. In Contrail Release 4.1 and earlier, when an SI fails, the service chain continues to forward packets and routes reoriginate on both sides of the service chain. The packets are dropped in the SI or by the vRouter causing a black hole.

Starting in Contrail Release 4.1, **segment-based** health check type is used to verify the health of a SI in a service chain. To identify a failure of an SI, segment-based health check is configured either on the egress or ingress interface of the SI. When SI health check fails, the vRouter agent drops an SI route or a connected route. A connected route is also dropped if the vRouter agent restarts due to a software failure, when a compute node reboots, or when long-lived graceful restart (LLGR) is not enabled. You can detect an SI failure by keeping track of corresponding connected routes of the service chain address.

**NOTE:** When an SI is scaled out, the connected route for an SI interface goes down only when all associated VMs have failed.

The control node uses the `service-chain-id` in `ServiceChainInfo` to link all SIs in a service chain. When the control node detects that any SI of the same `service-chain-id` is down, it stops reoriginating routes in



egress and ingress directions for all SIs. The control node reoriginates routes only when the connected routes of all the SIs are up.

# Examples: Configuring Service Chaining

## IN THIS CHAPTER

- [Example: Creating an In-Network Service Chain | 658](#)
- [Example: Creating an In-Network-NAT Service Chain | 672](#)
- [Example: Creating a Transparent Service Chain | 686](#)

## Example: Creating an In-Network Service Chain

### IN THIS SECTION

- [Hardware and Software Requirements | 658](#)
- [Overview | 659](#)
- [Configuration | 659](#)

This example provides instructions to create an in-network service chain by using the Contrail user interface.

### Hardware and Software Requirements

The following are the minimum requirements needed:

#### Hardware

- Processor: 4 core x86
- Memory: 32GB RAM
- Storage: at least 128GB hard disk

#### Software

- [Contrail Release 3.2 or later](#)

## Overview

A service chain is a set of services that are connected across networks. A service chain consists of service instances, left and right virtual networks, and a service policy attached to the networks. A service chain can have in-network services, in-network-nat services, and transparent services.

In an in-network service chain, packets are routed between service instance interfaces. When a packet is routed through the service chain, the source address of the packet entering the left interface of the service chain and source address of the packet exiting the right interface is the same. For more information, see *Service Chaining*.

## Configuration

### IN THIS SECTION

- [Create Virtual Network | 659](#)
- [Create Virtual Machine | 661](#)
- [Configure Service Template | 663](#)
- [Add Service Instance | 665](#)
- [Create Service Policy | 667](#)
- [Attach Service Policy | 669](#)
- [Launch Virtual Machine | 671](#)

These topics provide instructions to create an in-network service chain.

### Create Virtual Network

#### Step-by-Step Procedure

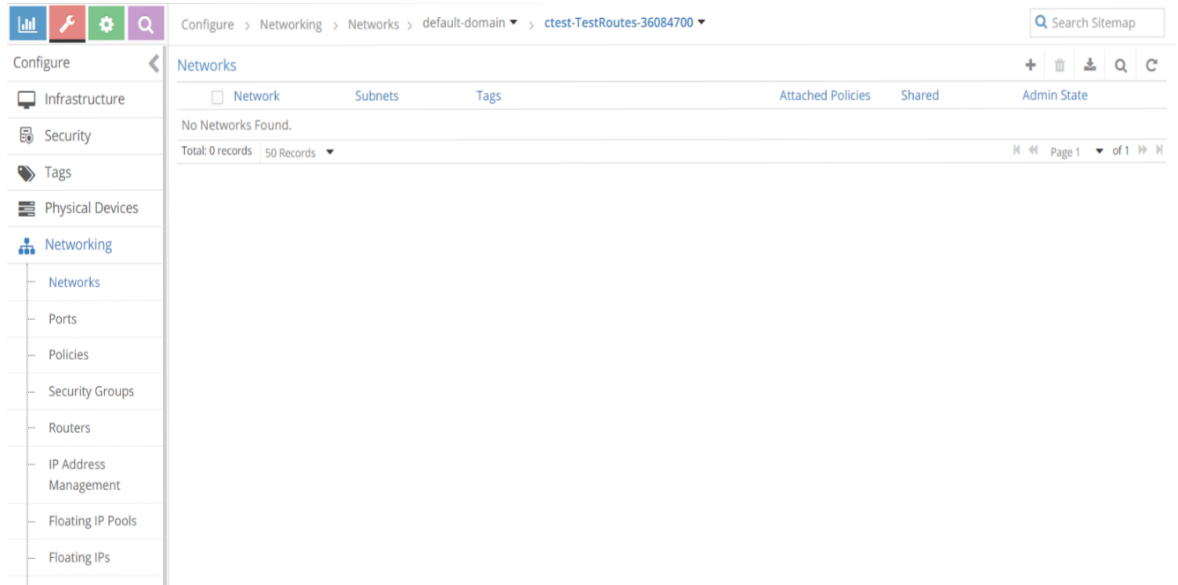
Use the Contrail user interface to create a left virtual network, right virtual network, and management virtual network.

To create a virtual network:

1. Click **Configure>Networking>Networks**.

The Networks page is displayed.

Figure 153: Networks Page



2. Click the add (+) icon to create a network.

The Create page is displayed.

Figure 154: Create Page

3. Enter a name for the network in the **Name** field.
4. Click **Subnets** and click add (+) to add subnets.

## Step-by-Step Procedure

In the row that is displayed,

- a. Click the arrow in the IPAM field and select **left-ipam** for that left virtual network, or select **right-ipam** for the right virtual network, or select **mgmt-ipam** for the management network.

Management network is not used to route packets. This network is used to help troubleshoot issues with the virtual machine.

### NOTE:

#### Step-by-Step Procedure

You can also create a new IPAM by following these steps:

- i. Click **Configure>Networking>IP Address Management** and click the add (+) icon.  
The Create page is displayed.
- ii. In the **Name** field, enter a name for the IPAM.
- iii. Click **Save**.  
The IP Address Management page is displayed.

- b. In the CIDR field, enter valid IPv4 or IPv6 subnet/mask.

5. Click **Save**.

The Networks page is displayed. All virtual networks that you created are displayed in the Networks page.

## Create Virtual Machine

### Step-by-Step Procedure

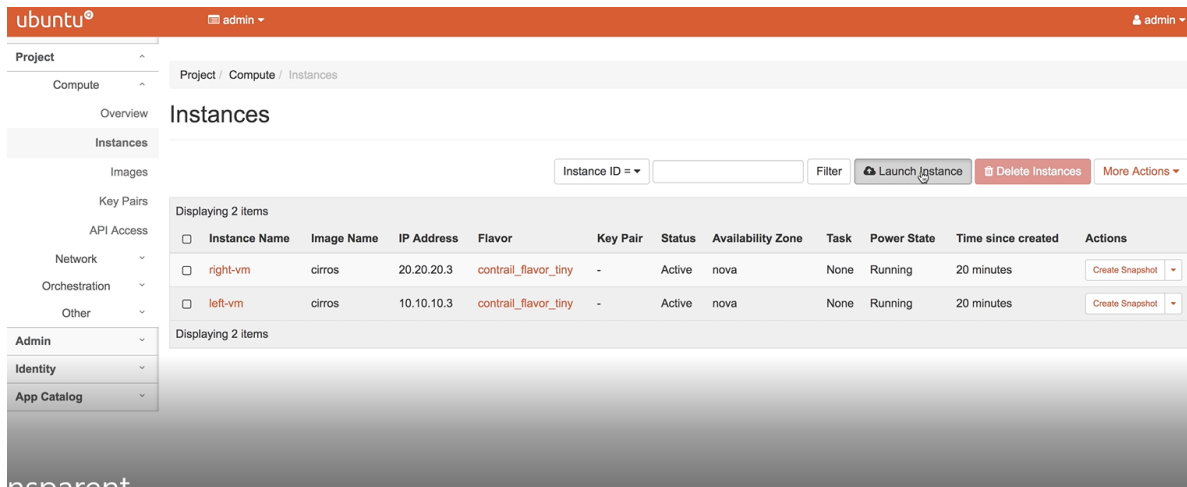
You use OpenStack to create virtual machines with left, right, and management interfaces.

Follow these steps to create virtual machines for left, right, and management networks.

1. Click **Project>Compute>Instances**.

The Instances page is displayed.

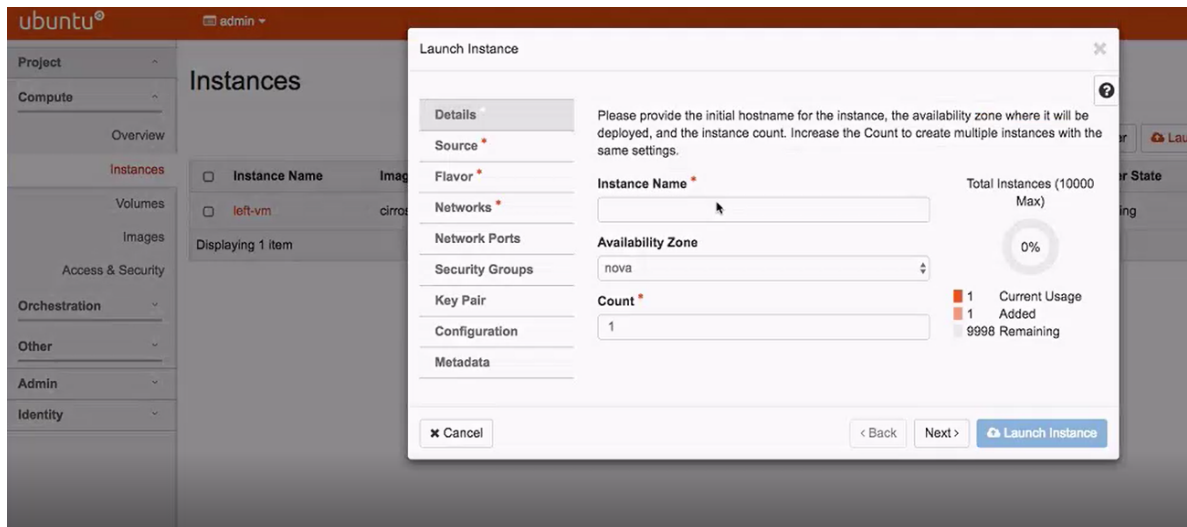
Figure 155: Instances Page



2. Click **Launch Instance** to create an instance.

The Details tab of the Launch Instance page is displayed.

Figure 156: Launch Instance



3. Enter a name for the virtual machine in the **Instance Name** field and click the **Source** tab.

The Source tab of the Launch Instance page is displayed.

4. Select an image file from the Available list by clicking the add (+) icon next to the image file.
5. Click the **Flavor** tab. See [Figure 156 on page 662](#).

The Flavor tab of the Launch Instance page is displayed.

**NOTE:** vSRX image with M1.large flavor is recommended for in-network virtual machine.

6. Select a flavor from the Available list by clicking the add (+) icon next to the flavor name.
7. Click the **Networks** tab. See [Figure 156 on page 662](#).

The Network tab of the Launch Instance page is displayed.

8. Select a network you want to associate with the virtual machine instance by clicking the add (+) icon next to the network name.
9. Click **Launch Instance** to launch the virtual machine instance. See [Figure 156 on page 662](#).

The Instances page is displayed.

All virtual machine instances that you created are displayed on the Instances page.

## Configure Service Template

### Step-by-Step Procedure

Follow these steps to configure a service template:

1. Click **Configure>Services>Service Templates**.

The Service Templates page is displayed.

2. Click the add (+) icon to create a service template. See [Figure 157 on page 664](#).

The Service Template tab of the Create page is displayed.

Enter the following information as given in [Table 48 on page 663](#):

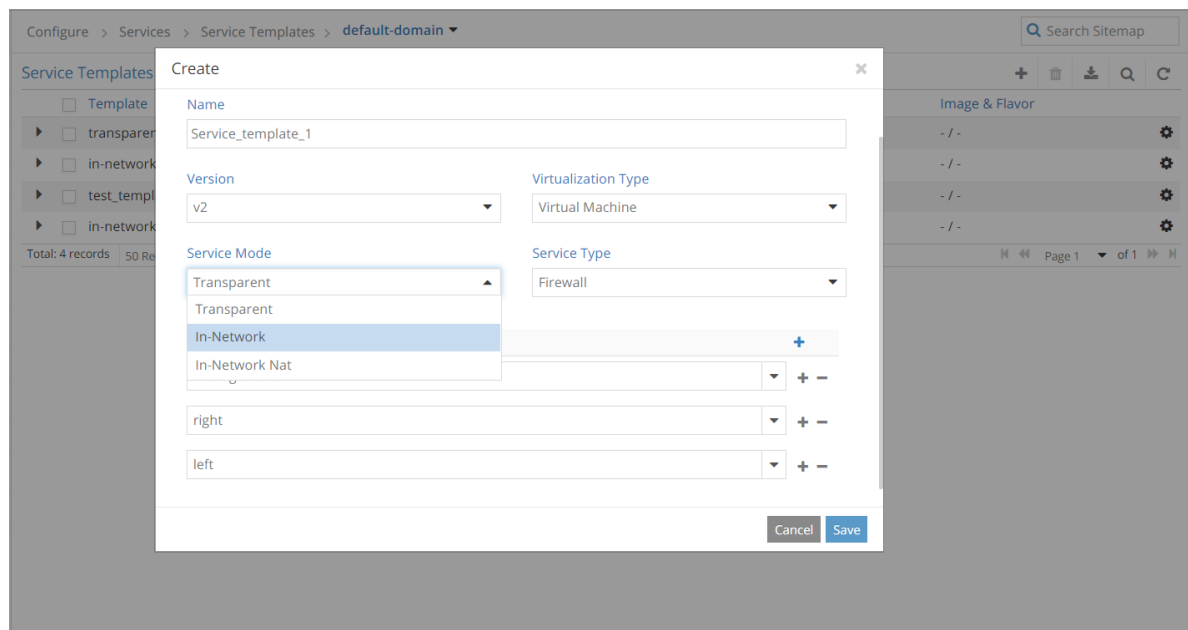
**Table 48: Add Service Template Fields**

| Field          | Action  |
|----------------|---|
| <b>Name</b>    | Enter a name for the service template.  |
| <b>Version</b> | Select <b>v2</b> as the version type.<br><br><b>NOTE:</b> Starting with Release 3.2, Contrail supports only <i>Service Chain Version 2 (v2)</i> . |

Table 48: Add Service Template Fields (Continued)

| Field                      | Action   |
|----------------------------|--|
| <b>Virtualization Type</b> | Select <b>Virtual Machine</b> as the virtualization type.  |
| <b>Service Mode</b>        | Select <b>In-Network</b> as the service mode.  |
| <b>Service Type</b>        | Select <b>Firewall</b> as the service type.  |
| <b>Interface(s)</b>        | <p>Click the add (+) icon and add the following interfaces:</p> <ul style="list-style-type: none"> <li>• management</li> <li>• left</li> <li>• right</li> </ul> <p><b>NOTE:</b> The interfaces created on the virtual machine must follow the same sequence as that of the interfaces in the service template.</p> |

Figure 157: Create Service Template





3. Click **Save** to save the service template.

The Service Templates page is displayed. All service templates that you created are displayed in the Service Templates page.

## Add Service Instance

### Step-by-Step Procedure

Follow these steps to add a service instance:

1. Click **Configure>Services>Service Instances**.

The Service Instances page is displayed.

2. Click the add (+) icon to add a service instance. See [Figure 158 on page 666](#).

The Service Instance tab of the Create page is displayed.

Enter the following information as given in [Table 49 on page 665](#):

**Table 49: Add Service Instance Fields**

| Field                   | Action   |
|-------------------------|--|
| <b>Name</b>             | Enter a name for the service instance.   |
| <b>Service Template</b> | Select <b>in-network - [in-network (management, left, right)] - v2</b> as the service template   |
| <b>Virtual Network</b>  | Select the virtual network for each interface type as given below: <ul style="list-style-type: none"> <li>• <b>management</b>—Select the management virtual network that you created.</li> <li>• <b>left</b>—Select the left virtual network that you created.</li> <li>• <b>right</b>—Select the right virtual network that you created.</li> </ul> |

Table 49: Add Service Instance Fields (Continued)

| Field              | Action  |
|--------------------|---|
| <b>Port Tuples</b> | <p>Click <b>Port Tuples</b> and click the add (+) icon to add new port tuples. See <a href="#">Figure 159 on page 667</a>.</p> <p>Click the arrow next to the newly added port tuple to select the virtual machine instance for each interface type as given below:</p> <ul style="list-style-type: none"> <li>• <b>management</b>—Select the management virtual machine instance that you created.</li> <li>• <b>left</b>—Select the left virtual machine instance that you created.</li> <li>• <b>right</b> —Select the right virtual machine instance that you created.</li> </ul> |

Figure 158: Create Service Instance

Configure > Services > Service Instances > default-domain > ctest-TestRoutes-36084700

Service Instances

Service Instance Tags Permissions

Name: service\_instance\_1

Service Template: in-network - [in-network (management, left,...)

Interface Type: management, left, right

Virtual Network: management\_vn, left\_vn, right\_vn

Port Tuples: Tuple +

Cancel Save

Figure 159: Create Port Tuples

▼ Port Tuples

Tuple +

---

▼ port-tuple0 + -

| Interface Type | Virtual Machine Interface          |
|----------------|------------------------------------|
| management     | Select Virtual Machine Interface ▼ |
| left           | Select Virtual Machine Interface ▼ |
| right          | Select Virtual Machine Interface ▼ |

---

▶ Service Health Check

Cancel Save

3. Click **Save** to save the service instance.

The Service Instances page is displayed. All service instances that you created are displayed in the Service Instances page.

## Create Service Policy

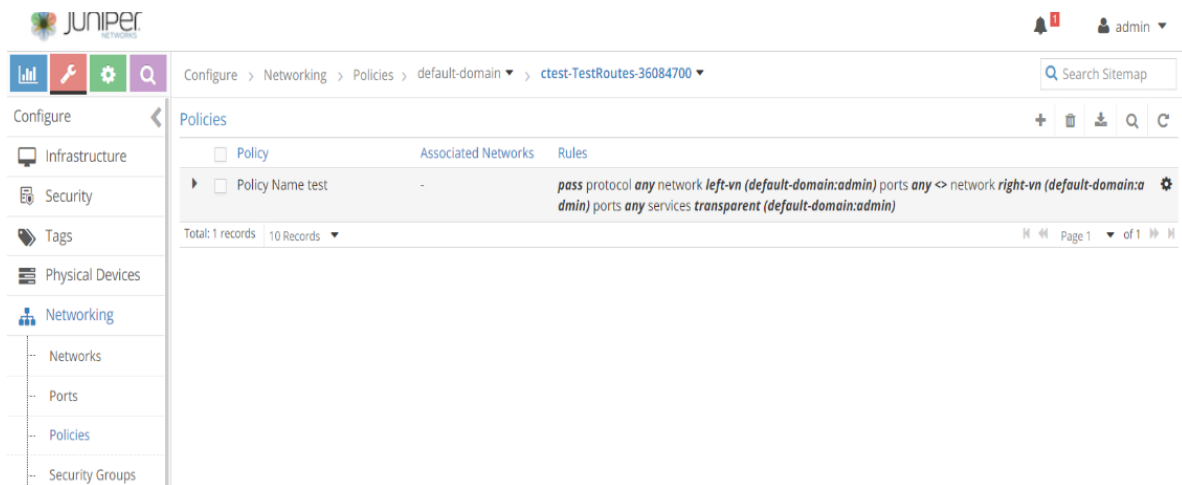
### Step-by-Step Procedure

Follow these steps to create a service policy:

1. Click **Configure>Networking>Policies**.

The Policies page is displayed.

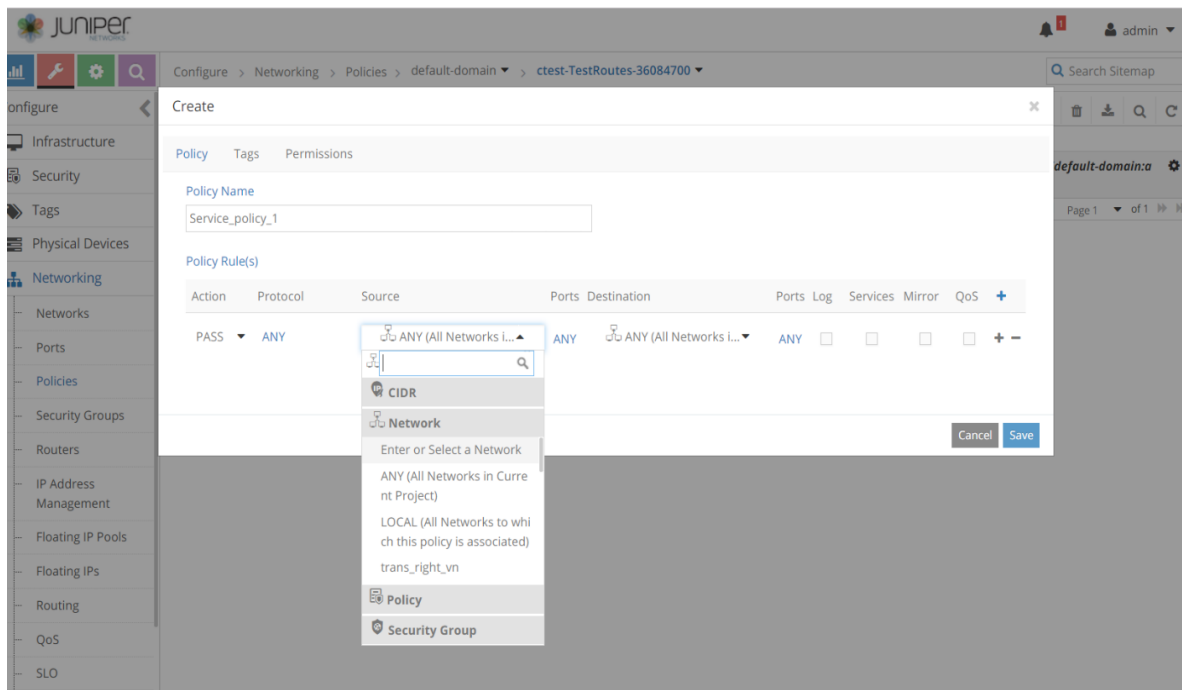
Figure 160: Policies Page



2. Click the add (+) icon to add a service policy. See [Figure 160 on page 668](#).

The Policy tab of the Create page is displayed.

Figure 161: Create Policy Page



3. Enter a name for the service policy in the **Policy Name** field.
4. Click add (+) in the **Policy Rule(s)** table to add a policy rule.

A row is added to the Policy Rule(s) table. See [Figure 161 on page 668](#).

5. In the row that is added:

a. Click the **Source** column and select **Network** from the source list.

From the networks list, select the left virtual network that you created.

b. Click the **Destination** column and select **Network** from the destination list.

From the networks list, select the right virtual network that you created.

c. Select the **Services** check box to enable services.

The Service Instance field is enabled.

d. Click the **Service Instance** field and select **in-network** from the service instance list.

6. Click **Save** to add the service policy.

The Policies page is displayed. All policies that you created are displayed in the Policies page.

## Attach Service Policy

### Step-by-Step Procedure

Follow these steps to attach a service policy to a network:

1. Click **Configure>Networking>Networks**.

The Networks page is displayed.

2. Add service policy to the left and right primary virtual networks.

### Step-by-Step Procedure

To add a service policy to a virtual network:

a. Click the settings icon given at the end of the row of the virtual network.

b. In the list that is displayed, click **Edit**. See [Figure 162 on page 670](#).

The Edit page is displayed. See [Figure 163 on page 670](#).

Figure 162: Networks Page

| Network             | Subnets | Tags | Attached Policies | Shared   | Admin State |
|---------------------|---------|------|-------------------|----------|-------------|
| trans_right_vn      | -       | -    | -                 | Disabled | Up          |
| trans_management_vn | -       | -    | -                 | Disabled | Up          |
| right_vn            | -       | -    | -                 | Disabled | Up          |
| left_vn             | -       | -    | -                 | Disabled | Up          |
| management_vn       | -       | -    | -                 | Disabled | Up          |
| trans_left_vn       | -       | -    | -                 | Disabled | Up          |

Figure 163: Edit Network Page

Configure > Networking > Networks > default-domain > ctest-TestRoutes-36084700

Edit

Network Tags Permissions

Name  
trans\_right\_vn

Network Policy(s)

- default-domain:ctest-TestRoutes-36084700:Policy Name test
- default-domain:admin:policy1

Host Route(s)

Advanced Options

DNS Server(s)

Route Target(s)

Export Route Target(s)

Cancel Save

- c. Click **Network Policy(s)** and select the network policy you want to add to the virtual network. See [Figure 163 on page 670](#).
- d. Click **Save**.

The policy is assigned to the network.

Repeat steps *a* through *d* to assign policies to other virtual networks.

## Launch Virtual Machine

### Step-by-Step Procedure

You launch virtual machines from OpenStack and test the traffic through the service chain by doing the following:

1. Launch the left virtual machine in left virtual network. For more information, see "[Create Virtual Machine](#)" on page 661.
2. Launch the right virtual machine in right virtual network. For more information, see "[Create Virtual Machine](#)" on page 661.
3. Ping the right virtual machine IP address from the left virtual machine.

Follow these steps to ping a virtual machine:

- a. Click **Project** > **Compute** > **Instances**.

All virtual machine instances that you created are displayed on the Instances page.

- b. From the list of virtual machines, click the left virtual machine.

The **Instances / Left Instance** page is displayed.

- c. Click the **Console** tab.

The Instance Console is displayed.

- d. Ping the right virtual machine IP address from the Instance Console.

### RELATED DOCUMENTATION

*Service Chaining*

---

[Example: Creating an In-Network-NAT Service Chain | 672](#)

---

[Example: Creating a Transparent Service Chain | 686](#)

## Example: Creating an In-Network-NAT Service Chain

### IN THIS SECTION

- [Hardware and Software Requirements | 672](#)
- [Overview | 672](#)
- [Configuration | 673](#)

This example provides step-by-step instructions to create an in-network-nat service chain by using the Contrail user interface.

### Hardware and Software Requirements

The following are the minimum requirements needed:

#### Hardware

- Processor: 4 core x86
- Memory: 32GB RAM
- Storage: at least 128GB hard disk

#### Software

- Contrail Release 3.2 or later

### Overview

A service chain is a set of services that are connected across networks. A service chain consists of service instances, left and right virtual networks, and a service policy attached to the networks. A service chain can have in-network services, in-network-nat services, and transparent services.

In an in-network-nat service chain, packets are routed between service instance interfaces. In-network-nat service chain does not require return traffic to be routed to the source network. When a packet is routed through the service chain, the source address of the packet entering the left interface of the service chain is updated and is not the same as the source address of the packet exiting the right interface. For more information, see *Service Chaining*.



## Configuration

### IN THIS SECTION

- [Create Virtual Network | 673](#)
- [Create Virtual Machine | 675](#)
- [Configure Service Template | 677](#)
- [Add Service Instance | 679](#)
- [Create Service Policy | 681](#)
- [Attach Service Policy | 683](#)
- [Launch Virtual Machine | 685](#)

These topics provide instructions to create an in-network-nat service chain.

### Create Virtual Network

#### Step-by-Step Procedure

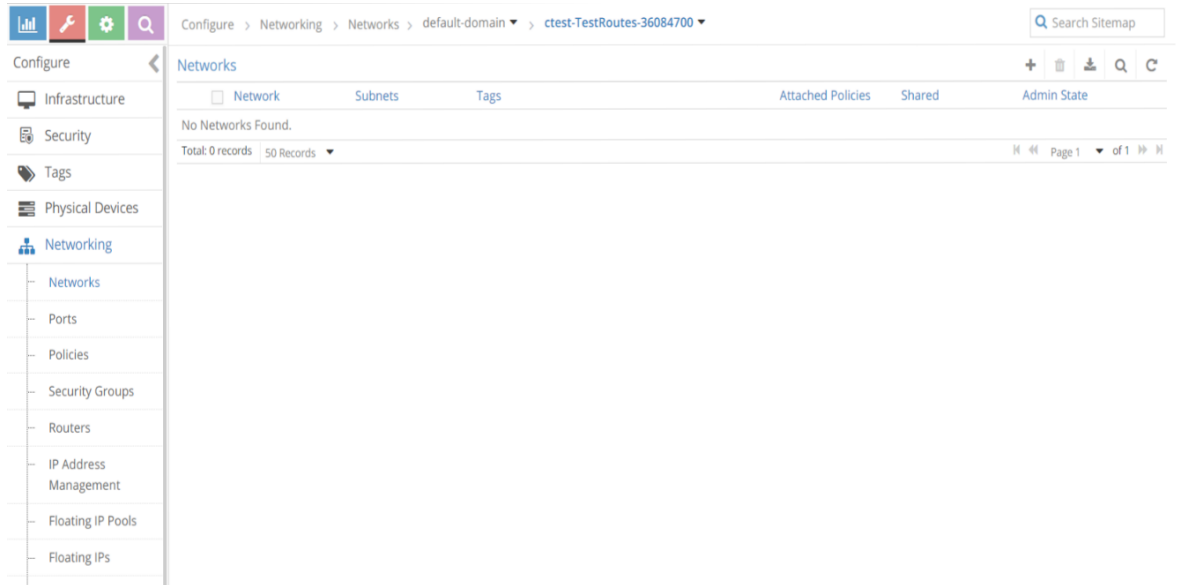
Use the Contrail user interface to create a left virtual network, right virtual network, and management virtual network.

To create a virtual network:

1. Click **Configure>Networking>Networks**.

The Networks page is displayed.

Figure 164: Networks Page



2. Click the add (+) icon to create a network.

The Create page is displayed.

Figure 165: Create Page

3. Enter a name for the network in the **Name** field.
4. Click **Subnets** and click add (+) to add subnets.

## Step-by-Step Procedure

In the row that is displayed,

- a. Click the arrow in the IPAM field and select **left-ipam** for the left virtual network, or select **right-ipam** for the right virtual network, or select **mgmt-ipam** for the management network.

Management network is not used to route packets. This network is used to help debug issues with the virtual machine.

### NOTE:

#### Step-by-Step Procedure

You can also create a new IPAM by following these steps:

- i. Click **Configure>Networking>IP Address Management** and click the add (+) icon.  
The IPAM tab of the Create page is displayed.
- ii. In the **Name** field, enter a name for the IPAM.
- iii. Click **Save**.  
The IP Address Management page is displayed.

- b. In the CIDR field, enter valid IPv4 or IPv6 subnet or mask.

5. Click **Save**.

The Networks page is displayed. All virtual networks that you created are displayed in the Networks page.

## Create Virtual Machine

### Step-by-Step Procedure

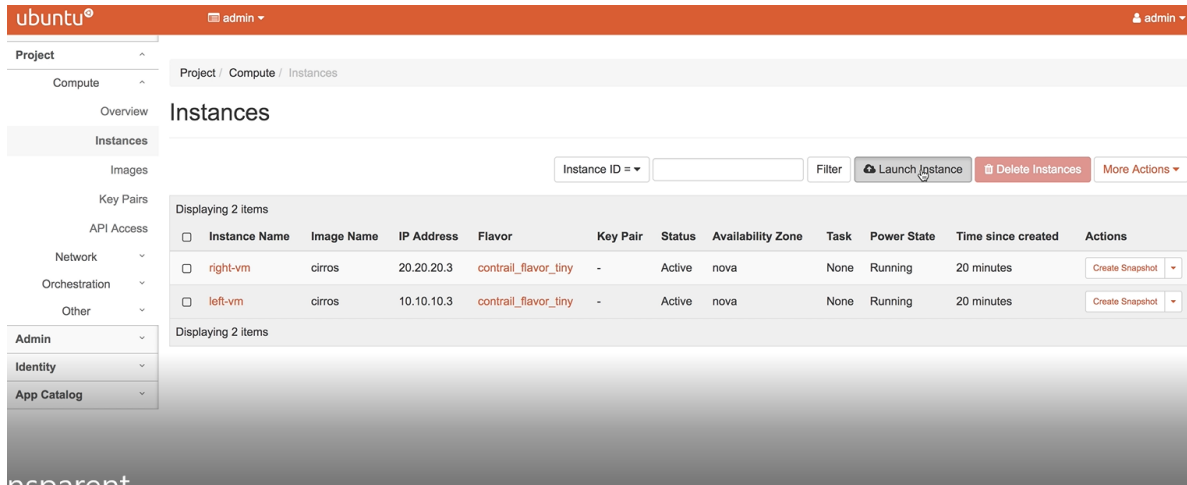
You use OpenStack to create virtual machines with left, right, and management interfaces.

Follow these steps to create virtual machines for left, right, and management networks.

1. Click **Project>Compute>Instances**.

The Instances page is displayed.

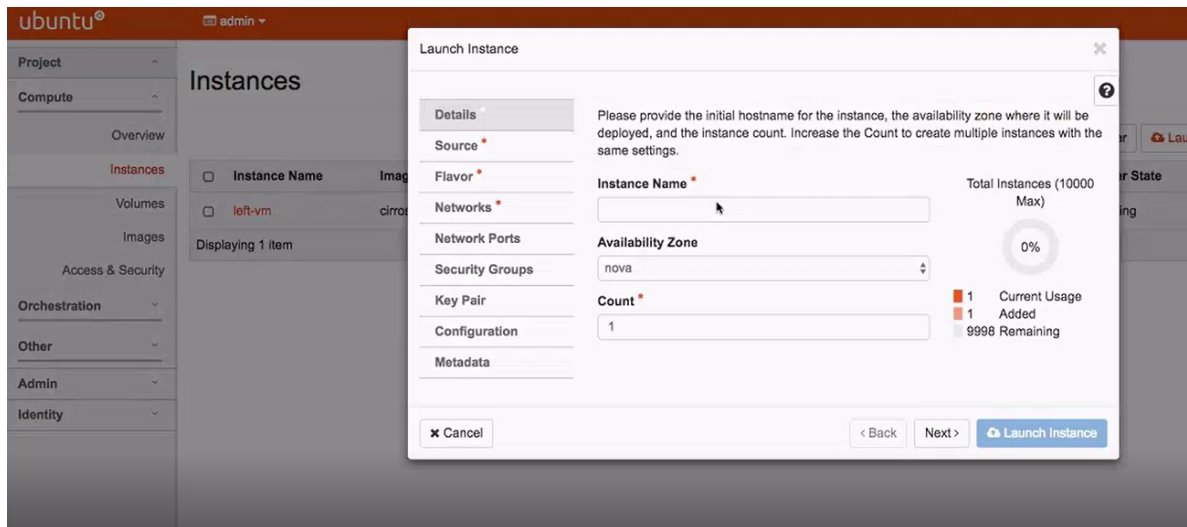
Figure 166: Instances Page



2. Click **Launch Instance** to create an instance.

The Details tab of the Launch Instance page is displayed.

Figure 167: Launch Instance



3. Enter a name for the virtual machine in the **Instance Name** field and click the **Source** tab.

The Source tab of the Launch Instance page is displayed.

4. Select an image file from the Available list by clicking the add (+) icon next to the image file.
5. Click the **Flavor** tab.

The Flavor tab of the Launch Instance page is displayed.

**NOTE:** vSRX image with M1.large flavor is recommended for in-network-nat virtual machine.

6. Select a flavor from the Available list by clicking the add (+) icon next to the flavor name.

7. Click the **Networks** tab.

The Network tab of the Launch Instance page is displayed.

8. Select a network you want to associate with the virtual machine instance by clicking the add (+) icon next to the network name.

9. Click **Launch Instance** to launch the virtual machine instance.

The Instances page is displayed.

All virtual machine instances that you created are displayed on the Instances page.

## Configure Service Template

### Step-by-Step Procedure

Follow these steps to configure a service template:

1. Click **Configure>Services>Service Templates**.

The Service Templates page is displayed.

2. Click the add (+) icon to create a service template. See [Figure 168 on page 678](#).

The Service Template tab of the Create page is displayed.

Enter the following information as given in [Table 50 on page 677](#):

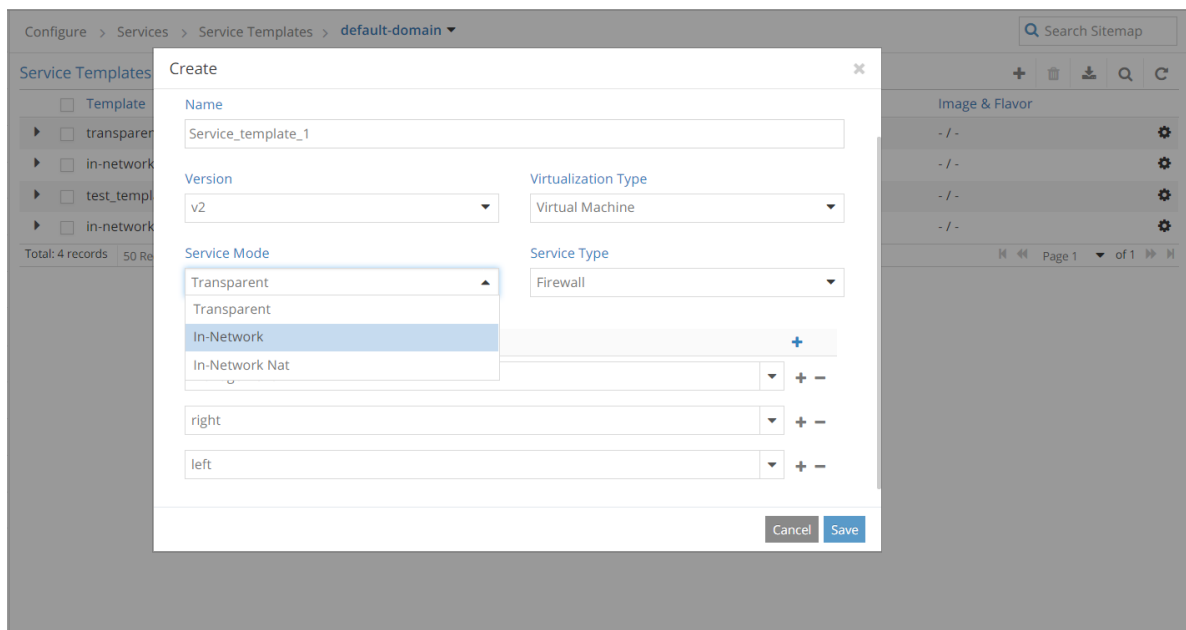
**Table 50: Add Service Template Fields**

| Field          | Action  |
|----------------|---|
| <b>Name</b>    | Enter a name for the service template.  |
| <b>Version</b> | Select <b>v2</b> as the version type.<br><br><b>NOTE:</b> Starting with Release 3.2, Contrail supports only <i>Service Chain Version 2 (v2)</i> . |

Table 50: Add Service Template Fields (Continued)

| Field                      | Action   |
|----------------------------|--|
| <b>Virtualization Type</b> | Select <b>Virtual Machine</b> as the virtualization type.  |
| <b>Service Mode</b>        | Select <b>In-Network Nat</b> as the service mode.  |
| <b>Service Type</b>        | Select <b>Firewall</b> as the service type.  |
| <b>Interface(s)</b>        | <p>Click the add (+) icon and add the following interfaces:</p> <ul style="list-style-type: none"> <li>• management</li> <li>• left</li> <li>• right</li> </ul> <p><b>NOTE:</b> The interfaces created on the virtual machine must follow the same sequence as that of the interfaces in the service template.</p> |

Figure 168: Create Service Template



3. Click **Save** to save the service template.

The Service Templates page is displayed. All service templates that you created are displayed in the Service Templates page.

## Add Service Instance

### Step-by-Step Procedure

Follow these steps to add a service instance:

1. Click **Configure>Services>Service Instances**.

The Service Instances page is displayed.

2. Click the add (+) icon to add a service instance. See [Figure 169 on page 680](#).

The Service Instance tab of the Create page is displayed.

Enter the following information as given in [Table 51 on page 679](#):

**Table 51: Add Service Instance Fields**

| Field                   | Action   |
|-------------------------|--|
| <b>Name</b>             | Enter a name for the service instance.   |
| <b>Service Template</b> | Select <b>in-network-nat - [in-network-nat (management, left, right)] - v2</b> as the service template   |
| <b>Virtual Network</b>  | Select the virtual network for each interface type as given below: <ul style="list-style-type: none"> <li>• <b>management</b>—Select the management virtual network that you created.</li> <li>• <b>left</b>—Select the left virtual network that you created.</li> <li>• <b>right</b>—Select the right virtual network that you created.</li> </ul> |

Table 51: Add Service Instance Fields (*Continued*)

| Field              | Action   |
|--------------------|--|
| <b>Port Tuples</b> | <p>Click <b>Port Tuples</b> and click the add (+) icon to add new port tuples. See <a href="#">Figure 170 on page 681</a>.</p> <p>Click the arrow next to the newly added port tuple to select the virtual machine instance for each interface type as given below:</p> <ul style="list-style-type: none"> <li>• <b>management</b>—Select the management virtual machine instance that you created.</li> <li>• <b>left</b>—Select the left virtual machine instance that you created.</li> <li>• <b>right</b>—Select the right virtual machine instance that you created.</li> </ul> |

Figure 169: Create Service Instance

The screenshot shows the 'Create' dialog for a Service Instance. The breadcrumb path is 'Configure > Services > Service Instances > default-domain > ctest-TestRoutes-36084700'. The dialog has tabs for 'Service Instance', 'Tags', and 'Permissions'. The 'Service Instance' tab is active, showing the following configuration:

- Name:** service\_instance\_1
- Service Template:** in-network-nat - [in-network-nat (managem... (dropdown)
- Interface Type:** management, left, right
- Virtual Network:** management\_vn, left\_vn, right\_vn (dropdowns)
- Port Tuples:** A section with a '+' icon to add a new tuple.

Buttons for 'Cancel' and 'Save' are located at the bottom right of the dialog.



Figure 170: Create Port Tuples

▼ Port Tuples

Tuple +

---

▼ port-tuple0 + -

| Interface Type | Virtual Machine Interface          |
|----------------|------------------------------------|
| management     | Select Virtual Machine Interface ▼ |
| left           | Select Virtual Machine Interface ▼ |
| right          | Select Virtual Machine Interface ▼ |

---

► Service Health Check

Cancel Save

3. Click **Save** to save the service instance.

The Service Instances page is displayed. All service instances that you created are displayed in the Service Instances page.

## Create Service Policy

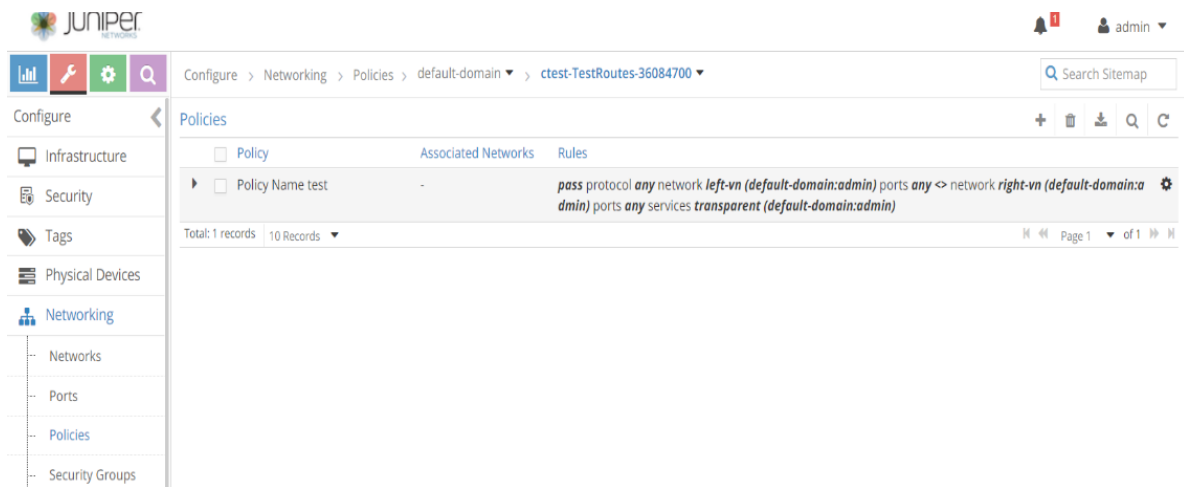
### Step-by-Step Procedure

Follow these steps to create a service policy:

1. Click **Configure>Networking>Policies**.

The Policies page is displayed.

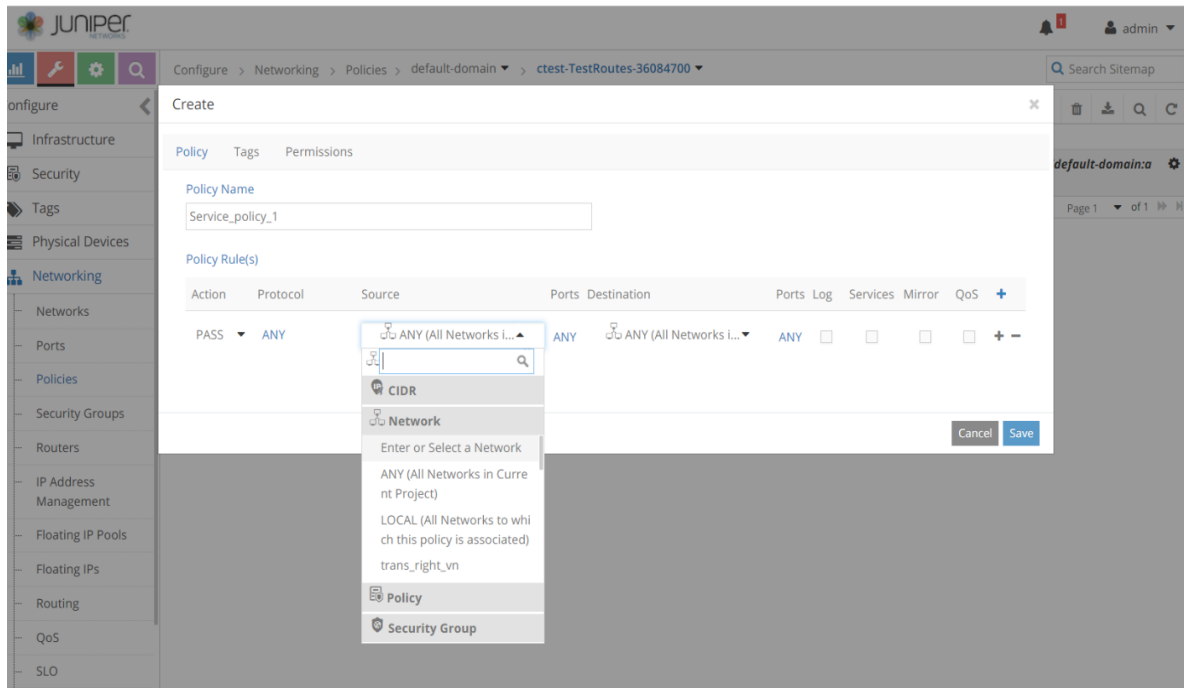
Figure 171: Policies Page



2. Click the add (+) icon to add a service policy. See [Figure 171 on page 682](#).

The Policy tab of the Create page is displayed.

Figure 172: Create Policy Page



3. Enter a name for the service policy in the **Policy Name** field.
4. Click add (+) in the **Policy Rule(s)** table to add a policy rule.

A row is added to the Policy Rule(s) table. See [Figure 172 on page 682](#).

5. In the row that is added:

a. Click the **Source** column and select **Network** from the source list.

From the networks list, select the left virtual network that you created.

b. Click the **Destination** column and select **Network** from the destination list.

From the networks list, select the right virtual network that you created.

c. Select the **Services** check box to enable services.

The Service Instance field is enabled.

d. Click the **Service Instance** field and select **in-network-nat** from the service instance list.

6. Click **Save** to add the service policy.

The Policies page is displayed. All policies that you created are displayed in the Policies page.

## Attach Service Policy

### Step-by-Step Procedure

Follow these steps to attach a service policy to a network:

1. Click **Configure>Networking>Networks**.

The Networks page is displayed.

2. Add service policy to the left and right virtual networks.

### Step-by-Step Procedure

To add a service policy to a virtual network:

a. Click the settings icon given at the end of the row of the virtual network.

b. In the list that is displayed, click **Edit**. See [Figure 173 on page 684](#).

The Edit page is displayed. See [Figure 174 on page 684](#).

Figure 173: Networks Page

| Network             | Subnets | Tags | Attached Policies | Shared   | Admin State |
|---------------------|---------|------|-------------------|----------|-------------|
| trans_right_vn      | -       | -    | -                 | Disabled | Up          |
| trans_management_vn | -       | -    | -                 | Disabled | Up          |
| right_vn            | -       | -    | -                 | Disabled | Up          |
| left_vn             | -       | -    | -                 | Disabled | Up          |
| management_vn       | -       | -    | -                 | Disabled | Up          |
| trans_left_vn       | -       | -    | -                 | Disabled | Up          |

Total: 6 records | 50 Records | Page 1 of 1

Figure 174: Edit Network Page

Configure > Networking > Networks > default-domain > ctest-TestRoutes-36084700

Edit

Network | Tags | Permissions

Name: trans\_right\_vn

Network Policy(s)

- default-domain:ctest-TestRoutes-36084700:Policy Name test
- default-domain:admin:policy1

Host Route(s)

Advanced Options

DNS Server(s)

Route Target(s)

Export Route Target(s)

Cancel Save

- c. Click **Network Policy(s)** and select the network policy you want to add to the virtual network. See [Figure 174 on page 684](#).
- d. Click **Save**.

The policy is assigned to the network.

Repeat steps *a* through *d* to assign policies to other virtual networks.

## Launch Virtual Machine

### Step-by-Step Procedure

You can launch virtual machines from OpenStack and test the traffic through the service chain by doing the following:

1. Launch the left virtual machine in left virtual network. For more information, see "[Create Virtual Machine](#)" on page 675.
2. Launch the right virtual machine in right virtual network. For more information, see "[Create Virtual Machine](#)" on page 675.
3. Ping the right virtual machine IP address from the left virtual machine.

Follow these steps to ping a virtual machine:

- a. Click **Project** > **Compute** > **Instances**.

All virtual machine instances that you created are displayed on the Instances page.

- b. From the list of virtual machines, click the left virtual machine.

The **Instances / Left Instance** page is displayed.

- c. Click the **Console** tab.

The Instance Console is displayed.

- d. Ping the right virtual machine IP address from the Instance Console.

### RELATED DOCUMENTATION

*Service Chaining*

---

[Example: Creating an In-Network Service Chain | 658](#)

---

[Example: Creating a Transparent Service Chain | 686](#)

## Example: Creating a Transparent Service Chain

### IN THIS SECTION

- [Hardware and Software Requirements | 686](#)
- [Overview | 686](#)
- [Configuration | 686](#)

This example provides step-by-step instructions to create a transparent service chain by using the Contrail user interface.

### Hardware and Software Requirements

The following are the minimum requirements needed:

#### Hardware

- Processor: 4 core x86
- Memory: 32GB RAM
- Storage: at least 128GB hard disk

#### Software

- Contrail Release 3.2 or later

### Overview

A service chain is a set of services that are connected across networks. A service chain consists of service instances, left and right virtual networks, and a service policy attached to the networks. A service chain can have in-network services, in-network-nat services, and transparent services. A transparent service chain is used for services that do not modify packets that are bridged between service instance interfaces. For more information, see *Service Chaining*.

### Configuration

#### IN THIS SECTION

- [Create Primary Virtual Network | 687](#)

- [Create Secondary Virtual Network | 689](#)
- [Create Virtual Machine | 690](#)
- [Configure Service Template | 691](#)
- [Add Service Instance | 693](#)
- [Create Service Policy | 696](#)
- [Attach Service Policy | 698](#)
- [Launch Virtual Machine | 699](#)

These topics provide instructions to create a transparent service chain.

## Create Primary Virtual Network

### Step-by-Step Procedure

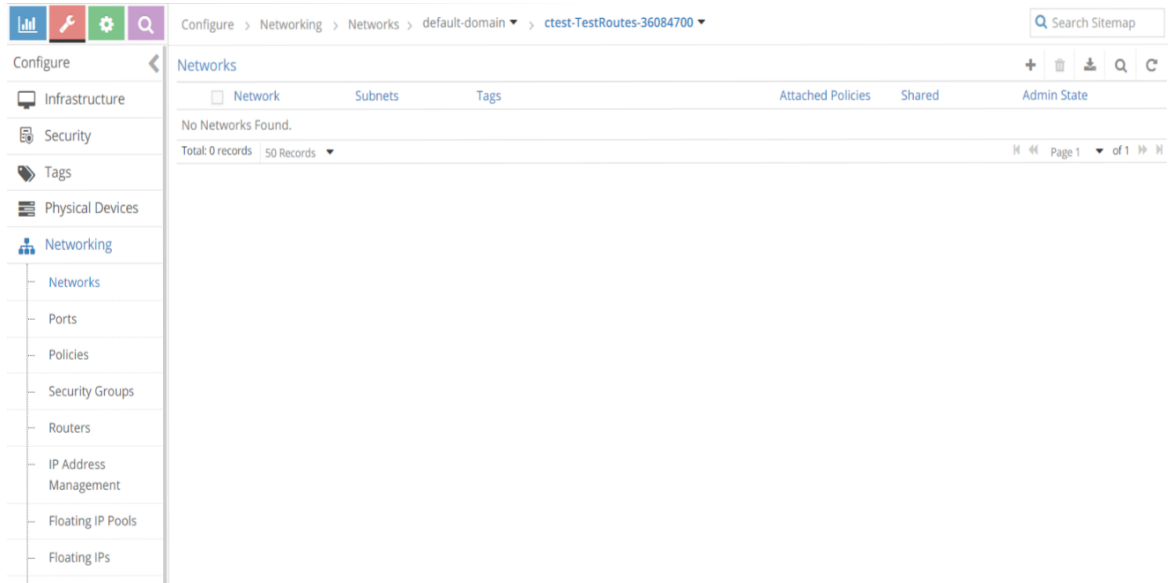
Use the Contrail user interface to create three primary virtual networks--left virtual network, right virtual network, and management virtual network. You attach service policies to the primary virtual networks that you create.

To create a virtual network:

1. Click **Configure>Networking>Networks**.

The Networks page is displayed.

Figure 175: Networks Page



2. Click the add (+) icon to create a network.

The Network tab of the Create page is displayed.

Figure 176: Create Page

3. Enter a name for the network in the **Name** field.
4. Click **Subnets** and click add (+) to add subnets.



## Step-by-Step Procedure

In the row that is displayed:

- a. Click the arrow in the IPAM field and select the **left-ipam** that you created for that left virtual network, or select the **right-ipam** that you created for the right virtual network, or select the **mgmt-ipam** that you created for the management network.

Management network is not used to route packets. This network is used to help debug issues with the virtual machine.

**NOTE:** You can also create a new IPAM by following the steps given below:

### Step-by-Step Procedure

- i. Click **Configure>Networking>IP Address Management** and click the add (+) icon.  
The IPAM tab of the Create page is displayed.
- ii. In the **Name** field, enter a name for the IPAM.
- iii. Click **Save**.  
The IP Address Management page is displayed.

- b. In the CIDR field, enter valid IPv4 or IPv6 subnet or mask.

5. Click **Save**.

The Networks page is displayed. All virtual networks that you created are displayed in the Networks page.

## Create Secondary Virtual Network

### Step-by-Step Procedure

Use the Contrail user interface to create three secondary virtual networks--left virtual network, right virtual network, and management virtual network. You associate the secondary virtual network to the transparent service instance that you create. For more information on creating virtual networks, see ["Create Primary Virtual Network" on page 687](#).

## Create Virtual Machine

### Step-by-Step Procedure

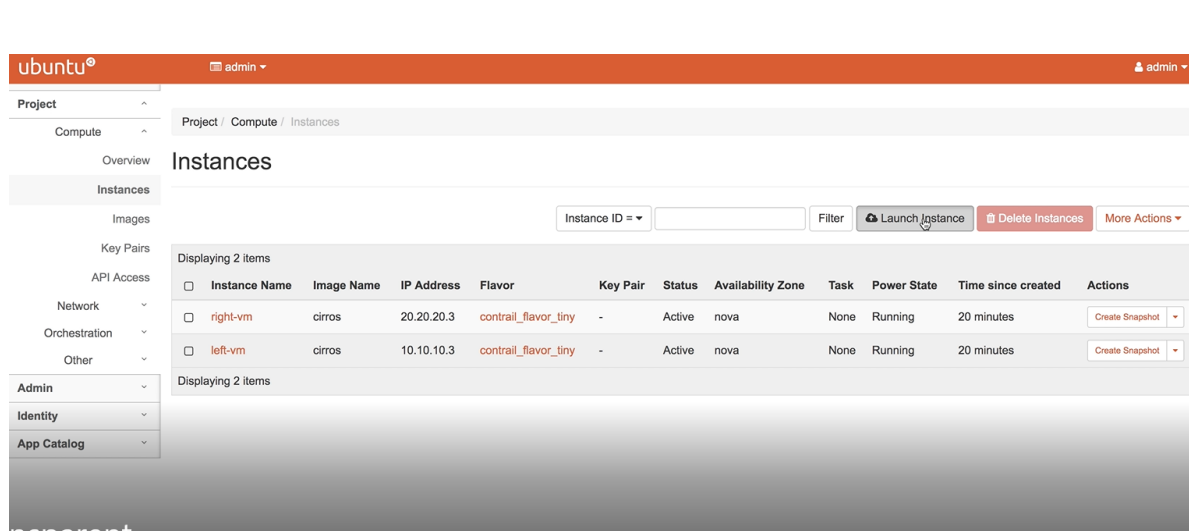
You use OpenStack to create virtual machines with left, right, and management interfaces.

Follow these steps to create virtual machines for left, right, and management networks.

#### 1. Click **Project>Compute>Instances**.

The Instances page is displayed.

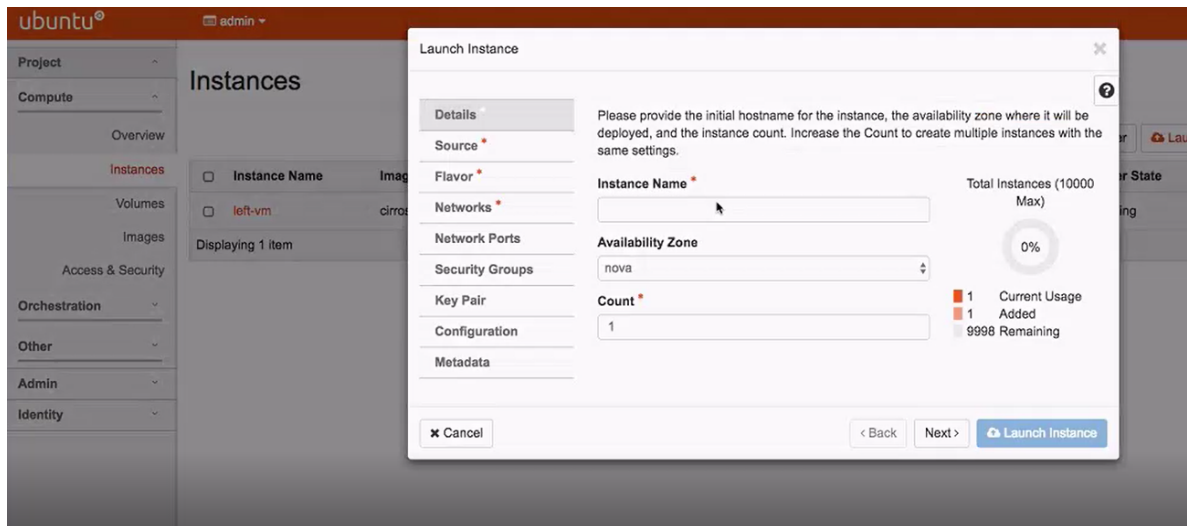
**Figure 177: Instances Page**



#### 2. Click **Launch Instance** to create an instance.

The Details tab of the Launch Instance page is displayed.

Figure 178: Launch Instance



3. Enter a name for the virtual machine in the **Instance Name** field and click the **Source** tab.

The Source tab of the Launch Instance page is displayed.

4. Select an image file from the Available list by clicking the add (+) icon next to the image file.
5. Click the **Flavor** tab. See [Figure 178 on page 691](#).

The Flavor tab of the Launch Instance page is displayed.

6. Select a flavor from the Available list by clicking the add (+) icon next to the flavor name.
7. Click the **Networks** tab. See [Figure 178 on page 691](#).

The Network tab of the Launch Instance page is displayed.

8. Select the secondary network you want to associate with the virtual machine instance by clicking the add (+) icon next to the network name.
9. Click **Launch Instance** to launch the virtual machine instance. See [Figure 178 on page 691](#).

The Instances page is displayed.

All virtual machine instances that you created are displayed on the Instances page.

## Configure Service Template

### Step-by-Step Procedure

Follow these steps to configure a service template:

1. Click **Configure>Services>Service Templates**.

The Service Templates page is displayed.

2. Click the add (+) icon to create a service template. See [Figure 179 on page 693](#).

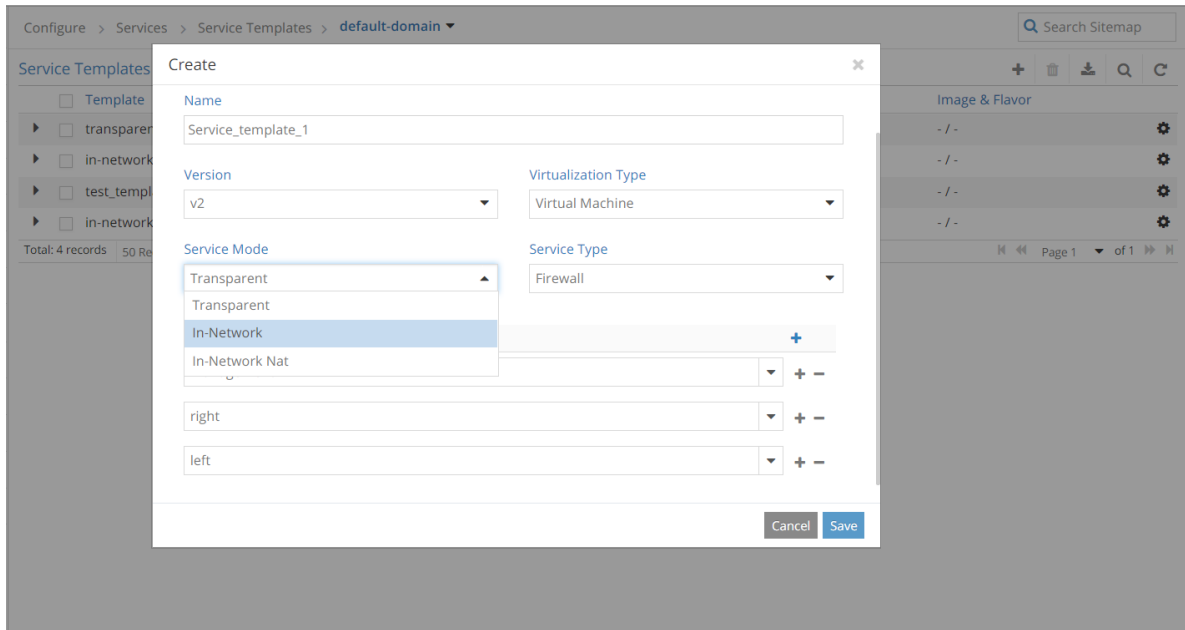
The Service Template tab of the Create page is displayed.

Enter the following information as given in [Table 52 on page 692](#):

**Table 52: Add Service Template Fields**

| Field                      | Action  |
|----------------------------|---|
| <b>Name</b>                | Enter a name for the service template.  |
| <b>Version</b>             | Select <b>v2</b> as the version type.<br><br><b>NOTE:</b> Starting with Release 3.2, Contrail supports only <i>Service Chain Version 2 (v2)</i> .   |
| <b>Virtualization Type</b> | Select <b>Virtual Machine</b> as the virtualization type.   |
| <b>Service Mode</b>        | Select <b>Transparent</b> as the service mode.  |
| <b>Service Type</b>        | Select <b>Firewall</b> as the service type.   |
| <b>Interface(s)</b>        | Click the add (+) icon and add the following interfaces: <ul style="list-style-type: none"> <li>• management</li> <li>• left</li> <li>• right</li> </ul> <p><b>NOTE:</b> The interfaces created on the virtual machine must follow the same sequence as that of the interfaces in the service template.</p> |

**Figure 179: Create Service Template**



3. Click **Save** to save the service template.

The Service Templates page is displayed. All service templates that you created are displayed in the Service Templates page.

## Add Service Instance

### Step-by-Step Procedure

Follow these steps to add a service instance:

1. Click **Configure>Services>Service Instances**.

The Service Instances page is displayed.

2. Click the add (+) icon to add a service instance. See [Figure 180 on page 695](#).

The Service Instance tab of the Create page is displayed.

Enter the following information as given in [Table 53 on page 694](#):

Table 53: Add Service Instance Fields

| Field                   | Action  |
|-------------------------|---|
| <b>Name</b>             | Enter a name for the service instance.  |
| <b>Service Template</b> | Select <b>transparent - [transparent (management, left, right)] - v2</b> as the service template.   |
| <b>Virtual Network</b>  | <p>Select the virtual network for each interface type as given below:</p> <ul style="list-style-type: none"> <li>• <b>management</b>—Select the secondary management virtual network that you created.</li> <li>• <b>left</b>—Select the secondary left virtual network that you created.</li> <li>• <b>right</b>—Select the secondary right virtual network that you created.</li> </ul>   |
| <b>Port Tuples</b>      | <p>Click <b>Port Tuples</b> and click the add (+) icon to add new port tuples. See <a href="#">Figure 181 on page 695</a>.</p> <p>Click the arrow next to the newly added port tuple to select the virtual machine instance for each interface type as given below:</p> <ul style="list-style-type: none"> <li>• <b>management</b> —Select the management virtual machine instance that you created.</li> <li>• <b>left</b>—Select the left virtual machine instance that you created.</li> <li>• <b>right</b>—Select the right virtual machine instance that you created.</li> </ul> |

Figure 180: Create Service Instance

Configure > Services > Service Instances > default-domain > ctest-TestRoutes-36084700

Service Instances

Service Instance Tags Permissions

Name: service\_instance\_1

Service Template: transparent - [transparent (management, le...]

Interface Type: management, left, right

Virtual Network: trans\_management\_vn, trans\_left\_vn, trans\_right\_vn

Port Tuples

Tuple +

Cancel Save

Figure 181: Create Port Tuples

Port Tuples

Tuple +

port-tuple0 + -

Interface Type: management, left, right

Virtual Machine Interface: Select Virtual Machine Interface

Service Health Check

Cancel Save

3. Click **Save** to save the service instance.

The Service Instances page is displayed. All service instances that you created are displayed in the Service Instances page.

## Create Service Policy

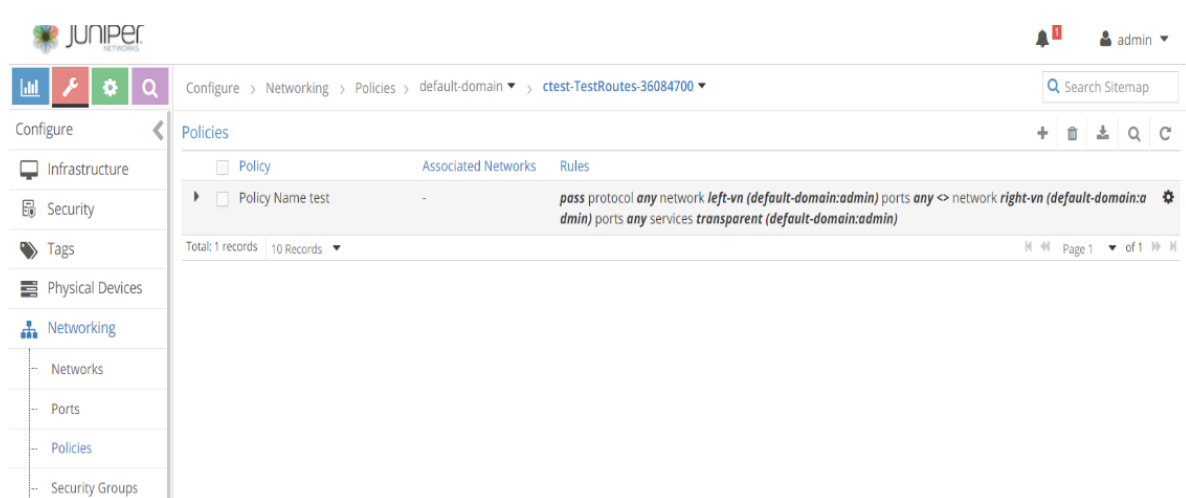
### Step-by-Step Procedure

Follow these steps to create a service policy:

1. Click **Configure>Networking>Policies**.

The Policies page is displayed.

**Figure 182: Policies Page**

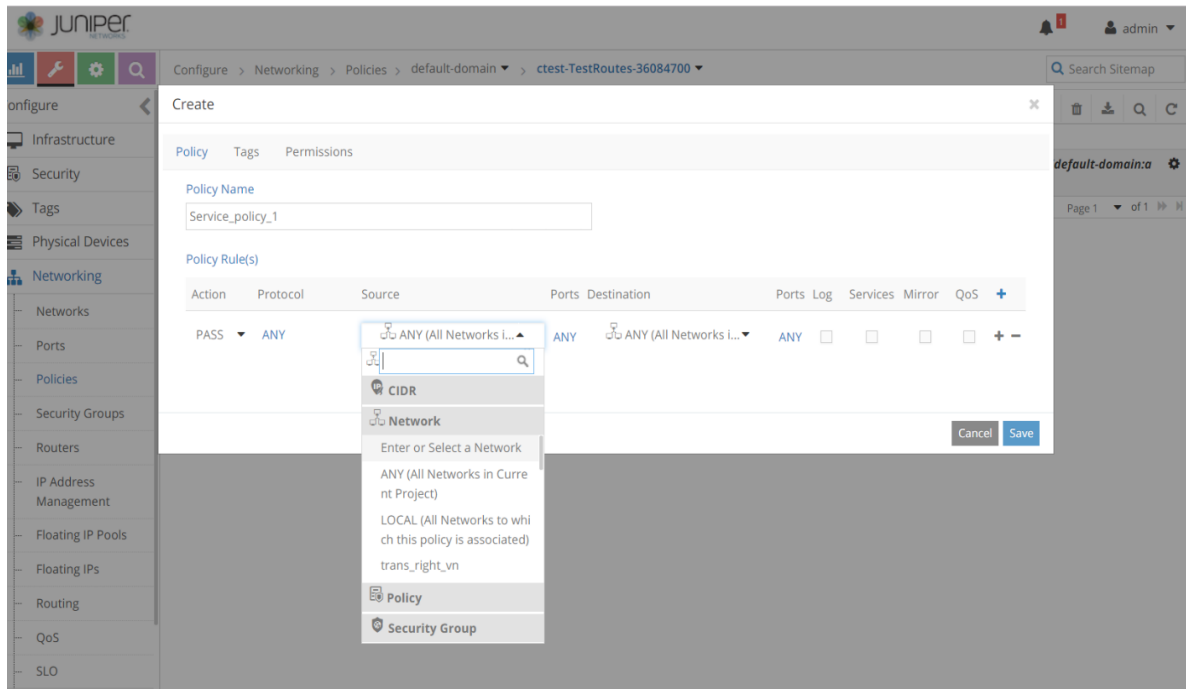


2. Click the add (+) icon to add a service policy. See [Figure 182 on page 696](#).

The Policy tab of the Create page is displayed.



Figure 183: Create Policy Page



3. Enter a name for the service policy in the **Policy Name** field.
4. Click add (+) in the **Policy Rule(s)** table to add a policy rule.
 

A row is added to the Policy Rule(s) table. See [Figure 183 on page 697](#).
5. In the row that is added:
  - a. Click the **Source** column and select **Network** from the source list.
 

From the networks list, select the primary left virtual network that you created.
  - b. Click the **Destination** column and select **Network** from the destination list.
 

From the networks list, select the primary right virtual network that you created.
  - c. Select the **Services** check box to enable services.
 

The Service Instance field is enabled.
  - d. Click the **Service Instance** field and select **transparent** from the service instance list.
6. Click **Save** to add the service policy.
 

The Policies page is displayed. All policies that you created are displayed in the Policies page.

## Attach Service Policy

### Step-by-Step Procedure

Follow these steps to attach a service policy to a network:

1. Click **Configure>Networking>Networks**.

The Networks page is displayed.

2. Add service policy to the left and right primary virtual networks.

### Step-by-Step Procedure

To add a service policy to a virtual network:

- a. Click the settings icon given at the end of the row of the virtual network.
- b. In the list that is displayed, click **Edit**. See [Figure 184 on page 698](#).

The Edit page is displayed. See [Figure 185 on page 699](#).

**Figure 184: Networks Page**

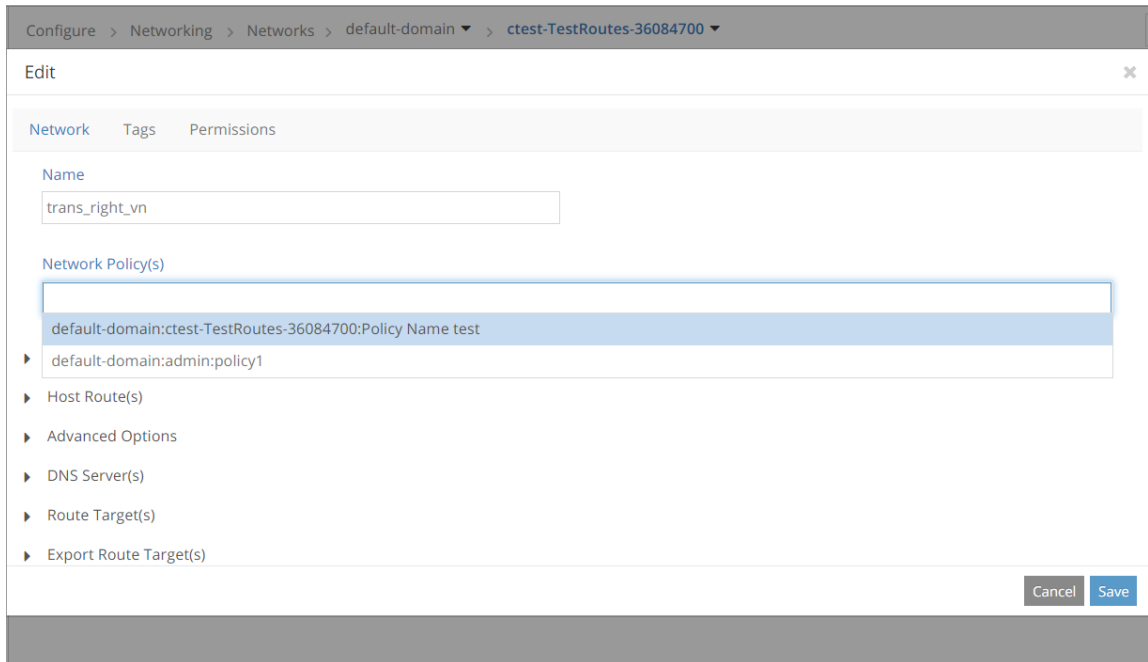
The screenshot shows the 'Networks' page in a management console. The breadcrumb navigation is 'Configure > Networking > Networks > default-domain > ctest-TestRoutes-36084700'. There is a search bar for the Sitemap. The table below lists several virtual networks. A context menu is open over the 'management\_vn' row, showing 'Edit' and 'Delete' options.

| Network             | Subnets | Tags | Attached Policies | Shared   | Admin State |
|---------------------|---------|------|-------------------|----------|-------------|
| trans_right_vn      | -       | -    | -                 | Disabled | Up          |
| trans_management_vn | -       | -    | -                 | Disabled | Up          |
| right_vn            | -       | -    | -                 | Disabled | Up          |
| left_vn             | -       | -    | -                 | Disabled | Up          |
| management_vn       | -       | -    | -                 | Disabled | Up          |
| trans_left_vn       | -       | -    | -                 | Disabled | Up          |

Total: 6 records | 50 Records

Page 1 of 1

**Figure 185: Edit Network Page**



- c. Click **Network Policy(s)** and select the network policy you want to add to the virtual network. See [Figure 185 on page 699](#).
- d. Click **Save**.

The policy is assigned to the network.

Repeat steps *a* through *d* to assign policies to other virtual networks.

## Launch Virtual Machine

### Step-by-Step Procedure

You can launch virtual machines from OpenStack and test the traffic through the service chain by doing the following:

1. Launch the left virtual machine in the primary left virtual network. For more information, see ["Create Virtual Machine" on page 690](#).
2. Launch the right virtual machine in the primary right virtual network. For more information, see ["Create Virtual Machine" on page 690](#).
3. Ping the right virtual machine IP address from the left virtual machine.

Follow these steps to ping a virtual machine:

- a. Click **Project > Compute > Instances**.

All virtual machine instances that you created are displayed on the Instances page.

- b. From the list of virtual machines, click the left virtual machine.

The **Instances / Left Instance** page is displayed.

- c. Click the **Console** tab.

The Instance Console is displayed.

- d. Ping the right virtual machine IP address from the Instance Console.

## RELATED DOCUMENTATION

*Service Chaining*

---

[Example: Creating an In-Network-NAT Service Chain | 672](#)

---

[Example: Creating an In-Network Service Chain | 658](#)

# Adding Physical Network Functions in Service Chains

## IN THIS CHAPTER

- [Using Physical Network Functions in Contrail Service Chains | 701](#)
- [Example: Adding a Physical Network Function Device to a Service Chain | 703](#)

## Using Physical Network Functions in Contrail Service Chains

### IN THIS SECTION

- [PNF Service Chaining Objects | 701](#)
- [Prerequisites and Assumptions | 702](#)

Contrail Release 3.0 and greater supports service appliance-based physical network functions devices (PNFs) in service chains, enabling the creation of service chains that include a combination of virtual network functions (VNFs) and PNFs. The PNFs are also supported with Contrail Device Manager.

### PNF Service Chaining Objects

As of Contrail Release 3.0, Contrail has objects used to support PNFs in service chains, including:

- service appliance (SA)—represents a single physical appliance
- service appliance set (SA set)—represents a collection of functionally equivalent SAs, all running the same software with the same capabilities

A service appliance is associated with a physical router that has physical interfaces for the left, right, management, or other interfaces.

There can be more than one service appliance and associated physical router and physical interface objects representing it.

A physical appliance can host more than one service appliance through a logical system or other virtualization capability.

The service template object supports a physical network function service template (PNF-ST). The PNF-ST is associated with a service appliance set, which represents a pool of service appliances that can be used when the PNF-ST is instantiated.

Only the transparent service mode is supported for PNF-STs.

## Prerequisites and Assumptions

The following are the prerequisites for implementing a service appliance with a Contrail controller.

- Before the controller can use a PNF SA, the controller must be connected to a service control gateway (SCG) router, such as an MX Series router.
- The Contrail Device Manager must manage the SCG router.
- The PNF SA must be configured, and it must be configured to operate as an Ethernet bridge. The Contrail controller does not automatically implement PNF SA configuration.
- Infrastructure interfaces (physical interfaces or aggregated Ethernet interfaces) on the SCG facing the SA must be preconfigured. The interfaces must be able to support VLAN-based units.
- Layer 2 VPNs as supported by the Contrail Device Manager are only available with Juniper Networks Junos Release 14.2R4 or greater. If an earlier version of Junos is used, you must mark the virtual networks in Contrail with the forwarding mode “L3 only.”
- Logical interfaces for connecting the service gateway VRFs to the customer and the Internet must be preconfigured.

## RELATED DOCUMENTATION

| [Example: Adding a Physical Network Function Device to a Service Chain](#) | 703

## Example: Adding a Physical Network Function Device to a Service Chain

### IN THIS SECTION

- [Prerequisites for Adding a PNF to a Service Chain | 703](#)

Beginning with Contrail 3.0, it is possible to add a physical network function (PNF) device to a service chain. This section provides an example of creating a service chain that includes a PNF.

### Prerequisites for Adding a PNF to a Service Chain

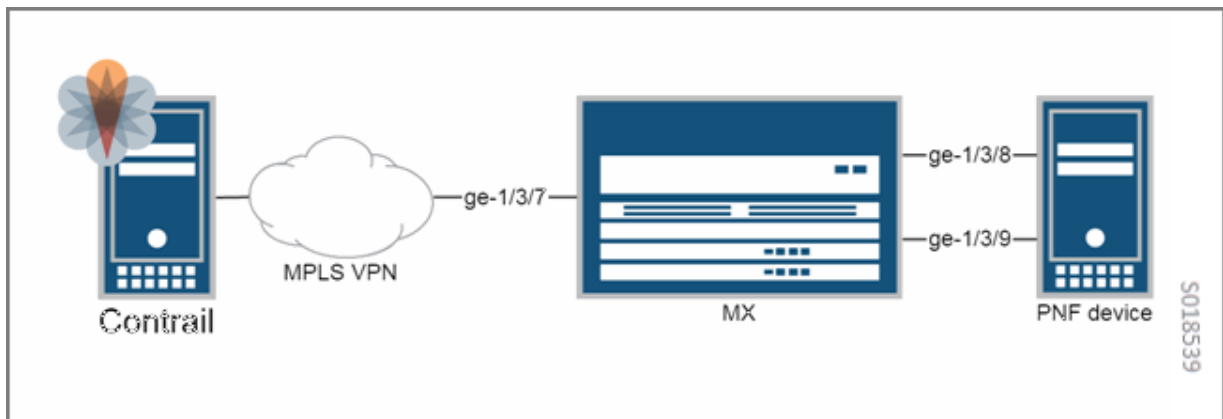
#### Prerequisites

The following are the minimum requirements needed before you can add a PNF to a service chain using the procedure shown in the example included in this topic:

- at least one MX Series device
- at least one PNF to connect to the MX device
- Juniper Networks Junos version that includes the feature `accept-local-nexthop`

**NOTE:** The Junos feature `accept-local-nexthop` is available starting with Junos Release 14.1X55. The Contrail service chain with PNF has been tested on Junos 14.1X55. Contact your Juniper Networks customer service representative for more information.

The prerequisite minimum topology is shown in the following figure.



The following must be preconfigured on the MX Series device.

```
interfaces {
  ge-1/3/7 {
    unit 0 {
      family inet {
        address 10.227.5.115/24;
      }
      family mpls;
    }
  }
  ge-1/3/9 {
    vlan-tagging;
  }
  ge-1/3/8 {
    vlan-tagging;
  }
}
protocols {
  bgp {
    family inet-vpn {
      unicast {
        accept-local-nextHop;
      }
    }
  }
}
```

If the MX is a service control gateway (SCG), the following configuration must also be present to support the service subscriptions:

```
firewall {
  family inet {
    filter skip_tdf_service {
      term term1 {
        then {
          skip-services;
          accept;
        }
      }
    }
  }
}
```



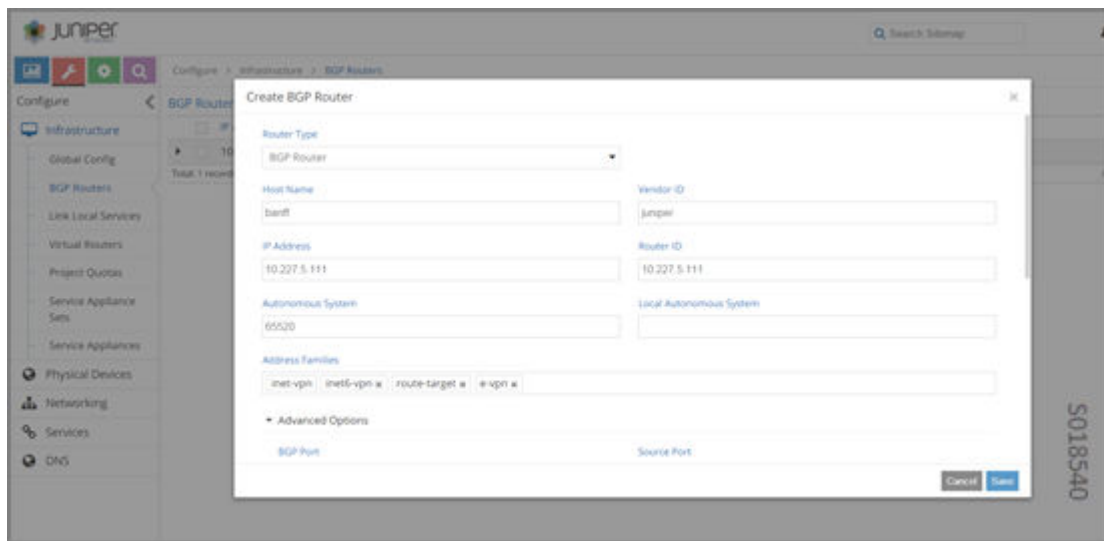
```

    }
  }
  routing-instances {
    <*-sc-entry-point> {
      forwarding-options {
        family inet {
          filter {
            input skip_tdf_service;
          }
        }
      }
    }
  }
}

```

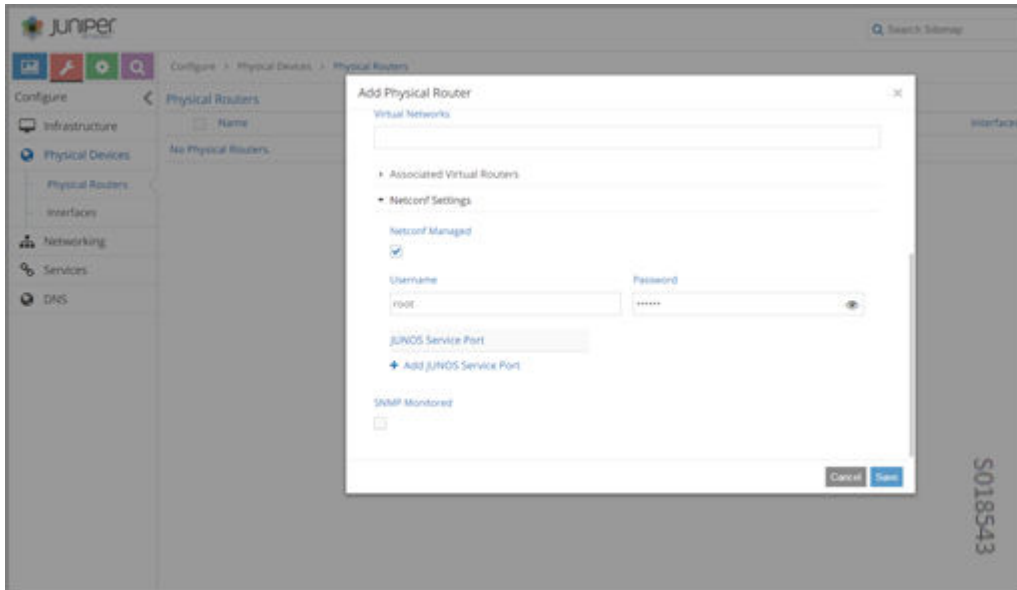
### Procedure: Adding a PNF to a Service Chain

1. At the Contrail UI, **Configure > Infrastructure > BGP Routers**, create a BGP router, with the Contrail controller as a peer, the address family you need, and a minimum configuration of the route-target, inet, and the inet-vpn. The following figure provides an example.

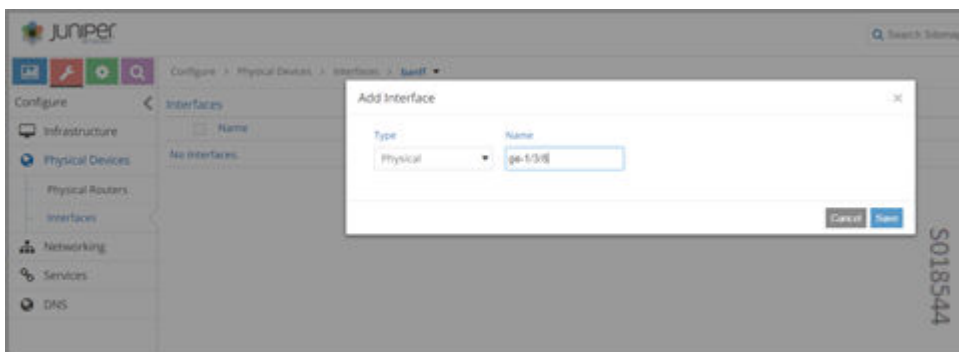


2. Create two virtual networks. Select **Configure > Networking > Networks** and create a network named **IN** and a network named **OUT**. The following figure provides an example.

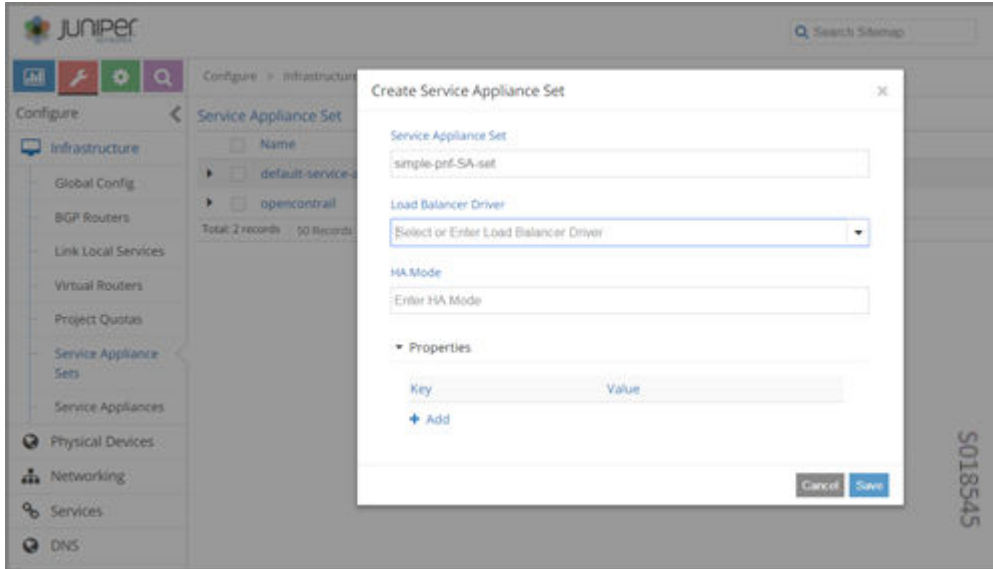




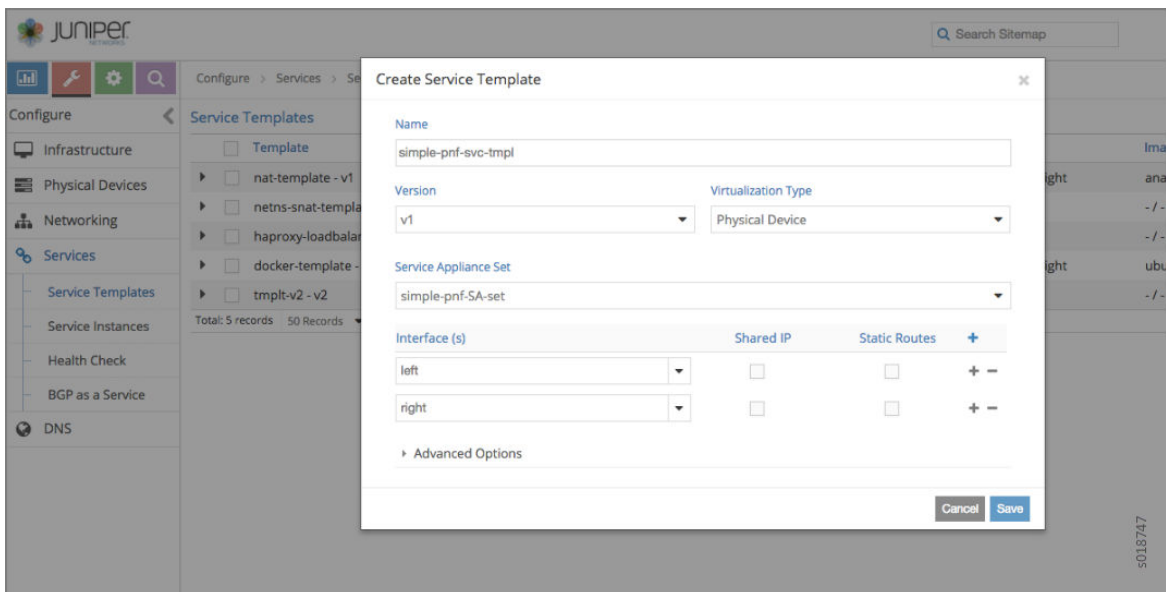
5. Add the physical interfaces that connect to the PNF device. Go to **Configure > Physical Devices > Interfaces** and select the PNF to get to the **Add Interfaces** window, where you enter the name and type for each interface. The following figure provides an example.



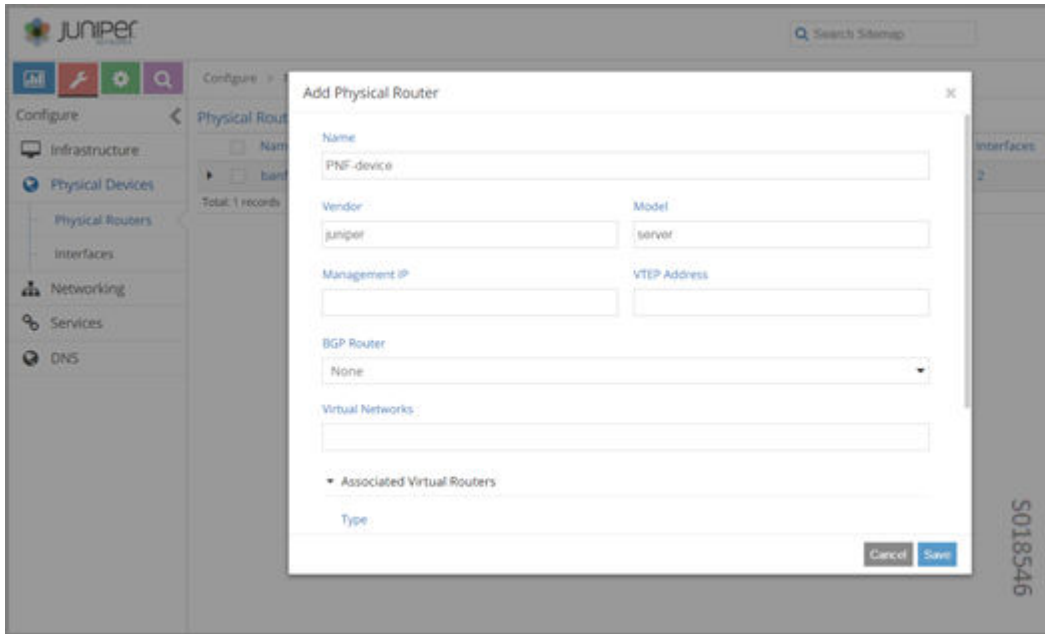
6. Add a service appliance set. Go to **Configure > Infrastructure > Service Appliance Sets** to get to the **Create Service Appliance Set** window, where you enter the name of the service appliance set. The following figure provides an example.



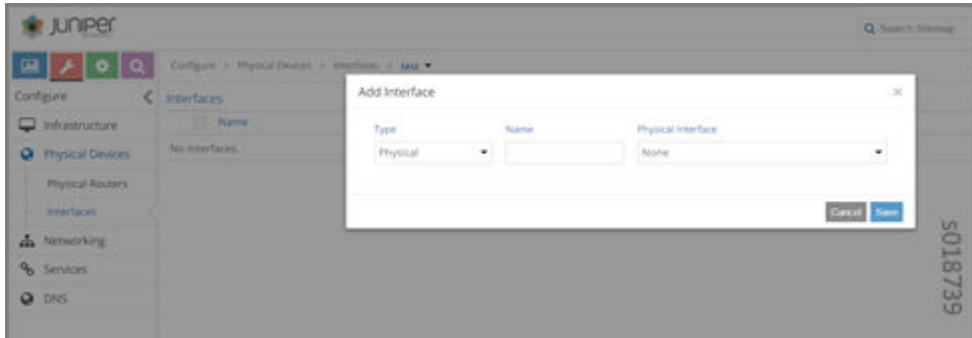
- Configure a service template, **Configure > Services > Service Templates** and click the **Create** button on **Service Templates** to get to **Add Service Template**. Ensure that the **Virtualization Type** is set to **Physical Device**, and that the template is associated to the service appliance set previously created. The following figure provides an example.



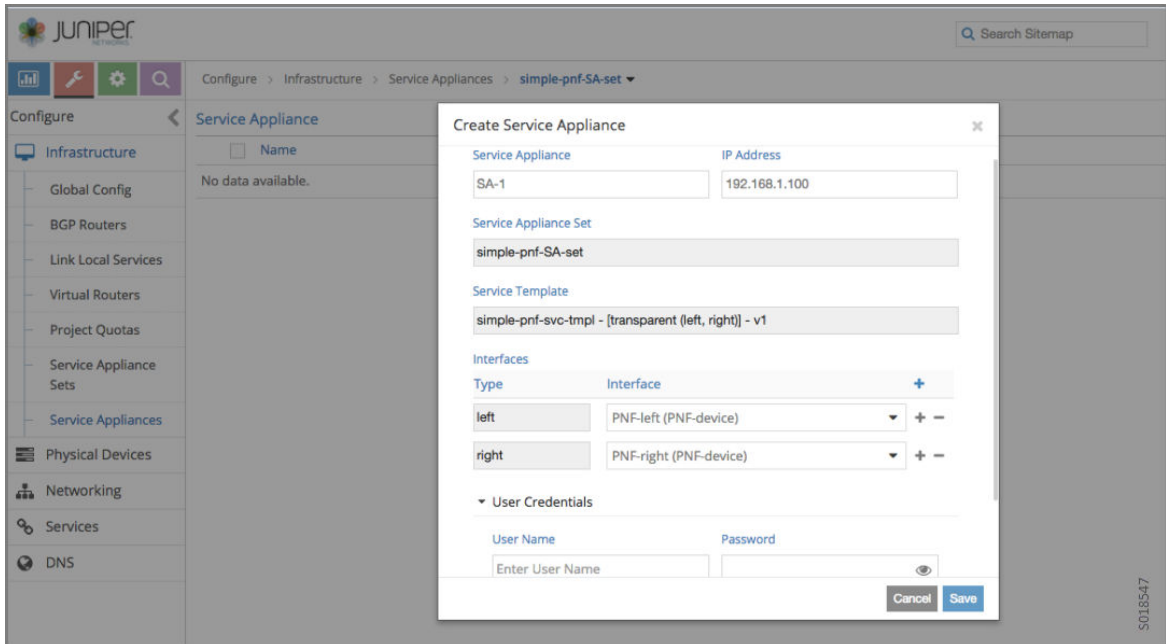
- Add a physical router that represents the PNF device. Go to **Configure > Physical Devices > Physical Routers** to get to the **Add Physical Router** window, where you enter a name for the physical router. The following figure provides an example.



9. Create two interfaces for the PNF. The interfaces should connect to the interfaces already created in this example, and should connect in the manner illustrated in the topology diagram. The interfaces for the other PR should be available from the selection field. The following figure provides an example.



10. Add a service appliance in the service appliance set. Go to **Configure > Infrastructure > Service Appliances** to get to the **Create Service Appliance** window, where you enter the name of the service appliance set and the IP address. Also add the left and right interfaces previously created. The following figure provides an example.



The remaining steps are the same as the steps to create a Contrail service chain, and are summarized in the following steps.

For more details about service chains, see:

- *Service Chaining*
- *Example: Creating a Transparent Service Chain*

11. Create a PNF service instance, go to **Configure > Services > Service Instances**, and click **Create**, then select the template to use and select the corresponding left, right, or management networks. When using a transparent service chain, the VN for the interfaces can be automatic.
12. Add a network policy to connect the virtual networks created for this example, go to **Configure > Networking > Policies**.
13. Associate the policy to both the left VN and the right VN (**IN** and **OUT** in this example). Navigate to **Configure > Networking > Network**.

## RELATED DOCUMENTATION

*Service Chaining*

*Example: Creating a Transparent Service Chain*

[Using Physical Network Functions in Contrail Service Chains](#) | 701

# Configuring High Availability

## IN THIS CHAPTER

- [Juniper OpenStack High Availability | 711](#)
- [High Availability Support Options | 719](#)
- [High Availability for Containerized Contrail | 723](#)

## Juniper OpenStack High Availability

### IN THIS SECTION

- [Introduction | 712](#)
- [Contrail High Availability | 712](#)
- [OpenStack High Availability | 712](#)
- [Supported Platforms | 712](#)
- [Juniper OpenStack High Availability Architecture | 713](#)
- [Juniper OpenStack Objectives | 713](#)
- [Limitations | 714](#)
- [Solution Components | 714](#)
- [Virtual IP with Load Balancing | 714](#)
- [Failure Handling | 715](#)
- [Deployment | 716](#)
- [Minimum Hardware Requirement | 716](#)
- [Compute | 716](#)
- [Network | 716](#)
- [Installation | 717](#)

## Introduction

The Juniper Networks software-defined network (SDN) controller has two major components: OpenStack and Contrail. High availability (HA) of the controller requires that both OpenStack and Contrail are resistant to failures. Failures can range from a service instance failure, node failure, link failure, to all nodes down due to a power outage. The basic expectation from a highly available SDN controller is that when failures occur, already provisioned workloads continue to work as expected without any traffic drop, and the controller is available to perform operations on the cluster. Juniper Networks OpenStack is a distribution from Juniper Networks that combines OpenStack and Contrail into one product.

## Contrail High Availability

Contrail has high availability already built into various components, including support for the Active-Active model of high availability, which works by deploying the Contrail node component with an appropriate required level of redundancy.

The Contrail control node runs BGP and maintains adjacency with the vRouter module in the compute nodes. Additionally, every vRouter maintains a connection with all available control nodes.

Contrail uses Cassandra as the database. Cassandra inherently supports fault tolerance and replicates data across the nodes participating in the cluster.

A highly available deployment of Contrail, at minimum, requires at least:

- **two** control nodes
- **three** config nodes (including analytics and webui)
- **three** database nodes

## OpenStack High Availability

High availability of OpenStack is supported by deploying the OpenStack controller nodes in a redundant manner on multiple nodes. Previous releases of Contrail supported only a single instance of the OpenStack controller, and multiple instances of OpenStack posed new problems that needed to be solved, including:

- State synchronization of stateful services (e.g. MySQL) across multiple instances.
- Load-balancing of requests across the multiple instances of services.

## Supported Platforms

Juniper OpenStack Controller has tested high availability on the following platforms:

- Linux – Ubuntu 12.04 with kernel version 3.13.0-34



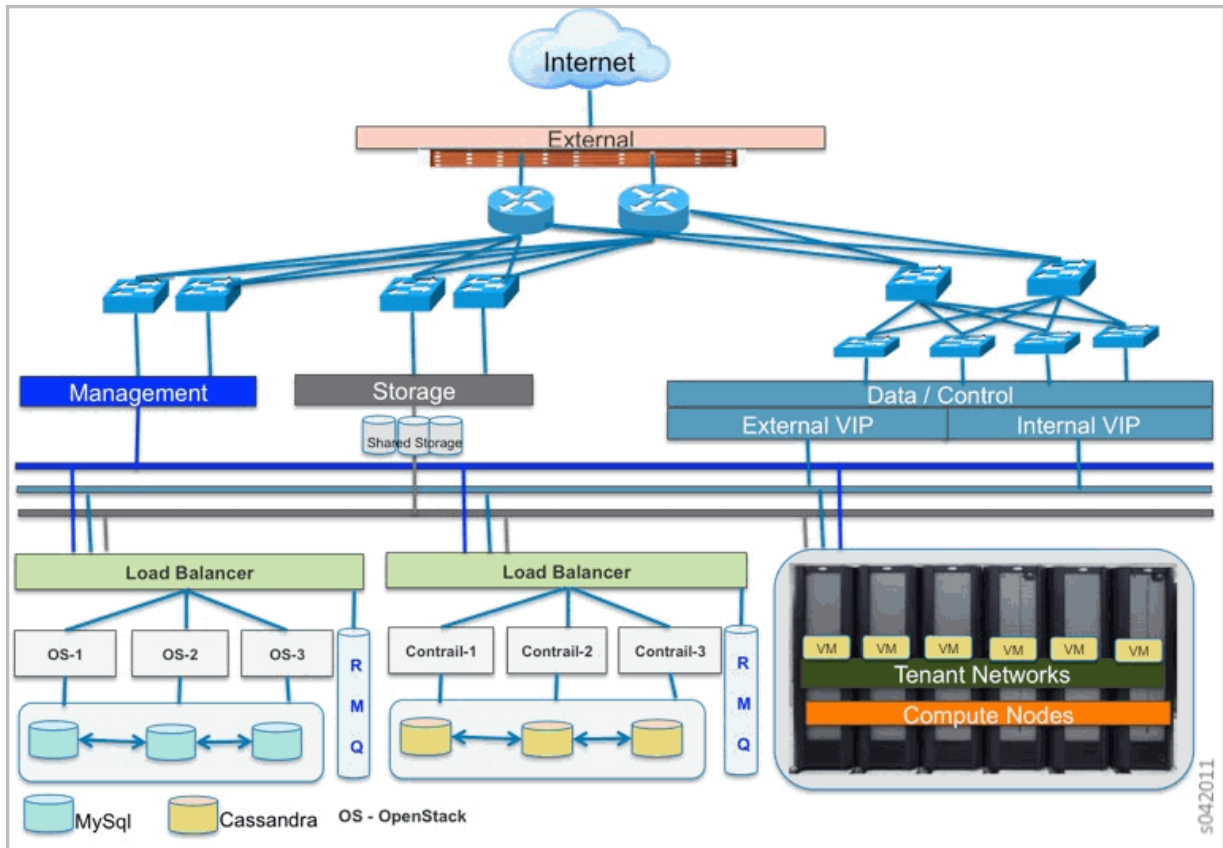
- Ubuntu Server 16.04 LTS (Xenial Xerus)

For a list of all operating system versions and the corresponding Linux or Ubuntu kernel versions supported by Contrail Release 4.1, see *Supported Platforms Contrail 4.1*.

## Juniper OpenStack High Availability Architecture

A typical cloud infrastructure deployment consists of a pool of resources of compute, storage, and networking infrastructure, all managed by a cluster of controller nodes.

The following figure illustrates a high-level reference architecture of a high availability deployment using Juniper OpenStack deployed as a cluster of controller nodes.



## Juniper OpenStack Objectives

The main objectives and requirements for Juniper OpenStack high availability are:

- 99.999% availability for tenant traffic.
- Anytime availability for cloud operations.
- Provide VIP-based access to the API and UI services.
- Load balance network operations across the cluster.

- Management and orchestration elasticity.
- Failure detection and recovery.

## Limitations

The following are limitations of Juniper OpenStack high availability:

- Only one failure is supported.
- During failover, a REST API call may fail. The application or user must reattempt the call.
- Although zero packet drop is the objective, in a distributed system such as Contrail, a few packets may drop during ungraceful failures.
- Juniper OpenStack high availability is not tested with any third party load balancing solution other than HAProxy.

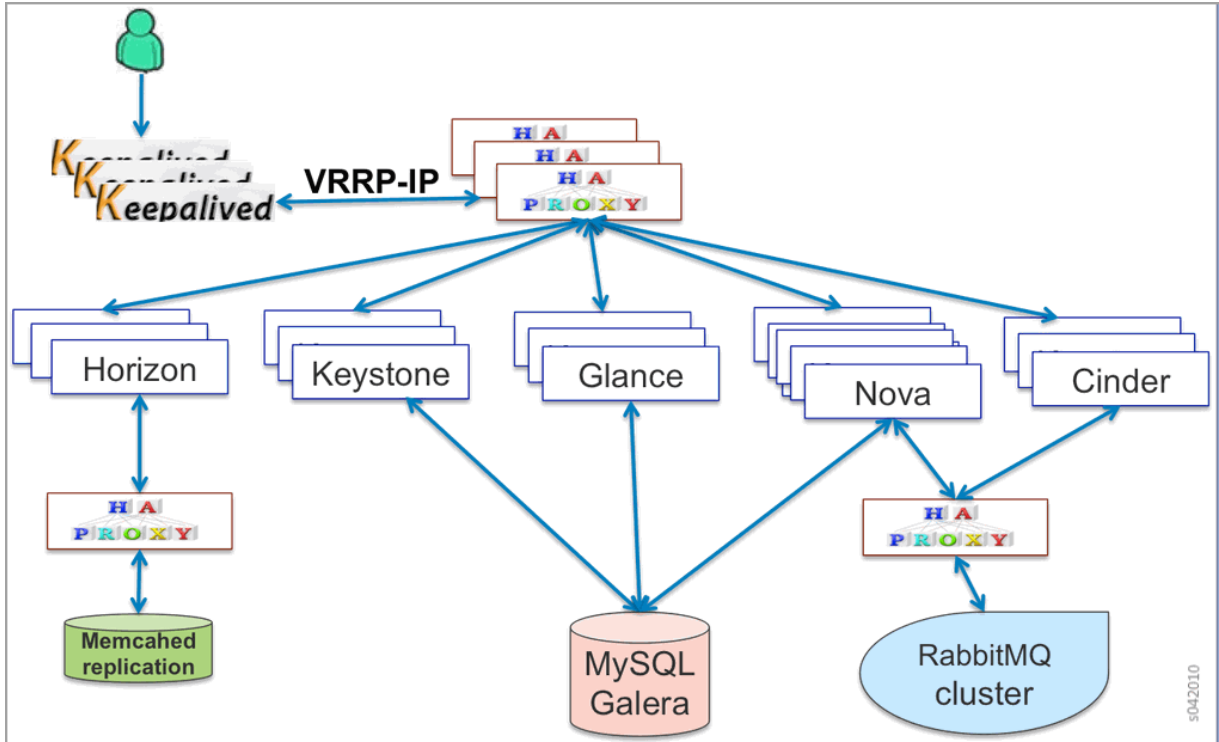
## Solution Components

Juniper Openstack's high availability active-active model provides scale out of the infrastructure and orchestration services. The model makes it very easy to introduce new services in the controller and in the orchestration layer.

## Virtual IP with Load Balancing

HAProxy is run on all nodes to load balance the connections across multiple instances of the services. To provide a Virtual IP (VIP), Keepalived (open source health check framework and hot standby protocol) runs and elects a master based on VRRP protocol. The VRRP master owns the VIP. If the master node fails, the VIP moves to a new master elected by VRRP.

The following figure shows OpenStack services provisioned to work with HAProxy and Keepalived, with HAProxy at the front of OpenStack services in a multiple operating system node deployment. The OpenStack database is deployed in clustered mode and uses Galera for replicating data across the cluster. RabbitMQ has clustering enabled as part of a multinode Contrail deployment. The RabbitMQ configuration is further tuned to support high availability.



## Failure Handling

This section describes how various types of failures are handled, including:

- Service failures
- Node failures
- Networking failures

### *Service Failures*

When an instance of a service fails, HAProxy detects the failure and load balances any subsequent requests across other active instances of the service. The supervisor process monitors for service failures and brings up the failed instances. As long as there is one instance of a service operational, the Juniper OpenStack controller continues to operate. This is true for both stateful and stateless services across Contrail and OpenStack.

### *Node Failures*

The Juniper OpenStack controller supports single node failures involving both graceful shutdown or reboots and ungraceful power failures. When a node that is the VIP master fails, the VIP moves to the next active node, as it is elected to be the VRRP master. HAProxy on the new VIP master sprays the connections over to the active service instances as before, while the failed down node is brought back online. Stateful services (MySQL, Galera, Zookeeper, and so on) require a quorum to be maintained when a node fails. As long as a quorum is maintained, the controller cluster continues to work without

problems. Data integrity is also inherently preserved by Galera, Rabbit, and other stateful components in use.

### *Network Failures*

A connectivity break, especially in the control data network causes the controller cluster to partition into two. As long as the caveat of minimum number of nodes is maintained for one of the partitions, the controller cluster continues to work. Stateful services detect the partitioning and reorganize their cluster around the reachable nodes. Existing workloads continue to function and pass traffic and new workloads can be provisioned. When the connectivity is restored, the joining node becomes part of the working cluster and the system gets restored to its original state.

## **Deployment**

### **Minimum Hardware Requirement**

A minimum of 3 servers (physical or virtual machines) are required to deploy a highly available Juniper OpenStack Controller. In Active-Active mode, the controller cluster uses Quorum-based consistency management for guaranteeing transaction integrity across its distributed nodes. This translates to the requirement of deploying  $2n+1$  nodes to tolerate  $n$  failures.

The Juniper OpenStack Controller offers a variety of deployment choices. Depending on the use case, the roles can be deployed either independently or in some combined manner. The type of deployment determines the sizing of the infrastructure. The numbers below present minimum requirements across compute, storage, and network.

### **Compute**

- Quad core Intel(R) Xeon 2.5 Gz or higher
- 32 GB or higher RAM for the controller hosts (increases with number of hypervisors being supported)
- Minimum 1 TB disk, SSD, HDD

### **Network**

A typical deployment separates control data traffic from the management traffic.

- Dual 10 GE that is bonded (using LAG 802.3ad) for redundant control data connection.
- Dual 1 GE bonded (using LAG 802.3 ad) for redundant management connection.
- Single 10G and 1G can be used if link redundancy is not desired.

The deployment needs virtual IP (VIP) addresses from the networks in which the NICs participate, external VIP on the management network and internal VIP on the control data network. External facing

services are load balanced using the external VIP and the internal VIP is used for communication between other services.

### *Packaging*

High availability support requires new components in the Contrail OpenStack deployment, which are packaged in `contrail-openstack-ha`, including HAProxy, Keepalived, Galera, and their requisite dependencies.

## **Installation**

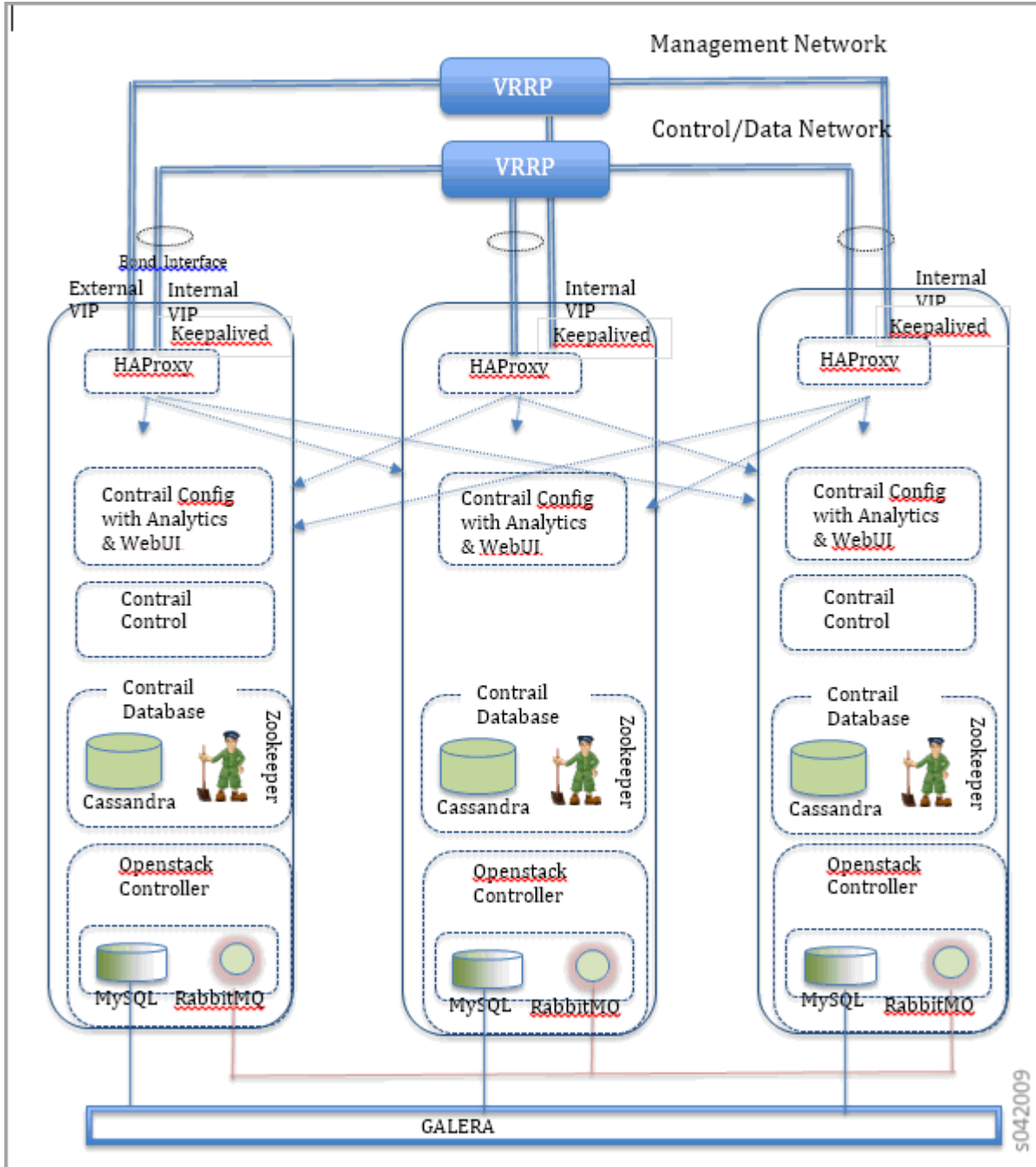
Installation is supported through provisioning. The provisioning file has parameters specifying external and internal VIPs. If OpenStack and Contrail roles are co-located on the nodes, only one set of external and internal VIPs is needed.

Install also supports separating OpenStack and Contrail roles on physically different servers. In this case, the external and internal VIPs specified are used for the OpenStack controller, and a separate set of VIPs, `contrail_external_vip` and `contrail_internal_vip`, are used for the Contrail controller nodes. It is also possible to specify separate RabbitMQs for OpenStack and Contrail controllers.

The following services are configured during high availability-enabled provisioning:

- Keepalived — Configures VRRP and VIP using `keepalived` package
- high availability proxy — Configured to load balance among services running on different nodes
- Galera — Openstack MySQL clustering to achieve high availability
- Glance — Support NFS server storage for glance images

Starting with Contrail Release 4.0, provisioning scripts use VIPs instead of the physical IP of the node in all OpenStack and Contrail configuration files. The following figure shows a typical three-node deployment, where Openstack and Contrail roles are co-located on three servers.



RELATED DOCUMENTATION

High Availability Support Options | 719

## High Availability Support Options

### IN THIS SECTION

- [Contrail High Availability Features | 719](#)
- [Configuration Options for Enabling Contrail High Availability | 719](#)
- [Supported Cluster Topologies for High Availability | 720](#)
- [Deploying OpenStack and Contrail on the Same Highly Available Nodes | 721](#)
- [Deploying OpenStack and Contrail on Different High Available Nodes | 721](#)
- [Deploying Contrail Only on High Available Nodes | 722](#)

This section describes how to set up Contrail options for high availability support.

### Contrail High Availability Features

The Contrail OpenStack high availability design and implementation provides:

- A high availability active-active implementation for scale-out of the cloud operation and for flexibility to expand the controller nodes to service the compute fabric.
- Anytime availability of the cloud for operations, monitoring, and workload monitoring and management.
- Self-healing of the service and states.
- VIP-based access to the cloud operations API provides an easy way to introduce new controllers and an API to the cluster with zero downtime. Improved capital efficiencies compared with dedicated hardware implementations, by using nodes assigned to controllers and making them a federated node in the cluster.
- Operational load distribution across the nodes in the cluster.

For more details about high availability implementation in Contrail, see ["Juniper OpenStack High Availability" on page 711](#).

### Configuration Options for Enabling Contrail High Availability

The following are options available to configure high availability within the Contrail configuration file (testbed.py).

| Option                | Description   |
|-----------------------|---|
| internal_vip          | The virtual IP of the OpenStack high availability nodes in the control data network. In a single interface setup, the internal_vip will be in the management data control network.                |
| external_vip          | The virtual IP of the OpenStack high availability nodes in the management network. In a single interface setup, the external_vip is not required.   |
| contrail_internal_vip | The virtual IP of the Contrail high availability nodes in the control data network. In a single interface setup, the contrail_internal_vip will be in the management data control network.        |
| contrail_external_vip | The virtual IP of the Contrail high availability nodes in the management network. In a single interface setup, the contrail_external_vip is not required.   |
| nfs_server            | The IP address of the NFS server that will be mounted to <b>/var/lib/glance/images</b> for the openstack node. The default is to <code>env.roledefs['compute'][0]</code> .                        |
| nfs_glance_path       | The NFS server path to save images. The default is to <b>/var/tmp/glance-images/</b> .  |
| openstack_manage_amqp | A flag to indicate the node on which rabbitmq is set up. True indicates rabbitmq is setup on OpenStack nodes. False indicates rabbitmq is set up on OpenStack nodes and the controller container. |

## Supported Cluster Topologies for High Availability

This section describes configurations for the cluster topologies supported, including:

- OpenStack and Contrail on the same highly available nodes
- OpenStack and Contrail on different highly available nodes
- Contrail only on highly available nodes



## Deploying OpenStack and Contrail on the Same Highly Available Nodes

OpenStack and Contrail services can be deployed in the same set of highly available nodes by setting the `internal_vip` parameter in the cluster configuration.

Because the high available nodes are shared by both OpenStack and Contrail services, it is sufficient to specify only `internal_vip`. However, if the nodes have multiple interfaces with management and data control traffic separated by provisioning multiple interfaces, then the `external_vip` also needs to be set in the cluster configuration.

### Example

```
env.ha = {  
  
    'internal_vip' : 'an-ip-in-control-data-network',  
  
    'external_vip' : 'an-ip-in-management-network',  
  
}
```

## Deploying OpenStack and Contrail on Different High Available Nodes

OpenStack and Contrail services can be deployed on different high available nodes by setting the `internal_vip` and the `contrail_internal_vip` parameter in the cluster configuration.

Because the OpenStack and Contrail services use different high available nodes, it is required to separately specify `internal_vip` for OpenStack high available nodes and `contrail_internal_vip` for Contrail high available nodes. If the nodes have multiple interfaces, with management and data control traffic separated by provisioning multiple interfaces, then the `external_vip` and `contrail_external_vip` options also must be set in the cluster configuration.

### Example

```
env.ha = {  
  
    'internal_vip' : 'an-ip-in-control-data-network',  
  
    'external_vip' : 'an-ip-in-management-network',  
  
    'contrail_internal_vip' : 'another-ip-in-control-data-network',  
  
    'contrail_external_vip' : 'another-ip-in-management-network',  
  
}
```

```
}

```

By default, the `rabbitmq` cluster is configured on OpenStack nodes. To manage a separate `rabbitmq` cluster for Contrail services, set the `openstack_manage_amqp` to `false` in the cluster configuration. In this case, OpenStack services use the `rabbitmq` cluster on OpenStack nodes and Contrail services use `rabbitmq` cluster on controller containers.

#### Example:

```
"openstack":{
    "openstack_manage_amqp": false
}
```

## Deploying Contrail Only on High Available Nodes

Contrail services can be deployed only on a set of high available nodes by setting the `contrail_internal_vip` parameter in the cluster configuration.

Because the high available nodes are used by only Contrail services, it is sufficient to specify only `contrail_internal_vip`. If the nodes have multiple interfaces with management and data control traffic are separated by provisioning multiple interfaces, the `contrail_external_vip` also needs to be set in the cluster configuration.

#### Example

```
env.ha = {
    'contrail_internal_vip' : 'an-ip-in-control-data-network',
    'contrail_external_vip' : 'an-ip-in-management-network',
}
```

By default, the `rabbitmq` cluster is configured on OpenStack nodes. To manage a separate `rabbitmq` cluster for Contrail services, set the `openstack_manage_amqp` to `false` in the cluster configuration. In this case, OpenStack services use the `rabbitmq` cluster on OpenStack nodes and Contrail services use `rabbitmq` cluster on controller containers.

**Example:**

```
"openstack":{  
    "openstack_manage_amqp": false  
}
```

**RELATED DOCUMENTATION**

| [Juniper OpenStack High Availability | 711](#)

## High Availability for Containerized Contrail

**IN THIS SECTION**

- [Containers for High Availability | 723](#)
- [How High Availability is Handled in Contrail Containers | 724](#)

Starting with Contrail 4.0, some modules of Contrail have been grouped by function and packaged in Docker containers. This document describes the Contrail container subsystems that can be deployed in high availability mode.

### Containers for High Availability

The following Contrail container subsystems can be deployed in high availability mode:

- contrail-lb
- contrail-controller
- contrail-analytics
- contrail-analyticsdb

## How High Availability is Handled in Conrail Containers

This section describes the mechanisms for accomplishing high availability in each of the Conrail containers.

### **conrail-lb**

The Conrail load balancer container `conrail-lb` runs HAProxy, a de facto standard open source load balancer, and the BIRD protocol internet routing daemon.

- HAProxy is used to load balance across multiple instances of Conrail services.
- BIRD is used to deploy `conrail-lb` in high availability mode.

Although the `conrail-lb` containers are expected to be deployed in different hosts, they can also be deployed in the same hosts where `conrail-controller` containers are deployed.

For more information about BIRD, see [BIRD Babel protocol documentation](#).

### **High Availability for conrail-controller Container**

The `conrail-controller` container runs the following services that can be scaled or clustered.

- `conrail-api`—Can be scaled up. It is load-balanced by HAProxy running in the `conrail-lb` containers. All clients can connect to the load-balancer IP to communicate with `conrail-lb`.
- `cassandra`—Clustered. The `cassandra` and client libraries have built-in load-balancing and failure detection mechanisms, and there is no need for `cassandra` to be behind HAProxy. However, multiple instances are clustered during deployment of the `conrail-controller` containers. All clients connect to the list of `conrail-controller` container IPs to communicate with `cassandra`.
- `zookeeper`—Clustered. The `zookeeper` and client libraries have a built-in high availability using leader-follower architecture, and there is no need for `zookeeper` to be behind HAProxy. However, multiple instances are clustered during deployment of the `conrail-controller` containers. All clients connect to the list of `conrail-controller` container IPs to communicate with `zookeeper`.
- `rabbitmq`—Clustered. The `rabbitmq` and client libraries can handle multiple `rabbitmq` instances. However, multiple instances are clustered and mirrored in queues during deployment of the `conrail-controller` containers. All clients connect to the list of `conrail-controller` container IPs to communicate with `rabbitmq`.

**NOTE:** Only an *odd* number of contrail-controllers are supported, due to a limitation with zookeeper for leader-follower election.

### **contrail-analytics**

The contrail-analytics container runs the following services that can be scaled or clustered.

- `contrail-analytics-api`—Can be scaled up. It is load-balanced by HAProxy running in the `contrail-lb` containers. All clients connect to the load-balancer IP to communicate with `contrail-analytics-api`.

### **contrail-analyticsdb**

The `contrail-analyticsdb` container runs the following services that can be scaled or clustered.

- `cassandra`—Clustered. The `cassandra` and client libraries have built-in load-balancing and failure detection mechanisms, and there is no need for `cassandra` to be behind HAProxy. However, multiple instances are clustered during deployment of the `contrail-controller` containers. All clients connect to the list of `contrail-analyticsdb` container IPs to communicate with `cassandra`.
- `kafka`—Clustered. Kafka uses a zookeeper cluster running in the `contrail-controller` containers. Multiple instances of `kafka` are clustered during deployment of `contrail-analyticsdb` containers. All clients connect to the list of the `contrail-analyticsdb` container IPs to communicate with `kafka`.

## **RELATED DOCUMENTATION**

| [Juniper OpenStack High Availability](#) | 711

# QoS Support in Contrail

## IN THIS CHAPTER

- [Quality of Service in Contrail | 726](#)
- [Configuring Network QoS Parameters | 735](#)
- [BGP as a Service | 737](#)

## Quality of Service in Contrail

### IN THIS SECTION

- [Overview: Quality of Service | 726](#)
- [Contrail QoS Model | 727](#)
- [QoS Configuration Parameters for Provisioning | 727](#)
- [Queuing Implementation | 729](#)
- [Contrail QoS Configuration Objects | 729](#)
- [Example: Mapping Traffic to Forwarding Classes | 731](#)
- [QoS Configuration Object Marking on the Packet | 732](#)
- [Queuing | 733](#)
- [Queue Selection in Datapath | 733](#)
- [Parameters for QoS Scheduling Configuration | 734](#)

### Overview: Quality of Service

Quality of service (QoS) in networking provides the ability to control reliability, bandwidth, latency, and other traffic management features. Network traffic can be marked with QoS bits (DSCP, 802.1p, and MPLS EXP) that intermediate network switches and routers can use to provide service guarantees.

## Contrail QoS Model

The Contrail QoS model has the following features:

- All packet forwarding devices, such as vRouter and the gateway, combine to form a system.
- Interfaces to the system are the ports from which the system sends and receives packets, such as tap interfaces and physical ports.
- Fabric interfaces are where the overlay traffic is tunneled.
- QoS is applied at the ingress to the system, for example, upon traffic from the interfaces to the fabric.
- At egress, packets are stripped of their tunnel headers and sent to interface queues, based on the forwarding class. No marking from the outer packet to the inner packet is considered at this time.

### Features of Fabric Interfaces

Fabric interfaces, unlike other interfaces, are always shared. Therefore, fabric interfaces are common property. Consequently, traffic classes and QoS marking on the fabric must be controlled by the system administrator. The administrator might choose to provision different classes of service on the fabric.

In Contrail, classes of service are determined by both of the following:

- Queueing on the fabric interface, including queues, scheduling of queues, and drop policies, and
- forwarding class, a method of marking that controls how packets are sent to the fabric, including marking and identifying which queue to use.

Tenants can define which forwarding class their traffic can use, deciding which packets use which forwarding class. The Contrail QoS configuration object has a mapping table, mapping the incoming DSCP or 802.1p value to the forwarding class mapping.

The QoS configuration can also be applied to a virtual network, an interface, or a network policy.

## QoS Configuration Parameters for Provisioning

### Testbed.py Parameters

Testbed.py can be used for provisioning Contrail through Releases 3.x.x. Starting with Contrail 4.0, testbed.py can only be used if you are provisioning with SM-Lite. Use parameters in this section if you are using testbed.py for provisioning.

For QoS, the hardware queues (NIC queues) are mapped to logical queues in the agent, using the following keys:

- `hardware_q_id`—Identifier for the hardware queue.
- `logical_queue`— Defines the logical queues to map to each hardware queue.
- `default`—Defines the default hardware queue for QoS when set to `True`.

Options to define a default hardware queue:

- Set the queue as default, without any logical queue mapping.

```
{'hardware_q_id': '1', 'default': 'True'}
```

- Set the hardware queue as default with logical queue mapping.

```
{'hardware_q_id': '6', 'logical_queue':['17-20'], 'default': 'True'}
```

```
env.qos = {host4: [ {'hardware_q_id': '3', 'logical_queue':['1', '6-10', '12-15']},
                   {'hardware_q_id': '5', 'logical_queue':['2']},
                   {'hardware_q_id': '8', 'logical_queue':['3-5']},
                   {'hardware_q_id': '1', 'default': 'True'}],
          host5: [ {'hardware_q_id': '2', 'logical_queue':['1', '3-8', '10-15']},
                   {'hardware_q_id': '6', 'logical_queue':['17-20'], 'default': 'True'}]
}
```

The following are the keys for defining QoS priority groups.

- `priority_id`—Priority group for QoS.
- `scheduling`—Defines the scheduling algorithm used for the priority group, `strict` or `roundrobin (rr)`.
- `bandwidth`—Total hardware queue bandwidth used by priority group.

Bandwidth cannot be specified if `strict` scheduling is used for priority group, so set it to 0.

### Example: QoS Priority Group

```
env.qos_niantic = {host4:[
    { 'priority_id': '1', 'scheduling': 'strict', 'bandwidth': '0'},
    { 'priority_id': '2', 'scheduling': 'rr', 'bandwidth': '20'},
    { 'priority_id': '3', 'scheduling': 'rr', 'bandwidth': '10'}],
  host5:[
    { 'priority_id': '1', 'scheduling': 'strict', 'bandwidth': '0'},
    { 'priority_id': '1', 'scheduling': 'rr', 'bandwidth': '30'}]
}
```



## Queuing Implementation

Starting with Contrail 3.2, queuing is added. The vRouter provides the infrastructure to use queues supplied by the network interface, a method that is also called hardware queueing. Network interface cards (NICs) that implement hardware queueing have their own set of scheduling algorithms associated with the queues. The Contrail implementation is designed to work with most NICs, however, the method is tested only on an Intel-based 10G NIC, also called Niantic.

## QoS Features by Release

QoS features are introduced in the following Contrail releases:

- 3.1—QoS configuration and forwarding classes
- 3.2—queuing
- Not planned—egress marking and queuing

## Contrail QoS Configuration Objects

Contrail QoS configuration objects include the:

- forwarding class
- QoS configuration object (`qos-config`)

The forwarding class object specifies parameters for marking and queuing, including:

- The DSCP, 802.1p, and MPLS EXP values to be written on packets.
- The queue index to be used for the packet.

The QoS configuration object specifies a mapping from DSCP, 802.1p, and MPLS EXP values to the corresponding forwarding class.

The QoS configuration has an option to specify the default forwarding class ID to use to select the forwarding class for all unspecified DSCP, 802.1p, and MPLS EXP values.

If the default forwarding class ID is not specified by the user, it defaults to the forwarding class with ID 0.

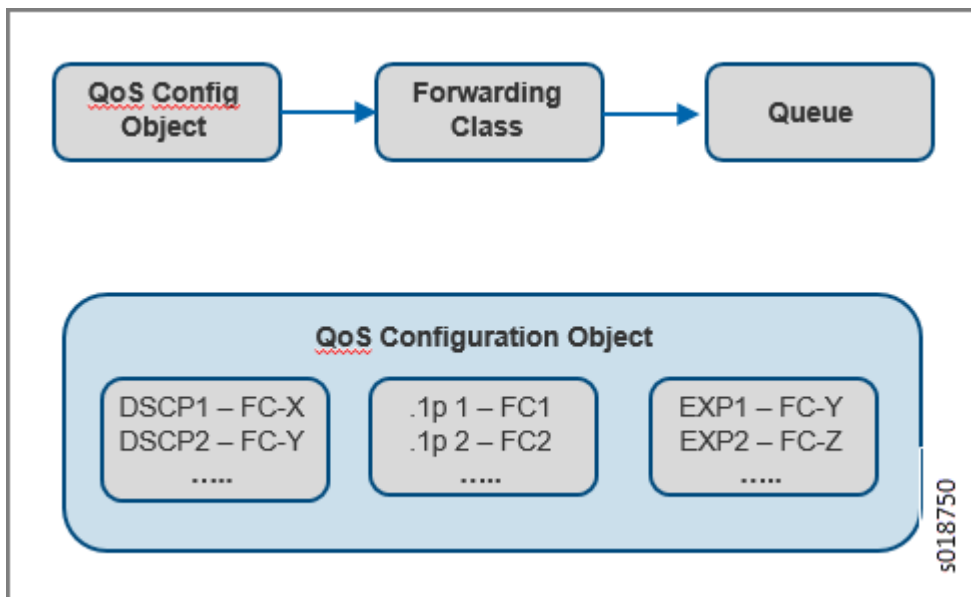
Processing of QoS marked packets to look up the corresponding forwarding class to be applied works as follows:

- For an IP packet, the DSCP map is used .
- For a Layer 2 packet, the 802.1p map is used.

- For an MPLS-tunneled packet with MPLS EXP values specified, the EXP bit value is used with the MPLS EXP map.
- If the QoS configuration is untrusted, only the default forwarding class is specified, and all incoming values of the DSCP, 802.1p, and EXP bits in the packet are mapped to the same default forwarding class.

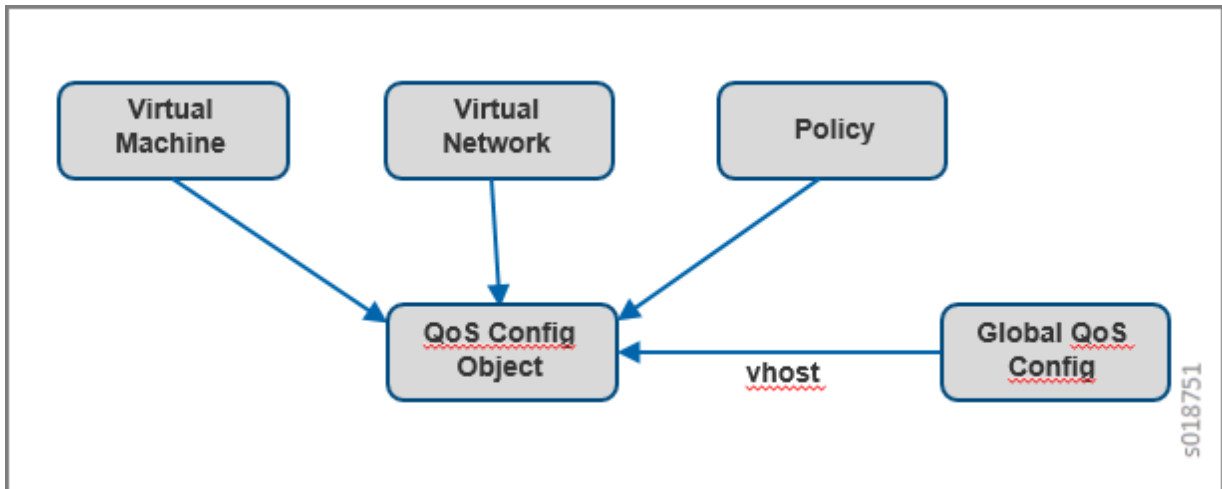
Figure 186 on page 730 shows the processing of QoS packets.

Figure 186: Processing of QoS Packets



A virtual machine interface, virtual network, and network policy can refer to the QoS configuration object. The QoS configuration object can be specified on the vhost so that underlay traffic can also be subjected to marking and queuing. See [Figure 187 on page 731](#).

Figure 187: Referring to the QoS Object



### Example: Mapping Traffic to Forwarding Classes

This example shows how traffic forwarding classes are defined and how the QoS configuration object is defined to map the QoS bits to forwarding classes.

[Table 54 on page 731](#) shows two forwarding class objects defined. FC1 marks the traffic with high priority values and queues it to Queue 0. FC2 marks the traffic as best effort and queues the traffic to Queue 1.

**Table 54: Forwarding Class Mapping**

| Name | ID | DSCP | 802.1p | MPLS EXP | Queue |
|------|----|------|--------|----------|-------|
| FC1  | 1  | 10   | 7      | 7        | 0     |
| FC2  | 2  | 38   | 0      | 0        | 1     |

In [Table 55 on page 732](#), the QoS configuration object DSCP values of 10, 18, and 26 are mapped to a forwarding class with ID 1, which is forwarding class FC1. All other IP packets are mapped to the forwarding class with ID 2, which is FC2. All traffic with an 802.1p value of 6 or 7 are mapped to forwarding class FC1, and the remaining traffic is mapped to FC2.

**Table 55: QoS Configuration Object Mapping**

| DSCP | Forwarding Class ID | 802.1p | Forwarding Class ID | MPLS EXP | Forwarding Class ID |
|------|---------------------|--------|---------------------|----------|---------------------|
| 10   | 1                   | 6      | 1                   | 5        | 1                   |
| 18   | 1                   | 7      | 1                   | 7        | 1                   |
| 26   | 1                   | *      | 2                   | *        | 1                   |
| *    | 2                   |        |                     |          |                     |

## QoS Configuration Object Marking on the Packet

The following describes how QoS configuration object marking is handled in various circumstances.

### Traffic Originated by a Virtual Machine Interface

- If a VM interface sends an IP packet to another VM in a remote compute node, the DSCP value in the IP header is used to look into the qos-config table, and the tunnel header is marked with DSCP, 802.1p, and MPLS EXP bits as specified by the forwarding class.
- If a VM sends a Layer 2 non-IP packet with an 802.1p value, the 802.1p value is used to look into the qos-config table, and the corresponding forwarding class DSCP, 802.1p, and MPLS EXP value is written to the tunnel header.
- If a VM sends an IP packet to a VM in the same compute node, the DSCP value in the IP header is matched in the qos-config table, and the corresponding forwarding class is used to overwrite the IP header with new DSCP and 802.1p values.

### Traffic Destined to a Virtual Machine Interface

For traffic destined to a VMI, if a tunneled packet is received, the tunnel headers are stripped off and the packet is sent to the interface. No marking is done from the outer packet to inner packet.

## Traffic from a vhost Interface

The QoS configuration can be applied on IP traffic coming from a vhost interface. The DSCP value in the packet is used to look into the qos-config object specified on the vhost, and the corresponding forwarding class DSCP and 802.1p values are overwritten on the packet.

## Traffic from fabric interface

The QoS configuration can be applied while receiving the packet on an Ethernet interface of a compute node, and the corresponding forwarding class DSCP and 802.1p values are overwritten on the packet.

## QoS Configuration Priority by Level

The QoS configuration can be specified at different levels.

The levels that can be configured with QoS and their order of priority:

1. in policy
2. on virtual-network
3. on virtual-machine-interface

## Queuing

Contrail Release 3.2 adds QoS support for queuing.

This section provides an overview of the queuing features available starting with Contrail 3.2.

For more details about any of these topics, see: <https://github.com/Juniper/contrail-controller/wiki/QoS>.

The queue to which a packet is sent is specified by the forwarding class.

## Queue Selection in Datapath

In vRouter, in the data path, the forwarding class number specifies the actual physical hardware queue to which the packet needs to be sent, not to a logical selection as in other parts of Contrail. There is a mapping table in the vRouter configuration file, to translate the physical queue number from the logical queue number.

## Hardware Queueing in Linux kernel based vRouter

If Xmit-Packet-Steering (XPS) is enabled, the kernel chooses the queue, from those available in a list of queues. If the kernel selects the queue, packets will not be sent to the vRouter-specified queue.

To disable this mapping:

- have a kernel without CONFIG\_XPS option
- write zeros to the mapping file in `/sys/class/net//queues/tx-X/xps_cpus` .

When this mapping is disabled, the kernel will send packets to the specific hardware queue.

To verify:

See individual queue statistics in the output of 'ethtool -S ' command.

## Parameters for QoS Scheduling Configuration

The following shows sample scheduling configuration for hardware queues on the compute node.

The priority group ID and the corresponding scheduling algorithm and bandwidth to be used by the priority group can be configured.

Possible values for the scheduling algorithm include:

- strict
- rr (round-robin)

When round-robin scheduling is used, the percentage of total hardware queue bandwidth that can be used by the priority group is specified in the bandwidth parameter.

The following configuration and provisioning is applicable only for compute nodes running Niantic NICs and running kernel based vrouter.

```
qos_niantic = {
  'compute1': [
    { 'priority_id': '1', 'scheduling': 'strict', 'bandwidth': '0'},
    { 'priority_id': '2', 'scheduling': 'rr', 'bandwidth': '20'},
    { 'priority_id': '3', 'scheduling': 'rr', 'bandwidth': '10'}
  ],
  'compute2' :[
    { 'priority_id': '1', 'scheduling': 'strict', 'bandwidth': '0'},
    { 'priority_id': '1', 'scheduling': 'rr', 'bandwidth': '30'}
  ]
}
```

```
]
}
```

## RELATED DOCUMENTATION

[Configuring Network QoS Parameters | 735](#)

<https://github.com/Juniper/contrail-controller/wiki/QoS>.

## Configuring Network QoS Parameters

### IN THIS SECTION

- [Overview | 735](#)
- [QoS Configuration Examples | 735](#)
- [Limitations | 737](#)

### Overview

You can use the OpenStack Nova command-line interface (CLI) to specify a quality of service (QoS) setting for a virtual machine's network interface, by setting the quota of a Nova flavor. Any virtual machine created with that Nova flavor will inherit all of the specified QoS settings. Additionally, if the virtual machine that was created with the QoS settings has multiple interfaces in different virtual networks, the same QoS settings will be applied to all of the network interfaces associated with the virtual machine. The QoS settings can be specified in unidirectional or bidirectional mode.

The quota driver in Neutron converts QoS parameters into libvirt network settings of the virtual machine.

The QoS parameters available in the quota driver only cover rate limiting the network interface. There are no specifications available for policy-based QoS at this time.

### QoS Configuration Examples

Although the QoS setting can be specified in quota by using either Horizon or CLI, quota creation using CLI is more robust and stable, therefore, creating by CLI is the recommended method.

### Example

CLI for Nova flavor has the following format:

```
nova flavor-key <flavor_name> set quota:vif_<direction>_<param_name> = value
```

where:

<flavor\_name> is the name of an existing Nova flavor.

vif\_<direction>\_<param\_name> is the inbound or outbound QoS data name.

QoS vif types include the following:

- vif\_inbound\_average lets you specify the average rate of inbound (receive) traffic, in kilobytes/sec.
- vif\_outbound\_average lets you specify the average rate of outbound (transmit) traffic, in kilobytes/sec.
- Optional: vif\_inbound\_peak and vif\_outbound\_peak specify the maximum rate of inbound and outbound traffic, respectively, in kilobytes/sec.
- Optional: vif\_inbound\_burst and vif\_outbound\_peak specify the amount of kilobytes that can be received or transmitted, respectively, in a single burst at the peak rate.

Details for various QoS parameters for libvirt can be found at <http://libvirt.org/formatnetwork.html>.

The following example shows an inbound average of 800 kilobytes/sec, a peak of 1000 kilobytes/sec, and a burst amount of 30 kilobytes.

```
nova flavor-key m1.small set quota:vif_inbound_average=800
nova flavor-key m1.small set quota:vif_inbound_peak=1000
nova flavor-key m1.small set quota:vif_inbound_burst=30
```

The following is an example of specified outbound parameters:

```
nova flavor-key m1.small set quota:vif_outbound_average=800
nova flavor-key m1.small set quota:vif_outbound_peak=1000
nova flavor-key m1.small set quota:vif_outbound_burst=30
```

After the Nova flavor is configured for QoS, a virtual machine instance can be created, using either Horizon or CLI. The instance will have network settings corresponding to the nova flavor-key, as in the following:

```
<interface type="ethernet">
  <mac address="02:a3:a0:87:7f:61"/>
```



```

<model type="virtio"/>
<script path=""/>
<target dev="tapa3a0877f-61"/>
<bandwidth>
  <inbound average="800" peak="1000" burst="30"/>
  <outbound average="800" peak="1000" burst="30"/>
</bandwidth>
</interface>

```

## Limitations

- The stock libvirt does not support rate limiting of ethernet interface types. Consequently, settings like those in the example for the guest interface will not result in any `tc qdisc` settings for the corresponding tap device in the host. For more details, refer to issue [#1367095](#) in [Launchpad.net](#), where you can find patches and instructions to make libvirt work for network rate limiting of virtual machine interfaces.
- The nova flavor-key `rxtx_factor` takes a float as an input and acts as a scaling factor for receive (inbound) and transmit (outbound) throughputs. This key is only available to Neutron extensions (private extensions). The Contrail Neutron plugin doesn't implement this private extension. Consequently, setting the nova flavor-key `rxtx_factor` will not have any effect on the QoS setting of the network interface(s) of any virtual machine created with that nova flavor.
- The outbound rate limits of a virtual machine interface are not strictly achieved. The outbound throughput of a virtual machine network interface is always less than the average outbound limit specified in the virtual machine's libvirt configuration file. The same behavior is also seen when using a Linux bridge.

## RELATED DOCUMENTATION

[Quality of Service in Contrail](#) | 726

## BGP as a Service

### IN THIS SECTION

[Contrail BGPaaS Features](#) | 738

- BGPaaS Customer Use Cases | 739
- Configuring BGPaaS | 740

The BGP as a Service (BGPaaS) feature allows a guest virtual machine (VM) to place routes in its own virtual routing and forwarding (VRF) instance using BGP.

## Contrail BGPaaS Features

Using BGPaaS with Contrail requires the guest VM to have connectivity to the control node and to be able to advertise routes into the VRF instance.

With the BGPaaS feature:

- The vRouter agent is able to accept BGP connections from the VMs and proxy them to the control node.
- The vRouter agent always selects one of the control nodes that it is using as an XMPP server.

Starting with Contrail Release 3.0, the following features have been added to BGPaaS:

- All BGPaaS sessions are configured to have bidirectional exchange of routes.
- If inet6 routes are being advertised to the tenant VM, they are advertised with the IPv6 subnet's default gateway address as the BGP next hop.
- If multiple tenant VMs in the same virtual network have BGPaaS sessions and they use eBGP, standard loop prevention rules prevent routes advertised by one tenant VM from being advertised to other tenant VMs

A second BGP session for high availability can also be configured appropriately using one more BGP router object in the Contrail configuration and the peering session (from the VNF's point of view) to the DNS IP address (reserved by Contrail).

The following are caveats:

- BGP sessions must use IPv4 transport.
- The VNF must support RFC 2545, *Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing*, to carry IPv6 routes over the IPv4 peer.
- Only IPv4 (inet) and IPv6 (inet6) address families are supported.

The initial implementation of BGPaaS Version 1, supported in Contrail Release 3.0, allowed a tenant VM to establish BGP sessions to the default gateway and DNS server in the VM's subnet. A limitation of this

implementation was that the tenant VM could advertise routes into the virtual network to which the VM belonged, however, the VM could not receive any routes. The tenant VM was required to use a static default route, with the subnet's default gateway as the next hop.

Contrail Release 3.1 eliminates the previous limitation and provides route export functionality for BGPaaS sessions. The next hop for all routes advertised to the tenant VM is set to the default gateway address of the subnet of the tenant VM. This allows the tenant BGP implementation to be relatively simple, by not requiring support for recursive resolution of BGP next hops.

The BGPaaS object is associated with a virtual machine interface (VMI), not just a virtual machine (VM), which enables a tenant VM to have BGP sessions in multiple virtual networks, if required.

Starting with Contrail Release 3.1, the following features and properties have been added to BGPaaS:

- By default, all BGPaaS sessions are configured to have bidirectional exchange of routes. The Boolean property `bgpaas-suppress-route-advertisement` ensures no advertisement of routes to the tenant VM.
- If `inet6` routes are being advertised to the tenant VM, they are advertised with the IPv6 subnet's default gateway address as the BGP next hop. A Boolean property, `bgpaas-ipv4-mapped-ipv6-nexthop`, causes the IPv4 subnet's default gateway, in IPv4-mapped IPv6 format, to be used instead as the next hop.
- If multiple tenant VMs in the same virtual network have BGPaaS sessions and they use eBGP, the standard BGP AS path loop prevention rules prevent routes advertised by one tenant VM from being advertised to the other tenant VMs. The `as-override` field, added to the existing `BgpSessionAttributes` in the BGPaaS object, causes the control node to replace the AS number of the tenant VM with its own AS number, when advertising routes learned from a tenant VM to another tenant VM in the same virtual network. The tenant VM does not need to implement any new functionality.

## BGPaaS Customer Use Cases

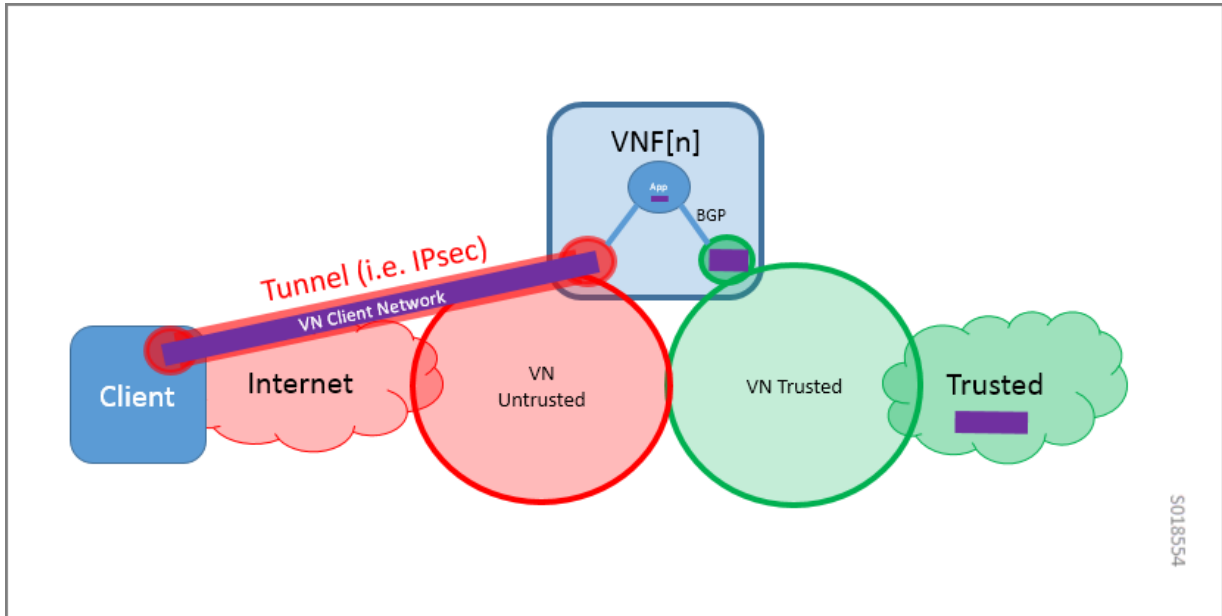
This section provides example scenarios for implementing BGPaaS with Contrail.

### Dynamic Tunnel Insertion Within a Tenant Overlay

Various applications need to insert dynamic tunnels into virtual networks. Virtual network functions (VNFs) provide the function of tunnel termination. Tunnel termination types vary across application types, such as business VPN, mobility small site backhaul, VPC, and the like. The key requirement is that tunnels need to insert dynamically new network reachability information into the virtual network. The predominant methods of tunnel network reachability insertion use BGP.

BGPaaS allows the migration of brownfield VNFs into Contrail, preserving the application behavior and requirement for BGP, without rewriting the application.

The following figure is a generic example showing the need to insert a dynamic tunnel into a virtual network.



### Dynamic Network Reachability of Applications

The Domain Name System (DNS) is a widespread application that uses BGP as a mechanism to tune reachability of its services, based on metrics such as load, maintenance, availability, and the like. As DNS services are migrated to environments using overlays, a mechanism to preserve the existing application behavior and requirements is needed, including the ability to announce and withdraw reachability to the available application.

This requirement is not limited to DNS. Other applications, such as virtualized evolved packet core (vEPC) and others, use BGP as a mechanism for network reachability based on availability and load.

### Liveness Detection for High Availability

Various keepalive mechanisms for tenant reachability have been provided by network components such as BGP, OSPF, PING, VRRP, BFD, or application-specific mechanisms. With BGP on the vRouter agent, BGP can be used to provide a liveness detection mechanism between the tenant on the local compute node and the services that the specific tenant VM is providing.

### Configuring BGPaaS

The following are methods for configuring BGPaaS:

## Configuring BGPaaS Using VNC API

To use VNC APIs to configure BGPaaS:

1. Access the default project.

```
default_project = self._vnc_lib.project_read(fq_name=[u'default-domain', 'bgpaas-tenant'])
```

2. Create a BGPaaS object.

```
bgpaas_obj = BgpAsAService(name='bgpaas_1', parent_obj=default_project)
```

3. Attach the BGP object to a precreated VMI.

```
bgpaas_obj.add_virtual_machine_interface(vmi)
```

4. Set the ASN. It must be an eBGP session.

```
bgpaas_obj.set_autonomous_system('65000')
```

If the ASN is not set, the primary instance IP will be chosen.

```
bgpaas_obj.set_bgpaas_ip_address(u'10.1.1.5')
```

5. Set session attributes.

```
bgp_addr_fams = AddressFamilies(['inet', 'inet6'])
bgp_sess_attrs = BgpSessionAttributes(address_families=bgp_addr_fams, hold_time=60)
bgpaas_obj.set_bgpaas_session_attributes(bgp_sess_attrs)
self._vnc_lib.bgp_as_a_service_create(bgpaas_obj)
```

## Deleting a BGPaaS Object

To delete a BGPaaS object:

```
fq_name=[u'default-domain', 'bgpaas-tenant', 'bgpaas_1']
bgpaas_obj = self._vnc_lib.bgp_as_a_service_read(fq_name=fq_name)
bgpaas_obj.del_virtual_machine_interface(vmi)
self._vnc_lib.bgp_as_a_service_update(bgpaas_obj)
self._vnc_lib.bgp_as_a_service_delete(id=bgpaas_obj.get_uuid())
```

## Using the Contrail User Interface to Configure BGPaaS

To configure BGPaaS within a tenant:

1. Within a tenant in Contrail, navigate to **Configure > Services > BGP as a Service**. Select the + icon to access the window **Create BGP as a Service**.

The screenshot shows the Juniper configuration interface with a 'Create BGP as a Service' dialog box open. The dialog box contains the following fields and values:

- Name:** bgpaas-1
- IP Address:** 11.95.197.1
- Autonomous System:** 50000
- Address Family:** inet x inet6 x
- Virtual Machine Interface(s):** 7d8fb01c-26af-4128-b030-ab446d3da0e5 (11.95.197.1) x
- Advanced Options:**
  - Hold Time:** 90
  - Admin State:**

Buttons for 'Cancel' and 'Save' are located at the bottom right of the dialog box.

2. Enter the relevant information at the **Create BGP as a Service** window, including ASN, address family, and VMI identification.
3. Click **Save** to create the BGP object.

# Load Balancers

## IN THIS CHAPTER

- [Using Load Balancers in Contrail | 743](#)
- [Support for OpenStack LBaaS Version 2.0 APIs | 758](#)
- [Configuring Load Balancing as a Service in Contrail | 760](#)

## Using Load Balancers in Contrail

### IN THIS SECTION

- [Invoking LBaaS Drivers | 743](#)
- [Using a Service Appliance Set as the LBaaS Provider | 746](#)
- [Understanding the Load Balancer Agent | 747](#)
- [F5 Networks Load Balancer Integration in Contrail | 748](#)
- [Example: Creating a Load Balancer | 751](#)
- [Using the Avi Networks Load Balancer for Contrail | 752](#)

As of Contrail Release 3.0, load balancer LBaaS features are available. This topic includes:

### Invoking LBaaS Drivers

The provider field specified in the pool configuration determines which load balancer drivers are selected. The load balancer driver selected is responsible for configuring the external hardware or virtual machine load balancer.

Supported load balancer drivers include:

- HAProxy

- A10 Networks
- F5 Networks
- Avi Networks

Starting with Contrail 3.0, the Neutron LBaaS plugin creates required configuration objects (such as pool, VIP, members, and monitor) in the Contrail API server, instead of within the Neutron plugin context, as in previous releases.

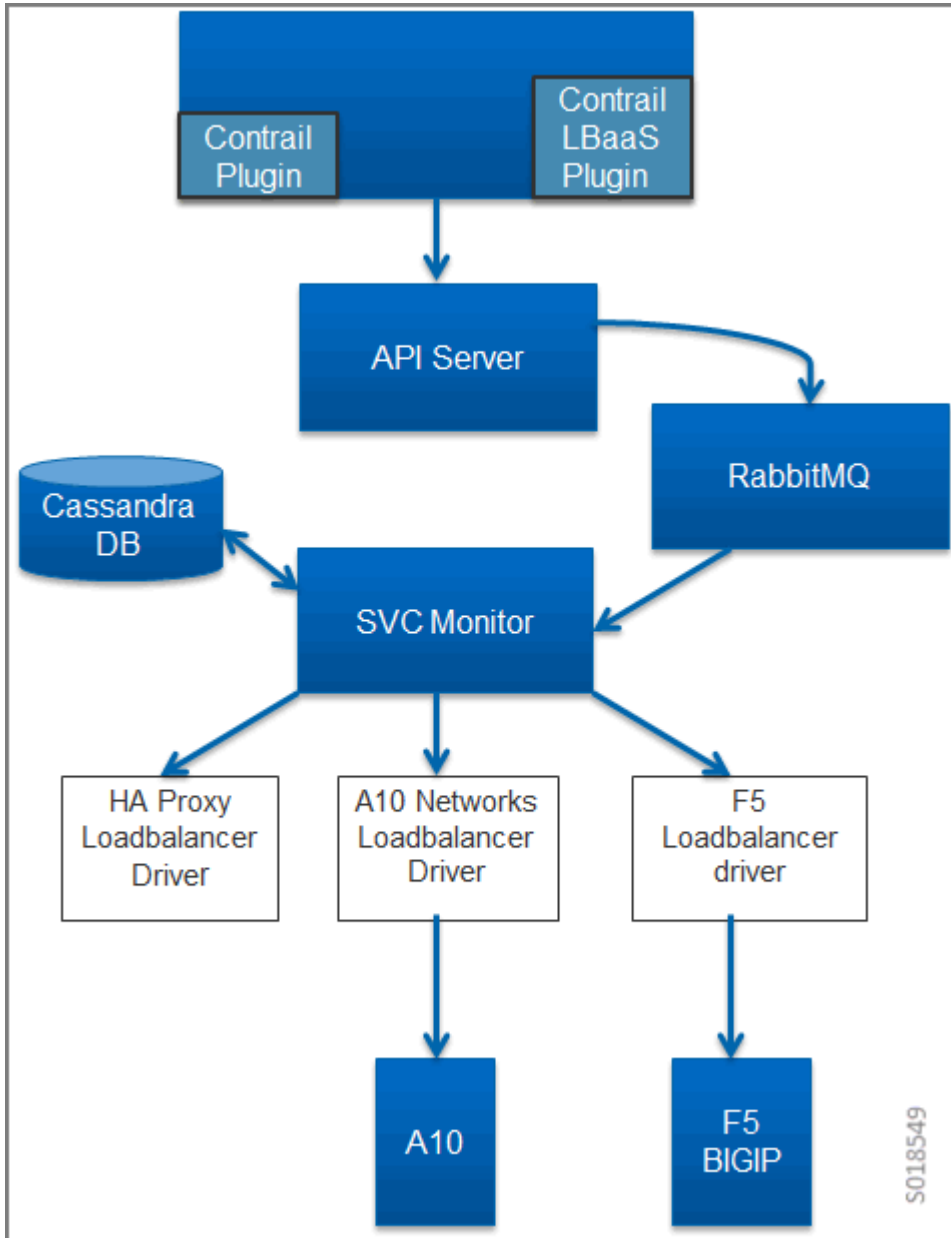
This method of configuration has the following benefits:

- Configuration objects can be created in multiple ways: from Neutron, from virtual controller APIs, or from the Contrail UI.
- The load balancer driver can make inline calls, such as REST or SUDS, to configure the external load balancer device.
- The load balancer driver can use Contrail service monitor infrastructure, such as database, logging, and API server.

[Figure 188 on page 745](#) provides an overview of the Contrail LBaaS components.



Figure 188: Contrail LBaaS Components



## Using a Service Appliance Set as the LBaaS Provider

In OpenStack Neutron, the load balancer provider is statically configured in `neutron.conf`, which requires restart of the Neutron server when configuring a new provider. The following is an example of the service provider configuration in `neutron.conf`.

```
[service_providers]
service_provider = LOADBALANCER:Opencontrail:neutron_plugin_contrail.plugins.opencontrail.
loadbalancer.driver.OpencontrailLoadbalancerDriver:default
```

In Contrail Release 3.0 and greater, the Neutron LBaaS provider is configured by using the object `service-appliance-set`. All of the configuration parameters of the LBaaS driver are populated to the `service-appliance-set` object and passed to the driver.

During initialization, the service monitor creates a default service appliance set with a default LBaaS provider, which uses an HAProxy-based load balancer. The service appliance set consists of individual service appliances for load balancing the traffic. The service appliances can be physical devices or virtual machines.

### Sample Configuration: Service Appliance Set

The following is a sample configuration of the service appliance set for the LBaaS provider:

```
{
  "service-appliance-set": {
    "fq_name": [
      "default-global-system-config",
      "f5"
    ],
    "service_appliance_driver":
      "svc_monitor.services.loadbalancer.drivers.f5.f5_driver.OpencontrailF5LoadbalancerDriver",
    "parent_type": "global-system-config",
    "service_appliance_set_properties": {
      "key_value_pair": [
        {
          "key": "sync_mode",
          "value": "replication"
        },
        {
          "key": "global_routed_mode",
          "value": "True"
        }
      ]
    }
  }
}
```

```

    ]
  },
  "name": "f5"
}
}

```

### Sample Configuration: Single Service Appliance

The following is a sample configuration of a single service appliance:

```

{
  "service-appliance": {
    "fq_name": [
      "default-global-system-config",
      "f5",
      "bigip"
    ],
    "parent_type": "service-appliance-set",
    "service_appliance_ip_address": "<ip address>",
    "service_appliance_user_credentials": {
      "username": "admin",
      "password": "<password>"
    },
    "name": "bigip"
  }
}

```

## Understanding the Load Balancer Agent

The load balancer agent is a module in the service monitor. The service monitor listens on the RabbitMQ configuration messaging queue (`vnc_config.object-update`) to get configuration objects. The dependency tracker triggers changes to all related objects, based on configuration updates.

The dependency tracker is informed to notify the pool object whenever the VIP, member, or health monitor object is modified.

Whenever there is an update to the pool object, either directly due to a pool update or due to a dependency update, the load balancer agent in the service monitor is notified.

The load balancer agent module handles the following:

- Loading and unloading LBaaS driver-based service appliance set configuration.

- Providing the abstract driver class for the load balancer driver.
- Invoking the LBaaS driver.
- Load balancer-related configuration.

## F5 Networks Load Balancer Integration in Contrail

This section details use of the F5 load balancer driver with Contrail.

Contrail Release 3.0 implements an LBaaS driver that supports a physical or virtual F5 Networks load balancer, using the abstract load balancer driver class, `ContrailLoadBalancerAbstractDriver`.

This driver is invoked from the load balancer agent of the `contrail-svc-monitor`. The driver makes a BIG-IP interface call to configure the F5 Networks device. All of the configuration parameters used to tune the driver are configured in the `service-appliance-set` object and passed to the driver by the load balancer agent while loading the driver.

The F5 load balancer driver uses the BIG-IP interface version V1.0.6, which is a Python package extracted from the load balancer plugin provided by F5 Networks. The driver uses either a SOAP API or a REST API.

### F5 Load Balancer Global Routed Mode

The F5 load balancer driver is programmed in `global routed` mode using a property of the `service-appliance-set`.

This section describes the features and requirements of the F5 load balancer driver configured in `global routed` mode.

The following are features of the `global routed` mode.

- All virtual IP addresses (VIPs) are assumed to be routable from clients and all members are routable from the F5 device.
- All access to and from the F5 device is assumed to be globally routed, with no segregation between tenant services on the F5 device. Consequently, do NOT configure overlapping addresses across tenants and networks.
- The F5 device can be attached to the corporate network or to the IP fabric.

The following are requirements to support `global routed` mode of an F5 device used with LBaaS:

- The entire configuration of the F5 device for Layer 2 and Layer 3 is preprovisioned.
- All tenant networks and all IP fabrics are in the same namespace as the corporate network.

- All VIPs are in the same namespace as the tenant and corporate networks.

## Traffic Flow in Global Routed Mode

This section describes and illustrates the behavior of traffic flow in global routed mode.

The information in this section is based on a model that includes the following network topology:

Corporate Network --- DC Gateway (MX device) --- IP Fabric --- Compute nodes

The Corporate Network, the IP Fabric and all tenant networks use IP addresses from a single namespace, there is no overlap of the addresses in the networks. The F5 devices can be attached to the Corporate Network or to the IP Fabric, and are configured to use the global routed mode.

The role of the MX Series device is to route post-proxy traffic, coming from the F5 device in the underlay, to the pool members in the overlay. In the reverse direction, the MX device takes traffic coming from the pool members in the overlay and routes it back to the F5 device in the underlay.

The MX device is preprovisioned with the following:

- VRF connected to pool network 2
- ability to route traffic from inet.0 to the pool network

The MX routes the traffic from inet.0 to public VRF and sends traffic to the compute node where the pool member is instantiated.

The F5 device is preprovisioned with the following:

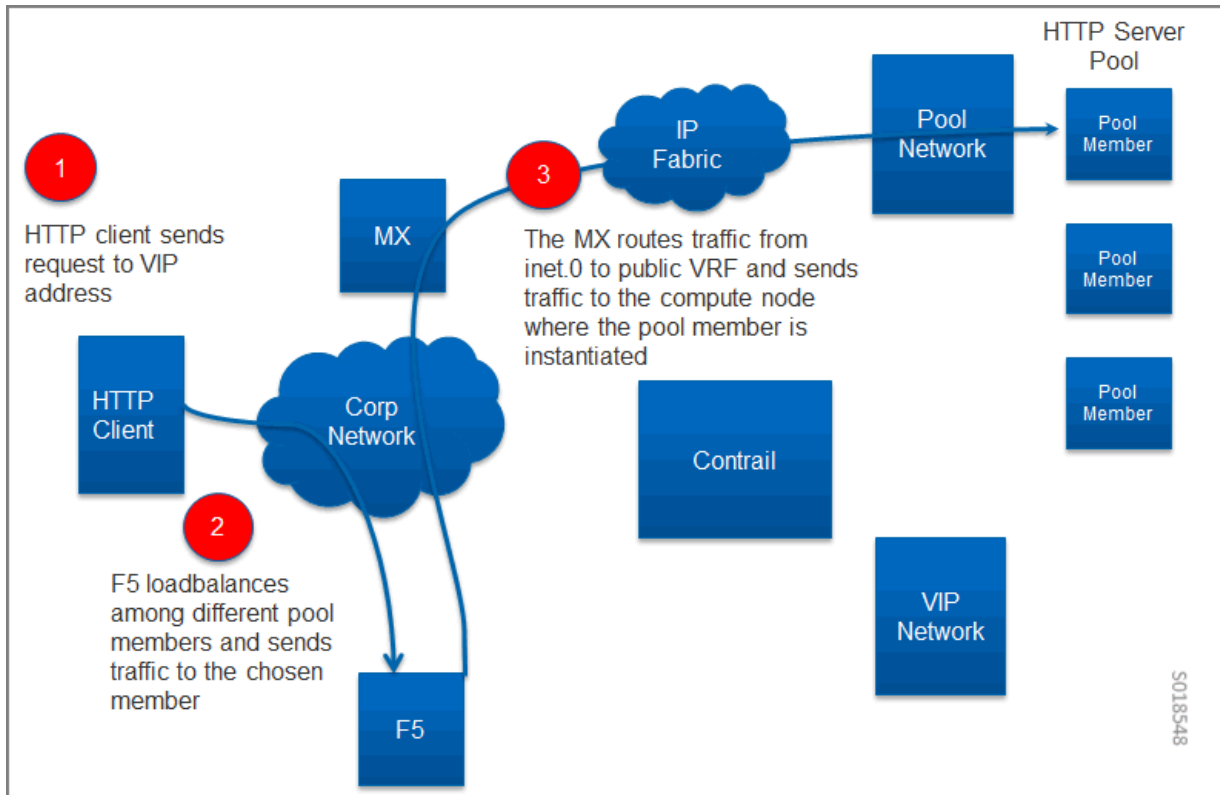
- publish route to attract VIP traffic
- pool network subnet route that points to the MX device

The F5 device is responsible for attracting traffic destined to all the VIPs, by advertising a subnet route that covers all VIPs using IGP.

The F5 device load balances among different pool members and sends traffic to the chosen member.

[Figure 189 on page 750](#) shows the global routed traffic flow.

Figure 189: Global Routed Traffic Flow



A similar result can also be achieved on the switch to which the F5 is attached, by publishing the VIP subnet in IGP and using a static route to point the VIP traffic to the F5 device.

The MX should attract the reverse traffic from the pool members going back to the F5.

### Routing Traffic to Pool Members

For post load balancing traffic going from the F5 device to the pool members, the MX Series device needs to attract traffic for all the tenant networks.

### Routing Reverse Traffic from Pool Members to the F5 Device

The MX should attract the reverse traffic from the pool members going back to the F5.

### Initial Configuration on an F5 Device

- The operator is responsible for ensuring that the F5 device attracts traffic to all VIP subnets by injecting the route for the VIP subnet into IGP. Alternately, the switch to which F5 is connected can advertise the VIP subnet route and use the static route to send VIP traffic to the F5 device.

- In the global routed mode, the F5 uses AutoMap SNAT for all VIP traffic.

### Initial Configuration on an MX Series Device Used as DC Gateway

- The operator must identify a super-net that contains all tenant network subnets (pool members across multiple pools) and advertise its route into corporate and fabric networks, using IGP (preferred) or static routes.
- The operator must add a static route for the super-net into inet.0 with a next-hop of public.inet.0.
- The operator must create a public VRF and get its default route imported into the VRF. This is to attract the return traffic from pool members to the F5 device (VIP destination).

### Configuration on MX Device for Each Pool Member

- For each member virtual network, the operator adds a policy to connect the member pool virtual network to the public virtual network.
- As new member virtual networks are connected to the public virtual network by policy, corresponding targets are imported by the public VRF on MX. The Contrail Device Manager generates the configuration of import, export targets for public VRF on the MX device.
- The operator must ensure that security group rules for the member virtual network ports allow traffic coming from the F5 device.

### Example: Creating a Load Balancer

Use the following steps to create a load balancer in Contrail Release 3.0 and greater.

1. To configure a service appliance set, use the script in `/opt/contrail/utils` to create a load balancer provider. With the script, you specify the driver and name of the selected provider. Additional configuration can be performed using the key-value pair property configuration.

```
/opt/contrail/utils/service_appliance_set.py --api_server_ip <ip address>--api_server_port 8082 --oper add --
admin_user admin --admin_password <password> --admin_tenant_name admin --name f5 --driver
"svc_monitor.services.loadbalancer.drivers.f5.f5_driver.OpencontrailF5LoadbalancerDriver" --properties
'{"use_snat": "True", "num_snat": "1", "global_routed_mode":"True", "sync_mode": "replication", "vip_vlan":
"trial2"}'
```

2. Add the actual device information of the load balancer.

```
/opt/contrail/utils/service_appliance.py --api_server_ip <ip address>--api_server_port 8082 --oper add --
admin_user admin --admin_password <password> --admin_tenant_name admin --name bigip --service_appliance_set f5
--device_ip 10.204.216.113 --user_credential '{"user": "admin", "password": "<password>"}'
```

3. Refer to the load balancer provider while configuring the pool.

```
neutron lb-pool-create --lb-method ROUND_ROBIN --name web_service --protocol HTTP --provider "f5" --subnet-id
<subnet id>
```

4. Add members to the load balancer pool. Both bare metal webserver and overlay webserver are allowed as pool members. The F5 device can load balance the traffic among all pool members.

```
neutron lb-member-create --address <ip address>--protocol-port 8080 --weight 3 web_service
```

```
neutron lb-member-create --address <ip address> --protocol-port 8080 --weight 2 web_service
```

5. Create a VIP for the load balancer pool.

```
neutron lb-vip-create --name httpserver --protocol-port 80 --protocol HTTP web_service --subnet-id <subnet id>
```

6. Create the health monitor and associate it with the load balancer pool.

```
neutron lb-healthmonitor-create --delay 3 --type HTTP --max-retries 3 --timeout 3
```

```
neutron lb-healthmonitor-associate <nnnnn-nnnnn-nnnn-> web_service
```

## Using the Avi Networks Load Balancer for Contrail

If you are using the Avi LBaaS driver in an OpenStack Contrail environment, there are two possible modes that are mutually-exclusive. The Avi Vantage cloud configuration is exactly the same in both modes:

- Neutron-based Avi LBaaS driver  
In this mode, the Avi LBaaS driver derives from Neutron and resides in the Neutron server process. This mode enables coexistence of multiple Neutron LBaaS providers.
- Contrail-based Avi LBaaS driver  
In this mode, the Avi LBaaS driver derives from Contrail and resides in the service-monitor process. This mode enables coexistence of multiple Contrail LBaaS providers.

**NOTE:** In a Contrail environment, you cannot have a mix of Contrail LBaaS and Neutron LBaaS. You must select a mode that is compatible with the current environment.

## Installing the Avi LBaaS Neutron Driver

Use the following procedure to install the Avi Networks LBaaS load balancer driver for the Neutron server for Contrail.

The following steps are performed on the Neutron server host.



1. Determine the installed version of the Contrail Neutron plugin.

```
$ contrail-version neutron-plugin-contrail
Package Version
-----
neutron-plugin-contrail 3.0.2.0-51
```

2. Adjust the `neutron.conf` database connection URL.

```
$ vi /etc/neutron/neutron.conf
# if using mysql
connection = mysql+pymysql://neutron:c0ntrail123@127.0.0.1/neutron
```

3. Populate and upgrade the Neutron database schema.

```
# to upgrade to head
$ neutron-db-manage upgrade head
# to upgrade to a specific version
$ neutron-db-manage --config-file /etc/neutron/neutron.conf upgrade liberty
```

4. Drop foreign key constraints.

```
# obtain current mysql token
$ cat /etc/contrail/mysql.token
fabe17d9dd5ae798f7ea

$ mysql -u root -p
Enter password: fabe17d9dd5ae798f7ea

mysql> use neutron;

mysql> show create table vips;
# CONSTRAINT `vips_ibfk_1` FOREIGN KEY (`port_id`) REFERENCES `ports` (`id`) - ports table is
not used by Contrail
mysql> alter table vips drop FOREIGN KEY vips_ibfk_1;

mysql> show create table lbaas_loadbalancers;
# CONSTRAINT `fk_lbaas_loadbalancers_ports_id` FOREIGN KEY (`vip_port_id`) REFERENCES `ports`
```

```
(`id`)  
mysql> alter table lbaas_loadbalancers drop FOREIGN KEY fk_lbaas_loadbalancers_ports_id;
```

5. To install the Avi LBaaS plugin, continue with steps from the readme file that downloads with the Avi LBaaS software. You can perform either a local installation or a manual installation. The following are sample installation steps.

- For a local installation:

```
# LBaaS v1 driver  
$ ./install.sh --aname avi_adc --aip  
  
<controller_ip|controller_vip>  
--auser  
  
--apass  
  
# LBaaS v2 driver  
$ ./install.sh --aname avi_adc_v2 --aip  
<controller_ip|controller_vip>  
--auser  
  
--apass  
  
--v2
```

- For a manual installation:

```
# LBaaS v1 driver  
$ vi /etc/neutron/neutron.conf  
#service_plugins =  
neutron_plugin_contrail.plugins.opencontrail.loadbalancer.plugin.LoadBalancerPlugin  
service_plugins = neutron_lbaas.services.loadbalancer.plugin.LoadBalancerPlugin  
[service_providers]  
service_provider =  
LOADBALANCER:Avi_ADC:neutron_lbaas.services.loadbalancer.drivers.avi.avi_driver.AviLbaaSDri  
ver  
  
[avi_adc]  
address=10.1.11.4  
user=admin
```

```

password=avi123
cloud=jcos

# LBaaS v2 driver
$ vi /etc/neutron/neutron.conf
#service_plugins =
neutron_plugin_contrail.plugins.opencontrail.loadbalancer.plugin.LoadBalancerPlugin
service_plugins = neutron_lbaas.services.loadbalancer.plugin.LoadBalancerPluginv2
[service_providers]
service_provider = LOADBALANCERV2:avi_adc_v2:neutron_lbaas.drivers.avi.driver.AviDriver

[avi_adc_v2]
controller_ip=10.1.11.3
username=admin
password=avi123

$ service neutron-server restart
$ neutron service-provider-list

```

## Installing the Avi LBaaS Contrail Driver

Use the following procedure to install the Avi Networks LBaaS load balancer driver for Contrail.

The following steps are performed on the Contrail api-server host.

1. Determine the installed version of the Contrail Neutron plugin.

```

$ contrail-version neutron-plugin-contrail
Package Version
-----
neutron-plugin-contrail 3.0.2.0-51

```

2. Install the Avi driver.

```

# LBaaS v2 driver
$ ./install.sh --aname oavi_adc_v2 --aip

<controller_ip|controller_vip>
  --auser

  --apass

```

```
--v2 --no-restart --no-confmodify
```

### 3. Set up the service appliance set.

**NOTE:** If `neutron_lbaas` doesn't exist on the `api-server` node, adjust the driver path to the correct path location for `neutron_lbaas`.

```
$ /opt/contrail/utils/service_appliance_set.py --api_server_ip 10.xx.xx.100 --api_server_port 8082 --oper add
--admin_user admin --admin_password <password> --admin_tenant_name admin --name ocavi_adc_v2 --driver
"neutron_lbaas.drivers.avi.avi_ocdriver.OpencontrailAviLoadbalancerDriver" --properties '{"address":
"10.1.xx.3", "user": "admin", "password": "avi123", "cloud": "Default-Cloud"}'
```

### 4. To delete the service appliance set.

```
$ /opt/contrail/utils/service_appliance_set.py --api_server_ip 10.xx.xx.100 --api_server_port 8082 --oper del
--admin_user admin --admin_password <password> --admin_tenant_name admin --name ocavi_adc_v2
```

## Configuring the Avi Controller

1. If OpenStack endpoints are private IPs and Contrail provides a public front-end IP to those endpoints, use iptables to DNAT. On the AviController only, perform iptable NAT to reach the private IPs.

```
$ iptables -t nat -I OUTPUT --dest 17x.xx.xx.50 -j DNAT --to-dest 10.xx.xx.100
```

2. To configure the Avi controller during cloud configuration, select the “Integration with Contrail” checkbox and provide the endpoint URL of the Contrail VNC api-server. Use the Keystone credentials from the OpenStack configuration to authenticate with the api-server service.

### Example Configuration Settings

```
: > show cloud jcos
+-----+-----+
| Field          | Value                                     |
+-----+-----+
| uuid           | cloud-104bb7e6-a9d2-4b34-a4c5-d94be659bb91 |
| name           | jcos                                     |
| vtype          | CLOUD_OPENSTACK                         |
| openstack_configuration |                                           |
|  username      | admin                                    |
|  admin_tenant  | demo                                     |
|  keystone_host | 17x.xx.xx.50                             |
```

```

|  mgmt_network_name      | mgmtnw      |
|  privilege              | WRITE_ACCESS |
|  use_keystone_auth      | True        |
|  region                 | RegionOne   |
|  hypervisor             | KVM         |
|  tenant_se              | True        |
|  import_keystone_tenants | True        |
|  anti_affinity          | True        |
|  port_security          | False       |
|  security_groups        | True        |
|  allowed_address_pairs  | True        |
|  free_floatingips       | True        |
|  img_format             | OS_IMG_FMT_AUTO |
|  use_admin_url          | True        |
|  use_internal_endpoints | False       |
|  config_drive           | True        |
|  insecure               | True        |
|  intf_sec_ips           | False       |
|  external_networks      | False       |
|  neutron_rbac           | True        |
|  nuage_port              | 8443        |
|  contrail_endpoint      | http://10.10.10.100:8082 |
|  apic_mode              | False       |
|  dhcp_enabled           | True        |
|  mtu                    | 1500 bytes  |
|  prefer_static_routes   | False       |
|  enable_vip_static_routes | False       |
|  license_type           | LIC_CORES   |
|  tenant_ref             | admin       |
+-----+-----+

```

## RELATED DOCUMENTATION

[Configuring Load Balancing as a Service in Contrail | 760](#)

[Support for OpenStack LBaaS Version 2.0 APIs | 758](#)

## Support for OpenStack LBaaS Version 2.0 APIs

### IN THIS SECTION

- [Platform Support | 758](#)
- [Using OpenStack LBaaS Version 2.0 | 758](#)
- [Support for Multiple Certificates per Listener | 759](#)
- [Neutron Load-Balancer Creation | 759](#)

Starting with Release 3.1, Contrail provides support for the OpenStack Load Balancer as a Service (LBaaS) Version 2.0 APIs in the Liberty release of OpenStack.

### Platform Support

[Table 56 on page 758](#) shows which Contrail with OpenStack release combinations support which version of OpenStack LBaaS APIs.

**Table 56: Contrail OpenStack Platform Support for LBaaS Versions**

| Contrail OpenStack Platform                            | LBaaS Support                          |
|--|--|
| Contrail-3.1-Liberty (and subsequent OS releases)      | Only LBaaS v2 is supported.            |
| Contrail-3.0-Liberty (and subsequent OS releases)      | LBaaS v1 is default. LBaaS v2 is Beta. |
| <Contrail-any-release>-Kilo (and previous OS releases) | Only LBaaS v1 is supported.            |

### Using OpenStack LBaaS Version 2.0

The OpenStack LBaaS Version 2.0 extension enables tenants to manage load balancers for VMs, for example, load-balancing client traffic from a network to application services, such as VMs, on the same network. The LBaaS Version 2.0 extension is used to create and manage load balancers, listeners, pools, members of a pool, and health monitors, and to view the status of a resource.

For LBaaS v2.0, the Contrail controller aggregates the configuration by provider. For example, if haproxy is the provider, the controller generates the configuration for haproxy and eliminates the need to send all of the load-balancer resources to the vrouter-agent; only the generated configuration is sent, as part of the service instance.

For more information about OpenStack v2.0 APIs, refer to the section *LBaaS 2.0 (STABLE) (lbaas, loadbalancers, listeners, health\_monitors, pools, members)*, at <http://developer.openstack.org/api-ref-networking-v2-ext.html>.

LBaaS v2.0 also allows users to listen to multiple ports for the same virtual IP, by decoupling the virtual IP address from the port.

The object model has the following resources:

- Load balancer—Holds the virtual IP address
- Listeners—One or many listeners with different ports, protocols, and so on
- Pools
- Members
- Health monitors

## Support for Multiple Certificates per Listener

Multiple certificates per listener are supported, with OpenStack Barbican as the storage for certificates. OpenStack Barbican is a REST API designed for the secure storage, provisioning, and management of secrets such as passwords, encryption keys, and X.509 certificates.

The following is an example CLI to store certificates in Barbican:

```
- barbican --os-identity-api-version 2.0 secret store --payload-content-type='text/plain' --name='certificate' --payload="$(cat server.crt)"
```

For more information about OpenStack Barbican, see: <https://wiki.openstack.org/wiki/Barbican>.

## Neutron Load-Balancer Creation

The following is an example of Neutron load-balancer creation:

```
- neutron net-create private-net
- neutron subnet-create --name private-subnet private-net 10.30.30.0/24
- neutron lbaas-loadbalancer-create $(neutron subnet-list | awk '/ private-subnet / {print $2}')
```

```
--name lb1

- neutron lbaas-listener-create --loadbalancer lb1 --protocol-port 443 --protocol
TERMINATED_HTTPS --name listener1 --default-tls-container=$(barbican --os-identity-api-version
2.0 container list | awk '/ tls_container / {print $2}')

- neutron lbaas-pool-create --name pool1 --protocol HTTP --listener listener1 --lb-algorithm
ROUND_ROBIN

- neutron lbaas-member-create --subnet private-subnet --address 30.30.30.10 --protocol-port 80
mypool

- neutron lbaas-member-create --subnet private-subnet --address 30.30.30.11 --protocol-port 80
mypool
```

## RELATED DOCUMENTATION

<https://wiki.openstack.org/wiki/Barbican>

<http://developer.openstack.org/api-ref-networking-v2-ext.html>

[Using Load Balancers in Contrail | 743](#)

[Configuring Load Balancing as a Service in Contrail | 760](#)

## Configuring Load Balancing as a Service in Contrail

### IN THIS SECTION

- [Overview: Load Balancing as a Service | 761](#)
- [Contrail LBaaS Implementation | 762](#)
- [Configuring LBaaS Using CLI | 763](#)



## Overview: Load Balancing as a Service

Load Balancing as a Service (LBaaS) is a feature available through OpenStack Neutron. Contrail Release 1.20 and greater allows the use of the Neutron API for LBaaS to apply open source load balancing technologies to provision a load balancer in the Contrail system.

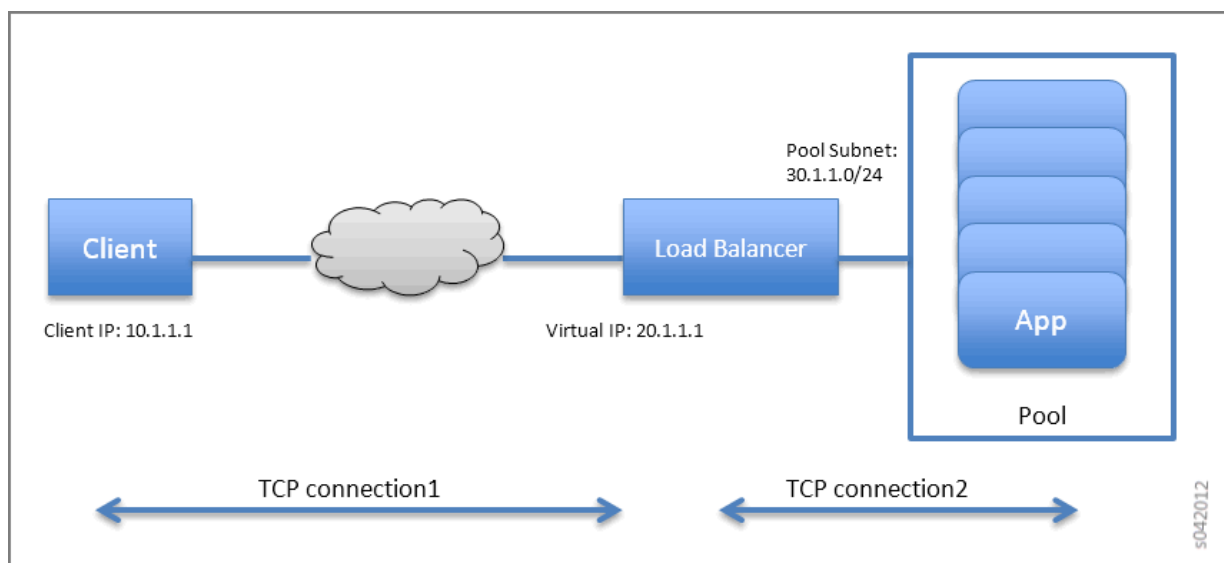
The LBaaS load balancer enables the creation of a pool of virtual machines serving applications, all front-ended by a virtual-ip. The LBaaS implementation has the following features:

- Load balancing of traffic from clients to a pool of backend servers. The load balancer proxies all connections to its virtual IP.
- Provides load balancing for HTTP, TCP, and HTTPS.
- Provides health monitoring capabilities for applications, including HTTP, TCP, and ping.
- Enables floating IP association to virtual-ip for public access to the backend pool.

In the following figure, the load balancer is launched with the virtual IP address 20.1.1.1. The backend pool of virtual machine applications (App Pool) is on the subnet 30.1.1.0/24. Each of the application virtual machines gets an IP address (virtual-ip) from the pool subnet. When a client connects to the virtual-ip for accessing the application, the load balancer proxies the TCP connection on its virtual-ip, then creates a new TCP connection to one of the virtual machines in the pool.

The pool member is selected using one of following methods:

- weighted round robin (WRR), based on the weight assignment
- least connection, selects the member with the fewest connections
- source IP selects based on the source-ip of the packet



Additionally, the load balancer monitors the health of each pool member using the following methods:

- Monitors TCP by creating a TCP connection at intervals.
- Monitors HTTP by creating a TCP connection and issuing an HTTP request at intervals.
- Monitors ping by checking if a member can be reached by pinging.

## Contrail LBaaS Implementation

Contrail supports the OpenStack LBaaS Neutron APIs and creates relevant objects for LBaaS, including `virtual-ip`, `loadbalancer-pool`, `loadbalancer-member`, and `loadbalancer-healthmonitor`. Contrail creates a service instance when a `loadbalancer-pool` is associated with a `virtual-ip` object. The service scheduler then launches a namespace on a randomly selected virtual router and spawns HAProxy into that namespace. The configuration for HAProxy is picked up from the load balancer objects. Contrail supports high availability of namespaces and HAProxy by spawning active and standby on two different routers.

### A Note on Installation

To use the LBaaS feature, HAProxy, version 1.5 or greater and `iproute2`, version 3.10.0 or greater must both be installed on the Contrail compute nodes.

If you are using `fab` commands for installation, the `haproxy` and `iproute2` packages will be installed automatically with LBaaS if you set the following:

```
env.enable_lbaas=True
```

Use the following to check the version of the `iproute2` package on your system:

```
root@nodeh5:/var/log# ip -V
ip utility, iproute2-ss130716
root@nodeh5:/var/log#
```

### Limitations

LBaaS currently has these limitations:

- A pool should not be deleted before deleting the VIP.
- Multiple VIPs cannot be associated with the same pool. If pool needs to be reused, create another pool with the same members and bind it to the second VIP.
- Members cannot be moved from one pool to another. If needed, first delete the members from one pool, then add to a different pool.
- In case of active-standby failover, namespaces might not get cleaned up when the agent restarts.

- The floating-ip association needs to select the VIP port and not the service ports.

## Configuring LBaaS Using CLI

The LBaaS feature is enabled on Contrail through Neutron API calls. The following procedure shows how to create a pool network and a VIP network using CLI. The VIP network is created in the public network and members are added in the pool network.

### Creating a Load Balancer

Use the following steps to create a load balancer in Contrail.

1. Create a VIP network.

```
neutron net-create vipnet

neutron subnet-create --name vipsubnet vipnet 20.1.1.0/24
```

2. Create a pool network.

```
neutron net-create poolnet

neutron subnet-create --name poolsubnet poolnet 10.1.1.0/24
```

3. Create a pool for HTTP.

```
neutron lb-pool-create --lb-method ROUND_ROBIN --name mypool --protocol HTTP --subnet-id poolsubnet
```

4. Add members to the pool.

```
neutron lb-member-create --address 10.1.1.2 --protocol-port 80 mypool

neutron lb-member-create --address 10.1.1.3 --protocol-port 80 mypool
```

5. Create a VIP for HTTP and associate it to the pool.

```
neutron lb-vip-create --name myvip --protocol-port 80 --protocol HTTP--subnet-id vipsubnet mypool
```

### Deleting a Load Balancer

Use the following steps to delete a load balancer in Contrail.

1. Delete the VIP.

```
neutron lb-vip-delete <vip-uuid>
```

2. Delete members from the pool.

```
neutron lb-member-delete <member-uuid>
```

3. Delete the pool.

```
neutron lb-pool-delete <pool-uuid>
```

### Managing Healthmonitor for Load Balancer

Use the following commands to create a healthmonitor, associate a healthmonitor to a pool, disassociate a healthmonitor, and delete a healthmonitor.

1. Create a healthmonitor.

```
neutron lb-healthmonitor-create --delay 20 --timeout 10 --max-retries 3 --type HTTP
```

2. Associate a healthmonitor to a pool.

```
neutron lb-healthmonitor-associate <healthmonitor-uuid> mypool
```

3. Disassociate a healthmonitor from a pool.

```
neutron lb-healthmonitor-disassociate <healthmonitor-uuid> mypool
```

### Configuring an SSL VIP with an HTTP Backend Pool

Use the following steps to configure an SSL VIP with an HTTP backend pool.

1. Copy an SSL certificate to all compute nodes.

```
scp ssl_certificate.pem <compute-node-ip> <certificate-path>
```

2. Update the information in `/etc/contrail/contrail-vrouter-agent.conf`.

```
# SSL certificate path haproxy
haproxy_ssl_cert_path=<certificate-path>
```

3. Restart `contrail-vrouter-agent`.

```
service contrail-vrouter-agent restart
```

4. Create a VIP for port 443 (SSL).

```
neutron lb-vip-create --name myvip --protocol-port 443 --protocol HTTP --subnet-id vipsubnet mypool
```

## RELATED DOCUMENTATION

[Using Load Balancers in Contrail | 743](#)

[Support for OpenStack LBaaS Version 2.0 APIs | 758](#)

# Optimizing Contrail

## IN THIS CHAPTER

- [Route Target Filtering | 766](#)
- [Source Network Address Translation \(SNAT\) | 769](#)
- [Multiqueue Virtio Interfaces in Virtual Machines | 772](#)
- [vRouter Command Line Utilities | 774](#)

## Route Target Filtering

### IN THIS SECTION

- [Introduction | 766](#)
- [Debugging and Troubleshooting Route Target Filtering | 767](#)
- [RTF Limitations in Contrail 1.10 | 768](#)

### Introduction

BGP route target filtering (RTF) is a method for limiting the distribution of VPN routes to only those systems in the network for which the routes are necessary. If RTF is not active, the Contrail control node advertises all VPN routes to all of its VPN peers, which are either other control nodes or gateway routers such as an MX Series router. On the receiving side, the control node stores all VPN routes it receives from peers in the VPN table (for example, `bgp.l3vpn.0`). Any routes that do not include a route target extended community that is referenced by the local `vrf-import` policies are discarded by Junos.

The control node must send all route updates to its peers, even for unnecessary routes that are discarded. Continuous route updates are both CPU- and memory-intensive. The only routes that are necessary to advertise to gateway routers are those that belong to the virtual networks that are

configured for public access. It is not necessary to advertise VM routes belonging to other virtual networks to gateway routers.

If a datacenter has more than two control nodes, the `vrouter-agent` only subscribes to two of the control nodes, indicated by the discovery service. When a VM is initially launched in a virtual network, it sends an XMPP subscribe request for the virtual network VRF and publishes the VM route to the connected control node. It is not necessary to advertise routes belonging to this type of VRF to control nodes that don't have the `vrouter-agent` subscribed in that VRF.

RTF is used to optimize the route distribution among control nodes and to the gateway routers to avoid unwanted route updates. If the BGP peer has not advertised or configured with RTF address family, then all routes belonging to the VPN table will be advertised.

RTF implementation in the control node does not support advertising and receiving of default route targets.

Constrained route distribution using route target reachability information is defined in RFC 4684, *"Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)"*.

## Debugging and Troubleshooting Route Target Filtering

Use the tips in this section to troubleshoot issues with RTF. Use various `http introspect` commands to reveal details about BGP neighbors for RTF. The following is a sample portion of an `http introspect` page.

When you access an introspect page, only the first panel of detail columns appears. Use a scroll bar or arrow keys to reveal more columns to the right, and vice versa.

The screenshot shows a web interface for `BgpNeighborListResp`. It contains a table with the following data:

| peer    | peer_address  | deleted | peer_asn | local_address | local_asn | encoding |
|---------|---------------|---------|----------|---------------|-----------|----------|
| nodec13 | 10.204.216.70 | false   | 0        | 10.204.216.70 | 0         | XMPP     |

At the bottom left of the table area, there is a "more" link. On the right side of the table, there is a vertical scrollbar and a small number "5041999".

- Use the following `http introspect` URL to display the details of each peer:

`http://(your_node_name):8083/Snh_BgpNeighborReq`

For BGP peers, verify the configured and negotiated capability and the BGP table registration.

For XMPP peers, look at the `routing_instances` column to get details about the VRF to which the displayed `vrouter-agent` has subscribed and to see the import `rtargets` of the VRFs.

- Use the following http introspect URL to dump the `bgp.rtarget.0` table to display the `RTargetRoutes`:

```
http://(your_node_name):8083/Snh_ShowRouteReq?x=bgp.rtarget.0
```

- Use the following http introspect URL to dump the details for each of the route targets configured on the control node:

```
http://(your_node_name):8083/Snh_ShowRtGroupReq?
```

For any given route target, this introspect displays the BGP table that imports and exports the route, the BGP peers that have shown interest in this route, and all dependent routes (when this route target has the extended community BGP attribute).

## RTF Limitations in Contrail 1.10

The following are RTF limitations in Contrail 1.10.

- The control node does not support advertising a default route target, which is an `rtarget` route with `target:0:0` or `0/0` as the prefix. This type of `rtarget` route enables a BGP peer to receive all VPN routes without `rtarget` filtering.
- The control node does not support receiving a default route target. If `rtarget` routes with a default `rtarget` prefix are received, they are silently ignored.
- A `keep all` configuration, typical for BGP peering for a control node on an MX Series router, does not have impact, because all VPN routes with an extended community route target, for which the MX has advertised the `rtarget` route, are sent to the MX. An example of this type of typical configuration is the following:

```
set protocols bgp group contrail-control-nodes type internal
set protocols bgp group contrail-control-nodes local-address 10.204.216.253
set protocols bgp group contrail-control-nodes keep all
set protocols bgp group contrail-control-nodes family inet-vpn unicast
set protocols bgp group contrail-control-nodes family route-target
set protocols bgp group contrail-control-nodes neighbor 10.204.216.16
```



## Source Network Address Translation (SNAT)

### IN THIS SECTION

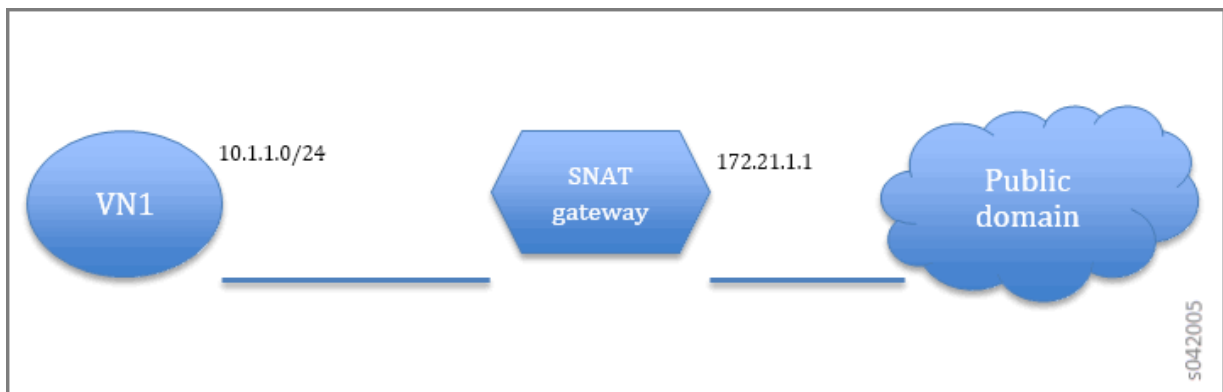
- [Overview | 769](#)
- [Neutron APIs for Routers | 770](#)
- [Network Namespace | 770](#)
- [Using the Web UI to Configure Routers with SNAT | 771](#)

### Overview

Source Network Address Translation (source-nat or SNAT) allows traffic from a private network to go out to the internet. Virtual machines launched on a private network can get to the internet by going through a gateway capable of performing SNAT. The gateway has one arm on the public network and as part of SNAT, it replaces the source IP of the originating packet with its own public side IP. As part of SNAT, the source port is also updated so that multiple VMs can reach the public network through a single gateway public IP.

The following diagram shows a virtual network with the private subnet of 10.1.1.0/24. The default route for the virtual network points to the SNAT gateway. The gateway replaces the source-ip from 10.1.1.0/24 and uses its public address 172.21.1.1 for outgoing packets. To maintain unique NAT sessions the source port of the traffic also needs to be replaced.

**Figure 190: Virtual Network With a Private Subnet**



## Neutron APIs for Routers

OpenStack supports SNAT gateway implementation through its Neutron APIs for routers. The SNAT flag can be enabled or disabled on the external gateway of the router. The default is True (enabled).

The OpenContrail plugin supports the Neutron APIs for routers and creates the relevant service-template and service-instance objects in the API server. The service scheduler in OpenContrail instantiates the gateway on a randomly-selected virtual router. OpenContrail uses network namespace to support this feature.

### Example Configuration: SNAT for Contrail

The SNAT feature is enabled on OpenContrail through Neutron API calls.

The following configuration example shows how to create a test network and a public network, allowing the test network to reach the public domain through the SNAT gateway.

1. Create the public network and set the router external flag.

```
neutron net-create public

neutron subnet-create public 172.21.1.0/24

neutron net-update public -- --router:external=True
```

2. Create the test network.

```
neutron net-create test

neutron subnet-create --name test-subnet test 10.1.1.0/24
```

3. Create the router with one interface in test.

```
neutron router-create r1

neutron router-interface-add r1 test-subnet
```

4. Set the external gateway for the router.

```
neutron router-gateway-set r1 public
```

## Network Namespace

Setting the external gateway is the trigger for OpenContrail to set up the Linux network namespace for SNAT.

The network namespace can be cleared by issuing the following Neutron command:

```
neutron router-gateway-clear r1
```

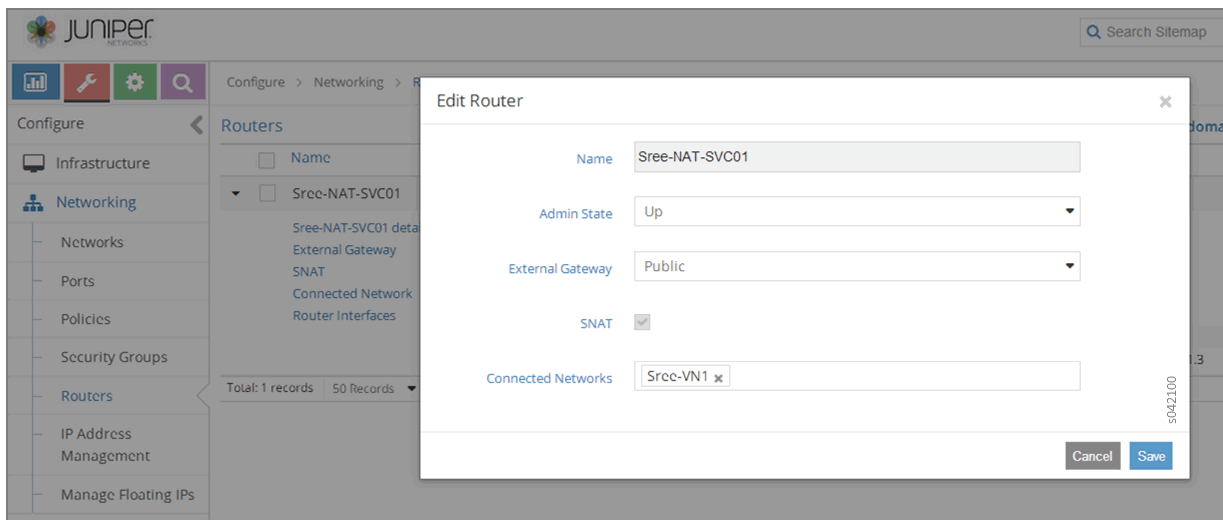
## Using the Web UI to Configure Routers with SNAT

You can use the Contrail user interface to configure routers for SNAT and to check the SNAT status of routers.

To enable SNAT for a router, go to **Configure > Networking > Routers**. In the list of routers, select the router for which SNAT should be enabled. Click the Edit cog to reveal the **Edit Routers** window. Click the check box for SNAT to enable SNAT on the router.

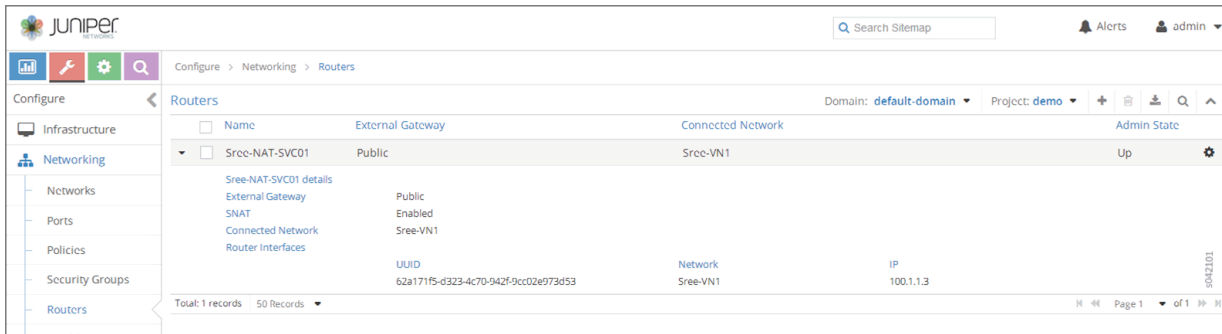
The following shows a router for which SNAT has been **Enabled**.

**Figure 191: Edit Router Window to Enable SNAT**



When a router has been **Enabled** for SNAT, the configuration can be seen by selecting **Configure > Networking > Routers**. In the list of routers, click open the router of interest. In the list of features for that router, the status of SNAT is listed. The following shows a router that has been opened in the list. The status of the router shows that SNAT is **Enabled**.

Figure 192: Router Status for SNAT



| Name           | External Gateway | Connected Network | Admin State |
|----------------|------------------|-------------------|-------------|
| Sree-NAT-SVC01 | Public           | Sree-VN1          | Up          |

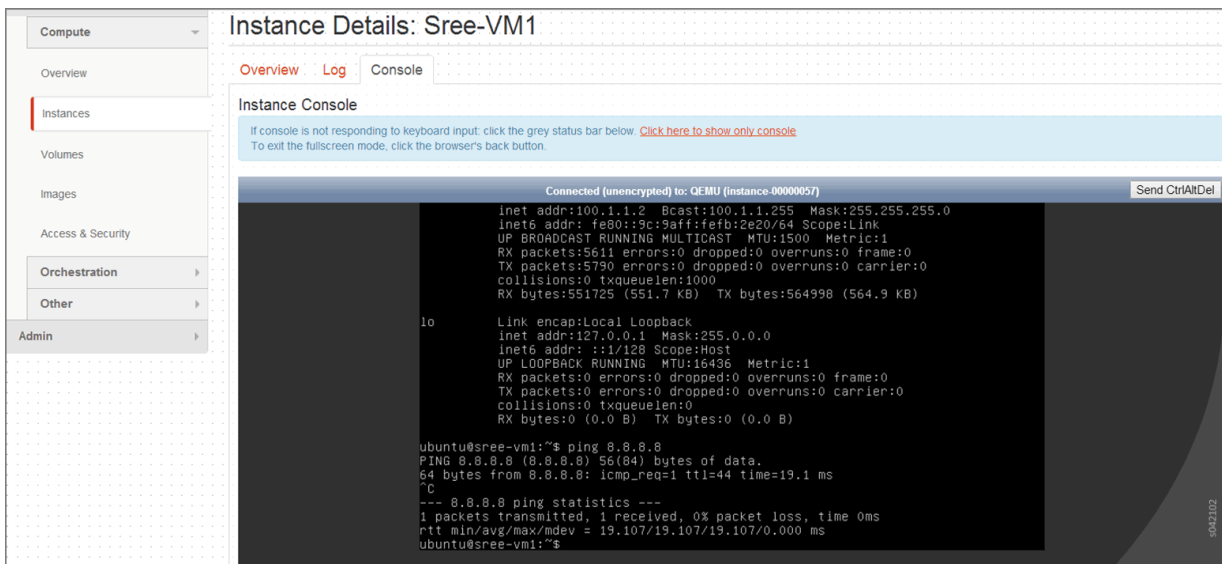
Sree-NAT-SVC01 details

|                   |          |
|-------------------|----------|
| External Gateway  | Public   |
| SNAT              | Enabled  |
| Connected Network | Sree-VN1 |
| Router Interfaces |          |

Total: 1 records 50 Records

You can view the real time status of a router with SNAT by viewing the instance console, as in the following.

Figure 193: Instance Details Window



Instance Details: Sree-VM1

Overview Log Console

Instance Console

If console is not responding to keyboard input: click the grey status bar below [Click here to show only console](#)  
To exit the fullscreen mode, click the browser's back button.

```

Connected (unencrypted) to: QEMU (instance-00000057)
inet addr:100.1.1.2 Bcast:100.1.1.255 Mask:255.255.255.0
inet6 addr: fe80::9c:9aff:fe20:764 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:5611 errors:0 dropped:0 overruns:0 frame:0
TX packets:5790 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:551725 (551.7 KB) TX bytes:564998 (564.9 KB)

lo
Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

ubuntu@sree-vm1:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data:
64 bytes from 8.8.8.8: icmp_req=1 ttl=44 time=19.1 ms
^C
--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 19.107/19.107/19.107/0.000 ms
ubuntu@sree-vm1:~$

```

## Multiqueue Virtio Interfaces in Virtual Machines

### IN THIS SECTION

● [Multiqueue Virtio Overview | 773](#)

Contrail 3.2 adds support for multiqueue for the DPDK-based vrouter.

Contrail 3.1 supports multiqueue virtio interfaces for Ubuntu kernel-based router, only.

## Multiqueue Virtio Overview

OpenStack Liberty supports the ability to create VMs with multiple queues on their virtio interfaces. Virtio is a Linux platform for I/O virtualization, providing a common set of I/O virtualization drivers. Multiqueue virtio is an approach that enables the processing of packet sending and receiving to be scaled to the number of available virtual CPUs (vCPUs) of a guest, through the use of multiple queues.

## Requirements and Setup for Multiqueue Virtio Interfaces

To use multiqueue virtio interfaces, ensure your system meets the following requirements:

- The OpenStack version must be Liberty or greater.
- The maximum number of queues in the VM interface is set to the same value as the number of vCPUs in the guest.
- The VM image metadata property is set to enable multiple queues inside the VM.

## Setting Virtual Machine Metadata for Multiple Queues

Use the following command on the OpenStack node to enable multiple queues on a VM:

```
source /etc/contrail/openstackrc
nova image-meta <image_name> set hw_vif_multiqueue_enabled="true"
```

After the VM is spawned, use the following command on the virtio interface in the guest to enable multiple queues inside the VM:

```
ethtool -L <interface_name> combined <#queues>
```

Packets will now be forwarded on all queues in the VM to and from the vRouter running on the host.

**NOTE:** Multiple queues in the VM are only supported with the kernel mode vRouter in Contrail 3.1.

Contrail 3.2 adds support for multiple queues with the DPDK-based vrouter, using OpenStack Mitaka. The DPDK vrouter has the same setup requirements as the kernel mode vrouter. However, in the `ethtool -L` setup command, the number of queues cannot be higher than the number of CPU cores assigned to vrouter in the testbed file.

## vRouter Command Line Utilities

### IN THIS SECTION

- [Overview | 774](#)
- [vif Command | 775](#)
- [flow Command | 779](#)
- [vrfstats Command | 781](#)
- [rt Command | 782](#)
- [dropstats Command | 783](#)
- [mpls Command | 787](#)
- [mirror Command | 789](#)
- [vxlan Command | 791](#)
- [nh Command | 793](#)

### Overview

This section describes the shell prompt utilities available for examining the state of the vrouter kernel module in Contrail.

The most useful commands for inspecting the Contrail vrouter module are summarized in the following table.

| Command   | Description  |
|-----------|--|
| vif       | Inspect vrouter interfaces associated with the vrouter module. |
| flow      | Display active flows in a system.                              |
| vrfstats  | Display next hop statistics for a particular VRF.              |
| rt        | Display routes in a VRF.                                       |
| dropstats | Inspect packet drop counters in the vrouter.                   |
| mpls      | Display the input label map programmed into the vrouter.       |
| mirror    | Display the mirror table entries.                              |
| vxlan     | Display the vxlan table entries.                               |
| nh        | Display the next hops that the vrouter knows.                  |
| --help    | Display all command options available for the current command. |

The following sections describe each of the vrouter utilities in detail.

## vif Command

The vrouter requires vrouter interfaces (`vif`) to forward traffic. Use the `vif` command to see the interfaces that are known by the vrouter.

**NOTE:** Having interfaces only in the OS (Linux) is not sufficient for forwarding. The relevant interfaces must be added to vrouter. Typically, the set up of interfaces is handled by components like `nova-compute` or `vrouter agent`.

**Example: vif --list**

```
# vif --list
vif0/0 OS: pkt0
    Type:Agent HWaddr:00:00:5e:00:01:00 IPaddr:0
    Vrf:65535 Flags:L3 MTU:1514 Ref:2
    RX packets:6591 bytes:648577 errors:0
    TX packets:12150 bytes:1974451 errors:0
vif0/1 OS: vhost0
    Type:Host HWaddr:00:25:90:c3:08:68 IPaddr:0
    Vrf:0 Flags:L3 MTU:1514 Ref:3
    RX packets:3446598 bytes:4478599344 errors:0
    TX packets:851770 bytes:1337017154 errors:0
vif0/2 OS: p1p0p0 (Speed 1000, Duplex 1)
    Type:Physical HWaddr:00:25:90:c3:08:68 IPaddr:0
    Vrf:0 Flags:L3 MTU:1514 Ref:22
    RX packets:1643238 bytes:1391655366 errors:2812
    TX packets:3523278 bytes:6806058059 errors:0
vif0/18 OS: tap3214fc7e-88
    Type:Virtual HWaddr:00:00:5e:00:01:00 IPaddr:0
    Vrf:13 Flags:PL3L2 MTU:9160 Ref:6
    RX packets:60 bytes:4873 errors:0
    TX packets:21 bytes:2158 errors:0
```

**Table 57: vif Fields**

| vif Output Field | Description   |
|------------------|---|
| vif0/X           | The router assigned name, where 0 is the router id and X is the index allocated to the interface within the router.   |
| OS: pkt0         | The pkt0 (in this case) is the name of the actual OS (Linux) visible interface name. For physical interfaces, the speed and the duplex settings are also displayed. |



Table 57: vif Fields (Continued)

| vif Output Field | Description  |
|------------------|--|
| Type:xxxxx       | <p>Type:Virtual HWaddr:00:00:5e:00:01:00 IPaddr:0</p> <p>The type of interface and its IP address, as defined by vrouter. The values can be different from what is seen in the OS. Types defined by vrouter include:</p> <ul style="list-style-type: none"> <li>• Virtual – Interface of a virtual machine (VM).</li> <li>• Physical – Physical interface (NIC) in the system.</li> <li>• Host – An interface toward the host.</li> <li>• Agent – An interface used to trap packets to the vrouter agent when decisions need to be made for the forwarding path.</li> </ul>  |
| Vrf:xxxxx        | <p>Vrf:65535 Flags:L3 MTU:1514 Ref:2</p> <p>The identifier of the vrf to which the interface is assigned, the flags set on the interface, the MTU as understood by vrouter, and a reference count of how many individual entities actually hold reference to the interface (mainly of debugging value).</p> <p>Flag options identify that the following are enabled for the interface:</p> <ul style="list-style-type: none"> <li>• P - Policy</li> <li>• L3 - Layer 3 forwarding</li> <li>• L2 - Layer 2 bridging</li> <li>• X - Cross connect mode, only set on physical and host interfaces, indicating that packets are moved between physical and host directly, with minimal intervention by vrouter. Typically set when the agent is not alive or not in good shape.</li> <li>• Mt - Mirroring transmit direction</li> <li>• Mr - Mirroring receive direction</li> <li>• Tc - Checksum offload on the transmit side. Valid only on the physical interface.</li> </ul> |

Table 57: vif Fields (Continued)

| vif Output Field | Description  |
|------------------|--|
| Rx               | RX packets:60 bytes:4873 errors:0<br>Packets received by vrouter from this interface.      |
| Tx               | TX packets:21 bytes:2158 errors:0<br>Packets transmitted out by vrouter on this interface. |

### vif Options

Use `vif --help` to display all options available for the `vif` command. Following is a brief description of each option.

**NOTE:** It is not recommended to use the following options unless you are very experienced with the system utilities.

```
# vif --help
Usage: vif [--create <intf_name> --mac <mac>]
        [--add <intf_name> --mac <mac> --vrf <vrf>
        --type [vhost|agent|physical|virtual][--policy, --mode <mode:x>]]
        [--delete <intf_id>]
        [--get <intf_id>][--kernel]
        [--set <intf_id> --vlan <vlan_id> --vrf <vrf_id>]
        [--list]
        [--help]
```

| Option   | Description   |
|----------|---|
| --create | Creates a 'Host' interface with name <intf_name> and mac <mac> on the host kernel. The 'vhost0' interface that you see on Linux is a typical example of invocation of this command. |

*(Continued)*

| Option   | Description  |
|----------|--|
| --add    | Adds the existing interfaces in the host OS to vrouter, with type and flag options.  |
| --delete | Deletes the interface from vrouter. The <intf_id> is the vrouter interface id as given by vif0/X, where X is the IID   |
| --get    | Displays a specific interface. The <intf_id> is the vrouter interface id, unless the command is appended by the '-kernel' option, in which case the ID can be the kernel ID.   |
| --set    | Set working parameters of an interface. The only ones supported are the vlan id and the vrf. The vlan id as understood by vrouter differs from what one typically expects, and is relevant as of now only for interfaces of service instances. |
| --list   | Display all of the interfaces of which the vrouter is aware.   |
| --help   | Display all options available for the current command.   |

## flow Command

Use the `flow` command to display all active flows in a system.

### Example: `flow -l`

Use `-l` to list everything in the flow table. The `-l` is the only relevant debugging option.

```
# flow -l
Flow table
  Index      Source:Port          Destination:Port    Proto(V)
-----
263484      1.1.1.252:1203      1.1.1.253:0        1 (3)
              (Action:F, S(nh):91, Statistics:22/1848)
```

```
379480      1.1.1.253:1203      1.1.1.252:0      1 (3)
              (Action:F, S(nh):75, Statistics:22/1848)
```

Each record in the flow table listing displays the index of the record, the source ip: source port, the destination ip: destination port, the inet protocol, and the source vrf to which the flow belongs.

Each new flow has to be approved by the vrouter agent. The agent does this by setting actions for each flow. There are three main actions associated with a flow table entry: Forward ('F'), Drop ('D'), and Nat ('N').

For NAT, there are additional flags indicating the type of NAT to which the flow is subject, including: SNAT (S), DNAT (D), source port translation (Ps), and destination port translation (Pd).

S(nh) indicates the source nexthop index used for the RPF check to validate that the traffic is from a known source. If the packet must go to an ECMP destination, E:X is also displayed, where 'X' indicates the destination to be used through the index within the ECMP next hop.

The Statistics field indicates the Packets/Bytes that hit this flow entry.

There is a Mirror Index field if the traffic is mirrored, listing the indices into the mirror table (which can be dumped by using `mirror --dump`).

If there is an explicit association between the forward and the reverse flows, as is the case with NAT, you will see a double arrow in each of the records with either side of the arrow displaying the flow index for that direction.

#### Example: flow -r

Use `-r` to view all of the flow setup rates.

```
# flow -r
New =  2, Flow setup rate =  3 flows/sec, Flow rate =  3 flows/sec, for last 548 ms
New =  2, Flow setup rate =  3 flows/sec, Flow rate =  3 flows/sec, for last 543 ms
New = -2, Flow setup rate = -3 flows/sec, Flow rate = -3 flows/sec, for last 541 ms
New =  2, Flow setup rate =  3 flows/sec, Flow rate =  3 flows/sec, for last 544 ms
New = -2, Flow setup rate = -3 flows/sec, Flow rate = -3 flows/sec, for last 542 ms
```

#### Example: flow --help

Use `--help` to display all options available for the flow command.

```
# flow --help
Usage:flow [-f flow_index][-d flow_index][-i flow_index]
           [--mirror=mirror table index]
           [-1]
```

```

-f <flow_index>    Set forward action for flow at flow_index <flow_index>
-d <flow_index>    Set drop action for flow at flow_index <flow_index>
-i <flow_index>    Invalidate flow at flow_index <flow_index>
--mirror           mirror index to mirror to
-l                List all flows
-r                Start dumping flow setup rate
--help            Print this help

```

## vrfstats Command

Use `vrfstats` to display statistics per next hop for a vrf. It is typically used to determine if packets are hitting the expected next hop.

### Example: `vrfstats --dump`

The `--dump` option displays the statistics for all vrfs that have seen traffic. In the following example, there was traffic only in Vrf 0 (the public vrf). `Receives` shows the number of packets that came in the fabric destined to this location. `Encaps` shows the number of packets destined to the fabric.

If there is VM traffic going out on the fabric, the respective tunnel counters will increment.

```

# vrfstats --dump
Vrf: 0
Discards 414, Resolves 3, Receives 165334
Ecmp Composites 0, L3 Mcast Composites 0, L2 Mcast Composites 0, Fabric Composites 0, Multi
Proto Composites 0
Udp Tunnels 0, Udp Mpls Tunnels 0, Gre Mpls Tunnels 0
L2 Encaps 0, Encaps 130955

```

### Example: `vrfstats --get 0`

Use `--get 0` to retrieve statistics for a particular vrf.

```

# vrfstats --get 0
Vrf: 0
Discards 418, Resolves 3, Receives 166929
Ecmp Composites 0, L3 Mcast Composites 0, L2 Mcast Composites 0, Fabric Composites 0, Multi
Proto Composites 0
Udp Tunnels 0, Udp Mpls Tunnels 0, Gre Mpls Tunnels 0
L2 Encaps 0, Encaps 132179

```

**Example: vrfstats --help**

```
Usage: vrfstats --get <vrf>
                --dump
                --help

--get <vrf>    Displays packet statistics for the vrf <vrf>

--dump         Displays packet statistics for all vrfs

--help         Displays this help message
```

**rt Command**

Use the `rt` command to display all routes in a vrf.

**Example: rt --dump**

The following example displays `inet` family routes for vrf `0`.

```
# rt --dump 0

Kernel IP routing table 0/0/unicast

Destination      PPL      Flags      Label      Nexthop
0.0.0.0/8        0        -          -          5
1.0.0.0/8        0        -          -          5
2.0.0.0/8        0        -          -          5
3.0.0.0/8        0        -          -          5
4.0.0.0/8        0        -          -          5
5.0.0.0/8        0        -          -          5
```

In this example output, the first line displays the routing table that is being dumped. In `0/0/unicast`, the first `0` is for the router id, the next `0` is for the vrf id, and `unicast` identifies the unicast table. The router maintains separate tables for unicast and multicast routes. By default, if the `-table` option is not specified, only the unicast table is dumped.

Each record in the table output specifies the destination prefix length, the parent route prefix length from which this route has been expanded, the flags for the route, the MPLS label if the destination is a VM in another location, and the next hop id. To understand the second field “PPL”, it is good to keep in mind that the unicast routing table is internally implemented as an ‘mtree’.

The `Flags` field can have two values. `L` indicates that the label field is valid, and `H` indicates that route should proxy arp for this IP.

The `Nexthop` field indicates the next hop ID to which the route points.

### Example: `rt --dump --table mcst`

To dump the multicast table, use the `-table` option with `mcst` as the argument.

```
# rt --dump 0 --table mcst

Kernel IP routing table 0/0/multicast

(Src,Group)                               Nexthop

0.0.0.0,255.255.255.255
```

### dropstats Command

Use the `dropstats` command to see packet drop counters in router.

### Example: `dropstats`

```
# dropstats

GARP                                     0

ARP notme                               12904

Invalid ARPs                             0

Invalid IF                               0

Trap No IF                               0

IF TX Discard                             0
```

|                           |    |
|---------------------------|----|
| IF Drop                   | 49 |
| IF RX Discard             | 0  |
| Flow Unusable             | 0  |
| Flow No Memory            | 0  |
| Flow Table Full           | 0  |
| Flow NAT no rflow         | 0  |
| Flow Action Drop          | 0  |
| Flow Action Invalid       | 0  |
| Flow Invalid Protocol     | 0  |
| Flow Queue Limit Exceeded | 0  |
| Discards                  | 34 |
| TTL Exceeded              | 0  |
| Mcast Clone Fail          | 0  |
| Cloned Original           | 0  |
| Invalid NH                | 2  |
| Invalid Label             | 0  |
| Invalid Protocol          | 0  |
| Rewrite Fail              | 0  |
| Invalid Mcast Source      | 0  |
| Push Fails                | 0  |



|                          |   |
|--------------------------|---|
| Pull Fails               | 0 |
| Duplicated               | 0 |
| Head Alloc Fails         | 0 |
| Head Space Reserve Fails | 0 |
| PCOW fails               | 0 |
| Invalid Packet           | 0 |
| Misc                     | 0 |
| Nowhere to go            | 0 |
| Checksum errors          | 0 |
| No Fmd                   | 0 |
| Ivalid VNID              | 0 |
| Fragment errors          | 0 |
| Invalid Source           | 0 |

### dropstats ARP Block

GARP packets from VMs are dropped by vrouter, an expected behavior. In the example output, the first counter GARP indicates how many packets were dropped.

ARP requests that are not handled by vrouter are dropped, for example, requests for a system that is not a host. These drops are counted by `ARP notme` counters.

The `Invalid ARPs` counter is incremented when the Ethernet protocol is ARP, but the ARP operation was neither a request nor a response.

### dropstats Interface Block

Invalid IF counters are incremented normally during transient conditions, and should not be a concern.

Trap No IF counters are incremented when vrouter is not able to find the interface to trap the packets to vrouter agent, and should not happen in a working system.

IF TX Discard and IF RX Discard counters are incremented when vrouter is not in a state to transmit and receive packets, and typically happens when vrouter goes through a reset state or when the module is unloaded.

IF Drop counters indicate packets that are dropped in the interface layer. The increase can typically happen when interface settings are wrong.

### **dropstats Flow Block**

When packets go through flow processing, the first packet in a flow is cached and the vrouter agent is notified so it can take actions on the packet according to the policies configured. If more packets arrive after the first packet but before the agent makes a decision on the first packet, then those new packets are dropped. The dropped packets are tracked by the Flow unusable counter.

The Flow No Memory counter increments when the flow block doesn't have enough memory to perform internal operations.

The Flow Table Full counter increments when the vrouter cannot install a new flow due to lack of available slots. A particular flow can only go in certain slots, and if all those slots are occupied, packets are dropped. It is possible that the flow table is not full, but the counter might increment.

The Flow NAT no rflow counter tracks packets that are dropped when there is no reverse flow associated with a forward flow that had action set as NAT. For NAT, the vrouter needs both forward and reverse flows to be set properly. If they are not set, packets are dropped.

The Flow Action Drop counter tracks packets that are dropped due to policies that prohibit a flow.

The Flow Action Invalid counter usually does not increment in the normal course of time, and can be ignored.

The Flow Invalid Protocol usually does not increment in the normal course of time, and can be ignored.

The Flow Queue Limit Exceeded usually does not increment in the normal course of time, and can be ignored.

### **dropstats Miscellaneous Operational Block**

The Discard counter tracks packets that hit a discard next hop. For various reasons interpreted by the agent and during some transient conditions, a route can point to a discard next hop. When packets hit that route, they are dropped.

The `TTL Exceeded` counter increments when the MPLS time-to-live goes to zero.

The `Mcast Clone Fail` happens when the vrouter is not able to replicate a packet for flooding.

The `Cloned Original` is an internal tracking counter. It is harmless and can be ignored.

The `Invalid NH` counter tracks the number of packets that hit a next hop that was not in a state to be used (usually in transient conditions) or a next hop that was not expected, or no next hops when there was a next hop expected. Such increments happen rarely, and should not continuously increment.

The `Invalid Label` counter tracks packets with an MPLS label unusable by vrouter because the value is not in the expected range.

The `Invalid Protocol` typically increments when the IP header is corrupt.

The `Rewrite Fail` counter tracks the number of times vrouter was not able to write next hop rewrite data to the packet.

The `Invalid Mcast Source` tracks the multicast packets that came from an unknown or unexpected source and thus were dropped.

The `Duplicated` counter tracks the number of duplicate packets that are created after dropping the original packets. An original packet is duplicated when generic send offload (GSO) is enabled in the vRouter or the original packet is unable to include the header information of the vRouter agent.

The `Invalid Source` counter tracks the number of packets that came from an invalid or unexpected source and thus were dropped.

The remaining counters are of value only to developers.

## **mpls Command**

The `mpls` utility command displays the input label map that has been programmed in the vrouter.

### **Example: mpls --dump**

The `-dump` command dumps the complete label map. The output is divided into two columns. The first field is the label and the second is the next hop corresponding to the label. When an MPLS packet with the specified label arrives in the vrouter, it uses the next hop corresponding to the label to forward the packet.

```
# mpls -dump
```

```
MPLS Input Label Map
```

```
Label    NextHop
```

```
-----
```

```
16      9
```

```
17     11
```

You can inspect the operation on nh 9 as follows:

```
# nh --get 9

Id:009 Type:Encap    Fmly: AF_INET  Flags:Valid, Policy,  Rid:0  Ref_cnt:4

    EncapFmly:0806 Oif:3 Len:14 Data:02 d0 60 aa 50 57 00 25 90 c3 08 69 08 00
```

The nh output shows that the next hop directs the packet to go out on the interface with index 3 (oif:3) with the given rewrite data.

To check the index of 3, use the following:

```
# vif -get 3

vif0/3 OS: tapd060aa50-57

    Type:Virtual HWaddr:00:00:5e:00:01:00 IPaddr:0

    Vrf:1 Flags:PL3L2 MTU:9160 Ref:6

    RX packets:1056 bytes:103471 errors:0

    TX packets:1041 bytes:102372 errors:0
```

The -get 3 output shows that the index of 3 corresponds to a tap interface that goes to a VM.

You can also dump individual entries in the map using the -get option, as follows:

```
# mpls -get 16
```

### MPLS Input Label Map

| Label | NextHop |
|-------|---------|
| ----- |         |
| 16    | 9       |

#### Example: mpls -help

```
# mpls -help

Usage: mpls --dump

        mpls --get <label>

        mpls --help

--dump  Dumps the mpls incoming label map

--get   Dumps the entry corresponding to label <label>
        in the label map

--help  Prints this help message
```

### mirror Command

Use the `mirror` command to dump the mirror table entries.

#### Example: Inspect Mirroring

The following example inspects a mirror configuration where traffic is mirrored from network `vn1` (1.1.1.0/24) to network `vn2` (2.2.2.0/24). A ping is run from 1.1.1.253 to 2.2.2.253, where both IPs are valid VM IPs, then the flow table is listed:

```
# flow -l

Flow table
```

| Index   | Source:Port    | Destination:Port | Proto(V) |
|---|----------------|------------------|----------|
| 135024  | 2.2.2.253:1208 | 1.1.1.253:0      | 1 (1)    |
| (Action:F, S(nh):17, Statistics:208/17472 Mirror Index : 0) |                |                  |          |
| 387324  | 1.1.1.253:1208 | 2.2.2.253:0      | 1 (1)    |
| (Action:F, S(nh):8, Statistics:208/17472 Mirror Index : 0)  |                |                  |          |

In the example output, Mirror Index:0 is listed, it is the index to the mirror table. The mirror table can be dumped with the `-dump` option, as follows:

```
# mirror --dump
```

| Mirror Table |         |       |            |
|--------------|---------|-------|------------|
| Index        | NextHop | Flags | References |
| 0            | 18      |       | 3          |

The mirror table entries point to next hops. In the example, the index 0 points to next hop 18. The References indicate the number of flow entries that point to this entry.

A next hop get operation on ID 18 is performed as follows:

```
# nh --get 18
```

|             |                  |                   |  |       |           |
|-------------|------------------|-------------------|--|-------|-----------|
| Id:018      | Type:Tunnel      | Fmly: AF_INET     | Flags:Valid, Udp,                              | Rid:0 | Ref_cnt:2 |
| Oif:0       | Len:14           | Flags Valid, Udp, | Data:00 00 00 00 00 00 00 25 90 c3 08 69 08 00 |       |           |
| Vrf:-1      | Sip:192.168.1.10 | Dip:250.250.2.253 |  |       |           |
| Sport:58818 | Dport:8099       |                   |  |       |           |

The `nh --get` output shows that mirrored packets go to a system with IP 250.250.2.253. The packets are tunneled as a UDP datagram and sent to the destination. `Vrf:-1` indicates that a lookup has to be done in the source Vrf for the destination.

You can also get an individual mirror table entry using the `--get` option, as follows:

```
# mirror --get 10

Mirror Table

Index   NextHop   Flags   References
-----
10      1         1       1
```

#### Example: mirror --help

```
# mirror --help

Usage: mirror --dump

       mirror --get <index>

       mirror --help

--dump  Dumps the mirror table

--get   Dumps the mirror entry corresponding to index <index>

--help  Prints this help message
```

### vxlan Command

The `vxlan` command can be used to dump the vxlan table. The vxlan table maps a network ID to a next hop, similar to an MPLS table.

If a packet comes with a vxlan header and if the VNID is one of those in the table, the vrouter will use the next hop identified to forward the packet.

**Example: vxlan --dump**

```
# vxlan --dump

VXLAN Table

VNID    NextHop
-----
 4      16
 5      16
```

**Example: vxlan --get**

You can use the `--get` option to dump a specific entry, as follows:

```
# vxlan --get 4

VXLAN Table

VNID    NextHop
-----
 4      16
```

**Example: vxlan --help**

```
# vxlan --help

Usage: vxlan --dump

       vxlan --get <vnid>

       vxlan --help

--dump Dumps the vxlan table

--get  Dumps the entry corresponding to <vnid>
```



```
--help Prints this help message
```

## nh Command

The `nh` command enables you to inspect the next hops that are known by the vrouter. Next hops tell the vrouter the next location to send a packet in the path to its final destination. The processing of the packet differs based on the type of the next hop. The next hop types are described in the following table.

| Next Hop Type     | Description   |
|-------------------|---|
| Receive           | Indicates that the packet is destined for itself and the vrouter should perform Layer 4 protocol processing. As an example, all packets destined to the host IP will hit the receive next hop in the default VRF. Similarly, all traffic destined to the VMs hosted by the server and tunneled inside a GRE will hit the receive next hop in the default VRF first, because the outer packet that carries the traffic to the VM is that of the server.                                |
| Encap (Interface) | Used only to determine the outgoing interface and the Layer 2 information. As an example, when two VMs on the same server communicate with each other, the routes for each of them point to an encap next hop, because the only information needed is the Layer 2 information to send the packet to the tap interface of the destination VM. A packet destined to a VM hosted on one server from a VM on a different server will also hit an encap next hop, after tunnel processing. |
| Tunnel            | Encapsulates VM traffic in a tunnel and sends it to the server that hosts the destination VM. There are different types of tunnel next hops, based on the type of tunnels used. Vrouter supports two main tunnel types for Layer 3 traffic: MPLSoGRE and MPLSoUDP. For Layer 2 traffic, a VXLAN tunnel is used. A typical tunnel next hop indicates the kind of tunnel, the rewrite information, the outgoing interface, and the source and destination server IPs.                   |
| Discard           | A catch-all next hop. If there is no route for a destination, the packet hits the discard next hop, which drops the packet.   |
| Resolve           | Used by the agent to lazy install Layer 2 rewrite information.  |

*(Continued)*

| Next Hop Type | Description   |
|---------------|---|
| Composite     | Groups a set of next hops, called component next hops or sub next hops. Typically used when multi-destination distribution is needed, for example for multicast, ECMP, and so on.               |
| Vxlan         | A VXLAN tunnel is used for Layer 2 traffic. A typical tunnel next hop indicates the kind of tunnel, the rewrite information, the outgoing interface, and the source and destination server IPs. |

**Example: nh --list**

```

Id:000 Type:Drop      Fmly: AF_INET  Flags:Valid,   Rid:0  Ref_cnt:1781

Id:001 Type:Resolve   Fmly: AF_INET  Flags:Valid,   Rid:0  Ref_cnt:244

Id:004 Type:Receive   Fmly: AF_INET  Flags:Valid, Policy,   Rid:0
                Ref_cnt:2 Oif:1

Id:007 Type:Encap     Fmly: AF_INET  Flags:Valid, Multicast,   Rid:0  Ref_cnt:3
                EncapFmly:0806 Oif:3 Len:14 Data:ff ff ff ff ff ff 00 25 90 c4 82 2c 08 00

Id:010 Type:Encap     Fmly:AF_BRIDGE Flags:Valid, L2,   Rid:0  Ref_cnt:3
                EncapFmly:0000 Oif:3 Len:0 Data:

Id:012 Type:Vxlan Vrf Fmly: AF_INET  Flags:Valid,   Rid:0  Ref_cnt:2
                Vrf:1

Id:013 Type:Composite Fmly: AF_INET  Flags:Valid, Fabric,   Rid:0  Ref_cnt:3
                Sub NH(label): 19(1027)

Id:014 Type:Composite Fmly: AF_INET  Flags:Valid, Multicast, L3,   Rid:0  Ref_cnt:3

```

```

Sub NH(label): 13(0) 7(0)

Id:015 Type:Composite Fmly:AF_BRIDGE Flags:Valid, Multicast, L2,  Rid:0 Ref_cnt:3

Sub NH(label): 13(0) 10(0)

Id:016 Type:Tunnel Fmly: AF_INET Flags:Valid, MPLSoGRE,  Rid:0 Ref_cnt:1

Oif:2 Len:14 Flags Valid, MPLSoGRE,  Data:00 25 90 aa 09 a6 00 25 90 c4 82 2c 08 00

Vrf:0 Sip:10.204.216.72 Dip:10.204.216.21

Id:019 Type:Tunnel Fmly: AF_INET Flags:Valid, MPLSoUDP,  Rid:0 Ref_cnt:7

Oif:2 Len:14 Flags Valid, MPLSoUDP,  Data:00 25 90 aa 09 a6 00 25 90 c4 82 2c 08 00

Vrf:0 Sip:10.204.216.72 Dip:10.204.216.21

Id:020 Type:Composite Fmly:AF_UNSPEC Flags:Valid, Multi Proto,  Rid:0 Ref_cnt:2

Sub NH(label): 14(0) 15(0)

```

**Example: nh --get**

Use the --get option to display information for a single next hop.

```

# nh -get 9

Id:009 Type:Encap Fmly:AF_BRIDGE Flags:Valid, L2,  Rid:0 Ref_cnt:4

EncapFmly:0000 Oif:3 Len:0 Data:

```

**Example: nh --help**

```

# nh -help

Usage: nh --list

nh --get <nh_id>

nh --help

```

```
--list Lists All Nexthops
```

```
--get <nh_id> Displays nexthop corresponding to <nh_id>
```

```
--help Displays this help message
```

# 4

PART

## Monitoring and Troubleshooting Contrail

---

[Configuring Traffic Mirroring to Monitor Network Traffic | 798](#)

[Understanding Contrail Analytics | 819](#)

[Configuring Contrail Analytics | 847](#)

[Using Contrail Analytics to Monitor and Troubleshoot the Network | 880](#)

[Common Support Answers | 970](#)

---

# Configuring Traffic Mirroring to Monitor Network Traffic

## IN THIS CHAPTER

- [Configuring Traffic Analyzers and Packet Capture for Mirroring | 798](#)
- [Configuring Interface Monitoring and Mirroring | 811](#)
- [Analyzer Service Virtual Machine | 812](#)
- [Mirroring Enhancements | 816](#)
- [Mapping VLAN Tags from a Physical NIC to a VMI \(NIC-Assisted Mirroring\) | 817](#)

## Configuring Traffic Analyzers and Packet Capture for Mirroring

### IN THIS SECTION

- [Traffic Analyzer Images | 799](#)
- [Configuring Traffic Analyzers | 799](#)
- [Setting Up Traffic Mirroring Using Monitor > Debug > Packet Capture | 799](#)
- [Setting Up Traffic Mirroring Using Configure > Networking > Services | 804](#)

Contrail provides traffic mirroring so you can mirror specified traffic to a traffic analyzer where you can perform deep traffic inspection. Traffic mirroring enables you to designate certain traffic flows to be mirrored to a traffic analyzer, where you can view traffic flows in great detail.

Use **Monitor > Debug > Packet Capture** to configure packets to be captured and “mirrored” to a virtual machine configured as a traffic analyzer. The packet activity can then be inspected for monitoring and troubleshooting purposes. This section demonstrates how to set up packet capture to mirror traffic packets to an analyzer.

## Traffic Analyzer Images

Before using the Contrail interface to configure traffic analyzers and packet capture for mirroring, make sure that the following analyzer images are available in the VM image list for your system. The traffic analyzer images are enhanced for viewing details of captured packets in Wireshark. When creating a policy for the traffic analyzer, the traffic analyzer instance should always have the **Mirror to** field selected in the policy, do not select the **Apply Service** field for a traffic analyzer.

- **analyzer-vm-console-qcow2**—Standard traffic analyzer; should be named **analyzer** in the image list. This type of traffic analyzer is always configured with a single interface, and the interface should be a **Left** interface.
- **analyzer-vm-console-two-if qcow2**—This type of traffic analyzer has two interfaces, **Left** and **Management**. This traffic analyzer can have any name except the name **analyzer**, which is reserved for the single interface analyzer.

**NOTE:** The analyzer-vm images are valid for all versions of Contrail. Download the images from the Contrail 1.0 software download page: <https://www.juniper.net/support/downloads/?p=contrail#sw>.

## Configuring Traffic Analyzers

In Contrail Controller, you use a two-part configuration to mirror captured packet traffic to a traffic analyzer, where the traffic details can be inspected. The configuration has the following steps:

1. Configure analyzer(s) on the host.
2. Set up rules for packet capture.

Additionally, there are two ways to configure the packet capture for the analyzers from within the Contrail interface:

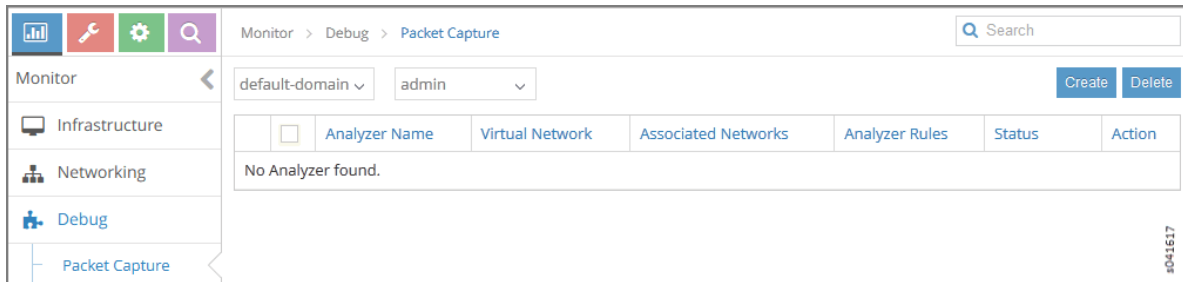
- Configure from **Monitor > Debug > Packet Capture**
- Configure from **Configure > Networking > Services**

## Setting Up Traffic Mirroring Using Monitor > Debug > Packet Capture

The following are the steps needed to set up packet capture in order to “mirror” the traffic to an analyzer VM for the purpose of reviewing various aspects of packet traffic moving through the system.

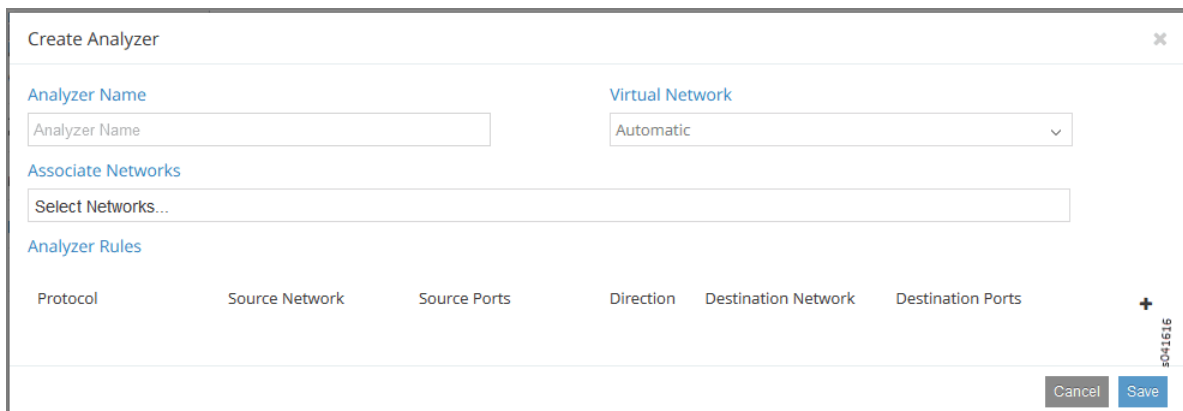
1. Select **Monitor > Debug > Packet Capture**. The **Packet Capture** screen appears; see [Figure 194 on page 800](#).

Figure 194: Packet Capture



2. Click **Create** to add an analyzer; see [Figure 195 on page 800](#).

Figure 195: Create Analyzer



3. In the **Analyzer Name** field, enter a name for the analyzer and in the **Virtual Network** field, select **Automatic** or select a specific virtual network from the drop-down list of available networks; click **Save** when finished.
4. To create rules for the analyzer, in the lower portion of the **Create Analyzer** screen, click the **+** button to add a rule.

The **Analyzer Rules** fields appear; see [Figure 196 on page 801](#).



Figure 196: Analyzer Rules

5. Select the rules to apply to determine which packets should be “mirrored”—sent to the analyzer for monitoring.

See [Table 58 on page 801](#) for guidelines for completing the rule fields.

Table 58: Analyzer Rule Fields

| Field              | Description   |
|--------------------|---|
| <b>IP Protocol</b> | <p>Select from a list to define from which protocol packets are to be captured:</p> <ul style="list-style-type: none"> <li>• ANY</li> <li>• TCP</li> <li>• UDP</li> <li>• ICMP</li> </ul> |

Table 58: Analyzer Rule Fields (*Continued*)

| Field                      | Description   |
|----------------------------|---|
| <b>Source Network</b>      | <p>Select from a list the source network from which packets are to be captured:</p> <ul style="list-style-type: none"> <li>• any</li> <li>• local</li> <li>• <i>domain:network 1</i></li> <li>• <i>domain:network 2</i></li> <li>• <i>domain:network .....</i></li> </ul> |
| <b>Source Ports</b>        | <p>If you want to capture only those packets that originate from a specific port number, enter the port number.</p>   |
| <b>Direction</b>           | <p>Select the direction of flow for the packets to be captured:</p> <ul style="list-style-type: none"> <li>• Bidirectional</li> <li>• Unidirectional</li> </ul>   |
| <b>Destination Network</b> | <p>Select from a list the destination network for the packets to be captured:</p> <ul style="list-style-type: none"> <li>• any</li> <li>• local</li> <li>• <i>domain:network 1</i></li> <li>• <i>domain:network 2</i></li> <li>• <i>domain:network .....</i></li> </ul>   |
| <b>Destination Ports</b>   | <p>If you want to capture only those packets that are destined to a specific port number, enter the port number.</p>  |
| Cancel, Save               | <p>When finished, click <b>Save</b> to commit your selections, or click <b>Cancel</b> to clear the entries and start over.</p>  |

- To associate virtual networks with the analyzer, click the **Associate Networks** field in the center portion of the screen. Select from a drop-down list of available networks the networks to associate with this analyzer; see [Figure 197 on page 803](#).

**Figure 197: Create Analyzer Associate Networks**

**NOTE:** If there is already a network policy attached to the virtual network selected, any conflicting rules configured for the analyzer will not take effect.

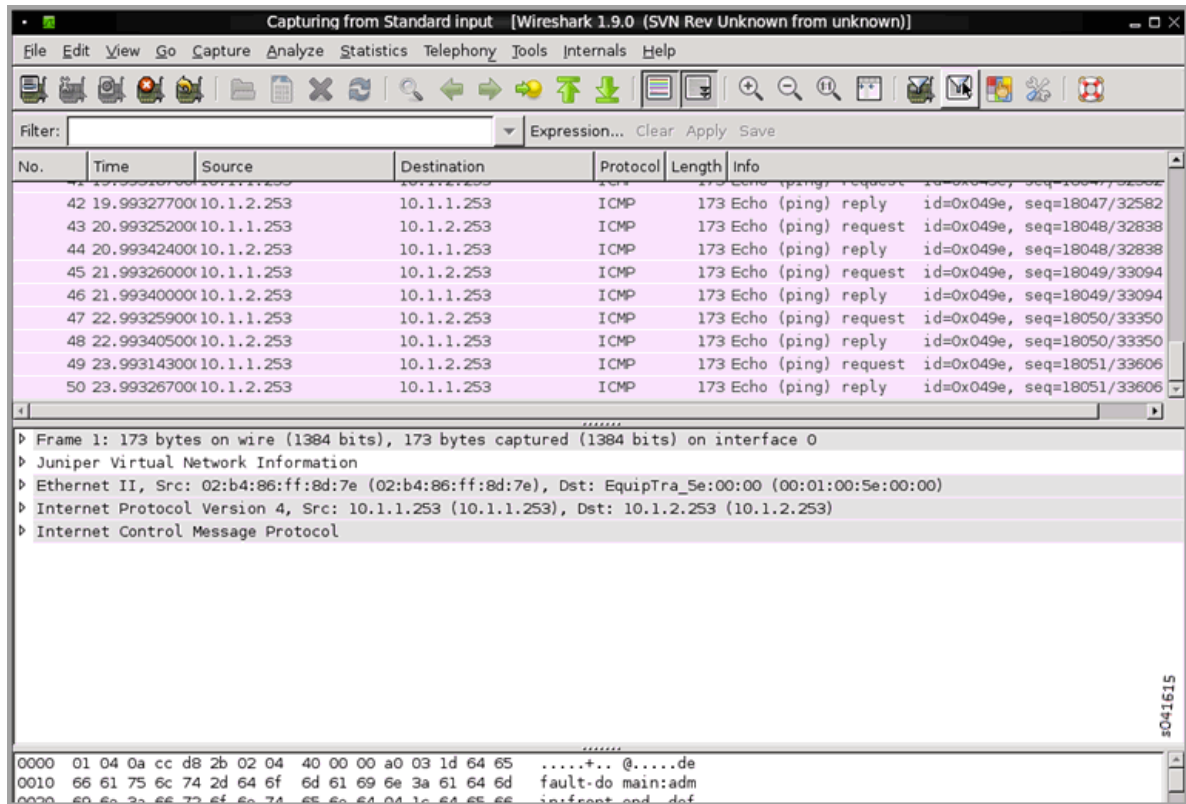
- View the analyzer activity from **Monitor > Debug > Packet Capture**. For the selected analyzer, click in the **Action** column and select **View Analyzer**; see [Figure 198 on page 803](#).

**Figure 198: Launch Analyzer VM**

| Analyzer Name | Virtual Network | Associated Networks | Analyzer Rules  | Status | Action        |
|---------------|-----------------|---------------------|---|--------|---------------|
| demo          | Automatic       | backend<br>frontend | protocol any network default-domain:admin:frontend port any <=> network default-domain:admin:backend port any mirror_to default-domain:admin:demo | Active | View Analyzer |

- The Wireshark **Packet Capture Display** appears; see [Figure 199 on page 804](#).

Figure 199: Packet Capture Display



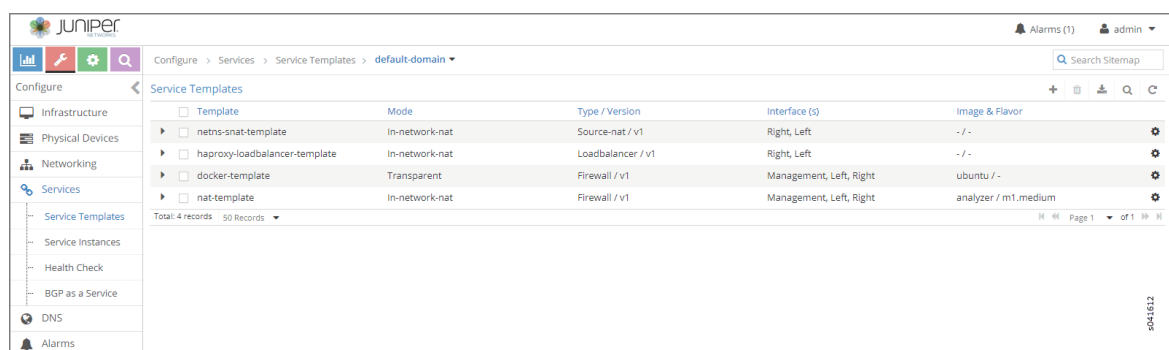
## Setting Up Traffic Mirroring Using Configure > Networking > Services

You can set up packet capture for mirroring to an analyzer within a service chain utilizing more than one interface by starting with a service template. The following procedure provides the steps needed.

1. Access **Configure > Services > Service Templates**.

The **Service Templates** screen appears; see [Figure 200 on page 804](#).

Figure 200: Service Templates



- To create a new service template, click the + icon.  
The **Create** window appears. Select the Service Template tab; see [Figure 201 on page 805](#).

**Figure 201: Create Service Template**

- Complete the fields by using the guidelines in [Table 59 on page 805](#).

**Table 59: Create Service Template Fields**

| Field                      | Description  |
|----------------------------|--|
| <b>Name</b>                | Enter a descriptive text name for this service template.   |
| <b>Version</b>             | Select <b>v2</b> from the drop-down list to indicate that this service template is based on templates version 2, valid for Contrail 3.0 and later. |
| <b>Virtualization Type</b> | Select <b>Virtual Machine</b> from the drop-down list to indicate the virtualization type for mirroring for this template.                         |

Table 59: Create Service Template Fields (Continued)

| Field               | Description   |
|---------------------|---|
| <b>Service Mode</b> | Select <b>Transparent</b> from the drop-down list to indicate that this service template is for transparent mirroring.  |
| <b>Service Type</b> | Select <b>Analyzer</b> from the drop-down list to indicate that this service template is for a traffic analyzer.  |
| <b>Interface(s)</b> | From the drop-down list, click the check boxes to indicate which interface types are used for this analyzer service template: <ul style="list-style-type: none"> <li>• Left</li> <li>• Right</li> <li>• Management</li> </ul> |
| <b>Save</b>         | When finished, click <b>OK</b> to commit the changes  |
| <b>Cancel</b>       | Click <b>Cancel</b> to clear the fields and start over.   |

4. Create a service instance by clicking the **Service Instances** link and clicking the + icon.

The **Create** window appears; make sure the Service Instance tab is selected. See [Figure 202 on page 807](#).

Figure 202: Create Service Instances

5. Complete the fields by using the guidelines in [Table 60 on page 807](#).

Table 60: Create Service Instances Fields

| Field                   | Description   |
|-------------------------|---|
| <b>Name</b>             | Enter a text name for this service instance.  |
| <b>Service Template</b> | Select from a drop-down list of available service templates the template to use for this service instance, analyzer-service-template in this example. |
| <b>Interface Type</b>   | Each interface configured in the service template for this instance appears in a list.  |
| <b>Virtual Network</b>  | Select from a drop-down list of available virtual networks the network for each interface that is configured for the instance.                        |

Table 60: Create Service Instances Fields *(Continued)*

| Field         | Description   |
|---------------|---|
| <b>Save</b>   | Click <b>Save</b> to commit your changes.                 |
| <b>Cancel</b> | Click <b>Cancel</b> to clear your changes and start over. |

- To create a network policy rule for this service instance, click **Configure > Networking > Policies**. The **Policies** window appears. Click the + icon to get to the **Create** window; see [Figure 203 on page 808](#).

Figure 203: Create Policy

- 
- 
- 
- 
- 
- 
- 
- Enter a name for the policy, then click the + icon in the lower portion of the screen to configure rules for the policy, see [Figure 204 on page 809](#).



Figure 204: Create Policy Rules

The screenshot shows a 'Create' dialog box with a close button (X) in the top right corner. Below the title bar, there are two tabs: 'Policy' (selected) and 'Permissions'. Under the 'Policy' tab, there is a 'Policy Name' field containing the text 'analyzer-policy'. Below that is a 'Policy Rule(s)' section with a table. The table has columns: Action, Protocol, Source, Ports, Direction, Destination, Ports, Log, Services, Mirror, QoS, and a plus sign. The first row of the table has the following values: Action: PASS (highlighted with a blue box), Protocol: ANY, Source: ANY (All Networks i...), Ports: ANY, Direction: <>, Destination: ANY (All Networks i...), Ports: (empty), Log: (checkbox), Services: (checkbox), Mirror: (checkbox), QoS: (checkbox), and a plus/minus sign. At the bottom right of the dialog, there are 'Cancel' and 'Save' buttons. A vertical ID 's041853' is visible on the right side of the dialog.

9. To add policy rules, complete the fields, using the guidelines in [Table 61 on page 809](#).

**NOTE:** When there is a network policy attached to the virtual network, any conflicting rules configured for the analyzer will not take effect.

Table 61: Add Rule Fields

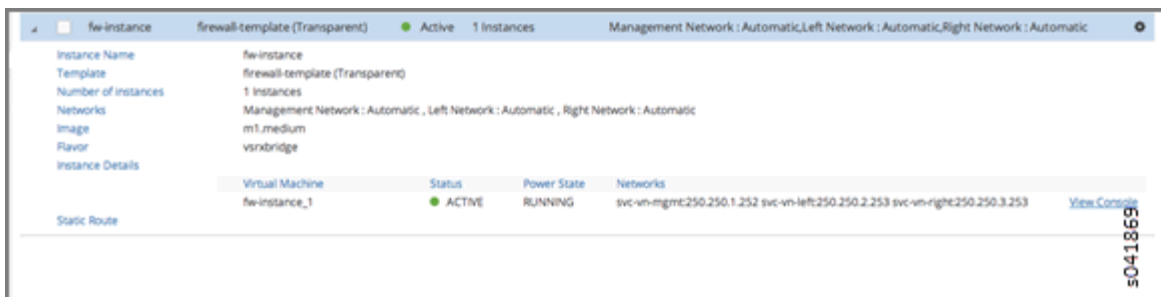
| Field            | Description  |
|------------------|--|
| <b>Action</b>    | Select PASS or DENY as the rule action.  |
| <b>Protocol</b>  | Select the protocol for the policy rule, or select ANY.  |
| <b>Source</b>    | Select from multiple drop-down lists the source for this rule, including options under CIDR, Network, Policy, or Security Group.   |
| <b>Ports</b>     | Select from a drop-down list the source ports for the rule.  |
| <b>Direction</b> | Select the direction of flow for the packets to be captured: <ul style="list-style-type: none"> <li>• &lt;&gt; (bidirectional)</li> <li>• &gt; (unidirectional)</li> </ul> |

**Table 61: Add Rule Fields (Continued)**

| Field              | Description   |
|--------------------|---|
| <b>Destination</b> | Select from multiple drop-down lists the destination for this rule, including options under CIDR, Network, Policy, or Security Group. |
| <b>Ports</b>       | Select from a list the destination ports for the packets to be captured.  |
| <b>check boxes</b> | Check any box that applies to this rule: Log, Services, Mirror, QoS.  |
| <b>Save</b>        | Click <b>Save</b> to commit your changes.   |
| <b>Cancel</b>      | Click <b>Cancel</b> to clear your changes and start over.   |

10. When finished, click **Save**.
11. To verify packet capture, at **Configure > Services > Service Instances**, select the analyzer service instance and click **View Console**.

The packet capture displays; see [Figure 205 on page 810](#). The analyzer service VM launches the Contrail-enhanced Wireshark as it starts and captures the mirrored packets destined to this service.

**Figure 205: Service Instances View Console**

## RELATED DOCUMENTATION

[Configuring Interface Monitoring and Mirroring | 811](#)

Mirroring Enhancements | 816

Analyzer Service Virtual Machine | 812

Mapping VLAN Tags from a Physical NIC to a VMI (NIC-Assisted Mirroring) | 817

## Configuring Interface Monitoring and Mirroring

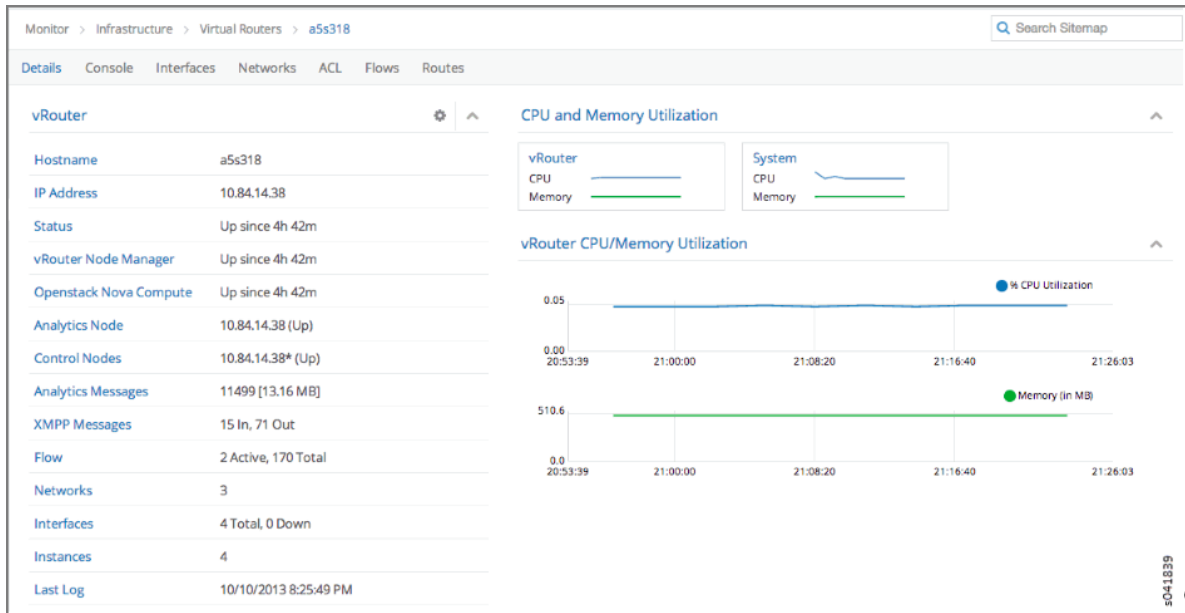
Contrail supports user monitoring of traffic on any guest virtual machine interface when using the Juniper Contrail user interface.

When interface monitoring (packet capture) is selected, a default analyzer is created and all traffic from the selected interface is mirrored and sent to the default analyzer. If a mirroring instance is already launched, the traffic will be redirected to the selected instance. The interface traffic is only mirrored during the time that the monitor packet capture interface is in use. When the capture screen is closed, interface mirroring stops.

To configure interface mirroring:

1. Select **Monitor > Infrastructure > Virtual Routers**, then select the vRouter that has the interface to mirror.
2. In the list of attributes for the vRouter, select **Interfaces**; see [Figure 206 on page 811](#).

Figure 206: Individual vRouter



A list of interfaces for that vRouter appears.

- For the interface to mirror, click the Action icon in the last column and select the option Packet Capture; see [Figure 207 on page 812](#).

**Figure 207: Interfaces**

| Name           | Label | Status | Network                         | IP Address    | Floating IP | Instance                             |
|----------------|-------|--------|---------------------------------|---------------|-------------|--------------------------------------|
| tap32e8b4a6-4b | 19    | Up     | default-domain:demo:svc-vn-left | 250.250.2.252 | None        | 2797154f-aa4e-4e1a-a4dc-e7aa95a5a4f3 |
| tap347d03ac-c2 | 16    | Up     | default-domain:demo:svc-vn-left | 250.250.2.253 | None        | 27b87ff5-5dc2-4b49-bd08-b15e96c311b3 |
| tap53aef323-79 | 17    | Up     | default-domain:demo:fe          | 7.7.7.253     | None        | cee63d6b-8843-4382-a699-fd01e85781b9 |
| tap96b4d43c-75 | 18    | Up     | default-domain:demo:be          | 8.8.8.253     | None        | 26c28bb7-0259-42ee-ad92-0c29c182dbfe |

The mirror packet capture starts and displays at this screen.

The mirror packet capture stops when you exit this screen.

## RELATED DOCUMENTATION

[Configuring Traffic Analyzers and Packet Capture for Mirroring | 798](#)

[Mirroring Enhancements | 816](#)

[Analyzer Service Virtual Machine | 812](#)

[Mapping VLAN Tags from a Physical NIC to a VMI \(NIC-Assisted Mirroring\) | 817](#)

## Analyzer Service Virtual Machine

### IN THIS SECTION

- [Packet Format for Analyzer | 813](#)
- [Metadata Format | 813](#)
- [Wireshark Changes | 814](#)
- [Troubleshooting Packet Display | 815](#)

The analyzer service virtual machine (`analyzer-vm-console.qcow2`) launches a Contrail-enhanced version of the network protocol analyzer Wireshark as the analyzer starts capturing mirror packets destined to the analyzer service.

## Packet Format for Analyzer

The analyzer uses the PCAP format, which has these parts:

- Global header
- PCAP packet header
- Packet data (original packet data)

The global header is added by the analyzer service by means of the Wireshark instance. The vRouter DP uses the configured UDP session to send mirrored packets to the analyzer, adding the PCAP packet header to the packet data as it sends it over the UDP socket to the analyzer.

The following additional information is also added to the packet data as metadata:

- Captured host (IP address)
- Ingress or egress
- Action (Pass/Deny/...)
- Source VN (fully qualified name)
- Destination VN (fully qualified name)

In the existing PCAP, a network ID is added in the global header. The metadata (additional flow information) is added in front of the existing packet as follows.

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Global header | Packet header| Meta data |Packet data| Packet header| Meta data |Packet data|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

## Metadata Format

The metadata is in type-length-value (TLV) format as follows.

1. Type: 1 Byte
2. Length: 1 Byte
3. Value: up to length

Type

1. 1 - Captured host IPv4 address
2. 2 - Action field
3. 3 - Source VN
4. 4 - Destination VN
5. 255 - TLV end

*Captured host address*

Length is 4 or 16 bytes based on IP address type

*Action field*

Length is 2 bytes. Multiple bits might be turned on, if there are more actions. Ingress or egress bit will be present in the Action field.

*Source VN or Destination VN*

Length is variable and up to 256 characters

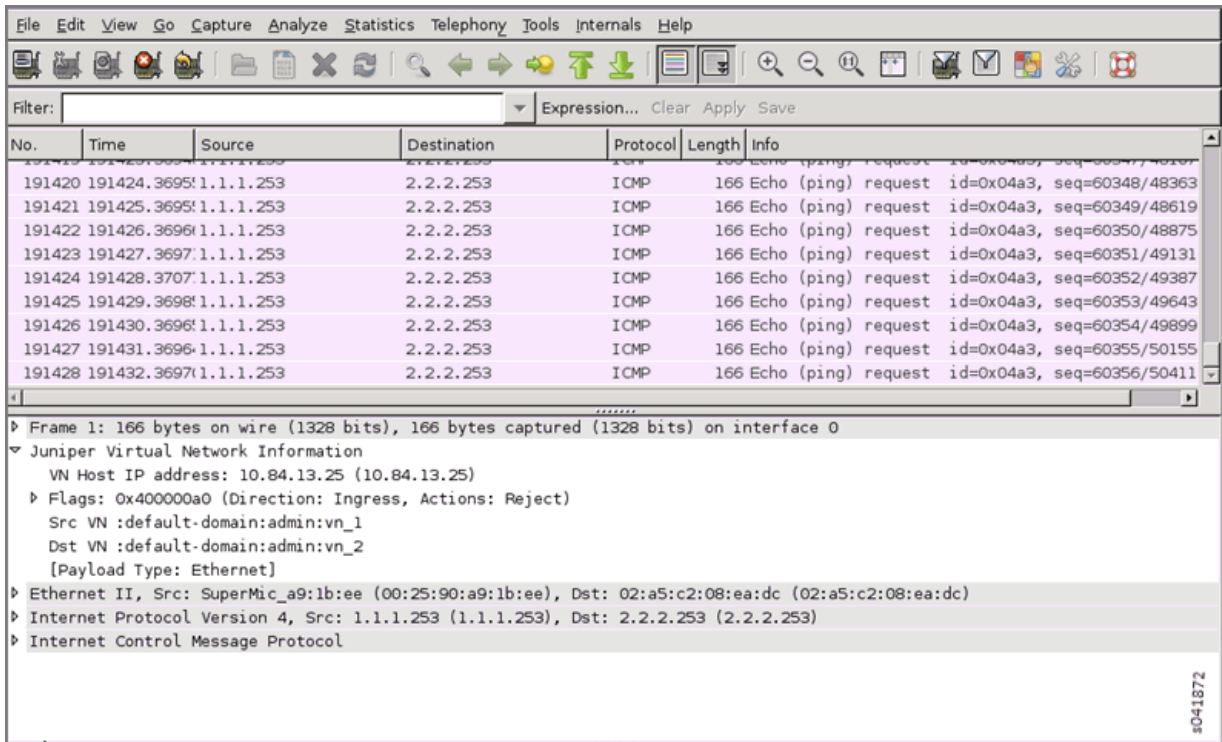
*TLV end*

A special type 255 (0xFF) is used to identify the end of TLV entries. The TLV end must be last, at the end of the metadata.

## Wireshark Changes

A plugin is added to the Wireshark code. The plugin parses the metadata and displays the packet fields; see example in [Figure 208 on page 815](#).

Figure 208: Wireshark Packet Display



## Troubleshooting Packet Display

Follow these steps if the packets are not displaying:

1. Use `tcpdump` on the tap interfaces to see if packets are going towards the analyzer VM.
2. Check `introspect` to see whether the flow action has mirror activity in it or not.

## RELATED DOCUMENTATION

[Configuring Traffic Analyzers and Packet Capture for Mirroring | 798](#)

[Configuring Interface Monitoring and Mirroring | 811](#)

[Mirroring Enhancements | 816](#)

[Mapping VLAN Tags from a Physical NIC to a VMI \(NIC-Assisted Mirroring\) | 817](#)

## Mirroring Enhancements

### IN THIS SECTION

- [Mirroring Specified Traffic | 816](#)
- [Configuring Headers and Next Hops | 816](#)
- [How Mirroring is Implemented | 817](#)

### Mirroring Specified Traffic

Specific traffic can be mirrored to a traffic analyzer in Contrail by:

- Configuring rules to identify the flows to be mirrored, and
- Specifying the analyzer to which the traffic is mirrored

Additionally, mirroring can be configured on virtual machine (VM) interfaces to send all the traffic to and from the interface to the specified analyzer.

### Configuring Headers and Next Hops

When a packet is mirrored, a Juniper header is added to provide additional information in the analyzer, then the packet is encapsulated and sent to the destination.

Starting with Contrail 3.x releases, mirroring is enhanced with the following options:

- Option to control addition of the Juniper header in the mirrored packet.
  - When disabled, the Juniper header is not added to the mirrored packet.
- Option to control whether the next hop used is dynamic or static.
  - If dynamic is selected, the next hop based on the destination is used. Packets are forwarded to the destination based on the encapsulation priority.
  - If static is chosen, the next hop is created for the specified destination with VxLAN encapsulation using the configured VNI, destination VTEP, and MAC to transmit the mirrored packets.

The following combinations are supported:

- Dynamic next hop with Juniper header added



The default combination and the only supported case up to Release 3.0.2

- Dynamic next hop, without Juniper header
- Static next hop, without Juniper header, with the original Layer 2 packet

## How Mirroring is Implemented

The Contrail vrouter agent adds a mirror entry in the vrouter and points to the next hop to be used. The data for the Juniper header is taken from the flow entry. For interface mirroring, the Juniper header has a TLV in the metadata to use the interface name instead of providing a destination VN.

For more information about implementation details, see <https://github.com/Juniper/contrail-controller/wiki/Mirroring>.

### RELATED DOCUMENTATION

---

[Configuring Traffic Analyzers and Packet Capture for Mirroring | 798](#)

---

[Configuring Interface Monitoring and Mirroring | 811](#)

---

[Analyzer Service Virtual Machine | 812](#)

---

[Mapping VLAN Tags from a Physical NIC to a VMI \(NIC-Assisted Mirroring\) | 817](#)

## Mapping VLAN Tags from a Physical NIC to a VMI (NIC-Assisted Mirroring)

When mirroring is enabled, the vRouter throughput reduces because of the additional packet handling overhead caused by cloning the packet to be mirrored, encapsulating it in the required header, and forwarding it to the mirror destination. Impact to throughput increases in proportion to the amount of traffic that needs to be mirrored.

A solution to avoid impact on throughput due to mirroring is to use the mirroring capabilities of an installed Network Interface Card (NIC).

Contrail Release 4.0 has the ability to mirror specific traffic to a traffic analyzer or to a physical probe using the Network interface card (NIC) instead of the vRouter to mirror packets. When NIC-assisted mirroring is enabled, ingress packets to be mirrored sent from a VM are routed to the NIC with a configured VLAN tag. The NIC is configured for VLAN port-mirroring and mirrors any packet with the VLAN tag.

In this approach, the vRouter doesn't mirror the packets. When NIC-assisted mirroring is enabled, the ingress packets coming from the VM that are to be mirrored are sent to the NIC with a configured VLAN tag.

The NIC is programmed to do VLAN port mirroring, so that any packet with the configured VLAN is mirrored additionally by the NIC. This change in vRouter is only for traffic coming from the VMs. Traffic coming from the fabric is directly mirrored from the NIC itself and there is no additional mirroring need in vRouter. The programming of the NIC itself for appropriate mirroring is outside the scope of the current activity. An example is the Niantic 82599 10G NIC, which supports VLAN port mirroring options.

The following are cautions to observe when using NIC-assisted mirroring:

- VM traffic sent to another VM running on the same compute node will not be mirrored when NIC-assisted mirroring is selected.
- Traffic coming in from the fabric interface will not be mirrored.
- When a VLAN interface is used as the fabric interface, traffic will be tagged first with the NIC-assisted mirroring VLAN, followed by the VLAN tag on the fabric interface. The NIC-assisted mirroring VLAN will be the inner tag and the fabric interface VLAN will be the outer tag.

The NIC must be programmed for VLAN port mirroring. While configuring mirroring in Contrail, the user can indicate NIC-assisted mirroring with the VLAN tag. The Contrail UI supports NIC-assisted mirroring configuration in the Ports page and in the Policies page with an additional flag for NIC-assisted mirroring and the VLAN tag to be used.

## RELATED DOCUMENTATION

---

[Configuring Traffic Analyzers and Packet Capture for Mirroring | 798](#)

---

[Configuring Interface Monitoring and Mirroring | 811](#)

---

[Mirroring Enhancements | 816](#)

---

[Analyzer Service Virtual Machine | 812](#)

# Understanding Contrail Analytics

## IN THIS CHAPTER

- [Understanding Contrail Analytics | 819](#)
- [Contrail Alerts | 820](#)
- [Underlay Overlay Mapping in Contrail | 824](#)

## Understanding Contrail Analytics

Contrail is a distributed system of compute nodes, control nodes, configuration nodes, database nodes, web UI nodes, and analytics nodes.

The analytics nodes are responsible for the collection of system state information, usage statistics, and debug information from all of the software modules across all of the nodes of the system. The analytics nodes store the data gathered across the system in a database that is based on the Apache Cassandra open source distributed database management system. The database is queried by means of an SQL-like language and representational state transfer (REST) APIs.

System state information collected by the analytics nodes is aggregated across all of the nodes, and comprehensive graphical views allow the user to get up-to-date system usage information easily.

Debug information collected by the analytics nodes includes the following types:

- System log (syslog) messages—informational and debug messages generated by system software components.
- Object log messages—records of changes made to system objects such as virtual machines, virtual networks, service instances, virtual routers, BGP peers, routing instances, and the like.
- Trace messages—records of activities collected locally by software components and sent to analytics nodes only on demand.

Statistics information related to flows, CPU and memory usage, and the like is also collected by the analytics nodes and can be queried at the user interface to provide historical analytics and time-series information. The queries are performed using REST APIs.

Analytics data is written to a database in Contrail. The data expires after the default time-to-live (TTL) period of 48 hours. This default TTL time can be changed as needed by changing the value of the `database_ttl` value in the cluster configuration.

## RELATED DOCUMENTATION

*[Contrail Alerts](#)*

*[Analytics Scalability](#)*

*[High Availability for Analytics](#)*

*[Ceilometer Support in a Contrail Cloud](#)*

*[Underlay Overlay Mapping in Contrail](#)*

*[Monitoring the System](#)*

*[Debugging Processes Using the Contrail Introspect Feature](#)*

*[Monitor > Infrastructure > Dashboard](#)*

*[Monitor > Infrastructure > Control Nodes](#)*

*[Monitor > Infrastructure > Virtual Routers](#)*

*[Monitor > Infrastructure > Analytics Nodes](#)*

*[Monitor > Infrastructure > Config Nodes](#)*

*[Monitor > Networking](#)*

[Understanding Flow Sampling | 959](#)

*[Query > Flows](#)*

*[Query > Logs](#)*

*[System Log Receiver in Contrail Analytics](#)*

*[Example: Debugging Connectivity Using Monitoring for Troubleshooting](#)*

## Contrail Alerts

### IN THIS SECTION

- [Alert API Format | 821](#)
- [Analytics APIs for Alerts | 822](#)
- [Analytics APIs for SSE Streaming | 823](#)

Starting with Contrail 3.0 and greater, Contrail alerts are provided on a per-user visible entity (UVE) basis.

Contrail analytics raise or clear alerts using Python-coded rules that examine the contents of the UVE and the configuration of the object. Some rules are built in. Others can be added using Python *stevedore* plugins.

This topic describes Contrail alerts capabilities.

## Alert API Format

The Contrail alert analytics API provides the following:

- Read access to the alerts as part of the UVE GET APIs.
- Alert acknowledgement using POST requests.
- UVE and alert streaming using server-sent events (SSEs).

For example:

GET `http://<analytics-ip>:8081/analytics/uves/control-node/a6s40?flat`

```
{
  NodeStatus: {...},
  ControlCpuState: {...},
  UVEAlarms: {
    alarms: [
      {
        description: [
          {
            value: "0 != 2",
            rule: "BgpRouterState.num_up_bgp_peer != BgpRouterState.num_bgp_peer"
          }
        ],
        ack: false,
        timestamp: 1442995349253178,
        token: "eyJ0aW1lc3RhbXAiOiAxNDQyOTk1MzQ5MTUzMTc4LCAiaHR0cF9wb3J0Ijog
NTk5NSwgImhvc3RfaXAiOiAiMTAuODQuMTMuNDAlfQ=="
      }
    ]
  }
}
```

```

        type: "BgpConnectivity",
        severity: 4
    }
]
},
BgpRouterState: {...}
}

```

In the example:

- Alerts are raised on a per-UVE basis and can be retrieved by a GET on a UVE.
- An ack indicates if the alert has been acknowledged or not.
- A token is used by clients when requesting acknowledgements

## Analytics APIs for Alerts

The following examples show the API to use to display alerts and alarms and to acknowledge alarms.

- To retrieve a list of alerts raised against the control node named aXXsYY.

```
GET http://<analytics-ip>:<rest-api-port>/analytics/uves/control-node/aXXsYY&cfilt=UVEAlarms
```

This is available for all UVE table types.

- To retrieve a list of all alarms in the system.

```
GET http://<analytics-ip>:<rest-api-port>/analytics/alarms
```

- To acknowledge an alarm.

```
POST http://<analytics-ip>:<rest-api-port>/analytics/alarms/acknowledge
Body: {"table": <object-type>,"name": <key>, "type": <alarm type>, "token": <token>}
```

Acknowledged and unacknowledged alarms can be queried specifically using the following URL query parameters along with the GET operations listed previously.

```
ackFilt=True
ackFilt=False
```

## Analytics APIs for SSE Streaming

The following examples show the API to use to retrieve all or portions of SE streams.

- To retrieve an SSE-based stream of UVE updates for the control node alarms.

```
GET http://<analytics-ip>:<rest-api-port>/analytics/uve-stream?tablefilt=control-node
```

This is available for all UVE table types. If the tablefilt URL query parameter is not provided, all UVEs are retrieved.

- To retrieve only the alerts portion of the SSE-based stream of UVE updates instead of the entire content.

```
GET http://<analytics-ip>:<rest-api-port>/analytics/alarm-stream?tablefilt=control-node
```

This is available for all UVE table types. If the tablefilt URL query parameter is not provided, all UVEs are retrieved.

## Built-in Node Alerts

The following built-in node alerts can be retrieved using the APIs listed in *Analytics APIs for Alerts*.

```
control-node: {
  PartialSysinfoControl: "Basic System Information is absent for this node in
  BgpRouterState.build_info",
  ProcessStatus: "NodeMgr reports abnormal status for process(es) in NodeStatus.process_info",
  XmppConnectivity: "Not enough XMPP peers are up in BgpRouterState.num_up_bgp_peer",
  BgpConnectivity: "Not enough BGP peers are up in BgpRouterState.num_up_bgp_peer",
  AddressMismatch: "Mismatch between configured IP Address and operational IP Address",
  ProcessConnectivity: "Process(es) are reporting non-functional components in
  NodeStatus.process_status"
},

vrouter: {
  PartialSysinfoCompute: "Basic System Information is absent for this node in
  VrouterAgent.build_info",
  ProcessStatus: "NodeMgr reports abnormal status for process(es) in NodeStatus.process_info",
  ProcessConnectivity: "Process(es) are reporting non-functional components in
  NodeStatus.process_status",
  VrouterInterface: "VrouterAgent has interfaces in error state in VrouterAgent.error_intf_list",
```

```
VrouterConfigAbsent: "Vrouter is not present in Configuration",
},

config-node: {
  PartialSysinfoConfig: "Basic System Information is absent for this node in
  ModuleCpuState.build_info",
  ProcessStatus: "NodeMgr reports abnormal status for process(es) in NodeStatus.process_info",
  ProcessConnectivity: "Process(es) are reporting non-functional components in
  NodeStatus.process_status"
},

analytics-node: {
  ProcessStatus: "NodeMgr reports abnormal status for process(es) in NodeStatus.process_info"
  PartialSysinfoAnalytics: "Basic System Information is absent for this node in
  CollectorState.build_info",
  ProcessConnectivity: "Process(es) are reporting non-functional components in
  NodeStatus.process_status"
},

database-node: {
  ProcessStatus: "NodeMgr reports abnormal status for process(es) in NodeStatus.process_info",
  ProcessConnectivity: "Process(es) are reporting non-functional components in
  NodeStatus.process_status"
},
```

## Underlay Overlay Mapping in Contrail

### IN THIS SECTION

- [Overview: Underlay Overlay Mapping using Contrail Analytics | 825](#)
- [Underlay Overlay Analytics Available in Contrail | 825](#)
- [Architecture and Data Collection | 826](#)
- [New Processes/Services for Underlay Overlay Mapping | 826](#)
- [External Interfaces Configuration for Underlay Overlay Mapping | 827](#)
- [Physical Topology | 827](#)
- [SNMP Configuration | 828](#)



- [Link Layer Discovery Protocol \(LLDP\) Configuration | 828](#)
- [IPFIX and sFlow Configuration | 828](#)
- [Sending pRouter Information to the SNMP Collector in Contrail | 831](#)
- [pRouter UVEs | 831](#)
- [Contrail User Interface for Underlay Overlay Analytics | 833](#)
- [Enabling Physical Topology on the Web UI | 834](#)
- [Viewing Topology to the Virtual Machine Level | 834](#)
- [Viewing the Traffic of any Link | 834](#)
- [Trace Flows | 835](#)
- [Search Flows and Map Flows | 836](#)
- [Overlay to Underlay Flow Map Schemas | 837](#)
- [Module Operations for Overlay Underlay Mapping | 840](#)
- [SNMP Collector Operation | 840](#)
- [Topology Module Operation | 842](#)
- [IPFIX and sFlow Collector Operation | 843](#)
- [Troubleshooting Underlay Overlay Mapping | 844](#)
- [Script to add pRouter Objects | 844](#)

## Overview: Underlay Overlay Mapping using Contrail Analytics

Today's cloud data centers consist of large collections of interconnected servers that provide computing and storage capacity to run a variety of applications. The servers are connected with redundant TOR switches, which in turn, are connected to spine routers. The cloud deployment is typically shared by multiple tenants, each of whom usually needs multiple isolated networks. Multiple isolated networks can be provided by overlay networks that are created by forming tunnels (for example, gre, ip-in-ip, mac-in-mac) over the underlay or physical connectivity.

As data flows in the overlay network, Contrail can provide statistics and visualization of the traffic in the underlay network.

## Underlay Overlay Analytics Available in Contrail

Starting with Contrail Release 2.20, you can view a variety of analytics related to underlay and overlay traffic in the Contrail Web user interface. The following are some of the analytics that Contrail provides for statistics and visualization of overlay underlay traffic.

- View the topology of the underlay network.

A user interface view of the physical underlay network with a drill down mechanism to show connected servers (contrail computes) and virtual machines on the servers.

- View the details of any element in the topology.

You can view details of a pRouter, vRouter, or virtual machine link between two elements. You can also view traffic statistics in a graphical view corresponding to the selected element.

- View the underlay path of an overlay flow.

Given an overlay flow, you can get the underlay path used for that flow and map the path in the topology view.

## Architecture and Data Collection

Accumulation of the data to map an overlay flow to its underlay path is performed in several steps across Contrail modules.

The following outlines the essential steps:

1. The SNMP collector module polls physical routers.

The SNMP collector module receives the authorizations and configurations of the physical routers from the Contrail config module, and polls all of the physical routers, using SNMP protocol. The collector uploads the data to the Contrail analytics collectors. The SNMP information is stored in the pRouter UVEs (physical router user visible entities).

2. IPFIX and sFlow protocols are used to collect the flow statistics.

The physical router is configured to send flow statistics to the collector, using one of the collection protocols: Internet Protocol Flow Information Export (IPFIX) or sFlow (an industry standard for sampled flow of packet export at Layer 2).

3. The topology module reads the SNMP information.

The Contrail topology module reads SNMP information from the pRouter UVEs from the analytics API, computes the neighbor list, and writes the neighbor information into the pRouter UVEs. This neighbor list is used by the Contrail WebUI to display the physical topology.

4. The Contrail user interface reads and displays the topology and statistics.

The Contrail user interface module reads the topology information from the Contrail analytics and displays the physical topology. It also uses information stored in the analytics to display graphs for link statistics, and to show the map of the overlay flows on the underlay network.

## New Processes/Services for Underlay Overlay Mapping

The `contrail-snmp-collector` and the `contrail-topology` are new daemons that are both added to the `contrail-analytics` node. The `contrail-analytics` package contains these new features and their associated files. The `contrail-status` displays the new services.

**Example: `contrail-status`**

The following is an example of using `contrail-status` to show the status of the new process and service for underlay overlay mapping.

```
user@host:~# contrail-status

== Contrail Control ==

supervisor-control:      active

contrail-control         active

...

== Contrail Analytics ==

supervisor-analytics:    active

...

contrail-query-engine    active

contrail-snmp-collector  active

contrail-topology        active
```

### Example: Service Command

The service command can be used to start, stop, and restart the new services. See the following example.

```
user@host:~# service contrail-snmp-collector status

contrail-snmp-collector    RUNNING pid 12179, uptime 1 day, 14:59:11
```

## External Interfaces Configuration for Underlay Overlay Mapping

This section outlines the external interface configurations necessary for successful underlay overlay mapping for Contrail analytics.

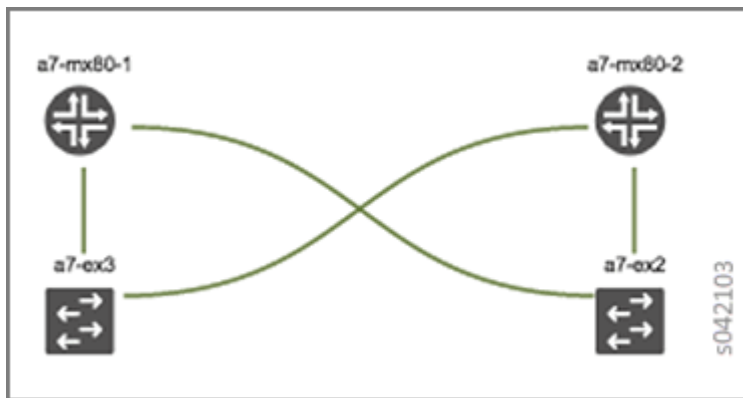
### Physical Topology

The typical physical topology includes:

- Servers connected to the ToR switches.
- ToR switches connected to spine switches.
- Spine switches connected to core switches.

The following is an example of how the topology is depicted in the Contrail WebUI analytics.

**Figure 209: Analytics Topology**



## SNMP Configuration

Configure SNMP on the physical devices so that the `contrail-snmp-collector` can read SNMP data.

The following shows an example SNMP configuration from a Juniper Networks device.

```
set snmp community public authorization read-only
```

## Link Layer Discovery Protocol (LLDP) Configuration

Configure LLDP on the physical device so that the `contrail-snmp-collector` can read the neighbor information of the routers.

The following is an example of LLDP configuration on a Juniper Networks device.

```
set protocols lldp interface all
```

```
set protocols lldp-med interface all
```

## IPFIX and sFlow Configuration

Flow samples are sent to the `contrail-collector` by the physical devices. Because the `contrail-collector` supports the sFlow and IPFIX protocols for receiving flow samples, the physical devices, such as MX Series devices or ToR switches, must be configured to send samples using one of those protocols.

### Example: sFlow Configuration

The following shows a sample sFlow configuration. In the sample, the IP variable *<source ip>* refers to the loopback or IP that can be reachable of the device that acts as an sflow source, and the other IP variable *<collector\_IP\_data>* is the address of the collector device.

```

root@host> show configuration protocols sflow | display set

set protocols sflow polling-interval 0

set protocols sflow sample-rate ingress 10

set protocols sflow source-ip <source ip>4

set protocols sflow collector <collector_IP_data> udp-port 6343

set protocols sflow interfaces ge-0/0/0.0

set protocols sflow interfaces ge-0/0/1.0

set protocols sflow interfaces ge-0/0/2.0

set protocols sflow interfaces ge-0/0/3.0

set protocols sflow interfaces ge-0/0/4.0

```

### Example: IPFIX Configuration

The following is a sample IPFIX configuration from a Juniper Networks device. The IP address variable *<ip\_sflow\_collector>* represents the sflow collector (control-collector analytics node) and *<source ip>* represents the source (outgoing) interface on the router/switch device used for sending flow data to the collector. This could also be the lo0 address, if it is reachable from the Contrail cluster.

```

root@host> show configuration chassis | display set

set chassis tfeb slot 0 sampling-instance sample-ins1

set chassis network-services

root@host> show configuration chassis tfeb | display set

```

```
set chassis tfeb slot 0 sampling-instance sample-ins1
```

```
root@host > show configuration services flow-monitoring | display set
```

```
set services flow-monitoring version-ipfix template t1 flow-active-timeout 30
```

```
set services flow-monitoring version-ipfix template t1 flow-inactive-timeout 30
```

```
set services flow-monitoring version-ipfix template t1 template-refresh-rate packets 10
```

```
set services flow-monitoring version-ipfix template t1 ipv4-template
```

```
root@host > show configuration interfaces | display set | match sampling
```

```
set interfaces ge-1/0/0 unit 0 family inet sampling input
```

```
set interfaces ge-1/0/1 unit 0 family inet sampling input
```

```
root@host> show configuration forwarding-options sampling | display set
```

```
set forwarding-options sampling instance sample-ins1 input rate 1
```

```
set forwarding-options sampling instance sample-ins1 family inet output flow-server <ip_sflow collector> port 4739
```

```
set forwarding-options sampling instance sample-ins1 family inet output flow-server <ip_sflow collector> version-ipfix template t1
```

```
set forwarding-options sampling instance sample-ins1 family inet output inline-jflow source-address <source ip>
```

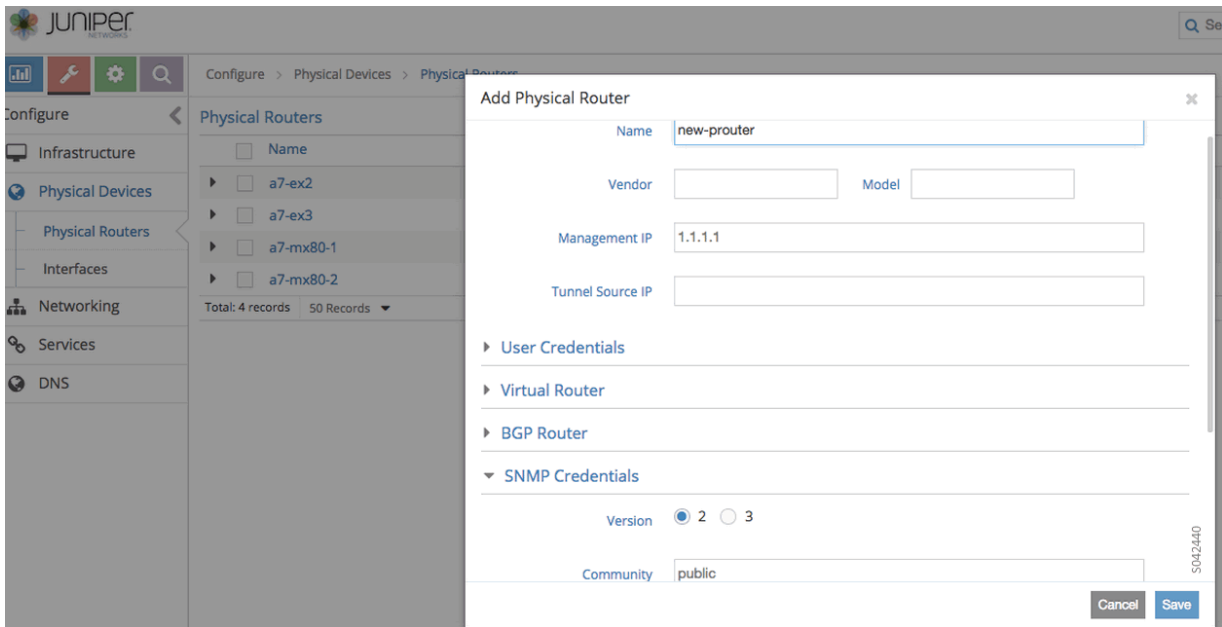
## Sending pRouter Information to the SNMP Collector in Contrail

Information about the physical devices must be sent to the SNMP collector before the full analytics information can be read and displayed. Typically, the pRouter information is taken from the `contrail-config` file.

*SNMP collector getting pRouter information from contrail-config file*

The physical routers are added to the `contrail-config` by using the Contrail user interface or by using direct API, by means of provisioning or other scripts. Once the configuration is in the `contrail-config`, the `contrail-snmp-collector` gets the physical router information from `contrail-config`. The SNMP collector uses this list and the other configuration parameters to perform SNMP queries and to populate pRouter UVEs.

**Figure 210: Add Physical Router Window**



## pRouter UVEs

pRouter UVEs are accessed from the REST APIs on your system from `contrail-analytics-api`, using a URL of the form:

`http://<host ip>:8081/analytics/uves/prouters`

The following is sample output from a pRouter REST API:

Figure 211: Sample Output From a pRouter REST API

```
[
- {
  href: "http://10.84.63.130:8081/analytics/uves/prouter/a7-mx80-1?flat",
  name: "a7-mx80-1"
},
- {
  href: "http://10.84.63.130:8081/analytics/uves/prouter/a7-mx80-2?flat",
  name: "a7-mx80-2"
},
- {
  href: "http://10.84.63.130:8081/analytics/uves/prouter/a7-ex3?flat",
  name: "a7-ex3"
},
- {
  href: "http://10.84.63.130:8081/analytics/uves/prouter/a7-ex2?flat",
  name: "a7-ex2"
}
]
5042104
```

Details of a pRouter UVE can be obtained from your system, using a URL of the following form:

`http://<host ip>:8081/analytics/uves/prouter/a7-ex3?flat`

The following is sample output of a pRouter UVE.



Figure 212: Sample Output From a pRouter UVE

```

{
- PRouterFlowEntry: {
  flow_export_source_ip: "10.84.63.114"
},
- PRouterLinkEntry: {
  - link_table: [
    - {
      remote_interface_name: "ge-1/0/1",
      local_interface_name: "ge-0/0/0.0",
      remote_interface_index: 517,
      local_interface_index: 503,
      type: 1,
      remote_system_name: "a7-mx80-1"
    },
    - {
      remote_interface_name: "ge-1/0/1",
      local_interface_name: "ge-0/0/1.0",
      remote_interface_index: 517,
      local_interface_index: 505,
      type: 1,
      remote_system_name: "a7-mx80-2"
    },
    - {
      remote_interface_name: "eth1",
      local_interface_name: "ge-0/0/2.0",
      remote_interface_index: 1,
      local_interface_index: 507,
      type: 2,
      remote_system_name: "a7s35"
    },
    - {
      remote_interface_name: "eth1",
      local_interface_name: "ge-0/0/3.0",
      remote_interface_index: 1,
      local_interface_index: 509,
      type: 2,
      remote_system_name: "a7s36"
    }
  ]
},
- PRouterEntry: {
  + ipMib: [...],
  + ifTable: [...],
  + ifXTable: [...],
  + arpTable: [...],
  + lldpTable: {...},
  + ifStats: {...}
}
}

```

5042435

## Contrail User Interface for Underlay Overlay Analytics

The topology view and related functionality is accessed from the Contrail Web user interface, **Monitor > Physical Topology**.

## Enabling Physical Topology on the Web UI

To enable the **Physical Topology** section in the Contrail Web UI:

1. Add the following lines to the `/etc/contrail/config.global.js` file of all the `contrail-webui` nodes:

```
config.optFeatureList = {};
config.optFeatureList.mon_infra_underlay = true;
```

2. Restart webui supervisor.

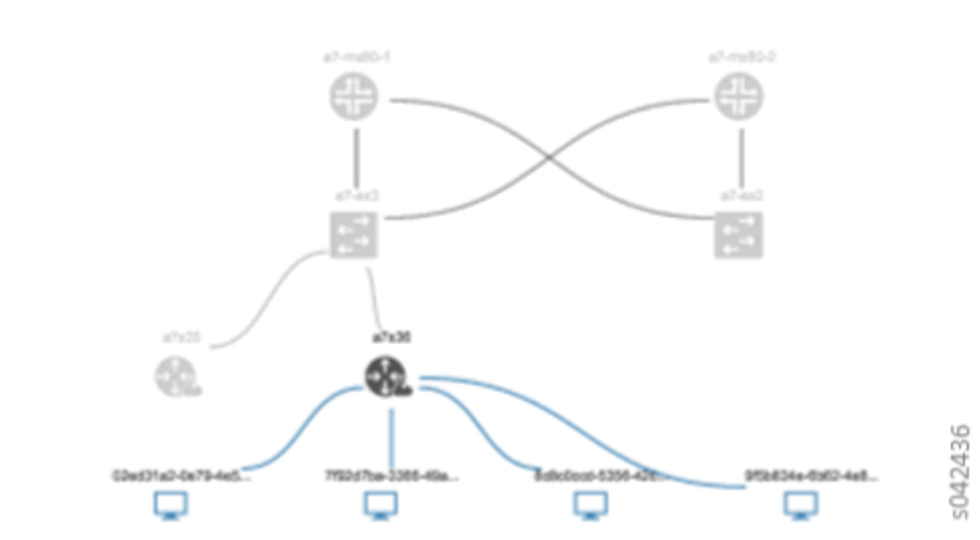
```
service supervisor-webui restart
```

The **Physical Topology** section is now available on the Contrail Web UI.

## Viewing Topology to the Virtual Machine Level

In the Contrail user interface, it is possible to drill down through displayed topology to the virtual machine level. The following diagram shows the virtual machines instantiated on a7s36 vRouter and the full physical topology related to each.

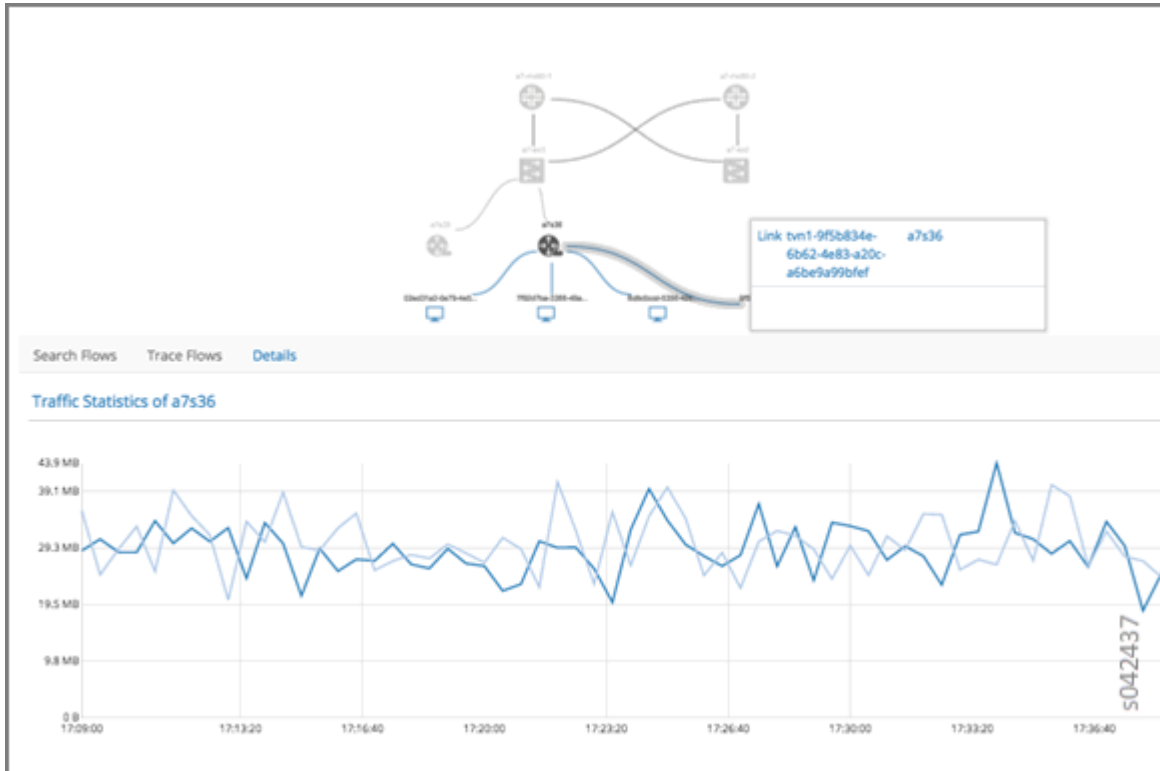
Figure 213: Physical Topology Related to a vRouter



## Viewing the Traffic of any Link

At **Monitor > Physical Topology**, double click any link on the topology to display the traffic statistics graph for that link. The following is an example.

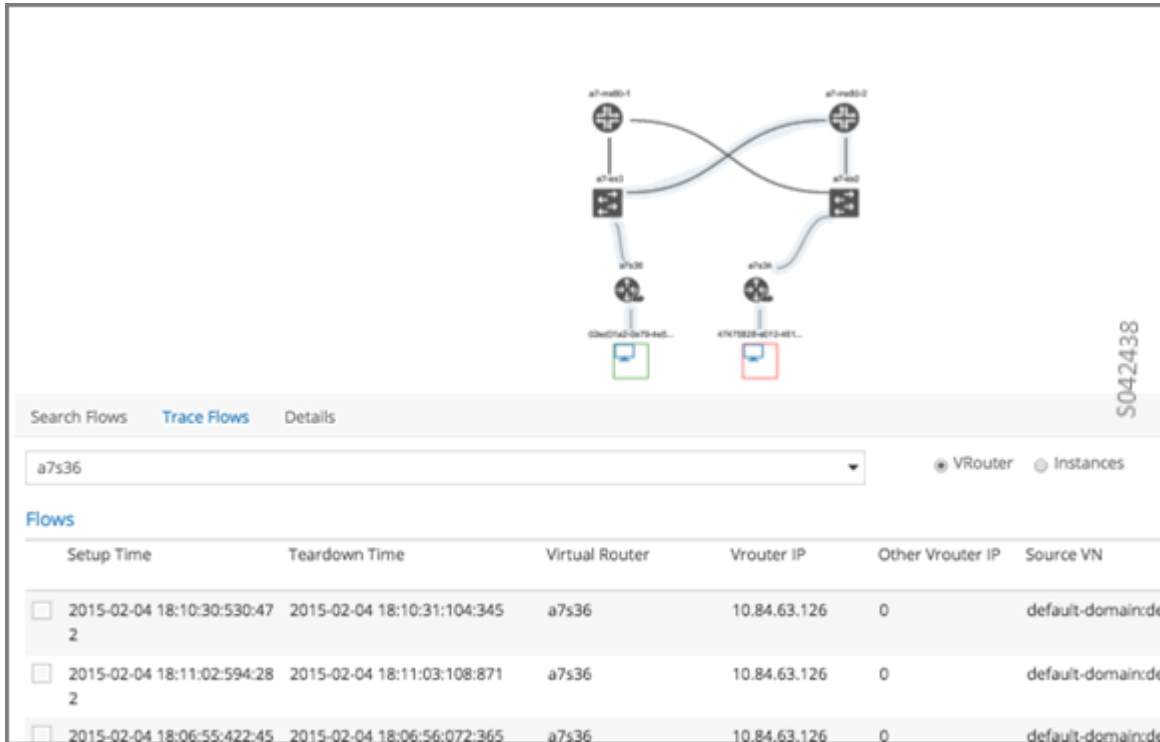
Figure 214: Traffic Statistics Graph



### Trace Flows

Click the **Trace Flows** tab to see a list of active flows. To see the path of a flow, click a flow in the active flows list, then click the **Trace Flow** button. The path taken in the underlay by the selected flow displays. The following is an example.

Figure 215: List of Active Flows



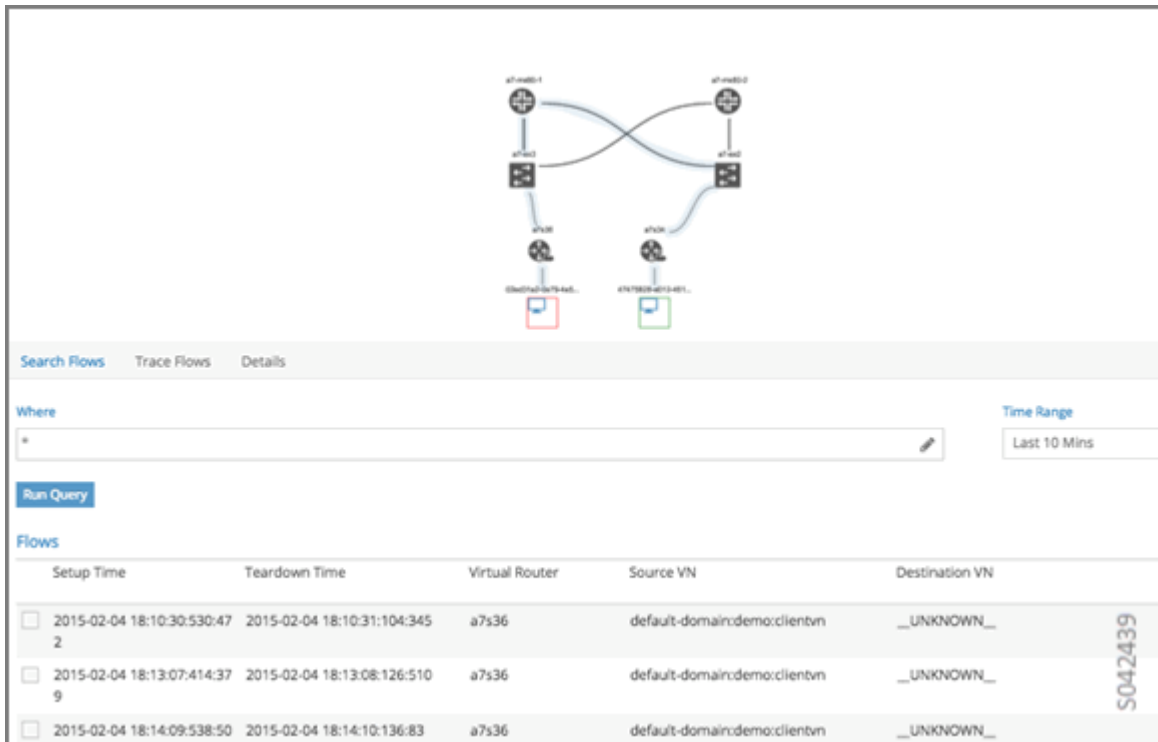
*Limitations of Trace Flow Feature*

Because the Trace Flow feature uses ip traceroute to determine the path between the two vRouters involved in the flow, it has the same limitations as the ip traceroute, including that Layer 2 routers in the path are not listed, and therefore do not appear in the topology.

**Search Flows and Map Flows**

Click the **Search Flows** tab to open a search dialog, then click the **Search** button to list the flows that match the search criteria. You can select a flow from the list and click **Map Flow** to display the underlay path taken by the selected flow in the topology. The following is an example.

Figure 216: Underlay Path



## Overlay to Underlay Flow Map Schemas

The schema to query the underlay mapping information for an overlay flow is obtained from a REST API, which can be accessed on your system using a URL of the following form:

<http://<host ip>:8081/analytics/table/OverlayToUnderlayFlowMap/schema>

### Example: Overlay to Underlay Flow Map Schema

```
{
  "type": "FLOW",
  "columns": [
    {
      "datatype": "string",
      "index": true,
      "name": "o_svn",
      "select": false,
      "suffixes": ["o_sip"]
    },
    {
      "datatype": "string",
      "index": false,
      "name": "o_sip",
      "select": false,
      "suffixes": null
    },
    {
      "datatype": "string",
      "index": true,
      "name": "o_dvn",
      "select": false,
      "suffixes": ["o_dip"]
    },
    {
      "datatype": "string",
      "index": false,
      "name": "o_dip",
      "select": false,
      "suffixes": null
    }
  ]
}
```

```

{"datatype": "int", "index": false, "name": "o_sport", "select": false, "suffixes": null},

{"datatype": "int", "index": false, "name": "o_dport", "select": false, "suffixes": null},

{"datatype": "int", "index": true, "name": "o_protocol", "select": false, "suffixes":
["o_sport", "o_dport"]},

{"datatype": "string", "index": true, "name": "o_vrouter", "select": false, "suffixes": null},

{"datatype": "string", "index": false, "name": "u_prouter", "select": null, "suffixes": null},

{"datatype": "int", "index": false, "name": "u_pifindex", "select": null, "suffixes": null},

{"datatype": "int", "index": false, "name": "u_vlan", "select": null, "suffixes": null},

{"datatype": "string", "index": false, "name": "u_sip", "select": null, "suffixes": null},

{"datatype": "string", "index": false, "name": "u_dip", "select": null, "suffixes": null},

{"datatype": "int", "index": false, "name": "u_sport", "select": null, "suffixes": null},

{"datatype": "int", "index": false, "name": "u_dport", "select": null, "suffixes": null},

{"datatype": "int", "index": false, "name": "u_protocol", "select": null, "suffixes": null},

{"datatype": "string", "index": false, "name": "u_flowtype", "select": null, "suffixes": null},

{"datatype": "string", "index": false, "name": "u_otherinfo", "select": null, "suffixes": null}}

```

The schema for underlay data across pRouters is defined in the Contrail installation at:

<http://<host ip>:8081/analytics/table/StatTable.UFlowData.flow/schema>

### Example: Flow Data Schema for Underlay

```

{"type": "STAT",

"columns": [

{"datatype": "string", "index": true, "name": "Source", "suffixes": null},

{"datatype": "int", "index": false, "name": "T", "suffixes": null},

```

```
{"datatype": "int", "index": false, "name": "CLASS(T)", "suffixes": null},
{"datatype": "int", "index": false, "name": "T=", "suffixes": null},
{"datatype": "int", "index": false, "name": "CLASS(T=)", "suffixes": null},
{"datatype": "uuid", "index": false, "name": "UUID", "suffixes": null},
{"datatype": "int", "index": false, "name": "COUNT(flow)", "suffixes": null},
{"datatype": "string", "index": true, "name": "name", "suffixes": ["flow.pifindex"]},
{"datatype": "int", "index": false, "name": "flow.pifindex", "suffixes": null},
{"datatype": "int", "index": false, "name": "SUM(flow.pifindex)", "suffixes": null},
{"datatype": "int", "index": false, "name": "CLASS(flow.pifindex)", "suffixes": null},
{"datatype": "int", "index": false, "name": "flow.sport", "suffixes": null},
{"datatype": "int", "index": false, "name": "SUM(flow.sport)", "suffixes": null},
{"datatype": "int", "index": false, "name": "CLASS(flow.sport)", "suffixes": null},
{"datatype": "int", "index": false, "name": "flow.dport", "suffixes": null},
{"datatype": "int", "index": false, "name": "SUM(flow.dport)", "suffixes": null},
{"datatype": "int", "index": false, "name": "CLASS(flow.dport)", "suffixes": null},
{"datatype": "int", "index": true, "name": "flow.protocol", "suffixes": ["flow.sport",
"flow.dport"]},
{"datatype": "int", "index": false, "name": "SUM(flow.protocol)", "suffixes": null},
{"datatype": "int", "index": false, "name": "CLASS(flow.protocol)", "suffixes": null},
{"datatype": "string", "index": true, "name": "flow.sip", "suffixes": null},
{"datatype": "string", "index": true, "name": "flow.dip", "suffixes": null},
{"datatype": "string", "index": true, "name": "flow.vlan", "suffixes": null},
```

```

{"datatype": "string", "index": false, "name": "flow.flowtype", "suffixes": null},
{"datatype": "string", "index": false, "name": "flow.otherinfo", "suffixes": null}]}}

```

### Example: Typical Query for Flow Map

The following is a typical query. Internally, the analytics-api performs a query into the FlowRecordTable, then into the StatTable.UFlowData.flow, to return list of (prouter, pifindex) pairs that give the underlay path taken for the given overlay flow.

```

FROM

OverlayToUnderlayFlowMap

SELECT

prouter, pifindex

WHERE

o_svn, o_sip, o_dvn, o_dip, o_sport, o_dport, o_protocol = <overlay flow>

```

## Module Operations for Overlay Underlay Mapping

### SNMP Collector Operation

The Contrail SNMP collector uses a Net-SNMP library to talk to a physical router or any SNMP agent. Upon receiving SNMP packets, the data is translated to the Python dictionary, and corresponding UVE objects are created. The UVE objects are then posted to the SNMP collector.

The SNMP module sleeps for some configurable period, then forks a collector process and waits for the process to complete. The collector process goes through a list of devices to be queried. For each device, it forks a greenlet task (Python coroutine), accumulates SNMP data, writes the summary to a JSON file, and exits. The parent process then reads the JSON file, creates UVEs, sends the UVEs to the collector, then goes to sleep again.

The pRouter UVE sent by the SNMP collector carries only the raw MIB information.

### Example: pRouter Entry Carried in pRouter UVE

The definition below shows the pRouterEntry carried in the pRouterUVE. Additionally, an example L1dpTable definition is shown.



The following create a virtual table as defined by:

```
http://<host ip>:8081/analytics/table/StatTable.UFlowData.flow/schema
```

```
struct LldpTable {  
  
    1: LldpLocalSystemData lldpLocalSystemData  
  
    2: optional list<LldpRemoteSystemsData> lldpRemoteSystemsData  
  
}  
  
struct PRouterEntry {  
  
    1: string name (key="ObjectPRouter")  
  
    2: optional bool deleted  
  
    3: optional LldpTable lldpTable  
  
    4: optional list<ArpTable> arpTable  
  
    5: optional list<IfTable> ifTable  
  
    6: optional list<IfXTable> ifXTable  
  
    7: optional list<IfStats> ifStats (tags="name:.ifIndex")  
  
    8: optional list<IpMib> ipMib  
  
}  
  
uve sandesh PRouterUVE {  
  
    1: PRouterEntry data  
  
}
```

## Topology Module Operation

The topology module reads UVEs posted by the SNMP collector and computes the neighbor table, populating the table with remote system name, local and remote interface names, the remote type (pRouter or vRouter) and local and remote ifindices. The topology module sleeps for a while, reads UVEs, then computes the neighbor table and posts the UVE to the collector.

The pRouter UVE sent by the topology module carries the neighbor list, so the clients can put together all of the pRouter neighbor lists to compute the full topology.

The corresponding pRouter UVE definition is the following.

```

struct LinkEntry {

    1: string remote_system_name

    2: string local_interface_name

    3: string remote_interface_name

    4: RemoteType type

    5: i32 local_interface_index

    6: i32 remote_interface_index

}

struct PRouterLinkEntry {

    1: string name (key="ObjectPRouter")

    2: optional bool deleted

    3: optional list<LinkEntry> link_table

}

uve sandesh PRouterLinkUVE {

    1: PRouterLinkEntry data

}

```

## IPFIX and sFlow Collector Operation

An IPFIX and sFlow collector has been implemented in the Contrail collector. The collector receives the IPFIX and sFlow samples and stores them as statistics samples in the analytics database.

### Example: IPFIX sFlow Collector Data

The following definition shows the data stored for the statistics samples and the indices that can be used to perform queries.

```
struct UFlowSample {  
  
    1: u64 pifindex  
  
    2: string sip  
  
    3: string dip  
  
    4: u16 sport  
  
    5: u16 dport  
  
    6: u16 protocol  
  
    7: u16 vlan  
  
    8: string flowtype  
  
    9: string otherinfo  
  
}  
  
struct UFlowData {  
  
    1: string name (key="ObjectPRouterIP")  
  
    2: optional bool deleted  
  
    3: optional list<UFlowSample> flow
```

```
(tags="name:.pifindex, .sip, .dip, .protocol:.sport, .protocol:.dport, .vlan")

}
```

## Troubleshooting Underlay Overlay Mapping

This section provides a variety of links where you can research errors that may occur with underlay overlay mapping.

### System Logs

Logs for `contrail-snmp-collector` and `contrail-topology` are in the following locations on an installed Contrail system:

```
/var/log/contrail/contrail-snmp-collector-stdout.log
```

```
/var/log/contrail/contrail-topology.log
```

### Introspect Utility

Use URLs of the following forms on your Contrail system to access the introspect utilities for SNMP data and for topology data.

- SNMP data introspect

```
http://<host ip>:5920/Snh_SandeshUVECacheReq?x=PRouterEntry
```

- Topology data introspect

```
http://<host ip>:5921/Snh_SandeshUVECacheReq?x=PRouterLinkEntry
```

## Script to add pRouter Objects

The usual mechanism for adding pRouter objects to `contrail-config` is through Contrail UI. But you also have the ability to add these objects using the Contrail `vnc-api`. To add one pRouter, save the file with the name `cfg-snmp.py`, and then execute the command as shown:

```
python cfg-snmp.py
```

**Example: Content for cfg-snmp.py**

```
#!/python

from vnc_api import vnc_api

from vnc_api.gen.resource_xsd import SNMPCredentials

vnc = vnc_api.VncApi('admin', 'abcde123', 'admin')

apr = vnc_api.gen.resource_client.PhysicalRouter(name='a7-mx80-1')

apr.set_physical_router_management_ip('ip_address')

apr.set_physical_router_dataplane_ip('ip_address')

apr.set_physical_router_snmp_credentials(SNMPCredentials(version=2, v2_community='public'))

vnc.physical_router_create(apr)

#$ABC123

apr = vnc_api.gen.resource_client.PhysicalRouter(name='a7-mx80-2')

apr.set_physical_router_management_ip('ip_address')

apr.set_physical_router_dataplane_ip('ip_address')

apr.set_physical_router_snmp_credentials(SNMPCredentials(version=2, v2_community='public'))

vnc.physical_router_create(apr)

#$ABC123'

apr = vnc_api.gen.resource_client.PhysicalRouter(name='a7-ex3')

apr.set_physical_router_management_ip('source_ip')

apr.set_physical_router_dataplane_ip('source_ip')
```

```
apr.set_physical_router_snmp_credentials(SNMPCredentials(version=2, v2_community='public'))

vnc.physical_router_create(apr)

#$ABC123'

apr = vnc_api.gen.resource_client.PhysicalRouter(name='a7-ex2')

apr.set_physical_router_management_ip('ip_address')

apr.set_physical_router_dataplane_ip('ip_address')

apr.set_physical_router_snmp_credentials(SNMPCredentials(version=2, v2_community='public'))

vnc.physical_router_create(apr)

#$ABC123'
```

## RELATED DOCUMENTATION

*Understanding Contrail Analytics*

*Contrail Alerts*

# Configuring Contrail Analytics

## IN THIS CHAPTER

- [Analytics Scalability | 847](#)
- [High Availability for Analytics | 849](#)
- [Role-Based Access Control for Analytics | 849](#)
- [System Log Receiver in Contrail Analytics | 851](#)
- [Sending Flow Messages to the Contrail System Log | 852](#)
- [More Efficient Flow Queries | 853](#)
- [Ceilometer Support in a Contrail Cloud | 853](#)
- [User Configuration for Analytics Alarms and Log Statistics | 860](#)
- [Alarms History | 870](#)
- [Node Memory and CPU Information | 872](#)
- [Role- and Resource-Based Access Control for the Contrail Analytics API | 873](#)
- [Configuring Analytics as a Standalone Solution | 874](#)
- [Configuring Secure Sandesh and Introspect for Contrail Analytics | 877](#)

## Analytics Scalability

The Contrail monitoring and analytics services (*collector* role) collect and store data generated by various system components and provide the data to the Contrail interface by means of representational state transfer (REST) application program interface (API) queries.

The Contrail components are horizontally scalable to ensure consistent performance as the system grows. Scalability is provided for the generator components (*control* and *compute* roles) and for the REST API users (*webui* role).

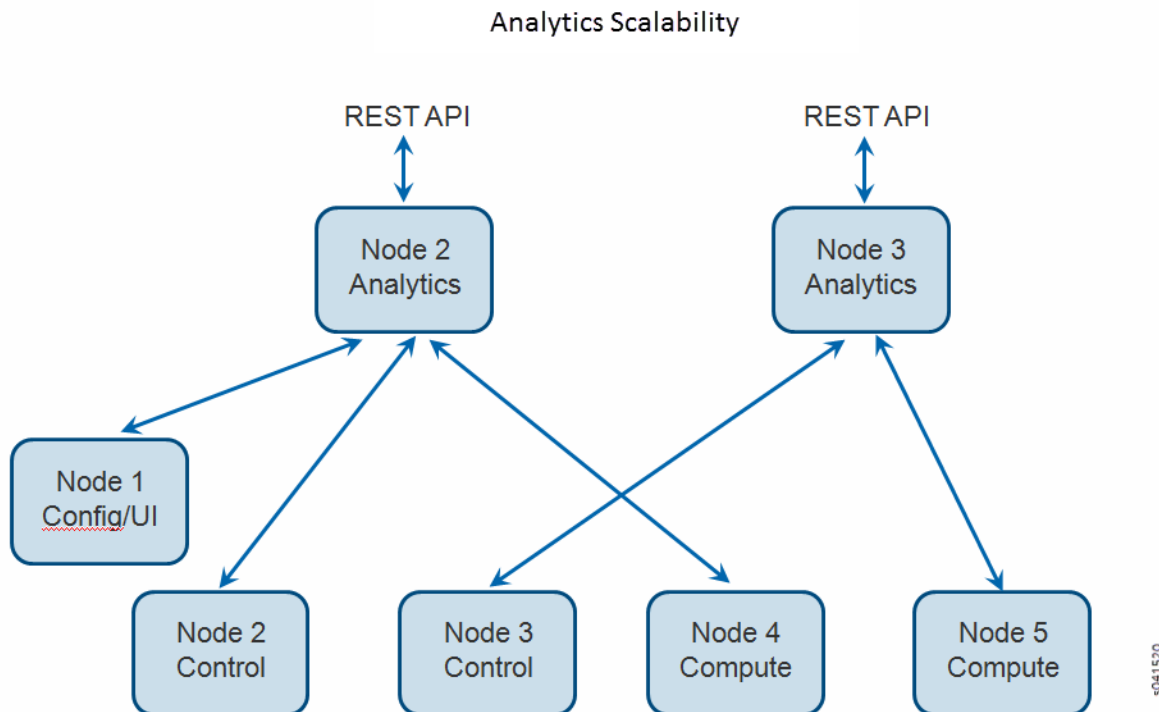
This section provides a brief description of the recommended configuration of analytics in Contrail to achieve horizontal scalability.

The following is the recommended locations for the various component roles of the Contrail system for a 5-node configuration.

- Node 1 –config role, web-ui role
- Node 2 –control role, analytics role, database role
- Node 3 –control role, analytics role, database role
- Node 4 –compute role
- Node 5 –compute role

Figure 217 on page 848 illustrates scalable connections for analytics in a 5-node system, with the nodes configured for roles as recommended above. The analytics load is distributed between the two analytics nodes. This configuration can be extended to any number of analytics nodes.

**Figure 217: Analytics Scalability**



The analytics nodes collect and store data and provide this data through various REST API queries. Scalability is provided for the control nodes, the compute nodes, and the REST API users, with the API output displayed in the Contrail user interface. As the number of control and compute nodes increase in the system, the analytics nodes can also be increased.



## High Availability for Analytics

Contrail supports multiple instances of analytics for high availability and load balancing.

Contrail analytics provides two broad areas of functionality:

- **contrail-collector** –Receives status, logs, and flow information from all Contrail processing elements (for example, generators) and records them.

Every generator is connected to one of the **contrail-collector** instances at any given time. If an instance fails (or is shut down), all the generators that are connected to it are automatically moved to another functioning instance, typically in a few seconds or less. Some messages may be lost during this movement. UVEs are resilient to message loss, so the state shown in a UVE is kept consistent to the state in the generator.

- **contrail-opserver** –Provides an external API to report UVEs and to query logs and flows.

Each analytics component exposes a northbound REST API represented by the **contrail-opserver** service (port 8081) so that the failure of one analytics component or one **contrail-opserver** service should not impact the operation of other instances.

These are the ways to manage connectivity to the **contrail-opserver** endpoints:

- Periodically poll the **contrail-opserver** service on a set of analytics nodes to determine the list of functioning endpoints, then make API requests from one or more of the functioning endpoints.
- The Contrail user interface makes use of the same northbound REST API to present dashboards, and reacts to any **contrail-opserver** high availability event automatically.

## Role-Based Access Control for Analytics

The analytics API uses role-based access control (RBAC) to provide the ability to access UVE and query information based on the permissions of the user for the UVE or queried object.

Contrail Release 4.1 extends authenticated access so that tenants can view network monitoring information about the networks for which they have read permissions. RBAC for analytics is a Beta feature in Contrail Release 4.1.

The analytics API can map query and UVE objects to configuration objects on which RBAC rules are applied, so that read permissions can be verified using the VNC API.

RBAC is applied to analytics in the following ways:

- For statistics queries, annotations are added to the Sandesh file so that indices and tags on statistics queries can be associated with objects and UVEs. These are used by the contrail-analytics-api to determine the object level read permissions.
- For flow and log queries, the object read permissions are evaluated for each AND term in the where query.
- For UVEs list queries (e.g. analytics/uve/virtual-networks/), the contrail-analytics-api gets a list of UVEs that have read permissions for a given token. For a UVE query for a specific resource (e.g. analytics/uves/virtual-network/vn1), contrail-analytics-api checks the object level read permissions using VNC API.

Tenants cannot view system logs and flow logs, those logs are displayed for cloud-admin roles only.

A non-admin user can see only non-global UVEs, including:

- virtual\_network
- virtual\_machine
- virtual\_machine\_interface
- service\_instance
- service\_chain
- tag
- firewall\_policy
- firewall\_rule
- address\_group
- service\_group
- application\_policy\_set

In `/etc/contrail/contrail-analytics-api.conf`, in the section `DEFAULTS`, the parameter `aaa_mode` now supports `rbac` as one of the values.

## System Log Receiver in Contrail Analytics

### IN THIS SECTION

- [Overview | 851](#)
- [Redirecting System Logs to Contrail Collector | 851](#)
- [Exporting Logs from Contrail Analytics | 851](#)

### Overview

The `contrail-collector` process on the Contrail Analytics node can act as a system log receiver.

### Redirecting System Logs to Contrail Collector

You can enable the `contrail-collector` to receive system logs by giving a valid `syslog_port` as a command line option:

```
--DEFAULT.syslog_port <arg>
```

or by adding `syslog_port` in the `DEFAULT` section of the configuration file at `/etc/contrail/contrail-collector.conf`.

For nodes to send system logs to the `contrail-collector`, the system log configuration for the node should be set up to direct the system logs to `contrail-collector`.

### Example

Add the following line in `/etc/rsyslog.d/50-default.conf` on an Ubuntu system to redirect the system logs to `contrail-collector`.

```
*.* @<collector_ip>:<collector_syslog_port> :: @ for udp, @@ for tcp
```

The logs can be retrieved by using Contrail tool, either by using the `contrail-logs` utility on the analytics node or by using the Contrail user interface on the system log query page.

### Exporting Logs from Contrail Analytics

You can also export logs stored in Contrail analytics to another system log receiver by using the `contrail-logs` utility.

The `contrail-logs` utility can take these options: `--send-syslog`, `--syslog-server`, `--syslog-port`, to query Contrail analytics, then send the results as system logs to a system log server. This is an on-demand command, one can write a cron job or a job that continuously invokes `contrail-logs` to achieve continuous sending of logs to another system log server.

## Sending Flow Messages to the Contrail System Log

The `contrail-vrouter-agent` can be configured to send flow messages and other messages to the system log (syslog). To send flow messages to syslog, configure the following parameters in `/etc/contrail/contrail-vrouter-agent.conf`.

The following parameters are under the section `DEFAULT`:

- `log_flow=1`—Enables logging of all flow messages.
- `use_syslog=1`—Enables sending of all messages, including flow messages, to syslog.
- `syslog_facility=LOG_LOCAL0`—Enables sending messages from the `contrail-vrouter-agent` to the syslog, using the facility `LOCAL0`. You can configure `LOCAL0` to your required facility.
- `log_level=SYS_INFO`—Changes the logging level of `contrail-vrouter-agent` to `INFO`.

If syslog is enabled, flow messages are *not* sent to Contrail Analytics because the two destinations are mutually exclusive.

Flow log sampling settings apply regardless of the flow log destination specified. If sampling is enabled, the syslog messages will be sampled using the same rules that would apply to Contrail Analytics. If non-sampled flow data is required, sampling must be disabled by means of configuration settings.

Flow events for termination will include both the appropriate tear-down fields and the appropriate setup fields.

The flow messages will be sent to the syslog with a severity of `INFO`.

The user can configure the remote system log (`rsyslog`) on the compute node to send syslog messages with facility `LOCAL0`, severity of `INFO` (and lower), to the remote syslog server. Messages with a higher severity than `INFO` can be logged to a local file to allow for debugging.

Flow messages appear in the syslog in a format similar to the following log example:

```
May 24 14:40:13 a7s10 contrail-vrouter-agent[29930]: 2016-05-24 Tue 14:40:13:921.098 PDT a7s10 [Thread
139724471654144, Pid 29930]: [SYS_INFO]: FlowLogDataObject: flowdata= [ [ [ flowuuid = 7ea8bf8f-b827-496e-
b93e-7622a0c8eaea direction_ing = 1 sourcevn = default-domain:mock-gen-test:vn8 sourceip = 1.0.0.9 destvn =
default-domain:mock-gen-test:vn58 destip = 1.0.0.59 protocol = 1 sport = -29520 dport = 20315 setup_time =
1464125225556930 bytes = 1035611592 packets = 2024830 diff_bytes = 27240 diff_packets = 40 ], ] ]
```

**NOTE:** Several individual flow messages might be packed into a single syslog message for improved efficiency.

## More Efficient Flow Queries

Flow queries are now analyzed on a 7-tuple basis, enabling more efficient flow queries by focusing on elements more important for analysis, and de-emphasizing lesser elements. More efficient queries enable load reduction and allow application of security policy.

An enhanced security framework is implemented to manage connectivity between workloads, or VMIs. Each VMI is tagged with the attributes of Deployment, App, Tier, and Site, and the user specifies security policies for VMIs using the values of these tags. Contrail can analyze the traffic flow between groups of VMI, where groups are categorized according to one or more values of the tags.

The existing FlowLogData is replaced by SessionEndpointData, which is a combination of the local VMI tags and VNs, the security policy and security rule, and route attributes for the remote endpoint. A SessionAggregate map and counts both enable traffic analysis within and across security policies by means of session sampling and session aggregate counts.

The flow export feature is disabled by default. Until the `session_export_rate` is set explicitly, flow queries will not return any results regardless of the traffic. To use this feature, set the session export rate in the Contrail WebUI at **Config->Global Config->Forwarding Options**.

## Ceilometer Support in a Contrail Cloud

### IN THIS SECTION

- [Overview | 854](#)
- [Ceilometer Details | 854](#)
- [Verification of Ceilometer Operation | 855](#)
- [Contrail Ceilometer Plugin | 857](#)
- [Ceilometer Installation and Provisioning | 860](#)

Ceilometer is an OpenStack feature that provides an infrastructure for collecting SDN metrics from OpenStack projects. The metrics can be used by various rating engines to transform events into billable items. The Ceilometer collection process is sometimes referred to as “metering”. The Ceilometer service provides data that can be used by platforms that provide metering, tracking, billing, and similar services. This topic describes how to configure the Ceilometer service for Contrail.

## Overview

Contrail Release 2.20 and later supports the OpenStack Ceilometer service, on the OpenStack Juno release on Ubuntu 14.04.1 LTS.

The prerequisites for installing Ceilometer are:

- Contrail Cloud installation
- Provisioned using `enable_ceilometer = True` in the `provisioning` file.

**NOTE:** Ceilometer services are only installed on the first OpenStack controller node and do not support high availability in Contrail Release 2.20.

## Ceilometer Details

Ceilometer is used to reliably collect measurements of the utilization of the physical and virtual resources comprising deployed clouds, persist these data for subsequent retrieval and analysis, and trigger actions when defined criteria are met.

The Ceilometer architecture consists of:

|                           |   |
|---------------------------|---|
| <b>Polling agent</b>      | Agent designed to poll OpenStack services and build meters. The polling agents are also run on the compute nodes in addition to the OpenStack controller. |
| <b>Notification agent</b> | Agent designed to listen to notifications on message queue and convert them to events and samples.  |
| <b>Collector</b>          | Gathers and records event and metering data created by the notification and polling agents.   |
| <b>API server</b>         | Provides a REST API to query and view data recorded by the collector service.   |
| <b>Alarms</b>             | Daemons to evaluate and notify based on defined alarming rules.   |
| <b>Database</b>           | Stores the metering data, notifications, and alarms. The supported databases are MongoDB, SQL-based databases compatible with SQLAlchemy, and HBase. The  |

recommended database is MongoDB, which has been thoroughly tested with Contrail and deployed on a production scale.

## Verification of Ceilometer Operation

The Ceilometer services are named slightly differently on the Ubuntu and RHEL Server 7.0.

On Ubuntu, the service names are:

|                           |  |
|---------------------------|--|
| <b>Polling agent</b>      | ceilometer-agent-central and ceilometer-agent-compute    |
| <b>Notification agent</b> | ceilometer-agent-notification                            |
| <b>Collector</b>          | ceilometer-collector                                     |
| <b>API Server</b>         | ceilometer-api   |
| <b>Alarms</b>             | ceilometer-alarm-evaluator and ceilometer-alarm-notifier |

On RHEL Server 7.0, the service names are:

|                           |  |
|---------------------------|--|
| <b>Polling agent</b>      | openstack-ceilometer-central and openstack-ceilometer-compute                |
| <b>Notification agent</b> | openstack-ceilometer-notification  |
| <b>Collector</b>          | openstack-ceilometer-collector   |
| <b>API server</b>         | openstack-ceilometer-api   |
| <b>Alarms</b>             | openstack-ceilometer-alarm-evaluator and openstack-ceilometer-alarm-notifier |

To verify the Ceilometer installation, users can verify that the Ceilometer services are up and running by using the `openstack-status` command.

For example, using the `openstack-status` command on an all-in-one node running Ubuntu 14.04.1 LTS with release 2.2 of Contrail installed shows the following Ceilometer services as active:

```
== Ceilometer services ==
ceilometer-api:         active
ceilometer-agent-central: active
ceilometer-agent-compute: active
ceilometer-collector:   active
ceilometer-alarm-notifier: active
```

```
ceilometer-alarm-evaluator: active
ceilometer-agent-notification:active
```

You can issue the `ceilometer meter-list` command on the OpenStack controller node to verify that meters are being collected, stored, and reported via the REST API. The following is an example of the output:

```
user@host:~# (source /etc/contrail/openstackrc; ceilometer meter-list)
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| Name                | Type      | Unit   | Resource ID                |
| User ID             | Project ID|         |                             |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| ip.floating.receive.bytes | cumulative | B      | a726f93a-65fa-4cad-828b-54dbfcf4a119 |
| None                | None      |         |                             |
| ip.floating.receive.packets | cumulative | packet | a726f93a-65fa-4cad-828b-54dbfcf4a119 |
| None                | None      |         |                             |
| ip.floating.transmit.bytes | cumulative | B      | a726f93a-65fa-4cad-828b-54dbfcf4a119 |
| None                | None      |         |                             |
| ip.floating.transmit.packets | cumulative | packet | a726f93a-65fa-4cad-828b-54dbfcf4a119 |
| None                | None      |         |                             |
| network              | gauge     | network | 7fa6796b-756e-4320-9e73-87d4c52ecc83 |
| 15c0240142084d16b3127d6f844adb9 | ded208991de34fe4bb7dd725097f1c7e |
| network              | gauge     | network | 9408e287-d3e7-41e2-89f0-5c691c9ca450 |
| 15c0240142084d16b3127d6f844adb9 | ded208991de34fe4bb7dd725097f1c7e |
| network              | gauge     | network | b3b72b98-f61e-4e1f-9a9b-84f4f3ddec0b |
| 15c0240142084d16b3127d6f844adb9 | ded208991de34fe4bb7dd725097f1c7e |
| network              | gauge     | network | cb829abd-e6a3-42e9-a82f-0742db55d329 |
| 15c0240142084d16b3127d6f844adb9 | ded208991de34fe4bb7dd725097f1c7e |
| network.create       | delta     | network | 7fa6796b-756e-4320-9e73-87d4c52ecc83 |
| 15c0240142084d16b3127d6f844adb9 | ded208991de34fe4bb7dd725097f1c7e |
| network.create       | delta     | network | 9408e287-d3e7-41e2-89f0-5c691c9ca450 |
| 15c0240142084d16b3127d6f844adb9 | ded208991de34fe4bb7dd725097f1c7e |
| network.create       | delta     | network | b3b72b98-f61e-4e1f-9a9b-84f4f3ddec0b |
| 15c0240142084d16b3127d6f844adb9 | ded208991de34fe4bb7dd725097f1c7e |
| network.create       | delta     | network | cb829abd-e6a3-42e9-a82f-0742db55d329 |
| 15c0240142084d16b3127d6f844adb9 | ded208991de34fe4bb7dd725097f1c7e |
| port                 | gauge     | port   | 0d401d96-c2bf-4672-abf2-880eecf25ceb |
| 01edcedd989f43b3a2d6121d424b254d | 82ab961f88994e168217ddd746fdd826 |
| port                 | gauge     | port   | 211b94a4-581d-45d0-8710-c6c69df15709 |
| 01edcedd989f43b3a2d6121d424b254d | 82ab961f88994e168217ddd746fdd826 |
| port                 | gauge     | port   | 2287ce25-4eef-4212-b77f-3cf590943d36 |
```



```

01edcedd989f43b3a2d6121d424b254d | 82ab961f88994e168217ddd746fdd826 |
| port.create | delta | port | f62f3732-222e-4c40-8783-5bcbc1fd6a1c |
01edcedd989f43b3a2d6121d424b254d | 82ab961f88994e168217ddd746fdd826 |
| port.create | delta | port | f8c89218-3cad-48e2-8bd8-46c1bc33e752 |
01edcedd989f43b3a2d6121d424b254d | 82ab961f88994e168217ddd746fdd826 |
| port.update | delta | port | 43ed422d-b073-489f-877f-515a3cc0b8c4 |
15c0240142084d16b3127d6f844adb9 | ded208991de34fe4bb7dd725097f1c7e |
| subnet | gauge | subnet | 09105ed1-1654-4b5f-8c12-f0f2666fa304 |
15c0240142084d16b3127d6f844adb9 | ded208991de34fe4bb7dd725097f1c7e |
| subnet | gauge | subnet | 4bf00aac-407c-4266-a048-6ff52721ad82 |
15c0240142084d16b3127d6f844adb9 | ded208991de34fe4bb7dd725097f1c7e |
| subnet.create | delta | subnet | 09105ed1-1654-4b5f-8c12-f0f2666fa304 |
15c0240142084d16b3127d6f844adb9 | ded208991de34fe4bb7dd725097f1c7e |
| subnet.create | delta | subnet | 4bf00aac-407c-4266-a048-6ff52721ad82 |
15c0240142084d16b3127d6f844adb9 | ded208991de34fe4bb7dd725097f1c7e |
+-----+-----+-----+-----+
+-----+-----+-----+-----+

```

**NOTE:** The `ceilometer meter-list` command lists the meters only if images have been created, or instances have been launched, or if subnet, port, floating IP addresses have been created, otherwise the meter list is empty. You also need to source the `/etc/contrail/openstackrc` file when executing the command.

## Contrail Ceilometer Plugin

The Contrail Ceilometer plugin adds the capability to meter the traffic statistics of floating IP addresses in Ceilometer. The following meters for each floating IP resource are added by the plugin in Ceilometer.

```

ip.floating.receive.bytes
ip.floating.receive.packets
ip.floating.transmit.bytes
ip.floating.transmit.packets

```

The Contrail Ceilometer plugin configuration is done in the `/etc/ceilometer/pipeline.yaml` file when Contrail is installed by the Fabric provisioning scripts.

The following example shows the configuration that is added to the file:

```

sources:
  - name: contrail_source

```

```

interval: 600
meters:
  - "ip.floating.receive.packets"
  - "ip.floating.transmit.packets"
  - "ip.floating.receive.bytes"
  - "ip.floating.transmit.bytes"
resources:
  - contrail://<IP-address-of-Contrail-Analytics-Node>:8081
sinks:
  - contrail_sink
sinks:
  - name: contrail_sink
publishers:
  - rpc://
transformers:

```

The following example shows the Ceilometer meter list output for the floating IP meters:

```

+-----+-----+-----+
+-----+-----+-----+
+-----+-----+-----+
| Name                | Type    | Unit  | Resource
ID                    |         |       |         | User ID
| Project ID         |         |       |         |
+-----+-----+-----+
+-----+-----+-----+
+-----+-----+-----+
| ip.floating.receive.bytes | cumulative | B      | 451c93eb-
e728-4ba1-8665-6e7c7a8b49e2 |         |       |         | None
| None                |         |       |         |
| ip.floating.receive.bytes | cumulative | B      | 9cf76844-8f09-4518-a09e-
e2b8832bf894                |         |       |         |
None                        |         |       |         |
| ip.floating.receive.packets | cumulative | packet | 451c93eb-
e728-4ba1-8665-6e7c7a8b49e2 |         |       |         | None
| None                |         |       |         |
| ip.floating.receive.packets | cumulative | packet | 9cf76844-8f09-4518-a09e-
e2b8832bf894                |         |       |         |
None                        |         |       |         |
| ip.floating.transmit.bytes | cumulative | B      | 451c93eb-
e728-4ba1-8665-6e7c7a8b49e2 |         |       |         | None
| None                |         |       |         |

```

```

| ip.floating.transmit.bytes | cumulative | B          | 9cf76844-8f09-4518-a09e-
e2b8832bf894                | None      |           |
None                          |           |           |
| ip.floating.transmit.packets | cumulative | packet    | 451c93eb-
e728-4ba1-8665-6e7c7a8b49e2 | None      |           |
| None                          |           |           |
| ip.floating.transmit.packets | cumulative | packet    | 9cf76844-8f09-4518-a09e-
e2b8832bf894                | None      |           |
None                          |           |           |

```

In the meter `-list` output, the Resource ID refers to the floating IP.

The following example shows the output from the `ceilometer resource-show -r 451c93eb-e728-4ba1-8665-6e7c7a8b49e2` command:

```

+-----+-----+-----+-----+-----+-----+
| Property | Value |
+-----+-----+-----+-----+-----+-----+
| metadata | {'router_id': u'None', u'status': u'ACTIVE', u'tenant_id': |
|          | u'ceed483222f9453ab1d7bccd353971bc', u'floating_network_id': |
|          | u'6d0cca50-4be4-4b49-856a-6848133eb970', u'fixed_ip_address': |
|          | u'2.2.2.4', u'floating_ip_address': u'3.3.3.4', u'port_id': u'c6ce2abf- |
|          | ad98-4e56-ae65-ab7c62a67355', u'id': |
|          | u'451c93eb-e728-4ba1-8665-6e7c7a8b49e2', u'device_id': |
|          | u'00953f62-df11-4b05-97ca-30c3f6735ffd'} |
| project_id | None |
| resource_id | 451c93eb-e728-4ba1-8665-6e7c7a8b49e2 |
| source      | openstack |
| user_id     | None |
+-----+-----+-----+-----+-----+-----+

```

The following example shows the output from the `ceilometer statistics` command and the `ceilometer sample-list` command for the **ip.floating.receive.packets** meter:

```

+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+
| Period | Period Start          | Period End          | Count | Min | Max |
Sum     | Avg                  | Duration           | Duration Start    | Duration End      |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+

```

```

| 0 | 2015-02-13T19:50:40.795000 | 2015-02-13T19:50:40.795000 | 2892 | 0.0 | 325.0 |
1066.0 | 0.368603042877 | 439069.674 | 2015-02-13T19:50:40.795000 | 2015-02-18T21:48:30.469000 |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+
+-----+-----+-----+-----+-----+-----+
+-----+-----+
| Resource ID | Name | Type | Volume |
Unit | Timestamp |
+-----+-----+-----+-----+-----+-----+
+-----+-----+
| 9cf76844-8f09-4518-a09e-e2b8832bf894 | ip.floating.receive.packets | cumulative | 208.0 |
packet | 2015-02-18T21:48:30.469000 |
| 451c93eb-e728-4ba1-8665-6e7c7a8b49e2 | ip.floating.receive.packets | cumulative | 325.0 |
packet | 2015-02-18T21:48:28.354000 |
| 9cf76844-8f09-4518-a09e-e2b8832bf894 | ip.floating.receive.packets | cumulative | 0.0 |
packet | 2015-02-18T21:38:30.350000 |

```

## Ceilometer Installation and Provisioning

There are two scenarios possible for Contrail Ceilometer plugin installation.

1. If you install your own OpenStack distribution, you can install the Contrail Ceilometer plugin on the OpenStack controller node.
2. When using Contrail Cloud services, the Ceilometer controller services are installed and provisioned as part of the OpenStack controller node and the compute agent service is installed as part of the compute node when `enable_ceilometer` is set as `True` in the cluster `config` or `testbed` files.

## User Configuration for Analytics Alarms and Log Statistics

### IN THIS SECTION

- [Configuring Alarms Based on User-Visible Entities Data | 861](#)
- [Examples: Detecting Anomalies | 863](#)
- [Configuring the User-Defined Log Statistic | 864](#)
- [Implementing the User-Defined Log Statistic | 867](#)

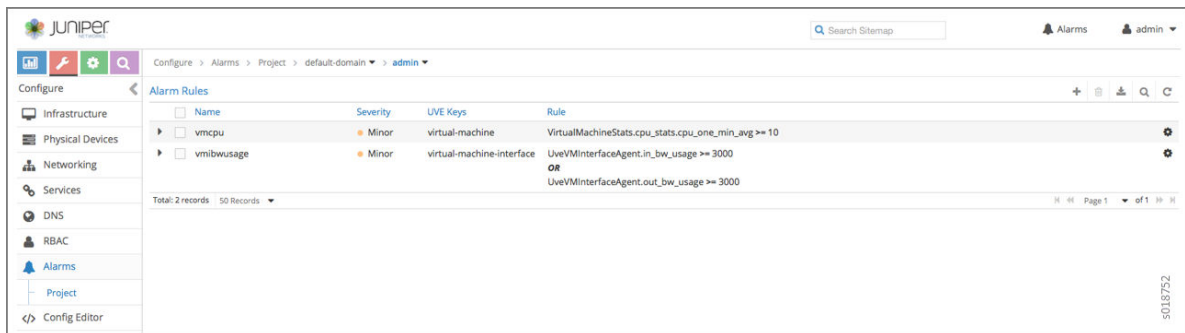
## Configuring Alarms Based on User-Visible Entities Data

Starting with Contrail 3.1, users can dynamically configure alarms based on the user-visible entities (UVE) data. An alarm configuration object is created based on the alarm configuration XSD schema. The alarm configuration object is added to the Contrail configuration database, using the Contrail API server REST API interface.

An alarm configuration object can be anchored in the configuration data model under `global-system-config` or `project`, depending on the alarm type. Under `global-system-config`, you should configure virtual network system-wide alarms, such as those for the analytics node, the config node, and so on. Under `project`, you should configure alarms related to project objects, such as virtual networks and similar objects.

To configure and monitor alarms using the Contrail UI:

1. Navigate to **Configure > Alarms > Project**, and select the desired project to access the **Alarm Rules** page.

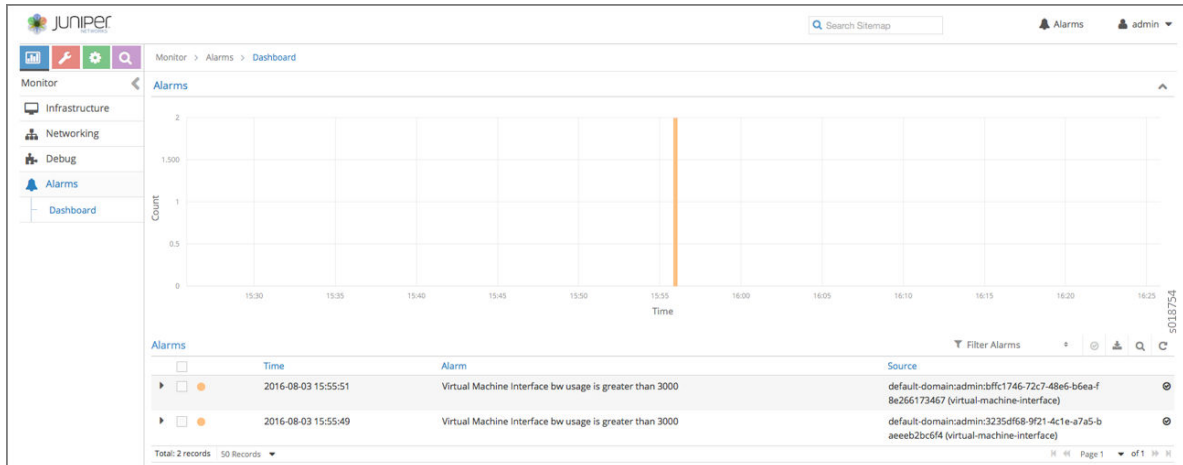


2. Click the Gear icon to add a new alarm configuration or to edit an existing alarm configuration. Use the **Edit** screen to define descriptions and to set up alarm rules. See [Table 62 on page 862](#) for field descriptions.

Table 62: Alarm Rules Fields

| Field       | Description  |
|-------------|--|
| Name        | Enter a name for the alarm.  |
| Severity    | Select the severity level of the alarm from the list.  |
| UVE Keys    | Select the list of UVE types to apply to this alarm.   |
| Description | Enter a description of the alarm.  |
| Rule        | Set up the alarm rules. Alarm rules are expressed as OR of AND terms. Each term has operand1, operand2, and the operation. Operand1 is the UVE attribute. Operand2 can be either another UVE attribute or a JSON value. The rules are evaluated in the <code>contrail-alarm-gen</code> service and an alarm is raised or cleared as needed on respective conditions. |

3. To monitor alarms, navigate to **Monitor > Alarms > Dashboard**. The **Dashboard** screen lists the active alarms in the system.



## Examples: Detecting Anomalies

The purpose of anomaly detection in Contrail is to identify a condition in which a metric deviates from its expected value, within given parameters.

Contrail uses a statistical process control model for time-series anomaly detection that can be computed online, in real-time. Raw metrics are sent as statistics by Sandesh generators embedded inside the UVEs. The model uses the running average and running standard deviation for a given raw metric. The model does not account for seasonality and linear trends in the metric.

The following example represents part of the UVE sent by the vRouter to the collector. The raw metrics are `phy_band_in_bps` and `phy_band_out_bps`.

The derived statistics are in `in_bps_ewm` and `out_bps_ewm`, which are generated when the model's EWM algorithm is applied to the raw metrics. The raw metrics and the derived statistics are part of the UVE and are sent to the collector.

```

struct EWMResult {
    3: u64 samples
    6: double mean
    7: double stddev
}

struct VrouterStatsAgent { // Agent stats

    1: string name (key="ObjectVRouter")

    2: optional bool deleted ...

    /** @display_name:Vrouter Physical Interface Input bandwidth Statistics*/

```

```

50: optional map<string,u64> phy_band_in_bps (tags="name:._key")

/** @display_name:Vrouter Physical Interface Output bandwidth Statistics*/

51: optional map<string,u64> phy_band_out_bps (tags="name:._key")

52: optional map<string,derived_stats_results.EWMResult> in_bps_ewm
(mstats="phy_band_in_bps:DSEWM:0.2")

53: optional map<string,derived_stats_results.EWMResult> out_bps_ewm
(mstats="phy_band_out_bps:DSEWM:0.2")
}

```

The following shows part of the UVE that lists the raw metric `phy_band_out_bps` and the derived statistic `out_bps_ewm`. The user can define an alarm based on the values in `sigma` or `in_stddev`.

```

- out_bps_ewm: {
  - eth0: {
    sigma: -0.425095,
    samples: 177,
    stddev: 6348.16,
    mean: 206712
  }
},
- phy_band_out_bps: {
  eth0: "204013"
},

```

s018755

## Configuring the User-Defined Log Statistic

Any deployment of Contrail cloud over an orchestration system requires tools for monitoring and troubleshooting the entire cloud deployment. Cloud data centers are built with a large collection of interconnected servers that provide computing and storage capacity for a variety of applications. The monitoring of the cloud and its infrastructure requires monitoring logs and messages sent to a variety of servers from many micro services.

Contrail analytics stores all of the monitored messages in the Contrail database node, and the analytics generates a large amount of useful information that aids in monitoring and troubleshooting the network.

Starting with Contrail Release 3.1, the user-defined log statistic feature provides additional abilities for monitoring and troubleshooting by enabling the user to set a counter on any regular Perl-type

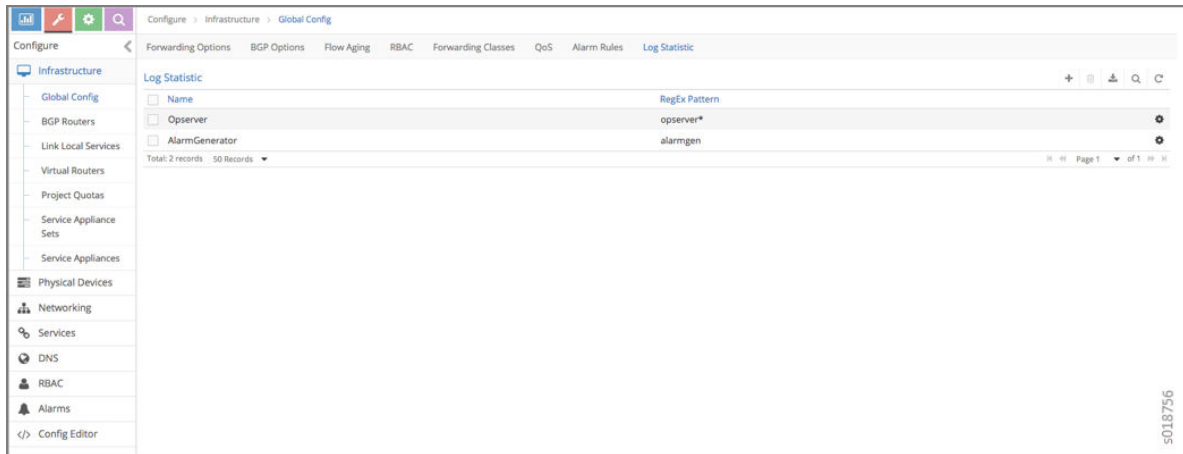


expression. Each time the pattern is found in any system logs, UVEs, or object logs, the counter is incremented.

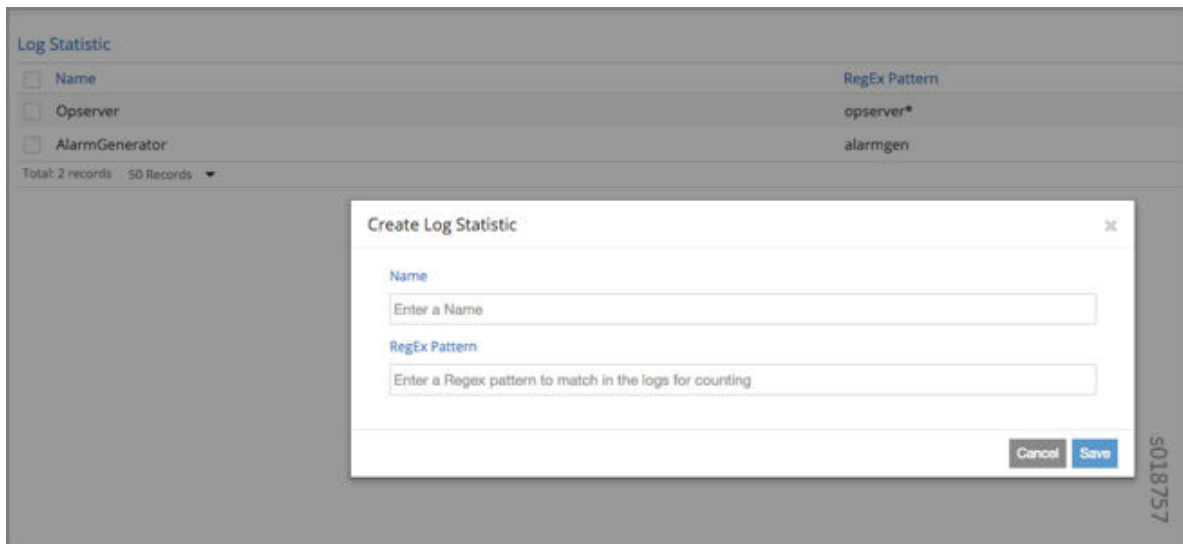
The user-defined log statistic can be configured from the Contrail UI or from the command line, using `vnc_api`.

To configure the user-defined log statistic from the Contrail UI:

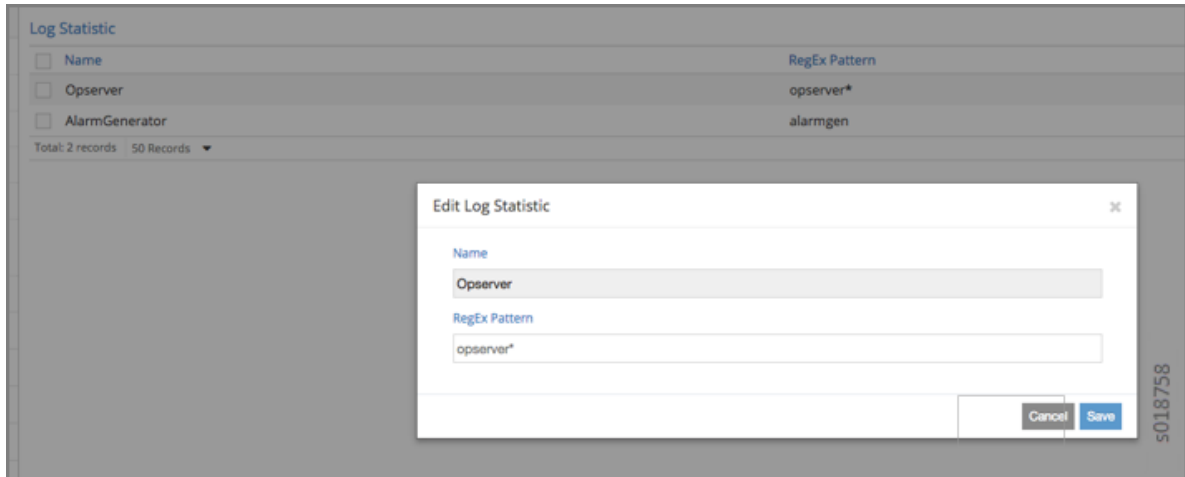
1. Navigate to **Configure > Infrastructure > Global Config** and select **Log Statistic**.



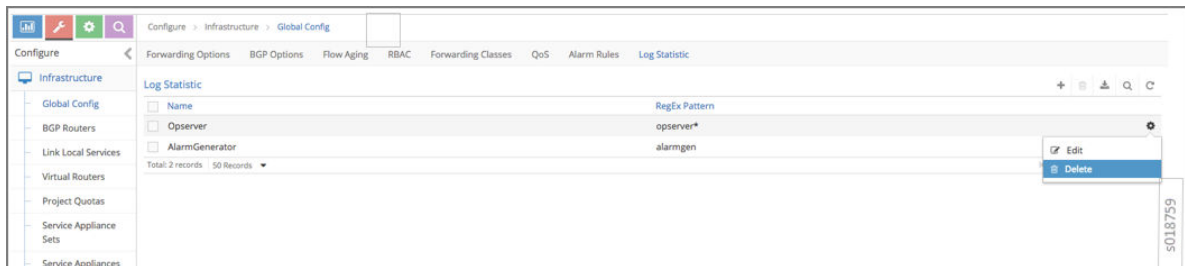
2. To create a log statistic, click the plus (+) icon to access the **Create Log Statistic** screen. Enter a name for the user-defined log statistic, and in the **RegExp Pattern** field, enter the Perl-type expression to look for and count.



3. To edit an existing log statistic, select the name of the statistic and click the Gear icon, then select **Edit** to access the **Edit Log Statistic** screen.



4. To delete a log statistic, select the name of the statistic and click the gear icon, then select the **Delete** option.



To configure the user-defined statistic from the vnc\_api:

```

user@host:~# python
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.

>> from vnc_api import vnc_api
>> from vnc_api.gen.resource_xsd import UserDefinedLogStat
>> from vnc_api.gen.resource_client import GlobalSystemConfig
>> vnc = vnc_api.VncApi('<username>', '<password>', '<tenant>')
>> gsc_uuid = vnc.global_system_configs_list()['global-system-configs'][0]['uuid']
>> gsc = vnc.global_system_config_read(id=gsc_uuid)

```

To list the counters:

```
>> [(x.name, x.pattern) for x in gsc.user_defined_log_statistics.statlist]

[('HostnameCounter', 'dummy'), ('MyIp', '10.84.14.38')]
```

To add a counter:

```
>> g=GlobalSystemConfig()
>> g.add_user_defined_counter(UserDefinedLogStat('Foo', 'Ba.*r'))
>> vnc.global_system_config_update(g)
```

To verify an addition:

```
>> gsc = vnc.global_system_config_read(id=gsc_uuid)
>> [(x.name, x.pattern) for x in gsc.user_defined_log_statistics.statlist]

[('HostnameCounter', 'dummy'), ('MyIp', '10.84.14.38'), ('Foo', 'Ba.*r')]
```

## Implementing the User-Defined Log Statistic

The statistics are sent as a counter that has been aggregated over a time period of 60 seconds.

A current sample from your system can be obtained from the UVE at:

<http://<analytics-ip>:8081/analytics/uves/user-defined-log-statistic/<name>>

You can also use the statistics table `UserDefinedLogStatTable` to get historical data with all supported aggregations such as SUM, AVG, and the like.

The schema for the table is at the following location:

<http://<ip>:8081/analytics/table/StatTable.UserDefinedCounter.count/schema>

### Schema for User-Defined Statistics Table

The following is the schema for the user-defined statistic table:

```
{
  "type": "STAT",
  "columns": [
    {
```

```
"datatype": "string",
  "index": true,
  "name": "Source",
  "suffixes": null
},
{
  "datatype": "int",
  "index": false,
  "name": "T",
  "suffixes": null
},
{
  "datatype": "int",
  "index": false,
  "name": "CLASS(T)",
  "suffixes": null
},
{
  "datatype": "int",
  "index": false,
  "name": "T=",
  "suffixes": null
},
{
  "datatype": "int",
  "index": false,
  "name": "CLASS(T=)",
  "suffixes": null
},
{
  "datatype": "uuid",
  "index": false,
  "name": "UUID",
  "suffixes": null
},
{
  "datatype": "int",
  "index": false,
  "name": "COUNT(count)",
  "suffixes": null
},
{
  "datatype": "int",
```

```
"index": false,
"name": "count.previous",
"suffixes": null
},
{
  "datatype": "int",
  "index": false,
  "name": "SUM(count.previous)",
  "suffixes": null
},
{
  "datatype": "int",
  "index": false,
  "name": "CLASS(count.previous)",
  "suffixes": null
},
{
  "datatype": "int",
  "index": false,
  "name": "MAX(count.previous)",
  "suffixes": null
},
{
  "datatype": "int",
  "index": false,
  "name": "MIN(count.previous)",
  "suffixes": null
},
{
  "datatype": "percentiles",
  "index": false,
  "name": "PERCENTILES(count.previous)",
  "suffixes": null
},
{
  "datatype": "avg",
  "index": false,
  "name": "AVG(count.previous)",
  "suffixes": null
},
{
  "datatype": "string",
  "index": true,
```

```
"name": "name",  
"suffixes": null  
}  
]  
}
```

## Alarms History

### IN THIS SECTION

- [Viewing Alarms History | 870](#)

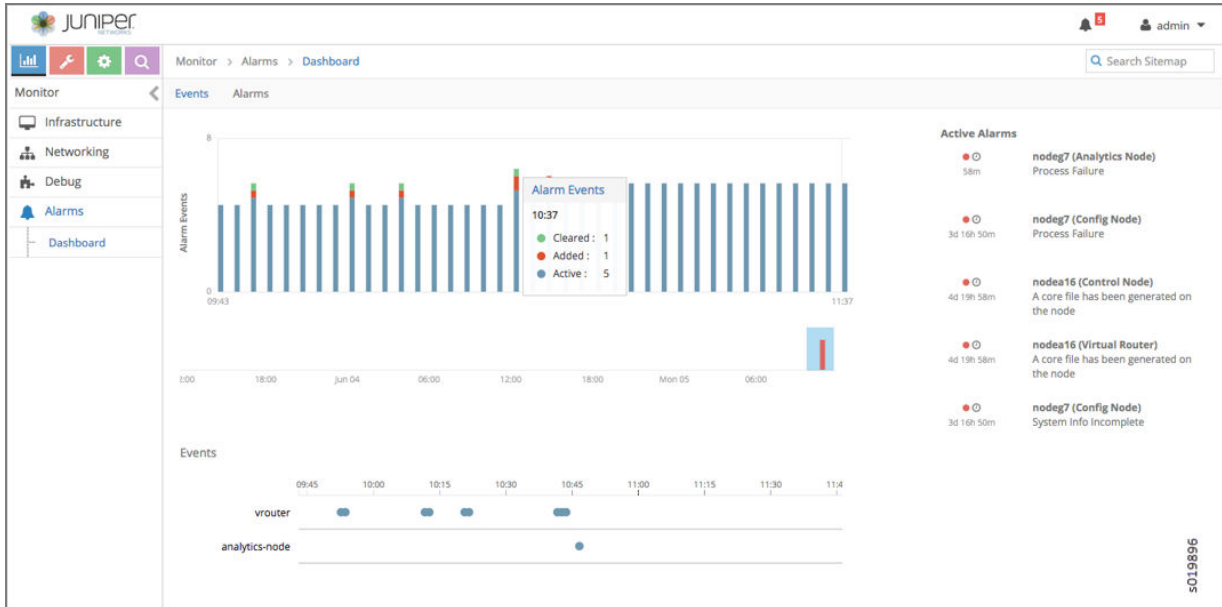
Starting with Contrail Release 4.0, you can view a history of alarms that were raised or reset. You can also view a history of user-visible entities (UVEs) that have been changed.

### Viewing Alarms History

In the Contrail Web user interface, new fields at **Monitor > Alarms > Dashboard > Alarms History** now display alarms history, including alarms that were set or reset. [Figure 218 on page 871](#) shows the alarms history, identifying the volume and types of alarms by time and the node types in which events are occurring. The right side panel lists by name the nodes in which active events are occurring.

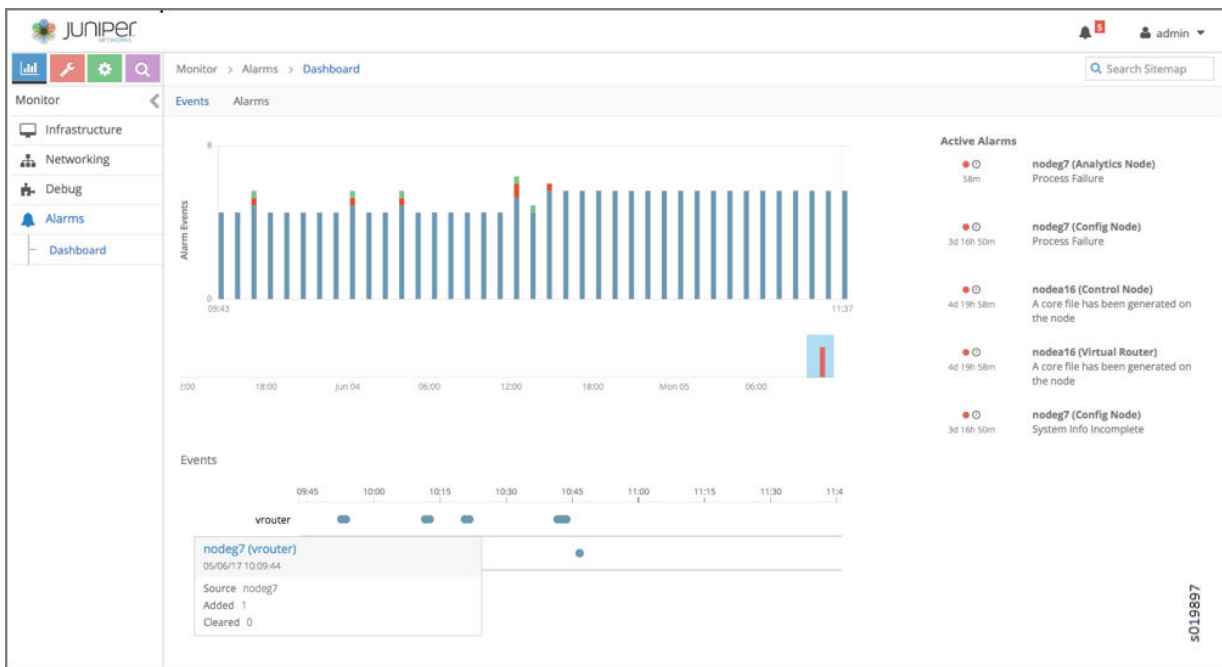
You can also use a `contrail-status` query to view the alarms history. Additionally, the `contrail-status` displays a history of added, updated, and removed information for UVEs in Contrail.

Figure 218: Alarms History Page



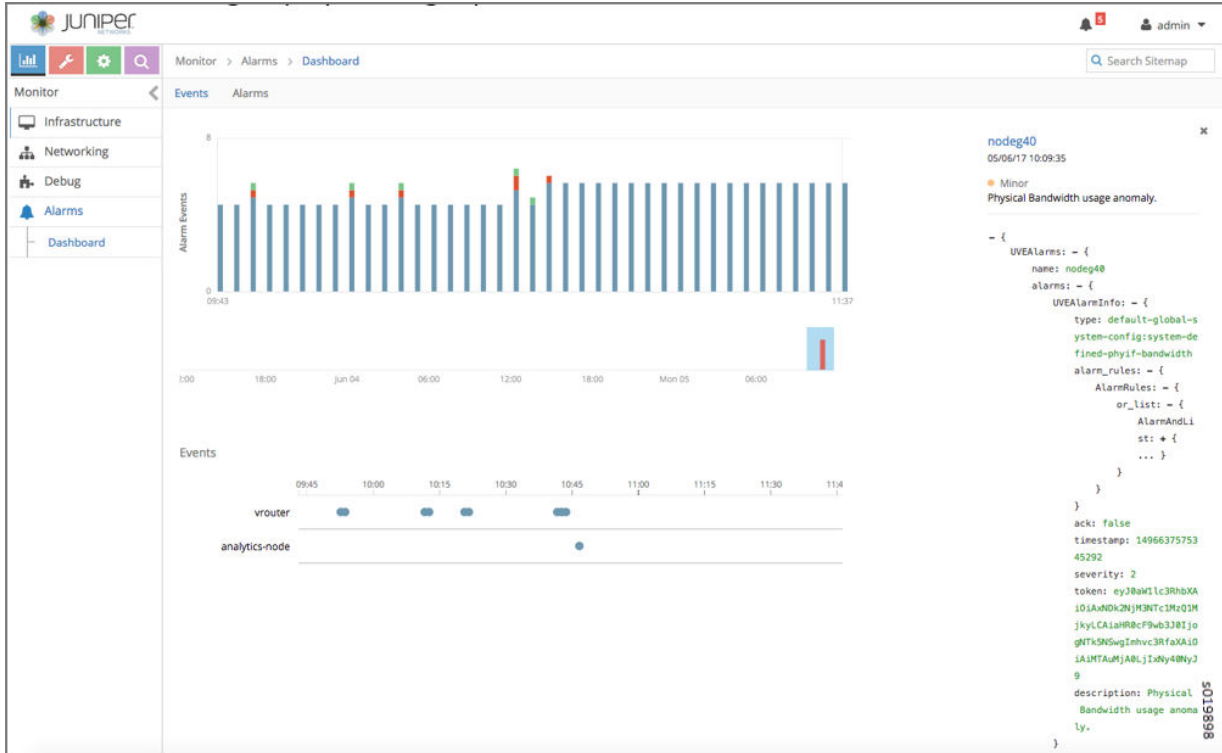
Tooltips are available on the Alarms History page. In the Events area, you can click on any node type listed to display a tooltip showing details of the events that have been added and cleared in that node, see [Figure 219 on page 871](#).

Figure 219: Events Log Tooltip



You can expand the event log in the right side panel to display a detailed event log. Click the name of any node in the list in the right panel, and the details of the current alarms are visible in the expanded view, see [Figure 220 on page 872](#).

**Figure 220: Detailed Event Log**



**RELATED DOCUMENTATION**

[User Configuration for Analytics Alarms and Log Statistics | 860](#)

## Node Memory and CPU Information

To help in monitoring and debugging, the following statistics have been added for all node types. The statistics are updated every 60 seconds.

- System CPU info
- System memory and CPU usage



- Memory and CPU usage of all processes

You can see a current sample from the UVE in your system at:

`http://<analytics-ip>:8081/analytics/uves/<node-type>/<hostname>?flat`

You can also use the statistics tables to get historical data with all supported aggregations, such as SUM, AVG, and so on:

- `NodeStatus.process_mem_cpu_usage`
- `NodeStatus.system_mem_cpu_usage`

The schema for the tables are at the following locations on your system:

`http://<analytics-ip>:8081/analytics/table/StatTable.NodeStatus.process_mem_cpu_usage/schema`

`http://<analytics-ip>:8081/analytics/table/StatTable.NodeStatus.system_mem_cpu_usage/schema`

## RELATED DOCUMENTATION

| [User Configuration for Analytics Alarms and Log Statistics](#) | 860

## Role- and Resource-Based Access Control for the Contrail Analytics API

In previous releases of Contrail, any user can access the Contrail analytics API by using queries to get historical information and by using UVEs to get state information.

Starting with Contrail Release 3.1, it is possible to restrict access such that only the `cloud-admin` user can access the Contrail analytics API.

Implementation details are as follows:

- An external user makes a REST API call to `contrail-analytics-api`, passing a token representing the user with the HTTP header `X-Auth-Token`.
- Based on the user role, `contrail-analytics-api` will only allow access for the `cloud-admin` user and reject the request (`HTTPUnauthorized`) for other users.

To set the `cloud_admin` user, use the following fields in `/etc/contrail/contrail-analytics-api.conf`:

- `aaa_mode`—Takes one of these values:
  - `no-auth`

- `cloud-admin`
- `cloud_admin_role`—The user with this role has full access to everything. By default, this is set to "admin". This role must be configured in Keystone.

## RELATED DOCUMENTATION

| [User Configuration for Analytics Alarms and Log Statistics](#) | 860

## Configuring Analytics as a Standalone Solution

### IN THIS SECTION

- [Overview: Contrail Analytics as a Standalone Solution](#) | 874
- [Configuration Examples for Standalone](#) | 875

Starting with Contrail 4.0, it is possible to configure Contrail Analytics as a standalone solution.

### Overview: Contrail Analytics as a Standalone Solution

Starting with Contrail 4.0 (containerized Contrail), Contrail Analytics can be configured as a standalone solution.

The following services are necessary for a standalone solution:

- `config`
- `webui`
- `analytics`
- `analyticsdb`

A standalone Contrail Analytics solution consists of the following containers:

- controller container with only `config` and `webui` services enabled
- `analytics` container

- analyticsdb container

## Configuration Examples for Standalone

The following are examples of default inventory file configurations for the controller container for standalone Contrail analytics.

### Examples: Inventory File Controller Components

The following are example analytics standalone solution inventory file configurations for Contrail controller container components.

#### Single Node Cluster

```
[contrail-controllers]
10.xx.32.10          controller_components=['config','webui']

[contrail-analyticsdb]
10.xx.32.10

[contrail-analytics]
10.xx.32.10
```

#### Multi-Node Cluster

```
[contrail-controllers]
10.xx.32.10          controller_components=['config','webui']
10.xx.32.11          controller_components=['config','webui']
10.xx.32.12          controller_components=['config','webui']

[contrail-analyticsdb]
10.xx.32.10
10.xx.32.11
10.xx.32.12

[contrail-analytics]
10.xx.32.10
```

```
10.xx.32.11
```

```
10.xx.32.12
```

## JSON Configuration Examples

The following are example JSON file configurations for (server.json) for Contrail analytics standalone solution.

### Example: JSON Single Node Cluster

```
{
  "cluster_id": "cluster1",
  "domain": "sm-domain.com",
  "id": "server1",
  "parameters" : {
    "provision": {
      "contrail_4": {
        "controller_components": "['config','webui']"
      },
      ...
    }
  }
}
```

### Example: JSON Multi-Node Cluster

```
{
  "cluster_id": "cluster1",
  "domain": "sm-domain.com",
  "id": "server1",
  "parameters" : {
    "provision": {
      "contrail_4": {
        "controller_components": "['config','webui']"
      },
      ...
    }
  },
  {
    "cluster_id": "cluster1",
```

```

"domain": "sm-domain.com",
"id": "server2",
"parameters" : {
  "provision": {
    "contrail_4": {
      "controller_components": "['config','webui']"
    },
    ...
  },
  ...
},
{
  "cluster_id": "cluster1",
  "domain": "sm-domain.com",
  "id": "server3",
  "parameters" : {
    "provision": {
      "contrail_4": {
        "controller_components": "['config','webui']"
      },
      ...
    },
    ...
  }
}

```

## RELATED DOCUMENTATION

[Configuring Secure Sandesh and Introspect for Contrail Analytics | 877](#)

*Understanding Contrail Analytics*

## Configuring Secure Sandesh and Introspect for Contrail Analytics

### IN THIS SECTION

- [Configuring Secure Sandesh Connection | 878](#)
- [Configuring Secure Introspect Connection | 878](#)

## Configuring Secure Sandesh Connection

All Contrail services use Sandesh, a southbound interface protocol based on Apache Thrift, to send analytics data such as system logs, object logs, UVEs, flow logs, and the like, to the collector service in the Contrail Analytics node. The Transport Layer Security (TLS) protocol is used for certificate exchange, mutual authentication, and negotiating ciphers to secure the Sandesh connection from potential tampering and eavesdropping.

To configure a secure Sandesh connection, configure the following parameters in all Contrail services that connect to the collector (Sandesh clients) and the Sandesh server.

| Parameter                    | Description                                 | Default   |
|------------------------------|---|---|
| [SANDESH].sandesh_keyfile    | Path to the node's private key              | <b>/etc/contrail/ssl/private/server-privkey.pem</b> |
| [SANDESH].sandesh_certfile   | Path to the node's public certificate       | <b>/etc/contrail/ssl/certs/server.pem</b>           |
| [SANDESH].sandesh_ca_cert    | Path to the CA certificate                  | <b>/etc/contrail/ssl/certs/ca-cert.pem</b>          |
| [SANDESH].sandesh_ssl_enable | Enable or disable secure Sandesh connection | <b>false</b>  |

## Configuring Secure Introspect Connection

All Contrail services are embedded with a web server that can be used to query the internal state of the data structures, view trace messages, and perform other extensive debugging. The Transport Layer Security (TLS) protocol is used for certificate exchange, mutual authentication, and negotiating ciphers to secure the introspect connection from potential tampering and eavesdropping.

To configure a secure introspect connection, configure the following parameters in the Contrail service, see [Table 63 on page 879](#).

**Table 63: Secure Introspect Parameters**

| Parameter                       | Description                                    | Default   |
|---------------------------------|--|---|
| [SANDESH].sandesh_keyfile       | Path to the node's private key                 | <b>/etc/contrail/ssl/private/server-privkey.pem</b> |
| [SANDESH].sandesh_certfile      | Path to the node's public certificate          | <b>/etc/contrail/ssl/certs/server.pem</b>           |
| [SANDESH].sandesh_ca_cert       | Path to the CA certificate                     | <b>/etc/contrail/ssl/certs/ca-cert.pem</b>          |
| [SANDESH].introspect_ssl_enable | Enable or disable secure introspect connection | <b>false</b>  |

# Using Contrail Analytics to Monitor and Troubleshoot the Network

## IN THIS CHAPTER

- [Monitoring the System | 880](#)
- [Debugging Processes Using the Contrail Introspect Feature | 884](#)
- [Monitor > Infrastructure > Dashboard | 889](#)
- [Monitor > Infrastructure > Control Nodes | 893](#)
- [Monitor > Infrastructure > Virtual Routers | 904](#)
- [Monitor > Infrastructure > Analytics Nodes | 918](#)
- [Monitor > Infrastructure > Config Nodes | 926](#)
- [Monitor > Networking | 930](#)
- [Query > Flows | 942](#)
- [Query > Logs | 952](#)
- [Understanding Flow Sampling | 959](#)
- [Example: Debugging Connectivity Using Monitoring for Troubleshooting | 962](#)

## Monitoring the System

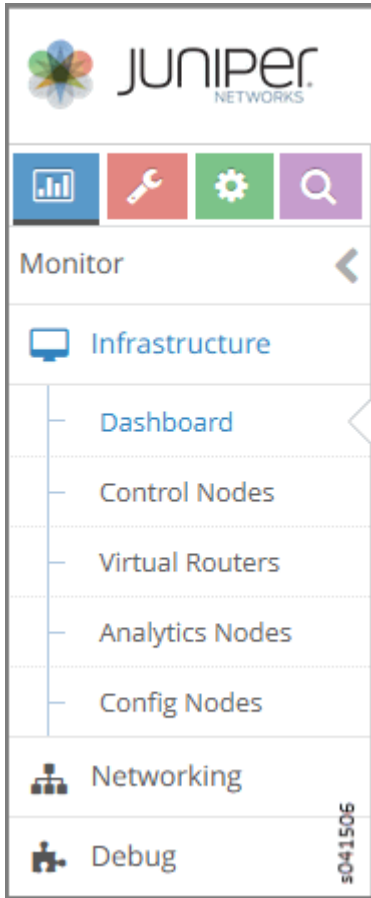
The **Monitor** icon on the Contrail Controller provides numerous options so you can view and analyze usage and other activity associated with all nodes of the system, through the use of reports, charts, and detailed lists of configurations and system activities.

Monitor pages support monitoring of infrastructure components—control nodes, virtual routers, analytics nodes, and config nodes. Additionally, users can monitor networking and debug components.

Use the menu options available from the **Monitor** icon to configure and view the statistics you need for better understanding of the activities in your system. See [Figure 221 on page 881](#)



Figure 221: Monitor Menu



See [Table 64 on page 881](#) for descriptions of the items available under each of the menu options from the **Monitor** icon.

Table 64: Monitor Menu Options

| Option                               | Description   |
|--------------------------------------|---|
| <b>Infrastructure &gt; Dashboard</b> | Shows “at-a-glance” status view of the infrastructure components, including the numbers of virtual routers, control nodes, analytics nodes, and config nodes currently operational, and a bubble chart of virtual routers showing the CPU and memory utilization, log messages, system information, and alerts. See <i>Monitor &gt; Infrastructure &gt; Dashboard</i> . |

Table 64: Monitor Menu Options (*Continued*)

| Option                                     | Description  |
|--|--|
| <b>Infrastructure &gt; Control Nodes</b>   | <p>View a summary for all control nodes in the system, and for each control node, view:</p> <ul style="list-style-type: none"> <li>• Graphical reports of memory usage and average CPU load.</li> <li>• Console information for a specified time period.</li> <li>• A list of all peers with details about type, ASN, and the like.</li> <li>• A list of all routes, including next hop, source, local preference, and the like.</li> </ul> <p><i>See Monitor &gt; Infrastructure &gt; Control Nodes.</i></p>  |
| <b>Infrastructure &gt; Virtual Routers</b> | <p>View a summary of all vRouters in the system, and for each vRouter, view:</p> <ul style="list-style-type: none"> <li>• Graphical reports of memory usage and average CPU load.</li> <li>• Console information for a specified time period.</li> <li>• A list of all interfaces with details such as label, status, associated network, IP address, and the like.</li> <li>• A list of all associated networks with their ACLs and VRFs.</li> <li>• A list of all active flows with source and destination details, size, and time.</li> </ul> <p><i>See Monitor &gt; Infrastructure &gt; Virtual Routers.</i></p> |
| <b>Infrastructure &gt; Analytics Nodes</b> | <p>View activity for the analytics nodes, including memory and CPU usage, analytics host names, IP address, status, and more. <i>See Monitor &gt; Infrastructure &gt; Analytics Nodes.</i></p>   |
| <b>Infrastructure &gt; Config Nodes</b>    | <p>View activity for the config nodes, including memory and CPU usage, config host names, IP address, status, and more. <i>See Monitor &gt; Infrastructure &gt; Config Nodes.</i></p>  |

Table 64: Monitor Menu Options (*Continued*)

| Option                           | Description   |
|----------------------------------|---|
| <b>Networking &gt; Networks</b>  | <p>For all virtual networks for all projects in the system, view graphical traffic statistics, including:</p> <ul style="list-style-type: none"> <li>• Total traffic in and out.</li> <li>• Inter VN traffic in and out.</li> <li>• The most active ports, peers, and flows for a specified duration.</li> <li>• All traffic ingress and egress from connected networks, including their attached policies.</li> </ul> <p>See <i>Monitor &gt; Networking</i>.</p> |
| <b>Networking &gt; Dashboard</b> | <p>For all virtual networks for all projects in the system, view graphical traffic statistics, including:</p> <ul style="list-style-type: none"> <li>• Total traffic in and out.</li> <li>• Inter VN traffic in and out.</li> </ul> <p>You can view the statistics in varying levels of granularity, for example, for a whole project, or for a single network. See <i>Monitor &gt; Networking</i>.</p>   |
| <b>Networking &gt; Projects</b>  | View essential information about projects in the system including name, associated networks, and traffic in and out.  |
| <b>Networking &gt; Networks</b>  | View essential information about networks in the system including name and traffic in and out.  |
| <b>Networking &gt; Instances</b> | View essential information about instances in the system including name, associated networks, interfaces, vRouters, and traffic in and out.   |

Table 64: Monitor Menu Options (*Continued*)

| Option                           | Description   |
|----------------------------------|---|
| <b>Debug &gt; Packet Capture</b> | <ul style="list-style-type: none"> <li>• Add and manage packet analyzers.</li> <li>• Attach packet captures and configure their details.</li> <li>• View a list of all packet analyzers in the system and the details of their configurations, including source and destination networks, ports, and IP addresses.</li> </ul> |

## RELATED DOCUMENTATION

*Monitor > Infrastructure > Dashboard*

*Monitor > Infrastructure > Control Nodes*

*Monitor > Infrastructure > Virtual Routers*

*Monitor > Networking*

*Query > Logs*

*Query > Flows*

## Debugging Processes Using the Contrail Introspect Feature

This topic describes how to use the Sandesh infrastructure and the Contrail Introspect feature to debug processes.

Introspect is a mechanism for taking a program object and querying information about it.

Sandesh is the name of a unified infrastructure in the Contrail Virtual Networking solution.

Sandesh is a way for the Contrail daemons to provide a request-response mechanism. Requests and responses are defined in Sandesh format and the Sandesh compiler generates code to process the requests and send responses.

Sandesh also provides a way to use a Web browser to send Sandesh requests to a Contrail daemon and get the Sandesh responses. This feature is used to debug processes by looking into the operational status of the daemons.

Each Contrail daemon starts an HTTP server, with the following page types:

- The main index.html listing all Sandesh modules and the links to them.
- Sandesh module pages that present HTML forms for each Sandesh request.
- XML-based dynamically-generated pages that display Sandesh responses.
- An automatically generated page that shows all code needed for rendering and all HTTP server-client interactions.

You can display the HTTP introspect of a Contrail daemon directly by accessing the following Introspect ports:

- `<controller-ip>:8083`. This port displays the `contrail-control` introspect port.
- `<compute-ip>:8085` This port displays the `contrail-vrouter-agent` introspect port.

Another way to launch the Introspect page is by browsing to a particular node page using the Contrail Web user interface.

[Figure 222 on page 886](#) shows the contrail-control infrastructure page. Notice the Introspect link at the bottom of the Control Nodes Details tab window.

Figure 222: Control Nodes Details Tab Window

The screenshot shows the Juniper Networks Control Nodes Details Tab Window. The window is titled 'Control Node' and displays the following information:

- Hostname:** b6s24
- IP Address:** 192.168.68.2
- Version:** 2.20 (Build 64)
- Overall Node Status:** Up since 7d 6h 11m
- Processes:**
  - Control Node:** Up since 7d 20h 6m
  - Ifmap Connection:** 192.168.68.2 (Up since 7d 20h 5m)
  - Analytics Node:** 192.168.68.3 (Up), 192.168.68.1
  - Analytics Messages:** 441669 [1000.57 MB]
  - Peers:** BGP Peers: 4 Total
  - vRouters:** 10 Established in Sync, 7 subscribed for configuration
- CPU:** 0.04 %
- Memory:** 86.76 MB
- Last Log:** July 28, 2015 at 2:23:33 PM PDT
- Status:** Introspect (circled in red)

The window also displays two graphs:

- CPU and Memory Utilization:** A line graph showing CPU Share (%) and Memory utilization over time. The CPU share is approximately 0.04% and memory is approximately 86.76 MB.
- Control Node CPU/Memory Utilization:** A line graph showing CPU Share (%) and Memory utilization over time. The CPU share is approximately 0.04% and memory is approximately 86.76 MB.

The window also displays a list of processes and a status field. The status field is circled in red and contains the text 'Introspect'.

The following are the Sandesh modules for the Contrail control process (contrail-control) Introspect port.

- bgp\_peer.xml
- control\_node.xml
- cpuinfo.xml
- discovery\_client\_stats.xml
- ifmap\_log.xml
- ifmap\_server\_show.xml
- rtarget\_group.xml

- sandesh\_trace.xml
- sandesh\_uve.xml
- service\_chaining.xml
- static\_route.xml
- task.xml
- xmpp\_server.xml

Figure 223 on page 887 shows the Controller Introspect window.

**Figure 223: Controller Introspect Window**

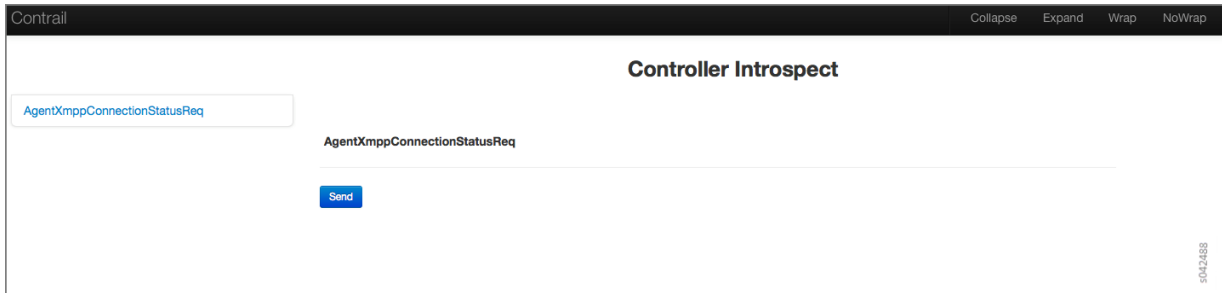


Figure 224 on page 887 shows an example of the BGP Peer (bgp\_peer.xml) Introspect page.

**Figure 224: BGP Peer Introspect Page**

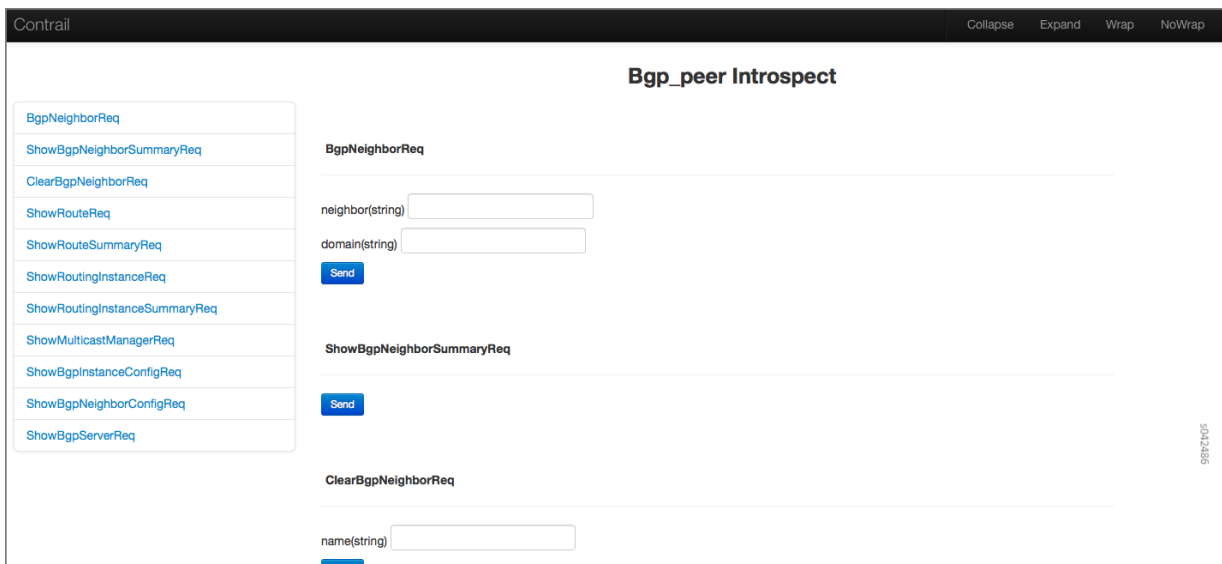


Figure 225 on page 888 shows an example of the BGP Neighbor Summary Introspect page.

Figure 225: BGP Neighbor Summary Introspect Page

| Contrail                   |         |            |               |               |          |          |           |             |               |              |
|----------------------------|---------|------------|---------------|---------------|----------|----------|-----------|-------------|---------------|--------------|
| ShowBgpNeighborSummaryResp |         |            |               |               |          |          |           |             |               |              |
| neighbors                  |         |            |               |               |          |          |           |             |               |              |
| peer                       | deleted | deleted_at | peer_address  | peer_id       | peer_asn | encoding | peer_type | state       | local_address | local_id     |
| b6s23                      | false   | -          | 192.168.68.1  | 192.168.68.1  | 64512    | BGP      | internal  | Established | 192.168.68.2  | 192.168.68.2 |
| b6s25                      | false   | -          | 192.168.68.3  | 192.168.68.3  | 64512    | BGP      | internal  | Established | 192.168.68.2  | 192.168.68.2 |
| mx1                        | false   | -          | 192.168.100.1 | 192.168.100.1 | 64512    | BGP      | internal  | Established | 192.168.68.2  | 192.168.68.2 |
| mx2                        | false   | -          | 192.168.100.2 | 192.168.100.2 | 64512    | BGP      | internal  | Established | 192.168.68.2  | 192.168.68.2 |
| b6s28                      | false   | -          | 192.168.68.6  | -             | 0        | XMP      | internal  | Established | 192.168.68.2  | -            |
| b6s18                      | false   | -          | 192.168.69.5  | -             | 0        | XMP      | internal  | Established | 192.168.68.2  | -            |
| b6s13                      | false   | -          | 192.168.69.8  | -             | 0        | XMP      | internal  | Established | 192.168.68.2  | -            |
| b6s7                       | false   | -          | 192.168.69.11 | -             | 0        | XMP      | internal  | Established | 192.168.68.2  | -            |
| b6s33                      | false   | -          | 192.168.68.11 | -             | 0        | XMP      | internal  | Established | 192.168.68.2  | -            |
| b6s9                       | false   | -          | 192.168.69.10 | -             | 0        | XMP      | internal  | Established | 192.168.68.2  | -            |
| b6s26                      | false   | -          | 192.168.68.4  | -             | 0        | XMP      | internal  | Established | 192.168.68.2  | -            |

The following are the Sandesh modules for the Contrail vRouter agent (**contrail-vrouter-agent**) Introspect port.

- agent.xml
- agent\_stats\_interval.xml
- cfg.xml
- controller.xml
- cpuinfo.xml
- diag.xml
- discovery\_client\_stats.xml
- flow\_stats\_interval.xml
- ifmap\_agent.xml
- kstate.xml
- multicast.xml
- pkt.xml
- port\_ipc.xml
- sandesh\_trace.xml



- sandesh\_uve.xml
- services.xml
- stats\_interval.xml
- task.xml
- xmpp\_server.xml

Figure 226 on page 889 shows an example of the Agent (agent.xml) Introspect page.

Figure 226: Agent Introspect Page

The screenshot shows the 'AgentXmppConnectionStatus' page in the Contrail interface. It features a table with columns for controller IP, state, configuration, and connection details. Two peers are listed, both in an 'Established' state.

| peer | controller_ip | state       | cfg_controller | mcast_controller | last_state | last_event            | last_state_at               | flap_count | flap_time                   | rx |
|------|---------------|-------------|----------------|------------------|------------|-----------------------|-----------------------------|------------|-----------------------------|----|
|      | 192.168.68.3  | Established | Yes            | No               | OpenSent   | xmsm::EvXmppKeepalive | 2015-Jul-21 01:20:57.616019 | 2          | 2015-Jul-21 01:20:57.555077 | rx |
|      | 192.168.68.2  | Established | No             | Yes              | OpenSent   | xmsm::EvXmppKeepalive | 2015-Jul-21 01:20:59.599875 | 2          | 2015-Jul-21 01:20:59.548692 | rx |

## Monitor > Infrastructure > Dashboard

### IN THIS SECTION

- Monitor Dashboard | 890
- Monitor Individual Details from the Dashboard | 890
- Using Bubble Charts | 891
- Color-Coding of Bubble Charts | 892

Use **Monitor > Infrastructure > Dashboard** to get an “at-a-glance” view of the system infrastructure components, including the numbers of virtual routers, control nodes, analytics nodes, and config nodes currently operational, a bubble chart of virtual routers showing the CPU and memory utilization, log messages, system information, and alerts.

## Monitor Dashboard

Click **Monitor > Infrastructure > Dashboard** on the left to view the **Dashboard**. See [Figure 227 on page 890](#).

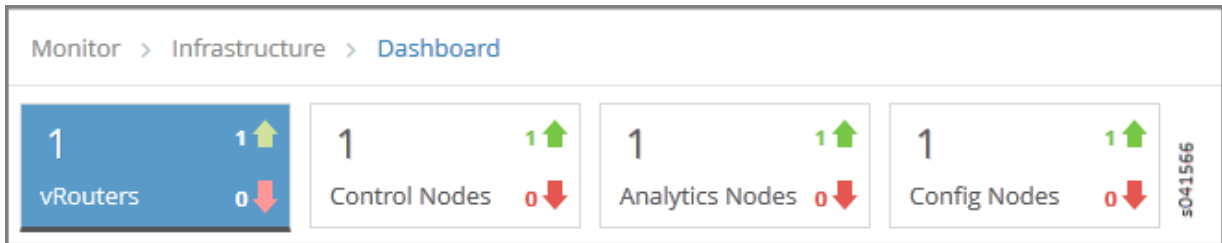
**Figure 227: Monitor > Infrastructure > Dashboard**



## Monitor Individual Details from the Dashboard

Across the top of the **Dashboard** screen are summary boxes representing the components of the system that are shown in the statistics. See [Figure 228 on page 891](#). Any of the control nodes, virtual routers, analytics nodes, and config nodes can be monitored individually and in detail from the **Dashboard** by clicking an associated box, and drilling down for more detail.

Figure 228: Dashboard Summary Boxes



Detailed information about monitoring each of the areas represented by the boxes is provided in the links in [Table 65 on page 891](#).

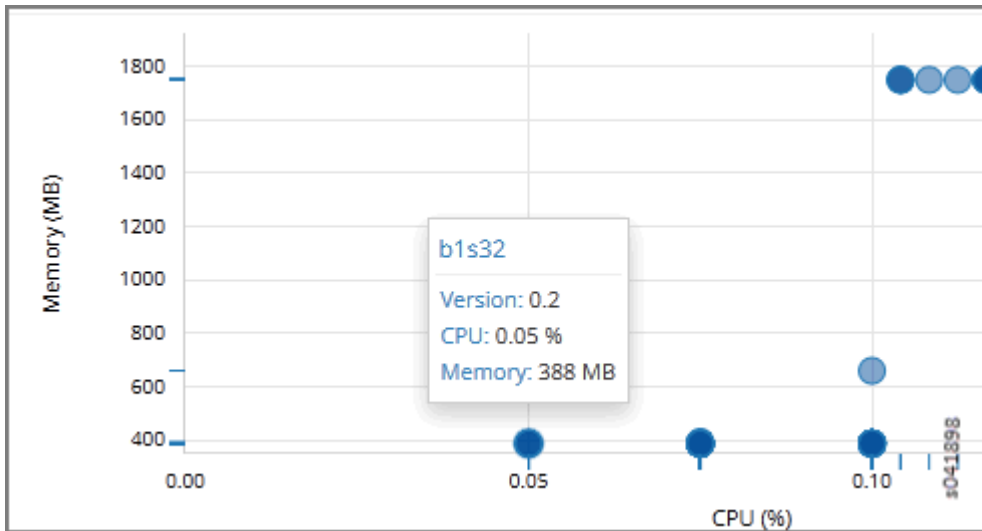
Table 65: Dashboard Summary Boxes

| Box                    | For More Information                                    |
|------------------------|---|
| <b>vRouters</b>        | <i>Monitor &gt; Infrastructure &gt; Virtual Routers</i> |
| <b>Control Nodes</b>   | <i>Monitor &gt; Infrastructure &gt; Control Nodes</i>   |
| <b>Analytics Nodes</b> | <i>Monitor &gt; Infrastructure &gt; Analytics Nodes</i> |
| <b>Config Nodes</b>    | <i>Monitor &gt; Infrastructure &gt; Config Nodes</i>    |

## Using Bubble Charts

Bubble charts show the CPU and memory utilization of components contributing to the current analytics display, including vRouters, control nodes, config nodes, and the like. You can hover over any bubble to get summary information about the component it represents; see [Figure 229 on page 892](#). You can click through the summary information to get more details about the component.

Figure 229: Bubble Summary Information



### Color-Coding of Bubble Charts

Bubble charts use the following color-coding scheme:

#### *Control Nodes*

- Blue—working as configured.
- Red—error, at least one configured peer is down.

#### *vRouters*

- Blue—working, but no instance is launched.
- Green—working with at least one instance launched.
- Red—error, there is a problem with connectivity or a vRouter is in a failed state.

### RELATED DOCUMENTATION

*Monitor > Infrastructure > Virtual Routers*

*Monitor > Infrastructure > Control Nodes*

*Monitor > Infrastructure > Analytics Nodes*

*Monitor > Infrastructure > Config Nodes*

## Monitor > Infrastructure > Control Nodes

### IN THIS SECTION

- [Monitor Control Nodes Summary | 893](#)
- [Monitor Individual Control Node Details | 894](#)
- [Monitor Individual Control Node Console | 896](#)
- [Monitor Individual Control Node Peers | 899](#)
- [Monitor Individual Control Node Routes | 901](#)

Use **Monitor > Infrastructure > Control Nodes** to gain insight into usage statistics for control nodes.

### Monitor Control Nodes Summary

Select **Monitor > Infrastructure > Control Nodes** to see a graphical chart of average memory usage versus average CPU percentage usage for all control nodes in the system. Also on this screen is a list of all control nodes in the system. See [Figure 230 on page 893](#). See [Table 66 on page 894](#) for descriptions of the fields on this screen.

Figure 230: Control Nodes Summary



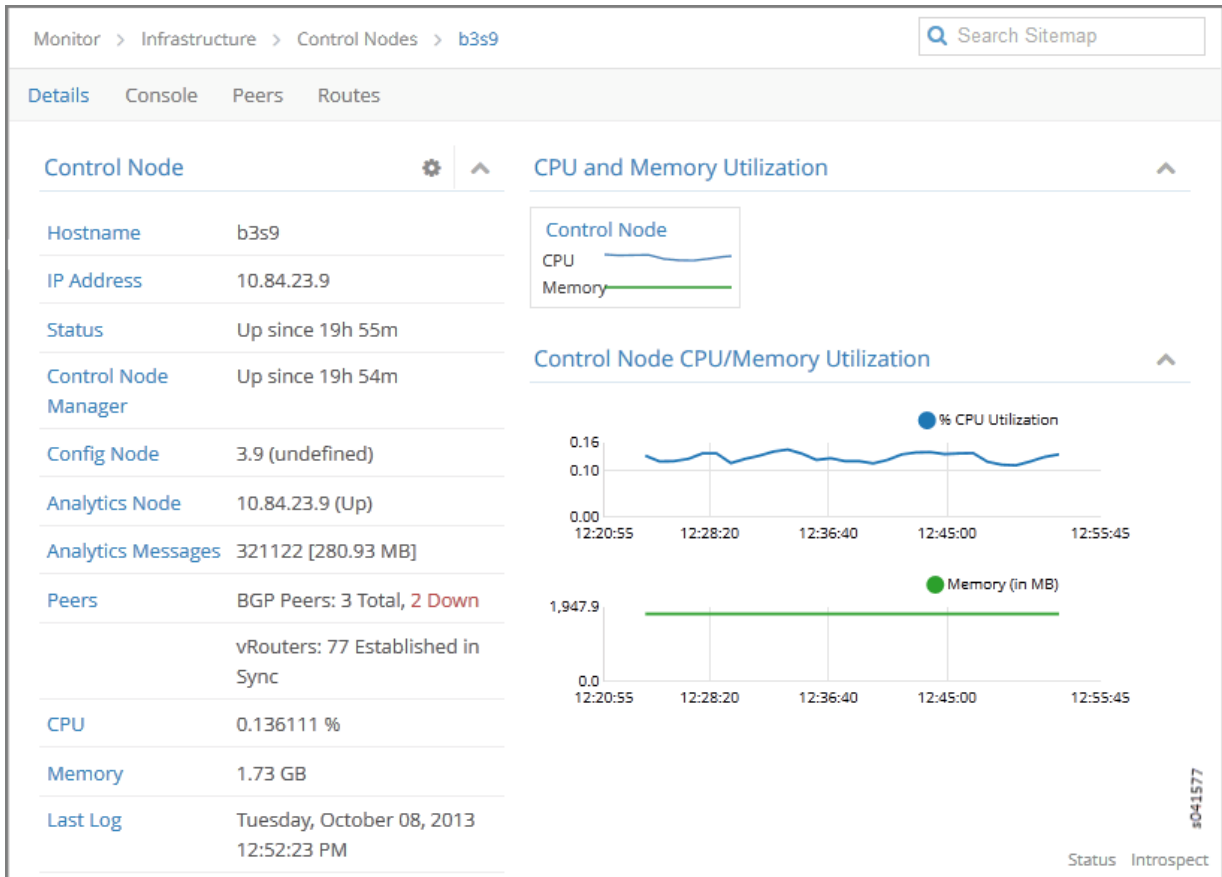
Table 66: Control Nodes Summary Fields

| Field                               | Description   |
|-------------------------------------|---|
| <b>Host name</b>                    | The name of the control node.   |
| <b>IP Address</b>                   | The IP address of the control node.   |
| <b>Version</b>                      | The software version number that is installed on the control node.                      |
| <b>Status</b>                       | The current operational status of the control node – Up or Down.                        |
| <b>CPU (%)</b>                      | The CPU percentage currently in use by the selected control node.                       |
| <b>Memory</b>                       | The memory in MB currently in use and the total memory available for this control node. |
| <b>Total Peers</b>                  | The total number of peers for this control node.  |
| <b>Established in Sync Peers</b>    | The total number of peers in sync for this control node.                                |
| <b>Established in Sync vRouters</b> | The total number of vRouters in sync for this control node.                             |

### Monitor Individual Control Node Details

Click the name of any control nodes listed under the **Control Nodes** title to view an array of graphical reports of usage and numerous details about that node. There are several tabs available to help you probe into more details about the selected control node. The first tab is the **Details** tab; see [Figure 231 on page 895](#).

Figure 231: Individual Control Node—Details Tab



The Details tab provides a summary of the status and activity on the selected node, and presents graphical displays of CPU and memory usage. See [Table 67 on page 895](#) for descriptions of the fields on this tab.

Table 67: Individual Control Node—Details Tab Fields

| Field                       | Description   |
|-----------------------------|---|
| <b>Hostname</b>             | The host name defined for this control node.        |
| <b>IP Address</b>           | The IP address of the selected node.                |
| <b>Status</b>               | The operational status of the control node.         |
| <b>Control Node Manager</b> | The operational status of the control node manager. |

Table 67: Individual Control Node—Details Tab Fields *(Continued)*

| Field   | Description  |
|---|--|
| <b>Config Node</b>                              | The IP address of the configuration node associated with this control node.                                    |
| <b>Analytics Node</b>                           | The IP address of the node from which analytics (monitor) information is derived.                              |
| <b>Analytics Messages</b>                       | The total number of analytics messages in and out from this node.  |
| <b>Peers</b>                                    | The total number of peers established for this control node and how many are in sync and of what type.         |
| <b>CPU</b>                                      | The average percent of CPU load incurred by this control node.   |
| <b>Memory</b>                                   | The average memory usage incurred by this control node.  |
| <b>Last Log</b>                                 | The date and time of the last log message issued about this control node.                                      |
| <b>Control Node CPU/<br/>Memory Utilization</b> | A graphic display x, y chart of the average CPU load and memory usage incurred by this control node over time. |

## Monitor Individual Control Node Console

Click the **Console** tab for an individual control node to display system logging information for a defined time period, with the last 5 minutes of information as the default display. See [Figure 232 on page 897](#).



Figure 232: Individual Control Node—Console Tab

See [Table 68 on page 897](#) for descriptions of the fields on the **Console** tab screen.

Table 68: Control Node: Console Tab Fields

| Field               | Description  |
|---------------------|--|
| <b>Time Range</b>   | Select a timeframe for which to review logging information as sent to the console. There are 11 options, ranging from the <b>Last 5 mins</b> through to the <b>Last 24 hrs</b> . The default display is for the <b>Last 5 mins</b> . |
| <b>Log Category</b> | Select a log category to display: <ol style="list-style-type: none"> <li>1. All</li> <li>2. _default_</li> <li>3. XMPP</li> <li>4. TCP</li> </ol>  |

Table 68: Control Node: Console Tab Fields (*Continued*)

| Field               | Description  |
|---------------------|--|
| <b>Log Type</b>     | Select a log type to display.  |
| <b>Log Level</b>    | Select a log severity level to display: <ol style="list-style-type: none"> <li>1. SYS_EMERG</li> <li>2. SYS_ALERT</li> <li>3. SYS_CRIT</li> <li>4. SYS_ERR</li> <li>5. SYS_WARN</li> <li>6. SYS_NOTICE</li> <li>7. SYS_INFO</li> <li>8. SYS_DEBUG</li> </ol>                                   |
| <b>Search</b>       | Enter any text string to search and display logs containing that string.   |
| <b>Limit</b>        | Select from a list an amount to limit the number of messages displayed: <ol style="list-style-type: none"> <li>1. No Limit</li> <li>2. Limit 10 messages</li> <li>3. Limit 50 messages</li> <li>4. Limit 100 messages</li> <li>5. Limit 200 messages</li> <li>6. Limit 500 messages</li> </ol> |
| <b>Auto Refresh</b> | Click the check box to automatically refresh the display if more messages occur.   |
| <b>Display Logs</b> | Click this button to refresh the display if you change the display criteria.   |

**Table 68: Control Node: Console Tab Fields** *(Continued)*

| Field           | Description  |
|-----------------|--|
| <b>Reset</b>    | Click this button to clear any selected display criteria and reset all criteria to their default settings. |
| <b>Time</b>     | This column lists the time received for each log message displayed.  |
| <b>Category</b> | This column lists the log category for each log message displayed.   |
| <b>Log Type</b> | This column lists the log type for each log message displayed.   |
| <b>Log</b>      | This column lists the log message for each log displayed.  |

### Monitor Individual Control Node Peers

The **Peers** tab displays the peers for an individual control node and their peering state. Click the expansion arrow next to the address of any peer to reveal more details. See [Figure 233 on page 900](#).

Figure 233: Individual Control Node—Peers Tab

Monitor > Infrastructure > Control Nodes > b3s9 Q Search Sitemap

Details Console **Peers** Routes

### Peers Q ^

| Peer           | Peer Type | Peer ASN | Status               | Last flap | Messages (Recv/Sent) |
|----------------|-----------|----------|----------------------|-----------|----------------------|
| ▶ 10.84.23.252 | BGP       | 64512    | Active, -            | -         | 0/ 0                 |
| ▶ 10.84.23.8   | BGP       | 64512    | Established, in sync | -         | 3754/ 3758           |
| ▶ 10.84.23.253 | BGP       | 64512    | Connect, -           | -         | 0/ 0                 |
| ▶ 10.84.21.4   | XMPP      | -        | Established, in sync | -         | 2751/ 5189           |
| ▶ 10.84.21.5   | XMPP      | -        | Established, in sync | -         | 2753/ 5802           |
| ▶ 10.84.21.6   | XMPP      | -        | Established, in sync | -         | 2752/ 4264           |
| ▲ 10.84.21.34  | XMPP      | -        | Established, in sync | -         | 2753/ 5659           |

Details:

```

- {
  name: "b3s9:10.84.21.34",
  value: - {
    XmppPeerInfoData: - {
      state_info: - {
        last_state: "Active",
        state: "Established",
        last_state_at: 1381190447915913
      },
      peer_stats_info: - {

```

s041579

See [Table 69 on page 900](#) for descriptions of the fields on the **Peers** tab screen.

Table 69: Control Node: Peers Tab Fields

| Field            | Description                               |
|------------------|---|
| <b>Peer</b>      | The hostname of the peer.                 |
| <b>Peer Type</b> | The type of peer.                         |
| <b>Peer ASN</b>  | The autonomous system number of the peer. |
| <b>Status</b>    | The current status of the peer.           |

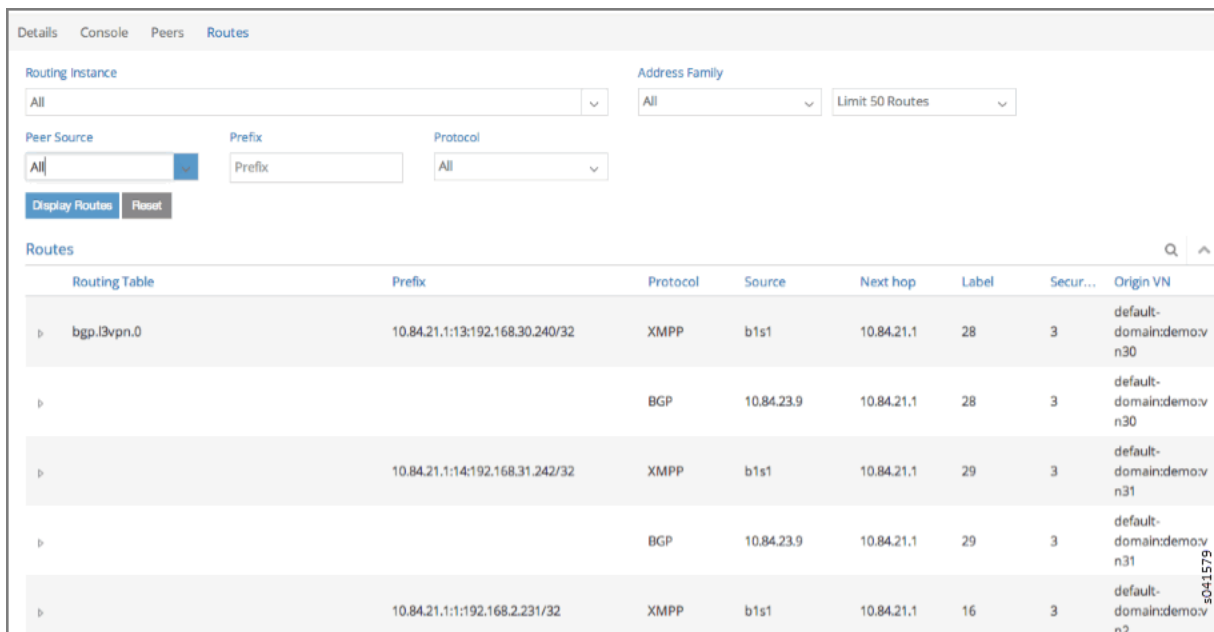
**Table 69: Control Node: Peers Tab Fields (Continued)**

| Field                       | Description  |
|-----------------------------|--|
| <b>Last flap</b>            | The last flap detected for this peer.                    |
| <b>Messages (Recv/Sent)</b> | The number of messages sent and received from this peer. |

### Monitor Individual Control Node Routes

The **Routes** tab displays active routes for this control node and lets you query the results. Use horizontal and vertical scroll bars to view more results. Click the expansion icon next to a routing table name to reveal more details about the selected route. See [Figure 234 on page 901](#).

**Figure 234: Individual Control Node—Routes Tab**



See [Table 70 on page 902](#) for descriptions of the fields on the **Routes** tab screen.

Table 70: Control Node: Routes Tab Fields

| Field                   | Description   |
|-------------------------|---|
| <b>Routing Instance</b> | You can select a single routing instance from a list of all instances for which to display the active routes.   |
| <b>Address Family</b>   | Select an address family for which to display the active routes: <ol style="list-style-type: none"> <li>1. All (default)</li> <li>2. l3vpn</li> <li>3. inet</li> <li>4. inetmcast</li> </ol>              |
| (Limit Field)           | Select to limit the display of active routes: <ol style="list-style-type: none"> <li>1. Limit 10 Routes</li> <li>2. Limit 50 Routes</li> <li>3. Limit 100 Routes</li> <li>4. Limit 200 Routes</li> </ol>  |
| <b>Peer Source</b>      | Select from a list of available peers the peer for which to display the active routes, or select All.   |
| <b>Prefix</b>           | Enter a route prefix to limit the display of active routes to only those with the designated prefix.  |
| <b>Protocol</b>         | Select a protocol for which to display the active routes: <ol style="list-style-type: none"> <li>1. All (default)</li> <li>2. XMPP</li> <li>3. BGP</li> <li>4. ServiceChain</li> <li>5. Static</li> </ol> |

Table 70: Control Node: Routes Tab Fields *(Continued)*

| Field                 | Description  |
|-----------------------|--|
| <b>Display Routes</b> | Click this button to refresh the display of routes after selecting different display criteria. |
| <b>Reset</b>          | Click this button to clear any selected criteria and return the display to default values.     |
| <i>Column</i>         | <i>Description</i>   |
| <b>Routing Table</b>  | The name of the routing table that stores this route.  |
| <b>Prefix</b>         | The route prefix for each active route displayed.  |
| <b>Protocol</b>       | The protocol used by the route.  |
| <b>Source</b>         | The host source for each active route displayed.   |
| <b>Next hop</b>       | The IP address of the next hop for each active route displayed.                                |
| <b>Label</b>          | The label for each active route displayed.   |
| <b>Security</b>       | The security value for each active route displayed.  |
| <b>Origin VN</b>      | The virtual network from which the route originates.   |
| <b>AS Path</b>        | The AS path for each active route displayed.   |

## Monitor > Infrastructure > Virtual Routers

### IN THIS SECTION

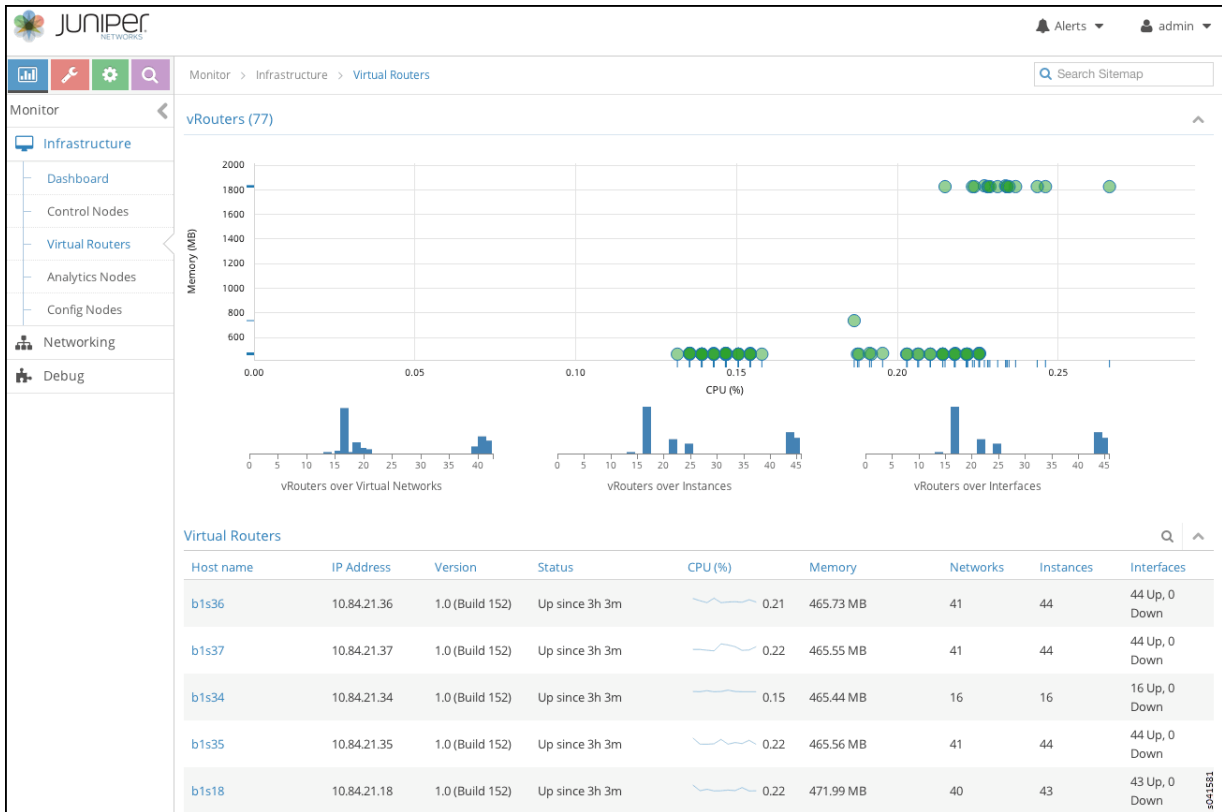
- [Monitor vRouters Summary | 904](#)
- [Monitor Individual vRouters Tabs | 906](#)
- [Monitor Individual vRouter Details Tab | 906](#)
- [Monitor Individual vRouters Interfaces Tab | 908](#)
- [Monitor Individual vRouters Networks Tab | 910](#)
- [Monitor Individual vRouters ACL Tab | 911](#)
- [Monitor Individual vRouters Flows Tab | 913](#)
- [Monitor Individual vRouters Routes Tab | 914](#)
- [Monitor Individual vRouter Console Tab | 915](#)

### Monitor vRouters Summary

Click **Monitor > Infrastructure > Virtual Routers** to view the **vRouters** summary screen. See [Figure 235](#) on page 905.



Figure 235: vRouters Summary



See [Table 71 on page 905](#) for descriptions of the fields on the **vRouters Summary** screen.

Table 71: vRouters Summary Fields

| Field             | Description  |
|-------------------|--|
| <b>Host name</b>  | The name of the vRouter. Click the name of any vRouter to reveal more details. |
| <b>IP Address</b> | The IP address of the vRouter.   |
| <b>Version</b>    | The version of software installed on the system.                               |
| <b>Status</b>     | The current operational status of the vRouter – Up or Down.                    |
| <b>CPU (%)</b>    | The CPU percentage currently in use by the selected vRouter.                   |

Table 71: vRouters Summary Fields *(Continued)*

| Field              | Description  |
|--------------------|--|
| <b>Memory (MB)</b> | The memory currently in use and the total memory available for this vRouter. |
| <b>Networks</b>    | The total number of networks for this vRouter.                               |
| <b>Instances</b>   | The total number of instances for this vRouter.                              |
| <b>Interfaces</b>  | The total number of interfaces for this vRouter.                             |

## Monitor Individual vRouters Tabs

Click the name of any vRouter to view details about performance and activities for that vRouter. Each individual vRouters screen has the following tabs.

- **Details**—similar display of information as on individual control nodes **Details** tab. See [Figure 236 on page 907](#).
- **Console**—similar display of information as on individual control nodes **Console** tab. See [Figure 242 on page 916](#).
- **Interfaces**—details about associated interfaces. See [Figure 237 on page 909](#).
- **Networks**—details about associated networks. See [Figure 238 on page 910](#).
- **ACL**—details about access control lists. See [Figure 239 on page 912](#).
- **Flows**—details about associated traffic flows. See [Figure 240 on page 913](#).
- **Routes**—details about associated routes. See [Figure 241 on page 915](#).

## Monitor Individual vRouter Details Tab

The **Details** tab provides a summary of the status and activity on the selected node, and presents graphical displays of CPU and memory usage; see [Figure 236 on page 907](#). See [Table 72 on page 907](#) for descriptions of the fields on this tab.

Figure 236: Individual vRouters—Details Tab

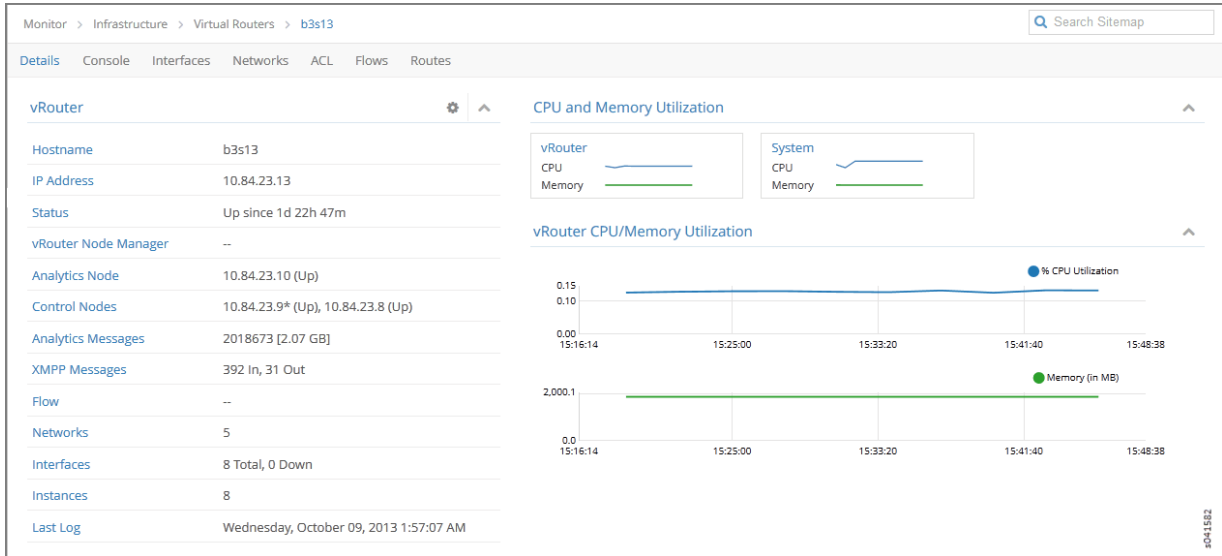


Table 72: vRouters Details Tab Fields

| Field                       | Description   |
|-----------------------------|---|
| <b>Hostname</b>             | The hostname of the vRouter.  |
| <b>IP Address</b>           | The IP address of the selected vRouter.   |
| <b>Status</b>               | The operational status of the vRouter.  |
| <b>vRouter Node Manager</b> | The operational status of the vRouter node manager.                               |
| <b>Analytics Node</b>       | The IP address of the node from which analytics (monitor) information is derived. |
| <b>Control Nodes</b>        | The IP address of the configuration node associated with this vRouter.            |
| <b>Analytics Messages</b>   | The total number of analytics messages in and out from this node.                 |
| <b>XMPP Messages</b>        | The total number of XMPP messages that have gone in and out of this vRouter.      |
| <b>Flow</b>                 | The number of active flows and the total flows for this vRouter.                  |

Table 72: vRouters Details Tab Fields (Continued)

| Field                                 | Description  |
|---------------------------------------|--|
| <b>Networks</b>                       | The number of networks associated with this vRouter.   |
| <b>Interfaces</b>                     | The number of interfaces associated with this vRouter.   |
| <b>Instances</b>                      | The number of instances associated with this vRouter.  |
| <b>Last Log</b>                       | The date and time of the last log message issued about this vRouter.   |
| <b>vRouter CPU/Memory Utilization</b> | Graphs (x, y) displaying CPU and memory utilization averages over time for this vRouter, in comparison to system utilization averages. |

### Monitor Individual vRouters Interfaces Tab

The **Interfaces** tab displays details about the interfaces associated with an individual vRouter. Click the expansion arrow next to any interface name to reveal more details. Use horizontal and vertical scroll bars to access all portions of the screen. See [Figure 237 on page 909](#). See [Table 73 on page 909](#) for descriptions of the fields on the **Interfaces** tab screen.

Figure 237: Individual vRouters—Interfaces Tab

| Name           | Label | Status | Network                  | IP Address     | Floating IP | Instance                             |
|----------------|-------|--------|--------------------------|----------------|-------------|--------------------------------------|
| tap25e5cee3-07 | 18    | Up     | default-domain:demo:vn30 | 192.168.30.247 | None        | 005132fd-0d83-4db7-88c8-bd49d68e9480 |
| tap4d91aab1-f1 | 25    | Up     | default-domain:demo:vn26 | 192.168.26.247 | None        | 65d6c6e9-7a82-43d8-a706-f74d81715920 |
| tap5a8cd9dd-5b | 27    | Up     | default-domain:demo:vn23 | 192.168.23.249 | None        | a159c518-4fb6-402a-ae0d-eb5b4457b551 |
| tap603a5e0b-8b | 16    | Up     | default-domain:demo:vn19 | 192.168.19.247 | None        | fe622580-b0cf-4c6d-89e5-d2065e7e87e4 |
| tap68ad232c-76 | 19    | Up     | default-domain:demo:vn28 | 192.168.28.247 | None        | 91089d89-76b5-46c2-abc9-b9693bcb37ac |

Details :

```

- {
  index: "6",
  name: "tap68ad232c-76",
  uuid: "68ad232c-76d1-4fe2-a200-42182497545e",
  vrf_name: "default-domain:demo:vn28:vn28",
  active: "Active",
  dhcp_service: "Enable",

```

Table 73: vRouters: Interfaces Tab Fields

| Field              | Description   |
|--------------------|---|
| <b>Name</b>        | The name of the interface.  |
| <b>Label</b>       | The label for the interface.                                      |
| <b>Status</b>      | The current status of the interface.                              |
| <b>Network</b>     | The network associated with the interface.                        |
| <b>IP Address</b>  | The IP address of the interface.                                  |
| <b>Floating IP</b> | Displays any floating IP addresses associated with the interface. |

**Table 73: vRouters: Interfaces Tab Fields (Continued)**

| Field           | Description   |
|-----------------|---|
| <b>Instance</b> | The name of any instance associated with the interface. |

## Monitor Individual vRouters Networks Tab

The **Networks** tab displays details about the networks associated with an individual vRouter. Click the expansion arrow at the name of any network to reveal more details. See [Figure 238 on page 910](#). See [Table 74 on page 911](#) for descriptions of the fields on the **Networks** tab screen.

**Figure 238: Individual vRouters—Networks Tab**

Monitor > Infrastructure > Virtual Routers > b1s36 Search Sitemap

Details Console Interfaces **Networks** ACL Flows Routes

**Networks** Q ^

| Name                       | ACLs                                 | VRF                           |
|----------------------------|--------------------------------------|-------------------------------|
| ▶ default-domain:demo:vn24 | a372751f-6497-41e9-b409-fa4ab5ce6b7f | default-domain:demo:vn24:vn24 |
| ▶ default-domain:demo:vn22 | 195af177-0a28-49a1-9cf0-2ceac22af5a1 | default-domain:demo:vn22:vn22 |
| ▶ default-domain:demo:vn30 | 362cce6e-2894-42d6-ba03-3ee98cac8809 | default-domain:demo:vn30:vn30 |
| ▶ default-domain:demo:vn21 | 5918a068-1cd5-4993-9cff-386a807940ca | default-domain:demo:vn21:vn21 |
| ▶ default-domain:demo:vn28 | dd87c461-97c0-4d47-bff0-89040e7d6ab0 | default-domain:demo:vn28:vn28 |
| ▶ default-domain:demo:vn19 | f0465432-6fc0-4fb3-967c-392100617408 | default-domain:demo:vn19:vn19 |
| ▲ default-domain:demo:vn2  | 1c46e7e0-f799-4bc6-ae09-e4654c263aa6 | default-domain:demo:vn2:vn2   |

Details:

```

- {
  name: "default-domain:demo:vn2",
  uuid: "63d08f7a-b342-4892-9171-edab9f4c397f",
  acl_uuid: "1c46e7e0-f799-4bc6-ae09-e4654c263aa6",
  mirror_acl_uuid: - {},
  mirror_cfg_acl_uuid: - {},
  vrf_name: "default-domain:demo:vn2:vn2",
  ipam_data: - {
    list: - {
  
```

f041584

**Table 74: vRouters: Networks Tab Fields**

| Field         | Description   |
|---------------|---|
| <b>Name</b>   | The name of each network associated with this vRouter.                  |
| <b>ACLs</b>   | The name of the access control list associated with the listed network. |
| <b>VRF</b>    | The identifier of the VRF associated with the listed network.           |
| <b>Action</b> | Click the icon to select the action: Edit, Delete                       |

### Monitor Individual vRouters ACL Tab

The **ACL** tab displays details about the access control lists (ACLs) associated with an individual vRouter. Click the expansion arrow next to the UUID of any ACL to reveal more details. See [Figure 239 on page 912](#). See [Table 75 on page 912](#) for descriptions of the fields on the **ACL** tab screen.

Figure 239: Individual vRouters—ACL Tab

Monitor > Infrastructure > Virtual Routers > b1s36

Search Sitemap

Details Console Interfaces Networks **ACL** Flows Routes

ACL

| UUID                                 | Flows | Action | Protocol | Source Network or Prefix | Source Port | Destination Network or Prefix | D |
|--------------------------------------|-------|--------|----------|--------------------------|-------------|-------------------------------|---|
| 195af177-0a28-49a1-9cf0-2ceac22af5a1 | 8     | pass   | any      | -                        | any         | -                             | a |
|                                      |       | pass   | any      | -                        | any         | -                             | a |
|                                      |       | pass   | any      | -                        | any         | -                             | a |
| 1c46e7e0-f799-4bc6-ae09-e4654c263aa6 | 8     | pass   | any      | -                        | any         | -                             | a |

Details:

```

- {
  uuid: "1c46e7e0-f799-4bc6-ae09-e4654c263aa6",
  dynamic_acl: "false",
  entries: - {
    list: - {
      AclEntrySandeshData: - [
        - {
          ace_id: "1",

```

Table 75: vRouters: ACL Tab Fields

| Field                           | Description  |
|---------------------------------|--|
| <b>UUID</b>                     | The universal unique identifier (UUID) associated with the listed ACL.   |
| <b>Flows</b>                    | The flows associated with the listed ACL.                                |
| <b>Action</b>                   | The traffic action defined by the listed ACL.                            |
| <b>Protocol</b>                 | The protocol associated with the listed ACL.                             |
| <b>Source Network or Prefix</b> | The name or prefix of the source network associated with the listed ACL. |
| <b>Source Port</b>              | The source port associated with the listed ACL.                          |



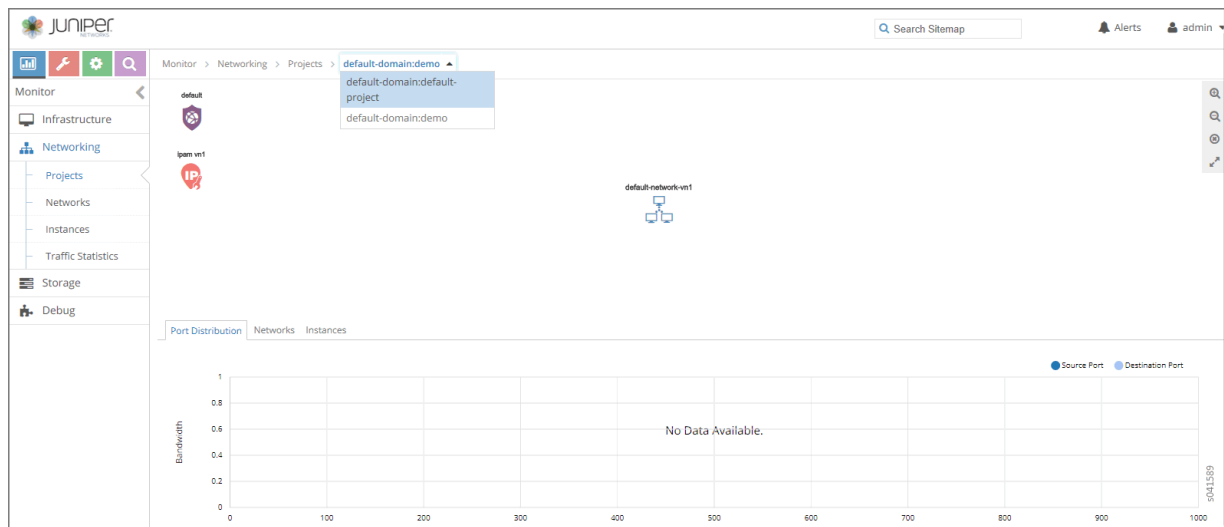
**Table 75: vRouters: ACL Tab Fields (Continued)**

| Field                                | Description   |
|--------------------------------------|---|
| <b>Destination Network or Prefix</b> | The name or prefix of the destination network associated with the listed ACL. |
| <b>Destination Port</b>              | The destination port associated with the listed ACL.                          |
| <b>ACE Id</b>                        | The ACE ID associated with the listed ACL.                                    |

### Monitor Individual vRouters Flows Tab

The **Flows** tab displays details about the flows associated with an individual vRouter. Click the expansion arrow next to any ACL/SG UUID to reveal more details. Use the horizontal and vertical scroll bars to access all portions of the screen. See [Figure 240 on page 913](#). See [Table 76 on page 914](#) for descriptions of the fields on the **Flows** tab screen.

**Figure 240: Individual vRouters—Flows Tab**



**Table 76: vRouters: Flows Tab Fields**

| Field                | Description   |
|----------------------|---|
| <b>ACL UUID</b>      | The default is to show <b>All</b> flows, however, you can select from a drop down list any single flow to view its details. |
| <b>ACL / SG UUID</b> | The universal unique identifier (UUID) associated with the listed ACL or SG.  |
| <b>Protocol</b>      | The protocol associated with the listed flow.   |
| <b>Src Network</b>   | The name of the source network associated with the listed flow.   |
| <b>Src IP</b>        | The source IP address associated with the listed flow.  |
| <b>Src Port</b>      | The source port of the listed flow.   |
| <b>Dest Network</b>  | The name of the destination network associated with the listed flow.  |
| <b>Dest IP</b>       | The destination IP address associated with the listed flow.   |
| <b>Dest Port</b>     | The destination port associated with the listed flow.   |
| <b>Bytes/Pkts</b>    | The number of bytes and packets associated with the listed flow.  |
| <b>Setup Time</b>    | The setup time associated with the listed flow.   |

### Monitor Individual vRouters Routes Tab

The **Routes** tab displays details about unicast and multicast routes in specific VRFs for an individual vRouter. Click the expansion arrow next to the route prefix to reveal more details. See [Figure 241 on page 915](#). See [Table 77 on page 915](#) for descriptions of the fields on the **Routes** tab screen.

Figure 241: Individual vRouters—Routes Tab

Monitor > Infrastructure > Virtual Routers > b1s36

Details Console Interfaces Networks ACL Flows Routes

VRF: default-domain:default-project:ip-fabric:\_\_default\_\_

Show Routes:  Unicast  Multicast

| Prefix          | Next hop ... | Next hop details   |
|-----------------|--------------|--|
| 0.0.0.0 / 0     | arp          | Interface: p2p0p0 Mac: 40:b4:f0:68:20:4e IP: 10.84.21.254              |
| 10.84.21.0 / 24 | resolve      | Source: Local Destination VN: default-domain:default-project:ip-fabric |
| 10.84.21.1 / 32 | arp          | Interface: p2p0p0 Mac: 0:25:90:ab:b0:2c IP: 10.84.21.1                 |
| 10.84.21.2 / 32 | arp          | Interface: p2p0p0 Mac: 0:25:90:ab:b0:38 IP: 10.84.21.2                 |
| 10.84.21.3 / 32 | arp          | Interface: p2p0p0 Mac: 0:25:90:ab:af:ce IP: 10.84.21.3                 |
| 10.84.21.4 / 32 | arp          | Interface: p2p0p0 Mac: 0:25:90:ab:ae:82 IP: 10.84.21.4                 |
| 10.84.21.5 / 32 | arp          | Interface: p2p0p0 Mac: 0:25:90:ab:b0:16 IP: 10.84.21.5                 |

Details:

```

- {
  dispPrefix: "10.84.21.5 / 32",
  path: - {
    nh: - {
      type: "arp",
      ref_count: "1",
      valid: "true",
      policy: "disabled",
      sip: "10.84.21.5",
      vrf: "default-domain:default-project:ip-fabric:__default__",
    }
  }
}

```

9041837

Table 77: vRouters: Routes Tab Fields

| Field                   | Description  |
|-------------------------|--|
| <b>VRF</b>              | Select from a drop down list the virtual routing and forwarding (VRF) to view. |
| <b>Show Routes</b>      | Select to show the route type: <b>Unicast</b> or <b>Multicast</b> .            |
| <b>Prefix</b>           | The IP address prefix of a route.  |
| <b>Next hop</b>         | The next hop method for this route.  |
| <b>Next hop details</b> | The next hop details for this route.   |

## Monitor Individual vRouter Console Tab

Click the **Console** tab for an individual vRouter to display system logging information for a defined time period, with the last 5 minutes of information as the default display. See [Figure 242 on page 916](#). See [Table 78 on page 916](#) for descriptions of the fields on the **Console** tab screen.

Figure 242: Individual vRouter—Console Tab

Table 78: Control Node: Console Tab Fields

| Field               | Description  |
|---------------------|--|
| <b>Time Range</b>   | Select a timeframe for which to review logging information as sent to the console. There are several options, ranging from <b>Last 5 mins</b> through to the <b>Last 24 hrs</b> , plus a <b>Custom</b> time range. |
| <b>From Time</b>    | If you select <b>Custom</b> in <b>Time Range</b> , enter the start time.   |
| <b>To Time</b>      | If you select <b>Custom</b> in <b>Time Range</b> , enter the end time.   |
| <b>Log Category</b> | Select a log category to display: <ul style="list-style-type: none"> <li>• All</li> <li>• _default_</li> <li>• XMPP</li> <li>• TCP</li> </ul>  |
| <b>Log Type</b>     | Select a log type to display.  |

Table 78: Control Node: Console Tab Fields (*Continued*)

| Field               | Description  |
|---------------------|--|
| <b>Log Level</b>    | Select a log severity level to display: <ul style="list-style-type: none"> <li>• SYS_EMERG</li> <li>• SYS_ALERT</li> <li>• SYS_CRIT</li> <li>• SYS_ERR</li> <li>• SYS_WARN</li> <li>• SYS_NOTICE</li> <li>• SYS_INFO</li> <li>• SYS_DEBUG</li> </ul>                                     |
| <b>Limit</b>        | Select from a list an amount to limit the number of messages displayed: <ul style="list-style-type: none"> <li>• No Limit</li> <li>• Limit 10 messages</li> <li>• Limit 50 messages</li> <li>• Limit 100 messages</li> <li>• Limit 200 messages</li> <li>• Limit 500 messages</li> </ul> |
| <b>Auto Refresh</b> | Click the check box to automatically refresh the display if more messages occur.   |
| <b>Display Logs</b> | Click this button to refresh the display if you change the display criteria.   |
| <b>Reset</b>        | Click this button to clear any selected display criteria and reset all criteria to their default settings.   |
| <i>Columns</i>      |  |

Table 78: Control Node: Console Tab Fields (*Continued*)

| Field           | Description   |
|-----------------|---|
| <b>Time</b>     | This column lists the time received for each log message displayed. |
| <b>Category</b> | This column lists the log category for each log message displayed.  |
| <b>Log Type</b> | This column lists the log type for each log message displayed.      |
| <b>Log</b>      | This column lists the log message for each log displayed.           |

## Monitor > Infrastructure > Analytics Nodes

### IN THIS SECTION

- [Monitor Analytics Nodes | 918](#)
- [Monitor Analytics Individual Node Details Tab | 920](#)
- [Monitor Analytics Individual Node Generators Tab | 921](#)
- [Monitor Analytics Individual Node QE Queries Tab | 922](#)
- [Monitor Analytics Individual Node Console Tab | 923](#)

Select **Monitor > Infrastructure > Analytics Nodes** to view the console logs, generators, and query expansion (QE) queries of the analytics nodes.

### Monitor Analytics Nodes

Select **Monitor > Infrastructure > Analytics Nodes** to view a summary of activities for the analytics nodes; see [Figure 243 on page 919](#). See [Table 79 on page 919](#) for descriptions of the fields on the analytics summary.

Figure 243: Analytics Nodes Summary

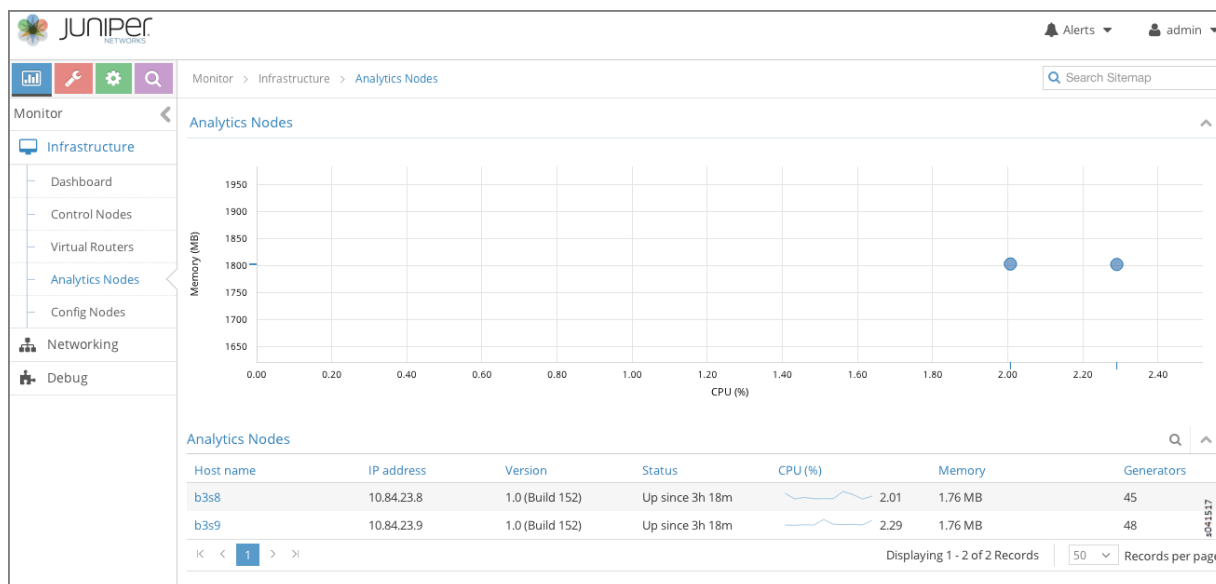


Table 79: Fields on Analytics Nodes Summary

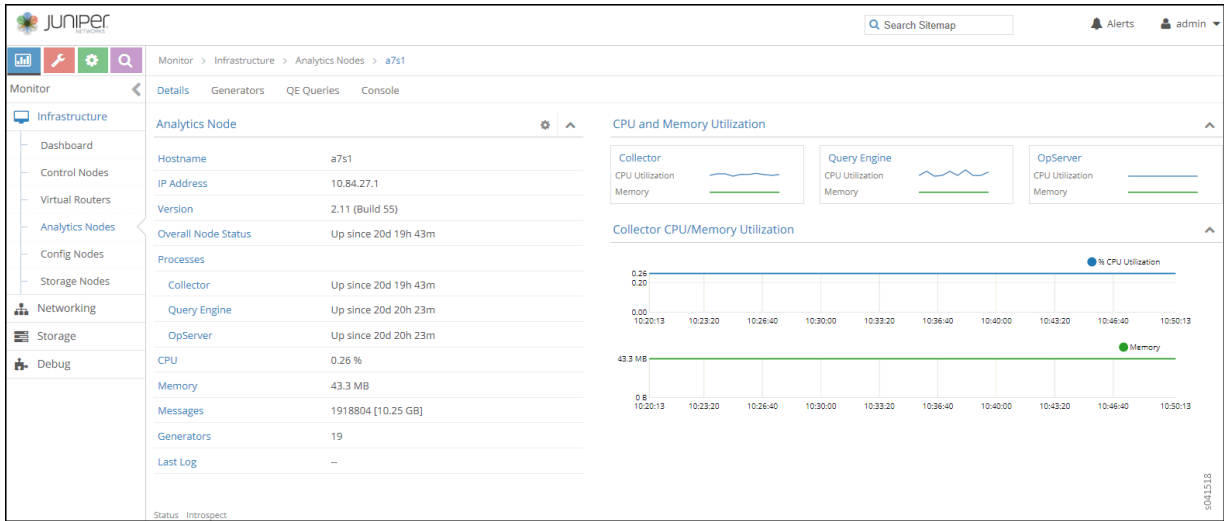
| Field             | Description   |
|-------------------|---|
| <b>Host name</b>  | The name of this node.  |
| <b>IP address</b> | The IP address of this node.  |
| <b>Version</b>    | The version of software installed on the system.  |
| <b>Status</b>     | The current operational status of the node – Up or Down – and the length of time it is in that state. |
| <b>CPU (%)</b>    | The average CPU percentage usage for this node.   |
| <b>Memory</b>     | The average memory usage for this node.   |
| <b>Generators</b> | The total number of generators for this node.   |

## Monitor Analytics Individual Node Details Tab

Click the name of any analytics node displayed on the analytics summary to view the **Details** tab for that node. See [Figure 244 on page 920](#).

See [Table 80 on page 920](#) for descriptions of the fields on this screen.

**Figure 244: Monitor Analytics Individual Node Details Tab**



**Table 80: Monitor Analytics Individual Node Details Tab Fields**

| Field                      | Description   |
|----------------------------|---|
| <b>Hostname</b>            | The name of this node.  |
| <b>IP Address</b>          | The IP address of this node.  |
| <b>Version</b>             | The installed version of the software.  |
| <b>Overall Node Status</b> | The current operational status of the node – Up or Down – and the length of time in this state. |
| <b>Processes</b>           | The current status of each analytics process, including Collector, Query Engine, and OpServer.  |



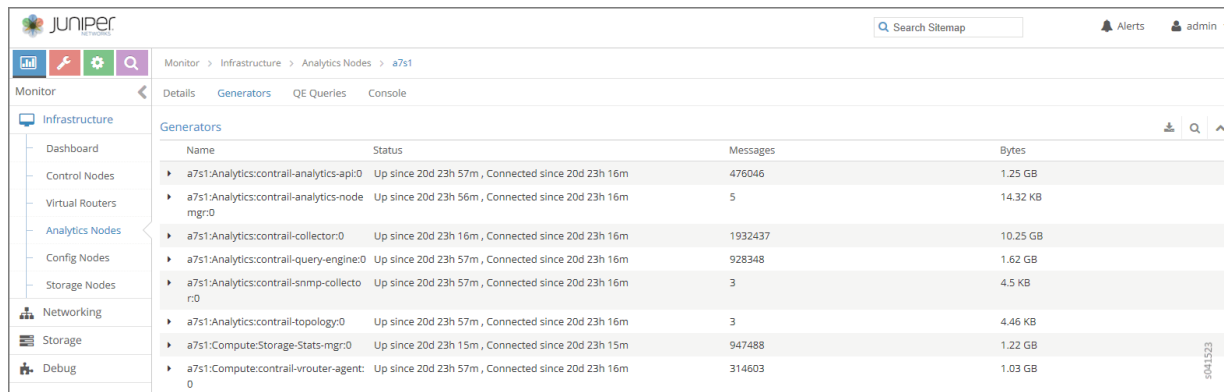
Table 80: Monitor Analytics Individual Node Details Tab Fields (Continued)

| Field             | Description   |
|-------------------|---|
| <b>CPU (%)</b>    | The average CPU percentage usage for this node.                   |
| <b>Memory</b>     | The average memory usage of this node.                            |
| <b>Messages</b>   | The total number of messages for this node.                       |
| <b>Generators</b> | The total number of generators associated with this node.         |
| <b>Last Log</b>   | The date and time of the last log message issued about this node. |

## Monitor Analytics Individual Node Generators Tab

The **Generators** tab displays information about the generators for an individual analytics node; see [Figure 245 on page 921](#). Click the expansion arrow next to any generator name to reveal more details. See [Table 81 on page 922](#) for descriptions of the fields on the **Peers** tab screen.

Figure 245: Individual Analytics Node—Generators Tab



| Name   | Status  | Messages | Bytes    |
|--|---|----------|----------|
| a7s1-Analytics:contrail-analytics-api:0      | Up since 20d 23h 57m, Connected since 20d 23h 16m | 476046   | 1.25 GB  |
| a7s1-Analytics:contrail-analytics-node mgr:0 | Up since 20d 23h 56m, Connected since 20d 23h 16m | 5        | 14.32 KB |
| a7s1-Analytics:contrail-collector:0          | Up since 20d 23h 16m, Connected since 20d 23h 16m | 1932437  | 10.25 GB |
| a7s1-Analytics:contrail-query-engine:0       | Up since 20d 23h 57m, Connected since 20d 23h 16m | 928348   | 1.62 GB  |
| a7s1-Analytics:contrail-snmp-collecto r:0    | Up since 20d 23h 57m, Connected since 20d 23h 16m | 3        | 4.5 KB   |
| a7s1-Analytics:contrail-topology:0           | Up since 20d 23h 57m, Connected since 20d 23h 16m | 3        | 4.46 KB  |
| a7s1:Compute:Storage-Stats-mgr:0             | Up since 20d 23h 15m, Connected since 20d 23h 15m | 947488   | 1.22 GB  |
| a7s1:Compute:contrail-vrouter-agent: 0       | Up since 20d 23h 57m, Connected since 20d 23h 16m | 314603   | 1.03 GB  |

**Table 81: Monitor Analytics Individual Node Generators Tab Fields**

| Field           | Description  |
|-----------------|--|
| <b>Name</b>     | The host name of the generator.  |
| <b>Status</b>   | The current status of the peer— Up or Down — and the length of time in that state. |
| <b>Messages</b> | The number of messages sent and received from this peer.                           |
| <b>Bytes</b>    | The total message size in bytes.   |

### Monitor Analytics Individual Node QE Queries Tab

The **QE Queries** tab displays the number of query expansion (QE) messages that are in the queue for this analytics node. See [Figure 246 on page 922](#).

See [Table 82 on page 922](#) for descriptions of the fields on the **QE Queries** tab screen.

**Figure 246: Individual Analytics Node—QE QueriesTab**

| Enqueue Time             | Query | Progress |
|--------------------------|-------|----------|
| No QE Queries to display |       |          |

**Table 82: Analytics Node QE Queries Tab Fields**

| Field               | Description  |
|---------------------|--|
| <b>Enqueue Time</b> | The length of time this message has been in the queue waiting to be delivered. |
| <b>Query</b>        | The query message.   |

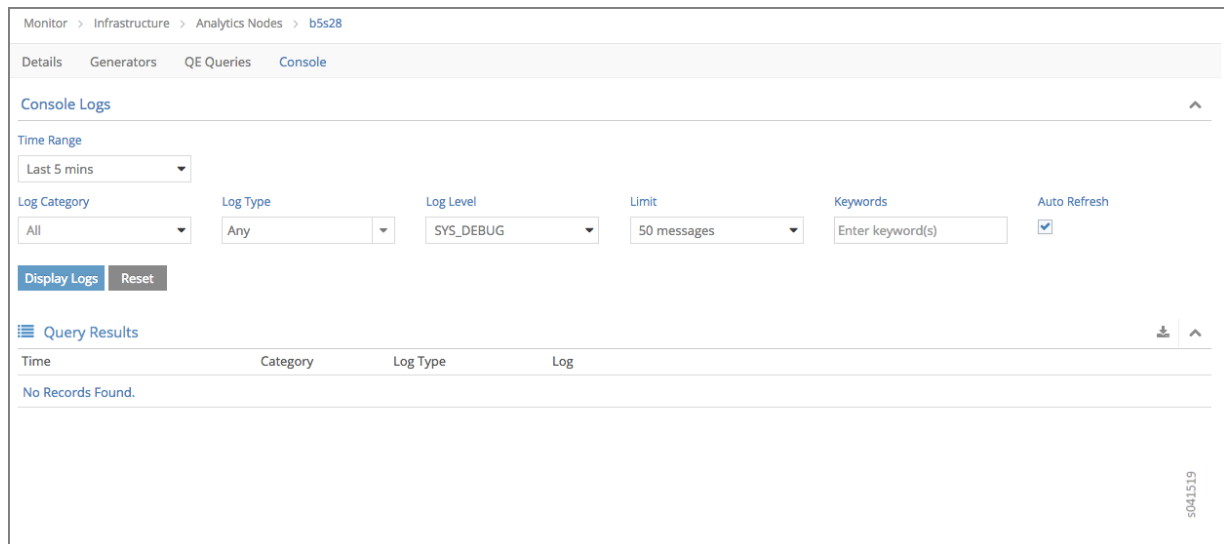
**Table 82: Analytics Node QE Queries Tab Fields (Continued)**

| Field               | Description                                       |
|---------------------|---|
| <b>Progress (%)</b> | The percentage progress for the message delivery. |

## Monitor Analytics Individual Node Console Tab

Click the **Console** tab for an individual analytics node to display system logging information for a defined time period. See [Figure 247 on page 923](#). See [Table 83 on page 923](#) for descriptions of the fields on the **Console** tab screen.

**Figure 247: Analytics Individual Node—Console Tab**



**Table 83: Monitor Analytics Individual Node Console Tab Fields**

| Field             | Description  |
|-------------------|--|
| <b>Time Range</b> | Select a timeframe for which to review logging information as sent to the console. There are 11 options, ranging from the <b>Last 5 mins</b> through to the <b>Last 24 hrs</b> . The default display is for the <b>Last 5 mins</b> . |

Table 83: Monitor Analytics Individual Node Console Tab Fields (Continued)

| Field               | Description  |
|---------------------|--|
| <b>Log Category</b> | Select a log category to display: <ol style="list-style-type: none"> <li>1. All</li> <li>2. _default_</li> <li>3. XMPP</li> <li>4. TCP</li> </ol>  |
| <b>Log Type</b>     | Select a log type to display.  |
| <b>Log Level</b>    | Select a log severity level to display: <ol style="list-style-type: none"> <li>1. SYS_EMERG</li> <li>2. SYS_ALERT</li> <li>3. SYS_CRIT</li> <li>4. SYS_ERR</li> <li>5. SYS_WARN</li> <li>6. SYS_NOTICE</li> <li>7. SYS_INFO</li> <li>8. SYS_DEBUG</li> </ol> |
| <b>Keywords</b>     | Enter any text string to search for and display logs containing that string.   |

Table 83: Monitor Analytics Individual Node Console Tab Fields *(Continued)*

| Field               | Description  |
|---------------------|--|
| (Limit field)       | Select the number of messages to display: <ol style="list-style-type: none"> <li>1. No Limit</li> <li>2. Limit 10 messages</li> <li>3. Limit 50 messages</li> <li>4. Limit 100 messages</li> <li>5. Limit 200 messages</li> <li>6. Limit 500 messages</li> </ol> |
| <b>Auto Refresh</b> | Click the check box to automatically refresh the display if more messages occur.   |
| <b>Display Logs</b> | Click this button to refresh the display if you change the display criteria.   |
| <b>Reset</b>        | Click this button to clear any selected display criteria and reset all criteria to their default settings.   |
| <b>Time</b>         | This column lists the time received for each log message displayed.  |
| <b>Category</b>     | This column lists the log category for each log message displayed.   |
| <b>Log Type</b>     | This column lists the log type for each log message displayed.   |
| <b>Log</b>          | This column lists the log message for each log displayed.  |

## Monitor > Infrastructure > Config Nodes

### IN THIS SECTION

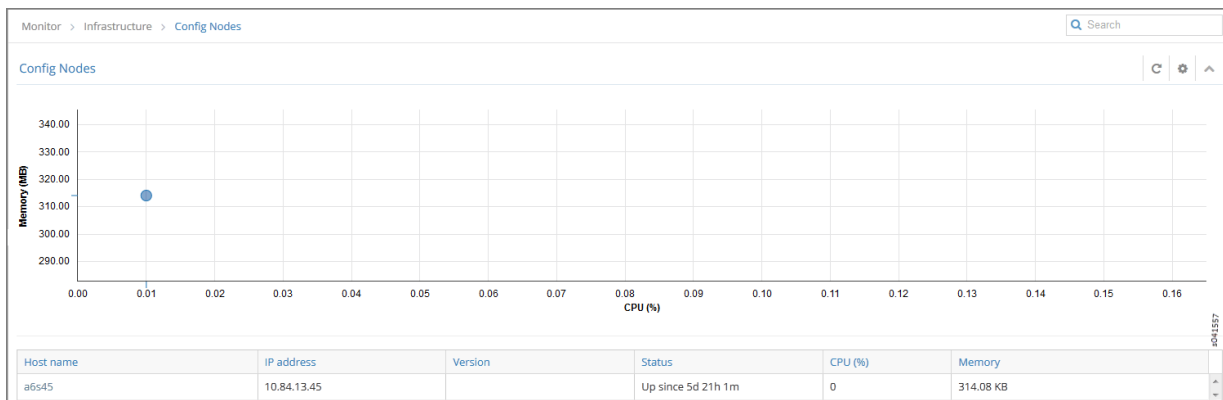
- Monitor Config Nodes | 926
- Monitor Individual Config Node Details | 927
- Monitor Individual Config Node Console | 928

Select **Monitor > Infrastructure > Config Nodes** to view the information about the system config nodes.

### Monitor Config Nodes

Select **Monitor > Infrastructure > Config Nodes** to view a summary of activities for the analytics nodes. See [Figure 248 on page 926](#).

**Figure 248: Config Nodes Summary**



[Table 84 on page 926](#) describes the fields in the Config Nodes summary.

**Table 84: Config Nodes Summary Fields**

| Field            | Description            |
|------------------|------------------------|
| <b>Host name</b> | The name of this node. |

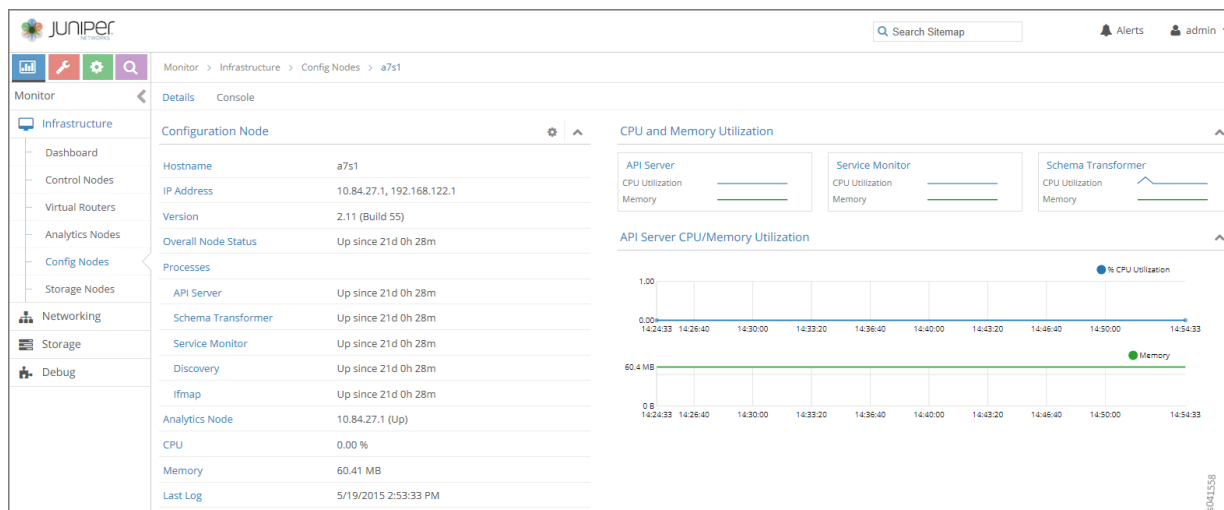
**Table 84: Config Nodes Summary Fields (Continued)**

| Field             | Description   |
|-------------------|---|
| <b>IP address</b> | The IP address of this node.  |
| <b>Version</b>    | The version of software installed on the system.  |
| <b>Status</b>     | The current operational status of the node – Up or Down – and the length of time it is in that state. |
| <b>CPU (%)</b>    | The average CPU percentage usage for this node.   |
| <b>Memory</b>     | The average memory usage for this node.   |

## Monitor Individual Config Node Details

Click the name of any config node displayed on the config nodes summary to view the **Details** tab for that node; see [Figure 249 on page 927](#).

**Figure 249: Individual Config Nodes— Details Tab**



[Table 85 on page 928](#) describes the fields on the Details screen.

**Table 85: Individual Config Nodes— Details Tab Fields**

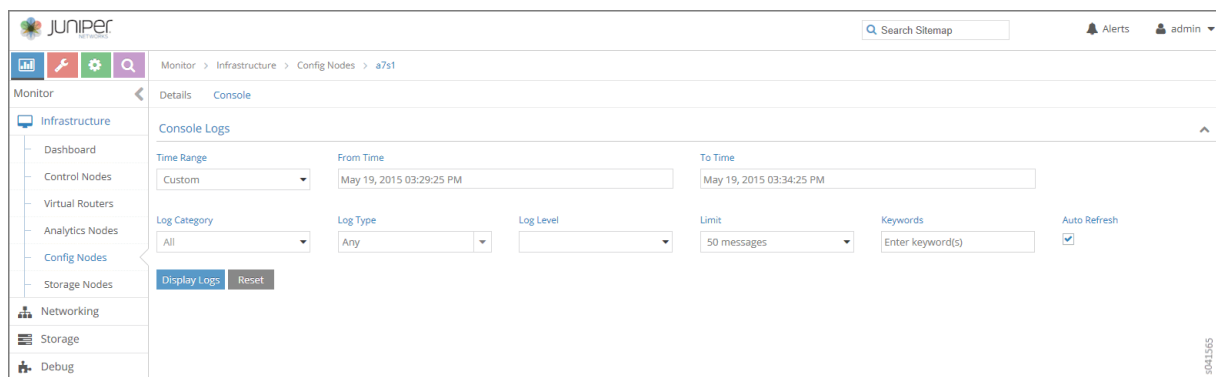
| Field                      | Description  |
|----------------------------|--|
| <b>Hostname</b>            | The name of the config node.   |
| <b>IP Address</b>          | The IP address of this node.   |
| <b>Version</b>             | The installed version of the software.   |
| <b>Overall Node Status</b> | The current operational status of the node – Up or Down – and the length of time it is in this state.  |
| <b>Processes</b>           | The current operational status of the processes associated with the config node, including AI Server, Schema Transformer, Service Monitor, and the like. |
| <b>Analytics Node</b>      | The analytics node associated with this node.  |
| <b>CPU (%)</b>             | The average CPU percentage usage for this node.  |
| <b>Memory</b>              | The average memory usage by this node.   |

### Monitor Individual Config Node Console

Click the **Console** tab for an individual config node to display system logging information for a defined time period. See [Figure 250 on page 929](#).



Figure 250: Individual Config Node—Console Tab



See [Table 86 on page 929](#) for descriptions of the fields on the **Console** tab screen.

Table 86: Individual Config Node-Console Tab Fields

| Field               | Description   |
|---------------------|---|
| <b>Time Range</b>   | Select a timeframe for which to review logging information as sent to the console. Use the drop down calendar in the fields From Time and To Time to select the date and times to include in the time range for viewing.  |
| <b>Log Category</b> | Select from the drop down menu a log category to display. The option to view All is also available.   |
| <b>Log Type</b>     | Select a log type to display.   |
| <b>Log Level</b>    | Select a log severity level to display:   |
| <b>Limit</b>        | Select from a list an amount to limit the number of messages displayed: <ol style="list-style-type: none"> <li>1. All</li> <li>2. Limit 10 messages</li> <li>3. Limit 50 messages</li> <li>4. Limit 100 messages</li> <li>5. Limit 200 messages</li> <li>6. Limit 500 messages</li> </ol> |

Table 86: Individual Config Node-Console Tab Fields (*Continued*)

| Field               | Description  |
|---------------------|--|
| <b>Keywords</b>     | Enter any key words by which to filter the log messages displayed.   |
| <b>Auto Refresh</b> | Click the check box to automatically refresh the display if more messages occur.                           |
| <b>Display Logs</b> | Click this button to refresh the display if you change the display criteria.                               |
| <b>Reset</b>        | Click this button to clear any selected display criteria and reset all criteria to their default settings. |

## Monitor > Networking

### IN THIS SECTION

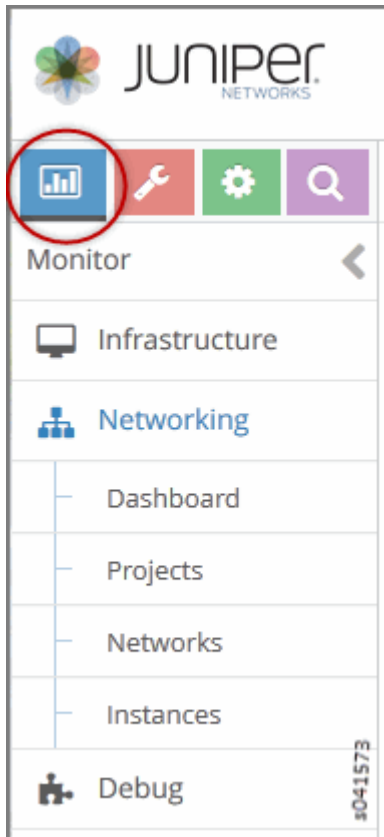
- [Monitor > Networking Menu Options | 930](#)
- [Monitor -> Networking -> Dashboard | 931](#)
- [Monitor > Networking > Projects | 933](#)
- [Monitor Projects Detail | 934](#)
- [Monitor > Networking > Networks | 937](#)

The **Monitor -> Networking** pages give an overview of the networking traffic statistics and health of domains, projects within domains, virtual networks within projects, and virtual machines within virtual networks.

### Monitor > Networking Menu Options

[Figure 251 on page 931](#) shows the menu options available under **Monitor > Networking**.

Figure 251: Monitor Networking Menu Options



### Monitor -> Networking -> Dashboard

Select **Monitor -> Networking -> Dashboard** to gain insight into usage statistics for domains, virtual networks, projects, and virtual machines. When you select this option, the Traffic Statistics for Domain window is displayed as shown in [Figure 252 on page 932](#).

Figure 252: Traffic Statistics for Domain Window

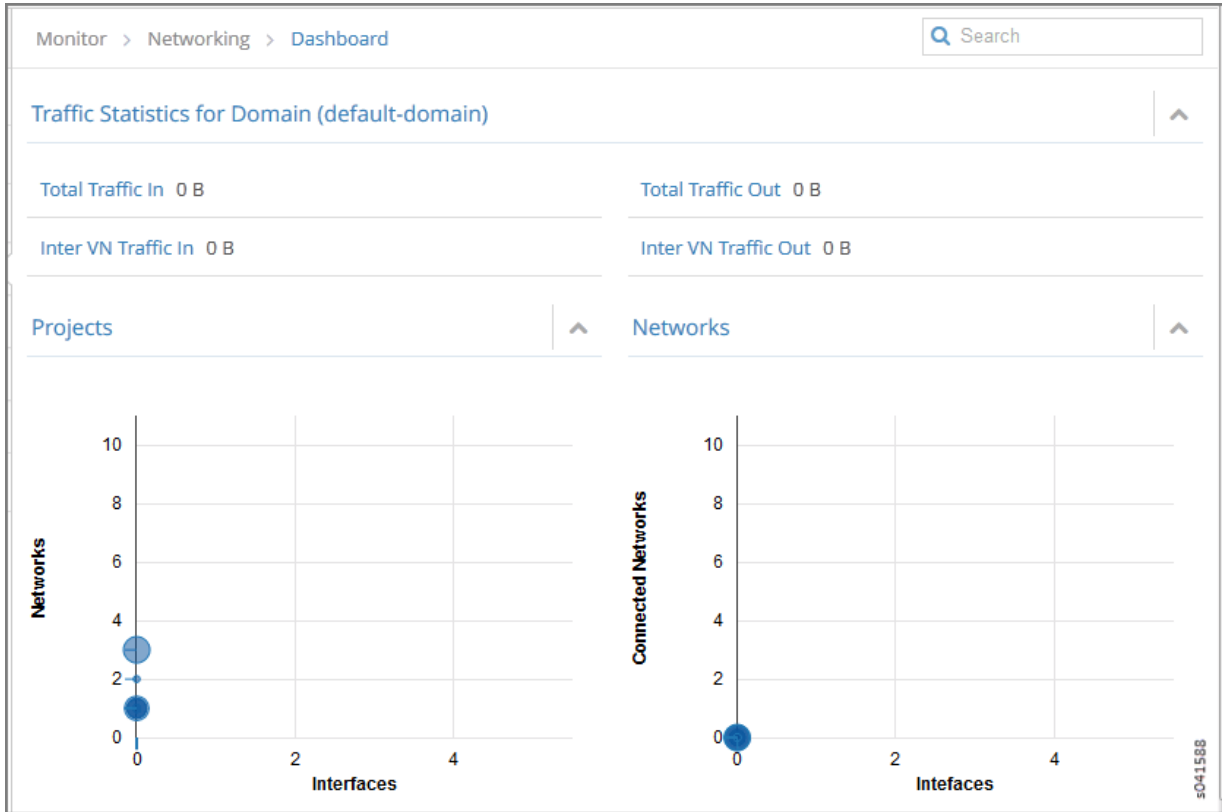


Table 87 on page 932 describes the fields in the Traffic Statistics for Domain window.

Table 87: Projects Summary Fields

| Field                       | Description   |
|-----------------------------|---|
| <b>Total Traffic In</b>     | The volume of traffic into this domain                          |
| <b>Total Traffic Out</b>    | The volume of traffic out of this domain.                       |
| <b>Inter VN Traffic In</b>  | The volume of inter-virtual network traffic into this domain.   |
| <b>Inter VN Traffic Out</b> | The volume of inter-virtual network traffic out of this domain. |

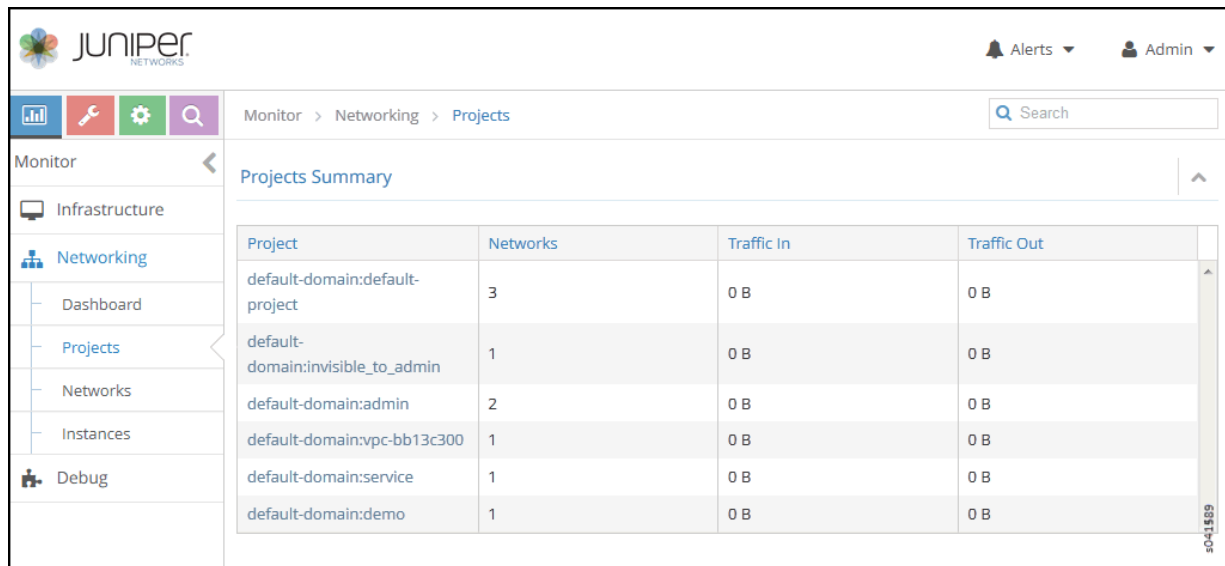
Table 87: Projects Summary Fields (Continued)

| Field           | Description   |
|-----------------|---|
| <b>Projects</b> | This chart displays the networks and interfaces for projects with the most throughput over the past 30 minutes. Click <b>Projects</b> then select <b>Monitor &gt; Networking &gt; Projects</b> , to display more detailed statistics. |
| <b>Networks</b> | This chart displays the networks for projects with the most throughput over the past 30 minutes. Click <b>Networks</b> then select <b>Monitor &gt; Networking &gt; Networks</b> , to display more detailed statistics.                |

## Monitor > Networking > Projects

Select **Monitor > Networking > Projects** to see information about projects in the system. See [Figure 253 on page 933](#).

Figure 253: Monitor &gt; Networking &gt; Projects



| Project                           | Networks | Traffic In | Traffic Out |
|-----------------------------------|----------|------------|-------------|
| default-domain:default-project    | 3        | 0 B        | 0 B         |
| default-domain:invisible_to_admin | 1        | 0 B        | 0 B         |
| default-domain:admin              | 2        | 0 B        | 0 B         |
| default-domain:vpc-bb13c300       | 1        | 0 B        | 0 B         |
| default-domain:service            | 1        | 0 B        | 0 B         |
| default-domain:demo               | 1        | 0 B        | 0 B         |

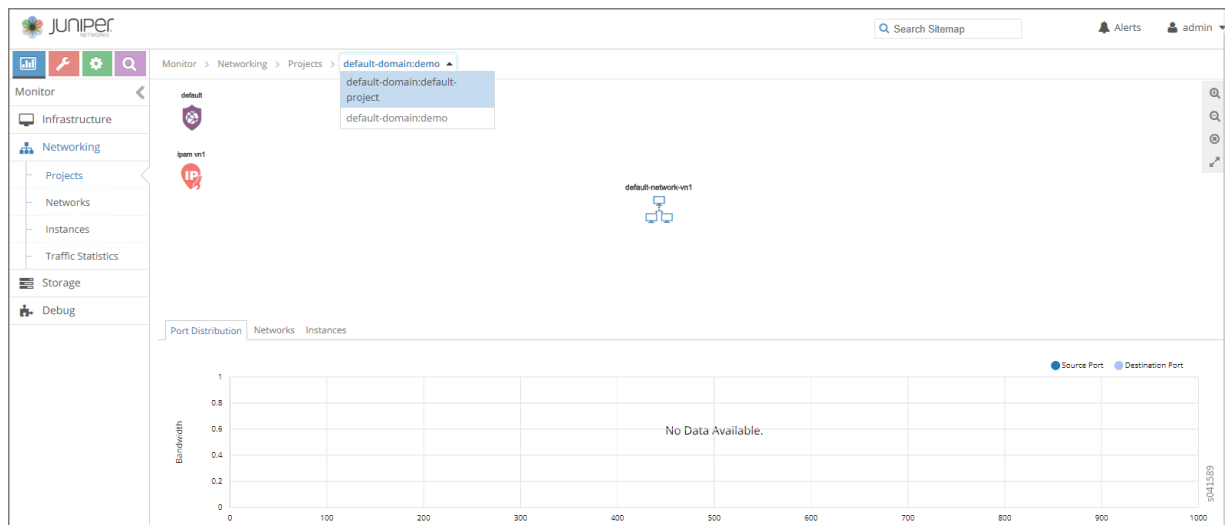
See [Table 88 on page 934](#) for descriptions of the fields on this screen.

**Table 88: Projects Summary Fields**

| Field              | Description  |
|--------------------|--|
| <b>Projects</b>    | The name of the project. You can click the name to access details about connectivity for this project. |
| <b>Networks</b>    | The volume of inter-virtual network traffic out of this domain.  |
| <b>Traffic In</b>  | The volume of traffic into this domain.  |
| <b>Traffic Out</b> | The volume of traffic out of this domain.  |

## Monitor Projects Detail

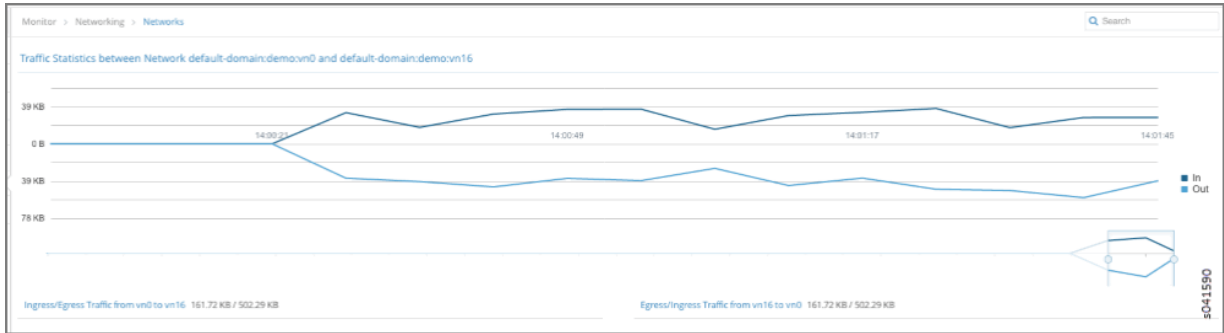
You can click any of the projects listed on the Projects Summary to get details about connectivity, source and destination port distribution, and instances. When you click an individual project, the Summary tab for Connectivity Details is displayed as shown in [Figure 254 on page 934](#). Hover over any of the connections to get more details.

**Figure 254: Monitor Projects Connectivity Details**

In the Connectivity Details window you can click the links between the virtual networks to view the traffic statistics between the virtual networks.

The Traffic Statistics information is also available when you select **Monitor > Networking > Networks** as shown in [Figure 255 on page 935](#).

**Figure 255: Traffic Statistics Between Networks**



In the Connectivity Details window you can click the Instances tab to get a summary of details for each of the instances in this project.

**Figure 256: Projects Instances Summary**

|   | Instance | Virtual Network            | Interfaces | vRouter | IP Address                             | Floating IP | Traffic (In/Out)      |
|---|----------|----------------------------|------------|---------|--|-------------|-----------------------|
| ▶ | out      | default-domain:admin:right | 1          | hp1     | 2.2.2.252                              |             | 129.87 KB / 119.83 KB |
| ▶ | NAT1_1   | default-domain:admin:right | 1          | hp1     | 2.2.2.253<br>250.250.1.253<br>(1 more) |             | 3.69 MB / 1.15 MB     |
| ▶ | in       | default-domain:admin:left  | 1          | hp1     | 1.1.1.252                              |             | 132.75 KB / 122.02 KB |

See Table 3 for a description of the fields on this screen.

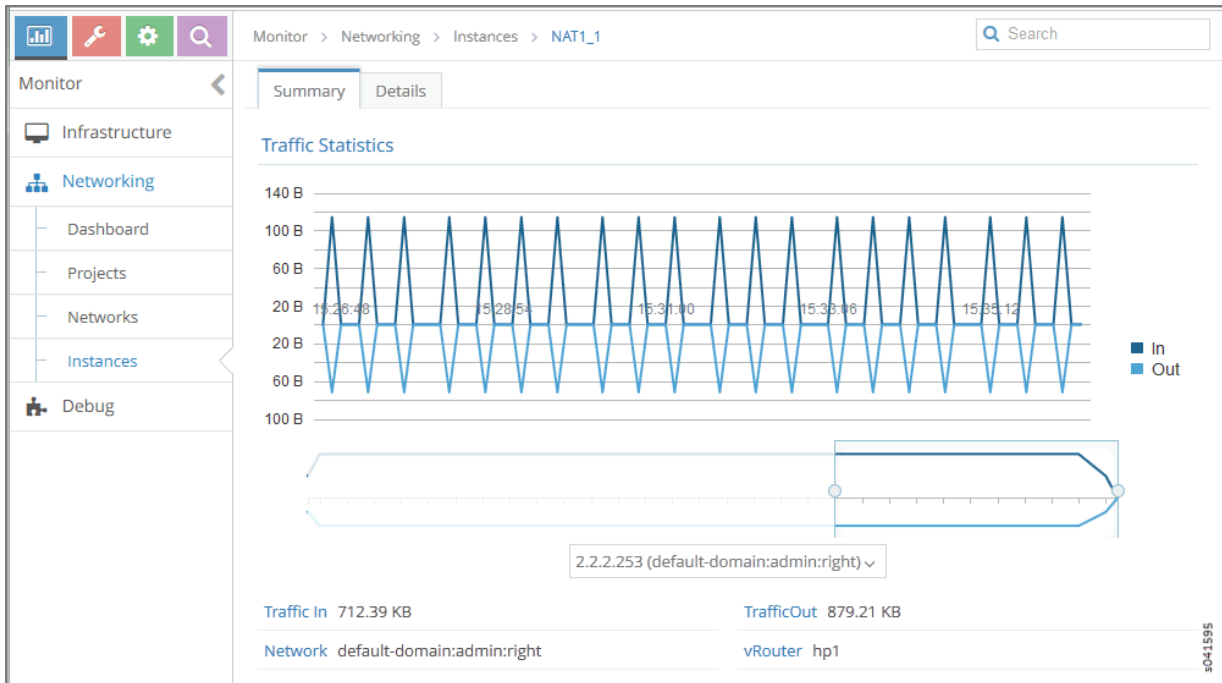
Table 89: Projects Instances Summary Fields

| Field                   | Description   |
|-------------------------|---|
| <b>Instance</b>         | The name of the instance. Click the name then select <b>Monitor &gt; Networking &gt; Instances</b> to display details about the traffic statistics for this instance. |
| <b>Virtual Network</b>  | The virtual network associated with this instance.  |
| <b>Interfaces</b>       | The number of interfaces associated with this instance.   |
| <b>vRouter</b>          | The name of the vRouter associated with this instance.  |
| <b>IP Address</b>       | Any IP addresses associated with this instance.   |
| <b>Floating IP</b>      | Any floating IP addresses associated with this instance.  |
| <b>Traffic (In/Out)</b> | The volume of traffic in KB or MB that is passing in and out of this instance.  |

Select **Monitor > Networking > Instances** to display instance traffic statistics as shown in [Figure 257](#) on [page 937](#).



Figure 257: Instance Traffic Statistics



## Monitor > Networking > Networks

Select **Monitor > Networking > Networks** to view a summary of the virtual networks in your system. See [Figure 258 on page 937](#).

Figure 258: Network Summary

Monitor > Networking > Networks

Networks Summary

| Network  | Instances | Traffic (In/Out)<br>(Last 1 hr) | Throughput (In/Out) |
|--|-----------|---------------------------------|---------------------|
| default-domain:default-project:_link_local_            | 0         | 0 B / 0 B                       | 0 bps / 0 bps       |
| default-domain:default-project:default-virtual-network | 0         | 0 B / 0 B                       | 0 bps / 0 bps       |
| default-domain:default-project:ip-fabric               | 0         | 0 B / 0 B                       | 0 bps / 0 bps       |
| default-domain:demo:default-network-vn1                | 0         | 0 B / 0 B                       | 0 bps / 0 bps       |

Ingress Flows 0  
Egress Flows 0  
ACL Rules 2  
Interfaces 0  
Total Traffic(In/Out) -/-

Total: 4 records 50 Records

Page 1 of 1

5041623

**Table 90: Network Summary Fields**

| Field                      | Description  |
|----------------------------|--|
| <b>Network</b>             | The domain and network name of the virtual network. Click the arrow next to the name to display more information about the network, including the number of ingress and egress flows, the number of ACL rules, the number of interfaces, and the total traffic in and out. |
| <b>Instances</b>           | The number of instances launched in this network.  |
| <b>Traffic (In/Out)</b>    | The volume of inter-virtual network traffic in and out of this network.  |
| <b>Throughput (In/Out)</b> | The throughput of inter-virtual network traffic in and out of this network.  |

At **Monitor > Networking > Networks** you can click on the name of any of the listed networks to get details about the network connectivity, traffic statistics, port distribution, instances, and other details, by clicking the tabs across the top of the page.

[Figure 259 on page 938](#) shows the **Summary** tab for an individual network, which displays connectivity details and traffic statistics for the selected network.

**Figure 259: Individual Network Connectivity Details—Summary Tab**

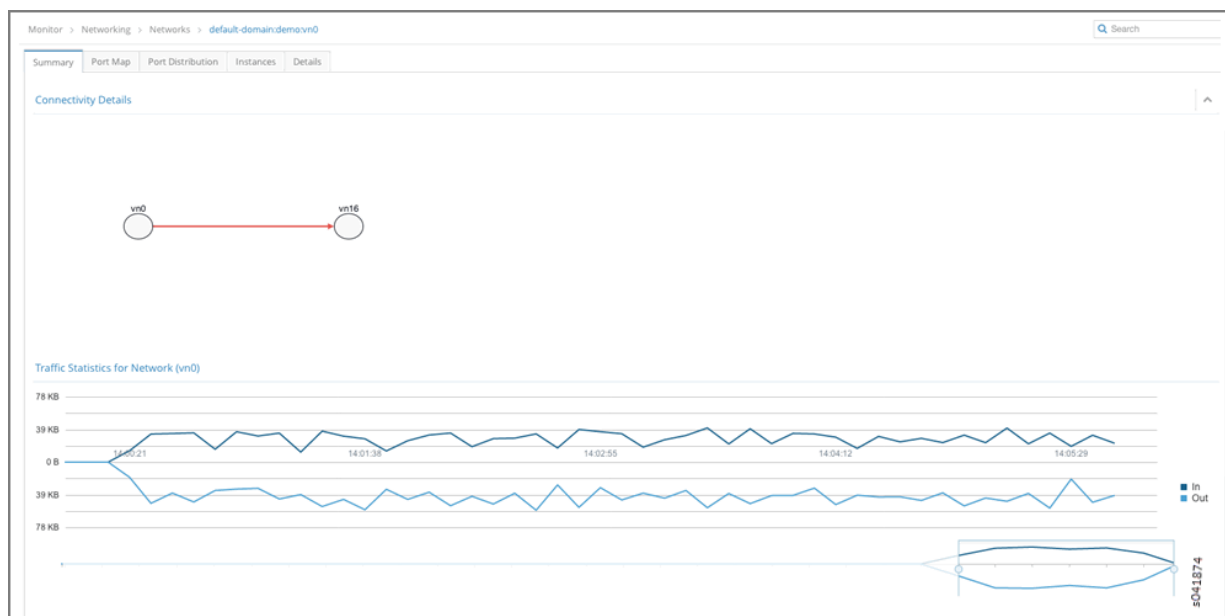


Figure 260 on page 939 shows the **Port Map** tab for an individual network, which displays the relative distribution of traffic for this network by protocol, by port.

**Figure 260: Individual Network-- Port Map Tab**

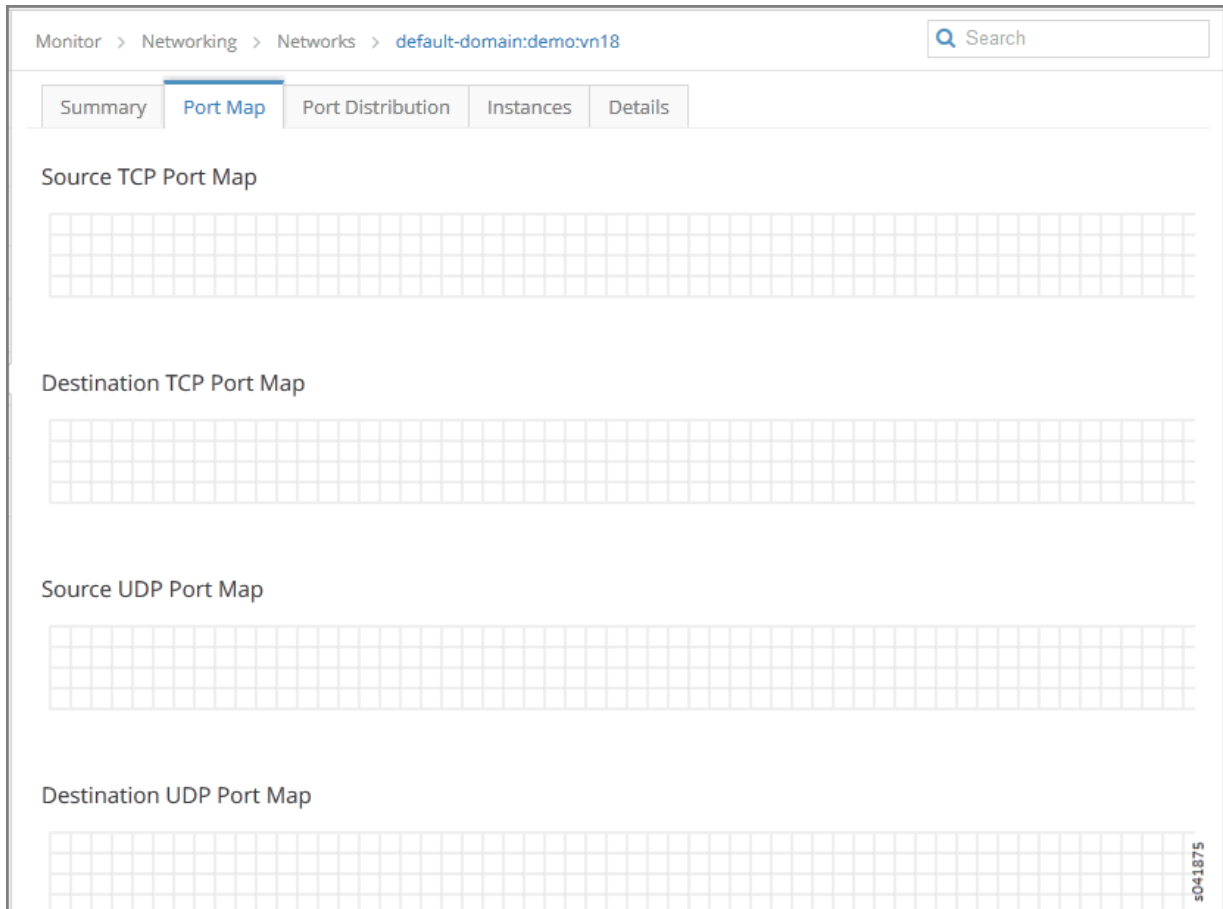


Figure 261 on page 940 shows the **Port Distribution** tab for an individual network, which displays the relative distribution of traffic in and out by source port and destination port.

Figure 261: Individual Network-- Port Distribution Tab

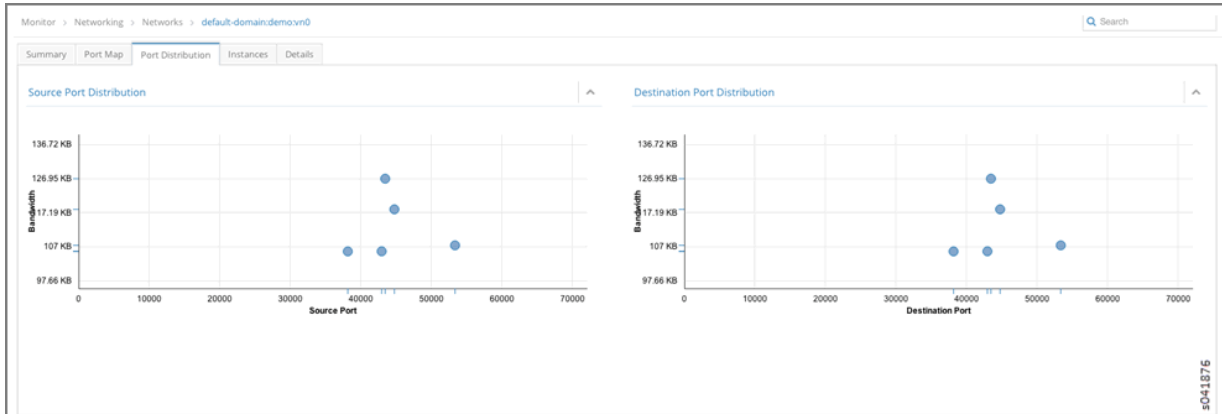


Figure 262 on page 941 shows the **Instances** tab for an individual network, which displays details for each instance associated with this network, including the number of interfaces, the associated vRouter, the instance IP address, and the volume of traffic in and out.

Additionally, you can click the arrow near the instance name to reveal even more details about the instance—the interfaces and their addresses, UUID, CPU (usage), and memory used of the total amount available.

Figure 262: Individual Network Instances Tab

Monitor > Networking > Networks > default-domain:demo:vn18 Search

Summary | Port Map | Port Distribution | **Instances** | Details

### Instances Summary

| Instance  | Interfaces | vRouter | IP Address     | Floating IP | Traffic (In/Out)   |
|---|------------|---------|----------------|-------------|--------------------|
| ▶ vn18_vm-b342ca93-9acd-4275-acb8-df7b5843884c  | 1          | b1s29   | 192.168.18.225 |             | 1.13 KB / 712.00 B |
| ▲ vn18_vm-22a42bf6-fccc-4db3-b5ac-80082bbebfef  | 1          | b1s42   | 192.168.18.236 |             | 1.13 KB / 712.00 B |
| <p><b>Interfaces</b> IP Address: 192.168.18.236 Label: 17 Mac Address: 02:e9:94:e7:0e:56 Network: default-domain:demo:vn18 Traffic (In/Out): 1.13 KB/712.00 B</p> <p><b>UUID</b> 22a42bf6-fccc-4db3-b5ac-80082bbebfef</p> <p><b>CPU</b> 0.01</p> <p><b>Memory (Used/Total)</b> 1.23 GB / 15.63 GB</p> |            |         |                |             |                    |
| ▶ vn18_vm-f676567a-826f-4e9d-9a81-b4649b7fcde2  | 1          | b1s15   | 192.168.18.235 |             | 1.13 KB / 712.00 B |

f041877

Figure 263 on page 942 shows the **Details** tab for an individual network, which displays the code used to define this network --the User Virtual Environment (UVE) code.

Figure 263: Individual Network Details Tab

Monitor > Networking > Networks > default-domain:demo:vn18

Summary Port Map Port Distribution Instances **Details**

UVE Information

```

{
  "value": [
    {
      "name": "default-domain:demo:vn18",
      "value": {
        "UveVirtualNetworkAgent": {
          "out_bytes": 209754,
          "mirror_acl": null,
          "vrf_stats_list": [
            {
              "discards": 0,
              "name": "default-domain:demo:vn18:vn18",
              "encaps": 0,
              "receives": 0,
              "resolves": 0,
              "composites": 0,
              "tunnels": 0
            }
          ],
          "total_acl_rules": 3,
          "in_bandwidth_usage": 0,
          "out_bandwidth_usage": 0
        }
      }
    }
  ]
}

```

5041878

## Query > Flows

### IN THIS SECTION

- [Query > Flows > Flow Series | 943](#)
- [Example: Query Flow Series | 946](#)
- [Query > Flow Records | 948](#)
- [Query > Flows > Query Queue | 951](#)

Select **Query > Flows** to perform rich and complex SQL-like queries on flows in the Contrail Controller. You can use the query results for such things as gaining insight into the operation of applications in a virtual network, performing historical analysis of flow issues, and pinpointing problem areas with flows.

### Query > Flows > Flow Series

Select **Query > Flows > Flow Series** to create queries of the flow series table. The results are in the form of time series data for flow series. See [Figure 264 on page 943](#)

**Figure 264: Query Flow Series Window**

The screenshot displays the 'Query Flow Series' window in the Juniper Networks Contrail Controller. The breadcrumb navigation at the top reads 'Query > Flows > Flow Series'. On the left, a sidebar shows the 'Flows' menu item selected, with sub-items for 'Flow Series', 'Flow Records', 'Query Queue', and 'Logs'. The main content area is titled 'Query Flow Series' and includes a search bar. Below the title, there are several configuration fields: 'Time Range' is set to 'Last 30 Mins'; the 'Select' field is empty; the 'Where' field contains an asterisk (\*); and the 'Filter' field is empty. To the right of the 'Where' field is a 'Direction' dropdown menu currently set to 'INGRESS'. At the bottom left of the main area is a blue 'Run Query' button. The Juniper logo and the user name 'Admin' are visible in the top right corner of the interface.

The query fields available on the screen for the **Flow Series** tab are described in [Table 91 on page 944](#). Enter query data into the fields to create a SQL-like query to display and analyze flows.

Table 91: Query Flow Series Fields

| Field             | Description   |
|-------------------|---|
| <b>Time Range</b> | <p>Select a range of time to display the flow series:</p> <ul style="list-style-type: none"> <li>• Last 10 Mins</li> <li>• Last 30 Mins</li> <li>• Last 1 Hr</li> <li>• Last 6 Hrs</li> <li>• Last 12 Hrs</li> <li>• Custom</li> </ul> <p>Click <b>Custom</b> to enter a specific custom time range in two fields: <b>From Time</b> and <b>To Time</b>.</p> |
| <b>Select</b>     | <p>Click the edit button (pencil icon) to open a <b>Select</b> window (Figure 265 on page 945), where you can click one or more boxes to select the fields to display from the flow series, such as <b>Source VN, Dest VN, Bytes, Packets</b>, and more.</p>  |
| <b>Where</b>      | <p>Click the edit button (pencil icon) to open a query-writing window, where you can specify query values for variables such as <b>sourcevn, sourceip, destvn, destip, protocol, sport, dport</b>.</p>  |
| <b>Direction</b>  | <p>Select the desired flow direction: <b>INGRESS</b> or <b>EGRESS</b>.</p>  |
| <b>Filter</b>     | <p>Click the edit button (pencil icon) to open a <b>Filter</b> window (Figure 266 on page 946), where you can select filter items to sort by, the sort order, and limits to the number of results returned.</p>   |
| <b>Run Query</b>  | <p>Click <b>Run Query</b> to retrieve the flows that match the query you created. The flows are listed on the lower portion of the screen in a box with columns identifying the selected fields for each flow.</p>  |
| (graph buttons)   | <p>When <b>Time Granularity</b> is selected, you have the option to view results in graph or flowchart form. Graph buttons appear on the screen above the <b>Export</b> button. Click a graph button to transform the tabular results into a graphical chart display.</p>   |



Table 91: Query Flow Series Fields (Continued)

| Field         | Description  |
|---------------|--|
| <b>Export</b> | The Export button is displayed after you click <b>Run Query</b> . This allows you to export the list of flows to a text .csv file. |

The **Select** window allows you to select one or more attributes of a flow series by clicking the check box for each attribute desired, see [Figure 265 on page 945](#). The upper section of the **Select** window includes field names, and the lower portion lets you select units. Select **Time Granularity** and then select **SUM(Bytes)** or **SUM(Packets)** to aggregate bytes and packets in intervals.

Figure 265: Flow Series Select

The screenshot shows a window titled "Select" with a close button (X) in the top right corner. The window contains a list of attributes, each with a checkbox to its left. The attributes are arranged in three columns:

- Column 1: Source VN, Source IP, Source Port, Bytes, Packets
- Column 2: Destination VN, Destination IP, Destination Port, SUM(Bytes), SUM(Packets)
- Column 3: Time Granularity, Protocol, Virtual Router

At the bottom right of the window, there are two buttons: "Cancel" and "Apply". The "Apply" button is highlighted in blue. On the right side of the window, there is a vertical label "50-1600".

Use the **Filter** window to refine the display of query results for flows, by defining an attribute by which to sort the results, the sort order of the results, and any limit needed to restrict the number of results. See [Figure 266 on page 946](#).

Figure 266: Flow Series Filter

Filter

Sort By

- Source VN
- Destination VN
- Protocol
- Source IP
- Destination IP
- Virtual Router
- Source Port
- Destination Port
- Bytes
- Sum(Bytes)
- Packets
- Sum(Packets)

Sort Order: ASC

Limit By: [ ]

Cancel Apply

5041599

### Example: Query Flow Series

The following is an example flow series query that returns the time series of the summation traffic in bytes for all combinations of source VN and destination VN for the last 10 minutes, with the bytes aggregated in 10 second intervals. See [Figure 267 on page 946](#).

Figure 267: Example: Query Flow Series

Query Flow Series

Time Range: Last 10 Mins

Select: sourcevn, destvn, time-granularity, sum(bytes)

Where: \*

Filter: [ ]

Time Granularity: 10 secs

Direction: INGRESS

Run Query

5041599

The query returns tabular time series data, see [Figure 268 on page 947](#), for the following combinations of Source VN and Dest VN:

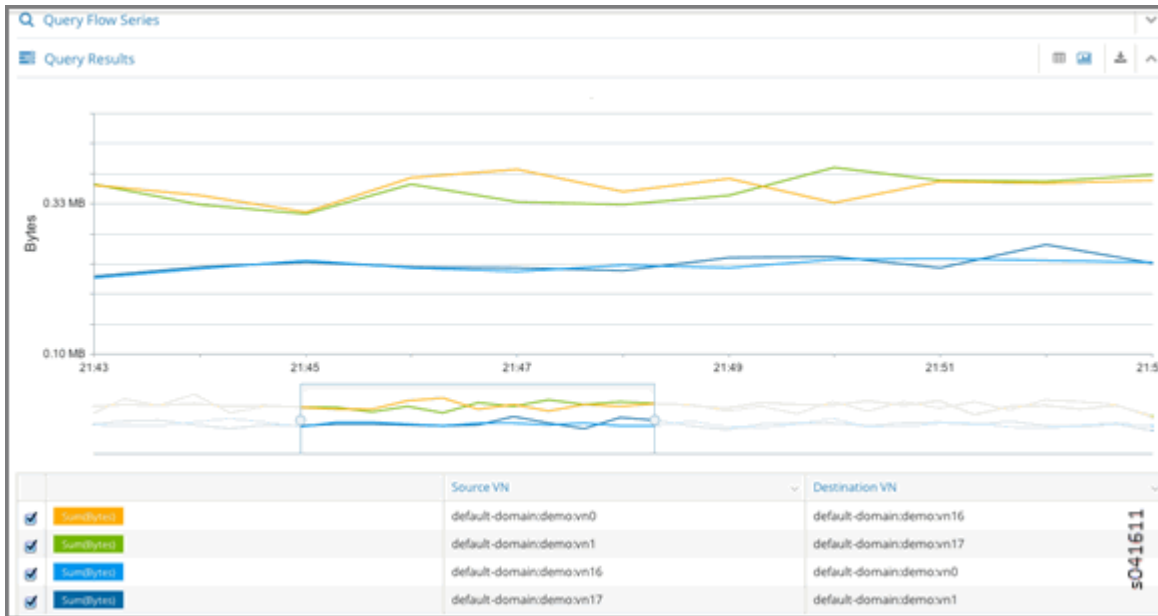
1. Flow Class 1: Source VN = default-domain:demo:front-end, Dest VN=\_\_UNKNOWN\_\_
2. Flow Class 2: Source VN = default-domain:demo:front-end, Dest VN=default-domain:demo:back-end

**Figure 268: Query Flow Series Tabular Results**

| Time                    | Source VN                | Dest. VN                 | Direction | SUM(Bytes) |
|-------------------------|--------------------------|--------------------------|-----------|------------|
| 2013-08-05 18:59:30:0:0 | default-domain:demo:vn0  | default-domain:demo:vn16 | INGRESS   | 421,128    |
| 2013-08-05 18:59:40:0:0 | default-domain:demo:vn0  | default-domain:demo:vn16 | INGRESS   | 227,000    |
| 2013-08-05 18:59:50:0:0 | default-domain:demo:vn0  | default-domain:demo:vn16 | INGRESS   | 216,816    |
| 2013-08-05 19:00:00:0:0 | default-domain:demo:vn0  | default-domain:demo:vn16 | INGRESS   | 387,036    |
| 2013-08-05 18:59:30:0:0 | default-domain:demo:vn1  | default-domain:demo:vn17 | INGRESS   | 52,944     |
| 2013-08-05 18:59:40:0:0 | default-domain:demo:vn1  | default-domain:demo:vn17 | INGRESS   | 52,692     |
| 2013-08-05 18:59:50:0:0 | default-domain:demo:vn1  | default-domain:demo:vn17 | INGRESS   | 58,040     |
| 2013-08-05 19:00:00:0:0 | default-domain:demo:vn1  | default-domain:demo:vn17 | INGRESS   | 42,480     |
| 2013-08-05 18:59:30:0:0 | default-domain:demo:vn16 | default-domain:demo:vn0  | INGRESS   | 17,832     |
| 2013-08-05 18:59:40:0:0 | default-domain:demo:vn16 | default-domain:demo:vn0  | INGRESS   | 27,320     |
| 2013-08-05 18:59:50:0:0 | default-domain:demo:vn16 | default-domain:demo:vn0  | INGRESS   | 20,792     |
| 2013-08-05 19:00:00:0:0 | default-domain:demo:vn16 | default-domain:demo:vn0  | INGRESS   | 10,404     |

Because **Time Granularity** is selected, the results can also be displayed as graphical charts. Click the graph button on the right side of the tabular results. The results are displayed in a graphical flow chart. See [Figure 269 on page 948](#).

Figure 269: Query Flow Series Graphical Results



## Query > Flow Records

Select **Query > Flow Records** to create queries of individual flow records for detailed debugging of connectivity issues between applications and virtual machines. Queries at this level return records of the active flows within a given time period.

Figure 270: Flow Records

Query > Flows > Flow Records

Search

Query

Query Flow Records

Time Range: Last 10 Mins

Select:

Where:

Direction: INGRESS

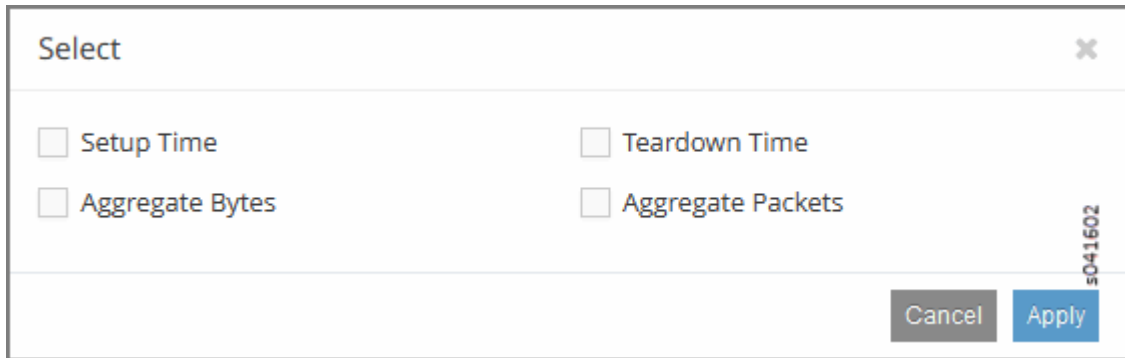
Run Query

The query fields available on the screen for the **Flow Records** tab are described in [Table 92 on page 949](#). Enter query data into the fields to create an SQL-like query to display and analyze flows.

Table 92: Query Flow Records Fields

| Field             | Description  |
|-------------------|--|
| <b>Time Range</b> | <p>Select a range of time for the flow records:</p> <ul style="list-style-type: none"> <li>• Last 10 Mins</li> <li>• Last 30 Mins</li> <li>• Last 1 Hr</li> <li>• Last 6 Hrs</li> <li>• Last 12 Hrs</li> <li>• Custom</li> </ul> <p>Click <b>Custom</b> to enter a specified custom time range in two fields: <b>From Time</b> and <b>To Time</b>.</p> |
| <b>Select</b>     | <p>Click the edit button (pencil icon) to open a <b>Select</b> window (<a href="#">Figure 271 on page 950</a>), where you can click one or more boxes to select attributes to display for the flow records, including <b>Setup Time</b>, <b>Teardown Time</b>, <b>Aggregate Bytes</b>, and <b>Aggregate Packets</b>.</p>                               |
| <b>Where</b>      | <p>Click the edit button (pencil icon) to open a query-writing window where you can specify query values for <b>sourcevn</b>, <b>sourceip</b>, <b>destvn</b>, <b>destip</b>, <b>protocol</b>, <b>sport</b>, <b>dport</b> .</p>   |
| <b>Direction</b>  | <p>Select the desired flow direction: <b>INGRESS</b> or <b>EGRESS</b>.</p>   |
| <b>Run Query</b>  | <p>Click <b>Run Query</b> to retrieve the flow records that match the query you created. The records are listed on the lower portion of the screen in a box with columns identifying the fields for each flow.</p>   |
| <b>Export</b>     | <p>The <b>Export</b> button is displayed after you click <b>Run Query</b>, allowing you to export the list of flows to a text <b>.csv</b> file.</p>  |

The **Select** window allows you to select one or more attributes to display for the flow records selected, see [Figure 271 on page 950](#).

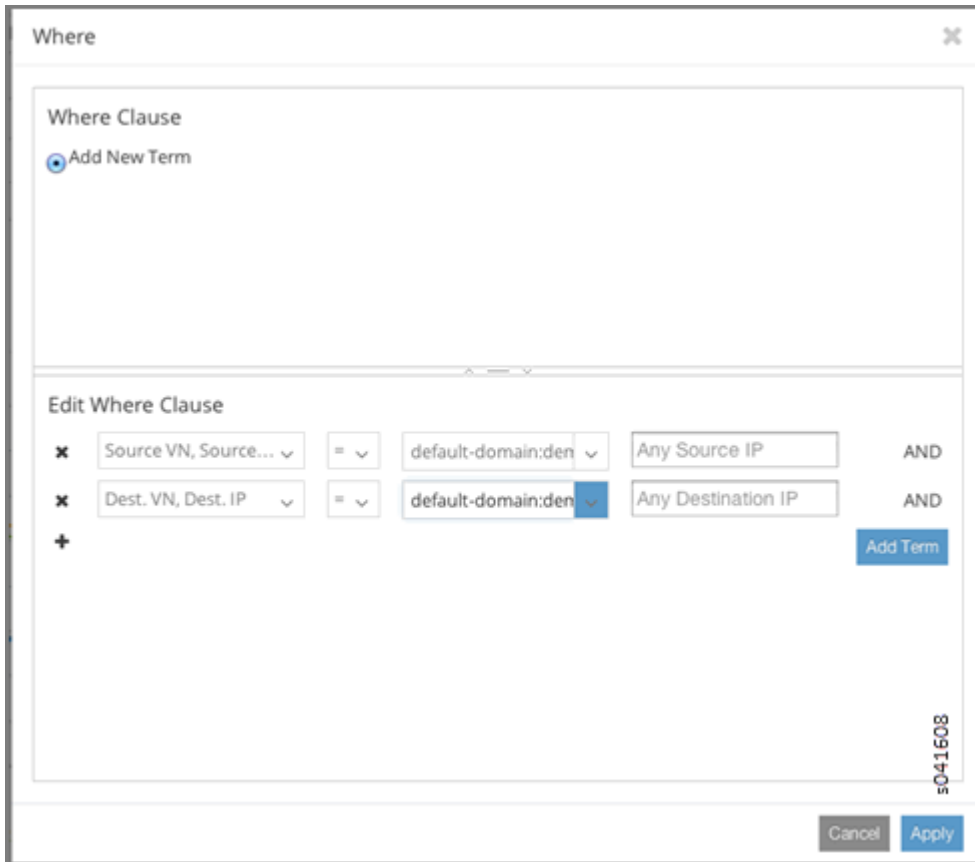
**Figure 271: Flow Records Select Window**A screenshot of a 'Select' dialog window. The window has a title bar with the text 'Select' and a close button (X) in the top right corner. The main area contains four checkboxes arranged in a 2x2 grid: 'Setup Time', 'Teardown Time', 'Aggregate Bytes', and 'Aggregate Packets'. All checkboxes are currently unchecked. At the bottom right of the window, there are two buttons: 'Cancel' (grey) and 'Apply' (blue). A vertical label '5041602' is positioned on the right side of the dialog, partially overlapping the 'Apply' button.

You can restrict the query to a particular source VN and destination VN combination using the **Where** section.

The **Where Clause** supports logical AND and logical OR operations, and is modeled as a logical OR of multiple AND terms. For example: ( (term1 AND term2 AND term3..) OR (term4 AND term5) OR...).

Each term is a single variable expression such as **Source VN = VN1**.

Figure 272: Where Clause Window



### Query > Flows > Query Queue

Select **Query > Flows > Query Queue** to display queries that are in the queue waiting to be performed on the data. See [Figure 273 on page 951](#).

Figure 273: Flows Query Queue

| Date                | Query  | Progress | Records | Status    | Time Taken |
|---------------------|--|----------|---------|-----------|------------|
| 2013-10-09 18:07:06 | {"table": "FlowSeriesTable", "start_time": 1381267020000000, "end_time": 1381277820000000, "select_fields": [ "flow_class_id", "direction_ing", "sum(bytes)", "T=60" ], "dir": 1 } | 100%     | 180     | completed | 150 secs   |
| 2013-10-09 17:55:48 | {"table": "FlowSeriesTable", "start_time": 1381267020000000, "end_time": 1381277820000000, "select_fields": [ "flow_class_id", "direction_ing", "sum(bytes)", "T=60" ], "dir": 1 } | 100%     | 180     | completed | 145 secs   |
| 2013-10-09 17:29:39 | {"table": "FlowSeriesTable", "start_time": 1381267020000000, "end_time": 1381277820000000, "select_fields": [ "flow_class_id", "direction_ing", "sum(bytes)", "T=60" ], "dir": 1 } | 100%     | 180     | completed | 170 secs   |
| 2013-10-09 16:57:10 | {"table": "FlowSeriesTable", "start_time": 1381267020000000, "end_time": 1381277820000000, "select_fields": [ "flow_class_id", "direction_ing", "sum(bytes)", "T=60" ], "dir": 1 } | 100%     | 180     | completed | 270 secs   |
| 2013-10-09 16:39:48 | {"table": "FlowSeriesTable", "start_time": 1381360140000000, "end_time": 1381361940000000, "select_fields": [ "flow_class_id", "direction_ing", "T=60", "sum(bytes)" ], "dir": 1 } | 100%     | 30      | completed | 60 secs    |
| 2013-10-09 11:07:29 | {"table": "FlowSeriesTable", "start_time": 1381338420000000, "end_time": 1381342020000000, "select_fields": [ "flow_class_id", "direction_ing", "sum(bytes)", "T=60" ], "dir": 1 } | 100%     | 7       | completed | 15 secs    |

Displaying 1 - 6 of 31 Records

The query fields available on the screen for the **Flow Records** tab are described in [Table 93 on page 952](#). Enter query data into the fields to create an SQL-like query to display and analyze flows.

**Table 93: Query Flow Records Fields**

| Field             | Description  |
|-------------------|--|
| <b>Date</b>       | The date and time the query was started.   |
| <b>Query</b>      | A display of the parameters set for the query.   |
| <b>Progress</b>   | The percentage completion of the query to date.  |
| <b>Records</b>    | The number of records matching the query to date.  |
| <b>Status</b>     | The status of the query, such as <b>completed</b> .  |
| <b>Time Taken</b> | The amount of time in seconds it has taken the query to return the matching records.   |
| (Action icon)     | Click the <b>Action</b> icon and select <b>View Results</b> to view a list of the records that match the query, or click <b>Delete</b> to remove the query from the queue. |

## RELATED DOCUMENTATION

[Understanding Flow Sampling | 959](#)

## Query > Logs

### IN THIS SECTION

- [Query > Logs Menu Options | 953](#)
- [Query > Logs > System Logs | 953](#)
- [Sample Query for System Logs | 955](#)

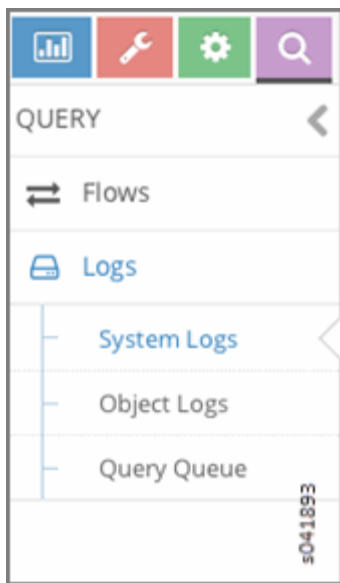


The **Query > Logs** option allows you to access the system log and object log activity of any Contrail Controller component from one central location.

### Query > Logs Menu Options

Click **Query > Logs** to access the **Query Logs** menu, where you can select **System Logs** to view system log activity, **Object Logs** to view object logs activity, and **Query Queue** to create custom queries of log activity; see [Figure 274 on page 953](#).

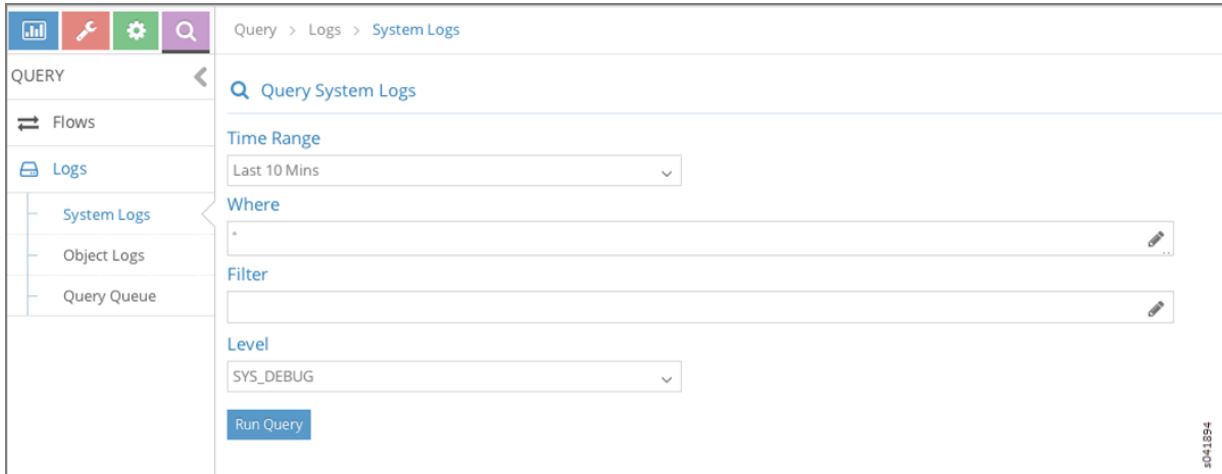
**Figure 274: Query > Logs**



### Query > Logs > System Logs

Click **Query > Logs > System Logs** to access the **Query System Logs** menu, where you can view system logs according to criteria that you determine. See [Figure 275 on page 954](#).

Figure 275: Query > Logs > System Logs



The query fields available on the **Query System Logs** screen are described in [Table 94 on page 954](#).

Table 94: Query System Logs Fields

| Field             | Description   |
|-------------------|---|
| <b>Time Range</b> | <p>Select a range of time for which to see the system logs:</p> <ul style="list-style-type: none"> <li>• Last 10 Mins</li> <li>• Last 30 Mins</li> <li>• Last 1 Hr</li> <li>• Last 6 Hrs</li> <li>• Last 12 Hrs</li> <li>• Custom</li> </ul> <p>If you click Custom, enter a desired time range in two new fields: <b>From Time</b> and <b>To Time</b>.</p> |
| <b>Where</b>      | <p>Click the edit button (pencil icon) to open a query-writing window, where you can specify query values for variables such as Source, Module, MessageType, and the like, in order to retrieve specific information.</p>   |

Table 94: Query System Logs Fields (Continued)

| Field            | Description   |
|------------------|---|
| <b>Level</b>     | Select the message severity level to view: <ul style="list-style-type: none"> <li>• SYS_NOTICE</li> <li>• SYS_EMERG</li> <li>• SYS_ALERT</li> <li>• SYS_CRIT</li> <li>• SYS_ERR</li> <li>• SYS_WARN</li> <li>• SYS_INFO</li> <li>• SYS_DEBUG</li> </ul> |
| <b>Run Query</b> | Click this button to retrieve the system logs that match the query. The logs are listed in a box with columns showing the <b>Time</b> , <b>Source</b> , <b>Module Id</b> , <b>Category</b> , <b>Log Type</b> , and <b>Log</b> message.                  |
| <b>Export</b>    | This button appears after you click <b>Run Query</b> , allowing you to export the list of system messages to a text/csv file.   |

### Sample Query for System Logs

This section shows a sample system logs query designed to show all **System Logs** from ModuleId = VRouterAgent on Source = b1s16 and filtered by **Level** = SYS\_DEBUG.

1. At the **Query System Logs** screen, click in the **Where** field to access the **Where** query screen and enter information defining the location to query in the **Edit Where Clause** section and click **OK**; see [Figure 276 on page 956](#).

Figure 276: Edit Where Clause

The screenshot shows a dialog box titled "Where" with a close button (X) in the top right corner. Inside the dialog, there is a section titled "Where Clause" with a radio button labeled "Add New Term" selected. Below this is a section titled "Edit Where Clause" containing two rows of criteria:

- Row 1: A dropdown menu with "ModuleId", an equals sign (=) in a dropdown, another dropdown with "VRouterAgent", and the word "AND".
- Row 2: A dropdown menu with "Source", an equals sign (=) in a dropdown, another dropdown with "b1s16", and the word "AND".

Below the second row is a plus sign (+) and a blue button labeled "Add Term". At the bottom of the dialog are two buttons: "OK" and "Cancel". A small vertical text "FO41895" is visible in the bottom right corner of the dialog frame.

2. The information you defined at the Where screen displays on the **Query System Logs**. Enter any more defining information needed; see [Figure 277 on page 957](#). When finished, click **Run Query** to display the results.

Figure 277: Sample Query System Logs

Query System Logs

Time Range  
Last 10 Mins

Where  
(ModuleId = VRouterAgent AND Source = b1s16)

Filter

Level  
SYS\_DEBUG

Run Query

#041896

## Query > Logs > Object Logs

Object logs allow you to search for logs associated with a particular object, for example, all logs for a specified virtual network. Object logs record information related to modifications made to objects, including creation, deletion, and other modifications; see [Figure 278 on page 957](#).

Figure 278: Query &gt; Logs &gt; Object Logs

Query Object Logs

Time Range  
Last 12 Hrs

Object Type  
Virtual Network

Object Id  
default-domain:demo:vn14

Select  
ObjectLog, SystemLog

Where  
\*

Filter

Run Query

#041897

The query fields available on the **Object Logs** screen are described in [Table 95 on page 958](#).

Table 95: Object Logs Query Fields

| Field              | Description  |
|--------------------|--|
| <b>Time Range</b>  | <p>Select a range of time for which to see the logs:</p> <ul style="list-style-type: none"> <li>• Last 10 Mins</li> <li>• Last 30 Mins</li> <li>• Last 1 Hr</li> <li>• Last 6 Hrs</li> <li>• Last 12 Hrs</li> <li>• Custom</li> </ul> <p>If you click Custom, enter a desired time range in two new fields: <b>From Time</b> and <b>To Time</b>.</p> |
| <b>Object Type</b> | <p>Select the object type for which to show logs:</p> <ul style="list-style-type: none"> <li>• Virtual Network</li> <li>• Virtual Machine</li> <li>• Virtual Router</li> <li>• BGP Peer</li> <li>• Routing Instance</li> <li>• XMPP Connection</li> </ul>  |
| <b>Object Id</b>   | <p>Select from a list of available identifiers the name of the object you wish to use.</p>   |
| <b>Select</b>      | <p>Click the edit button (pencil icon) to open a window where you can select searchable types by clicking a checkbox:</p> <ul style="list-style-type: none"> <li>• ObjectLog</li> <li>• SystemLog</li> </ul>   |

Table 95: Object Logs Query Fields (Continued)

| Field            | Description   |
|------------------|---|
| <b>Where</b>     | Click the edit button (pencil icon) to open the query-writing window, where you can specify query values for variables such as <b>Source</b> , <b>ModuleId</b> , and <b>MessageType</b> , in order to retrieve information as specific as you wish. |
| <b>Run Query</b> | Click this button to retrieve the system logs that match the query. The logs are listed in a box with columns showing the <b>Time</b> , <b>Source</b> , <b>Module Id</b> , <b>Category</b> , <b>Log Type</b> , and <b>Log</b> message.              |
| <b>Export</b>    | This button appears after you click <b>Run Query</b> , allowing you to export the list of system messages to a text/csv file.   |

## Understanding Flow Sampling

### IN THIS SECTION

- [Flow Sampling | 959](#)
- [Flow Handling | 960](#)
- [Flow Aging | 961](#)
- [TCP State-Based Flow Handling and Aging | 961](#)

This topic describes how flow records are sampled and exported to the Contrail collector, flow handling, and flow aging.

### Flow Sampling

The Contrail vRouter agent exports flow records to the Contrail collector when a flow is created or deleted. It also updates flow statistics at regular intervals.

If all flow records are exported from the agent, depending on the scale of flows, some of the exported flows might be dropped due to queue overflow.

In Contrail Release 2.22 and later, to reduce queue overflow, flow records are sampled and exported to the Contrail Collector based on sampling.

The flows that are exported are selected based on the following parameters used in the algorithm:

- The configured flow export rate. This is configured as part of the `global-vrouter-config` object.
- The actual flow export rate.
- The sampling threshold. This is a dynamic value calculated internally. If the flow statistics in a flow sample are above this threshold, the flow record is exported.

Each flow is subjected to the following algorithm at regular intervals. The algorithm determines whether to export the sample or not.

- Flows with traffic that is greater than or equal to the sampling threshold are always exported. The byte and packet counts are reported without modification.
- Flows with traffic that is less than the sampling threshold are exported according to the probability. The byte and packet counts are adjusted upwards according to the probability.

The probability is calculated as  $(\text{bytes during the interval}) / (\text{sampling threshold})$ .

- The system generates a random number less than the sampling threshold. If the byte count during the interval is less than the random number, then the flow sample is not exported.
- If none of these conditions are met, the flow sample is exported after normalizing the byte count and packet count during the interval. Normalization is done by dividing the byte count and packet count during the interval by the probability. This normalization is used as a heuristic to account for statistics of flow samples that are dropped.

The actual flow export rate is close to the configured export rate. If there is a large deviation, the sampling threshold is adjusted to bring the actual flow export rate close to the configured flow export rate.

## Flow Handling

When a virtual machine sends or receives IP traffic, forward and reverse flow entries are set up. When the first packet arrives, a flow key is used to hash into a flow table (identify a hash bucket). The flow key is based on five-tuples consisting of source and destination IP addresses, ports, and the IP protocol.

A flow entry is created and the packet is sent to the Contrail vRouter agent. Configured policies are applied and the flow action is updated. The agent also creates a flow entry for the reverse direction where relevant. Subsequent packets match the established flow entries and are forwarded, dropped, NAT translated, and so on, based on the flow action.



When the hash bucket is full, entries are created in an overflow table. In releases earlier than Contrail Release 2.22, the overflow table was a global table, which is searched sequentially. In Contrail Release 2.22 and later, the overflow entries are maintained as a list against the hash bucket.

By default, the maximum number of flow table and overflow table entries are 512,000 and 8000 respectively. These can be modified by configuring them as vRouter module parameters, for example: `vr_flow_entries` and `vr_oflow_entries`.

For more information about the vRouter module parameters, see <https://github.com/Juniper/contrail-controller/wiki/Vrouter-Module-Parameters>.

## Flow Aging

Flows are aged out based on inactivity for a specified period of time. By default, the timeout value is 180 seconds. This can be modified by configuring the `flow_cache_timeout` parameter under the `DEFAULT` section in the `/etc/contrail/contrail-vrouter-agent.conf` file.

## TCP State-Based Flow Handling and Aging

### TCP State-Based Flow Handling

In Contrail Release 2.22 and later, the Contrail vRouter monitors TCP flows to identify entries that can be reused without going through the standard aging cycle.

All flow entries that match TCP flows that have experienced a connection teardown, either through the standard TCP closure cycle (FIN/ACK-FIN/ACK) or the RST indicator, are torn down by the vRouter and are immediately available for use by new qualified flows.

The vRouter also keeps track of connection establishment cycles and exports the necessary information to the vRouter agent, such as SYN/ACK and a digested established flag. This allows the vRouter agent to tear down flows that do not experience a full connection cycle.

Flows that the vRouter identifies as reuse candidates, or eviction candidates, are marked as such in the flow entry. Flows are in the evicted state when they become available for other new flows to be reused.

This two-step transition is used so that the flow entry remains valid until the packet reaches the destination, preventing the flow from getting remapped to another flow entry in the interim.

### Protocol-Based Flow Aging

Although TCP flows are deleted based on TCP state, you are sometimes required to age out specific protocol flows more aggressively. One example is when a DNS server is run in one VM. In this case, multiple flows are set up for DNS. A pair of flows are set up to serve each query. Because the flows are

no longer required after the query is served, the timeout can be lower for these flows. To handle these cases, protocol-based flow aging is used.

With protocol-based flow aging, the aging timeout can be configured per protocol. All other protocols continue to use the default aging timeout.

Protocol-based flow aging is supported in Contrail Release 2.22 and later.

The configuration for protocol-based flow aging can be done in the `global-vrouter-config` object. For example, to have all DNS flows aged out in five seconds, use the following entry: `protocol = udp, port = 53` will be set an aging timeout of 5 seconds.

## Fat Flow

Contrail supports optimization to reduce the number of flows set up by reusing a flow. Consequently, a single flow pair (fat flow) can be used for any number of sessions between two endpoints for the same application protocol.

Any number of DNS sessions from a client to the server can use a single flow pair. The effect is that the flow hash key is reduced from five-tuples to four-tuples, consisting of source and destination IP addresses, the server port, and the IP protocol. The client port is not used in the flow key

This feature can be configured by specifying the list of *fat-flow* protocols on a virtual machine interface (VMI). For each such application protocol, the list contains the protocol and port pairs. If you want to enable the fat flow feature on the client side, the configuration must be applied on the client VMI as well.

## RELATED DOCUMENTATION

| [Query > Flows](#)

## Example: Debugging Connectivity Using Monitoring for Troubleshooting

### IN THIS SECTION

- [Using Monitoring to Debug Connectivity | 963](#)

## Using Monitoring to Debug Connectivity

This example shows how you can use monitoring to debug connectivity in your Contrail system. You can use the demo setup in Contrail to use these steps on your own.

1. Navigate to **Monitor -> Networking -> Networks -> default-domain:demo:vn0, Instance** ed6abd16-250e-4ec5-a382-5cbc458fb0ca with **IP address** 192.168.0.252 in the virtual network vn0; see [Figure 279 on page 963](#)

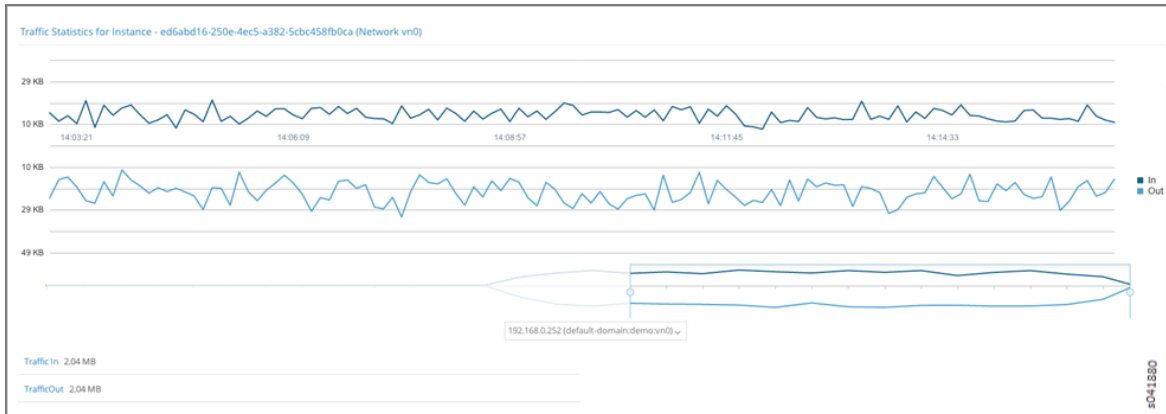
**Figure 279: Navigate to Instance**



| Instance                             | Traffic In | Traffic Out |
|--------------------------------------|------------|-------------|
| ed6abd16-250e-4ec5-a382-5cbc458fb0ca | 1.73 MB    | 1.74 MB     |
| 682b7414-c8ba-45ee-91bc-9c22cd56c9fd | 1.72 MB    | 1.72 MB     |

2. Click the instance to view **Traffic Statistics for Instance**. see [Figure 280 on page 963](#).

**Figure 280: Traffic Statistics for Instance**

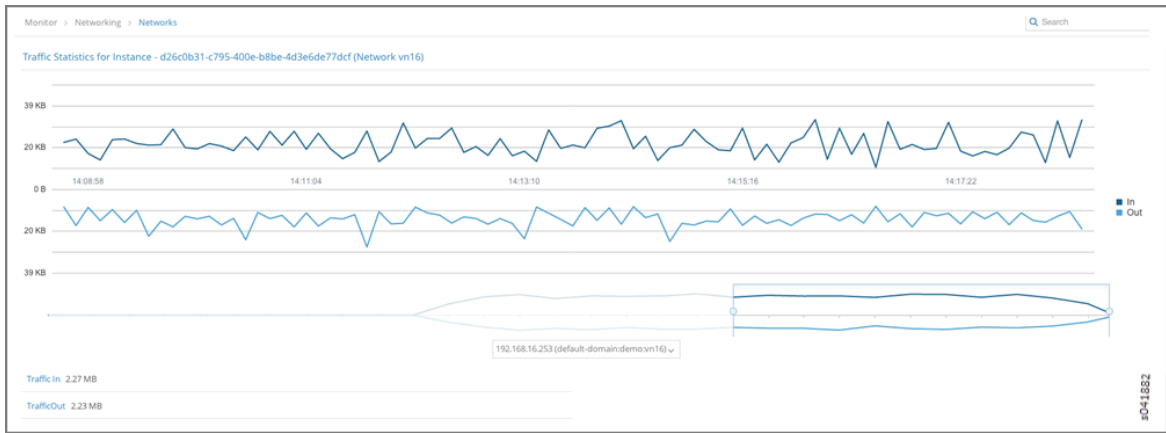


3. **Instance** d26c0b31-c795-400e-b8be-4d3e6de77dcf with **IP address** 192.168.0.253 in the virtual network vn16. see [Figure 281 on page 964](#) and [Figure 282 on page 964](#).

Figure 281: Navigate to Instance

| Instance                             | Traffic In | Traffic Out |
|--------------------------------------|------------|-------------|
| d26c0b31-c795-400e-b8be-4d3e6de77dcf | 2.18 MB    | 2.13 MB     |
| 23045415-b679-4d9a-8f9d-96c162de28be | 2.11 MB    | 2.16 MB     |

Figure 282: Traffic Statistics for Instance



- From **Monitor->Infrastructure->Virtual Routers->a3s18->Interfaces**, we can see that Instance ed6abd16-250e-4ec5-a382-5cbc458fb0ca is hosted on **Virtual Router a3s18**; see [Figure 283 on page 964](#).

Figure 283: Navigate to a3s18 Interfaces

| Name           | Label | Status | Network                  | IP Address     | Floating IP | Instance                             |
|----------------|-------|--------|--------------------------|----------------|-------------|--------------------------------------|
| tap1dae0121-4c | 16    | Up     | default-domain:demo:vn0  | 192.168.0.252  | None        | ed6abd16-250e-4ec5-a382-5cbc458fb0ca |
| tap249de2e1-97 | 18    | Up     | default-domain:demo:vn16 | 192.168.16.252 | None        | 23045415-b679-4d9a-8f9d-96c162de28be |
| tap5b3b3963-74 | 19    | Up     | default-domain:demo:vn17 | 192.168.17.252 | None        | 99311eda-261e-47e8-b4a7-8d126d74998f |
| tapc740843c-6b | 17    | Up     | default-domain:demo:vn1  | 192.168.1.252  | None        | 20244e9f-a4ed-4a32-803f-15c75323572e |

- From **Monitor->Infrastructure->Virtual Routers->a3s19->Interfaces**, we can see that Instance d26c0b31-c795-400e-b8be-4d3e6de77dcf is hosted on **Virtual Router a3s19**; see [Figure 284 on page 965](#).

Figure 284: Navigate to a3s19 Interfaces

| Name           | Label | Status | Network                  | IP Address     | Floating IP | Instance                             |
|----------------|-------|--------|--------------------------|----------------|-------------|--------------------------------------|
| tap29585b2f-c2 | 19    | Up     | default-domain:demo:vn16 | 192.168.16.253 | None        | d26c0b31-c795-400e-bbbe-4d3e6de77dcf |
| tap8257421d-43 | 18    | Up     | default-domain:demo:vn1  | 192.168.1.253  | None        | eebce321-7536-46e7-a454-ceff1f3ac095 |
| tapc83e9d87-66 | 17    | Up     | default-domain:demo:vn17 | 192.168.17.253 | None        | b2425f5-67e-4060-9478-81a4a825541    |
| tape5ea97e3-55 | 16    | Up     | default-domain:demo:vn0  | 192.168.0.253  | None        | 682b7414-cba-45ee-91bc-9c22cd6ec69d  |

- Virtual Routers a3s18 and a3s19 have the ACL entries to allow connectivity between default-domain:demo:vn0 and default-domain:demo:vn16 networks; see Figure 285 on page 965 and Figure 286 on page 965.

Figure 285: ACL Connectivity a3s18

| UUID                     | Flows | Action | Protocol | Source Network or Prefix | Source Port | Destination Network or Prefix | Destination Port | Source Policy Rule | ACE Id |
|--------------------------|-------|--------|----------|--------------------------|-------------|-------------------------------|------------------|--------------------|--------|
| a724928e-3f30-477a-ad... | 16    | pass   | any      | default-domain:demo:vn0  | any         | default-domain:demo:vn16      | any              |                    | 1      |
|                          |       | pass   | any      | default-domain:demo:vn16 | any         | default-domain:demo:vn0       | any              |                    | 2      |
|                          |       | pass   | any      | default-domain:demo:vn0  | any         | default-domain:demo:vn0       | any              |                    | 3      |
| b32143a3-0ed0-4ae2-9c... | 16    | pass   | any      | default-domain:demo:vn1  | any         | default-domain:demo:vn17      | any              |                    | 1      |
|                          |       | pass   | any      | default-domain:demo:vn17 | any         | default-domain:demo:vn1       | any              |                    | 2      |
|                          |       | pass   | any      | default-domain:demo:vn1  | any         | default-domain:demo:vn1       | any              |                    | 3      |
| bbc9810-ef9c-41f9-aa7... | 16    | pass   | any      | default-domain:demo:vn0  | any         | default-domain:demo:vn16      | any              |                    | 1      |
|                          |       | pass   | any      | default-domain:demo:vn16 | any         | default-domain:demo:vn0       | any              |                    | 2      |
|                          |       | pass   | any      | default-domain:demo:vn16 | any         | default-domain:demo:vn16      | any              |                    | 3      |
| d1b47291-7a21-4fde-8d... | 16    | pass   | any      | default-domain:demo:vn1  | any         | default-domain:demo:vn17      | any              |                    | 1      |
|                          |       | pass   | any      | default-domain:demo:vn17 | any         | default-domain:demo:vn1       | any              |                    | 2      |
|                          |       | pass   | any      | default-domain:demo:vn17 | any         | default-domain:demo:vn17      | any              |                    | 3      |

Figure 286: ACL Connectivity a3s19

| UUID                     | Flows | Action | Protocol | Source Network or Prefix | Source Port | Destination Network or Prefix | Destination Port | Source Policy Rule | ACE Id |
|--------------------------|-------|--------|----------|--------------------------|-------------|-------------------------------|------------------|--------------------|--------|
| a724928e-3f30-477a-ad... | 16    | pass   | any      | default-domain:demo:vn0  | any         | default-domain:demo:vn16      | any              |                    | 1      |
|                          |       | pass   | any      | default-domain:demo:vn16 | any         | default-domain:demo:vn0       | any              |                    | 2      |
|                          |       | pass   | any      | default-domain:demo:vn0  | any         | default-domain:demo:vn0       | any              |                    | 3      |
| b32143a3-0ed0-4ae2-9c... | 16    | pass   | any      | default-domain:demo:vn1  | any         | default-domain:demo:vn17      | any              |                    | 1      |
|                          |       | pass   | any      | default-domain:demo:vn17 | any         | default-domain:demo:vn1       | any              |                    | 2      |
|                          |       | pass   | any      | default-domain:demo:vn1  | any         | default-domain:demo:vn1       | any              |                    | 3      |
| bbc9810-ef9c-41f9-aa7... | 16    | pass   | any      | default-domain:demo:vn0  | any         | default-domain:demo:vn16      | any              |                    | 1      |
|                          |       | pass   | any      | default-domain:demo:vn16 | any         | default-domain:demo:vn0       | any              |                    | 2      |
|                          |       | pass   | any      | default-domain:demo:vn16 | any         | default-domain:demo:vn16      | any              |                    | 3      |
| d1b47291-7a21-4fde-8d... | 16    | pass   | any      | default-domain:demo:vn1  | any         | default-domain:demo:vn17      | any              |                    | 1      |
|                          |       | pass   | any      | default-domain:demo:vn17 | any         | default-domain:demo:vn1       | any              |                    | 2      |
|                          |       | pass   | any      | default-domain:demo:vn17 | any         | default-domain:demo:vn17      | any              |                    | 3      |

- Next, verify the routes on the control node for routing instances default-domain:demo:vn0:vn0 and default-domain:demo:vn16:vn16; see Figure 287 on page 966 and Figure 288 on page 966.

Figure 287: Routes default-domain:demo:vn0:vn0

| Prefix                             | Address Family | Protocol | Source     | Next hop   | Label | Local Preference | AS Path    |
|------------------------------------|----------------|----------|------------|------------|-------|------------------|------------|
| 192.168.0.252/32                   | inet           | XMPP     | a3s18      | 10.84.17.4 | 16    | 100              | -          |
|                                    | inet           | BGP      | 10.84.17.3 | 10.84.17.4 | 16    | 100              | AS_PATH: 0 |
| 192.168.0.253/32                   | inet           | XMPP     | a3s19      | 10.84.17.5 | 16    | 100              | -          |
|                                    | inet           | BGP      | 10.84.17.3 | 10.84.17.5 | 16    | 100              | AS_PATH: 0 |
| 192.168.16.252/32                  | inet           | XMPP     | a3s18      | 10.84.17.4 | 17    | 100              | -          |
|                                    | inet           | BGP      | 10.84.17.3 | 10.84.17.4 | 17    | 100              | AS_PATH: 0 |
| 192.168.16.253/32                  | inet           | XMPP     | a3s19      | 10.84.17.5 | 17    | 100              | -          |
|                                    | inet           | BGP      | 10.84.17.3 | 10.84.17.5 | 17    | 100              | AS_PATH: 0 |
| 10.84.17.4:1:192.168.0.255,0.0.0.0 | inetmcast      | XMPP     | a3s18      | 10.84.17.4 | 0     | 100              | -          |
| 10.84.17.4:1:255.255.255.0,0.0.0.0 | inetmcast      | XMPP     | a3s18      | 10.84.17.4 | 0     | 100              | -          |
| 10.84.17.5:1:192.168.0.255,0.0.0.0 | inetmcast      | XMPP     | a3s19      | 10.84.17.5 | 0     | 100              | -          |
| 10.84.17.5:1:255.255.255.0,0.0.0.0 | inetmcast      | XMPP     | a3s19      | 10.84.17.5 | 0     | 100              | -          |

Figure 288: Routes default-domain:demo:vn16:vn16

| Prefix                              | Address Family | Protocol | Source     | Next hop   | Label | Local Preference | AS Path    |
|-------------------------------------|----------------|----------|------------|------------|-------|------------------|------------|
| 192.168.0.252/32                    | inet           | XMPP     | a3s18      | 10.84.17.4 | 16    | 100              | -          |
|                                     | inet           | BGP      | 10.84.17.3 | 10.84.17.4 | 16    | 100              | AS_PATH: 0 |
| 192.168.0.253/32                    | inet           | XMPP     | a3s19      | 10.84.17.5 | 16    | 100              | -          |
|                                     | inet           | BGP      | 10.84.17.3 | 10.84.17.5 | 16    | 100              | AS_PATH: 0 |
| 192.168.16.252/32                   | inet           | XMPP     | a3s18      | 10.84.17.4 | 17    | 100              | -          |
|                                     | inet           | BGP      | 10.84.17.3 | 10.84.17.4 | 17    | 100              | AS_PATH: 0 |
| 192.168.16.253/32                   | inet           | XMPP     | a3s19      | 10.84.17.5 | 17    | 100              | -          |
|                                     | inet           | BGP      | 10.84.17.3 | 10.84.17.5 | 17    | 100              | AS_PATH: 0 |
| 10.84.17.4:2:192.168.16.255,0.0.0.0 | inetmcast      | XMPP     | a3s18      | 10.84.17.4 | 0     | 100              | -          |
| 10.84.17.4:2:255.255.255.0,0.0.0.0  | inetmcast      | XMPP     | a3s18      | 10.84.17.4 | 0     | 100              | -          |
| 10.84.17.5:2:192.168.16.255,0.0.0.0 | inetmcast      | XMPP     | a3s19      | 10.84.17.5 | 0     | 100              | -          |
| 10.84.17.5:2:255.255.255.0,0.0.0.0  | inetmcast      | XMPP     | a3s19      | 10.84.17.5 | 0     | 100              | -          |

- We can see that VRF default-domain:demo:vn0:vn0 on Virtual Router a3s18 has the appropriate route and next hop to reach VRF default-domain:demo:front-end on Virtual Router a3s19; see [Figure 289 on page 967](#).

Figure 289: Verify Route and Next Hop a3s18

Monitor > Infrastructure > Virtual Routers > a3s18

Details Console Interfaces Networks ACL Flows Routes

VRF default-domain:demo:vn0:vn0 Show Routes  Unicast  Multicast

| Prefix               | Next ho... | Next hop details   |
|----------------------|------------|--|
| 169.254.169.254 / 32 | receive    | Source: MData Dest VN: default-domain:default-project:__link_local__ |
| 192.168.0.252 / 32   | interface  | Interface: tap1dae0121-4c Dest VN: default-domain:demo:vn0           |
|                      | interface  | Interface: tap1dae0121-4c Dest VN: default-domain:demo:vn0           |
|                      | interface  | Interface: tap1dae0121-4c Dest VN: default-domain:demo:vn0           |
| 192.168.0.253 / 32   | tunnel     | Dest IP: 10.84.17.5 Dest VN: default-domain:demo:vn0 Label: 16       |
|                      | tunnel     | Dest IP: 10.84.17.5 Dest VN: default-domain:demo:vn0 Label: 16       |
| 192.168.0.254 / 32   | interface  | Interface: pkt0 Dest VN: default-domain:demo:vn0                     |
| 192.168.16.252 / 32  | interface  | Interface: tap249de2e1-97 Dest VN: default-domain:demo:vn16          |
|                      | interface  | Interface: tap249de2e1-97 Dest VN: default-domain:demo:vn16          |
| 192.168.16.253 / 32  | tunnel     | Dest IP: 10.84.17.5 Dest VN: default-domain:demo:vn16 Label: 19      |

#041889

9. We can see that VRF default-domain:demo:vn16:vn16 on Virtual Router a3s19 has the appropriate route and next hop to reach VRF default-domain:demo:vn0:vn0 on Virtual Router a3s18; see [Figure 290 on page 968](#).

Figure 290: Verify Route and Next Hop a3s19

| Prefix               | Next ho... | Next hop details   |
|----------------------|------------|--|
| 169.254.169.254 / 32 | receive    | Source: MData Dest VN: default-domain:default-project:__link_local__ |
| 192.168.0.252 / 32   | tunnel     | Dest IP: 10.84.17.4 Dest VN: default-domain:demo:vn0 Label: 16       |
|                      | tunnel     | Dest IP: 10.84.17.4 Dest VN: default-domain:demo:vn0 Label: 16       |
| 192.168.0.253 / 32   | interface  | Interface: tape5ea97e3-55 Dest VN: default-domain:demo:vn0           |
|                      | interface  | Interface: tape5ea97e3-55 Dest VN: default-domain:demo:vn0           |
| 192.168.16.252 / 32  | tunnel     | Dest IP: 10.84.17.4 Dest VN: default-domain:demo:vn16 Label: 18      |
|                      | tunnel     | Dest IP: 10.84.17.4 Dest VN: default-domain:demo:vn16 Label: 18      |
| 192.168.16.253 / 32  | interface  | Interface: tap29585b2f-c2 Dest VN: default-domain:demo:vn16          |
|                      | interface  | Interface: tap29585b2f-c2 Dest VN: default-domain:demo:vn16          |
|                      | interface  | Interface: tap29585b2f-c2 Dest VN: default-domain:demo:vn16          |
| 192.168.16.254 / 32  | interface  | Interface: pkt0 Dest VN: default-domain:demo:vn16                    |

- Finally, flows between instances (IPs 192.168.0.252 and 192.168.16.253) can be verified on Virtual Routers a3s18 and a3s19; see [Figure 291 on page 968](#) and [Figure 292 on page 969](#).

Figure 291: Flows for a3s18

| Protocol | Source Network | Source IP      | Source Port | Destination Network | Destination IP | Destination Port | Bytes/Pkts   | Setup Time                  |
|----------|----------------|----------------|-------------|---------------------|----------------|------------------|--------------|-----------------------------|
| TCP      | vn0            | 192.168.0.252  | 43434       | vn16                | 192.168.16.253 | 9100             | 1884588/5417 | 21:00:22:131180 2013-Aug-06 |
| TCP      | vn16           | 192.168.16.253 | 9100        | vn0                 | 192.168.0.252  | 43434            | 1969668/5891 | 21:00:22:131193 2013-Aug-06 |
| TCP      | vn16           | 192.168.16.253 | 9101        | vn0                 | 192.168.0.252  | 53369            | 1903500/5805 | 21:00:22:206222 2013-Aug-06 |
| TCP      | vn0            | 192.168.0.252  | 53369       | vn16                | 192.168.16.253 | 9101             | 1890088/5302 | 21:00:22:206207 2013-Aug-06 |
| UDP      | vn0            | 192.168.0.252  | 39522       | vn16                | 192.168.16.252 | 9200             | 0/0          | 21:00:22:382861 2013-Aug-06 |
| UDP      | vn0            | 192.168.0.252  | 44794       | vn16                | 192.168.16.253 | 9201             | 1707392/3144 | 21:00:24:104277 2013-Aug-06 |
| UDP      | vn16           | 192.168.16.253 | 9201        | vn0                 | 192.168.0.252  | 44794            | 1735788/3107 | 21:00:24:104293 2013-Aug-06 |
| UDP      | vn0            | 192.168.0.252  | 40561       | vn16                | 192.168.16.253 | 9200             | 1693476/3067 | 21:00:22:037377 2013-Aug-06 |
| UDP      | vn16           | 192.168.16.253 | 9200        | vn0                 | 192.168.0.252  | 40561            | 1643324/3061 | 21:00:22:037387 2013-Aug-06 |
| UDP      | vn0            | 192.168.0.252  | 39522       | vn16                | 192.168.16.252 | 9200             | 1676616/3074 | 21:00:22:306703 2013-Aug-06 |
| TCP      | vn0            | 192.168.0.252  | 34236       | vn16                | 192.168.16.252 | 9100             | 1891368/5686 | 21:00:22:395695 2013-Aug-06 |
| TCP      | vn0            | 192.168.0.252  | 34236       | vn16                | 192.168.16.252 | 9100             | 0/0          | 21:00:22:400371 2013-Aug-06 |



Figure 292: Flows for a3s19

Monitor > Infrastructure > Virtual Routers > a3s19 Q Search

Details Console Interfaces Networks ACL Flows Routes Active Flows: 64

| Protocol | Source Network | Source IP      | Source Port | Destination Network | Destination IP | Destination Port | Bytes/Pkts   | Setup Time                  |
|----------|----------------|----------------|-------------|---------------------|----------------|------------------|--------------|-----------------------------|
| UDP      | vn0            | 192.168.0.252  | 44794       | vn16                | 192.168.16.253 | 9201             | 1069380/1975 | 21:00:24.111374 2013-Aug-06 |
| UDP      | vn16           | 192.168.16.253 | 9201        | vn0                 | 192.168.0.252  | 44794            | 1100604/1963 | 21:00:24.111380 2013-Aug-06 |
| UDP      | vn0            | 192.168.0.252  | 40561       | vn16                | 192.168.16.253 | 9200             | 1046756/1877 | 21:00:22.047747 2013-Aug-06 |
| UDP      | vn16           | 192.168.16.253 | 9200        | vn0                 | 192.168.0.253  | 47270            | 1061900/1921 | 21:00:25.373941 2013-Aug-06 |
| UDP      | vn16           | 192.168.16.253 | 9200        | vn0                 | 192.168.0.252  | 40561            | 1010568/1914 | 21:00:22.047756 2013-Aug-06 |
| TCP      | vn16           | 192.168.16.253 | 9100        | vn0                 | 192.168.0.253  | 53314            | 1217772/3649 | 21:00:23.465564 2013-Aug-06 |
| TCP      | vn0            | 192.168.0.252  | 43434       | vn16                | 192.168.16.253 | 9100             | 1196536/3400 | 21:00:22.137665 2013-Aug-06 |
| TCP      | vn16           | 192.168.16.253 | 9100        | vn0                 | 192.168.0.252  | 43434            | 1239616/3724 | 21:00:22.137679 2013-Aug-06 |
| UDP      | vn16           | 192.168.16.253 | 9200        | vn0                 | 192.168.0.253  | 47270            | 0/0          | 21:00:25.347868 2013-Aug-06 |
| TCP      | vn16           | 192.168.16.253 | 9100        | vn0                 | 192.168.0.253  | 53314            | 0/0          | 21:00:23.440090 2013-Aug-06 |
| UDP      | vn16           | 192.168.16.253 | 9201        | vn0                 | 192.168.0.253  | 53930            | 1088692/1953 | 21:00:25.443166 2013-Aug-06 |
| TCP      | vn16           | 192.168.16.253 | 9101        | vn0                 | 192.168.0.253  | 34551            | 0/0          | 21:00:23.514246 2013-Aug-06 |
| TCP      | vn16           | 192.168.16.253 | 9101        | vn0                 | 192.168.0.253  | 34551            | 1304273/1664 | 21:00:23.514451 2013-Aug-06 |

# Common Support Answers

## IN THIS CHAPTER

- [Debugging Ping Failures for Policy-Connected Networks | 970](#)
- [Debugging BGP Peering and Route Exchange in Contrail | 978](#)
- [Troubleshooting the Floating IP Address Pool in Contrail | 996](#)
- [Removing Stale Virtual Machines and Virtual Machine Interfaces | 1025](#)
- [Troubleshooting Link-Local Services in Contrail | 1030](#)

## Debugging Ping Failures for Policy-Connected Networks

This topic presents troubleshooting scenarios and steps for resolving reachability issues (ping failures) when working with policy-connected virtual networks.

These are the methods used to configure reachability for a virtual network or virtual machine:

- Use network policy to exchange virtual network routes.
- Use a floating IP address pool to associate an IP address from a destination virtual network to virtual machine(s) in the source virtual network.
- Use an ASN/RT configuration to exchange virtual network routes with an MX Series router gateway.
- Use a service instance static route configuration to route between service instances in two virtual networks.

This topic focuses on troubleshooting reachability for the first method --- using network policy to exchange routes between virtual networks.

### *Troubleshooting Procedure for Policy-Connected Network*

#### 1. Check the state of the virtual machine and interface.

Before doing anything else, check the status of the source and destination virtual machines.

- Is the **Status** of each virtual machine **Up**?

- Are the corresponding tap interfaces **Active**?

Check the virtual machine status in the Contrail UI:

**Figure 293: Virtual Machine Status Window**



| Name           | Label | Status | Network     | IP Address | Floating IP ^  | Instance                                 |
|----------------|-------|--------|-------------|------------|----------------|--|
| tapb80d9c6a-67 | 16    | Up     | vn1 (admin) | 31.1.1.253 | 10.204.219.108 | 54533ef-403e-40b0-bc47-6d748e6f0c8 / vn1 |

Check the tap interface status in the http agent introspect, for example: [http://nodef1.englab.juniper.net:8085/Snh\\_ItfReq?name=](http://nodef1.englab.juniper.net:8085/Snh_ItfReq?name=)

**Figure 294: Tap Interface Status Window**



| Index | name           | uuid                                 | vrf_name                     | active |
|-------|----------------|--------------------------------------|------------------------------|--------|
| 4     | tapb80d9c6a-67 | b80d9c6a-672e-4c1e-9b83-e053d6c911ab | default-domain:admin:vn1:vn1 | Active |

When the virtual machine status is verified **Up**, and the tap interface is **Active**, you can focus on other factors that affect traffic, including routing, network policy, security policy, and service instances with static routes.

## 2. Check reachability and routing.

Use the following troubleshooting guidelines whenever you are experiencing ping failures on virtual network routes that are connected by means of network policy.

Check the network policy configuration:

- Verify that the policy is attached to each of the virtual networks.
- Each attached policy should have either an explicit rule allowing traffic from one virtual network to the other, or an allow all traffic rule.
- Verify that the order of the actions in the policy rules is correct, because the actions are applied in the order in which they are listed.
- If there are multiple policies attached to a virtual network, verify that the policies are attached in a logical order. The first policy listed is applied first, and its rules are applied first, then the next policy is applied.

- Finally, if either of the virtual networks does not have an explicit rule to allow traffic from the other virtual network, the traffic flow will be treated as an **UNRESOLVED** or **SHORT** flow and all packets will be dropped.

Use the following sequence in the Contrail UI to check policies, attachments, and traffic rules:

Check VN1-VN2 ACL information from the compute node:

**Figure 295: Policies, Attachments, and Traffic Rule Status Window**

| UUID                                | Flows  | Action | Protocol | Source Network or Prefix | Source Port | Destination Network or Prefix | Destination Port | ACE Id |
|-------------------------------------|--------|--------|----------|--------------------------|-------------|-------------------------------|------------------|--------|
| 8b0329d7-ad9e-41ac-a2e-30f4dbc2b5ae | 100000 | pass   | 1-1      | default-domainadminvn1   | any         | default-domainadminvn2        | any              | 1      |
|                                     |        | pass   | 1-1      | default-domainadminvn2   | any         | default-domainadminvn1        | any              | 2      |
|                                     |        | pass   | any      | default-domainadminvn1   | any         | default-domainadminvn1        | any              | 3      |
|                                     |        | deny   | any      | default-domainadminvn1   | any         | default-domainadminvn2        | any              | 4      |
|                                     |        | deny   | any      | default-domainadminvn2   | any         | default-domainadminvn1        | any              | 5      |

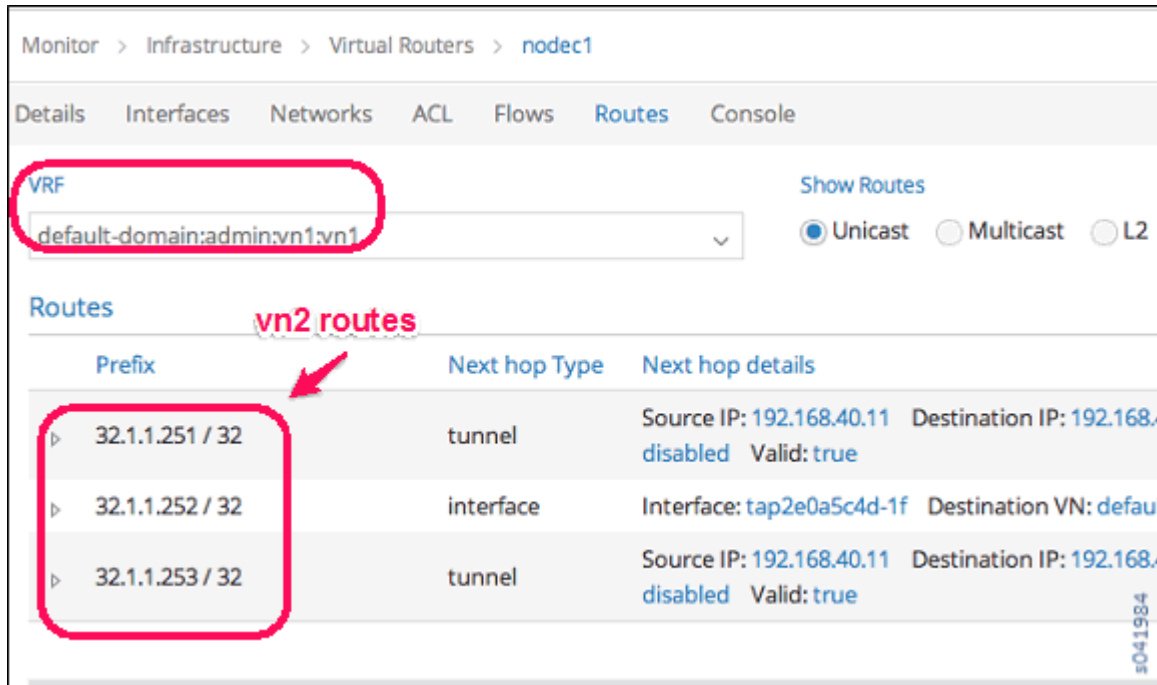
Check the virtual network policy configuration with route information:

**Figure 296: Virtual Network Policy Configuration Window**

| Network | Attached Policies                           | IP Blocks   |
|---------|---|-------------|
| vn1     | default-analyzer-analyzer-policy<br>vn1-vn2 | 31.1.1.0/24 |
| vn2     | allow_all                                   | 32.1.1.0/24 |

Check the VN1 route information for VN2 routes:

Figure 297: Virtual Network Route Information Window

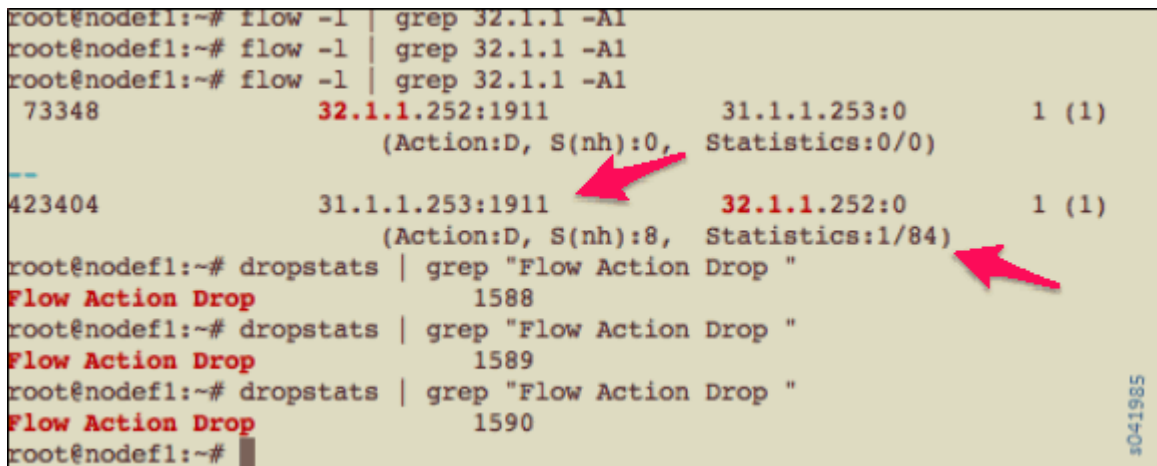


If a route is missing, ping fails. Flow inspection in the compute node displays **Action: D(drop)**.

Repeated dropstats commands confirms the drop by incrementing the **Flow Action Drop** counter with each iteration of dropstats.

Flow and dropstats commands issued at the compute node:

Figure 298: Flow and Dropstats Command List



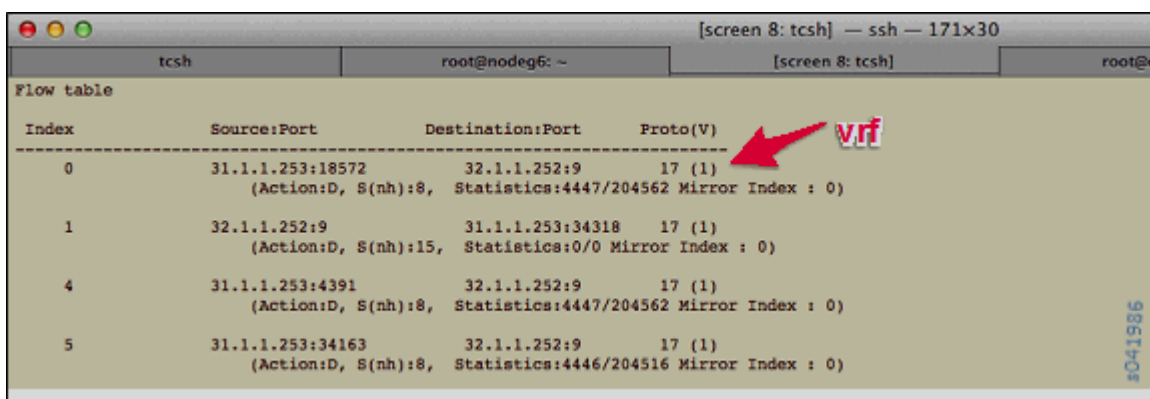
To help in debugging flows, you can use the detailed flow query from the agent introspect page for the compute node.

Fields of interest include:

- Inputs [from **flow -l** output]: **src/dest ip, src/dest ports, protocol, and vrf**
- Output from detailed flow query: **short\_flow, src\_vn, action\_str->action**

Flow command output:

Figure 299: Flow Command Output Window



| Index | Source:Port  | Destination:Port | Proto(V) | vrf |
|-------|--|------------------|----------|-----|
| 0     | 31.1.1.253:18572<br>(Action:D, S(nh):8, Statistics:4447/204562 Mirror Index : 0) | 32.1.1.252:9     | 17 (1)   |     |
| 1     | 32.1.1.252:9<br>(Action:D, S(nh):15, Statistics:0/0 Mirror Index : 0)            | 31.1.1.253:34318 | 17 (1)   |     |
| 4     | 31.1.1.253:4391<br>(Action:D, S(nh):8, Statistics:4447/204562 Mirror Index : 0)  | 32.1.1.252:9     | 17 (1)   |     |
| 5     | 31.1.1.253:34163<br>(Action:D, S(nh):8, Statistics:4446/204516 Mirror Index : 0) | 32.1.1.252:9     | 17 (1)   |     |

Fetching details of a single flow:

Figure 300: Fetch Flow Record Window

nodef1.englab.juniper.net:8085/pkt.xml#5nh\_FetchAllFlowRecords

Controller HTTP Introspect HTTP Introspect lists.opencontrail.org Ma... The Users M

### FetchFlowRecord

vrf(i32)

sip(string)

dip(string)

src\_port(i32)

dst\_port(i32)

protocol(byte)

s041987

Output from **FetchFlowRecord** shows unresolved IP addresses:

Figure 301: Unresolved IP Address Window

|                |                  |
|----------------|------------------|
| implicit_deny  | no               |
| short_flow     | yes              |
| setup_time_utc | 1394959054698162 |
| local_flow     | no               |
| src_vn         | __UNKNOWN__      |
| dst_vn         | __UNKNOWN__      |
| reverse_flow   | no               |

```

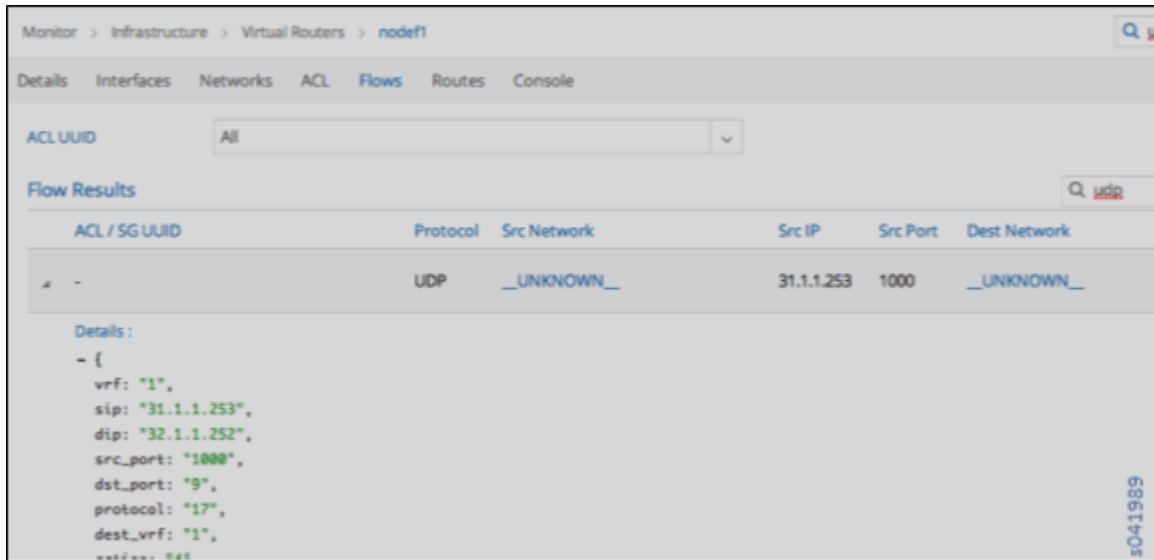
    },
    action_str: - {
      list: - {
        ActionStr: - {
          action: "drop"
        }
      }
    }
  }

```

s041988

You can also retrieve information about unresolved flows from the Contrail UI, as shown in the following:

Figure 302: Unresolved Flow Details Window

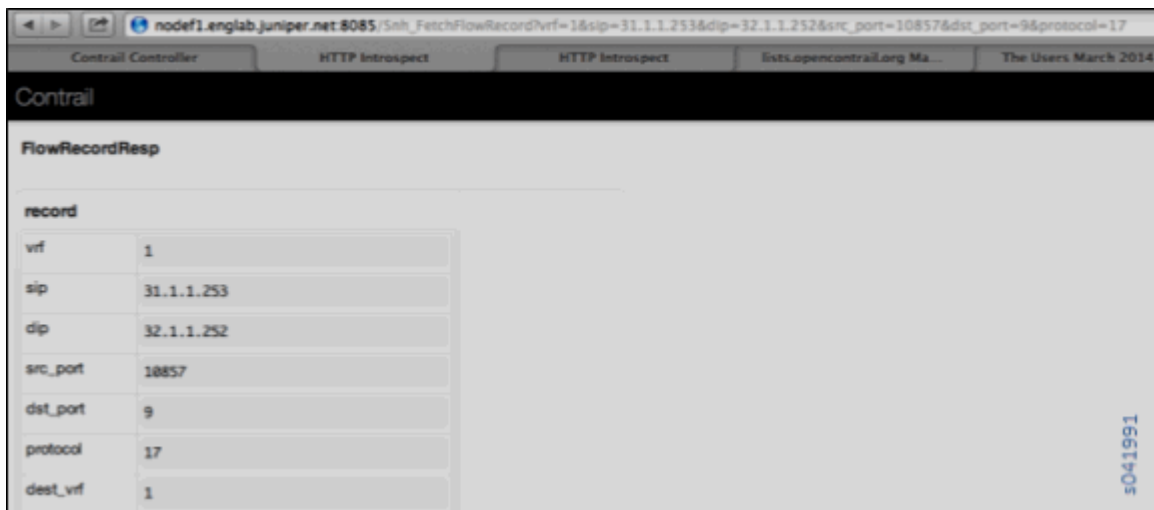


3. Check for protocol-specific network policy action.

If you are still experiencing reachability issues, troubleshoot any protocol-specific action, where routes are exchanged, but only specific protocols are allowed.

The following shows a sample query on a protocol-specific flow in the agent introspect:

Figure 303: Protocol-Specific Flow Sample



The following shows that although the virtual networks are resolved (not `__UNKNOWN__`), and not a short flow (the flow entry exists for a defined aging time), the policy action clearly displays **deny** as the action.



Figure 304: Protocol-Specific Flow Sample With Deny Action

|                   |   |                   |  |        |                                      |     |   |            |  |      |                                      |            |  |                   |  |        |      |
|-------------------|---|-------------------|--|--------|--------------------------------------|-----|---|------------|--|------|--------------------------------------|------------|--|-------------------|--|--------|------|
| implicit_deny     | no  |                   |  |        |                                      |     |   |            |  |      |                                      |            |  |                   |  |        |      |
| short_flow        | no  |                   |  |        |                                      |     |   |            |  |      |                                      |            |  |                   |  |        |      |
| setup_time_utc    | 1394972834710415  |                   |  |        |                                      |     |   |            |  |      |                                      |            |  |                   |  |        |      |
| local_flow        | no  |                   |  |        |                                      |     |   |            |  |      |                                      |            |  |                   |  |        |      |
| src_vn            | default-domain:admin:vn1  |                   |  |        |                                      |     |   |            |  |      |                                      |            |  |                   |  |        |      |
| dst_vn            | default-domain:admin:vn2  |                   |  |        |                                      |     |   |            |  |      |                                      |            |  |                   |  |        |      |
| reverse_flow      | no  |                   |  |        |                                      |     |   |            |  |      |                                      |            |  |                   |  |        |      |
| policy            | <table border="1"> <tr> <td colspan="2"><b>policy</b></td> </tr> <tr> <td>action</td> <td>g</td> </tr> <tr> <td>acl</td> <td> <table border="1"> <tr> <td colspan="2"><b>acl</b></td> </tr> <tr> <td>uuid</td> <td>8b0329d7-ad9e-41ac-af2e-30f4dbc2b50e</td> </tr> </table> </td> </tr> <tr> <td>action_str</td> <td> <table border="1"> <tr> <td colspan="2"><b>action_str</b></td> </tr> <tr> <td>action</td> <td>deny</td> </tr> </table> </td> </tr> </table> | <b>policy</b>     |  | action | g                                    | acl | <table border="1"> <tr> <td colspan="2"><b>acl</b></td> </tr> <tr> <td>uuid</td> <td>8b0329d7-ad9e-41ac-af2e-30f4dbc2b50e</td> </tr> </table> | <b>acl</b> |  | uuid | 8b0329d7-ad9e-41ac-af2e-30f4dbc2b50e | action_str | <table border="1"> <tr> <td colspan="2"><b>action_str</b></td> </tr> <tr> <td>action</td> <td>deny</td> </tr> </table> | <b>action_str</b> |  | action | deny |
| <b>policy</b>     |   |                   |  |        |                                      |     |   |            |  |      |                                      |            |  |                   |  |        |      |
| action            | g   |                   |  |        |                                      |     |   |            |  |      |                                      |            |  |                   |  |        |      |
| acl               | <table border="1"> <tr> <td colspan="2"><b>acl</b></td> </tr> <tr> <td>uuid</td> <td>8b0329d7-ad9e-41ac-af2e-30f4dbc2b50e</td> </tr> </table>   | <b>acl</b>        |  | uuid   | 8b0329d7-ad9e-41ac-af2e-30f4dbc2b50e |     |   |            |  |      |                                      |            |  |                   |  |        |      |
| <b>acl</b>        |   |                   |  |        |                                      |     |   |            |  |      |                                      |            |  |                   |  |        |      |
| uuid              | 8b0329d7-ad9e-41ac-af2e-30f4dbc2b50e  |                   |  |        |                                      |     |   |            |  |      |                                      |            |  |                   |  |        |      |
| action_str        | <table border="1"> <tr> <td colspan="2"><b>action_str</b></td> </tr> <tr> <td>action</td> <td>deny</td> </tr> </table>  | <b>action_str</b> |  | action | deny                                 |     |   |            |  |      |                                      |            |  |                   |  |        |      |
| <b>action_str</b> |   |                   |  |        |                                      |     |   |            |  |      |                                      |            |  |                   |  |        |      |
| action            | deny  |                   |  |        |                                      |     |   |            |  |      |                                      |            |  |                   |  |        |      |

### Summary

This topic explores one area –debugging for policy-based routing. However, in a complex system, a virtual network might have one or more configuration methods combined that influence reachability and routing.

For example, an environment might have a virtual network VN-X configured with policy-based routing to another virtual network VN-Y. At the same time, there are a few virtual machines in VN-X that have a floating IP to another virtual network VN-Z, which is connected to VN-XX via a NAT service instance. This is a complex scenario, and you need to debug step-by-step, taking into account all of the features working together.

Additionally, there are other considerations beyond routing and reachability that can affect traffic flow. For example, the rules of network policies and security groups can affect traffic to the destination. Also, if multi-path is involved, then ECMP and RPF need to be taken into account while debugging.

## Debugging BGP Peering and Route Exchange in Contrail

### IN THIS SECTION

- [Example Cluster | 978](#)
- [Verifying the BGP Routers | 978](#)
- [Verifying the Route Exchange | 982](#)
- [Debugging Route Exchange with Policies | 984](#)
- [Debugging Peering with an MX Series Router | 986](#)
- [Debugging a BGP Peer Down Error with Incorrect Family | 988](#)
- [Configuring MX Peering \(iBGP\) | 990](#)
- [Checking Route Exchange with an MX Series Peer | 992](#)
- [Checking the Route in the MX Series Router | 994](#)

Use the troubleshooting steps and guidelines in this topic when you have errors with Contrail BGP peering and route exchange.

### Example Cluster

Examples in this document refer to a virtual cluster that is set up as follows:

```
Config Nodes : ['nodea22', 'nodea20']

Control Nodes : ['nodea22', 'nodea20']

Compute Nodes : ['nodea22', 'nodea20']

Collector : ['nodea22']

WebU : nodea22

Openstack : nodea22
```

### Verifying the BGP Routers

Use this procedure to launch various introspects to verify the setup of the BGP routers in your system.

Use this procedure to launch various introspects to verify the setup of the BGP routers in your system.

### 1. Verify the BGP routers.

All of the configured control nodes and external BGP routers are visible from the following location, shown using the sample node setup.

http: //<host ip address>:8082/bgp-routers

**NOTE:** Throughout this procedure, replace <host ip address> with the correct location for your system to see the setup in your system.

Figure 305: Sample Output, BGP Routers:

```
{
  - bgp-routers: [
    - {
      href: "http://nodea22.englab.juniper.net:8082/bgp-router/1da579c5-0907-4c98-a7ad-37671f00cf60",
      - fq_name: [
        "default-domain",
        "default-project",
        "ip-fabric",
        "__default__",
        "nodea20"
      ],
      uuid: "1da579c5-0907-4c98-a7ad-37671f00cf60"
    },
    - {
      href: "http://nodea22.englab.juniper.net:8082/bgp-router/9702853f-5e48-417f-bd72-c00a12cc0200",
      - fq_name: [
        "default-domain",
        "default-project",
        "ip-fabric",
        "__default__",
        "nodea22"
      ],
      uuid: "9702853f-5e48-417f-bd72-c00a12cc0200"
    }
  ]
}
```

5041946

### 2. Verify the BGP peering.

The following statement is entered to check the `bgp_router_refs` object on the API server to validate the peering on the sample setup.

http: //<host ip address>:8082/bgp-router/1da579c5-0907-4c98-a7ad-37671f00cf60

Figure 306: Sample Output, BGP Router References:

```

- bgp_router_parameters: {
  vendor: "contrail",
  autonomous_system: 64512,
  vnc_managed: null,
  address: "10.204.216.16",
  identifier: "10.204.216.16",
  port: 179,
  - address_families: {
    - family: {
      "inet-vpn",
      "e-vpn"
    }
  }
},
- bgp_router_refs: [
  - {
    - to: {
      "default-domain",
      "default-project",
      "ip-fabric",
      "__default__",
      "nodea22"
    },
    href: "http://nodea22.enlab.juniper.net:8082/bgp-router/9702853f-5e48-417f-bd72-c00a12cc0200",
    - attr: {
      - session: [
        - {
          - attributes: [
            - {
              bgp_router: null,
              - address_families: {
                - family: {
                  "inet-vpn",
                  "e-vpn"
                }
              }
            },
            ],
          uid: null
        }
      ],
      uid: "9702853f-5e48-417f-bd72-c00a12cc0200"
    }
  },
],
],

```

s041947

### 3. Verify the command line arguments that are passed to the control-node.

On the control-node, use `ps aux | grep control-node` to see the arguments that are passed to the control-node.

#### Example

```

/usr/bin/control-node --map-user <ip address> --map-password <ip address>--hostname nodea22 --
host-ip <ip address> --bgp-port 179 --discovery-server <ip address>

```

The hostname is the `bgp-router` name. Ensure that the `bgp-router` config can be found for the hostname, using the procedure in Step 1.

### 4. Validate the BGP neighbor config and the BGP peering config object.

`http: //<host ip address>:8083/Snh_ShowBgpNeighborConfigReq?`

Figure 307: Sample Output, BGP Neighbor Config:

| ShowBgpNeighborConfigResp                              |   |          |                   |               |               |                                       |
|--|---|----------|-------------------|---------------|---------------|---------------------------------------|
| neighbors  |   |          |                   |               |               |                                       |
| instance_name  | name  | vendor   | autonomous_system | identifier    | address       | address_families                      |
| default-domain:default-project:ip-fabric:...default... | default-domain:default-project:ip-fabric:...nodes20 | controll | 64512             | 10.204.216.16 | 10.204.216.16 | address_families<br>inet-vpn<br>e-vpn |

http: //<host ip address>:8083/Snh\_ShowBgpPeeringConfigReq?

Figure 308: Sample Output, BGP Peering Config:

| ShowBgpPeeringConfigResp                               |   |                |          |  |
|--|---|----------------|----------|--|
| peerings   |   |                |          |  |
| instance_name  | name  | neighbor_count | sessions |  |
| default-domain:default-project:ip-fabric:...default... | attr(default-domain:default-project:ip-fabric:...nodes20,default-domain:default-project:ip-fabric:...nodes22) | 1              | sessions | uid attributes<br>attributes<br>bgp_router address_families<br>address_families<br>inet-vpn<br>e-vpn |

5. Check the BGP neighbor states on the sample setup.

http: //<host ip address>:8083/Snh\_BgpNeighborReq?ip\_address=&domain=

Figure 309: Sample Output, BGP Neighbor States:

| BgpNeighborListResp |                |          |               |           |          |           |             |            |                     |             |                             |                                    |
|---------------------|----------------|----------|---------------|-----------|----------|-----------|-------------|------------|---------------------|-------------|-----------------------------|------------------------------------|
| neighbors           |                |          |               |           |          |           |             |            |                     |             |                             |                                    |
| peer                | peer_address   | peer_asn | local_address | local_asn | encoding | peer_type | state       | send_state | last_event          | last_state  | last_state_at               | last_error                         |
| 10.204.216.16       | 10.204.216.16  | 64512    | 10.204.216.18 | 64512     | BCP      | Internal  | Established | In sync    | fas::EvBgpKeepalive | OpenConfirm | 2014-Feb-10 07:08:21.177692 | Unknown                            |
| 10.204.216.253      | 10.204.216.253 | 64512    | 10.204.216.18 | 64512     | BCP      | Internal  | Established | In sync    | fas::EvBgpUpdate    | OpenConfirm | 2014-Feb-10 11:24:45.851632 | Cease:Administrator reset the peer |

If the peer is not in an established state, check the **last\_error** and the **flap\_count**. Debug the BGP state machine by using information displayed in the output, such as **last\_state** and **last\_event**.

**NOTE:** The image displayed is truncated to fit this page. On the console screen you can scroll horizontally to see more columns and data.

### Verifying the Route Exchange

The following two virtual networks are used in the sample debugging session for route exchange.

```
vn1 -> 1.1.1.0/24

vn2 -> 2.2.2.0/24
```

#### Example Procedure for Verifying Route Exchange

1. Validate the presence of the routing instance for each virtual network in the sample system.

http ://<host ip address>:8083/Snh\_ShowRoutingInstanceReq?name=

**NOTE:** Throughout this example, replace <host ip address> with the correct location for the control node on your system.

Figure 310: Sample Output, Show Routing Instance:

| default-domain:demo:vn1:vn1 | default-domain:demo:vn1 | 4 | import_target  | export_target  | tables                             |         |
|-----------------------------|-------------------------|---|----------------|----------------|------------------------------------|---------|
|                             |                         |   | target:64512:1 | target:64512:1 | name                               | peers   |
|                             |                         |   |                |                | default-domain:demo:vn1:enet_0     | nodea20 |
|                             |                         |   |                |                | default-domain:demo:vn1:inet_0     | nodea20 |
|                             |                         |   |                |                | default-domain:demo:vn1:inetcast_0 | nodea20 |
|                             |                         |   |                |                |                                    |         |
| default-domain:demo:vn2:vn2 | default-domain:demo:vn2 | 5 | import_target  | export_target  | name                               | peers   |
|                             |                         |   | target:64512:2 | target:64512:2 | default-domain:demo:vn2:enet_0     | nodea22 |
|                             |                         |   |                |                | default-domain:demo:vn2:inet_0     | nodea22 |
|                             |                         |   |                |                | default-domain:demo:vn2:inetcast_0 | nodea22 |
|                             |                         |   |                |                |                                    |         |

In the sample output, you can see the **import\_target** and the **export\_target** configured on the routing instance. Also shown are the **xmpp peers (vroutes)** registered to the table.

The user can click on the **inet** table of the required routing instance to display the routes that belong to the instance.

Use the information in Step 2 to validate a route.

2. Validate a route in a given routing instance in the sample setup:

http ://<host ip address>:8083/Snh\_ShowRouteReq?x=default-domain:demo:vn1:vn1.inet.0

In the following sample output (truncated), the user can validate the BGP paths for the protocol and for the source of the route to verify which XMPP agent or vRouter has pushed the route. If the path source is BGP, the route is imported to the VRF table from a BGP peer, either another control-node or an external bgp router such as an MX Series router. BGP paths are displayed in the order of path selection.

Figure 311: Sample Output, Validate Route:

| ShowRouteReq                |                                    |   |          |               |                 |                  |   |        |               |       |              |                             |   |          |      |     |
|-----------------------------|------------------------------------|---|----------|---------------|-----------------|------------------|---|--------|---------------|-------|--------------|-----------------------------|---|----------|------|-----|
| tables                      |                                    |   |          |               |                 |                  |   |        |               |       |              |                             |   |          |      |     |
| routing_instance            | routing_table_name                 | prefixes  | paths    | primary_paths | secondary_paths | infeasible_paths | routes  |        |               |       |              |                             |   |          |      |     |
| default-domain:demo:vn1:vn1 | default-domain:demo:vn1:vn1.inet.0 | 1   | 2        | 1             | 1               | 0                | <table border="1"> <thead> <tr> <th>prefix</th> <th>last_modified</th> <th>paths</th> </tr> </thead> <tbody> <tr> <td>1.1.1.253/32</td> <td>2014-Feb-10 11:34:12.227200</td> <td> <table border="1"> <thead> <tr> <th>protocol</th> </tr> </thead> <tbody> <tr> <td>XMPP</td> </tr> <tr> <td>BGP</td> </tr> </tbody> </table> </td> </tr> </tbody> </table> | prefix | last_modified | paths | 1.1.1.253/32 | 2014-Feb-10 11:34:12.227200 | <table border="1"> <thead> <tr> <th>protocol</th> </tr> </thead> <tbody> <tr> <td>XMPP</td> </tr> <tr> <td>BGP</td> </tr> </tbody> </table> | protocol | XMPP | BGP |
| prefix                      | last_modified                      | paths   |          |               |                 |                  |   |        |               |       |              |                             |   |          |      |     |
| 1.1.1.253/32                | 2014-Feb-10 11:34:12.227200        | <table border="1"> <thead> <tr> <th>protocol</th> </tr> </thead> <tbody> <tr> <td>XMPP</td> </tr> <tr> <td>BGP</td> </tr> </tbody> </table> | protocol | XMPP          | BGP             |                  |   |        |               |       |              |                             |   |          |      |     |
| protocol                    |                                    |   |          |               |                 |                  |   |        |               |       |              |                             |   |          |      |     |
| XMPP                        |                                    |   |          |               |                 |                  |   |        |               |       |              |                             |   |          |      |     |
| BGP                         |                                    |   |          |               |                 |                  |   |        |               |       |              |                             |   |          |      |     |

3. Validate the l3vpn table.

http: //<host ip address>:8083/Snh\_ShowRouteReq?x=bgp.l3vpn.0

Figure 312: Sample Output, Validate L3vpn Table:

| ShowRouteReq  |                             |   |          |               |                 |                  |  |        |               |       |                              |                             |   |          |      |     |                              |                             |   |          |      |     |
|---|-----------------------------|---|----------|---------------|-----------------|------------------|--|--------|---------------|-------|------------------------------|-----------------------------|---|----------|------|-----|------------------------------|-----------------------------|---|----------|------|-----|
| tables  |                             |   |          |               |                 |                  |  |        |               |       |                              |                             |   |          |      |     |                              |                             |   |          |      |     |
| routing_instance                                    | routing_table_name          | prefixes  | paths    | primary_paths | secondary_paths | infeasible_paths | routes   |        |               |       |                              |                             |   |          |      |     |                              |                             |   |          |      |     |
| default-domain:default-project:ip-fabric:default... | l3vpn-0                     | 2   | 4        | 2             | 2               | 0                | <table border="1"> <thead> <tr> <th>prefix</th> <th>last_modified</th> <th>paths</th> </tr> </thead> <tbody> <tr> <td>10.204.216.10:1:1.1.1.253/32</td> <td>2014-Feb-10 11:34:12.227203</td> <td> <table border="1"> <thead> <tr> <th>protocol</th> </tr> </thead> <tbody> <tr> <td>XMPP</td> </tr> <tr> <td>BGP</td> </tr> </tbody> </table> </td> </tr> <tr> <td>10.204.216.10:1:2.2.2.253/32</td> <td>2014-Feb-10 11:34:35.409558</td> <td> <table border="1"> <thead> <tr> <th>protocol</th> </tr> </thead> <tbody> <tr> <td>XMPP</td> </tr> <tr> <td>BGP</td> </tr> </tbody> </table> </td> </tr> </tbody> </table> | prefix | last_modified | paths | 10.204.216.10:1:1.1.1.253/32 | 2014-Feb-10 11:34:12.227203 | <table border="1"> <thead> <tr> <th>protocol</th> </tr> </thead> <tbody> <tr> <td>XMPP</td> </tr> <tr> <td>BGP</td> </tr> </tbody> </table> | protocol | XMPP | BGP | 10.204.216.10:1:2.2.2.253/32 | 2014-Feb-10 11:34:35.409558 | <table border="1"> <thead> <tr> <th>protocol</th> </tr> </thead> <tbody> <tr> <td>XMPP</td> </tr> <tr> <td>BGP</td> </tr> </tbody> </table> | protocol | XMPP | BGP |
| prefix  | last_modified               | paths   |          |               |                 |                  |  |        |               |       |                              |                             |   |          |      |     |                              |                             |   |          |      |     |
| 10.204.216.10:1:1.1.1.253/32                        | 2014-Feb-10 11:34:12.227203 | <table border="1"> <thead> <tr> <th>protocol</th> </tr> </thead> <tbody> <tr> <td>XMPP</td> </tr> <tr> <td>BGP</td> </tr> </tbody> </table> | protocol | XMPP          | BGP             |                  |  |        |               |       |                              |                             |   |          |      |     |                              |                             |   |          |      |     |
| protocol  |                             |   |          |               |                 |                  |  |        |               |       |                              |                             |   |          |      |     |                              |                             |   |          |      |     |
| XMPP  |                             |   |          |               |                 |                  |  |        |               |       |                              |                             |   |          |      |     |                              |                             |   |          |      |     |
| BGP   |                             |   |          |               |                 |                  |  |        |               |       |                              |                             |   |          |      |     |                              |                             |   |          |      |     |
| 10.204.216.10:1:2.2.2.253/32                        | 2014-Feb-10 11:34:35.409558 | <table border="1"> <thead> <tr> <th>protocol</th> </tr> </thead> <tbody> <tr> <td>XMPP</td> </tr> <tr> <td>BGP</td> </tr> </tbody> </table> | protocol | XMPP          | BGP             |                  |  |        |               |       |                              |                             |   |          |      |     |                              |                             |   |          |      |     |
| protocol  |                             |   |          |               |                 |                  |  |        |               |       |                              |                             |   |          |      |     |                              |                             |   |          |      |     |
| XMPP  |                             |   |          |               |                 |                  |  |        |               |       |                              |                             |   |          |      |     |                              |                             |   |          |      |     |
| BGP   |                             |   |          |               |                 |                  |  |        |               |       |                              |                             |   |          |      |     |                              |                             |   |          |      |     |

The following sample output has been scrolled horizontally to display the BGP path attributes of each route. policies.

The extended community (communities column), determines the VRF table to which this VPN route is imported. The **origin\_vn** shows the virtual network where this route was created, information useful for applying ACL

The label (MPLS) and tunnel encap columns can be used for debugging data path issues.

**Figure 313: Sample Output, Validate L3vpn Table, Scrolled:**

| source        | as_path | next_hop      | tunnel_encap               | label | replicated | primary_table                  | communities  | origin_vn               | flags |
|---------------|---------|---------------|----------------------------|-------|------------|--------------------------------|--|-------------------------|-------|
| nodea20       | *       | 10.204.216.16 | tunnel_encap<br>gre<br>udp | 16    | true       | default-domain:demo:vn1:inet.0 | communities<br>security group: 5<br>originvn:64512:4<br>target:64512:1 | default-domain:demo:vn1 | 0     |
| 10.204.216.16 | *       | 10.204.216.16 | tunnel_encap<br>gre<br>udp | 16    | false      |                                | communities<br>security group: 5<br>originvn:64512:4<br>target:64512:1 | default-domain:demo:vn1 | 0     |
| nodea22       | *       | 10.204.216.18 | tunnel_encap<br>gre<br>udp | 16    | true       | default-domain:demo:vn2:inet.0 | communities<br>security group: 5<br>originvn:64512:5<br>target:64512:2 | default-domain:demo:vn2 | 0     |
| 10.204.216.16 | *       | 10.204.216.18 | tunnel_encap<br>gre<br>udp | 16    | false      |                                | communities<br>security group: 5<br>originvn:64512:5<br>target:64512:2 | default-domain:demo:vn2 | 0     |

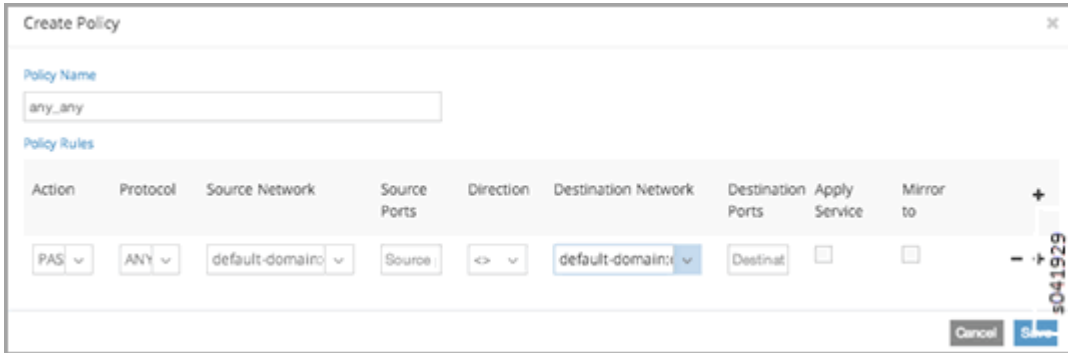
## Debugging Route Exchange with Policies

This section uses the sample output and the sample vn1 and vn2 to demonstrate methods of debugging route exchange with policies.

1. Create a network policy to allow vn1 and vn2 traffic and associate the policy to the virtual networks.



Figure 314: Create Policy Window



2. Validate that the routing instances have the correct import\_target configuration.

http: //<host ip address>:8083/Snh\_ShowRoutingInstanceReq?name=

Figure 315: Sample Output, Validate Import Target:

| default-domain:demo:vn1:vn1 | default-domain:demo:vn1 | 4 | import_target                    | export_target  |
|-----------------------------|-------------------------|---|----------------------------------|----------------|
|                             |                         |   | target:64512:1<br>target:64512:2 | target:64512:1 |
| default-domain:demo:vn2:vn2 | default-domain:demo:vn2 | 5 | import_target                    | export_target  |
|                             |                         |   | target:64512:1<br>target:64512:2 | target:64512:2 |

3. Validate that the routes are imported from VRF.

Use the BGP path attribute to check the replication status of the path. The route from the destination VRF should be replicated and validate the origin-vn.

Figure 316: Sample Output, Route Import:

| routing_instance        | routing_table_name             | prefixes | paths    | primary_paths | secondary_paths | infeasible_paths | routes   |        |               |       |          |              |                             |  |      |  |  |  |     |              |                             |  |      |  |  |  |     |
|-------------------------|--------------------------------|----------|----------|---------------|-----------------|------------------|--|--------|---------------|-------|----------|--------------|-----------------------------|--|------|--|--|--|-----|--------------|-----------------------------|--|------|--|--|--|-----|
| default-domain:demo:vn2 | default-domain:demo:vn2:inet.0 | 2        | 4        | 1             | 3               | 0                | <table border="1"> <thead> <tr> <th>prefix</th> <th>last_modified</th> <th>paths</th> <th>protocol</th> </tr> </thead> <tbody> <tr> <td>1.1.1.253/32</td> <td>2014-Feb-10 12:02:47.263344</td> <td></td> <td>XMPP</td> </tr> <tr> <td></td> <td></td> <td></td> <td>BGP</td> </tr> <tr> <td>2.2.2.253/32</td> <td>2014-Feb-10 11:34:35.469899</td> <td></td> <td>XMPP</td> </tr> <tr> <td></td> <td></td> <td></td> <td>BGP</td> </tr> </tbody> </table> | prefix | last_modified | paths | protocol | 1.1.1.253/32 | 2014-Feb-10 12:02:47.263344 |  | XMPP |  |  |  | BGP | 2.2.2.253/32 | 2014-Feb-10 11:34:35.469899 |  | XMPP |  |  |  | BGP |
| prefix                  | last_modified                  | paths    | protocol |               |                 |                  |  |        |               |       |          |              |                             |  |      |  |  |  |     |              |                             |  |      |  |  |  |     |
| 1.1.1.253/32            | 2014-Feb-10 12:02:47.263344    |          | XMPP     |               |                 |                  |  |        |               |       |          |              |                             |  |      |  |  |  |     |              |                             |  |      |  |  |  |     |
|                         |                                |          | BGP      |               |                 |                  |  |        |               |       |          |              |                             |  |      |  |  |  |     |              |                             |  |      |  |  |  |     |
| 2.2.2.253/32            | 2014-Feb-10 11:34:35.469899    |          | XMPP     |               |                 |                  |  |        |               |       |          |              |                             |  |      |  |  |  |     |              |                             |  |      |  |  |  |     |
|                         |                                |          | BGP      |               |                 |                  |  |        |               |       |          |              |                             |  |      |  |  |  |     |              |                             |  |      |  |  |  |     |

## Debugging Peering with an MX Series Router

This section sets up an example BGP MX Series peer and provides some troubleshooting scenarios.

1. Set the Global AS number of the control-node for an MX Series BGP peer, using the Contrail WebUI (eBGP).

Figure 317: Edit Global ASN Window

2. Configure the eBGP peer for the MX Series router. Use the Contrail Web UI or Python provisioning.

Figure 318: Create BGP Peer Window

Configuring the MX Series BGP peer with the Python provision utility:

```
python ./provision_mx.py --router_name mx --router_ip <ip address> --router_asn 12345 --
api_server_ip <ip address> --api_server_port 8082 --oper add --admin_user admin --
admin_password <password> --admin_tenant_name admin
```

### 3. Configure a control-node peer on the MX Series router, using Junos CLI:

```
set protocols bgp group contrail-control-nodes type external

set protocols bgp group contrail-control-nodes local-address <ip address>

set protocols bgp group contrail-control-nodes keep all

set protocols bgp group contrail-control-nodes peer-as 54321

set protocols bgp group contrail-control-nodes local-as 12345

set protocols bgp group contrail-control-nodes neighbor <ip address>
```

## Debugging a BGP Peer Down Error with Incorrect Family

Use this procedure to identify and resolve errors that arise from *families* mismatched configurations.

**NOTE:** This example uses locations at `http://<host ip address>:`. Be sure to replace `<host ip address>` with the correct address for your environment.

### 1. Check the BGP peer UVE.

`http://<host ip address>:8081/analytics/uves/bgp-peers`

### 2. Search for the MX Series BGP peer by name in the list.

In the sample output, *families* is the family advertised by the peer and *configured\_families* is what is provisioned. In the sample output, the families configured on the peer has a mismatch, thus the peer doesn't move to an established state. You can verify it in the peer UVE.

Figure 319: Sample BGP Peer UVE

```
{
- BgpPeerInfoData: {
- state_info: {
  last_state: "Idle",
  state: "Idle",
  last_state_at: 1394778927107639
},
- families: [
  "IPv4:Unicast"
],
peer_type: "external",
local_asn: 54321,
- configured_families: [
  "inet-vpn"
],
- event_info: {
  last_event_at: 1394778927107880,
  last_event: "fsm::EvStart"
},
local_id: 181196816,
send_state: "not advertising",
peer_address: "10.204.216.253",
peer_id: 181197053,
hold_time: 90,
peer_asn: 12345
}
}
```

### 3. Fix the families mismatch in the sample by updating the configuration on the MX Series router, using Junos CLI:

```
set protocols bgp group contrail-control-nodes family inet-vpn unicast
```

### 4. After committing the CLI configuration, the peer comes up. Verify this with UVE.

`http://<host ip address>:8081/analytics/uves/bgp-peers`

Figure 320: Sample Established BGP Peer UVE

```

{
  - BgpPeerInfoData: {
    - state_info: {
      last_state: "OpenConfirm",
      state: "Established",
      last_state_at: 1394779652932460
    },
    - families: [
      "IPv4:Vpn"
    ],
    peer_type: "external",
    local_asn: 54321,
    - configured_families: [
      "inet-vpn"
    ],
    - event_info: {
      last_event_at: 1394779652992071,
      last_event: "fsm::EvBgpUpdate"
    },
    local_id: 181196816,
    send_state: "in sync",
    peer_address: "10.204.216.253",
    peer_id: 181197053,
    peer_asn: 12345
  }
}

```

s041335

5. Verify the peer status on the MX Series router, using Junos CLI:

```

run show bgp neighbor <ip address>
Peer: <ip address> AS 54321 Local: <ip address> AS 12345

Type: External   State: Established   Flags: <ImportEval Sync>

Last State: OpenConfirm   Last Event: RecvKeepAlive

Last Error: None

Options: <Preference LocalAddress KeepAll AddressFamily PeerAS LocalAS Rib-group Refresh>

Address families configured: inet-vpn-unicast

Local Address: <ip address> Holdtime: 90 Preference: 170 Local AS: 12345 Local System AS:
64512

Number of flaps: 0

Error: 'Cease' Sent: 0 Recv: 2

```

```
Peer ID: <ip address>   Local ID: <ip address>   Active Holdtime: 90

Keepalive Interval: 30      Group index: 1   Peer index: 0

BFD: disabled, down

Local Interface: ge-1/0/2.0

NLRI for restart configured on peer: inet-vpn-unicast

NLRI advertised by peer: inet-vpn-unicast

NLRI for this session: inet-vpn-unicast

Peer does not support Refresh capability

Stale routes from peer are kept for: 300

Peer does not support Restarter functionality

Peer does not support Receiver functionality

Peer does not support 4 byte AS extension

Peer does not support Addpath
```

## Configuring MX Peering (iBGP)

1. Edit the Global ASN.

Figure 321: Edit Global ASN Window

2. Configure the MX Series IBGP peer, using Contrail WebUI or Python provisioning.

Figure 322: Create BGP Peer Window

Configuring the MX Series BGP peer with the Python provision utility:

```
python ./provision_mx.py --router_name mx--router_ip <ip address> --router_asn 64512 --api_server_ip <ip
address> --api_server_port 8082 --oper add --admin_user admin --admin_password <password> --admin_tenant_name
admin
```

3. Verify the peer from UVE.

```
http <://<host ip address>:8081/analytics/uves/bgp-peers
```

Figure 323: Sample Established IBGP Peer UVE

```

{
- BgpPeerInfoData: {
- state_info: {
  last_state: "OpenConfirm",
  state: "Established",
  last_state_at: 1394788178225128
},
- families: [
  "IPv4:Vpn"
],
peer_type: "internal",
local_asn: 64512,
- configured_families: [
  "inet-vpn"
],
- event_info: {
  last_event_at: 1394788178267208,
  last_event: "fsm::EvBgpUpdate"
},
local_id: 181196816,
send_state: "in sync",
peer_address: "10.204.216.253",
peer_id: 181197053,
peer_asn: 64512
}
}
    
```

4. You can verify the same information at the HTTP introspect page of the control node (8443 in this example).

http://<host ip address>:8083/Snh\_BgpNeighborReq?ip\_address=&domain=

Figure 324: Sample Established IBGP Peer Introspect Window

| peer           | peer_address   | peer_asn | local_address | local_asn | encoding | peer_type | state       | send_state | last_event          | last_state  |
|----------------|----------------|----------|---------------|-----------|----------|-----------|-------------|------------|---------------------|-------------|
| 10.204.216.253 | 10.204.216.253 | 64512    | 10.204.216.16 | 64512     | BGP      | internal  | Established | in sync    | fsm::EvBgpKeepalive | OpenConfirm |

### Checking Route Exchange with an MX Series Peer

1. Check the route table in the bgp.I3vpn.0 table.



Figure 325: Routing Instance Route Table

| routing_instance  | routing_table_name | prefixes | paths | primary_paths | secondary_paths | infeasible_paths | routes   |
|---|--------------------|----------|-------|---------------|-----------------|------------------|--|
| default-domain:default-project:ip-fabric:___default_... | bgp.13vgn.0        | 2        | 2     | 2             | 0               | 0                | routes<br>prefix<br>10.204.216.253:5:0.0.0.0/0<br>10.204.216.253:5:10.204.218.0/24 |

2. Configure a public virtual network.

Figure 326: Routing Instance Route Table

| routing_instance  | routing_table_name | prefixes | paths | primary_paths | secondary_paths | infeasible_paths | routes   |
|---|--------------------|----------|-------|---------------|-----------------|------------------|--|
| default-domain:default-project:ip-fabric:___default_... | bgp.13vgn.0        | 2        | 2     | 2             | 0               | 0                | routes<br>prefix<br>10.204.216.253:5:0.0.0.0/0<br>10.204.216.253:5:10.204.218.0/24 |

3. Verify the routes in the public.inet.0 table.

http: //<host ip address>:8083/Snh\_ShowRouteReq?x=default-domain:admin:public:public.inet.0

Figure 327: Routing Instance Public IPv4 Route Table

| tables | routing_instance                   | routing_table_name                        | prefixes | paths | primary_paths | secondary_paths | infeasible_paths | routes   |
|--------|------------------------------------|---|----------|-------|---------------|-----------------|------------------|--|
|        | default-domain:admin:public:public | default-domain:admin:public:public.inet.0 | 2        | 2     | 0             | 2               | 0                | routes<br>prefix<br>0.0.0.0/0<br>10.204.218.0/24 |

4. Launch a virtual machine in the public network and verify the route in the public.inet.0 table.

http: //<host ip address>:8083/ Snh\_ShowRouteReq?x=default-domain:admin:public:public.inet.0

**Figure 328: Virtual Machine Routing Instance Public IPv4 Route Table**

| tables                             |   |          |          |               |                 |                  |  |        |               |       |          |           |                             |  |     |                 |                             |  |     |               |                             |  |     |
|------------------------------------|---|----------|----------|---------------|-----------------|------------------|--|--------|---------------|-------|----------|-----------|-----------------------------|--|-----|-----------------|-----------------------------|--|-----|---------------|-----------------------------|--|-----|
| routing_instance                   | routing_table_name                        | prefixes | paths    | primary_paths | secondary_paths | infeasible_paths | routes   |        |               |       |          |           |                             |  |     |                 |                             |  |     |               |                             |  |     |
| default-domain:admin:public:public | default-domain:admin:public:public.inet.0 | 3        | 3        | 1             | 2               | 0                | <table border="1"> <thead> <tr> <th>prefix</th> <th>last_modified</th> <th>paths</th> <th>protocol</th> </tr> </thead> <tbody> <tr> <td>0.0.0.0/0</td> <td>2014-Mar-14 10:05:05.719026</td> <td></td> <td>BGP</td> </tr> <tr> <td>10.204.216.0/24</td> <td>2014-Mar-14 10:05:05.720017</td> <td></td> <td>BGP</td> </tr> <tr> <td>11.2.3.253/32</td> <td>2014-Mar-14 10:18:48.797958</td> <td></td> <td>BGP</td> </tr> </tbody> </table> | prefix | last_modified | paths | protocol | 0.0.0.0/0 | 2014-Mar-14 10:05:05.719026 |  | BGP | 10.204.216.0/24 | 2014-Mar-14 10:05:05.720017 |  | BGP | 11.2.3.253/32 | 2014-Mar-14 10:18:48.797958 |  | BGP |
| prefix                             | last_modified                             | paths    | protocol |               |                 |                  |  |        |               |       |          |           |                             |  |     |                 |                             |  |     |               |                             |  |     |
| 0.0.0.0/0                          | 2014-Mar-14 10:05:05.719026               |          | BGP      |               |                 |                  |  |        |               |       |          |           |                             |  |     |                 |                             |  |     |               |                             |  |     |
| 10.204.216.0/24                    | 2014-Mar-14 10:05:05.720017               |          | BGP      |               |                 |                  |  |        |               |       |          |           |                             |  |     |                 |                             |  |     |               |                             |  |     |
| 11.2.3.253/32                      | 2014-Mar-14 10:18:48.797958               |          | BGP      |               |                 |                  |  |        |               |       |          |           |                             |  |     |                 |                             |  |     |               |                             |  |     |

5. Verify the route in the bgp.l3vpn.0 table.

http: //<host ip address>:8083/Snh\_ShowRouteReq?x=bgp.l3vpn.0

**Figure 329: BGP Routing Instance Route Table**

| tables   |                    |          |       |               |                 |                  |  |        |                            |                                  |                               |
|--|--------------------|----------|-------|---------------|-----------------|------------------|--|--------|----------------------------|----------------------------------|-------------------------------|
| routing_instance                                       | routing_table_name | prefixes | paths | primary_paths | secondary_paths | infeasible_paths | routes   |        |                            |                                  |                               |
| default-domain:default-project:ip-fabric:___default... | bgp.l3vpn.0        | 3        | 3     | 2             | 1               | 0                | <table border="1"> <thead> <tr> <th>prefix</th> </tr> </thead> <tbody> <tr> <td>10.204.216.253:5:0.0.0.0/0</td> </tr> <tr> <td>10.204.216.253:5:10.204.216.0/24</td> </tr> <tr> <td>10.204.216.70:1:11.2.3.253/32</td> </tr> </tbody> </table> | prefix | 10.204.216.253:5:0.0.0.0/0 | 10.204.216.253:5:10.204.216.0/24 | 10.204.216.70:1:11.2.3.253/32 |
| prefix   |                    |          |       |               |                 |                  |  |        |                            |                                  |                               |
| 10.204.216.253:5:0.0.0.0/0                             |                    |          |       |               |                 |                  |  |        |                            |                                  |                               |
| 10.204.216.253:5:10.204.216.0/24                       |                    |          |       |               |                 |                  |  |        |                            |                                  |                               |
| 10.204.216.70:1:11.2.3.253/32                          |                    |          |       |               |                 |                  |  |        |                            |                                  |                               |

### Checking the Route in the MX Series Router

Use Junos CLI show commands from the router to check the route.

```
run show route table public.inet.0
```

```
public.inet.0: 5 destinations, 6 routes (5 active, 0 holddown, 0 hidden)
```

+ = Active Route, - = Last Active, \* = Both

0.0.0.0/0 \* [Static/5] 15w6d 08:50:34

> to <ip address> via ge-1/0/1.0

<ip address> \* [Direct/0] 15w6d 08:50:35

> via ge-1/0/1.0

<ip address> \* [Local/0] 15w6d 08:50:51

Local via ge-1/0/1.0

<ip address> \* [BGP/170] 01:13:34, localpref 100, from <ip address>

AS path: ?, validation-state: unverified

> via gr-1/0/0.32771, Push 16

[BGP/170] 01:13:34, localpref 100, from <ip address>

AS path: ?, validation-state: unverified

> via gr-1/0/0.32771, Push 16

<ip address> \* [BGP/170] 00:03:20, localpref 100, from <ip address>

AS path: ?, validation-state: unverified

> via gr-1/0/0.32769, Push 16

run show route table bgp.l3vpn.0 receive-protocol bgp <ip address> detail

bgp.l3vpn.0: 92 destinations, 130 routes (92 active, 0 holddown, 0 hidden)

\* <ip address> (1 entry, 0 announced)

Import Accepted

Route Distinguisher: <ip address>

VPN Label: 16

Nexthop: <ip address>

Localpref: 100

AS path: ?

Communities: target:64512:1 target:64512:10003 unknown iana 30c unknown iana 30c unknown  
type 8004 value fc00:1 unknown type 8071 value fc00:4

## Troubleshooting the Floating IP Address Pool in Contrail

### IN THIS SECTION

- [Example Cluster | 997](#)
- [Example | 998](#)
- [Example: MX80 Configuration for the Gateway | 999](#)
- [Ping the Floating IP from the Public Network | 1002](#)
- [Troubleshooting Details | 1003](#)
- [Get the UUID of the Virtual Network | 1003](#)
- [View the Floating IP Object in the API Server | 1004](#)
- [View floating-ips in floating-ip-pools in the API Server | 1008](#)
- [Check Floating IP Objects in the Virtual Machine Interface | 1011](#)
- [View the BGP Peer Status on the Control Node | 1015](#)
- [Querying Routes in the Public Virtual Network | 1016](#)
- [Verification from the MX80 Gateway | 1018](#)
- [Viewing the Compute Node Vnsw Agent | 1020](#)
- [Advanced Troubleshooting | 1022](#)

This document provides troubleshooting methods to use when you have errors with the floating IP address pool when using Contrail.

## Example Cluster

Examples in this document refer to a virtual cluster that is set up as follows:

```
Config Nodes : ['nodec6', 'nodec7', 'nodec8']  
  
Control Nodes : ['nodec7', 'nodec8']  
  
Compute Nodes : ['nodec9', 'nodec10']  
  
Collector      : ['nodec6', 'nodec8']  
  
WebUI          : nodec7  
  
Openstack      : nodec6
```

The following virtual networks are used in the examples in this document:

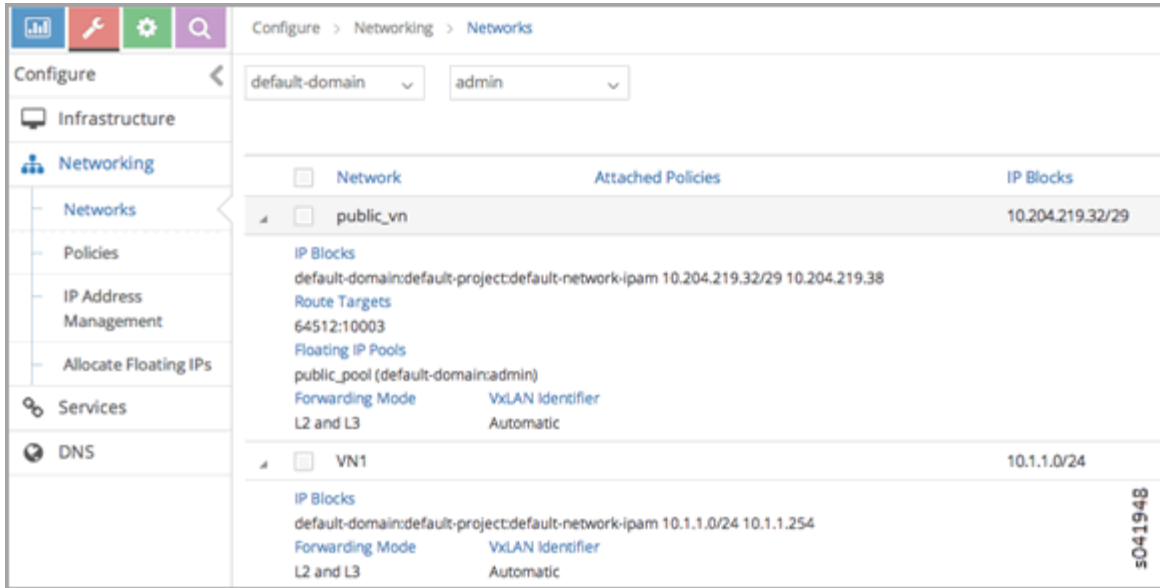
Public virtual network:

- Virtual network name: public\_vn
- Public addresses range: 10.204.219.32 to 10.204.219.37
- Route Target: 64512:10003
- Floating IP pool name: public\_pool

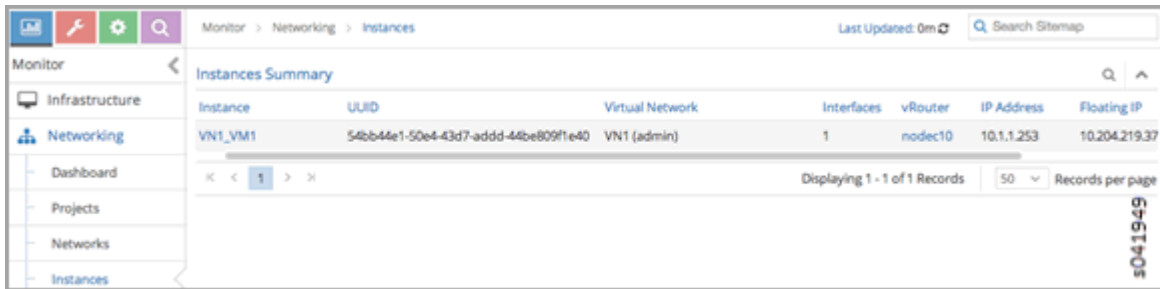
Private virtual network:

- Virtual network name: vn1
- Subnet: 10.1.1.0/24

### Example



A virtual machine is created in the virtual network VN1 with the name VN1\_VM1 and with the IP address 10.1.1.253. A floating IP address of 10.204.219.37 is associated to the VN1\_VM1 instance.



An MX80 router is configured as a gateway to peer with control nodes nodec7 and nodec8.

| IP Address  | Type         | Vendor   | HostName |
|---|--------------|----------|----------|
| 10.204.216.253  | BGP Peer     | mx       | mx1      |
| <b>Peers</b><br>Vendor: mx<br>BGP ASN: 64512<br>Router ID: 10.204.216.253<br>BGP Port: 179<br>Address family: inet-vpn          |              |          |          |
| 10.204.216.64   | Control Node | contrail | nodec7   |
| <b>Peers</b><br>Vendor: contrail<br>BGP ASN: 64512<br>Router ID: 10.204.216.64<br>BGP Port: 179<br>Address family: inet-vpn-vpn |              |          |          |
| 10.204.216.65   | Control Node | contrail | nodec8   |
| <b>Peers</b><br>Vendor: contrail<br>BGP ASN: 64512<br>Router ID: 10.204.216.65<br>BGP Port: 179<br>Address family: inet-vpn-vpn |              |          |          |

### Example: MX80 Configuration for the Gateway

The following is the Junos OS configuration for the MX80 gateway. The route 10.204.218.254 is the route to the external world.

```

chassis {

  fpc 1 {

    pic 0 {

      tunnel-services;

    }

  }

}

interfaces {

  ge-1/0/1 {

    unit 0 {
  
```

```
        family inet {
            address 10.204.218.1/24;
        }
    }
}

ge-1/0/2 {
    unit 0 {
        family inet {
            address 10.204.216.253/24;
        }
    }
}

routing-options {
    static {
        route 0.0.0.0/0 next-hop 10.204.216.254;
    }

    router-id 10.204.216.253;

    route-distinguisher-id 10.204.216.253;

    autonomous-system 64512;

    dynamic-tunnels {
        tun1 {
```





```
    }  
  }  
  
  routing-instances {  
    public {  
      instance-type vrf;  
  
      interface ge-1/0/1.0;  
  
      vrf-target target:64512:10003;  
  
      vrf-table-label;  
  
      routing-options {  
        static {  
          route 0.0.0.0/0 next-hop 10.204.218.254;  
        }  
      }  
    }  
  }  
}
```

## Ping the Floating IP from the Public Network

From the public network, ping the floating IP 10.204.219.37.

```
user1-test:~ user1$ ping 10.204.219.37  
  
PING 10.204.219.37 (10.204.219.37): 56 data bytes  
  
64 bytes from 10.204.219.37: icmp_seq=0 ttl=54 time=62.439 ms  
  
64 bytes from 10.204.219.37: icmp_seq=1 ttl=54 time=56.018 ms
```

```

64 bytes from 10.204.219.37: icmp_seq=2 ttl=54 time=55.915 ms

64 bytes from 10.204.219.37: icmp_seq=3 ttl=54 time=57.755 ms

^C

--- 10.204.219.37 ping statistics ---

5 packets transmitted, 4 packets received, 20.0% packet loss

round-trip min/avg/max/stddev = 55.915/58.032/62.439/2.647 ms

```

## Troubleshooting Details

The following sections show details of ways to get related information, view, troubleshoot, and validate floating IP addresses in a Contrail system.

### Get the UUID of the Virtual Network

Use the following to get the universal unique identifier (UUID) of the virtual network.

```

[root@nodec6 ~]# (source /etc/contrail/openstackrc; quantum net-list -F id -F name) 2>/dev/null

+-----+-----+
| id                | name                |
+-----+-----+
| 43707766-75f3-4d48-80d9-1b7240fb161d | public_vn          |
| 2ab7ea04-8f5f-4b8d-acbf-a7c29c9b4112 | VN1                |
| 1c59ded0-38e8-4168-b91f-4c51aba10d30 | default-virtual-network |
| 5b0a1040-91e4-47ff-bd4c-0a81e1901a1f | ip-fabric          |
| 7efddf64-ff3c-44d2-aeb2-45d7472b7a64 | __link_local__    |
+-----+-----+

```

## View the Floating IP Object in the API Server

Use the following to view the floating IP pool information in the API server. API server requests can be made on http port 8082.

The Contrail API servers have the virtual-network public\_vn object that contains floating IP pool information. Use the following to view the floating-ip-pools object information.

```
curl http://<API-Server_IP>:8082/virtual-network/<UUID_of_VN>
```

### *Example*

```
root@nodec6 ~]# curl http://nodec6:8082/virtual-network/43707766-75f3-4d48-80d9-1b7240fb161d |  
python -m json.tool
```

```
{  
  
  "virtual-network": {  
  
    "floating_ip_pools": [  
  
      {  
  
        "href": "http://127.0.0.1:8095/floating-ip-pool/663737c1-f3ab-40ff-9442-  
bdb6c225e3c3",  
  
        "to": [  
  
          "default-domain",  
  
          "admin",  
  
          "public_vn",  
  
          "public_pool"  
  
        ],  
  
        "uuid": "663737c1-f3ab-40ff-9442-bdb6c225e3c3"  
  
      }  
  
    ]  
  
  }  
  
}
```

```
],  
  
"fq_name": [  
    "default-domain",  
    "admin",  
    "public_vn"  
],  
  
"href": "http://127.0.0.1:8095/virtual-network/43707766-75f3-4d48-80d9-1b7240fb161d",  
  
"id_perms": {  
    "created": "2014-02-07T08:58:40.892803",  
    "description": null,  
    "enable": true,  
    "last_modified": "2014-02-07T10:06:42.234423",  
    "permissions": {  
        "group": "admin",  
        "group_access": 7,  
        "other_access": 7,  
        "owner": "admin",  
        "owner_access": 7  
    },  
  
    "uuid": {  
        "uuid_lslong": 9284482284331406877,  
        "uuid_mslong": 4859515279882014024
```

```
    }  
  
  },  
  
  "name": "public_vn",  
  
  "network_ipam_refs": [  
  
    {  
  
      "attr": {  
  
        "ipam_subnets": [  
  
          {  
  
            "default_gateway": "10.204.219.38",  
  
            "subnet": {  
  
              "ip_prefix": "10.204.219.32",  
  
              "ip_prefix_len": 29  
  
            }  
  
          }  
  
        ]  
  
      },  
  
      "href": "http://127.0.0.1:8095/network-ipam/39b0e8da-  
fcd4-4b35-856c-8d18570b1483",  
  
      "to": [  
  
        "default-domain",  
  
        "default-project",  
  
        "default-network-ipam"  
  
      ]  
  
    }  
  
  ]  
  
}
```

```
    ],  
  
    "uuid": "39b0e8da-fcd4-4b35-856c-8d18570b1483"  
  
  }  
  
],  
  
"parent_href": "http://127.0.0.1:8095/project/deef6549-8e6c-4e3e-9cde-c9bc2b72ce6f",  
  
"parent_type": "project",  
  
"parent_uuid": "deef6549-8e6c-4e3e-9cde-c9bc2b72ce6f",  
  
"route_target_list": {  
  
  "route_target": [  
  
    "target:64512:10003"  
  
  ]  
  
},  
  
"routing_instances": [  
  
  {  
  
    "href": "http://127.0.0.1:8095/routing-instance/3c6254ac-cfde-417e-916d-  
e7a1c0efad92",  
  
    "to": [  
  
      "default-domain",  
  
      "admin",  
  
      "public_vn",  
  
      "public_vn"  
  
    ],  
  
  }  
  
],
```

```

        "uuid": "3c6254ac-cfde-417e-916d-e7a1c0efad92"
    }
],
"uuid": "43707766-75f3-4d48-80d9-1b7240fb161d",
"virtual_network_properties": {
    "extend_to_external_routers": null,
    "forwarding_mode": "l2_l3",
    "network_id": 4,
    "vxlan_network_identifier": null
}
}
}
}

```

## View floating-ips in floating-ip-pools in the API Server

Once you have located the floating-ip-pools object, use the following to review its floating-ips object.

The floating-ips object should display the floating IP that is shown in the Contrail UI. The floating IP should have a reference to the virtual machine interface (VMI) object that is bound to the floating IP.

### *Example*

```

[root@nodec6 ~]# curl http://nodec6:8082/floating-ip-pool/663737c1-f3ab-40ff-9442-bdb6c225e3c3 |
python -m json.tool

```

```

{

```



```
"floating-ip-pool": {  
  "floating_ips": [  
    {  
      "href": "http://127.0.0.1:8095/floating-ip/f3eec4d6-889e-46a3-a8f0-879dfaff6ca0",  
      "to": [  
        "default-domain",  
        "admin",  
        "public_vn",  
        "public_pool",  
        "f3eec4d6-889e-46a3-a8f0-879dfaff6ca0"  
      ],  
      "uuid": "f3eec4d6-889e-46a3-a8f0-879dfaff6ca0"  
    }  
  ],  
  "fq_name": [  
    "default-domain",  
    "admin",  
    "public_vn",  
    "public_pool"  
  ],  
  "href": "http://127.0.0.1:8095/floating-ip-pool/663737c1-f3ab-40ff-9442-bdb6c225e3c3",  
  "id_perms": {
```

```
"created": "2014-02-07T08:58:41.136572",

"description": null,

"enable": true,

"last_modified": "2014-02-07T08:58:41.136572",

"permissions": {

    "group": "admin",

    "group_access": 7,

    "other_access": 7,

    "owner": "admin",

    "owner_access": 7

},

"uuid": {

    "uuid_lslong": 10683309858715198403,

    "uuid_mslong": 7365417021744038143

}

},

"name": "public_pool",

"parent_href": "http://127.0.0.1:8095/virtual-network/43707766-75f3-4d48-80d9-1b7240fb161d",

"parent_type": "virtual-network",

"parent_uuid": "43707766-75f3-4d48-80d9-1b7240fb161d",

"project_back_refs": [
```

```

    {
      "attr": {},
      "href": "http://127.0.0.1:8095/project/deef6549-8e6c-4e3e-9cde-c9bc2b72ce6f",
      "to": [
        "default-domain",
        "admin"
      ],
      "uuid": "deef6549-8e6c-4e3e-9cde-c9bc2b72ce6f"
    }
  ],
  "uuid": "663737c1-f3ab-40ff-9442-bdb6c225e3c3"
}
}

```

## Check Floating IP Objects in the Virtual Machine Interface

Use the following to retrieve the virtual machine interface of the virtual machine from either the `quantum port-list` command or from the Contrail UI. Then get the virtual machine interface identifier and check its floating IP object associations.

- Using `quantum port-list` to get the virtual machine interface:

### Example

```
[root@nodec6 ~]# quantum port-list -F id -F fixed_ips
```

```

+-----+
+-----+
| id                |

```

```
fixed_ips |
+-----+
+-----+

| cdca35ce-84ad-45da-9331-7bc67b7fcca6 | {"subnet_id":
"e80f480b-98d4-43cc-847c-711e637295db", "ip_address": "10.1.1.253"} |

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

- Using Contrail UI to get the virtual machine interface:



The screenshot shows the Contrail UI interface for monitoring a virtual machine instance. The breadcrumb path is "Monitor > Networking > Instances > VN1\_VM1". The left sidebar shows the navigation menu with "Networking" selected. The main content area displays the configuration for the virtual machine interface, including the floating IP object and the interface details.

```

"0",
"1073741824",
"4096"
],
interface_list: - [
- {
vm_name: "VN1_VM1",
name: "54bb44e1-50e4-43d7-addr-44be809f1e40:cdca35ce-84ad-45da-9331-7bc67b7fcca6",
floating_ips: - [
- {
virtual_network: "default-domain:admin:public_vn",
ip_address: "10.204.219.37"
}
],
label: 16,
mac_address: "02:cd:ca:35:ce:84",
active: true,
virtual_network: "default-domain:admin:VN1",
l2_active: true,
ip_address: "10.1.1.253",
gateway: "10.1.1.254"
}
]
]

```

### Checking Floating IP Objects on the Virtual Machine Interface

Once you have obtained the virtual machine interface identifier, check the floating-ip objects that are associated with the virtual machine interface.

```
[root@nodec6 ~]# curl http://127.0.0.1:8095/floating-ip/f3eec4d6-889e-46a3-a8f0-879dfaff6ca0 |
python -m json.tool
```

```
{  
  
  "floating-ip": {  
  
    "floating_ip_address": "10.204.219.37",  
  
    "fq_name": [  
  
      "default-domain",  
  
      "admin",  
  
      "public_vn",  
  
      "public_pool",  
  
      "f3eec4d6-889e-46a3-a8f0-879dfaff6ca0"  
  
    ],  
  
    "href": "http://127.0.0.1:8095/floating-ip/f3eec4d6-889e-46a3-a8f0-879dfaff6ca0",  
  
    "id_perms": {  
  
      "created": "2014-02-07T10:07:05.869899",  
  
      "description": null,  
  
      "enable": true,  
  
      "last_modified": "2014-02-07T10:36:36.820926",  
  
      "permissions": {  
  
        "group": "admin",  
  
        "group_access": 7,  
  
        "other_access": 7,  
  
        "owner": "admin",  
  

```

```
        "owner_access": 7
    },
    "uuid": {
        "uuid_lslong": 12173378905373109408,
        "uuid_mslong": 17577202821367744163
    }
},
"name": "f3eec4d6-889e-46a3-a8f0-879dfaff6ca0",
"parent_href": "http://127.0.0.1:8095/floating-ip-pool/663737c1-f3ab-40ff-9442-
bdb6c225e3c3",
"parent_type": "floating-ip-pool",
"parent_uuid": "663737c1-f3ab-40ff-9442-bdb6c225e3c3",
"project_refs": [
    {
        "attr": null,
        "href": "http://127.0.0.1:8095/project/deef6549-8e6c-4e3e-9cde-c9bc2b72ce6f",
        "to": [
            "default-domain",
            "admin"
        ],
        "uuid": "deef6549-8e6c-4e3e-9cde-c9bc2b72ce6f"
    }
]
```

```

    ],
    "uuid": "f3eec4d6-889e-46a3-a8f0-879dfaff6ca0",
    "virtual_machine_interface_refs": [
      {
        "attr": null,
        "href": "http://127.0.0.1:8095/virtual-machine-interface/
cdca35ce-84ad-45da-9331-7bc67b7fcca6",
        "to": [
          "54bb44e1-50e4-43d7-addd-44be809f1e40",
          "cdca35ce-84ad-45da-9331-7bc67b7fcca6"
        ],
        "uuid": "cdca35ce-84ad-45da-9331-7bc67b7fcca6"
      }
    ]
  }
}
}

```

## View the BGP Peer Status on the Control Node

Use the Contrail UI or the control node http introspect on port 8083 to view the BGP peer status. In the following example, the control nodes are **nodec7** and **nodec8**.

Ensure that the BGP peering state is displayed as **Established** for the control nodes and the gateway MX.

### *Example*

- Using the Contrail UI:

| Peer           | Peer Type | Peer ASN | Status               | Last flap            | Messages (Recv/Sent) |
|----------------|-----------|----------|----------------------|----------------------|----------------------|
| 10.204.216.253 | BGP       | 64512    | Established, in sync | -                    | 1707/ 1590           |
| 10.204.216.64  | BGP       | 64512    | Established, in sync | 2/7/2014 11:46:32 AM | 1596/ 1597           |

- Using the control-node Introspect:

[http://nodec7:8083/Snh\\_BgpNeighborReq?ip\\_address=&domain=](http://nodec7:8083/Snh_BgpNeighborReq?ip_address=&domain=)

[http://nodec8:8083/Snh\\_BgpNeighborReq?ip\\_address=&domain=](http://nodec8:8083/Snh_BgpNeighborReq?ip_address=&domain=)

## Querying Routes in the Public Virtual Network

On each control-node, a query on the routes in the **public\_vn** lists the routes that are pushed by the MX gateway, which in the following example are 0.0.0.0/0 and 10.204.218.0/24.

In the following results, the floating IP route of 10.204.217.32 is installed by the compute node (nodec10) that hosts that virtual machine.

### Example

- Using the Contrail UI:

| Routing Table                      | Prefix           | Protocol | Source         | Next hop       | Label | Secur... | Origin VN                    |
|------------------------------------|------------------|----------|----------------|----------------|-------|----------|------------------------------|
| default-domainadminpublic_vninet.0 | 0.0.0.0/0        | BGP      | 10.204.216.253 | 10.204.216.253 | 16    | -        | default-domainadminpublic_vn |
|                                    | 10.204.218.0/24  | BGP      | 10.204.216.253 | 10.204.216.253 | 16    | -        | default-domainadminpublic_vn |
|                                    | 10.204.219.32/32 | XMPP     | nodec10        | 10.204.216.67  | 16    | 1        | default-domainadminpublic_vn |

- Using the http Introspect:

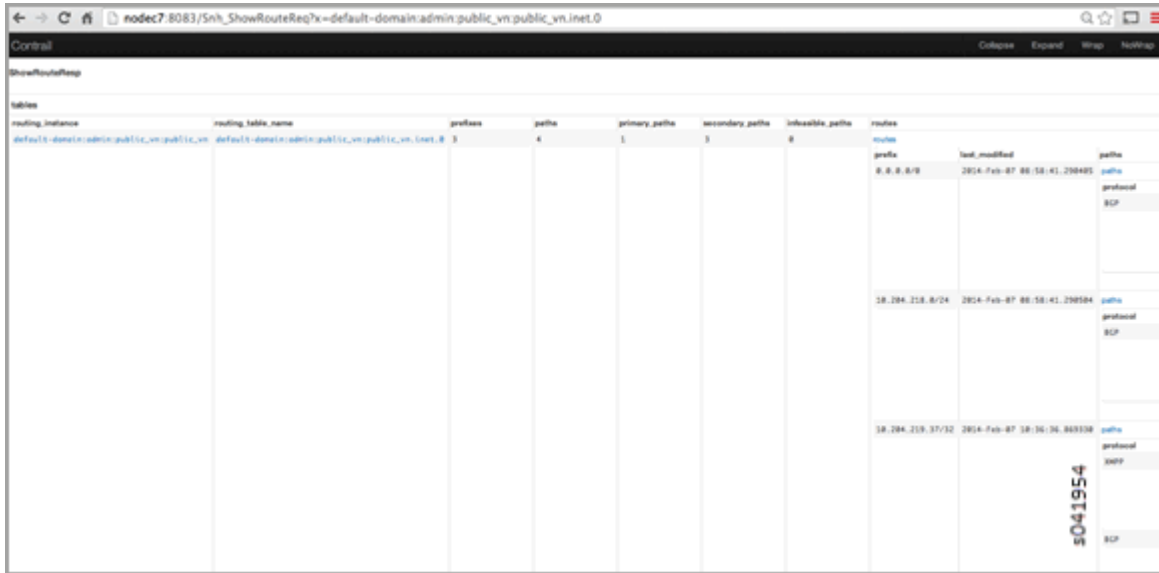
Following is the format for using an introspect query.

[http://<nodename/ip>:8083/Snh\\_ShowRouteReq?x=<RoutingInstance of public VN>.inet.0](http://<nodename/ip>:8083/Snh_ShowRouteReq?x=<RoutingInstance of public VN>.inet.0)

### Example



[http://nodec8:8083/Snh\\_BgpNeighborReq?ip\\_address=&domain=](http://nodec8:8083/Snh_BgpNeighborReq?ip_address=&domain=)

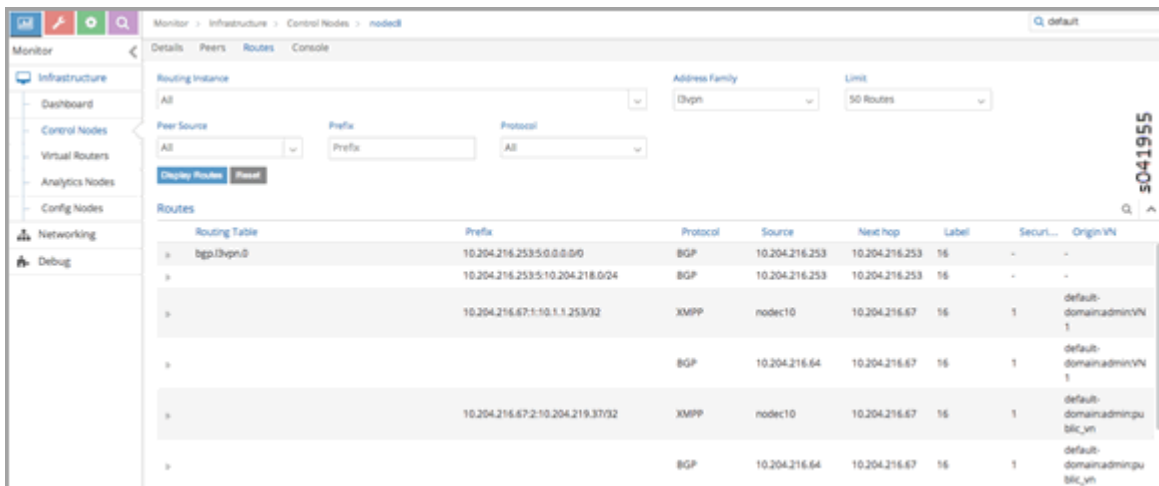


**View Corresponding BGP LL3VPN Routes**

Use the Contrail UI or the http introspect to view the public route’s corresponding BGP L3VPN routes, as in the following.

*Example*

- Using the Contrail UI:



- Using the control-node Introspect:

[http://nodec7:8083/Snh\\_ShowRouteReq?x=bgp.l3vpn.0](http://nodec7:8083/Snh_ShowRouteReq?x=bgp.l3vpn.0)

[http://nodec8:8083/Snh\\_ShowRouteReq?x=bgp.l3vpn.0](http://nodec8:8083/Snh_ShowRouteReq?x=bgp.l3vpn.0)

## Verification from the MX80 Gateway

This section provides options for verifying floating IP pools from the MX80 gateway.

### Verify BGP Sessions are Established

Use the following commands from the gateway to verify that BGP sessions are established with the control nodes nodec7 and nodec8:

```
root@mx-host> show bgp neighbor 10.204.216.64

Peer: 10.204.216.64+59287 AS 64512 Local: 10.204.216.253+179 AS 64512

Type: Internal    State: Established    Flags: <Sync>

Last State: OpenConfirm    Last Event: RecvKeepAlive

Last Error: Hold Timer Expired Error

Options: <Preference LocalAddress KeepAll AddressFamily Rib-group Refresh>

Address families configured: inet-vpn-unicast

Local Address: 10.204.216.253 Holdtime: 90 Preference: 170

Number of flaps: 216

Last flap event: HoldTime

Error: 'Hold Timer Expired Error' Sent: 68 Recv: 0

Error: 'Cease' Sent: 0 Recv: 43

Peer ID: 10.204.216.64    Local ID: 10.204.216.253    Active Holdtime: 90

Keepalive Interval: 30    Group index: 0    Peer index: 3

BFD: disabled, down

NLRI for restart configured on peer: inet-vpn-unicast
```

```

NLRI advertised by peer: inet-vpn-unicast

NLRI for this session: inet-vpn-unicast

Peer does not support Refresh capability

Stale routes from peer are kept for: 300

Peer does not support Restarter functionality

Peer does not support Receiver functionality

Peer does not support 4 byte AS extension

Peer does not support Addpath

```

### Show Routes Learned from Control Nodes

From the MX80, use `show route` to display the routes for the virtual machine 10.204.219.37 that are learned from both control-nodes.

In the following example, the routes learned are 10.204.216.64 and 10.204.216.65, pointing to a dynamic GRE tunnel next hop with a label of 16 (of the virtual machine).

```

public.inet.0: 4 destinations, 5 routes (4 active, 0 holddown, 0 hidden)

+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[Static/5] 10w6d 18:47:50
                   > to 10.204.218.254 via ge-1/0/1.0

10.204.218.0/24   *[Direct/0] 10w6d 18:47:51
                   > via ge-1/0/1.0

10.204.218.1/32  *[Local/0] 10w6d 18:48:07
                   Local via ge-1/0/1.0

```

```

10.204.219.37/32  *[BGP/170] 09:42:43, localpref 100, from 10.204.216.64

    AS path: ?, validation-state: unverified

> via gr-1/0/0.32779, Push 16

[BGP/170] 09:42:43, localpref 100, from 10.204.216.65

    AS path: ?, validation-state: unverified

> via gr-1/0/0.32779, Push 16

```

## Viewing the Compute Node Vnsw Agent

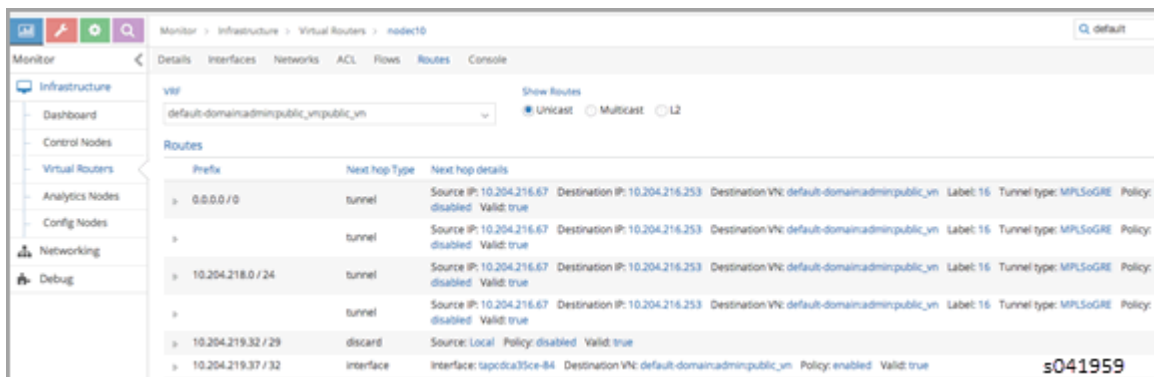
The compute node introspect can be accessed from port 8085. In the following examples, the compute nodes are nodec9 and nodec10.

### View Routing Instance Next Hops

On the routing instance of VN1, the routes 0.0.0.0/0 and 10.204.218.0/24 should have the next hop pointing to the MX gateway (10.204.216.253).

#### Example

#### 1. Using the Contrail UI:



| Prefix           | Next hop Type | Next hop details   |
|------------------|---------------|--|
| 0.0.0.0/0        | tunnel        | Source IP: 10.204.216.67 disabled Valid: true Destination IP: 10.204.216.253 Destination Vn: default-domainadminpublic_vn Label: 16 Tunnel type: MPLSoGRE Policy: disabled Valid: true |
| 10.204.218.0/24  | tunnel        | Source IP: 10.204.216.67 disabled Valid: true Destination IP: 10.204.216.253 Destination Vn: default-domainadminpublic_vn Label: 16 Tunnel type: MPLSoGRE Policy: disabled Valid: true |
| 10.204.219.32/29 | discard       | Source: Local Policy: disabled Valid: true   |
| 10.204.219.37/32 | interface     | Interface: sapcdca35ce-84 Destination Vn: default-domainadminpublic_vn Policy: enabled Valid: true   |

### Using the Unicast Route Table Index to View Next Hops

Alternatively, from the agent introspect, you can view the next hops at the unicast route table.

First, use the following to get the unicast route table index (ucindex ) for the routing instance default-domain:admin:public\_vn:public\_vn.

[http://nodedc10:8085/Snh\\_VrfListReq?x=default-domain:admin:public\\_vn:public\\_vn](http://nodedc10:8085/Snh_VrfListReq?x=default-domain:admin:public_vn:public_vn)

*Example*

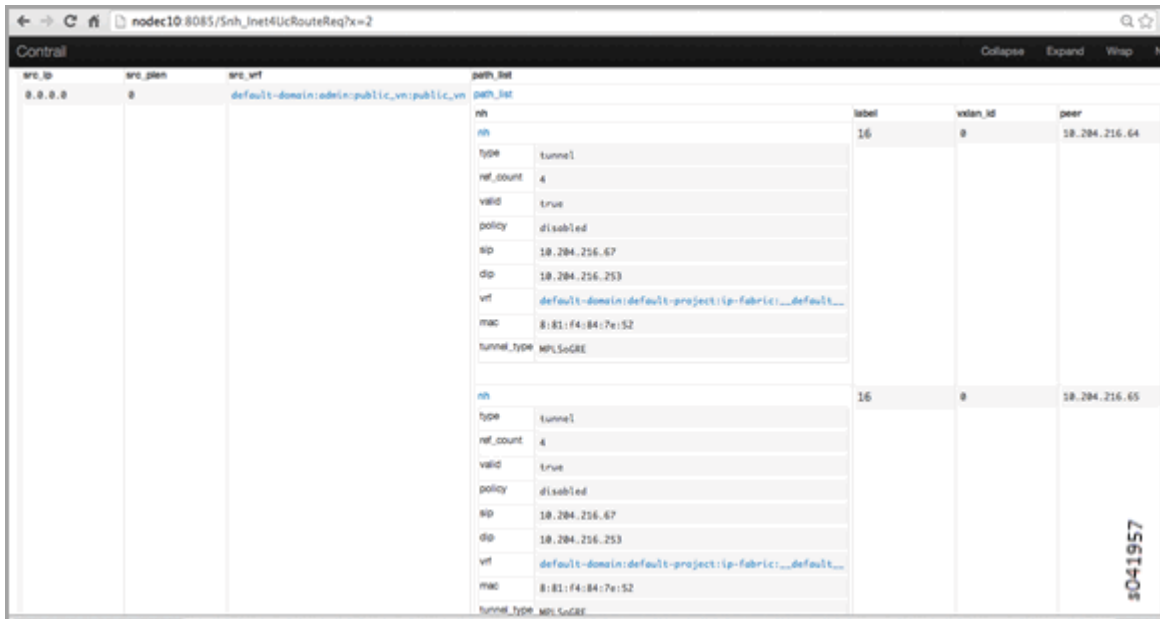
1. In the following example, the unicast route table index is 2.



Next, perform a route request query on ucindex 2, as shown in the following. The tunnel detail indicates the source and destination endpoints of the tunnel and the MPLS label 16 (the label of the virtual machine).

The query should also show a route for 10.204.219.37 with an interface next hop of tap-interface.

[http://nodedc10:8085/Snh\\_Inet4UcRouteReq?x=2](http://nodedc10:8085/Snh_Inet4UcRouteReq?x=2)



| 10.204.219.37 | 32 | default-domain:admin:public_vn:public_vn | path_list |                  |         |               |
|---------------|----|--|-----------|------------------|---------|---------------|
|               |    |  | nh        | label            | vlan_id | peer          |
|               |    |  | nh        | 16               | 0       | 10.204.216.64 |
|               |    |  | type      | interface        |         |               |
|               |    |  | ref_count | g                |         |               |
|               |    |  | valid     | true             |         |               |
|               |    |  | policy    | enabled          |         |               |
|               |    |  | itf       | tapcdca35ce-84   |         |               |
|               |    |  | mac       | 2:cd:ce:35:ce:84 |         |               |
|               |    |  | mcast     | disabled         |         |               |
|               |    |  | nh        | 16               | 0       | 10.204.216.65 |
|               |    |  | type      | interface        |         |               |
|               |    |  | ref_count | g                |         |               |
|               |    |  | valid     | true             |         |               |
|               |    |  | policy    | enabled          |         |               |
|               |    |  | itf       | tapcdca35ce-84   |         |               |
|               |    |  | mac       | 2:cd:ce:35:ce:84 |         |               |
|               |    |  | mcast     | disabled         |         |               |

A ping from the MX gateway to the virtual machine's floating IP in the public routing-instance should work.

## Advanced Troubleshooting

If you still have reachability problems after performing all of the tests in this article, for example, a ping between the virtual machine and the MX IP or to public addresses is failing, try the following:

- Validate that all the required Contrail processes are running by using the `contrail-status` command on all of the nodes.
- On the compute node where the virtual machine is present (nodec10 in this example), perform a `tcpdump` on the tap interface (`tcpdump -ni tapcdca35ce-84`). The output should show the incoming packets from the virtual machine.
- Check to see if any packet drops occur in the kernel vrouter module:

```
http://nodec10:8085/Snh_KDropStatsReq?
```

In the output, scroll down to find any drops. Note: You can ignore any `ds_invalid_arp` increments.

- On the physical interface where packets transmit onto the compute-node, perform a `tcpdump` matching the host IP of the MX to show the GRE encapsulated packets, as in the following.

```
[root@nodec10 ~]# cat /etc/contrail/agent.conf |grep -A 1 eth-port
```

```
<eth-port>
```

```
<name>p1p0p0</name>
```

```
</eth-port>
```

```
<metadata-proxy>
```

```
[root@nodec10 ~]# tcpdump -ni p1p0p0 host 10.204.216.253 -vv
```

```
tcpdump: WARNING: p1p0p0: no IPv4 address assigned
```

```
tcpdump: listening on p1p0p0, link-type EN10MB (Ethernet), capture size 65535 bytes
```

```
02:06:51.729941 IP (tos 0x0, ttl 64, id 57430, offset 0, flags [DF], proto GRE (47), length 112)
```

```
10.204.216.253 > 10.204.216.67: GREv0, Flags [none], length 92
```

```
MPLS (label 16, exp 0, [S], ttl 54)
```

```
IP (tos 0x0, ttl 54, id 35986, offset 0, flags [none], proto ICMP (1), length 84)
```

```
172.29.227.6 > 10.204.219.37: ICMP echo request, id 53240, seq 242, length 64
```

```
02:06:51.730052 IP (tos 0x0, ttl 64, id 324, offset 0, flags [none], proto GRE (47), length 112)
```

```
10.204.216.67 > 10.204.216.253: GREv0, Flags [none], length 92
```

```
MPLS (label 16, exp 0, [S], ttl 64)
```

```
IP (tos 0x0, ttl 64, id 33909, offset 0, flags [none], proto ICMP (1), length 84)
```

```
10.204.219.37 > 172.29.227.6: ICMP echo reply, id 53240, seq 242, length 64
```

```
02:06:52.732283 IP (tos 0x0, ttl 64, id 12675, offset 0, flags [DF], proto GRE (47), length 112)
```

```
10.204.216.253 > 10.204.216.67: GREv0, Flags [none], length 92
```

```
MPLS (label 16, exp 0, [S], ttl 54)
```

```

IP (tos 0x0, ttl 54, id 54155, offset 0, flags [none], proto ICMP (1), length 84)

172.29.227.6 > 10.204.219.37: ICMP echo request, id 53240, seq 243, length 64

02:06:52.732355 IP (tos 0x0, ttl 64, id 325, offset 0, flags [none], proto GRE (47), length
112)

10.204.216.67 > 10.204.216.253: GREv0, Flags [none], length 92

MPLS (label 16, exp 0, [S], ttl 64)

IP (tos 0x0, ttl 64, id 33910, offset 0, flags [none], proto ICMP (1), length 84)

10.204.219.37 > 172.29.227.6: ICMP echo reply, id 53240, seq 243, length 64

^C

4 packets captured

5 packets received by filter

0 packets dropped by kernel

[root@nodec10 ~]#

```

- On the MX gateway, use the following to inspect the GRE tunnel rx/tx (received/transmitted) packet count:

```

root@mx-host> show interfaces gr-1/0/0.32779 |grep packets

Input packets : 542

Output packets: 559

root@blr-mx1> show interfaces gr-1/0/0.32779 |grep packets

Input packets : 544

```



```
Output packets: 561
```

- Look for any packet drops in the FPC, as in the following:

```
show pfe statistics traffic fpc <id>
```

- Also inspect the dynamic tunnels, using the following:

```
show dynamic-tunnels database
```

## Removing Stale Virtual Machines and Virtual Machine Interfaces

### IN THIS SECTION

- [Problem Example | 1025](#)
- [Show Virtual Machines | 1027](#)
- [Show Virtual Machines Using Python API | 1028](#)
- [Delete Methods | 1030](#)

This topic gives examples for removing stale VMs (virtual machines) and VMIs (virtual machine interfaces). Before you can remove a stale VM or VMI, you must first remove any back references associated to the VM or VMI.

### Problem Example

The troubleshooting examples in this topic are based on the following problem example. A `net-delete` of the virtual machine `2a8120ec-bd18-49f4-aca0-acfc6e8fe74f` returned the following messages that there are two VMIs that still have back-references to the stale VM.

The two VMIs must be deleted first, then the Neutron `net-delete <vm_ID>` command will complete without errors.

```
From neutron.log:
```

```
2014-03-10 14:18:05.208
```

```
DEBUG [urllib3.connectionpool]

"DELETE/virtual-network/2a8120ec-bd18-49f4-aca0-acfc6e8fe74f HTTP/1.1" 409 203

2014-03-10 14:18:05.278

ERROR [neutron.api.v2.resource] delete failed

Traceback (most recent call last):

  File "/usr/lib/python2.7/dist-packages/neutron/api/v2/resource.py", line
84, in resource

    result = method(request=request, **args)

  File "/usr/lib/python2.7/dist-packages/neutron/api/v2/base.py", line
432, in delete

    obj_deleter(request.context, id, **kwargs)

  File

"/usr/lib/python2.7/dist-packages/neutron/plugins/juniper/contrail/contrail
plugin.py", line 294, in delete_network

    raise e

RefsExistError: Back-References from

http: //127.0.0.1:8082/virtual-machine-interface/51daf6f4-7366-4463-a819-bd1
17fe3a8c8,

http: //127.0.0.1:8082/virtual-machine-interface/30882e66-e175-4fbb-862e-354
bb700b579 still exist
```

## Show Virtual Machines

Use the following command to show all of the virtual machines known to the Contrail API server. Replace the variable `<config-node-IP>` shown in the example with the IP address of the config-node in your setup.

```
http://<config-node-IP>:8082/virtual-machines
```

### Example

In the following example, 03443891-99cc-4784-89bb-9d1e045f8aa6 is a stale VM that needs to be removed.

```
virtual-machines:
  [
    {
      href:"http: //example-node:8082/virtual-machine/
03443891-99cc-4784-89bb-9d1e045f8aa6",
      fq_name:
        [
          "03443891-99cc-4784-89bb-9d1e045f8aa6"
        ],
      uuid:"03443891-99cc-4784-89bb-9d1e045f8aa6"
    },
  ],
```

When the user attempts to delete the stale VM, a message displays that children to the VM still exist:

```
root@example-node:~# curl -X DELETE -H "Content-Type: application/json; charset=UTF-8" http: //
127.0.0.1:8082/virtual-machine/03443891-99cc-4784-89bb-9d1e045f8aa6
Children http: //127.0.0.1:8082/virtual-machine-interface/0c32a82a-7bd3-46c7-b262-6d85b9911a0d
still exist
root@example-node:~#
```

The user opens `http://example-node:8082/virtual-machine/03443891-99cc-4784-89bb-9d1e045f8aa6`, and sees a `virtual-machine-interface` (VMI) attached to it. The VMI must be removed before the VM can be removed.

However, when the user attempts to delete the VMI from the stale VM, they get a message that there is still a back-reference:

```
root@example-node:~# curl -X DELETE -H "Content-Type: application/json; charset=UTF-8" http: //
<example-IP>:8082/virtual-machine-interface/0c32a82a-7bd3-46c7-b262-6d85b9911a0d

Back-References from http: //<example-IP>:8082/instance-ip/6ffa29a1-023f-462b-b205-353da8e3a2a4
still exist

root@example-node:~#
```

Because there is a back-reference from an `instance-ip` object still present, the `instance-ip` object must first be deleted, as follows:

```
root@example-node:~# curl -X DELETE -H "Content-Type: application/json; charset=UTF-8" http: //
<example-IP>:8082/instance-ip/6ffa29a1-023f-462b-b205-353da8e3a2a4

root@example-node:~#
```

When the `instance-ip` is deleted, then the VMI and the VM can be deleted.

**NOTE:** To prevent inconsistency, be certain that the VM is not present in the Nova database before deleting the VM.

## Show Virtual Machines Using Python API

The following example shows how to view virtual machines using a Python API. This example shows virtual machines and back-references. Once you identify back-references and existing children, you can delete them first, then delete the stale VM.

```
root@example-node:~# source /opt/contrail/api-venv/bin/activate

File "<stdin>", line 1, in <module>

File "/opt/contrail/api-venv/lib/python2.7/site-packages/vnc_api/gen/vnc_api_client_gen.py",
line 3793, in virtual_machine_interface_delete
```

```

        content = self._request_server(rest.OP_DELETE, uri)

File "/opt/contrail/api-venv/lib/python2.7/site-packages/vnc_api/vnc_api.py", line 342, in
_request_server

    raise RefsExistError(content)

cfgm_common.exceptions.RefsExistError: Back-References from http: // <example-IP>:8082/instance-
ip/6ffa29a1-023f-462b-b205-353da8e3a2a4 still exist

>>> (api-venv)root@example-node:~# python

Python 2.7.5 (default, Mar 10 2014, 03:55:35)

[GCC 4.6.3] on linux2

Type "help", "copyright", "credits" or "license" for more information.

>>> from vnc_api.vnc_api import VncApi

>>> vh=VncApi()

>>> vh.virtual_machine_interface_delete(id='0c32a82a-7bd3-46c7-b262-6d85b9911a0d')

```

Traceback (most recent call last):

```

File "<stdin>", line 1, in <module>

    File "/opt/contrail/api-venv/lib/python2.7/site-packages/vnc_api/gen/vnc_api_client_gen.py",
line 3793, in virtual_machine_interface_delete

        content = self._request_server(rest.OP_DELETE, uri)

File "/opt/contrail/api-venv/lib/python2.7/site-packages/vnc_api/vnc_api.py", line 342, in
_request_server

    raise RefsExistError(content)

cfgm_common.exceptions.RefsExistError: Back-References from http: // <example-IP>:8082/instance-
ip/6ffa29a1-023f-462b-b205-353da8e3a2a4 still exist

```

&gt;&gt;&gt;

## Delete Methods

Use help (vh) to show all delete methods supported.

Typical commands for deleting VMs and VMIs include:

- `virtual_machine_delete()` to delete a virtual machine
- `instance_ip_delete()` to delete an instance-ip.

## Troubleshooting Link-Local Services in Contrail

### IN THIS SECTION

- [Overview of Link-Local Services | 1030](#)
- [Troubleshooting Procedure for Link-Local Services | 1031](#)
- [Metadata Service | 1032](#)
- [Troubleshooting Procedure for Link-Local Metadata Service | 1032](#)

Use the troubleshooting steps and guidelines in this topic when you have errors with Contrail link-local services.

### Overview of Link-Local Services

Virtual machines might be set up to access specific services hosted on the fabric infrastructure. For example, a virtual machine might be a Nova client that requires access to the Nova API service running in the fabric network. Access to services hosted on the fabric network can be provided by configuring the services as link-local services.

A link-local address and a service port is chosen for the specific service running on a TCP / UDP port on a server in the fabric. With the link-local service configured, virtual machines can access the service using the link-local address. For link-local services, Contrail uses the address range 169.254.169.x.

Link-local service can be configured using the Contrail WebUI: **Configure > Infrastructure > Link Local Services**.

Create Link Local Service ✕

| Service Name                             | Link Local Service Address |      | Fabric Address                          |             |     |
|--|----------------------------|------|---|-------------|-----|
|  | IP                         | Port | IP / DNS                                | Port        |     |
| ntp <span style="float: right;">▼</span> | 169.254.169.100            | 123  | IP <span style="float: right;">▼</span> | 172.17.28.5 | +   |
|  |                            |      |   |             | 123 |

Cancel Save

## Troubleshooting Procedure for Link-Local Services

Use the following steps when you are troubleshooting link-local services errors.

1. Verify the reachability of the fabric server that is hosting the link-local service from the compute node.
2. Check the state of the virtual machine and the interface:
  - Is the **Status** of virtual machine **Up**?
  - Is the corresponding tap interface **Active**?

Checking the virtual machine status in the Contrail UI:

Monitor > Infrastructure > Virtual Routers > nodec15 Search Sitemap

Details Interfaces Networks ACL Flows Routes Console

Interfaces 📄 🔍 > s041994

| Name           | Label | Status | Network    | IP Address | Floating IP | Instance   |
|----------------|-------|--------|------------|------------|-------------|--|
| tap4b094dbe-f0 | 18    | Up     | vn1 (demo) | 1.2.3.247  | None        | 4f4b917a-a071-4517-961a-0e41067fec63 / vn1-v<br>m2 |

Checking the tap interface status in the http agent introspect:

`http://<compute-node-ip>:8085/Snh_ItfReq?name=`

itf\_list s041995

| index | name           | uuid                                 | vrf_name                    | active |
|-------|----------------|--------------------------------------|-----------------------------|--------|
| 3     | tap722a7a11-6d | 722a7a11-6d2e-47e9-a4cc-687a105a240f | default-domain:demo:vn1:vn1 | Active |

3. Check the link-local configuration in the vrouter agent. Make sure the configured link-local service is displayed.

`http://<compute-node-ip>:8085/Snh_LinkLocalServiceInfo?`

| service_list           |                      |                        |                   |             |               |
|------------------------|----------------------|------------------------|-------------------|-------------|---------------|
| linklocal_service_name | linklocal_service_ip | linklocal_service_port | ipfabric_dns_name | ipfabric_ip | ipfabric_port |
| ntp                    | 169.254.169.100      | 123                    | -                 | ipfabric_ip | 123           |
|                        |                      |                        |                   | 172.17.28.5 |               |

4. Validate the BGP neighbor config and the BGP peering config object. When the virtual machine communicates with the configured link-local service, a forward and reverse flow for the communication is set up. Check that the flow for this communication is created and the flow action is NAT.

[http://<compute-node-ip>:8085/Snh\\_KFlowReq?flow\\_idx=](http://<compute-node-ip>:8085/Snh_KFlowReq?flow_idx=)

Check that all flow entries display NAT action programmed and display flags for the fields (source or destination IP and ports) that have NAT programmed. Also shown are the number of packets and bytes transmitted in the respective flows.

| flow_list |        |             |       |                 |       |       |        |        |   |  |
|-----------|--------|-------------|-------|-----------------|-------|-------|--------|--------|---|--|
| index     | rflow  | sip         | sport | dip             | dport | proto | vrf_id | action | flags                                     |  |
| 467472    | 234436 | 1.2.3.247   | 123   | 169.254.169.100 | 123   | 17    | 1      | NAT    | ACTIVE   VRFT   SNAT   SPAT   DNAT   SPAT |  |
| 234436    | 467472 | 172.17.28.5 | 123   | 10.204.216.72   | 43226 | 17    | 0      | NAT    | ACTIVE   VRFT   SNAT   DNAT   s041997     |  |

The forward flow displays the source IP of the virtual machine and the destination IP of the link-local service. The reverse flow displays the source IP of the fabric host and the destination IP of the compute node's vhost interface. If the service is hosted on the same compute node, the destination address of the reverse flow displays the metadata address allocated to the virtual machine.

Note that the **index** and **rflow** index for the two flows are reversed.

You can also view similar information in the vrouter agent introspect page, where you can see the policy and security group for the flow. Check that the flow actions display as **pass**.

[http://<compute-node-ip>:8085/Snh\\_FetchAllFlowRecords?](http://<compute-node-ip>:8085/Snh_FetchAllFlowRecords?)

## Metadata Service

OpenStack allows virtual instances to access metadata by sending an HTTP request to the link-local address 169.254.169.254. The metadata request from the instance is proxied to Nova, with additional HTTP header fields added, which Nova uses to identify the source instance. Then Nova responds with appropriate metadata.

The Contrail vrouter acts as the proxy, trapping the metadata requests, adding the necessary header fields, and sending the requests to the Nova API server.

## Troubleshooting Procedure for Link-Local Metadata Service

Metadata service is also a link-local service, with a fixed service name (metadata), a fixed service address (169.254.169.254:80), and a fabric address pointing to the server where the OpenStack Nova API server



is running. All of the configuration and troubleshooting procedures for Contrail link-local services also apply to the metadata service.

However, for metadata service, the flow is always set up to the compute node, so the vrouter agent will update and proxy the HTTP request. The vrouter agent listens on a local port to receive the metadata requests. Consequently, the reverse flow has the compute node as the source IP, the local port on which the agent is listening is the source port, and the instance's metadata IP is the destination IP address.

After performing all of the troubleshooting procedures for link-local services, the following additional steps can be used to further troubleshoot metadata service.

1. Check the metadata statistics for: the number of metadata requests received by the vrouter agent, the number of proxy sessions set up with the Nova API server, and number of internal errors encountered.

`http://<compute-node-ip>:8085/Snh_MetadataInfo?`

The port on which the vrouter agent listens for metadata requests is also displayed.

|                                       |       |
|---------------------------------------|-------|
| <code>metadata_server_port</code>     | 45094 |
| <code>metadata_requests</code>        | 2     |
| <code>metadata_responses</code>       | 0     |
| <code>metadata_proxy_sessions</code>  | 2     |
| <code>metadata_internal_errors</code> | 0     |

s041998

2. Check the metadata trace messages, which show the trail of metadata requests and responses.

`http://<compute-node-ip>:8085/Snh_SandeshTraceRequest?x=Metadata`

3. Check the Nova configuration. On the server running the OpenStack service, inspect the `nova.conf` file.

- Ensure that the metadata proxy is enabled, as follows:

`service_neutron_metadata_proxy = True`

`service_quantum_metadata_proxy = True (on older installations)`

- Check to see if the metadata proxy shared secret is set:

neutron\_metadata\_proxy\_shared\_secret

quantum\_metadata\_proxy\_shared\_secret (on older installations)

If the shared secret is set in `nova.conf`, the same secret must be configured on each compute node in the file `/etc/contrail/contrail-vrouter-agent.conf`, and the same shared secret must be updated in the METADATA section as `metadata_proxy_secret=<secret>`.

**4. Restart the vrouter agent after modifying the shared secret:**

```
service contrail-vrouter restart
```

# 5

PART

## Conrail Commands and APIs

---

[Conrail Commands | 1036](#)

[Conrail Application Programming Interfaces \(APIs\) | 1066](#)

---

# Contrail Commands

## IN THIS CHAPTER

- [Getting Contrail Node Status | 1036](#)
- [contrail-logs \(Accessing Log File Messages\) | 1048](#)
- [contrail-status \(Viewing Node Status\) | 1051](#)
- [contrail-version \(Viewing Version Information\) | 1053](#)
- [service \(Managing Services\) | 1056](#)
- [Backing Up Contrail Databases Using JSON Format | 1058](#)

## Getting Contrail Node Status

### IN THIS SECTION

- [Overview | 1036](#)
- [UVE for NodeStatus | 1037](#)
- [Node Status Features | 1037](#)
- [Using Introspect to Get Process Status | 1044](#)
- [contrail-status script | 1046](#)

### Overview

This topic describes how to view the status of a Contrail node on a physical server. Contrail nodes include config, control, analytics, compute, and so on.

## UVE for NodeStatus

The User-Visible Entity (UVE) mechanism is used to aggregate and send the status information. All node types send a NodeStatus structure in their respective node UVEs. The following is a control node UVE of NodeStatus:

```

struct NodeStatus {

    1: string name (key="ObjectBgpRouter")

    2: optional bool deleted

    3: optional string status

    // Sent by process

    4: optional list<process_info.ProcessStatus> process_status (aggtype="union")

    // Sent by node manager

    5: optional list<process_info.ProcessInfo> process_info (aggtype="union")

    6: optional string description

}

uve sandesh NodeStatusUVE {

    1: NodeStatus data

}

```

## Node Status Features

The most important features of NodeStatus include:

ProcessStatus

ProcessInfo

**ProcessStatus**

Also `process_status`, is sent by the processes corresponding to the virtual node, and displays the status of the process and an aggregate state indicating if the process is functional or non-functional. The `process_status` includes the state of the process connections (`ConnectionInfo`) to important services and other information necessary for the process to be functional. Each process sends its `NodeStatus` information, which is aggregated as union (`aggtype="union"`) at the analytics node. The following is the `ProcessStatus` structure:

```
1.  struct ProcessStatus {
2.      1: string module_id
3.      2: string instance_id
4.      3: string state
5.      4: optional list<ConnectionInfo> connection_infos
6.      5: optional string description
7.  }
8.
9.  struct ConnectionInfo {
10.     1: string type
11.     2: string name
12.     3: optional list<string> server_addrs
13.     4: string status
14.     5: optional string description
15. }
```

### ProcessInfo

Sent by the node manager, /usr/bin/contrail-nodemgr. Node manager is a monitor process per contrail virtual node that tracks the running state of the processes. The following is the ProcessInfo structure:

```

16. struct ProcessInfo {
17.     1: string                process_name
18.     2: string                process_state
19.     3: u32                   start_count
20.     4: u32                   stop_count
21.     5: u32                   exit_count
22.     // time when the process last entered running stage
23.     6: optional string       last_start_time
24.     7: optional string       last_stop_time
25.     8: optional string       last_exit_time
26.     9: optional list<string> core_file_list
27. }

```

### Example: NodeStatus

The following is an example output of NodeStatus obtained from the Rest API:

```

http://:8081/analytics/uves/control-...ilt=NodeStatus .
{
  NodeStatus:
  {
    process_info:
    [

```

```
{  
  
  process_name: "contrail-control",  
  
  process_state: "PROCESS_STATE_RUNNING",  
  
  last_stop_time: null,  
  
  start_count: 1,  
  
  core_file_list: [ ],  
  
  last_start_time: "1409002143776558",  
  
  stop_count: 0,  
  
  last_exit_time: null,  
  
  exit_count: 0  
  
},  
  
{  
  
  process_name: "contrail-control-nodemgr",  
  
  process_state: "PROCESS_STATE_RUNNING",  
  
  last_stop_time: null,  
  
  start_count: 1,  
  
  core_file_list: [ ],  
  
  last_start_time: "1409002141773481",  
  
  stop_count: 0,  
  
  last_exit_time: null,  
  
  exit_count: 0
```



```
},  
  
{  
  
  process_name: "contrail-dns",  
  
  process_state: "PROCESS_STATE_RUNNING",  
  
  last_stop_time: null,  
  
  start_count: 1,  
  
  core_file_list: [ ],  
  
  last_start_time: "1409002145778383",  
  
  stop_count: 0,  
  
  last_exit_time: null,  
  
  exit_count: 0  
  
},  
  
{  
  
  process_name: "contrail-named",  
  
  process_state: "PROCESS_STATE_RUNNING",  
  
  last_stop_time: null,  
  
  start_count: 1,  
  
  core_file_list: [ ],  
  
  last_start_time: "1409002147780118",  
  
  stop_count: 0,  
  
  last_exit_time: null,
```

```
    exit_count: 0
  }
],
process_status:
[
{
  instance_id: "0",
  module_id: "ControlNode",
  state: "Functional",
  description: null,
  connection_infos:
  [
  {
    server_addrs:
    [
      "10.84.13.45:8443"
    ],
  }
  {
    server_addrs:
    [
      "10.84.13.45:8086"
    ],
  }
  ],
}
```

```
    status: "Up",  
  
    type: "Collector",  
  
    name: null,  
  
    description: "Established"  
  
  },  
  
  {  
  
    server_addrs:  
    [  
  
      "10.84.13.45:5998"  
  
    ],  
  
    status: "Up",  
  
    type: "Discovery",  
  
    name: "Collector",  
  
    description: "SubscribeResponse"  
  
  },  
  
  {  
  
    server_addrs:  
    [  
  
      "10.84.13.45:5998"  
  
    ],  
  
    status: "Up",
```



The following is an example of the process state of contrail-control that is obtained by using

[http://server-ip:8083/Snh\\_SandeshUVECacheReq?x=NodeStatus](http://server-ip:8083/Snh_SandeshUVECacheReq?x=NodeStatus)

**NOTE:** The example output is the ProcessStatus of only one process of contrail-control. It does not show the full aggregated status of the control node through its UVE (as in the previous example).

```
root@a6s45:~# curl http://10.84.13.45:8083/Snh_SandeshU...q?x=NodeStatus
```

```
<?xml-stylesheet type="text/xsl" href="/universal_parse.xsl"?><_NodeStatusUVE_list
type="slist"><NodeStatusUVE type="sandesh"><data type="struct" identifier="1"><NodeStatus><name
type="string" identifier="1" key="ObjectBgpRouter">a6s45</name><process_status type="list"
identifier="4" aggtype="union"><list type="struct" size="1"><ProcessStatus><module_id
type="string" identifier="1">ControlNode</module_id><instance_id type="string" identifier="2"></
instance_id><state type="string" identifier="3">Functional</state><connection_infos type="list"
identifier="4"><list type="struct" size="5"><ConnectionInfo><type type="string"
identifier="1">IFMap</type><name type="string" identifier="2">IFMapServer</name><server_addr
type="list" identifier="3"><list type="string" size="1"><element>10.84.13.45:8443</element></
list></server_addr><status type="string" identifier="4">Up</status><description type="string"
identifier="5">Connection with IFMap Server (ironD)</description></
ConnectionInfo><ConnectionInfo><type type="string" identifier="1">Collector</type><name
type="string" identifier="2"></name><server_addr type="list" identifier="3"><list type="string"
size="1"><element>10.84.13.45:8086</element></list></server_addr><status type="string"
identifier="4">Up</status><description type="string" identifier="5">Established</description></
ConnectionInfo><ConnectionInfo><type type="string" identifier="1">Discovery</type><name
type="string" identifier="2">Collector</name><server_addr type="list" identifier="3"><list
type="string" size="1"><element>10.84.13.45:5998</element></list></server_addr><status
type="string" identifier="4">Up</status><description type="string"
identifier="5">SubscribeResponse</description></ConnectionInfo><ConnectionInfo><type
type="string" identifier="1">Discovery</type><name type="string" identifier="2">IfmapServer</
name><server_addr type="list" identifier="3"><list type="string"
size="1"><element>10.84.13.45:5998</element></list></server_addr><status type="string"
identifier="4">Up</status><description type="string" identifier="5">SubscribeResponse</
description></ConnectionInfo><ConnectionInfo><type type="string" identifier="1">Discovery</
type><name type="string" identifier="2">xmpp-server</name><server_addr type="list"
identifier="3"><list type="string" size="1"><element>10.84.13.45:5998</element></list></
server_addr><status type="string" identifier="4">Up</status><description type="string"
identifier="5">Publish Response - HeartBeat</description></ConnectionInfo></list></
connection_infos><description type="string" identifier="5"></description></ProcessStatus></
list></process_status></NodeStatus></data></NodeStatusUVE><SandeshUVECacheResp
```

```
type="sandesh"><returned type="u32" identifier="1">1</returned><more type="bool"
identifier="0">false</more></SandeshUVECacheResp></__NodeStatusUVE_list>
```

## contrail-status script

The contrail-status script is used to give the status of the Contrail processes on a server.

The contrail-status script first checks if a process is running, and if it is, performs introspect into the process to get its functionality status, then outputs the aggregate status.

The possible states to display include:

- active - the process is running and functional; the internal state is good
- inactive - not started or stopped by user
- failed - the process exited too quickly and has not restarted
- initializing - the process is running, but the internal state is not yet functional.

### Example Output: Contrail-Status Script

The following is an example output from the contrail-status script.

```
root@a6s45:~# contrail-status

== Contrail vRouter ==

supervisor-vrouter:      active

contrail-vrouter-agent   active

contrail-vrouter-nodemgr active

== Contrail Control ==

supervisor-control:      active

contrail-control         active

contrail-control-nodemgr active

contrail-dns             active
```

```
contrail-named          active
```

```
== Contrail Analytics ==
```

```
supervisor-analytics:  active
```

```
contrail-analytics-api  active
```

```
contrail-analytics-nodemgr active
```

```
contrail-collector      active
```

```
contrail-query-engine   active
```

```
== Contrail Config ==
```

```
supervisor-config:     active
```

```
contrail-api:0         active
```

```
contrail-config-nodemgr active
```

```
contrail-schema        active
```

```
contrail-svc-monitor   active
```

```
rabbitmq-server        active
```

```
== Contrail Web UI ==
```

```
supervisor-webui:      active
```

```
contrail-webui         active
```

```
contrail-webui-middleware active
```

```
redis-webui          active
```

```
== Contrail Database ==
```

```
supervisord-contrail-database:active
```

```
contrail-database    active
```

```
contrail-database-nodemgr  active
```

## contrail-logs (Accessing Log File Messages)

### IN THIS SECTION

- [Command-Line Options for Contrail-Logs | 1048](#)
- [Option Descriptions | 1049](#)
- [Example Uses | 1050](#)

A command-line utility, `contrail-logs`, uses REST APIs to retrieve system log messages, object log messages, and trace messages.

### Command-Line Options for Contrail-Logs

The command-line utility for accessing log file information is `contrail-logs` in the analytics node. The following are the options supported at the command line for `contrail-logs`, as viewed using the `--help` option.

```
[root@host]# contrail-logs --help
usage: contrail-logs [-h]
                   [--opserver-ip OPSERVER_IP]
                   [--opserver-port OPSERVER_PORT]
```



```

    [--start-time START_TIME]
    [--end-time END_TIME]
    [--last LAST]
    [--source SOURCE]
    [--module {ControlNode, VRouterAgent, ApiServer, Schema, OpServer,
Collector, QueryEngine, ServiceMonitor, DnsAgent}]
    [--category CATEGORY]
    [--level LEVEL]
    [--message-type MESSAGE_TYPE]
    [--reverse]
    [--verbose]
    [--all]
    [--object {ObjectVNTTable, ObjectVMTable, ObjectSITable, ObjectVRouter,
ObjectBgpPeer, ObjectRoutingInstance, ObjectBgpRouter, ObjectXmppConnection,
ObjectCollectorInfo, ObjectGeneratorInfo, ObjectConfigNode}]
    [--object-id OBJECT_ID]
    [--object-select-field {ObjectLog, SystemLog}]
    [--trace TRACE]

```

## Option Descriptions

The following are the descriptions for each of the option arguments available for `contrail-logs`.

```

optional arguments:
  -h, --help
                show this help message and exit
  --opserver-ip OPSERVER_IP
                IP address of OpServer (default: 127.0.0.1)
  --opserver-port OPSERVER_PORT
                Port of OpServer (default: 8081)
  --start-time START_TIME
                Logs start time (format now-10m, now-1h) (default: now-10m)
  --end-time END_TIME
                Logs end time (default: now)
  --last LAST
                Logs from last time period (format 10m, 1d) (default: None)
  --source SOURCE
                Logs from source address (default: None)
  --module {ControlNode, VRouterAgent, ApiServer, Schema, OpServer, Collector, QueryEngine,
ServiceMonitor, DnsAgent}

```

```

--category CATEGORY          Logs from module (default: None)
--level LEVEL                Logs of category (default: None)
--message-type MESSAGE_TYPE  Logs of level (default: None)
--reverse                    Logs of message type (default: None)
--verbose                    Show logs in reverse chronological order (default: False)
--all                        Show internal information (default: True)
--object {ObjectVNTable, ObjectVMTable, ObjectSITable, ObjectVRouter, ObjectBgpPeer,
ObjectRoutingInstance, ObjectBgpRouter, ObjectXmppConnection, ObjectCollectorInfo,
ObjectGeneratorInfo, ObjectConfigNode}
--object-id OBJECT_ID        Logs of object type (default: None)
--object-select-field {ObjectLog, SystemLog}
                             Logs of object name (default: None)
--trace TRACE                Select field to filter the log (default: None)
                             Dump trace buffer (default: None)

```

## Example Uses

The following examples show how you can use the option arguments available for `contrail-logs` to retrieve the information you specify.

1. View only the system log messages from all boxes for the last 10 minutes.

```
contrail-logs
```

2. View all log messages (systemlog, objectlog, uve, ...) from all boxes for the last 10 minutes.

```
contrail-logs --all
```

3. View only the control node system log messages from all boxes for the last 10 minutes.

```
contrail-logs --module ControlNode
```

--module accepts the following values - ControlNode, VRouterAgent, ApiServer, Schema, ServiceMonitor, Collector, OpServer, QueryEngine, DnsAgent

4. View the control node system log messages from source `a6s23.contrail.juniper.net` for the last 10 minutes.

```
contrail-logs --module ControlNode --source a6s23.contrail.juniper.net
```

5. View the XMPP category system log messages from all modules on all boxes for the last 10 minutes.

```
contrail-logs --category XMPP
```

6. View the system log messages from all the boxes from the last hour.

```
contrail-logs --last 1h
```

7. View the system log messages from the VN object named `demo:admin:vn1` from all boxes for the last 10 minutes.

```
contrail-logs --object ObjectVNTable --object-id demo:admin:vn1
```

--object accepts the following values - `ObjectVNTable`, `ObjectVMTable`, `ObjectSITable`, `ObjectVRouter`, `ObjectBgpPeer`, `ObjectRoutingInstance`, `ObjectBgpRouter`, `ObjectXmppConnection`, `ObjectCollectorInfo`

8. View the system log messages from all boxes for the last 10 minutes in reverse chronological order:

```
contrail-logs --reverse
```

9. View the system log messages from a specific time interval and display them in a specified date format.

```
contrail-logs --start-time "2013 May 12 18:30:27.0" --end-time "2013 May 12 18:31:27.0"
```

## contrail-status (Viewing Node Status)

### IN THIS SECTION

- [Syntax | 1052](#)
- [Description | 1052](#)
- [Required Privilege Level | 1052](#)
- [Sample Output | 1052](#)
- [Release Information | 1053](#)

## Syntax

```
[root@host ~]# contrail-status
```

## Description

Display a list of all components of a Contrail server node (such as control, configuration, database, Web-UI, analytics, or vrouter) and report their current status of active or inactive.

## Required Privilege Level

admin

## Sample Output

The following example usage displays on a server that is configured for the roles of **vrouter**, **controller**, **analytics**, **configuration**, **web-ui**, and **database**.

## Sample Output

```
root@host:~# contrail-status
== Contrail vRouter ==
supervisor-vrouter:      active
contrail-vrouter-agent  active
contrail-vrouter-nodemgr active

== Contrail Control ==
supervisor-control:     active
contrail-control        active
contrail-control-nodemgr active
contrail-dns            active
contrail-named          active

== Contrail Analytics ==
supervisor-analytics:   active
contrail-analytics-api  active
contrail-analytics-nodemgr active
contrail-collector      active
contrail-query-engine   active
```

```

== Contrail Config ==
supervisor-config:      active
contrail-api:0          active
contrail-config-nodemgr active
contrail-discovery:0    active
contrail-schema         active
contrail-svc-monitor    active
ifmap                   active
rabbitmq-server         active

== Contrail Web UI ==
supervisor-webui:      active
contrail-webui         active
contrail-webui-middleware active
redis-webui            active

== Contrail Database ==
supervisord-contrail-database:active
contrail-database      active
contrail-database-nodemgr active

```

## Release Information

Command introduced in Contrail Release 1.0.

## contrail-version (Viewing Version Information)

### IN THIS SECTION

- [Syntax | 1054](#)
- [Description | 1054](#)
- [Required Privilege Level | 1054](#)
- [Sample Output | 1054](#)
- [Sample Output | 1055](#)
- [Release Information | 1055](#)

## Syntax

```
[root@host]# contrail-version
```

## Description

Display a list of all installed components with their version and build numbers.

## Required Privilege Level

admin

## Sample Output

The following example shows version and build information for all installed components.

## Sample Output

```
root@host> contrail-version
```

| Package                      | Version               | Build-ID   Repo   RPM Name |
|------------------------------|-----------------------|----------------------------|
| contrail-analytics           | 1-1309090026.el6      | 141                        |
| contrail-analytics-venv      | 0.1-1309062310.el6    | 141                        |
| contrail-api                 | 0.1-1309090026.el6    | 141                        |
| contrail-api-lib             | 0.1-1309090026.el6    | 141                        |
| contrail-api-venv            | 0.1-1309080539.el6    | 141                        |
| contrail-control             | 2012.0-1309090026.el6 | 141                        |
| contrail-database            | 0.1-1309050028        | 141                        |
| contrail-dns                 | 1-1309090026.el6      | 141                        |
| contrail-fabric-utils        | 1-1309090026          | 141                        |
| contrail-libs                | 1-1309090026.el6      | 141                        |
| contrail-nodejs              | 0.8.15-1309090026.el6 | 141                        |
| contrail-openstack-analytics | 0.1-1309090026.el6    | 141                        |
| contrail-openstack-cfgm      | 0.1-1309090026.el6    | 141                        |
| contrail-openstack-control   | 0.1-1309090026.el6    | 141                        |

## Sample Output

The following example shows version and build information for only the installed contrail components.

## Sample Output

```

root@host> contrail-version | grep contrail
Package                                Version                                Build-ID | Repo | RPM Name
-----
contrail-analytics                     1-1309090026.el6                      141
contrail-analytics-venv                 0.1-1309062310.el6                     141
contrail-api                            0.1-1309090026.el6                     141
contrail-api-lib                       0.1-1309090026.el6                     141
contrail-api-venv                      0.1-1309080539.el6                     141
contrail-control                       2012.0-1309090026.el6                  141
contrail-database                      0.1-1309050028                         141
contrail-dns                           1-1309090026.el6                       141
contrail-fabric-utils                  1-1309090026                           141
contrail-libs                          1-1309090026.el6                       141
contrail-nodejs                        0.8.15-1309090026.el6                 141
contrail-openstack-analytics           0.1-1309090026.el6                     141
contrail-openstack-cfgm                0.1-1309090026.el6                     141
contrail-openstack-control             0.1-1309090026.el6                     141
contrail-openstack-database            0.1-1309090026.el6                     141
contrail-openstack-webui               0.1-1309090026.el6                     141
contrail-setup                         1-1309090026.el6                       141
contrail-webui                         1-1309090026                           141
openstack-quantum-contrail             2013.2-1309090026                      141

```

## Release Information

Command introduced in Contrail Release 1.0.

## service (Managing Services)

### IN THIS SECTION

- [Syntax | 1056](#)
- [Description | 1056](#)
- [Options | 1056](#)
- [Required Privilege Level | 1057](#)
- [Sample Output | 1057](#)
- [Release Information | 1057](#)

### Syntax

```
service contrail-service ( start | stop | restart | status )
```

### Description

Start, stop, or restart a Contrail service. Display the status of a Contrail service.

All contrail services are managed by the process supervisor, which is open source software written in Python. Each Contrail node type, such as compute, control, and so on, has an instance of supervisor that, when running, launches Contrail services as child processes. All supervisor instances display in `contrail-status` output with the prefix `supervisor`. If the supervisor instance of a particular node type is not up, none of the services for that node type are up. For more details about the open source supervisor process, see <http://www.supervisord.org>.

### Options

- `start`—start a named service.
- `stop`—stop a named service.
- `restart`—stop and restart a named service.
- `status`—display the status of a named service.



## Required Privilege Level

admin

## Sample Output

The following examples show usage for the `contrail-collector` service, which is only configured on nodes that have the roles of **analytics**, **configuration**, **web-ui**, or **database**.

## Sample Output

```
[root@host# service supervisor-analytics status
supervisord (pid 32116) is running... [
[root@host]# service contrail-collector restart

contrail-collector: stopped
contrail-collector: started

[root@host]# service contrail-collector stop

contrail-collector: stopped

[root@host]# service contrail-collector start

contrail-collector: started

[root@host]# service contrail-collector status

contrail-collector          RUNNING    pid 20071, uptime 0:00:04
```

## Release Information

Standard Linux command used for managing and viewing services in Contrail Controller Release 1.0.

## Backing Up Contrail Databases Using JSON Format

### IN THIS SECTION

- [Preliminary Cautions | 1058](#)
- [Simple Backup Using JSON Format | 1058](#)
- [Restore Simple Database Backup | 1059](#)
- [Example Backup and Restore With JSON | 1060](#)

This document shows how to backup Contrail databases (Cassandra and Zookeeper) using a JSON format. Instructions are given for both non-containerized and containerized versions of Contrail, starting with Contrail 4.0.

### Preliminary Cautions



**CAUTION:** Because the state of the Contrail database is associated with other system databases, such as OpenStack databases, database backups must be consistent across all systems and database changes associated with northbound APIs must be stopped on all systems before performing any backup operation. For example, you might block the external VIP for northbound APIs at the load balancer level, such as HAproxy.

### Simple Backup Using JSON Format

Perform a simple backup (database dump). Working from a controller node, use `db_json_exim.py`, located at `/usr/lib/python2.7/dist-packages/cfgm_common`.

**NOTE:** The controller node for non-containerized Contrail is a virtual machine (VM). The controller node for containerized Contrail is a controller container.

```
cd /usr/lib/python2.7/dist-packages/cfgm_common
```

```
python db_json_exim.py --export-to db-dump.json
```

- To see a cleaner version of the dump.

```
cat db-dump.json | python -m json.tool | less
```

- To omit keyspace in the dump, for example, to share with Juniper.

```
python db_json_exim.py --export-to db-dump.json --omit-keyspace dm_keyspace
```

## Restore Simple Database Backup

Use the following steps to restore a system from a simple backup.

1. Stop supervisor-config on all controllers, if present, or ensure it is already stopped.

```
service supervisor-config stop
```

2. Stop Cassandra on all config-db controllers or ensure it is already stopped.

```
service cassandra stop
```

3. Stop Zookeeper on all controllers or ensure it is already stopped.

```
service zookeeper stop
```

4. Stop Kafka on all controllers. Be sure to check analytics controllers.

```
service kafka stop
```

5. Stop contrail-hamon on all controllers, if it is running on controllers.

```
service contrail-hamon stop
```

6. Backup the Zookeeper data directory on all controllers.

```
cd /var/lib/zookeeper/
```

```
cp -R version-2/ version-2-save
```

7. Backup the Cassandra data directory on all controllers.

```
cd /var/lib/
```

```
cp -R cassandra cassandra-save
```

8. Wipe out the Zookeeper data directory contents on all controllers.

```
rm -rf /var/lib/zookeeper/version-2/*
```

9. Wipe out the Cassandra data directory contents on all controllers.

```
rm -rf /var/lib/cassandra/*
```

10. Start Zookeeper on all controllers.

```
service zookeeper start
```

**11. Start Cassandra on all controllers.**

```
service cassandra start
```

**12. Run python db\_json\_exim.py --import-from db-dump.json on any one controller.**

```
cd /usr/lib/python2.7/dist-packages/cfgm_common
python db_json_exim.py --import-from db-dump.json
```

**13. Start supervisor-config on all controllers (if present).**

```
service supervisor-config start
```

**14. Start Kafka on all controllers (check in analytics controllers).**

```
service kafka start
```

**15. Start contrail-hamon on all controllers, if previously stopped.**

```
service contrail-hamon start
```

**Example Backup and Restore With JSON**

This section provides an example of a simple database backup and restore of a system that has three controllers with config-db and separate IPs with the following host IDs:

- 5b5s42
- 5b5s43
- 5b5s44

**Example: Perform Simple Backup**

```
root@5b5s42:~# python db_json_exim.py --export-to db-dump.json
root@5b5s42:~# cat db-dump.json | python -m json.tool | less
{
  "cassandra": {
    "config_db_uuid": {
      "obj_fq_name_table": {
        "access_control_list": {
          <snip>
```

## Example: Perform Restore

1. Stop supervisor-config on all controllers, if present.

```

Non-Containerized Version: root@5b5s42:~# service supervisor-config stop
supervisor-config stop/waiting
root@5b5s42:~#
root@5b5s43:~# service supervisor-config stop
supervisor-config stop/waiting
root@5b5s43:~#
root@5b5s44:~# service supervisor-config stop
supervisor-config stop/waiting
root@5b5s44:~#

```

```

Containerized Version:
root@host-4.1:~# docker ps
CONTAINER ID        IMAGE
COMMAND            CREATED            STATUS             PORTS             NAMES
8802395bc033      172.30.109.59:5100/contrail410-contrail-analytics:mainline  "/lib/
systemd/syst..."  7 weeks ago       Up 2 weeks
f5aed0a2efc3      172.30.109.59:5100/contrail410-contrail-analyticsdb:mainline  "/lib/
systemd/syst..."  7 weeks ago       Up 2 weeks
0ff200b12112      172.30.109.59:5100/contrail410-contrail-controller:mainline  "/lib/
systemd/syst..."  7 weeks ago       Up 2 weeks
6fec888f8145      registry:2
entrypoint.sh /e..."  7 weeks ago       Up 2 weeks
root@host-4.1:~# docker exec -it 0ff200b12112 /bin/bash

```

2. Stop Cassandra on all controllers.

```

root@5b5s42:~# service cassandra stop
root@5b5s42:~#
root@5b5s43:~# service cassandra stop
root@5b5s43:~#
root@5b5s44:~# service cassandra stop
root@5b5s44:~#

```

3. Stop Zookeeper on all controllers.

```
root@5b5s42:~# service zookeeper stop
zookeeper stop/waiting
root@5b5s42:~#
root@5b5s43:~# service zookeeper stop
zookeeper stop/waiting
root@5b5s43:~#
root@5b5s44:~# service zookeeper stop
zookeeper stop/waiting
root@5b5s44:~#
```

4. Stop Kafka on all controllers.

```
root@5b5s42:~# service kafka stop
kafka: stopped
root@5b5s42:~#
root@5b5s43:~# service kafka stop
kafka: stopped
root@5b5s43:~#
root@5b5s44:~# service kafka stop
kafka: stopped
root@5b5s44:~#
```

5. Stop contrail-hamon on all controllers, if present.

```
root@5b5s42:~# service contrail-hamon stop
contrail-hamon stop/waiting
root@5b5s43:~# service contrail-hamon stop
contrail-hamon stop/waiting
root@5b5s44:~# service contrail-hamon stop
contrail-hamon stop/waiting
```

6. Backup the Zookeeper data directory on all controllers.

```
root@5b5s42:~# cd /var/lib/zookeeper/
root@5b5s42:/var/lib/zookeeper# cp -R version-2/ version-2-save
root@5b5s42:/var/lib/zookeeper#
root@5b5s43:~# cd /var/lib/zookeeper/
```

```

root@5b5s43:/var/lib/zookeeper# cp -R version-2/ version-2-save
root@5b5s43:/var/lib/zookeeper#
root@5b5s44:~# cd /var/lib/zookeeper/
root@5b5s44:/var/lib/zookeeper# cp -R version-2/ version-2-save
root@5b5s44:/var/lib/zookeeper#

```

7. Backup the Cassandra data directory on all controllers.

```

root@5b5s42:~# cd /var/lib/
root@5b5s42:/var/lib# cp -R cassandra cassandra-save
root@5b5s42:/var/lib#
root@5b5s43:~# cd /var/lib/
root@5b5s43:/var/lib# cp -R cassandra cassandra-save
root@5b5s43:/var/lib#
root@5b5s44:~# cd /var/lib/
root@5b5s44:/var/lib# cp -R cassandra/ cassandra-save
root@5b5s44:/var/lib#

```

8. Wipe out the Zookeeper data directory contents on all controllers.

```

root@5b5s42:~# rm -rf /var/lib/zookeeper/version-2/*
root@5b5s42:~#
root@5b5s43:~# rm -rf /var/lib/zookeeper/version-2/*
root@5b5s43:~#
root@5b5s44:~# rm -rf /var/lib/zookeeper/version-2/*
root@5b5s44:~#

```

9. Wipe out the Cassandra data directory contents on all controllers.

```

root@5b5s42:~# rm -rf /var/lib/cassandra/*
root@5b5s42:~#
root@5b5s43:~# rm -rf /var/lib/cassandra/*
root@5b5s43:~#
root@5b5s44:~# rm -rf /var/lib/cassandra/*
root@5b5s44:~#

```

**10. Start Zookeeper on all controllers.**

```
root@5b5s42:~# service zookeeper start
zookeeper start/running, process 14180
root@5b5s42:~#
root@5b5s43:~# service zookeeper start
zookeeper start/running, process 11635
root@5b5s43:~#
root@5b5s44:~# service zookeeper start
zookeeper start/running, process 28040
root@5b5s44:~#
```

**11. Start Cassandra on all controllers.**

```
root@5b5s42:~# service cassandra start
root@5b5s42:~#
root@5b5s43:~# service cassandra start
root@5b5s43:~#
root@5b5s44:~# service cassandra start
root@5b5s44:~#
```

**12. Run python db\_json\_exim.py --import-from db-dump.json on any *one* controller.**

```
root@5b5s42:~# python db_json_exim.py --import-from db-dump.json
root@5b5s42:~#
```

**13. Start supervisor-config on all controllers, if present.**

```
root@5b5s42:~# service supervisor-config start
supervisor-config start/running, process 19286
root@5b5s42:~#
root@5b5s43:~# service supervisor-config start
supervisor-config start/running, process 28937
root@5b5s43:~#
root@5b5s44:~# service supervisor-config start
supervisor-config start/running, process 21242
root@5b5s44:~#
```



**14.** Start Kafka on all controllers.

```
root@5b5s42:~# service kafka start
kafka: started
root@5b5s42:~#
root@5b5s43:~# service kafka start
kafka: started
root@5b5s43:~#
root@5b5s44:~# service kafka start
kafka: started
root@5b5s44:~#
```

**15.** Start contrail-hamon on all controllers, if present.

```
root@5b5s42:~# service contrail-hamon start
contrail-hamon start/running, process 1379
root@5b5s42:~#
root@5b5s43:~# service contrail-hamon start
contrail-hamon start/running, process 1230
root@5b5s43:~#
root@5b5s44:~# service contrail-hamon start
contrail-hamon start/running, process 26843
root@5b5s44:~#
```

# Contrail Application Programming Interfaces (APIs)

## IN THIS CHAPTER

- [Contrail Analytics Application Programming Interfaces \(APIs\) and User-Visible Entities \(UVEs\) | 1066](#)
- [Log and Flow Information APIs | 1080](#)
- [Working with Neutron | 1088](#)
- [Support for Amazon VPC APIs on Contrail OpenStack | 1092](#)

## Contrail Analytics Application Programming Interfaces (APIs) and User-Visible Entities (UVEs)

### IN THIS SECTION

- [User-Visible Entities | 1067](#)
- [Common UVEs in Contrail | 1068](#)
- [Virtual Network UVE | 1068](#)
- [Virtual Machine UVE | 1069](#)
- [vRouter UVE | 1069](#)
- [UVEs for Contrail Nodes | 1070](#)
- [Wild Card Query of UVEs | 1070](#)
- [Filtering UVE Information | 1070](#)

The Contrail **analytics-api** server provides a REST API interface to extract the operational state of the Contrail system.

APIs are used by the Contrail Web user interface to present the operational state to users. Other applications might also use the server's REST APIs for analytics or other uses.

This section describes some of the more common APIs and their uses. To see all of the available APIs, navigate the URL tree at the REST interface, starting at the root `http://<ip>:<analytics-api-port>`. You can also view Contrail API information at: <http://configuration-schema-documentation.s3-website-us-west-1.amazonaws.com/R3.2/>.

## User-Visible Entities

In Contrail, a User-Visible Entity (UVE) is an object entity that might span multiple components in Contrail and might require aggregation before the complete information of the UVE is presented. Examples of UVEs in Contrail are virtual network, virtual machine, vRouter, and similar objects. Complete operational information for a virtual network might span multiple vRouters, config nodes, control nodes, and the like. The analytics-api server aggregates all of this information through REST APIs.

To get information about a UVE, you must have the UVE type and the UVE key. In Contrail, UVEs are identified by type, such as virtual network, virtual machine, vRouter, and so on. A system-wide unique key is associated with each UVE. The key type could be different, based on the UVE type. For example, perhaps a virtual network uses its name as its UVE key, and in the same system, a virtual machine uses its UUID as its key.

The URL `/analytics/uves` shows the list of all UVE types available in the system.

The following is sample output from `/analytics/uves`:

```
[
  {
    href: "http://<system IP>:8081/analytics/uves/xmpp-peers",
    name: "xmpp-peers"
  },
  {
    href: "http://<system IP>:8081/analytics/uves/service-instances",
    name: "service-instances"
  },
  {
    href: "http://<system IP>:8081/analytics/uves/config-nodes",
    name: "config-nodes"
  },
  {
    href: "http://<system IP>:8081/analytics/uves/virtual-machines",
    name: "virtual-machines"
  },
  {
    href: "http://<system IP>:8081/analytics/uves/bgp-routers",
    name: "bgp-routers"
  }
]
```

```

},
{
href: "http://<system IP>:8081/analytics/uves/collectors",
name: "collectors"
},
{
href: "http://<system IP>:8081/analytics/uves/service-chains",
name: "service-chains"
},
{
href: "http://<system IP>:8081/analytics/uves/generators",
name: "generators"
},
{
href: "http://<system IP>:8081/analytics/uves/bgp-peers",
name: "bgp-peers"
},
{
href: "http://<system IP>:8081/analytics/uves/virtual-networks",
name: "virtual-networks"
},
{
href: "http://<system IP>:8081/analytics/uves/vrouters",
name: "vrouters"
},
{
href: "http://<system IP>:8081/analytics/uves/dns-nodes",
name: "dns-nodes"
}
]

```

## Common UVEs in Contrail

This section presents descriptions of some common UVEs in Contrail.

### Virtual Network UVE

This UVE provides information associated with a virtual network, such as:

- list of networks connected to this network
- list of virtual machines spawned in this network
- list of access control lists (ACLs) associated with this virtual network

- global input and output statistics
- input and output statistics per virtual network pair

The REST API to get a UVE for a specific virtual network is through HTTP GET, using the URL:

`/analytics/uves/virtual-network/<key>`

The REST API to get UVEs for all virtual machines is through HTTP GET, using the URL:

`/analytics/uves/virtual-networks`

## Virtual Machine UVE

This UVE provides information associated with a virtual machine, such as:

- list of interfaces in this virtual machine
- list of floating IPs associated with each interface
- input and output statistics

The REST API to get a UVE for a specific virtual machine is through HTTP GET, using the URL:

`/analytics/uves/virtual-machine/<key>`

The REST API to get UVEs for all virtual machines is through HTTP GET, using the URL:

`/analytics/uves/virtual-machines`

## vRouter UVE

This UVE provides information associated with a vRouter, such as:

- virtual networks present on this vRouter
- virtual machines spawned on the server of this vRouter
- statistics of the traffic flowing through this vRouter

The REST API to get a UVE for a specific vRouter is through HTTP GET, using the URL:

`/analytics/uves/vrouter/<key>`

The REST API to get UVEs for all virtual machines is through HTTP GET, using the URL:

`/analytics/uves/vrouters`

## UVEs for Contrail Nodes

There are multiple node types in Contrail (including the node type vRouter previously described). Other node types include control node, config node, analytics node, and compute node.

There is a UVE for each node type. The common information associated with each node UVE includes:

- the IP address of the node
- a list of processes running on the node
- the CPU and memory utilization of the running processes

Each UVE also has node-specific information, such as:

- the control node UVE has information about its connectivity to the vRouter and other control nodes
- the analytics node UVE has information about the number of generators connected

The REST API to get a UVE for a specific config node is through HTTP GET, using the URL:

```
/analytics/uves/config-node/<key>
```

The REST API to get UVEs for all config nodes is through HTTP GET, using the URL:

```
/analytics/uves/config-nodes
```

**NOTE:** Use similar syntax to get UVEs for each of the different types of nodes, substituting the node type that you want in place of config-node.

## Wild Card Query of UVEs

You can use wildcard queries when you want to get multiple UVEs at the same time. Example queries are the following:

The following HTTP GET with wildcard retrieves all virtual network UVEs:

```
/analytics/uves/virtual-network/*
```

The following HTTP GET with wildcard retrieves all virtual network UVEs with name starting with project1:

```
/analytics/uves/virtual-network/project1*
```

## Filtering UVE Information

It is possible to retrieve filtered UVE information. The following flags enable you to retrieve partial, filtered information about UVEs.

Supported filter flags include:

1. `sfilt` : filter by source (usually the hostname of the generator)
2. `mfilt` : filter by module (the module name of the generator)
3. `cfilt` : filter by content, useful when only part of a UVE needs to be retrieved
4. `kfilt` : filter by UVE keys, useful to get multiple, but not all, UVEs of a particular type

### Examples

The following HTTP GET with filter retrieves information about virtual network `vn1` as provided by the source `src1`:

```
/analytics/uves/virtual-network/vn1?sfilt=src1
```

The following HTTP GET with filter retrieves information about virtual network `vn1` as provided by all `ApiServer` modules:

```
/analytics/uves/virtual-network/vn1?mfilt=ApiServer
```

### Example Output: Virtual Network UVE

Example output for a virtual network UVE:

```
[user@host ~]# curl <system IP>:8081/analytics/virtual-network/default-domain:demo:front-end |
python -mjson.tool
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100 2576  100 2576    0     0  152k      0  --:--:--  --:--:--  --:--:--  157k
{
  "UveVirtualNetworkAgent": {
    "acl": [
      [
        {
          "@type": "string"
        },
        "a3s18:VRouterAgent"
      ]
    ],
    "in_bytes": {
      "#text": "2232972057",
      "@aggtype": "counter",
```

```

    "@type": "i64"
  },
  "in_stats": {
    "@aggtype": "append",
    "@type": "list",
    "list": {
      "@size": "3",
      "@type": "struct",
      "UveInterVnStats": [
        {
          "bytes": {
            "#text": "2114516371",
            "@type": "i64"
          },
          "other_vn": {
            "#text": "default-domain:demo:back-end",
            "@aggtype": "listkey",
            "@type": "string"
          },
          "tpkts": {
            "#text": "5122001",
            "@type": "i64"
          }
        },
        {
          "bytes": {
            "#text": "1152123",
            "@type": "i64"
          },
          "other_vn": {
            "#text": "__FABRIC__",
            "@aggtype": "listkey",
            "@type": "string"
          },
          "tpkts": {
            "#text": "11323",
            "@type": "i64"
          }
        }
      ],
      {
        "bytes": {
          "#text": "8192",
          "@type": "i64"
        }
      }
    ]
  }
}

```



```

    },
    "other_vn": {
        "#text": "default-domain:demo:front-end",
        "@aggtype": "listkey",
        "@type": "string"
    },
    "tpkts": {
        "#text": "50",
        "@type": "i64"
    }
}
]
}
},
"in_tpkts": {
    "#text": "5156342",
    "@aggtype": "counter",
    "@type": "i64"
},
"interface_list": {
    "@aggtype": "union",
    "@type": "list",
    "list": {
        "@size": "1",
        "@type": "string",
        "element": [
            "tap2158f77c-ec"
        ]
    }
},
"out_bytes": {
    "#text": "2187615961",
    "@aggtype": "counter",
    "@type": "i64"
},
"out_stats": {
    "@aggtype": "append",
    "@type": "list",
    "list": {
        "@size": "4",
        "@type": "struct",
        "UveInterVnStats": [
            {

```

```

"bytes": {
  "#text": "2159083215",
  "@type": "i64"
},
"other_vn": {
  "#text": "default-domain:demo:back-end",
  "@aggtype": "listkey",
  "@type": "string"
},
"tpkts": {
  "#text": "5143693",
  "@type": "i64"
}
},
{
  "bytes": {
    "#text": "1603041",
    "@type": "i64"
  },
  "other_vn": {
    "#text": "__FABRIC__",
    "@aggtype": "listkey",
    "@type": "string"
  },
  "tpkts": {
    "#text": "9595",
    "@type": "i64"
  }
},
{
  "bytes": {
    "#text": "24608",
    "@type": "i64"
  },
  "other_vn": {
    "#text": "__UNKNOWN__",
    "@aggtype": "listkey",
    "@type": "string"
  },
  "tpkts": {
    "#text": "408",
    "@type": "i64"
  }
}

```

```

    },
    {
      "bytes": {
        "#text": "8192",
        "@type": "i64"
      },
      "other_vn": {
        "#text": "default-domain:demo:front-end",
        "@aggtype": "listkey",
        "@type": "string"
      },
      "tpkts": {
        "#text": "50",
        "@type": "i64"
      }
    }
  ]
}
},
"out_tpkts": {
  "#text": "5134830",
  "@aggtype": "counter",
  "@type": "i64"
},
"virtualmachine_list": {
  "@aggtype": "union",
  "@type": "list",
  "list": {
    "@size": "1",
    "@type": "string",
    "element": [
      "dd09f8c3-32a8-456f-b8cc-fab15189f50f"
    ]
  }
}
},
"UveVirtualNetworkConfig": {
  "connected_networks": {
    "@aggtype": "union",
    "@type": "list",
    "list": {
      "@size": "1",
      "@type": "string",
      "element": [

```

```

        "default-domain:demo:back-end"
    ]
}
},
"routing_instance_list": {
    "@aggtype": "union",
    "@type": "list",
    "list": {
        "@size": "1",
        "@type": "string",
        "element": [
            "front-end"
        ]
    }
},
"total_acl_rules": [
    [
        {
            "#text": "3",
            "@type": "i32"
        },
        ":",
        "a3s14:Schema"
    ]
]
}
}
}

```

### Example Output: Virtual Machine UVE

Example output for a virtual machine UVE:

```

[user@host ~]# curl <system IP>:8081/analytics/virtual-machine/
f38eb47e-63d2-4b39-80de-8fe68e6af1e4 | python -mjson.tool
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload  Total   Spent    Left     Speed
100  736  100  736    0     0  160k      0  --:--:--  --:--:--  --:--:--  179k
{
  "UveVirtualMachineAgent": {
    "interface_list": [
      [
        {

```

```

    "@type": "list",
    "list": {
      "@size": "1",
      "@type": "struct",
      "VmInterfaceAgent": [
        {
          "in_bytes": {
            "#text": "2188895907",
            "@aggtype": "counter",
            "@type": "i64"
          },
          "in_pkts": {
            "#text": "5130901",
            "@aggtype": "counter",
            "@type": "i64"
          },
          "ip_address": {
            "#text": "192.168.2.253",
            "@type": "string"
          },
          "name": {
            "#text": "f38eb47e-63d2-4b39-80de-8fe68e6af1e4:ccb085a0-
c994-4034-be0f-6fd5ad08ce83",
            "@type": "string"
          },
          "out_bytes": {
            "#text": "2201821626",
            "@aggtype": "counter",
            "@type": "i64"
          },
          "out_pkts": {
            "#text": "5153526",
            "@aggtype": "counter",
            "@type": "i64"
          },
          "virtual_network": {
            "#text": "default-domain:demo:back-end",
            "@aggtype": "listkey",
            "@type": "string"
          }
        }
      ]
    }
  }

```

```

    },
    "a3s19:VRouterAgent"
  ]
]
}
}

```

### Example Output: vRouter UVE

Example output for a vRouter UVE:

```

[user@host ~]# curl <system IP>:8081/analytics/vrouter/a3s18 | python -mjson.tool
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  706  100  706    0     0  142k    0  --:--:--  --:--:--  --:--:--  172k
{
  "VrouterAgent": {
    "collector": [
      [
        {
          "#text": "10.xx.17.1",
          "@type": "string"
        },
        "a3s18:VRouterAgent"
      ]
    ],
    "connected_networks": [
      [
        {
          "@type": "list",
          "list": {
            "@size": "1",
            "@type": "string",
            "element": [
              "default-domain:demo:front-end"
            ]
          }
        },
        "a3s18:VRouterAgent"
      ]
    ],
    "interface_list": [

```

```

    [
      {
        "@type": "list",
        "list": {
          "@size": "1",
          "@type": "string",
          "element": [
            "tap2158f77c-ec"
          ]
        }
      },
      "a3s18:VRouterAgent"
    ]
  ],
  "virtual_machine_list": [
    [
      {
        "@type": "list",
        "list": {
          "@size": "1",
          "@type": "string",
          "element": [
            "dd09f8c3-32a8-456f-b8cc-fab15189f50f"
          ]
        }
      },
      "a3s18:VRouterAgent"
    ]
  ],
  "xmpp_peer_list": [
    [
      {
        "@type": "list",
        "list": {
          "@size": "2",
          "@type": "string",
          "element": [
            "10.xx.17.2",
            "10.xx.17.3"
          ]
        }
      },
      "a3s18:VRouterAgent"
    ]
  ]

```

```

    ]
  ]
}
}

```

## RELATED DOCUMENTATION

[Juniper Contrail Configuration API Server Documentation](#)

[Log and Flow Information APIs | 1080](#)

## Log and Flow Information APIs

### IN THIS SECTION

- [HTTP GET APIs | 1080](#)
- [HTTP POST API | 1081](#)
- [POST Data Format Example | 1081](#)
- [Query Types | 1083](#)
- [Examining Query Status | 1083](#)
- [Examining Query Chunks | 1084](#)
- [Example Queries for Log and Flow Data | 1084](#)

In Contrail, log and flow analytics information is collected and stored using a horizontally scalable Contrail collector and NoSQL database. The `analytics-api` server provides REST APIs to extract this information using queries. The queries use well-known SQL syntax, hiding the underlying complexity of the NoSQL tables.

### HTTP GET APIs

Use the following GET APIs to identify tables and APIs available for querying.

`/analytics/tables` -- lists the SQL-type tables available for querying, including the hrefs for each of the tables



`/analytics/table/<table>` -- lists the APIs available to get information for a given table

`/analytics/table/<table>/schema` -- lists the schema for a given table

## HTTP POST API

Use the following POST API information to extract data from a table.

`/analytics/query` -- format your query using the following SQL syntax:

1. `SELECT field1, field2 ...`
2. `FROM table1`
3. `WHERE field1 = value1 AND field2 = value2 ...`
4. `FILTER BY ...`
5. `SORT BY ...`
6. `LIMIT n`

Additionally, it is mandatory to include the start time and the end time for the data range to define the time period for the query data. The parameters of the query are passed through POST data, using the following fields:

1. `start_time` — the start of the time period
2. `end_time` — the end of the time period
3. `table` — the table from which to extract data
4. `select_fields` — the columns to display in the extracted data
5. `where` — the list of match conditions

## POST Data Format Example

The POST data is in JSON format, stored in an `id1` file. A sample file is displayed in the following.

**NOTE:** The result of the query API is also in JSON format.

```
/*
 * Copyright (c) 2013 Juniper Networks, Inc. All rights reserved.
 */
```

```

/*
 * query_rest.idl
 *
 * IDL definitions for query engine REST API
 *
 * PLEASE NOTE: After updating this file, do update json_parse.h
 *
 */

enum match_op {
    EQUAL = 1,
    NOT_EQUAL = 2,
    IN_RANGE = 3,
    NOT_IN_RANGE = 4, // not supported currently
    // following are only for numerical column fields
    LEQ = 5, // column value is less than or equal to filter value
    GEQ = 6, // column value is greater than or equal to filter value
    PREFIX = 7, // column value has the "value" field as prefix
    REGEX_MATCH = 8 // for filters only
}

enum sort_op {
    ASCENDING = 1,
    DESCENDING = 2,
}

struct match {
    1: string name;
    2: string value;
    3: match_op op;
    4: optional string value2; // this is for only RANGE match
}

typedef list<match> term; (AND of match)

enum flow_dir_t {
    EGRESS = 0,
    INGRESS = 1
}

struct query {
    1: string table; // Table to query (FlowSeriesTable, MessageTable, ObjectVNTTable,
ObjectVMTable, FlowRecordTable)
    2: i64 start_time; // Microseconds in UTC since Epoch
}

```

```

3: i64 end_time; // Microseconds in UTC since Epoch
4: list<string>> select_fields; // List of SELECT fields
5: list<term> where; // WHERE (OR of terms)
6: optional sort_op sort;
7: optional list<string> sort_fields;
8: optional i32 limit;
9: optional flow_dir_t dir; // direction of flows being queried
10: optional list<match> filter; // filter the processed result by value
}

struct flow_series_result_entry {
  1: optional i64 T; // Timestamp of the flow record
  2: optional string sourcevn;
  3: optional string sourceip;
  4: optional string destvn;
  5: optional string destip;
  6: optional i32 protocol;
  7: optional i32 sport;
  8: optional i32 dport;
  9: optional flow_dir_t direction_ing;
  10: optional i64 packets; // mutually exclusive to 12,13
  11: optional i64 bytes; // mutually exclusive to 12,13
  12: optional i64 sum_packets; // represented as "sum(packets)" in JSON
  13: optional i64 sum_bytes; // represented as "sum(bytes)" in JSON
};
typedef list<flow_series_result_entry> flow_series_result;

```

## Query Types

The `analytics-api` supports two types of queries. Both types use the same POST parameters as described in POST API.

- `sync` — Default query mode. The results are sent inline with the query processing.
- `async` — To execute a query in `async` mode, attach the following header to the POST request: `Expect: 202-accepted`.

## Examining Query Status

For an asynchronous query, the `analytics-api` responds with the code: `202 Accepted`. The response contents are a status entity href URL of the form: `/analytics/query/<QueryID>`. The `QueryID` is assigned by the `analytics-api`. To view the response contents, poll the status entity by performing a GET action on the URL. The status entity has a variable named `progress`, with a number between 0 and 100, representing

the approximate percentage completion of the query. When progress is 100, the query processing is complete.

## Examining Query Chunks

The status entity has an element named `chunks` that lists portions (chunks) of query results. Each element of this list has three fields: `start_time`, `end_time`, `href`. The `analytics-api` determines how many chunks to list to represent the query data. A chunk can include an empty string ("") to indicate that the data query is not yet available. If a partial result is available, the chunk href is of the form: `/analytics/query/<QueryID>/chunk-partial/<chunk number>`. When the final result of a chunk is available, the href is of the form: `/analytics/query/<QueryID>/chunk-final/<chunk number>`.

## Example Queries for Log and Flow Data

The following example query lists the tables available for query.

```
[root@host ~]# curl 127.0.0.1:8081/analytics/tables | python -mjson.tool
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  846  100  846    0     0  509k      0  --:--:--  --:--:--  --:--:--  826k
[
  {
    "href": "http://127.0.0.1:8081/analytics/table/MessageTable",
    "name": "MessageTable"
  },
  {
    "href": "http://127.0.0.1:8081/analytics/table/ObjectVNTable",
    "name": "ObjectVNTable"
  },
  {
    "href": "http://127.0.0.1:8081/analytics/table/ObjectVMTable",
    "name": "ObjectVMTable"
  },
  {
    "href": "http://127.0.0.1:8081/analytics/table/ObjectVRouter",
    "name": "ObjectVRouter"
  },
  {
    "href": "http://127.0.0.1:8081/analytics/table/ObjectBgpPeer",
    "name": "ObjectBgpPeer"
  },
  {
    "href": "http://127.0.0.1:8081/analytics/table/ObjectRoutingInstance",
```

```

    "name": "ObjectRoutingInstance"
  },
  {
    "href": "http://127.0.0.1:8081/analytics/table/ObjectXmppConnection",
    "name": "ObjectXmppConnection"
  },
  {
    "href": "http://127.0.0.1:8081/analytics/table/FlowRecordTable",
    "name": "FlowRecordTable"
  },
  {
    "href": "http://127.0.0.1:8081/analytics/table/FlowSeriesTable",
    "name": "FlowSeriesTable"
  }
]

```

The following example query lists details for the table named MessageTable.

```

[root@host ~]# curl 127.0.0.1:8081/analytics/table/MessageTable | python -mjson.tool
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload  Total  Spent    Left  Speed
100   192   100   192    0    0   102k    0 --:--:-- --:--:-- --:--:--  187k
[
  {
    "href": "http://127.0.0.1:8081/analytics/table/MessageTable/schema",
    "name": "schema"
  },
  {
    "href": "http://127.0.0.1:8081/analytics/table/MessageTable/column-values",
    "name": "column-values"
  }
]

```

The following example query lists the schema for the table named MessageTable.

```

[root@host ~]# curl 127.0.0.1:8081/analytics/table/MessageTable/schema | python -mjson.tool
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload  Total  Spent    Left  Speed
100    630   100    630    0    0   275k    0 --:--:-- --:--:-- --:--:--  307k
{
  "columns": [

```

```
{
  "datatype": "int",
  "index": "False",
  "name": "MessageTS"
},
{
  "datatype": "string",
  "index": "True",
  "name": "Source"
},
{
  "datatype": "string",
  "index": "True",
  "name": "ModuleId"
},
{
  "datatype": "string",
  "index": "True",
  "name": "Category"
},
{
  "datatype": "int",
  "index": "True",
  "name": "Level"
},
{
  "datatype": "int",
  "index": "False",
  "name": "Type"
},
{
  "datatype": "string",
  "index": "True",
  "name": "Messagetype"
},
{
  "datatype": "int",
  "index": "False",
  "name": "SequenceNum"
},
{
  "datatype": "string",
  "index": "False",
```

```

        "name": "Context"
    },
    {
        "datatype": "string",
        "index": "False",
        "name": "Xmlmessage"
    }
],
"type": "LOG"
}

```

The following set of example queries explore a message table.

```

root@a6s45:~# cat filename
{ "end_time": "now" , "select_fields": ["MessageTS", "Source", "ModuleId", "Category",
"Messageype", "SequenceNum", "Xmlmessage", "Type", "Level", "NodeType", "InstanceId"] , "sort":
1 , "sort_fields": ["MessageTS"] , "start_time": "now-10m" , "table": "MessageTable" , "where":
{"name": "ModuleId", "value": "contrail-control", "op": 1, "suffix": null, "value2": null},
{"name": "Messageype", "value": "BGPRouterInfo", "op": 1, "suffix": null, "value2": null} }

root@a6s45:~#
root@a6s45:~# curl -X POST --data @filename 127.0.0.1:8081/analytics/query --header "Content-
Type:application/json" | python -mjson.tool
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload  Total  Spent    Left  Speed
100  9765    0  9297  100   468    9168    461   0:00:01  0:00:01  ---:---:--  9177
{
  "value": [
    {
      "Category": null,
      "InstanceId": "0",
      "Level": 2147483647,
      "MessageTS": 1428442589947392,
      "Messageype": "BGPRouterInfo",
      "ModuleId": "contrail-control",
      "NodeType": "Control",
      "SequenceNum": 1302,
      "Source": "a6s45",
      "Type": 6,
      "Xmlmessage": "<BGPRouterInfo type=""><data type=""><BgpRouterState><name type=""
>a6s45</name><cpu_info type=""><CpuLoadInfo><num_cpu type="">4</num_cpu

```

```
><meminfo type=""><MemInfo><virt type="">438436</virt><peakvirt type=""
>561048</peakvirt><res type="">12016</res></MemInfo></meminfo><cpu_share
type="">0.0416667</cpu_share></CpuLoadInfo></cpu_info><cpu_share type=""
>0.0416667</cpu_share></BgpRouterState></data></BGPRouterInfo>"
    },
    {
      "Category": null,
      "InstanceId": "0",
      "Level": 2147483647,
    }
  ]
}
```

## Working with Neutron

### IN THIS SECTION

- [Data Structure | 1088](#)
- [Network Sharing in Neutron | 1089](#)
- [Commands for Neutron Network Sharing | 1090](#)
- [Support for Neutron APIs | 1090](#)
- [Contrail Neutron Plugin | 1091](#)
- [DHCP Options | 1091](#)
- [Incompatibilities | 1092](#)

OpenStack's networking solution, Neutron, has representative elements for Contrail elements for Network (VirtualNetwork), Port (VirtualMachineInterface), Subnet (IpamSubnets), and Security-Group. The Neutron plugin translates the elements from one representation to another.

### Data Structure

Although the actual data between Neutron and Contrail is similar, the listings of the elements differs significantly. In the Contrail API, the networking elements list is a summary, containing only the UUID, FQ name, and an href, however, in Neutron, all details of each resource are included in the list.

The Neutron plugin has an inefficient list retrieval operation, especially at scale, because it:

- reads a list of resources (for example. GET /virtual-networks), then



- iterates and reads in the details of the resource (GET /virtual-network/<uuid>).

As a result, the API server spends most of the time in this type of GET operation just waiting for results from the Cassandra database.

The following features in Contrail improve performance with Neutron:

- An optional detail query parameter is added in the GET of collections so that the API server returns details of all the resources in the list, instead of just a summary. This is accompanied by changes in the Contrail API library so that a caller gets returned a list of the objects.
- The existing Contrail list API takes in an optional parent\_id query parameter to return information about the resource anchored by the parent.
- The Contrail API server reads objects from Cassandra in a multiget format into obj\_uuid\_cf, where object contents are stored, instead of reading in an xget/get format. This reduces the number of round-trips to and from the Cassandra database.

## Network Sharing in Neutron

Using Neutron, a deployer can make a network accessible to other tenants or projects by using one of two attributes on a network:

- set the shared attribute to allow sharing
- set the router:external attribute, when the plugin supports an external\_net extension

### *Using the Shared Attribute*

When a network has the shared attribute set, users in other tenants or projects, including non-admin users, can access that network, using:

```
neutron net-list --shared
```

Users can also launch a virtual machine directly on that network, using:

```
nova boot <other-parameters> -nic net-id=<shared-net-id>
```

### *Using the Router:External Attribute*

When a network has the router:external attribute set, users in other tenants or projects, including non-admin users, can use that network for allocating floating IPs, using:

```
neutron floatingip-create <router-external-net-id>
```

then associating the IP address pool with their instances.

**NOTE:** The VN hosting the FIP pool should be marked shared and external.

## Commands for Neutron Network Sharing

The following table summarizes the most common Neutron commands used with Contrail.

| Action  | Command   |
|---|---|
| List all shared networks.                                 | <code>neutron net-list --shared</code>                            |
| Create a network that has the shared attribute.           | <code>neutron net-create &lt;net-name&gt; -shared</code>          |
| Set the shared attribute on an existing network.          | <code>neutron net-update &lt;net-name&gt; -shared</code>          |
| List all router:external networks.                        | <code>neutron net-list --router:external</code>                   |
| Create a network that has the router:external attribute.  | <code>neutron net-create &lt;net-name&gt; -router:external</code> |
| Set the router:external attribute on an existing network. | <code>neutron net-update &lt;net-name&gt; -router:external</code> |

## Support for Neutron APIs

The OpenStack Neutron project provides virtual networking services among devices that are managed by the OpenStack compute service. Software developers create applications by using the OpenStack Networking API v2.0 (Neutron).

Contrail provides the following features to increase support for OpenStack Neutron:

- Create a port independently of a virtual machine.
- Support for more than one subnet on a virtual network.
- Support for allocation pools on a subnet.
- Per tenant quotas.
- Enabling DHCP on a subnet.
- External router can be used for floating IPs.

For more information about using OpenStack Networking API v2.0 (Neutron), refer to: <http://docs.openstack.org/api/openstack-network/2.0/content/> and the OpenStack Neutron Wiki at: <http://wiki.openstack.org/wiki/Neutron> .

## Contrail Neutron Plugin

The Contrail Neutron plugin provides an implementation for the following core resources:

- Network
- Subnet
- Port

It also implements the following standard and upstreamed Neutron extensions:

- Security group
- Router IP and floating IP
- Per-tenant quota
- Allowed address pair

The following Contrail-specific extensions are implemented:

- Network IPAM
- Network policy
- VPC table and route table
- Floating IP pools

The plugin does not implement native bulk, pagination, or sort operations and relies on emulation provided by the Neutron common code.

## DHCP Options

In Neutron commands, DHCP options can be configured using `extra-dhcp-options` in `port-create`.

### Example

```
neutron port-create net1 --extra-dhcp-opt opt_name=<dhcp_option_name>,opt_value=<value>
```

The `opt_name` and `opt_value` pairs that can be used are maintained in GitHub: <https://github.com/Juniper/contrail-controller/wiki/Extra-DHCP-Options> .

## Incompatibilities

In the Contrail architecture, the following are known incompatibilities with the Neutron API.

- Filtering based on any arbitrary key in the resource is not supported. The only supported filtering is by `id`, `name`, and `tenant_id`.
- To use a floating IP, it is not necessary to connect the public subnet and the private subnet to a Neutron router. Marking a public network with `router:external` is sufficient for a floating IP to be created and associated, and packet forwarding to it will work.
- The default values for quotas are sourced from `/etc/contrail/contrail-api.conf` and not from `/etc/neutron/neutron.conf`.

## Support for Amazon VPC APIs on Contrail OpenStack

### IN THIS SECTION

- [Overview of Amazon Virtual Private Cloud | 1093](#)
- [Mapping Amazon VPC Features to OpenStack Contrail Features | 1093](#)
- [VPC and Subnets Example | 1094](#)
- [Euca2ools CLI for VPC and Subnets | 1095](#)
- [Security in VPC: Network ACLs Example | 1095](#)
- [Euca2ools CLI for Network ACLs | 1097](#)
- [Security in VPC: Security Groups Example | 1097](#)
- [Euca2ools CLI for Security Groups | 1098](#)
- [Elastic IPs in VPC | 1099](#)
- [Euca2ools CLI for Elastic IPs | 1099](#)
- [Euca2ools CLI for Route Tables | 1100](#)
- [Supported Next Hops | 1100](#)
- [Internet Gateway Next Hop Euca2ools CLI | 1101](#)
- [NAT Instance Next Hop Euca2ools CLI | 1101](#)
- [Example: Creating a NAT Instance with Euca2ools CLI | 1101](#)

## Overview of Amazon Virtual Private Cloud

The current Grizzly release of OpenStack supports Elastic Compute Cloud (EC2) API translation to OpenStack Nova, Quantum, and Keystone calls. EC2 APIs are used in Amazon Web Services (AWS) and virtual private clouds (VPCs) to launch virtual machines, assign IP addresses to virtual machines, and so on. A VPC provides a container where applications can be launched and resources can be accessed over the networking services provided by the VPC.

Contrail enhances its use of EC2 APIs to support the Amazon VPC APIs.

The Amazon VPC supports networking constructs such as: subnets, DHCP options, elastic IP addresses, network ACLs, security groups, and route tables. The Amazon VPC APIs are now supported on the Openstack Contrail distribution, so users of the Amazon EC2 APIs for their VPC can use the same scripts to move to an Openstack Contrail solution.

**Euca2ools** are command-line tools for interacting with Amazon Web Services (AWS) and other AWS-compatible web services, such as OpenStack. **Euca2ools** have been extended in OpenStack Contrail to add support for the Amazon VPC, similar to the support that already exists for the Amazon EC2 CLI.

For more information about Amazon VPC and AWS EC2, see:

- Amazon VPC documentation: [http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC\\_Introduction.html](http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Introduction.html)
- Amazon VPC API list: <http://docs.aws.amazon.com/AWSEC2/latest/APIReference/query-apis.html>

## Mapping Amazon VPC Features to OpenStack Contrail Features

The following table compares Amazon VPC features to their equivalent features in OpenStack Contrail.

**Table 96: Amazon VPC and OpenStack Contrail Feature Comparison**

| Amazon VPC Feature | OpenStack Contrail Feature  |
|--------------------|-----------------------------|
| VPC                | Project                     |
| Subnets            | Networks (Virtual Networks) |
| DHCP options       | IPAM                        |
| Elastic IP         | Floating IP                 |
| Network ACLs       | Network ACLs                |

**Table 96: Amazon VPC and OpenStack Contrail Feature Comparison (Continued)**

| Amazon VPC Feature | OpenStack Contrail Feature |
|--------------------|----------------------------|
| Security Groups    | Security Groups            |
| Route Table        | Route Table                |

## VPC and Subnets Example

When creating a new VPC, the user must provide a classless inter-domain routing (CIDR) block of which all subnets in this VPC will be part.

In the following example, a VPC is created with a CIDR block of 10.1.0.0/16. A subnet is created within the VPC CIDR block, with a CIDR block of 10.1.1.0/24. The VPC has a default network ACL named `acl-default`.

All subnets created in the VPC are automatically associated to the default network ACL. This association can be changed when a new network ACL is created. The last command in the list below creates a virtual machine using the image `ami-00000003` and launches with an interface in `subnet-5eb34ed2`.

```
# euca-create-vpc 10.1.0.0/16
VPC VPC:vpc-8352aa59 created

# euca-describe-vpcs
VpcId          CidrBlock      DhcpOptions
-----          -
vpc-8352aa59   10.1.0.0/16    None

# euca-create-subnet -c 10.1.1.0/24 vpc-8352aa59
Subnet: subnet-5eb34ed2 created

# euca-describe-subnets
Subnet-id      Vpc-id         CidrBlock
-----          -
subnet-5eb34ed2 vpc-8352aa59   10.1.1.0/24

# euca-describe-network-acls
AclId
-----
acl-default(def)
```

```
vpc-8352aa59
Rule      Dir      Action  Proto  Port  Range  Cidr
----      -
100       ingress allow   -1     0     65535 0.0.0.0/0
100       egress  allow   -1     0     65535 0.0.0.0/0
32767     ingress deny    -1     0     65535 0.0.0.0/0
32767     egress  deny    -1     0     65535 0.0.0.0/0

Association      SubnetId      AclId
-----
aclassoc-0c549d66  subnet-5eb34ed2  acl-default

# euca-run-instances -s subnet-5eb34ed2 ami-00000003
```

## Euca2ools CLI for VPC and Subnets

The following euca2ools CLI commands are used to create, define, and delete VPCs and subnets:

- euca-create-vpc
- euca-delete-vpc
- euca-describe-vpcs
- euca-create-subnet
- euca-delete-subnet
- euca-describe-subnets

## Security in VPC: Network ACLs Example

Network ACLs support ingress and egress rules for traffic classification and filtering. The network ACLs are applied at a subnet level.

In the following example, a new ACL, acl-ba7158, is created and an existing subnet is associated to the new ACL.

```
# euca-create-network-acl vpc-8352aa59
acl-ba7158c

# euca-describe-network-acls
AclId
```

```

-----
acl-default(def)
vpc-8352aa59

      Rule   Dir     Action  Proto  Port  Range  Cidr
      ----   ---     -
      100    ingress allow   -1     0    65535  0.0.0.0/0
      100    egress  allow   -1     0    65535  0.0.0.0/0
      32767  ingress deny    -1     0    65535  0.0.0.0/0
      32767  egress  deny    -1     0    65535  0.0.0.0/0

      Association      SubnetId      AclId
      -----
      aclassoc-0c549d66  subnet-5eb34ed2  acl-default

AclId
-----
acl-ba7158c
vpc-8352aa59

      Rule   Dir     Action  Proto  Port  Range  Cidr
      ----   ---     -
      32767  ingress deny    -1     0    65535  0.0.0.0/0
      32767  egress  deny    -1     0    65535  0.0.0.0/0

# euca-replace-network-acl-association -a aclassoc-0c549d66 acl-ba7158c
aclassoc-0c549d66

# euca-describe-network-acls
AclId
-----
acl-default(def)
vpc-8352aa59

      Rule   Dir     Action  Proto  Port  Range  Cidr
      ----   ---     -
      100    ingress allow   -1     0    65535  0.0.0.0/0
      100    egress  allow   -1     0    65535  0.0.0.0/0
      32767  ingress deny    -1     0    65535  0.0.0.0/0
      32767  egress  deny    -1     0    65535  0.0.0.0/0

      Association      SubnetId      AclId

```



```

-----
AclId
-----
acl-ba7158c
vpc-8352aa59

Rule   Dir   Action  Proto  Port  Range  Cidr
-----
32767  ingress deny    -1     0    65535  0.0.0.0/0
32767  egress  deny    -1     0    65535  0.0.0.0/0

Association      SubnetId      AclId
-----
aclassoc-0c549d66  subnet-5eb34ed2  acl-ba7158c

```

## Euca2ools CLI for Network ACLs

The following euca2ools CLI commands are used to create, define, and delete VPCs and subnets:

- euca-create-network-acl
- euca-delete-network-acl
- euca-replace-network-acl-association
- euca-describe-network-acls
- euca-create-network-acl-entry
- euca-delete-network-acl-entry
- euca-replace-network-acl-entry

## Security in VPC: Security Groups Example

Security groups provide virtual machine level ingress/egress controls. Security groups are applied to virtual machine interfaces.

In the following example, a new security group is created. The rules can be added or removed for the security group based on the commands listed for euca2ools. The last line launches a virtual machine using the newly created security group.

```
# euca-describe-security-groups
```

```

GroupId      VpcId      Name      Description
-----      -
sg-6d89d7e2  vpc-8352aa59  default

                Direction  Proto  Start  End  Remote
                -
                Ingress   any    0      65535 [0.0.0.0/0]
                Egress   any    0      65535 [0.0.0.0/0]

# euca-create-security-group -d "TestGroup" -v vpc-8352aa59 testgroup
GROUP  sg-c5b9d22a  testgroup  TestGroup

# euca-describe-security-groups

GroupId      VpcId      Name      Description
-----      -
sg-6d89d7e2  vpc-8352aa59  default

                Direction  Proto  Start  End  Remote
                -
                Ingress   any    0      65535 [0.0.0.0/0]
                Egress   any    0      65535 [0.0.0.0/0]

GroupId      VpcId      Name      Description
-----      -
sg-c5b9d22a  vpc-8352aa59  testgroup  TestGroup

                Direction  Proto  Start  End  Remote
                -
                Egress   any    0      65535 [0.0.0.0/0]

# euca-run-instances -s subnet-5eb34ed2 -g testgroup ami-00000003

```

## Euca2ools CLI for Security Groups

The following euca2ools CLI commands are used to create, define, and delete security groups:

- euca-create-security-group

- euca-delete-security-group
- euca-describe-security-groups
- euca-authorize-security-group-egress
- euca-authorize-security-group-ingress
- euca-revoke-security-group-egress
- euca-revoke-security-group-ingress

## Elastic IPs in VPC

Elastic IPs in VPCs are equivalent to the floating IPs in the Contrail Openstack solution.

In the following example, a floating IP is requested from the system and assigned to a particular virtual machine. The prerequisite is that the provider or Contrail administrator has provisioned a network named “public” and allocated a floating IP pool to it. This “public” floating IP pool is then internally used by the tenants to request public IP addresses that they can use and attach to virtual machines.

```
# euca-allocate-address --domain vpc
ADDRESS 10.84.14.253    eipalloc-78d9a8c9

# euca-describe-addresses --filter domain=vpc
Address      Domain    AllocationId      InstanceId(AssociationId)
-----
10.84.14.253    vpc      eipalloc-78d9a8c9

# euca-associate-address -a eipalloc-78d9a8c9 i-00000008
ADDRESS eipassoc-78d9a8c9

# euca-describe-addresses --filter domain=vpc
Address      Domain    AllocationId      InstanceId(AssociationId)
-----
10.84.14.253    vpc      eipalloc-78d9a8c9    i-00000008(eipassoc-78d9a8c9)
```

## Euca2ools CLI for Elastic IPs

The following euca2ools CLI commands are used to create, define, and delete elastic IPs:

- euca-allocate-address
- euca-release-address

- euca-describe-addresses
- euca-associate-address
- euca-disassociate-address

## Euca2ools CLI for Route Tables

Route tables can be created in an Amazon VPC and associated with subnets. Traffic exiting a subnet is then looked up in the route table and, based on the route lookup result, the next hop is chosen.

The following euca2ools CLI commands are used to create, define, and delete route tables:

- euca-create-route-table
- euca-delete-route-table
- euca-describe-route-tables
- euca-associate-route-table
- euca-disassociate-route-table
- euca-replace-route-table-association
- euca-create-route
- euca-delete-route
- euca-replace-route

## Supported Next Hops

The supported next hops for the current release are:

- Local Next Hop

Designating local next hop indicates that all subnets in the VPC are reachable for the destination prefix.

- Internet Gateway Next Hop

This next hop is used for traffic destined to the Internet. All virtual machines using the Internet gateway next hop are required to use an Elastic IP to reach the Internet, because the subnet IPs are private IPs.

- NAT instance

To create this next hop, the user needs to launch a virtual machine that provides network address translation (NAT) service. The virtual machine has two interfaces: one internal and one external, both

of which are automatically created. The only requirement here is that a “public” network should have been provisioned by the admin, because the second interface of the virtual machine is created in the “public” network.

## Internet Gateway Next Hop Euca2ools CLI

The following euca2ools CLI commands are used to create, define, and delete Internet gateway next hop:

- euca-attach-internet-gateway
- euca-create-internet-gateway
- euca-delete-internet-gateway
- euca-describe-internet-gateways
- euca-detach-internet-gateway

## NAT Instance Next Hop Euca2ools CLI

The following euca2ools CLI commands are used to create, define, and delete NAT instance next hops:

- euca-run-instances
- euca-terminate-instances

## Example: Creating a NAT Instance with Euca2ools CLI

The following example creates a NAT instance and creates a default route pointing to the NAT instance.

```
# euca-describe-route-tables
RouteTableId  Main  VpcId          AssociationId  SubnetId
-----
rtb-default  yes   vpc-8352aa59  rtbassoc-0c549d66  subnet-5eb34ed2

                Prefix          NextHop
                -----
                10.1.0.0/16      local

# euca-describe-images
IMAGE  ami-00000003  None (ubuntu)      2c88a895fdea4461a81e9b2c35542130
IMAGE  ami-00000005  None (nat-service) 2c88a895fdea4461a81e9b2c35542130

# euca-run-instances ami-00000005

# euca-create-route --cidr 0.0.0.0/0 -i i-00000006 rtb-default
```

```
# euca-describe-route-tables
```

```
RouteTableId  Main  VpcId          AssociationId  SubnetId
-----      ----  -
rtb-default   yes   vpc-8352aa59  rtbassoc-0c549d66  subnet-5eb34ed2
```

```
Prefix        NextHop
-----      -
10.1.0.0/16   local
0.0.0.0/0     i-00000006
```